Fall 1983

The Fundamental Hardware Elements and Options of Digital Signal Processing

Mark A. Hemmerlein
University of Central Florida

THE FUNDAMENTAL HARDWARE ELEMENTS AND OPTIONS OF DIGITAL SIGNAL PROCESSING

BY

MARK A. HEMMERLEIN
B.S.E., University of Central Florida, 1980

RESEARCH REPORT

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

Fall Term
1983

ABSTRACT

The hardware available for digital signal processing is evaluated. The elements required for implementation of digital signal processing applications are identified. These elements-- memories, ALUs and multipliers-- are analyzed. Then the operation of parts which make up a range of possible solutions are evaluated. Thus, the graduating engineer is aided in making a transition from the theoretical to the practical world.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The field of digital signal processing (DSP) continues to be a rapidly growing frontier in modern technology. In the past decade, advances in digital hardware have opened doors to new designs and inventions which were previously not possible. There are two major subdivisions of digital signal processing. The first is digital filtering, which can be broken into finite impulse response (FIR) and infinite impulse response (IIR) filters. The second is spectral analysis, which can be broken into the calculation of spectra by either discrete Fourier transform (DFT) or by statistical techniques for random signal analysis.

The purpose of this paper is to describe some of the currently available off-the-shelf hardware for digital signal processing. The purpose is not to discuss design techniques for digital filters or spectral analysis. It is to examine the operation and specifications of various industry standard parts.

II. BACKGROUND

The primary arithmetic operations for digital signal processing are addition and multiplication. Consider an example from the first major subdivision of digital signal processing-- digital filtering. The second order filter model below requires 5 multiplications and 4 additions for each value of y(n).

$$y(n)+A_1y(n-1)+A_2y(n-2)=B_0x(n)+B_1x(n-1)+B_2x(n-2)$$

Also consider an example from the second major subgroup of digital signal processing. The fast Fourier transform (FFT), which is at the heart of most spectral analysis applications, is used to determine the discrete Fourier transform (DFT) of the finite duration sequence x(n) of length N.

$$X(k)=\sum_{n=0}^{N-1}x(n)\exp-\left[\frac{2\pi}{N}nk\right]j$$

where k = 0, 1, 2. . . N-1

In order to perform this spectral analysis, $\frac{N}{2}\log_2 N$ complex multiplications and $N \log_2 N$ complex additions are required.

The four primary implementations to be considered for the above digital signal processing applications are (1) hardwired logic, including SSI/MSI, semi-custom and custom integrated circuits; (2) 8 or 16 bit general purpose microprocessors; (3) single chip signal processors; and (4) microprogrammable bit slice architectures.

There are some major tradeoffs for these four implementations. If high speed, critical instruction sets or long word lengths are required, the 8 or 16 bit standard microprocessors cannot be used. If parts count and design time must be minimized, then the microprocessor-based design is preferred. Yet, if a system requires high speed and long word lengths and still must be easily upgraded, then the bit slice microprogrammed design might be preferred. It can usually be changed by replacement of a PROM, which is much easier than changing hardwired logic.

Thus, the balance of the paper discusses the memories, adders, multipliers and controllers available for digital signal processing.

III. FUNDAMENTAL ELEMENTS

1. Registers and Memories

Memory is a fundamental element in most any digital system. Digital filters require small amounts of high speed RAM for the storage of the previous input and output values. The spectral analysis design requires a considerably larger RAM memory for FFT calculations. Each of the two designs also require memory for the control program. Memory can be classified in a number of ways. First, memory can be static or dynamic. Static memory retains its information indefinitely as long as power is on. Figure 1 shows a 1K by 4 bit static RAM. When W is high, the data input buffers are inhibited to prevent erroneous data from entering the array. Data within the array can only be changed during an overlap of $\overline{CS}$ and $\overline{WE}$ low. The MCM2148H-45 has a maximum access time of 45 ns. Access time t_A is defined as the time from when the address initially becomes stable to the time when the data becomes valid on the bus during a read cycle. Dynamic memory requires a periodic refresh in order to maintain validity. Dynamic memory achieves much greater density at the cost of additional refresh circuitry and increased maximum access time. Figure 2 shows a block diagram of the MCM4116B-15 which is a 16K bit dynamic RAM with a maximum access time of 150 ns. There are 14 address bits required to decode one of the 16,384 cell locations. The address lines are multiplexed onto the seven address inputs and latched into the on-chip

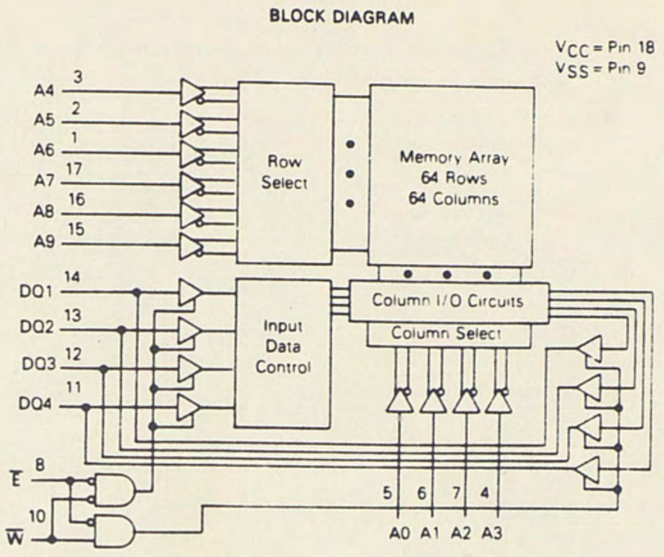


Figure 1. 4096 Bit Static Random Access Memory (Motorola, 1980)

