A DIGITAL SIGNAL PROCESSOR FOR LASER DOPPLER ANEMOMETER SYSTEMS

By

LLOYD NEAL SALSMAN

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1972

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1973

OKLAHOMA
STATE UNIVERSITY
LIBRARY

APR 10 1974

A DIGITAL SIGNAL PROCESSOR FOR LASER

DOPPLER ANEMOMETER SYSTEMS

Thesis Approved:

Paul P. McCollum
Thesis Adviser

Edward L. Shreve

W. G. Tiederman, Jr.

[illegible]
Dean of the Graduate College

877293

PREFACE

This study was undertaken at the request of the staff involved with making fluid measurements at the Basic Fluid Dynamics Laboratory. The objective was to automate the gathering of velocity measurements taken with the Laser Doppler Anemometer (LDA). Prior to the construction of the signal processor, all measurements were made by visual scrutiny of the anemometer waveforms. It is with this in mind that I acknowledge and appreciate the time and effort expended by Dr. W. G. Tiederman, Dr. D. K. McLaughlin, and Dr. Michael Reischman for their pioneer efforts in this area.

Construction costs of the processor were borne by the School of Mechanical and Aerospace Engineering and the School of Electrical Engineering at OSU. Facilities for the construction of the processor were made available to me at the Mechanical and Aerospace Engineering Labs.

I would like to thank Tom Goins at Texas Instruments Supply Company for his aid in procuring hardware for me; William Adcox, Michael Karpuk, and Maurice Colpitts for their assistance in constructing the processor. Professor P. A. McCollum, my adviser, who should take a large part of the blame for my interest in computers, provided the counsel and assistance I needed to complete the project.

I also wish to thank my grandmother, Mrs. Herman Dittmer who typed this manuscript in its entirety.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1. Investigation of Laser Doppler Anemometer Signal Processing

1.1 The Problem of Signal Processing

The Laser Doppler Anemometer system, which will be described, produces a burst of pulses. These pulses indicate the velocity of particles in the fluid. Unfortunately, a simple examination of these pulses is not possible. A variety of factors contrive to suppress or alter the pulses. This alteration of the pulse train renders velocity measurement difficult. Since the acquisition rate is usually high, rejection of altered pulses would help correct the measurement problem. A method for detecting and rejecting altered pulse trains was the object of this investigation. Using this method, a device for processing these pulse trains was designed and built.

The first part of this work involves the method for signal processing. The last part describes the design and construction of the signal processor.

2. Laser Doppler Anemometer

2.1 Basic Laser Anemometer Configuration

A laser doppler anemometer (LDA) is a device which measures particle velocities in a fluid using scattered laser light (see Asher (1)). The particle velocities, that would be measured using the LDA, may be used to measure some characteristics of the host fluid flow. Several advantages may be realized using the LDA over conventional fluid velocity measurement apparatus. As with every measurement scheme some complications arise. One solution to a complication of the LDA is the topic of this thesis.

2.2 Continuous Wave

One configuration of LDA uses a high concentration of particles such that light is almost continuously scattered. This light is processed by the LDA and, as one might expect, a continuous doppler signal is the result. This condition is necessary for the operation of the "TRACKER" doppler signal processor.

2.3 Individual Realization

Individual realization LDA use a low particle concentration such that only one particle may be scattering light at any particular time. The configuration is the same as the continuous wave system except for the particle density.

CHAPTER II

DOPPLER SIGNAL PROCESSORS FOR THE LDA

1. Major Difficulties

1.1 Doppler Dropout

One of the difficulties of both the individual realization and the continuous system is "doppler dropout". This term applies to a condition when a pulse is interfered with and one or more of the cycles has been cancelled out. (see Figure 1). Although this definition lends itself best to individual realization it can also be applied to continuous wave (continuous wave systems also include the loss of all signal as a doppler dropout).

1.2 Frequency Variations of Doppler Bursts

Turbulent fluid flows are commonly measured using the LDA and, since wide fluctuations in fluid velocity are expected, one might also expect the doppler frequency to vary also. This frequency variation is another difficulty in processing the LDA signal. Continuous wave LDA processers must be able to follow these variations with high acquisition speed and a minimum frequency overshoot.

2. Two Methods of Doppler Signal Processing

2.1 The "TRACKER" Method

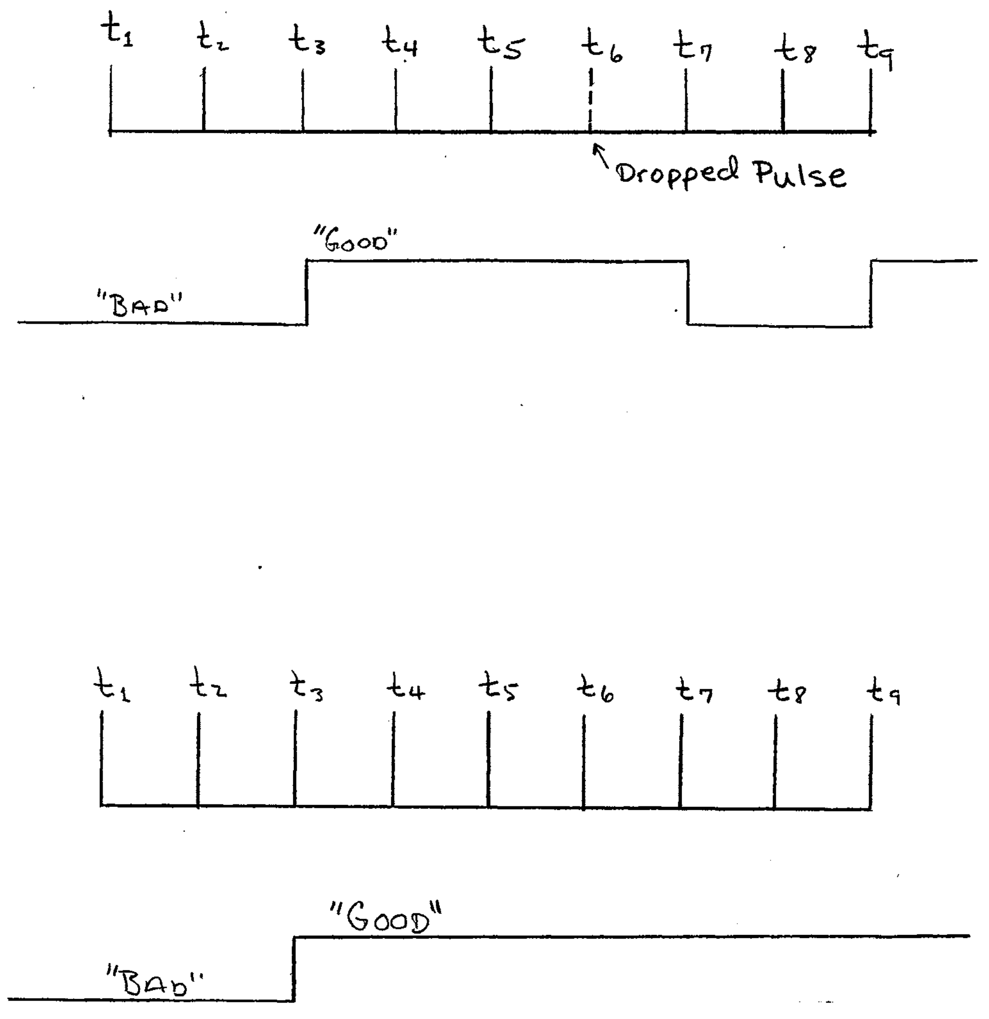
This method lends itself well to continuous wave LDA since it requires a continuous doppler signal (see Smid (10)). Two variations in the hardware exist, one uses a "frequency locked loop" the other, a "phase locked loop". The frequency locked loop has the advantage of being able to use heterodyning to lower the doppler signal and use this frequency directly in the loop. The phase locked loop requires additional hardware but uses much the same heterodyne technique.

The limiting factor of both systems seems to be their ability to track turbulent fluid flows. At least one investigation is being conducted to measure this characteristic.

This method is an application of analog electronics in a system that has found various application. This application seems to fall short because the reference oscillator in the loop cannot "slew" or change frequency fast enough to acquire and hold a turbulent flow's doppler signal. A loss of signal proves catastrophic to this scheme.

2.2 The "COUNTER" Method

This method lends itself well to individual realization LDA. Several different schemes have been used here



t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9
Dropped Pulse
"GOOD"
"BAD"
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9
"GOOD"
"BAD"

Figure 1. Doppler Signals with Dropout

but they all function similarly. A count of some reference oscillator is started when the leading edge of the doppler signal is detected (see Asher(1)). This count is carried on for a number of different cycle times (i.e., one counter may count 5 cycles, another 3). The various counters are then compared to determine if any cycles were lost by any counter. At least two methods of effecting this comparison are documented (see Asher (1)). If a mismatch occurs, then the period measurement is discarded.

This is the category of processors that has been used as the basis for the proposed doppler signal processor.

CHAPTER III

A SEQUENTIAL PHASE COMPARISON METHOD

1. Detecting Doppler Dropout

1.1 Sequential Phase Comparison

If the doppler signal was continuously monitored and a period count made for each pulse as it arrives, a method for the comparison of two previous periods might be used to locate a doppler dropout. As each low to high transition of the doppler signal occurs, a count of the period would be completed and a new period count started. As the new period count is taken, the old period count would be saved and compared to the period count made just before. If a match occurs, then the new count will replace the oldest period count and a comparison will be made with the old count. If this process is repeated continuously then any missing cycle will be detected shortly after the last period is accumulated (i.e., when the next pulse arrives). Three good pulses (or two cycles) are required to establish a "good" pulse train. The absence of pulses will be a special condition which is treated as a "bad" pulse train.

This sliding comparison allows the use of smaller capacity period counters and a coarser period or reference oscillator. When a dropout occurs, the difference in period counts should be large (at least 50%), and easily distingishable.

1.2 Sequential Comparison Under Tolerance

Occasionally a pulse may be distorted in phase and appear outside the expected area. That is, the pulse is present but occurs earlier or later than previous pulses indicate. To reduce this problem a comparison can be made using a fixed tolerance value. The tolerance value is applied during the comparison and allows the period to be above or below the expected value by a fixed percent. Any period count not within the bounds would cause the measurement to be discarded. The actual deviation measured this way might be used to indicate particle skewing within the LDA probe.

