INTRODUCTION

This paper describes the architecture of the CPU and Memory for the Central Air Data Computer (CADC) System used in the Grumman/ Navy F14A carrier-based fighter aircraft. The CADC performs specialized computerational functions in response to input stimuli such as pressure sensors, temperature sensors and closed loop feedback inputs. Outputs from the CADC system are used to drive pilot visual displays (such as, altimeter, temperature indicator, mach number indicator, etc.) and to provide control inputs for other aircraft systems. The outputs from the CADC are in the form of digital and analog signals. Figure 1 illustrates a block diagram for the CADC.

Being in a flight environment meant that certain constraints must greatly reflect the architecture of the CPU and Memory. These constraints were size, power, real-time computing capability and cost, not necessarily in that order. Other constraints such as temperature, acceleration and mechanical shock affected the overall design of the CADC.

The size of the CPU-Memory was limited to a maximum of 40 square inches. This included the arithmetic section, read-only memory, and read/write memory. Since the unit was to be packaged on a printed circuit card the number of layers of the p.c. card was an important consideration. The power consumption had a limit of 10 watts at ambient 25^{o}C. This was principally a function of the capabilities of the p.c. card to withstand the heat.

The required computing capacity for the CPU was not defined at the beginning. This meant that the system had to be somewhat flexible to changes in computational load. Of course limits had to be set to be able to work within the other constraints. What was known about the computation was the form of the equations to be implemented. This included polynominal evaluations, data limiting, data comparison and discrete or flag inputs and outputs. This meant that the arithmetic and logical functions of the system had to handle at least the following operations:

Multiply
Divide
Add
Subtract
Limits
Square Root
And
Or
Conditional Transfer
Unconditional Transfer
Receive Discrete Data
Receive Digital Data
Output Discrete Data
Output Digital Data

The last constraint of cost was certainly important since the system would eventually go into high volume production.

FUNCTIONS TO BE IMPLEMENTED

Before we proceed, a better understanding of the functions to be implemented is necessary. The function that most often occurred was the polynomial,

$$F(X) = a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0$$

where x was the input, either from outside the CPU (sensors) or from its own memory.

In order to save arithmetic time the polynomial was implemented in its "nested" form as follows:

$$F(X) = ((((((xa_6) + a_5)\ x + a_4)\ x + a_3)\ x + a_2)\ x + a_1)\ x + a_0$$

The data limit function was one that would accept 3 binary inputs, an upper limit (U), a lower limit (L) and a parameter (P). The output would then be as follows;