2. Associated Measurement Equipment

2.1 Doppler Signal Source

This is the output of the LDA system as shown in Figure (3). Before the comparator can analyze this signal it must be processed by a Schmitt trigger. After the doppler has been converted in a pulse train, it is ready to be scrutinized by the comparator.

2.2 Sequential Phase Comparator

The comparator gates the doppler signal to the period counter. If the comparator finds a doppler dropout, a high frequency burst is used to dump the counter and the counter is then ready to receive a new pulse train from the comparator.

2.3 Period Counter

This counter is external to the comparator. The counter outputs the period or average period of the doppler burst. This instrument may be more precise than the period counters used for the comparison. Output from this counter will be recorded as the doppler period from which the velocities will be calculated.

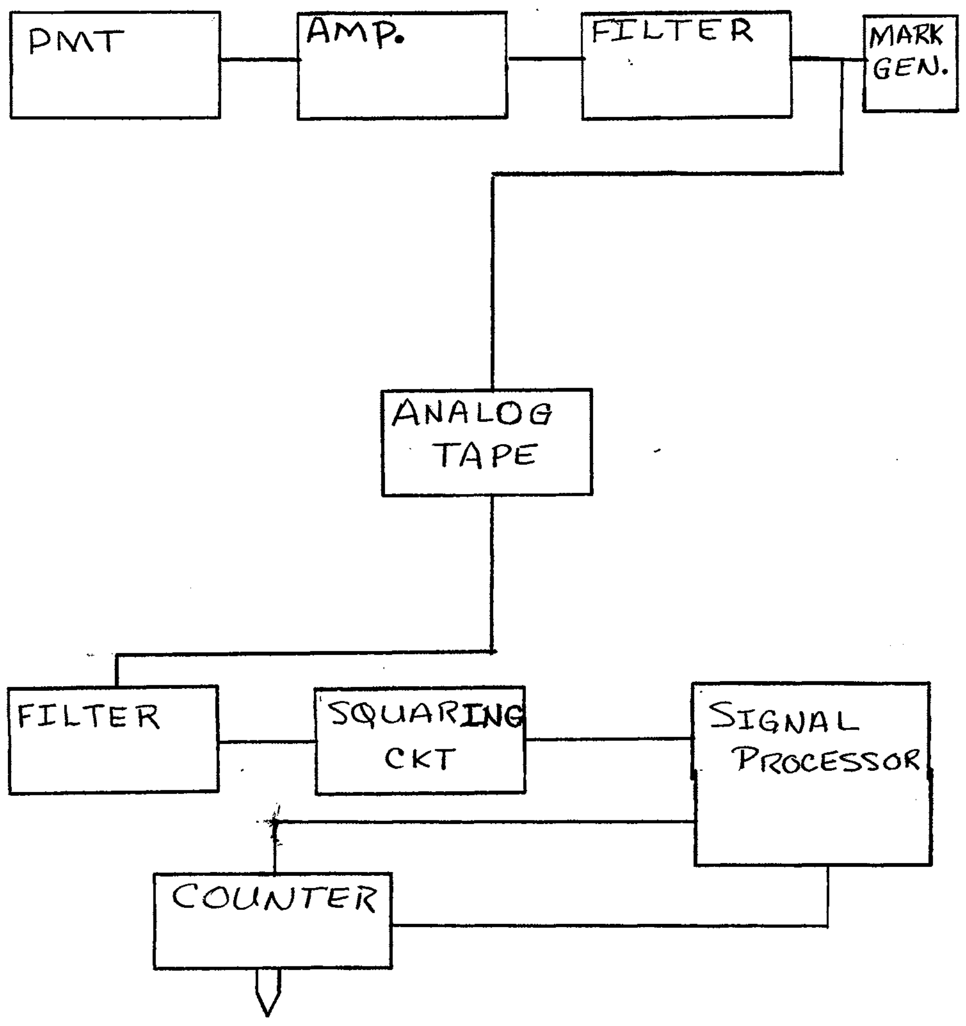


Figure 2. LDA Data Acquisition System

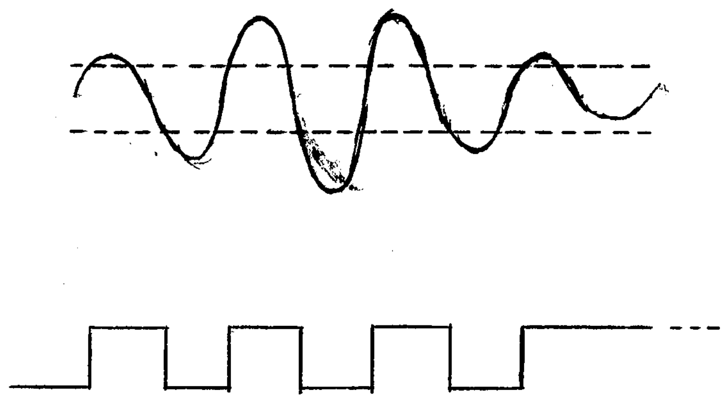


Figure 3. Doppler Burst and Schmitt Trigger

CHAPTER IV

IMPLEMENTING THE SEQUENTIAL PHASE COMPARISON

1. Doppler Period Count

1.1 Period Oscillator and Counter Chain

The Doppler period count is made by one of two binary counter chains. These counters have a common frequency standard, which is used as the "clock". This clock, referenced as the "period oscillator" (PO), is an external input that may be adjusted to provide optimum resolution for a given frequency range. The period oscillator controls the low frequency cutoff point and affects the resolution at high frequency.

As each pulse arrives, the current counter is stopped, the alternate counter is cleared, and the period oscillator is supplied to the alternate counter; the alternate counter is now designated the current counter. This process is repeated for each pulse.

1.2 Counter Overflow and Notification Section

If a period count exceeds the counter capacity, an error condition exists which must be reported to the comparator. The first overflow condition will notify the comparator of the error; all consecutive overflows will be ignored. As each period count is made, the processor is notified of the need for a comparison. After each comparison, the processor becomes dormant.

2. Effecting the Comparison Under Tolerance

2.1 The Normalized Error Formula

To perform the comparison under tolerance, it is convenient to use the normalized error formula defined:

$$\frac{A-B}{A} = t \qquad A \geqslant B \text{ and } A > 0 \tag{1}$$

This formula is the basis for the comparison algorithm used by the processor.

2.2 The Comparison Algorithm

The comparison algorithm is the procedure which is routinely followed by the processor as it examines the period counts. The normalized error formula has been rearranged to be more amenable to hardware implementation.

Four cases are possible and must be dealt with by the processor:

CASE I CA or CB overflow

Action: Comparison ALWAYS FAILS,

CASE II $CA = CB$

Action: Comparison ALWAYS PASSES,

CASE III $CA > CB \quad 0<t\leqslant 1$

substituting in (2)

$$\frac{CA-CB}{CA} < t \quad 0<t\leqslant 1$$

rearranging (2): (2)

$$CA - (1/t)\ (CA-CB) > 0 \tag{3}$$

CASE IV $CA < CB$

substituting in (1):

$$\frac{CB - CA}{CB} < t \quad 0 < t \leqslant 1 \tag{4}$$

Rearranging (4)

$$CB - (1/t)\ (CB - CA) > 0 \tag{5}$$

or

$$CB + (1/t)\ (CA-CB) > 0 \tag{6}$$

Where:

CA = positive integer period counter valve ("A" counter)

CB = positive integer period counter valve ("B" counter)

t = fractional normalized error

Case I detects the overflow error condition and always fails the pulse train. Case II removes the t=0 condition or zero error since it would require a "capacity shift" to be presented if it was allowed to enter case III or IV. Case III illustrates the manipulation of the normalized error formula to arrive at a comparison condition formula (3). This arrangement, (3), may be thought of as weighting the numerator and comparing it directly with the denominator. Case IV illustrates the opposite of Case III. Relation (6) is presented to identify a common term:

$$\frac{1}{t}(CA - CB) \qquad (7)$$

which is used to advantage by the processor.

2.3 Numerical Example Using the Comparison Algorithm

The signifigance of the comparison method can be demonstrated by an example comparison

Suppose that "A" counter contained the period count, five, and that the "B" counter contained the period count, three.

Since CA is greater than CB, case III used:

substituting in (2):

$$\frac{5 - 3}{5} = \frac{2}{5}$$

It is required to determine if CA and CB are within 50% of each other. The value of "t" then is .5.

substituting in (3):

$$5 - (1/.5)\ (5 - 3) > 0$$

or

$$5 - 4 > 0$$

Since one is greater than zero, the value of CA and CB satisfy the comparison condition and meet the tolerance bound of 50% (i.e., 40% < 50%). However, had the tolerance value been 25%, then:

substituting in (3):

$$5 - (1/.25)\ (5 - 3) > 0$$

or

$$5 - 8 > 0$$

which is false and does not satisfy the comparison condition. This result implies that CA and CB are not within 25% of each other as, indeed, they are not (i.e. 25%<40%).

3. Control of the Measurement Apparatus

3.1 Verification of the Meaurement

An external period averaging counter can be used to record the pulse period by averaging several pulse periods. Once a measurement has been initiated, it must be verified for the entire averaging cycle. This is, all periods counted by the period counter must be verified by the processor before it can be recorded as a representative measurement.

The verification process consists of the requirements for the pulses applied to the period averaging counter. The first requirement is that no pulses are supplied to the counter until valid periods are detected; that is, the "GO" signal is coincident with the Doppler pulses. The second requirement is that all periods being averaged must be averaging cycle, the measurement must be discarded.

Some observations may be in order at this point, if it was not necessary to allow the averaging counter to accummlate an invalid count and if the counter could be rapidly reset, the maximum data acquisition rate could be improved.

CHAPTER V

DESIGN OF THE SEQUENTIAL PHASE COMPARATOR

I. Processor Architecture

1.1 Requirements of the Comparator

In the previous chapter, a method for performing a comparison under tolerance was outlined. It is now required to design a device which will perform this comparison. The convenience of handling the period measurements digitally is probably the best reason for choosing a digital framework for the design. Analog techniques could be used but they would require additional conversions which might be quite lengthly and not very accurate. Expected fluctuations in frequency (velocity) would make scaling difficult for an analog system.

The comparator must be able to react rapidly and accurately over a wide range of frequencies and should not require a long time to do its work. Since the period counts are maintained digitally, a direct utilization of these values would be preferred. The measurement instruments and the data recording equipment are all digitally controlled. Different measurement situations may require different comparison criteria and the comparator should be flexible enough to accept minor changes.

1.2 Advantages of a Digital System

The digital framework has several convenient features which tend to make an analog system undesirable. Standard digital components are now available in the speed range required to cover the specified frequency range. The digital comparison time is independent of the period values being examined (i.e., the comparison time is constant). An analog comparison might take quite some time depending upon conversion speeds and settling time). The digital system could operate on the numerical period counts directly without conversion; the analog system must convert. Storage of the period count values might also prove difficult with analog storage devices.

Of course, particular care has been exercised to produce a comparison method which is particularly amenable to digital implementation. An analog approach has not been excluded from consideration; however, a digital solution seems a more reasonable approach.

1.3 Functions of the Comparator

The comparator must respond to two significant inputs. It must perform a comparison operation for each pulse received and, for each overflow condition, must clear previous period counter values (automatically signaling a "bad" measurement). These inputs are presented to the comparator by a two level interrupt system.

The highest level of interruption is used for a hardware or firmware malfunction. Hardware is used to designate a circuitry malfunction of the host processor while firmware[1] is used to designate a program malfunction (reasons for this nomenclature will be stated later). This interrupt is permanent in nature and causes further processing to cease, pending some form of manual intervention.

The second level of interruption is used to request a comparison operation (or an overflow condition). This interrupt should only occur while the processor is dormant. If an interrupt occurs while the processor is active, a level one interrupt may be signaled (this feature is optional). This condition is termed an "overrun". An overrun implies that data pulses were arriving at a rate greater than the processor's comparison rate.

The second level interrupt may also signal the overflow condition. A flag must be tested to determine if an overflow has occurred. If so, special processing can be performed.

The result of the comparison condenses to the setting of a flag to indicate the current condition of the pulse periods. If the previous compare was good a "GO" flag is

[1]Firmware refers to the functions provided by the microprogram, (i.e., control signals generated by the microcode).

displayed; if the previous compare was bad an "ABORT" flag is displayed.

2. Microprocessor Controlled Computers

2.1 Introduction to Microprocessors

The concept of a microprocessor was first introduced by M. V. Wilkes. Several variations of his idea have been made and applied to many modern pieces of computing equipment. Probably the most notable use has been in "compatible architecture" computer families, (e g., the IBM System 360 series). The flexibility, the economy, and the serviceability of a microprocessor usually outweigh its disadvantages. Speed is the primary sacrifice when using a microprocessor.

A simple microprocessor might be thought of as the control and data manipulating unit of a computer. Using Figure 4 as a guide, a computer may be considered as a collection of various units. The input is used to acquire information to be processed or stored. The storage unit is used to retain information that has been or is to be processed. The output unit is used to disseminate information that has been processed. The arithmetic and logic unit (ALU) is used to perform data manipulation. The control unit is used to coordinate the functions of all other units in a logical manner to accomplish the current task. The bus is used to distribute information among all the units.

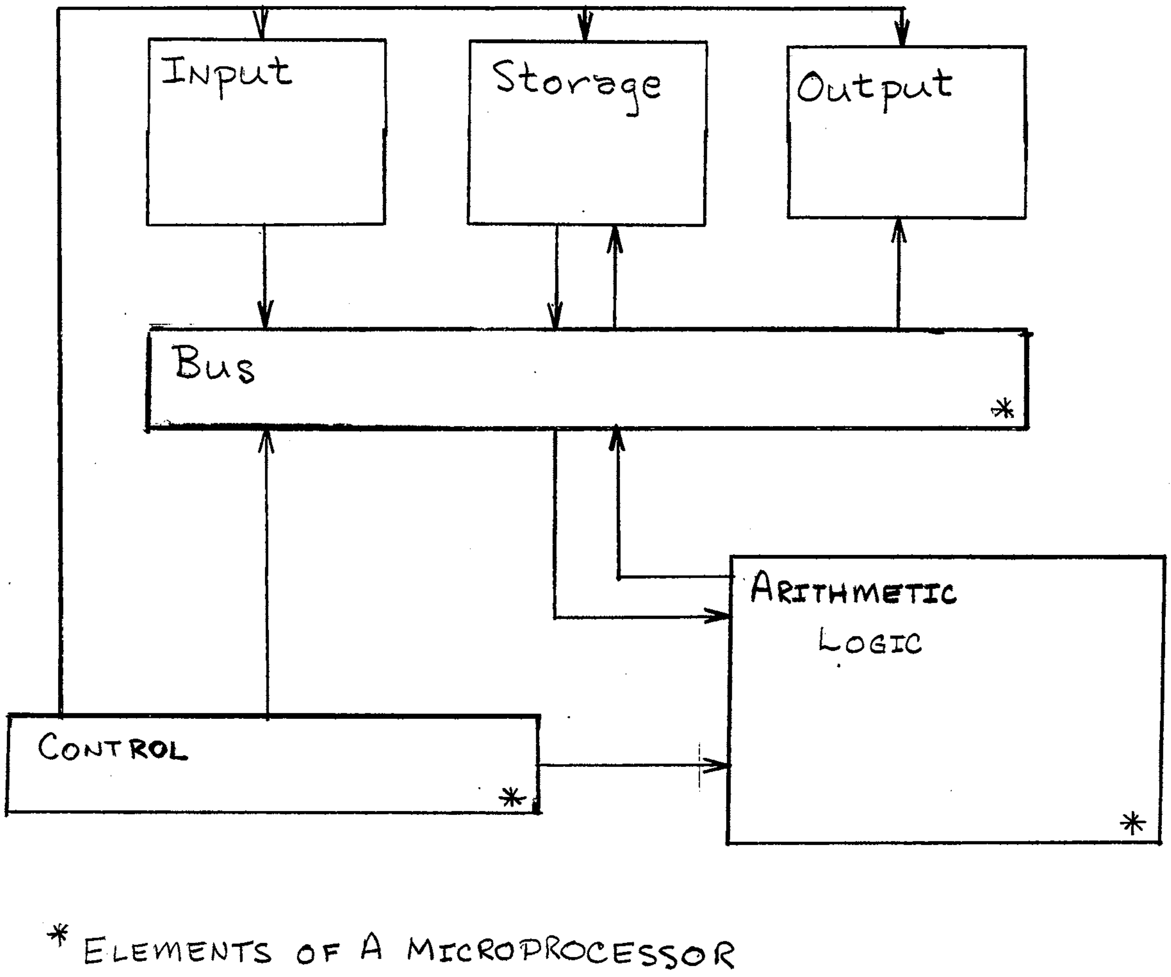


Figure 4. Block Diagram of A Computer

The indicated area is the microprocessor when the control and ALU can be separated and the control signals can be said to be generated by a "stored program". The "stored program" concept is probably the most important feature of modern computers. This "stored program" or microprogram can be used to tailor the functioning of its associate units. Alteration of the microprogram can cause a single processor to be adaptable to many different applications. An example of this flexibility is emulation of computer systems. Emulation of a computer system by microprogramming can be effected by causing the host computer to respond to the emulated computer's instruction set. The necessary changes to the host computer might require only a change in the microprogram.

2.2 Characteristics of a Microprocessor

The function of the control unit is the generation of control signals. All units respond to these control signals including the control unit. The content of the microinstruction word is dependent upon the hardware that is to be controlled. Often separate fields are assigned specific functions. All register controls may be from one field; all bus controls from another. Some microprocessors execute microinstructions sequentially and accomplish decisions by skipping instructions. Another technique includes the base address of the next instruction in a field within the microinstruction. The latter technique allows multiple alternatives at a cost of only one instruction. Accessing the microinstructions need not take place in a particular order and "patching" or inserting instructions in the execution sequence is relatively easy.

As the microprogram is executed, control signals are distributed to the units of the computer. Control signals are also transmitted to the control unit by the other units which may be used to alter the execution of the microprogram. These signals may be used to effect a "branch" within the microprogram or alter some function ordered by the microprogram.

If access is allowed to the microprogram, then modification of the computer's architecture would be relatively simple.

3. The Read Only Storage (ROS) Assembly

3.1 A Control Store Management Technique

The microprogram must be stored in some medium. Some media are field modifiable, others are not. The problem of maintaining a continuous control signal (used for functions spanning more than one instruction) is usually resolved by storing the instruction word while the next instruction is being fetched. As with most conventional memory systems, an address decoder is used to select a single instruction word from the memory. The memory organization for this particular system is two dimensional. As a word is selected by an address decoder, the bits which are stored in memory are transferred to the bit lines. All bit lines are "OR'ed" and are furnished as input to the bit buffer inverters. This arrangement implies that only one word at a time is selected. The bit buffer inverters are connected to bit latches. At the end of each fetch cycle, a sample pulse is applied to the bit latches, changing the output of the latches to reflect the contents of the new word.

This method allows continuous signals to be generated by including a bit in each word corresponding to the required signal. If each word contains the specified bit, a continuous signal will be produced. The bit latches furnish the control signals.

The addresses are generated by an address field combined with an address modifier or from manual entry switches. The basic components of the control store section are the bit latches (RL), the word drivers (RD), the diode matrix store (RM), and the sequencer (RS). This section of the microprocessor is independent of the others. This was planned so that different memory media could be used later. As soon as the microprogram was proven, a read only memory (ROM) could be cut to replace this temporary read only storage (ROS) unit. The ROM would contain the same bit structures (microinstructions) that were represented by diodes in the ROS unit. ROM bits are programmed by burning out fused links, transistors, or diodes. Once these adjustments have been made, it is not possible to alter the stored bits.