P if $U \geq P \geq L$

L if $P < L$

U if $P > L$

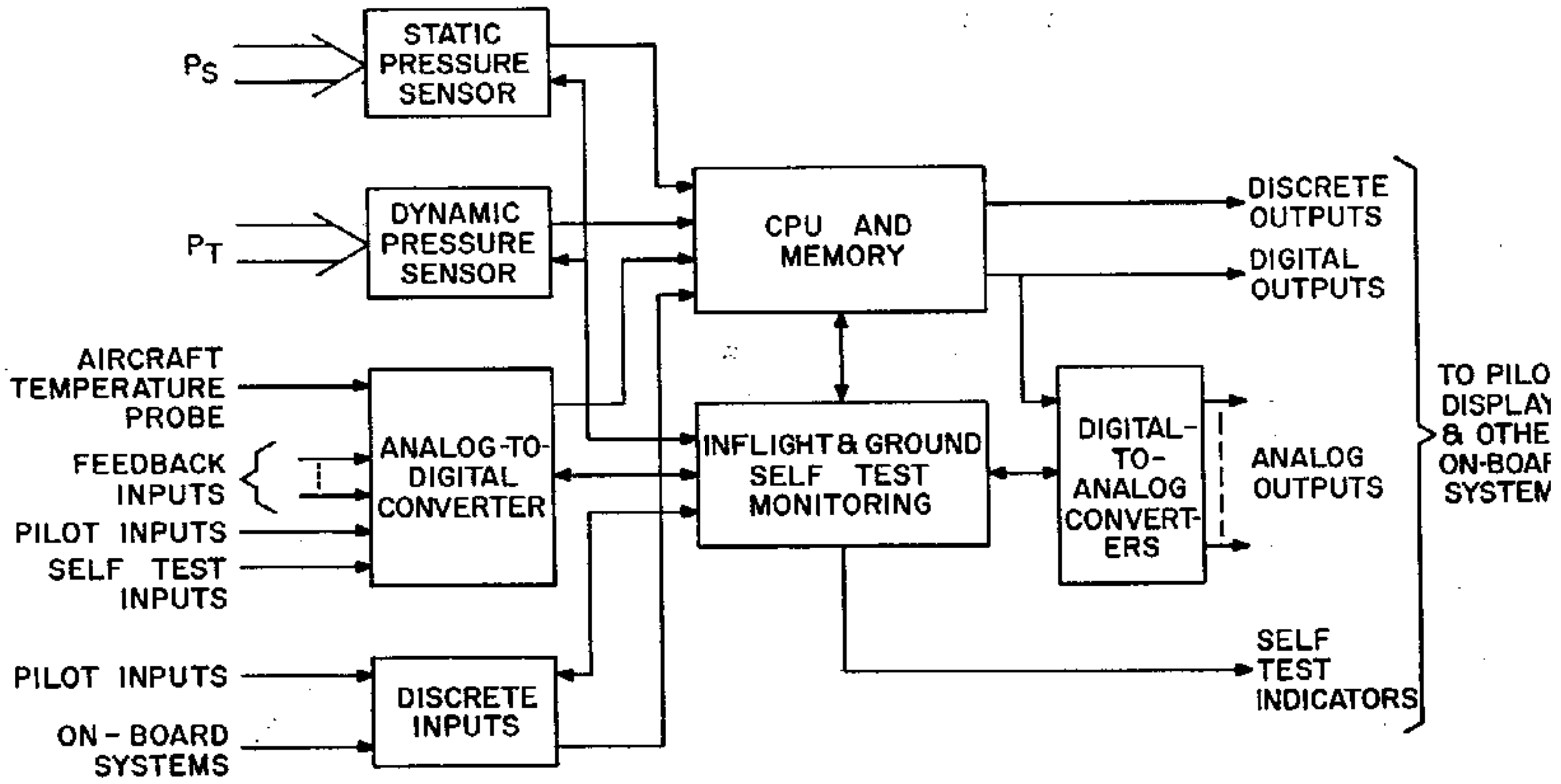


FIGURE 1 — [illegible] OF A TYPICAL CENTRAL AIR DATA COMPUTER [illegible] SYSTEMS

Even though such a function could be programmed in software, it was decided to build it in hardware since it was used often enough.

Data conditioning and scaling also had to be accomplished. This involved the following simple expressions:

$$F = \frac{A \pm B}{C}$$

$$F = \frac{A - B}{A + C}$$

Again the occurance of these were frequent enough to warrent hardware consideration. This will become apparent later when the hardware is discussed.

Since size and power consumption was of the ultimate importance, MOS technology was chosen as the means of circuit implementation. This allowed greater packaging densities to be obtained that otherwise would not be. The slowness of MOS devices and the high thresholds used allowed a design that was virtually immune to electrical noise on the ground or transmitted from packages within close proximity. The higher supply voltages required resulted in a more efficient power supply design.

NUMBER SYSTEM

The CPU is a fractional fixed point machine with the most significant bit a sign bit and the other bits representing data. Negative numbers are represented in two's complement notation. Two's complementation was chosen to avoid the ambiguity of double zeros.

The word length chosen for the system was 20 bits; 19 bits of data and 1 bit for sign. This length was chosen after a thorough analysis of the accuracy required for certain throughput calculations such as the rate of change of altitude function.

Early in the architecture study it was realized that package size and quantity should be kept to a minimum if we were to meet the size constraints established. With minimum packaging space requirements it was necessary to use packages with the fewest possible leads. This would minimize the complex p.c. card interconnect which was inevitable. Because of this the processor was designed to transfer data serially throughout the entire system.

PROCESSOR PARALELLISM

As is known by all computer designers, serial machines are usually not the best way to go if computational speed is needed. To get around this it was decided to have several arithmetic or processing units working at the same time. This resulted in a technique known as "pipeline processing" or "pipeline concurrency". As defined by Bell and Newell[1] "pipeline concurrency is the name given to a system of multiple functional units, each of which is responsible for partial interpretational and execution of the instruction stream."

This system uses multiple functional units each dedicated for a specific task. These functional units are:

1. Parallel Multiplier Unit (PMU)
2. Parallel Divider Unit (PDU)
3. Special Logic Function (SLF)
4. Data Steering Unit (SL)
5. Random Access Storage (Read/Write) (RAS)
6. Read-Only Memory Unit (ROM)

Figure 2 shows a block diagram of the functional units as they would work together in a typical system. Each unit was designed to operate as a separate entity and could be used without the need of any of the other units. This was done to provide maximum expandibility with minimum additional hardware. Each functional unit is controlled by its own micro-instruction ROM. The micro-instructions are also transferred serially to minimize package pin count. Temporary data storage is provided in the form of read/write memory.

1. Gordon C. Bell, and Allen Newell, Computer Structures: Readings and Examples, McGraw Hill Book Co., N.Y., 1971, pg. 84

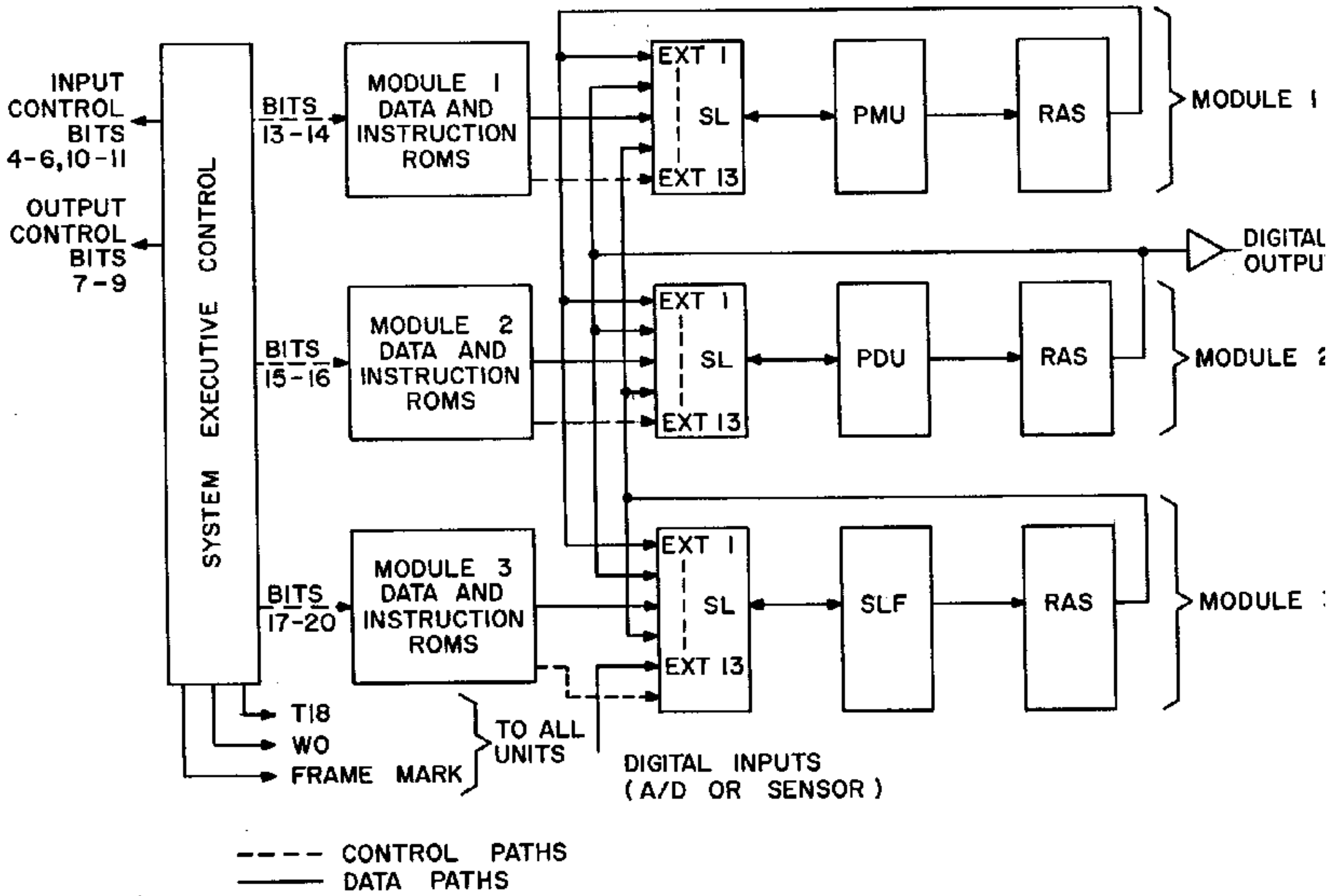


FIGURE 2 — BLOCK DIAGRAM OF CPU-MEMORY SHOWING I/O AND EXECUTIVE CONTROL

Before looking at the functions of each of these units a brief look at the timing is needed. Figure 3 illustrates this timing. The CPU-Memory clock is 375KHz. One complete clock period, defined as a bit time is 2.66μsec. Every 20 consecutive bit times are defined as a word. The first bit time of a word is called T0, and the last time of the word is called T19. Two types of words are used in the system, W_A and W_0. In W_A, the arithmetical algorithms operate and instruction words are shifted serially into each functional unit. In W_0, computational inputs and outputs are shifted serially among the units. A word mark used to distinguish word times is a signal coincident with T18 of every word time. Two consecutive word times, W_A and W_0, is called an operation (op) time. To distinguish the final operation time a frame mark is generated in the system executive control. The time between frame marks is called a frame. A frame includes one complete cycle of computations. The frame mark is microprogrammed to allow the user to restart the computational cycle when all previous computations are complete. Since this system must operate in real time it was therefore necessary to obtain the most computation from each functional unit during each frame time.

ARITHMETIC UNITS

The Parallel Multiplier Unit accepts two serial inputs, multiplicand and multiplier, in one word time (W_0) and produces their properly rounded product by means of a parallel algorithm in one more word time.

The product is shifted out in the next W_0, while inputs for the next operation are simultaneously shifted in. The multiplication operation is achieved using Booth's algorithm[2]. The PMU does not need an instruction word to operate, but is capable of operating continuously in this manner.

The Parallel Divider Unit accepts two serial inputs, dividend and divisor, in one word time (W_0) and produces the proper quotient