Since easy modification of the microprogram is required, a diode matrix seemed the best storage media. A diode matrix of convenient size was constructed for the microprocessor. The contents of the memory is changed by inserting or removing diodes along the word line. The most signifigant loss with this arrangement is speed. The physical size and the pulse response of the diode matrix cause an access time of less than 200 microseconds (measured) to be difficult (using a particular board material and diode). This loss was negligible when compared to the difficulties of other memory types.

3.2 Determining the Geometry of ROS

The "Geometry" of ROS refers to the number of control signals versus the number of instructions. The optimum arrangement would be to find the minimum area of a rectangle whose sides were proportional to the number of instructions and the number of control signals. Restrictions on the size of each parameter must also be considered. As the number of instructions increase, the microprogram execution time increases. As the number of control signals increase, the base cycle time increases as does the ALU's complexity (i.e., each instruction takes longer to completely execute). The optimum arrangement depends on the maximum allowable program execution time and the number of control signals required.

The geometry of the phase comparator ROS was fixed by the available hardware. This required that the microprogram be fitted into a memory 30 x 36 (i.e., 30 instructions of 36 control signals each). The control lines were selected and assigned along with bits for parity error detection from the 36 bits available. The microprogram was written in such a manner as to incorporate as many common logic paths as possible to conserve space.

The resulting ROS unit has a generous number of control signals available. This feature helps reduce the size of the microprogram (i.e., the original comparison microprogram's longest path was 13 instructions with a total storage requirement of 24 instructions).

3.3 Monophase or Polyphase Execution

Monophase execution (or vertical microprogramming) refers to a single simultaneous setting of all control signals as each microinstruction is fetched and executed. Polyphase execution (or horizontal microprogramming) refers to sequential execution of a single microinstruction over a definite time span. Execution of microinstructions in polyphase mode may be combinationally sychronized or synchronous.

Both modes have advantages but the implementation of the comparator probably does not require the complexity introduced by the polyphase mode. Normally the number of instructions required by the strict application of the monophase mode would be prohibitive; however, by introducing delays at several critical points within the ALU, the monophase mode works well and is simply constructed.

3.4 Parallel vs. Serial Fetching

The method used to determine the next ROS address to be accessed can be important. If it is possible to "pre-FETCH" the next microinstruction before it is needed, then considerable "overhead" time can be overlapped with the microinstruction execution time.

Parallel fetch implies that the next microinstruction address can be predicted (to some extent); a fetch is ordered for the particular address while the current microinstrùction is being executed. If the predicted address is incorrect (as in the case of a "branch"), then additional time must be allowed to fetch the count instruction.

Serial fetch implies that no "pre-FETCH" occurs and a "FETCH" cycle is required after each microinstruction is executed. Obviously, this is a simple minded approach and is wasteful of microprogram execution time.

The fetch technique used by the comparator microprocessor is a combination of both parallel and serial fetching. This technique was a parallel fetch during an early microinstruction fetch cycle. The microinstruction execution time is allowed to extend into this fetch time. If the microinstruction being executed is a branch, then the address modification (see "Branching") takes place early in the executive sequence and the correct microinstruction is fetched and executed. The longest computation time sets the instruction cycle time which includes the fetch cycle. An important feature to note is the ROS base address for the next instruction to be executed is included, as a field in each microinstruction word (ROSA).

3.5 Application of Encoded Fields

Most of the bits in a microinstruction specify a single control signal. A large number of signals, therefore, requires a large number of bits in the microinstruction word. Some signals may be mutually exclusive or logically exclusive. For example, if seven different signals were to be individually tested, a bit would be required for each signal using direct coding; but using an encoded field only three bits would be required.

An extension of this philosophy could be carried into "field modification"; that is, a control bit could change the interpretation of a specific field. This technique is used by the arithmetic/logic processor (ALP) and mode (M) fields.

Most fields in the microinstruction format are encoded. This encoding scheme was specified by the characteristics of the controlled logic within the ALU. The status and control trigger field (STAT) has 16 possible functions implemented with 4 bits.

Encoded fields must be decoded and the decoding operation takes time. This time is traded for a smaller

ROS INSTRUCTION WORD

FORMAT

BIT(S)	FIELD	DESCRIPTION
00,09,18,27	PAR	PARITY
01,02,	MPLX	MULTIPLEXER CONTROL
03-08,10-15	RF	REGISTER FILE CONTROL
16,17,19-22	ALP	ARITHMETIC/LOGIC CONTROL
23	CTL	CONTROL STATUS
24,25	SR	SHIFT REGISTER CONTROL
28-31	STAT	STATUS AND CONTROL TRIGGERS
26,32-35	ROSA	ROS ADDRESS

Figure 5. ROS Instruction Word Format

instruction word. With the physical limitation imposed by the 30 x 36 ROS, this tradeoff seems justified.

3.6 Branching

Each microinstruction has a field containing the ROS base address of the next microinstruction to be executed. This field allows random access to the ROS unit and makes insertion of microinstructions ("patching") easy. One reason for using the diode matrix was the ease of modification and repair.

The ROS base address field is also used to implement the "branch" function. This allows the execution sequence to be altered by a decision arrived at by the microprogram. To branch to another ROS instruction, an address must be placed in the ROS base address field. A condition which sets the address modifier (AM) flag must be tested. If the condition is present, then the next odd ROS address is fetched and executed. If the condition is absent, then the even ROS address is used. If an odd ROS base address is the condition testing instruction, a branch to this address is assumed.

3.7 Interruption Handling

Interruptions are handled by the ROS sequencer (RS). First level interruptions are associated with processor errors and require manual intervention to reset. These errors include data overrun, ROS parity, and a miscellaneous control error. Second level interruptions are expected while the microprocessor is dormant. These interruptions include the counter service request (a comparison or spill check is required) and the shift complete signal. Both conditions are the result of previous microinstruction execution and cause no errors.

The second level interrupts cause the "run ring latch" to be closed and execution to proceed until a microprogram request is received for asynchronous service or a first level interrupt occurs.

3.8 Manual Intervention

Manual control of the "run ring latch", the source of the ROS address, the ROS address, the ROS oscillator (ROSCO), and the tolerance value factor is allowed. The first four items are interpreted by the ROS sequencer (RS). They allow, respectively, the ability to immediately halt microprogram execution, to specify that the ROS address in the ROS address keys is to be used, the ROS address, the fetch/work cycle clock, and the multiplication factor for the ALU (interpreted by ALU input control (AI)). No other communication by the operator with the microprocessor is allowed.

4. The Arithmetic/Logic Unit (ALU)

4.1 Function and Characteristics of the ALU

The ALU performs the manipulation of data under control of the microprogram. The ALU communicates with the

ROS sequencer (RS) to perform branching and other functions. All data paths and processors and under direct microinstruction boundaries; this is a feature of the ROS control store just described. The ALU was assembled as four 4-bit processors logically connected as a single 16-bit processor. All four processors (AL) are interchangeable.

Of the thirty-six control lines all but nine are used to control the ALU. Most of these control lines are encoded. Decoding is provided on each card to minimize card interconnections. Two "bookend" cards process signals entering and leaving the processor: the ALU input control (AI) and the ALU output control (AO). The four remaining cards are the ALU processors (AL). Each AL card contains: two 4-bit counters, two 8 to 4-bit multiplexers, two 4x4-bit register files, a 4-bit ALU processor, and a 4-bit shift register (see Figures (6,7,8)).

4.2 The Counters and Counter Control (C)

Two 16-bit counters are supplied to accumulate the period count. These counters are alternately assigned as new data pulses arrive. The currently assigned counter is indicated by a flag (CC). The CC flag is used with the STAT function, LOAD, to store the idle counter in the appropriate register file (RF). If either counter overflows, a flag (SPILL) is set. This flag may be saved and interrogated by the microprogram. Once an overflow is signaled, consecutive overflows are ignored. The reference frequency, PO, applied to the counters is an external signal input.

4.3 The Multiplexer (MPLX)

The multiplexers (MPLX) are used to determine the data source for the register files (RF). Two sources are possible: the counter (C) or the shift register (SR). Two 32 to 16 bit multiplexers are supplied, one for each side of the ALP. These multiplexers are controlled by two common control lines. These lines allow selection of the data source and disabling the entire multiplexer.

4.4 The Register Files (RF)

The register files (RF) are used for local data storage. The counter values or shift register output may be stored here. The register files output only to the ALP. Two 4x16-bit register files are provided, one for each side of the ALP. Twelve control lines are required to control the register files. Each side of the register file requires two READ ADDRESS bits, two WRITE ADDRESS bits, one READ ENABLE, and one WRITE ENABLE. Simultaneous access is allowed to the register file ports. The STAT function, LOAD, modifies the WRITE ENABLE signal to the appropiate side of the register file to store the idle counter.

4.5 The Arithmetic/Logic Processor (ALP)

The ALP is used to process data. Sixteen arithmetic operations and sixteen logical operations are possible. Two sixteen bit words may be applied to the ALP, one from each

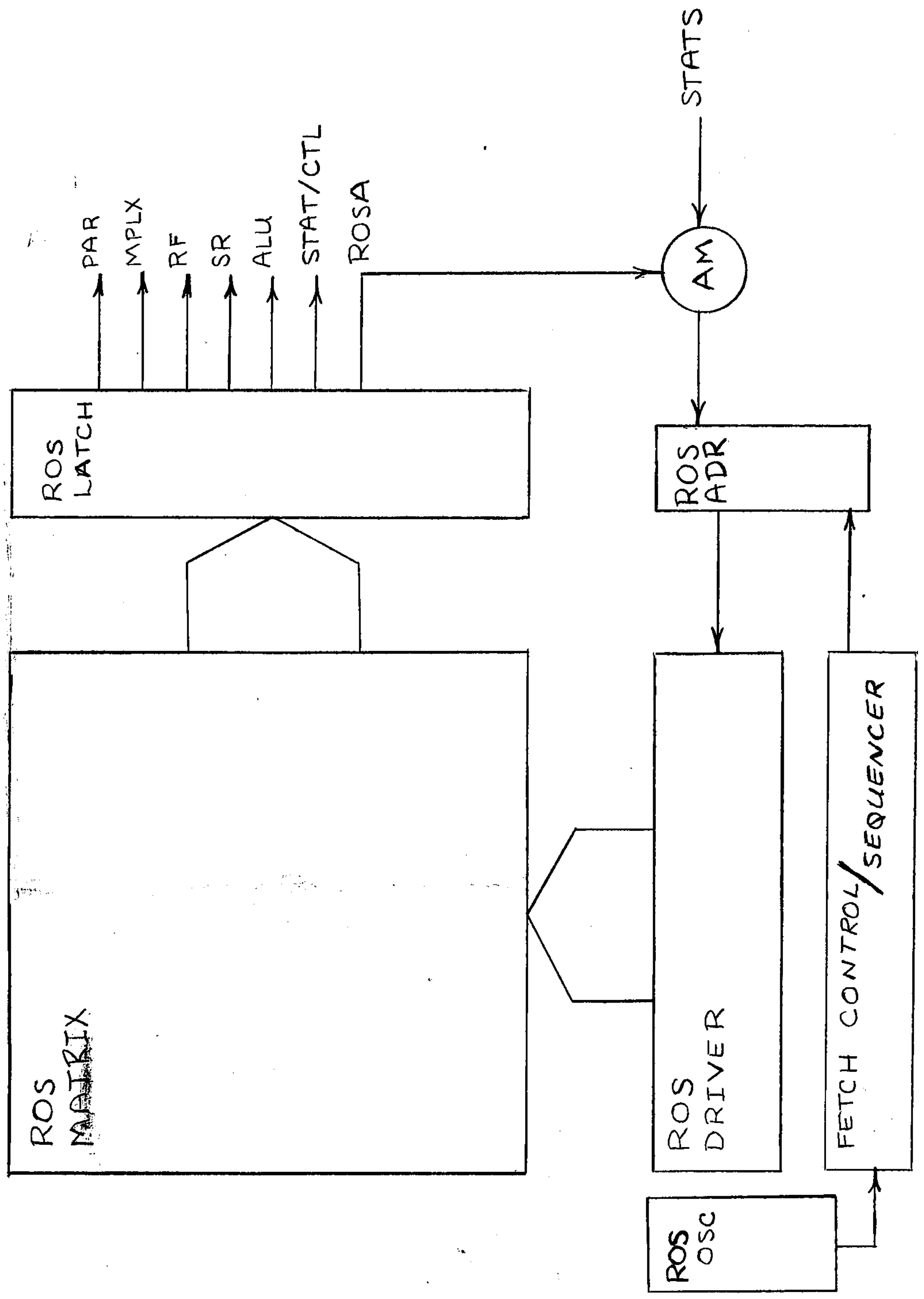


Figure 6. ROS Block Diagram

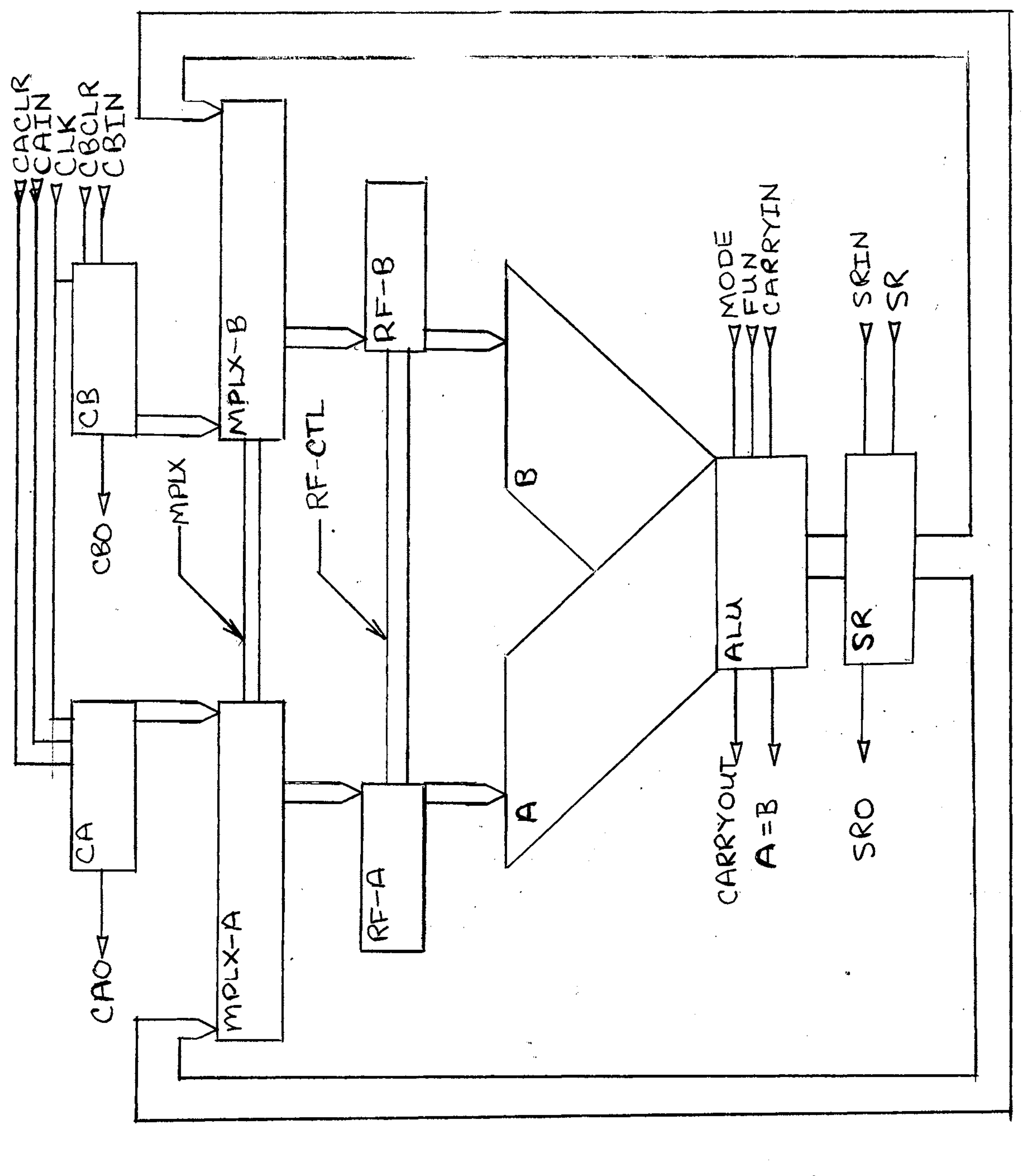


Figure 7. ALU Block Diagram

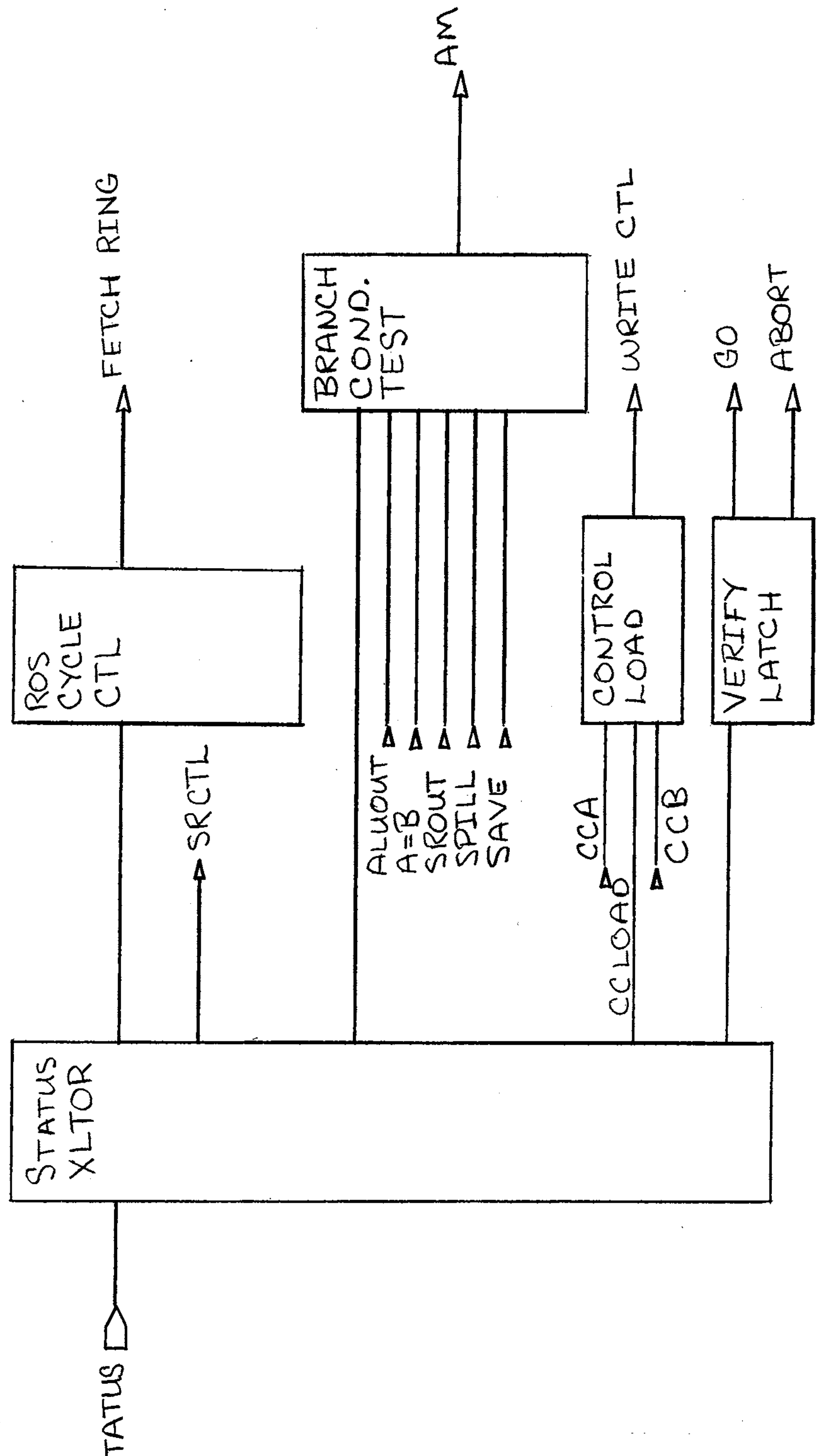


Figure 8. ALU Control Block Diagram

register file. One sixteen bit word is applied by the ALP to the shift register. "ALUOUT" and "A = B" outputs are available and their status may be stored by the micropro-gram (SAVE). Six control signals are required by the ALP: a four bit function code, a one bit modifier, and a CARRYIN bit. The function field depends upon the one bit modifier (MODE). This modifier selects arithmetic or logical funct-ions. "A = B" is a special output which may be used for exact comparison.