2. Yoahan Chu, Digital Computer Design Fundamentals, McGrawHill Book Co., N.Y., 1962, pg. 32.

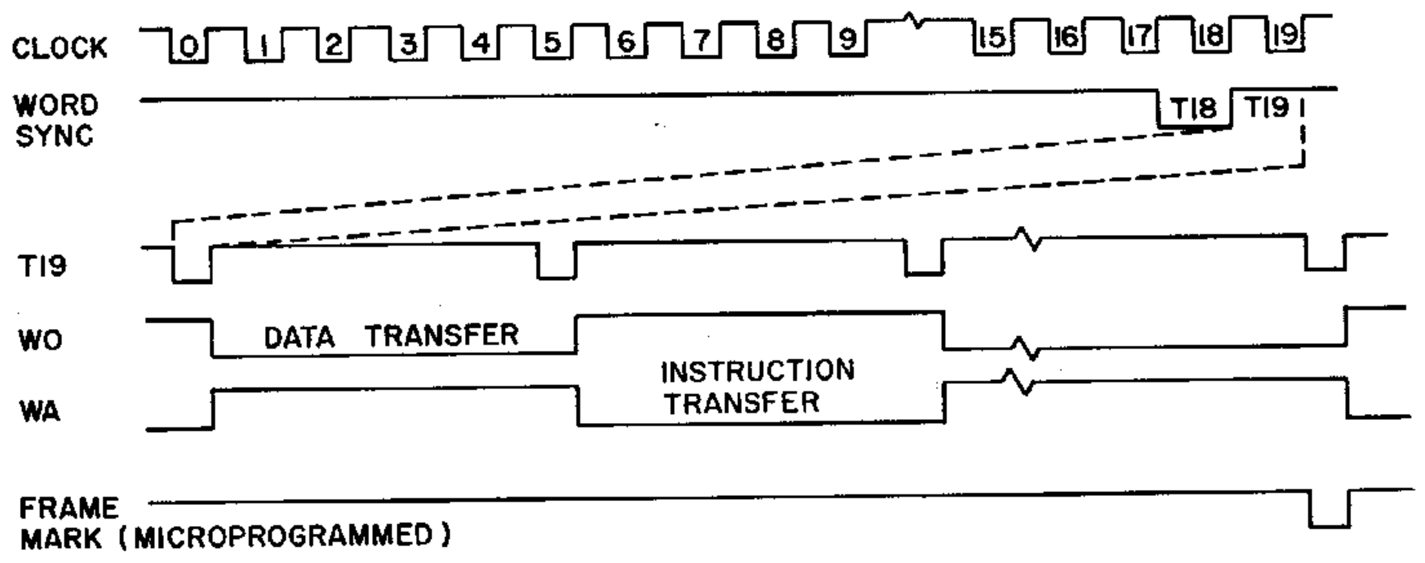


FIGURE 3 - TIMING DIAGRAM

by means of a parallel algorithm in one more word time (W_A). The quotient is shifted out in the next W_0, while inputs for the next operation are simultaneously shifted in. The division operation is achieved by using a non-restoring division algorithm.[3] An actual photograph of the PDU chip is shown in Figure 4.

The Special Logic Function performs logical operations and generates specific data and logic outputs. The unit accepts an instruction word which specifies details of the operation.

The fundamental logical operation of this unit is the limit function. It consists of three registers (U,P, and L) whose inputs arrive in W_0. One of these registers is picked at the end of W_0 by associated comparison logic. Other logic functions such as AND's, OR's, GRAY CODE (special), Conditional, and Unconditional data branching is also included in this logic.

The Data Steering Unit operates as a three channel serial digital data multiplexer. Information is shifted serially through the device during W_0. A 15 bit instruction word is accepted during W_A that specifies which input or input combinations (Add or subtract) is to be "steered" to each of three data outputs. The instruction word for this unit is the last 15 bits of the 20-bit instruction word. From the least significant end, the first four bits specifies the selection for Output 1. The next four bits specifies the selection for Output 2 and the last seven bits specifies the selection for Output 3. Addition or subtraction is performed by specifying that output combination to be "steered" to the output. By performing additions and subtractions in this manner the programmer can obtain the sum and difference during the same word time that the data is being transferred. This transfer may be either to or from the memory or arithmetic units. Specific instruction codes are interpreted as follows:

---

3. Yoahan Chu, <u>Digital Computer Design Fundamentals</u>, McGrawHill Book Co., N.Y., 1962, pg. 39.

LSB 6	7	8	9	Selected to Output 1
0	0	0	0	EXT 1
0	0	0	1	EXT 2
0	0	1	0	EXT 3
0	0	1	1	EXT 4
0	1	0	0	EXT 5
0	1	0	1	EXT 6
0	1	1	0	EXT 7
0	1	1	1	EXT 8
1	0	0	0	EXT 9
1	0	0	1	EXT 10
1	0	1	0	EXT 13
1	0	1	1	EXT 11
1	1	0	0	EXT 9 + EXT 4
1	1	0	1	EXT 10 + EXT 4
1	1	1	0	EXT 4 + EXT 8
1	1	1	1	EXT 2 - EXT 8

10	11	12	13	Selected to Output 2
0	0	0	0	EXT 1
0	0	0	1	EXT 2
0	0	1	0	EXT 3
0	0	1	1	EXT 4
0	1	0	0	EXT 5
0	1	0	1	EXT 6
0	1	1	0	EXT 7
0	1	1	1	EXT 8
1	0	0	0	EXT 9
1	0	0	1	EXT 10
1	0	1	0	0.111111111111111111 ($\overline{T19}$) (Maximum Positive Number)
1	0	1	1	EXT 11
1	1	0	0	EXT 9 + EXT 4
1	1	0	1	EXT 10 + EXT 4
1	1	1	0	EXT 4 + EXT 8
1	1	1	1	EXT 2 - EXT 8

14	15	16	17	Selected to Output 3
0	0	0	0	EXT 1
0	0	0	1	EXT 2
0	0	1	0	EXT 3
0	0	1	1	EXT 4
0	1	0	0	EXT 5
0	1	0	1	EXT 6
0	1	1	0	EXT 7
0	1	1	1	EXT 9
1	0	0	0	Σ_1 (controlled by bit 18)
1	0	0	1	Σ_2 (controlled by bits 18-20)
1	0	1	0	EXT 11
1	0	1	1	EXT 2 + EXT 4
1	1	0	0	EXT 2 - EXT 8
1	1	0	1	EXT 2 - EXT 4
1	1	1	0	EXT 4 - EXT 2
1	1	1	1	EXT 4 + EXT 8

18	
1	EXT 12 + EXT 8 = Σ_1

MSB 19	20	
0	0	Σ_1 + EXT 2 = Σ_2
0	1	Σ_1 - EXT 2 = Σ_2
1	0	Σ_1 + EXT 4 = Σ_2
1	1	Σ_1 - EXT 4 = Σ_2

Since each SL has three outputs the capability is there to perform three additions and/or subtractions during one word time. Output 3 has the added feature of performing two additions during the same data transfer. This would be A + B + C. This feature was incorporated to handle the many cases of concluding a polynomial evaluation and adding another value to it. This would look like $(A_1x+A_0) + Y$. As the product of A_1x is being transferred to memory the A_0 constant and Y value can be added to it. As can be seen the addition and subtraction, if used properly, can take "zero" time since transferring of data has to be accomplished anyway.

These four units comprise the hardware that can perform arithmetic operations. As shown in Figure 2 each PMU, PDU and SLF has a Data Steering Unit associated with them. As will be discussed later this allows many data paths to be operating consecutively. Thus computational power is greatly increased.

MEMORY UNITS

There are two memory units, Read-Only Memory (ROM) for instruction storage and Random Access (Read/Write) Storage (RAS) for intermediate data storage. The ROM operates as a 2560-bit random access/sequential access device. It internally stores fixed patterns of 128 words of 20 bit length for serial readout. The patterns are specified by the user. The ROM has provision to accept a 20-bit serial binary word address. The first seven bits indicates which of the 128 words is to be accessed, and the next three bits specifies, by manufacture mask decoding, which ROM out of a possible eight ROM group should have its output enabled.

The address management is accomplished by a register counter contained within the ROM. This counter has the following capability: (1) resettable, (2) steppable such that the memory can be sequenced through the 128-word field, (3) accepts a retain address command and holds the present address, and (4) accepts a numerical input for independent address modifying or loading.

To accomplish the above address management, six logic inputs (reset (R), retain, increment, load, add, sub) and one address data input are provided.

The address field is defined as follows: the first seven LSB's will select one of 128 20-bit words. The next three will be used to select one of eight ROM's, the remaining ten will be zero. The three-bit select field will determine, by mask decoding, which chip's (ROM) output is to be enabled. ROM's that are not enabled by the chip select field have disabled their outputs. The particular application described in this paper uses the ROM in the sequential access mode only.

The RAS memory operates as a 16-word random access read-write storage device. Information is shifted serially into and out of selected registers during W_0. Memory readout is nondestructive, i.e., if a register is selected to be read only, then the data are also rewritten into the selected register. The RAS accepts an instruction word of 5 bits during W_A. The least significant four bits of the instruction word specifies one of sixteen 20-bit serial registers. The contents of the selected register is shifted out to the data output serially during the next W_0 and W_A. At the W_0 time, information from the data input is written into the selected register if the fifth control bit is a "1". If the fifth bit is a "0", the register contents remain unaffected. An external logic input "inhibit write" is provided to inhibit writing. The instruction word selects registers according to the following configuration:

M				L	Selected Register
5	4	3	2	1	
	0	0	0	0	16
	0	0	0	1	1
	0	0	1	0	2
	0	0	1	1	3
	0	1	0	0	4
	0	1	0	1	5
	0	1	1	0	6
	0	1	1	1	7
	1	0	0	0	8
	1	0	0	1	9
	1	0	1	0	10
	1	0	1	1	11
	1	1	0	0	12
	1	1	0	1	13
	1	1	1	0	14
	1	1	1	1	15
0					Read
1					Read & Write

An actual photograph of the RAS chip is shown in Figure 5.

SYSTEM OPERATION AND INSTRUCTION FORMAT

This processor uses a total of 28 circuits. The breakdown is as follows:

1 --	Parallel Multiplier Unit (PMU)
1 --	Parallel Divider Unit (PDU)
1 --	Special Logic Function (SLF)
3 --	Random Access Storage Units (RAS)
3 --	Data Steering Units (SL)
19 --	Read-Only Memory Units (ROM)

An actual photograph of the processor is shown in Figure 6. The system has been divided into three modules, each module identified by its arithmetic unit. The modules are as follows:

Multiply Module 1	Divide Module 2	Special Logic Module 3
1 -- PMU	1 -- PDU	1 -- SLF
1 -- SL	1 -- SL	1 -- SL
1 -- RAS	1 -- RAS	1 -- RAS
3 -- Data ROM's	1 -- Data ROM	2 -- Data ROM's
3 -- Instruction ROM's	Instruction ROM's	Instruction ROM's

Each arithmetic unit has an associated steering unit and RAS unit. This is not an absolutely necessary association but is chosen to add flexibility to the programming. Figure 7 illustrates another possibility of a circuit arrangement. It is important to note that the hardware arrangement is flexible to the given problem statement. If many multiplies are needed in a given problem then two or more multiplier units could be used instead of one. The same can be true for any of the other circuits.

Each module as shown in Figure 2 is a specialized CPU with its own data and instruction storage and read/write storage. Each operates as told by its instruction word. All of the modules are operating consecutively, each doing what it was dedicated for. The interconnection of the modules determines how the units work together The timing pulses to the units provide synchronization between them.