4.6 The Shift Register (SR) and Shift Control

The SR is used to accomplish the left shift. The SR receives and stores the sixteen bit word from the ALP out-puts. The output word of the SR may be routed to the RF for storage via the MPLX. The shift out bit of the SR may be monitored, under microprogram control, for both a one or zero shiftout. Either condition may be interrogated by the microprogram. Two control bits and one STAT function are required by the SR.

The number of left shifts is determined by the binary number stored in the tolerance factor keys (TOLSWT). The shift occurs by command of the microprogram. The micropro-gram execution is suspended until the shift is complete. Interrogation of the shiftout flags may be done at this time.

4.7 Status and Control Triggers (STAT)

The STAT box consists of a 4-bit decoder which generates miscellaneous control functions. Functions performed by these control signals are: no operation, stopping ROS instruction execution, branch condition testing, status monitoring, and address modification generation. One four bit field is required to generate the sixteen control functions. The status preservation bit is used by this unit also.

5. Display Control and Interface Control System

5.1 The Display Control (DC)

The DC buffers the output lines that are to be displayed at the engineering console. These output lines include the ROS address register (ROSAR), the verify latch (ABORT), and the validity latch (PGTV). A lamp test function is available as a wire jumper on the card.

5.2 The Interface Control (IC)

The IC card is used to verify the current reading on the external period averaging counter. The value displayed by the counter is verified by an indicator on the engineering console. Input to the external counter is prepared by the IC; only good pulses will be transmitted. If good pulses have been sent when a bad pulse arrives, a high frequency burst is sent to the counter to force an early display. The validity control line is available to control external recording devices.

6. Programming Characteristics

6.1 ROS Instruction Word Format

Using Figure 5 as a guide, the various control fields are diagrammed. The 36-bit instruction word is quasiparallel fetched and executed in monophase. Most fields are encoded. Thirty instruction words are available on six diode matrix cards. The matrix is capable of field modification with standard tools.

6.2 Multiplexer Control Field (MPLX)

This 2-bit field controls both 32 to 16 multiplexers. MPLXSEL is used to select the counter or the shift register as the data source. MPLXSTRB causes the multiplexer to be disabled (outputs assume low state).

6.3 Register File Control Field (RF)

Two 6-bit groups define this field. One 6-bit group is associated with the "A" side of the ALP; the other group is associated with the "B" side of the ALP. RA-RFXX and RB-RFXX are 2-bit binary register read addresses for the "A" and "B" register files, respectively. WA-RFXX and WB-RFXX are 2-bit binary register write addresses for the "A" and "B" register files, respectively. RA-RFEN and RB-RFEN are 1-bit enable lines for the read function. WA-RFEN and WB-RFEN are 1-bit enable lines for the write function. Simultaneous read and write functions may be performed by the register file. The special STAT function, LOAD, requires WA-RFEN to be in the disabled state.

6.4 Arithmetic/Logic Processor Control Field (ALP)

The ALP field consists of six bits, four bits define the desired function, one bit defines the mode, and one bit controls the lowest order carry bit. The 4-bit function field (FUNX) defines two different sets of operation depending upon the field modifier bit (MODE). Arithmetic or logical operation may be selected by the MODE control line. The carry control bit (CARRYIN) introduces the carryin at the least significant bit of the ALP. The CARRYIN control line is used in performing arithmetic computations.

6.5 The Status Retention Control Field (CTL)

The CTL is a single bit field used to cause information to be retained by the processor for consequent interrogation. This control line is referenced SAVE. SAVE causes these status conditions to be sampled and held: the ALP A=B and ALUOUT signals, and the CTRSPILL signal. The shift register most significant bit is monitored under control of the SAVE control line and the SHIFT control line. Two shift conditions are monitored. If a "zero" is ever shifted into the most significant bit, a flag will be set (-SRO). If "one" is ever shifted into the significant bit, a different flag is set (+SRO). These two conditions allow for the detection of register overflow during a shift operation.

6.6 The Shift Control Field (SR)

The SR field consists of two bits. One control bit allows information to be loaded, broadside, into the register. The other control bit causes a predetermined number of left shifts to take place.

The SRSET control line is used after the shift register has been cleared (by STAT function, SRCLR) to cause information to be loaded in parallel from the ALP output port. This information, once loaded, is immediately available to the SR port of the multiplexer.

The SRSHFT control line is used to cause the microprogram execution to be suspended, the tolerance factor value to be loaded into the shift counter, the specified number of pulses (left shifts) to be delivered to the shift register clock inputs, and any significant value spilled from the shift in complete (all shift pulses delivered) the microprogram execution continues with the next instruction. This asynchronous shift technique was used to allow the shift to take place at a speed greater than that of the microprogram fetch and execution cycle.

6.7 The Status and Control Trigger Field (STAT)

The STAT field is a 4-bit binary encoded field. This field produces sixteen control signals. Each signal is referenced by a name of the form, STATx, where "x" is a hexadecimal number representing the code which generates this control signal (e.g., a STAT field of "0110" would generate "STAT6"). These control signals may be further processed before being distributed. The STAT field controls the branching and execution functions of the ROS sequencer (RS). The special control signals for the register files (RF) and the shift register (SR) are also generated here. Finally the verify latch which reflects the result of the compare is controlled by two STAT functions. Three STAT functions are spares.

STAT0 (NOOP) is a no-operation. This function is used when none of the special control signals are required. This function has no effect on the processor.

STAT1 (CLK1) causes the microprogram execution to be suspended pending a second level interruption by the counter control (C). All other second level interruptions will be ignored.

STAT2 (CLK2) causes the microprogram execution to be suspended pending completion of a SHIFT operation. All the second level interruptions will be ignored (Overrun detection is optional).

STAT3 (CLK3) is used to halt the microprogram for error diagnosis. This function is basically a maintenance feature and should not be used in normal microprograms.

STAT4 (STATA) is used to test the ALP "A=B" condition. If the "A=B" condition is satisfied, the address modifier

(AM) is set and a branch occurs.

STAT5 (STATB) is used to test the ALP "ALUOUT" bit. If "ALUOUT" bit is a logic "1", the address modifier (AM) is set and a branch occurs.

STAT6 (STATC) tests for the C "SPILL" condition. If a counter "SPILL" has occured, the address modifier (AM) is set and a branch occurs.

STAT7 (STATO) tests the shift register most significant bit monitor for a logic "1" shifted out. If this monitor latch is set, it indicates that a positive number was being left shifted and significance was lost during the shift. This conditon causes the address modifier (AM) to be set and a branch occurs.

STAT8 (GO) sets the verify latch to the "GO" state. This indicates that the last comparison result was within the tolerance bound.

STAT9 (ABORT) sets the verify latch to the "ABORT" state. This indicates that the last comparison result was outside the tolerance bound.

STATA (LOAD) examines the current counter latch (CC) and enables the register file (RF) bank ("A" or "B") corresponding to the idle counter. This function stores the value of the idle counter in appropriate register file (RF) bank.

STATB (CLEAR) sets the data stored in the shift register to all zeroes. This function is required before SRSET can be issued.

STATC (STATE) tests the shift register most significant bit monitor for a logic "0" shifted out. If this monitor latch is set, it indicates that a negative number was being left shifted and significance was lost during the shift. This condition causes the address modifier (AM) to be set and a branch occurs.

Since the STAT field is encoded, all functions described above are mutually exclusive. STAT4, STAT5, STAT6, STAT7, and STATC test values prepared by the SAVE control signal.

6.8 The ROS Control Field (ROS)

The ROS field contains 5-bit base address for the next instruction to be executed. The phrase "base address" is used as a reminder that, in case of a branch, the instruction that will be fetched and executed will be the next odd addressed instruction. The address modifier (AM) bit and the ROSA4 address bit are "OR'ed" to effect the branch.

CHAPTER VI

CONSTRUCTION OF THE SEQUENTIAL PHASE COMPARATOR

1. Selection of a Logic Family

1.1 Speed

The velocity requirements helped fix the maximum speed at which the processor needed to perform. This information also influenced the selection of the 16-bit word length. Estimates of the ROS speed combined with a maximum number of twenty instruction steps allowed a frequency of 1.25MH$_z$ to be used as the ROS cycle frequency. This placed the maximum pulse frequency at 50KH$_z$. After the processor was constructed the maximum pulse frequency was approximately 100KH$_z$. This increase was attributed to the reduction of instruction steps.

1.2 Noise

The equipment for the processor is in a high electrical noise environment. The construction materials used were not intended for high speed logic circuits. These two reasons indicate the need for a logic family that is relatively insensitive to noise on its inputs or power supply.

Transistor-Transistor Logic (TTL or T^2L) has excellent noise rejection specifications. MOS Logic has an even higher noise immunity because of the higher voltage logic levels available.

1.3 Commercially Available Functions

The size and construction time for the unit was reduced by the use of Medium Scale Integration (MSI) logic devices. Thus the number of "packages" of logic circuits was reduced. This reduction in size made construction easier by reducing the number of connections needed.

1.4 Transistor-Transistor Logic (TTL or T^2L)

TTL was chosen because of its superior speed, noise immunity, and the availability of complex functions. Propogation delays in single gate structures are typically less than 15ns. Counters and shift registers operate at maximum clock rates of 20 MH_z to 35 MH_z. The input noise band is typically 2 volts with 0.8 volts to ground and 2.8 volts to supply margins. Several hundred different functions are available in the logic family in addition to the common Small Scale Integration (SSI) units.

This logic family enjoys the support of the entire semiconductor industry and, therefore, its existence is guaranteed for some time.