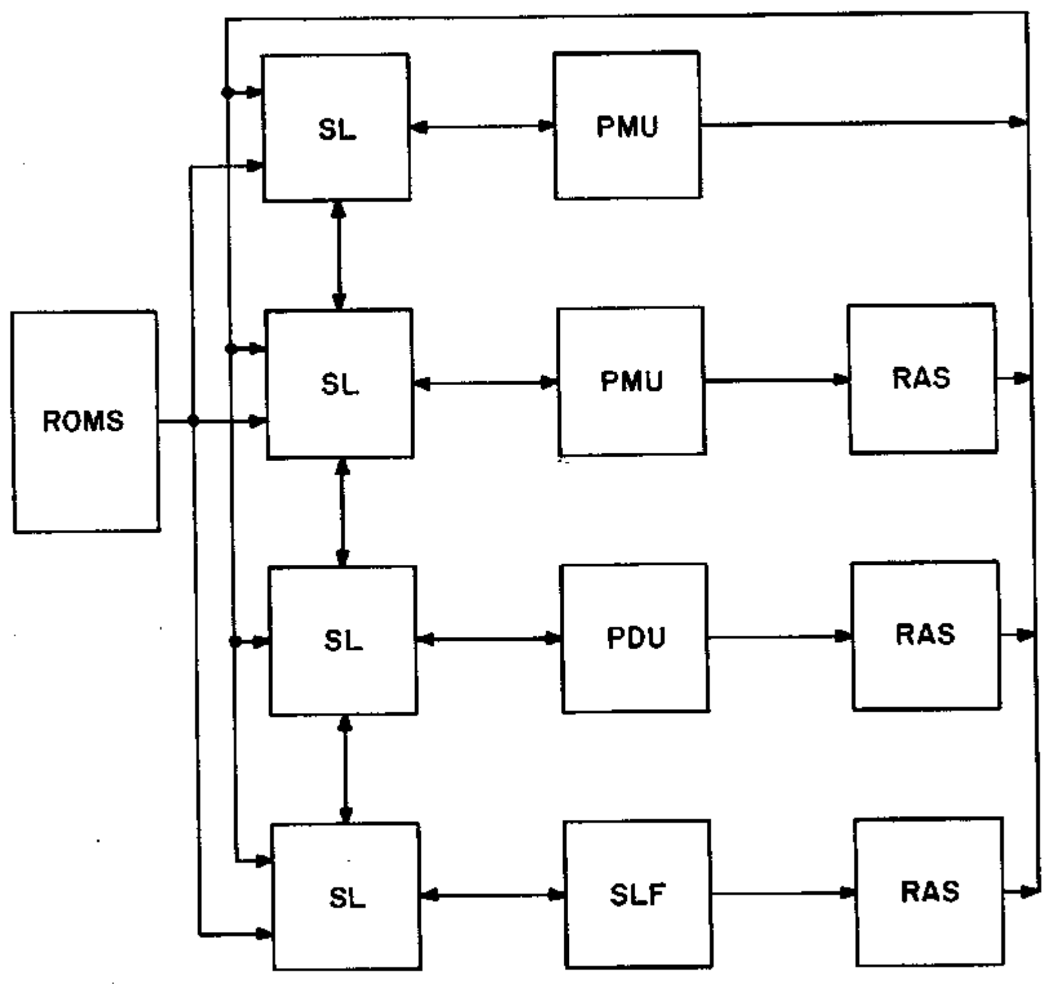
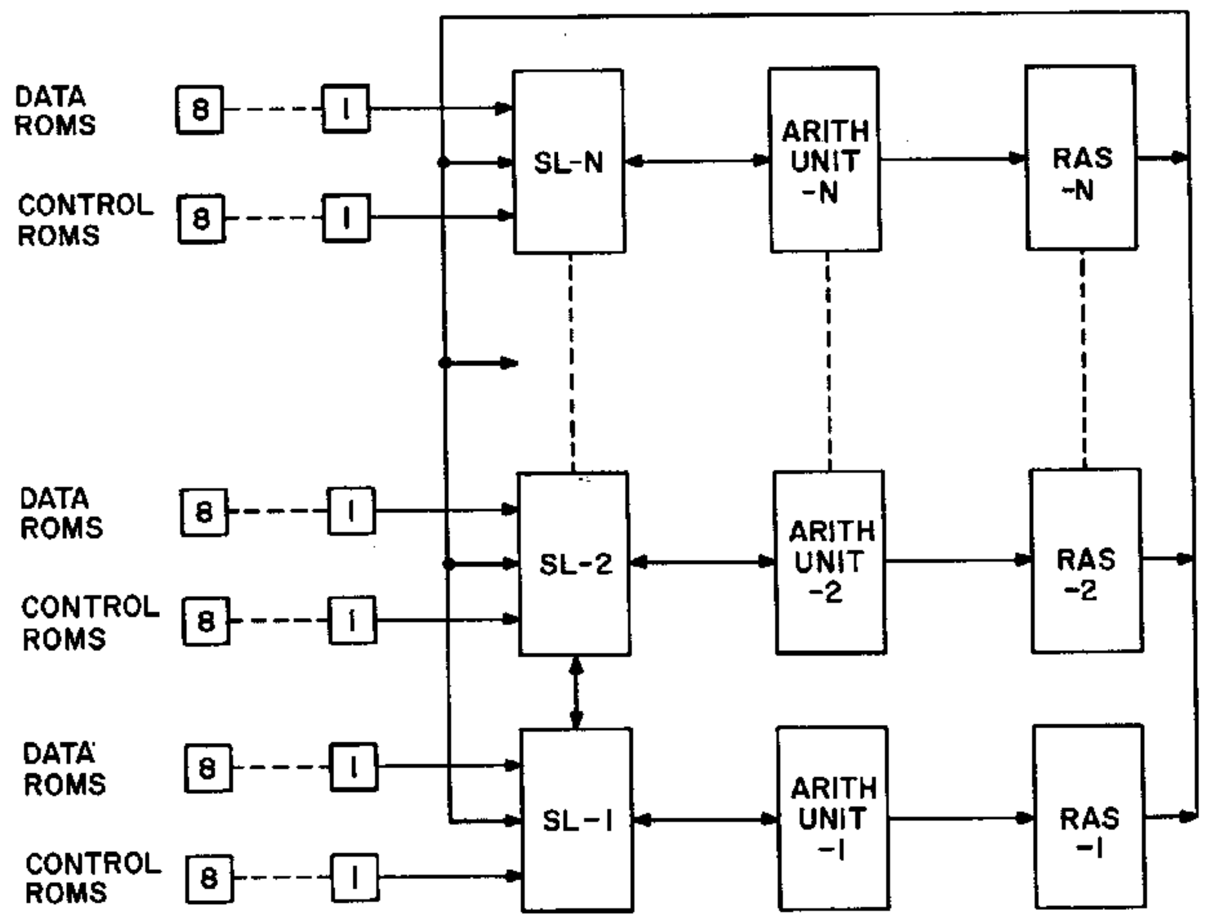


FIGURE 7- ANOTHER POSSIBLE CIRCUIT ARRANGEMENT

For a unit to accept data from a source the output of the source must be connected to the input of the steering unit for that module. The transfer of data from module to module is a function of the interconnect and the instruction word. Every W_A time a new instruction word is transferred from each instruction ROM group to the units in each module. These instruction words define the data path to be selected to each output of each steering unit during the next W_0 cycle. System operation continues in this manner, W_A (instruction fetch), W_0 (data fetch) until the system executive control resets the instruction ROM's. Data storage ROM's can be treated like the outputs from the other circuits. They are connected to the inputs of the steering units. The total number of ROM's used in each group is a function of the total amount of storage required and the update rate of the system. The number of arithmetic units is a function of the amount of calculation and the update rate of the system. As shown, the hardware is flexible to handle changes in two directions: computational load and update rate. This flexibility is required to handle the various types of systems to be implemented.

A general hardware arrangement illustrating the flexibility of the hardware is shown in Figure 8. The final configuration chosen by the user must depend on the total system programming requirements.

For the system in Figure 2 the following would be its three instruction word formats.



EACH ROM EQUALS 2560 BITS

FIGURE 8 – GENERAL HARDWARE ARRANGEMENT

Module 1 Instruction Word

LSB	1	Selects PMU RAS Registers	To PMU RAS
	2		
	3		
	4		
	5	Write	
	6	Selects Multiplicand Data	To PMU Steering
Out 1	7		
	8		
	9		
	10	Selects Multiplier Data	
	11		
Out 2	12		
	13		
	14	Selects RAS Storage Data	
	15		
	16		
Out 3	17		
	18		
	19		
MSB	20		

Module 2 Instruction Word

LSB	1	Selects PDU RAS Registers	To PDU RAS
	2		
	3		
	4		
	5	Write	
	6	Selects Divisor Data	To PDU Steering
Out 1	7		
	8		
	9		
	10	Selects Dividend Data	
	11		
Out 2	12		
	13		
	14	Selects RAS Storage Data	
	15		
	16		
Out 3	17		
	18		
	19		
MSB	20		

Module 3 Instruction Word

```
LSB      1  ┐                             ┐
         2  │   Selects SLF               │  Also Selects
         3  │   RAS Register              ├  SLF Functions    To SLF and
         4  ┘                             │                    SLF RAS
         5      Write                     ┘
         6  ┐                                        ┐
Out 1    7  │   Selects Lower                        │
         8  │   Limit Data                           │
         9  ┘                                        │
        10  ┐                                        │
Out 2   11  │   Selects Upper                        │
        12  │   Limit Data                           │
        13  ┘                                        ├   To SLF
        14  ┐                                        │     Steering
        15  │                                        │
        16  │                                        │
Out 3   17  │   Selects Parameter Data               │
        18  │   and RAS Storage Data                 │
        19  │                                        │
MSB     20  ┘                                        ┘
```

SYSTEM EXECUTIVE CONTROL

The system executive control is used to control many parts of the total system. In general, it is used to control the inputs (A/D, discretes, sensor inputs), the outputs (D/A, digital data, discrete), and the processor modules.

The system executive control consists of an associated group of ROM's, supplying a 20-bit control word every operation time. Each bit has a preassigned task based upon the hardware configuration. The assignment for each of the bits might be as follows:

MSB	20	Enables Discrete Inputs to Processor
	19	SLU Control ROM Enable
	18	SLU Data ROM-2 Enable
	17	SLU Data ROM-1 Enable
	16	PDU Control ROM Enable
	15	PDU Data ROM Enable
	14	PMU Control ROM Enable
	13	PMU Data ROM Enable
	12	Enables System Self Test
	11, 10	Analog - to - Digital Converter Control
	9, 8, 7	Output Digital - to - Analog Control
	6, 5, 4	Digital Sensor Input Control
LSB	3, 2, 1	Control of System Sync and Control Counters

System executive control bits 13 through 19 are assigned to control the processor modules. The reason for having such a control is to minimize the storage requirements in the ROM's by disabling their operation when they are not being used.