2. Physical Layout

2.1 General Requirements

Maintenance considerations required that each integrated circuit (IC) to individually replaceable. Each logical unit of the processor should be as redundant as possible to facilitate error location within each unit. Standard inputs should be adopted to assist in forming interchangable units. Physical size should be kept to a minimum; in no circumstance should the wiring board dimensions exceed twenty inches on the diagonal (see TTL noise specifications). Expense was to be spared as much as possible.

2.2 The Gate

Surplus equipment was salvaged for a swing-out gate with five removable bays. These bays contain seventeen circuit board edge connector sockets each. Two of the bays contain a vertical support and three circuit board edge connector sockets. These two bays also have doors. The two bays were designated BAY 1 and BAY 5. BAY 1 and BAY 5 were used for power supply enclosures. The three remaining bays; BAY 2, BAY 3, and BAY 4, were used to house the processor logic. The entire gate was mounted on hinges in the back of a steel cabinet. The steel cabinet contains the tangential blower, power supply connectors, and the engineering console.

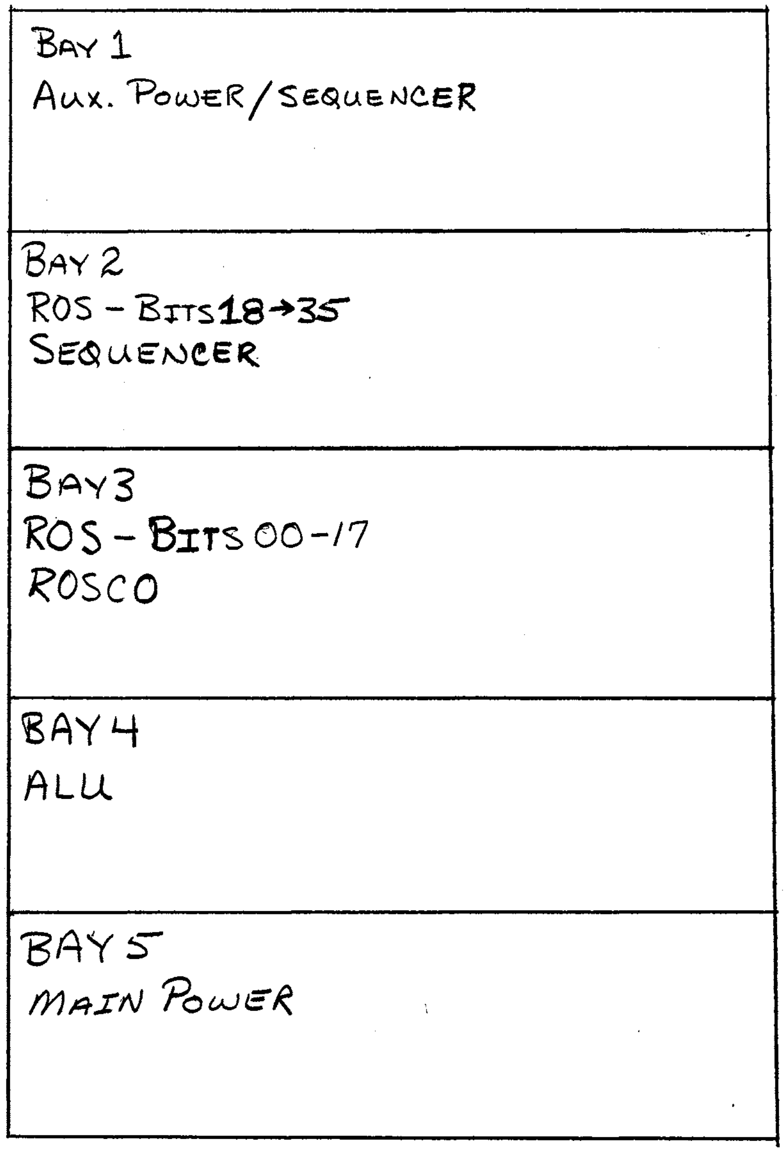


Figure 9. Gate Layout

2.3 The Power Supply Bays (Bay 1 and Bay 5

BAY 1 occupies the top position in the gate. A + 24 volt DC power supply is contained here with the rélays used for sequencing the other power supplies in the processor. The manual controls for applying system power are connected here. Access to the power supply is available thru the front door panel.

BAY 5 contains the +10 volt DC power supply used for operating the logic of the processor. The main line contactor is located in this bay also. BAY 5 occupies the lowest position in the gate and is immediately above the tangential blower. Access to the power supply is gained through the front panel door. Access to the contactor is gained from the rear of the gate.

2.4 ROS Storage Bays (BAY 2 and BAY 3).

BAYS 2 and 3 contain all the associated logic for the ROS memory and its controls. BAY 3 also contains the display controls. BAY 2 and BAY 3 have identical card complements except: BAY 2 contains the ROS sequencer (RS) and BAY 3 contains the display control (DC) and ROS oscillator (ROSCO). One +5v DC power supply (PSI), two ROS latch (RL) cards, three ROS matrix (RM) cards and two ROS driver cards (RD) may be found in each bay. Cards of the same type are completely interchangeable (e.g., RL's with RL's, RD's with RD's). The ROS matrix (RM) storage cards, of course, cannot be interchanged. BAY 2 contains all instruction addresses and ROS bits 18-35. BAY 3 contains all instruction addresses and ROS bits 00-17. A dual ground rail is located at the top and bottom of each bay. These rails are used to terminate the ground line carried by the twisted pair leads which interconnect the logic bays.

2.5 The Processor Bay (Bay 4)

BAY 4 contains the arithmetic/logic unit and its associated support equipment. BAY 4 has a dual +5 volt DC power supply instead of the single units in Bays 2 and 3. One dual +5 volt DC power supply card (PS2), one ALU input control card (AI), four arithmetic/logic processor cards (AL), one ALU output control card (AO), and one interface control card (IX) are contained in BAY4. The 4 AL cards are completely interchangeable 4-bit processors. The dual ground rail is also available on BAY4.

3. Power Supply Sequencing and Control

3.1 Power Sequencing

The power supplies are applied in a specific sequence. This procedure is followed to minimize the instantaneous load on the BAY5 power supply. A large fluctuation in the BAY5 voltage could cause loss of regulation on the individual bay voltage regulators. This condition could result in circuit damage. A six pushbutton switch is used to manually sequence the power supplies. An indication is given at the switch of a power supply or sequence malfunction.

3.2 Emergency Power Off

This feature is used to remove all power from the system in the event of a catastrophy. A T-handle switch is located on the front panel for this purpose. Operation of this switch requires a manual sequencer reset to resume operation (i.e., further operation of the switch will be ineffective).

4. Logic Bay Assembly

4.1 Wiring

The edge connector sockets are mounted in each bay. Connections are made by a pin which is inserted into the back of the socket. The pin is retained by a spring clip. In this manner the bay sockets may be interconnected by a single wire jumper terminated at each end by a pin. Two connections may be made on each of the thirty-six edge connector lands.

Connections between bays require some type of shield. Coaxial line terminated with pins is used for distribution of the clock signals and the Doppler pulse input. Other interconnections are made using twisted pair. One wire (blue) carries the signal; the other (white) is terminated at one end (usually the driving point) on the bay ground rail and insulated at the other end. The bay interconnection wires are held close to the gate frame by the diode and socket assembly boards.

4.2 BAY Wire Listing

Construction of the bay wiring was done using a list of computer generated instructions. Information found on the schematic is encoded by hand into a computer readable format. This information is processed by the computer and one of the results is a list of wiring instructions. The list is used by the technician to construct the back plane bay assembly.

The wire list is also used to compile a signal name directory and clock circuit loading conditions. The back plane wiring was verified by continuity tests performed with special test point cards and the wire list.

5. Logic Card Assembly

5.1 Preparation of Circuit Boards

As stated before, the hardware was salvaged from surplus parts. The circuit boards required for the logic cards were obtained by removing the old components from the circuit board, removing the pull-out handle, protecting the edge connector lands, and using copper etchant to remove the old printed circuit. The circuit boards were sanded smooth and the protective coating on the edge connector bands removed.

The circuit boards were drilled, according to a card layout drawing, to accommodate the IC sockets. A visual inspection for foreign material or out-of-line holes was made. Contact cement was used to secure the IC sockets to the circuit board. The Wire Wrap ® pins were then cleaned to remove excess cement. The edge connecting lands were supplied with a length of #24AWG solid wire to be used as a Wire Wrap ® connector. A continuity check was performed on the connector assembly.

5.2 Logic Card Wire Listing

A computer generated set of instructions was used to perform the wiring of the logic card information from the schematic diagram encoded by hand and processed by a computer program. One output of this "WIRE LIST" program is the wiring instructions. The wire list allows the construction of different boards to become routine. A continuity check is performed on the logic card after it has been wired.

A signal name directory is also produced by the program. This information can be useful in error diagnosis.

5.3 Function Tests

After all logic cards were assembled, functional tests were made to determine if various sections of the unit were operational. The shift register was loaded and shifted manually. The counters were alternated and incremented. The running latches and shift counters were tested. These tests were performed using the Hewlett-Packard IC Logic Trouble-shooting Kit (5015T). A "logic clip", which displays the status of all pins of a digital IC, is included as well as a logic probe and pulser. This equipment allowed individual function tests to be made independently of the other logic.

CHAPTER VII

THE MICROPROGRAM

1. Design of the Program

1.1 Preliminary Program Design Steps

A prose description of the functions to be implemented by the microprogram was made. Using the description as a guide, a sequential chart was made of these functions as they were to be performed. This chart was reworked to include all possible common processing steps. The need to conserve instruction storage is great. After this initial optimization attempt, a flowchart was prepared from the sequence chart.

The flowchart was constructed on a one instruction to one function block basis. These function blocks were broken down further to reflect the actual control signals generated. This second series of flowcharts have an identical number of function blocks which correspond to each field of the microinstruction. The STAT field function block is broken down into yet a third series of flowcharts representing the effect of each function. All flowcharts are cross-referenced such that the exact function of any microinstruction or field may be quickly determined. A computer program, OSFLOW was used to generate the flowcharts.