SOFTWARE COMMENTS

The software for this system is closely associated with the hardware arrangement. To program such a system efficiently requires complete knowledge of the hardware. At the present time there is no assembler available to properly handle the multiple instruction set and data path considerations to yield an efficient program.

Present programming requires hand coding at the machine code level. This is certainly tedious but it results in a more efficient circuit interconnect and minimum total instruction words. All of this reflects the number of arithmetic modules and ROM's necessary to handle a particular problem.

ADVANTAGES AND DISADVANTAGES

The advantages of such a system are determined by the constraints under which system design was accomplished. The size of the circuit packages were such that the original size constraint was met. The sizes are indicated below:

Multiplier Unit	- 28	pin dual in-line package
Divider Unit	- 28	"
Special Logic Unit	- 28	"
Steering Unit	- 28	"
R.O.M.	- 14	"
R.A.S.	- 14	"

The flexibility of the hardware to adapt to a great number of problem statements along with its R.O.M. reprogramming capability is certainly a feature desirable in all systems of this kind. The fact that the software and hardware can be used in conjunction with each other helps to reduce the total parts count in the system. When total parts are reduced then power consumption and cost will be reduced. Power consumption for the CPU-Memory discussed in this paper exceeded the original 10 watts. This was due to the large quantity of read-only memory needed to implement the problem statement. With present day technology the cost of such a commercial CPU-Memory system could be less than $2000 in large quantities. This is based on a price of less than $65 per circuit type.

The disadvantages of this system is reflected in the software. To hand code the entire program in a machine such as this is a time consuming and tedious effort. To use an assembler or translator to perform the coding would result in a less efficient program and possibly more hardware. To obtain the minimal hardware configuration a person with both hardware and software knowledge of the system is needed. Future designs of this nature need to be worked closer with the software requirements in order to minimize the effort required for programming.

SUMMARY

A CPU-Memory system has been presented that satisfied the requirements of a particular application. The "pipelined" approach resulted in a system with enough flexibility to satisfy the present requirement and to adapt to other areas of application. MOS technology provided "state of the art" power dissipations and packaging densities obtainable in no other way. This system proved to be the most effective in meeting the given requirements.

BIBLIOGRAPHY

1. Staff of American Micro-systems, Inc., Engineering Course in MOS Technology, American Micro-systems, Inc., Santa Clara, Calif., 1970

2. Thomas C. Bartee, Irwin L. Lebow and Irving S. Reed, Theory and Design of Digital Machines, McGraw Hill Book Company, N.Y., 1962

3. Lee L Boysel and Joseph P. Murphy, "Four-Phase LSI Logic Offers New Approach to Computer Designers," Computer Design, April 1970, pp. 141-146

4. T. C. Chen, "Parallelism, Pipelining and Computer Efficiency," Computer Design, Jan. 1971, p. 69

5. Robert W. Cook and Michael J. Flynn, "System Design of a Dynamic Microprocessor," IEEE Transactions on Computers, Vol. 19, No. 3, March 1970, pp. 213-222

6. William R. Graham, "The Parallel and the Pipeline Computers," Datamation, April 1970, p. 68

7. D. C. Gunderson, "Some Effects of Advances In Memory System Technology On Computer Organization," IEEE Computer Group Computer, Nov./Dec. 1970, pp. 7-11

8. Robert C. Haavind, "The Many Faces of Microprogramming," Computer Decisions, Sept. 1971, pp. 6-10

9. Raymond M. Holt, "MOS Processor For the F14A CADC," Garrett AiResearch Corp., Torrance, Calif., Technical Report No. 71-7266, April 1971

10. Samir S. Husson, Microprogramming: Principles and Practices, Prentice-Hall, New Jersey, 1970

11. R. L. Kleir and C. V. Ramamoorthy, "Optimization Strategies for Microprograms," IEEE Transactions on Computers, Vol. 20, July 1971, pp. 783-794

12. Frank J. Langley, "Small Computer Design Using Microprogramming and Multifunction LSI Arrays," Computer Design, April 1970, pp. 151-157

13. William J. Patzer and Gilbert C. Vandling, "Systems Implications of Microprogramming," Computer Design, December 1967, pp. 62-66

14. Montgomery Phister, Jr., Logical Design of Digital Computers, Wiley and Sons, N.Y., 1963

15. Stephen R. Redfield, "A Study in Microprogrammed Processors: A Medium Sized Microprogrammed Processor," IEEE Transactions on Computers, Vol. 20, July 1971, pp. 743-750

16. Robert F. Rosin, "Contemporary Concepts of Microprogramming and Emulation," Computer Surveys, Vol. 1, No. 4, Dec. 1969, pp. 197-212

17. "Design of Microprogrammable Systems," Signetics Memory Systems Application Note SM S0052AN

18. "Economic Advantages of Microprogramming", Signetics Memory Systems Application Note SM S0053AN

19. Donald E. Waldecker, "Comparison of a Microprogrammed and a Non-Microprogrammed Computer," Computer Design, June 1970, pp. 73-77

20. Thomas M. Whitney, "A Test Procedure for Microprogrammed Systems," Presented at the 3rd Annual Workshop on Microprogramming, Buffalo, N.Y., October 1970

21. M. V. Wilkes, "The Growth of Interest in Microprogramming: A Literature Survey," Computer Surveys, Vol. 1, No. 3, Sept. 1969, pp. 139-145