2. Alteration of the Microprogram

2.1 Modification of the Diode Matrix

Replacement of the six diode matrix cards (RM) allows the complete removal of the microprogram. Modifications to the microprogram should be made after following the design procedures just mentioned. Using the ROS instruction word format and the flowchart listings, diodes may be inserted or removed from the matrix. A diode in a particular word and bit position indicates a logic "1"; the absence of a diode is a logic "0".

Since each instruction word contains the ROS base address of the next instruction to be executed (except branches), "patching" the microprogram can be done quite easily. "Patching" refers to the insertion or deletion of microcode. Microinstructions may be added or deleted without altering more than one existing instruction.

Great care should be exercised in the modification of the microprogram. Any changes should be carefully "thought out" and extensively "desk checked" <u>before</u> any modifications are undertaken.

2.2 The Programmable ROM

The next step in the evolution of ROS would be to integrate the control store into a small group of ICs. Consideration of this step has been made and, if the increased speed and decreased bulk is warranted, the modification should be incorporated. Currently available field programmable ROMs can be used with the existing microinstruction format.

CHAPTER VIII

OPERATION AND TESTING

1. Operation of the Engineering Console Controls

1.1 General Control Functions

The processor should require little manual intervention since the successful operation of the processor is highly time dependent. For this reason the manual controls are few and they are limited in scope. The orientation of the controls is toward maintenance and malfunction diagnosis. The initial reset of the processor is all that need concern the operator.

The power distribution section of the console is used to turn the processor power supplies on and off. An Emergency Power Off (EPO) T-handle switch removes all power from the processor in the event of a catastrophe. A six push-button back-lighted switch is used to perform manual power sequencing. If the unit is successfully powered on, all indicators will be white. Activation of any supply will cause the tangential blower to be engaged. The blower will remain on until all supplies are off and the "blower off" switch operated. An indicator light is provided for the blower circuit.

Five ROS address keys (ROSAK) are available on the front panel. They are inactive unless enabled by another front panel control. These keys serve as an alternate source for the ROS address register (ROSAR). This function is to allow access to ROS at any address.

The ROS enable (ROSEN) key is used to select the ROSA field or the ROSAK for the next fetch address. The ROSEN key disables the address modifier (AM) thus eliminating any unwanted branches.

The single step (SS) key is used to select the ROS oscillator (ROSCO) or the manual pushbutton (MANPB) as the source for the ROS clock which controls the instruction fetch. The SS key may be used with the ROSEN key to "loop" on a particular ROS address (repeatedly fetch).

The manual pushbutton (MANPB) key is used with the SS key to manually control execution of the microprogram. This feature is strictly a maintenance aid.

The stop (STOP) key causes execution of the micropro-gram to be immediately terminated. This key has only main-tenance functions.

Four tolerance keys (TOLSWT) are available to the operator to set the tolerance factor. These keys should only be adjusted with the ROSAK set to zero and ROSEN key down.

1.2 How to Start the Microprocessor

The microprogram address that contains the special STAT function (CLKI) is indicated on the microprogram flowchart. The microprogram execution should begin at this point.

First, set the ROSEN key down, then put the ROS address for the first microprogram in the ROSAK. Set the SS key down then up. With no pulses being received the ROS address display (ROSAD) should retain the starting address (exclusive of the "SPILL" condition). For the provided comparison microprogram, the starting address is zero. Acttivating the SS key while processing will cause the processor to temporarily halt.

1.3 How to Change the Tolerance Factor

Place the processor in the halt state by setting the SS key. Set the TOLSWT to the desired value. Restart the microprogram as described.

1.4 Malfunction Detection

If the processor should halt for any reason, always note the contents of the ROSAD. Armed with this information, use of the flowchart will usually indicate the section that is malfunctioning.

2. Test Apparatus and Procedure

2.1 Generation of Variable Pulse Trains

A test pulse train can be generated with a serial word generator (similar to Hewlett-Packard 8006A). The tolerance and missing pulse detection system can be easily checked using a word generator and a dual trace occilloscope. Switches, corresponding to bits within the word, can be set to simulate good, bad, and marginal pulse trains.

Modifications to the microprogram can be tested by sending double pulse bursts manually with the "SPILL" detector disabled. This allows the processor to be "walked through" the new microprogram.

2.2 Connection of Test Equipment

As shown in Figure 10 , the word generator replaces the Schmitt Trigger and the oscilloscope is used to monitor the word generator output (return to zero format) and the "GO" latch output. A storage oscilloscope, triggered by the word generator sync output, could be used for convenient single burst displays.

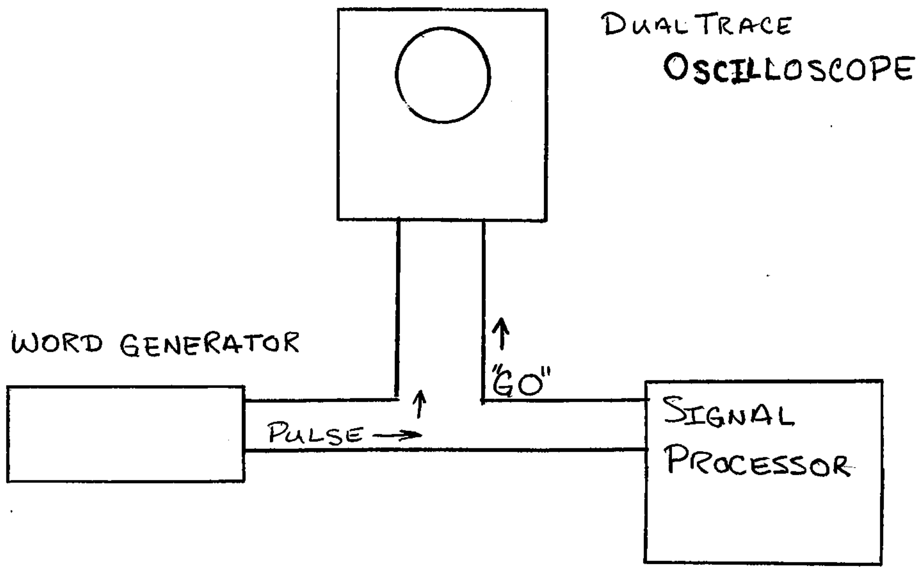


Figure 10. Test Setup

CHAPTER IX

CONCLUSION

1. Summary

1.1 The Laser Anemometer Date Processor Problem

The problem of missing pulse detection was investigated and a device constructed to process measurement information. Various constraints made an analog solution infeasible. The experimental nature of the proposed design required extreme flexibility. For these reasons a microprogrammed digital processor was chosen as the framework for the design.

Construction of the processor was accomplished using the TTL family logic and Wire Wrap ® techniques. Salvaged surplus equipment was used for the circuit boards and frames. A wire list generating program was developed and used to make reports. These reports were used to wire the processor and perform certain tests and checkout procedure.

The completed processor, with its comparison microprogram, was tested using the serial word generator and a dual trace oscilloscope. Further operational testing of the processor is being carried out by the Basic Fluid Dynamics Laboratory at the OSU Mechanical and Aerospace

Engineering Laboratory. Preliminary test results show agreement with those measurements taken by hand.

1.2 Possible Extensions

The large number of tolerance factors available allow a technique to be used to increase the effective upper frequency of the processor. Assume that the maximum "native" mode frequency is $100kH_z$ and that the tolerance should be 25%. If we have a sufficient number of pulses in the burst, then a digital division may be used to lower the input frequency. At the same time, if the tolerance divided by the same amount, the effect will be to raise the upper frequency. This arrangement can be used to extend the same range beyond the native mode limit.

The commercial TTL logic used could be replaced by the Schottky TTL or emitter coupled logic (ECL). Either of these logic families is capable of extending the frequency range of the processor. Using programmable ROMs also would increase the processor speed.

Program modifications can be made without great difficulty. Experiments with different comparison schemes might prove helpful.

BIBLIOGRAPHY

(1) Asher, J. A. "Laser Doppler Velocimeter System Development and Testing". General Electric Technical Information Series, Report No. 72CRD295. Schenectady: General Electric, 1972.

(2) Clark, Christopher R. Designing Logic Systems Using State Machines. New York: McGraw-Hill, 1973.

(3) Dollhoff, Terry L. "Microprogrammed Control for Small Computers". Computer Design, Vol. 12, No. 5 (1973), pp. 91-97.

(4) Flynn, M. J., and R. F. Rosin, "Microprogramming: An Introduction and A Viewpoint". IEEE Transactions on Computers", Vol. C-20, No. 7. IEEE, (1971).

(5) Henderson, J. T. "Write a Fortran Wire-Listing Program". Electronic Design, Vol. 21, No. 11 (1973), pp. 140-142.

(6) House, David L. "Micro Level Architecture in Minicomputer Design". Computer Design, Vol. 12, No. 10 (1973), pp. 75-80.

(7) Husson, Samir S. Microprogramming, Principles and Practices, Englewood Cliffs: Prentice-Hall, 1970.

(8) Lewis, Donald R., and W. Ralph Siena. "Microprocessor or Random Logic?" Electronic Design, Vol. 21, No. 18 (1973), pp. 106-110.

(9) Redfield, S. R. "A Study in Microprogrammed Processors: A Medium Sized Microprogrammed Processor." IEEE Transactions on Computers, Vol. C-20, No. 7, IEEE, (1971).

(10) Smid, R. W. "Frequency Tracking System". (Paper presented to LDA workshop, Oklahoma State University, 1973).

VITA

Lloyd Neal Salsman

Candidate for the Degree of

Master of Science

Thesis: A DIGITAL SIGNAL PROCESSOR FOR LASER DOPPLER ANEMOMETER SYSTEMS

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in El Reno, Oklahoma, October 15, 1949, the son of Mr. and Mrs. Norbert N. Salsman.

Education: Graduated from El Reno Highschool, El Reno, Oklahoma, in May, 1967; received Bachelor of Science degree in Electrical Engineering from Oklahoma State University in 1972; completed requirements for Master of Science degree at Oklahoma State University in December, 1973.

Professional Experience: System Programmer at Oklahoma State University Computer Center, 1968-1973.

Professional Organizations: Member of Eta Kappa Nu; member of IEEE.