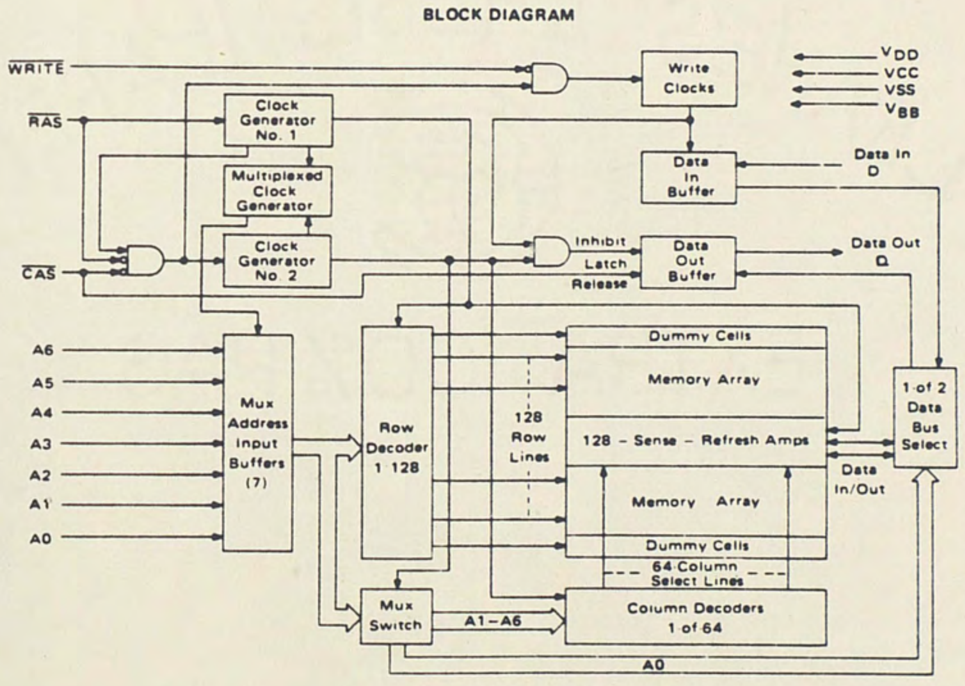


Figure 2. 16384 Bit Dynamic Random Access Memory (Motorola, 1980)

latches. The first clock, the row address strobe ($\overline{RAS}$), latches the seven row address bits into the chip. The second clock, the column address strobe ($\overline{CAS}$), latches the seven column address bits into the chip. Data to be written into a selected cell is then latched into an on-chip register by a combination of $\overline{WRITE}$ and $\overline{CAS}$ while $\overline{RAS}$ is active. Data is to be retrieved from the selected cell in a read cycle by maintaining $\overline{WRITE}$ in the inactive state throughout the portion of the memory cycle in which $\overline{CAS}$ is active low.

Memory can also be classified by writability. ROMs can only be read, not written. PROMs can be written to only once by the user. EPROMs can be erased ultravioletly and reprogrammed. EEPROMs can be both erased and written to electrically. ROMs, PROMs, EPROMs and EEPROMs all retain information after the power is turned off and are usually used to store the controlling program. The read access for the above chips are all very similar. Figure 3 shows a block diagram for an 8192 bit PROM. By applying a unique address to $A_0 - A_9$ and enabling the chip the output can be read on lines $O_0 - O_7$. The access time for such a PROM is around 60 ns. The EEPROM 2816 has a read access time of 250 ns, but it can also be written to electrically in 10 ms. RAMs can be written and read from at about the same speeds. Unlike the previously mentioned forms of memory, the RAM's information will be destroyed when power goes down.

The final way to classify memory is by random or sequential accessibility. In random access memory, successive addresses can be chosen arbitrarily. RAMs may also be accessed through multiple ports. Figure 4 shows the AM29705 Dual Port RAM. The RAM features two separate

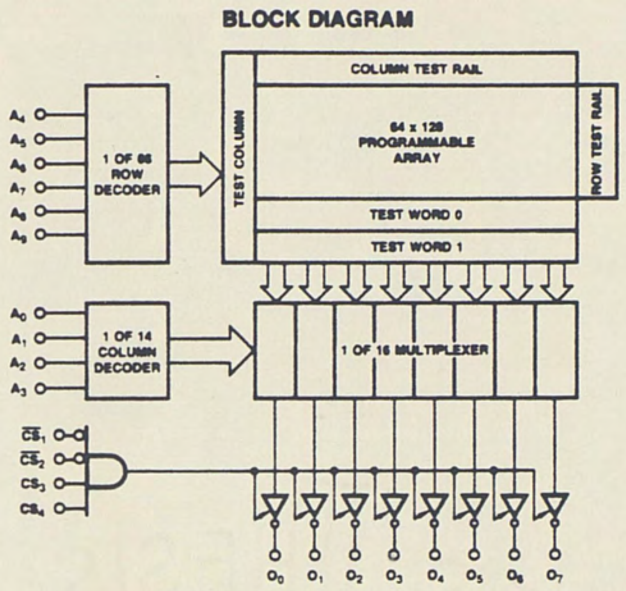


Figure 3. 8192 Bit PROM
(Advanced Micro Devices, 1982)

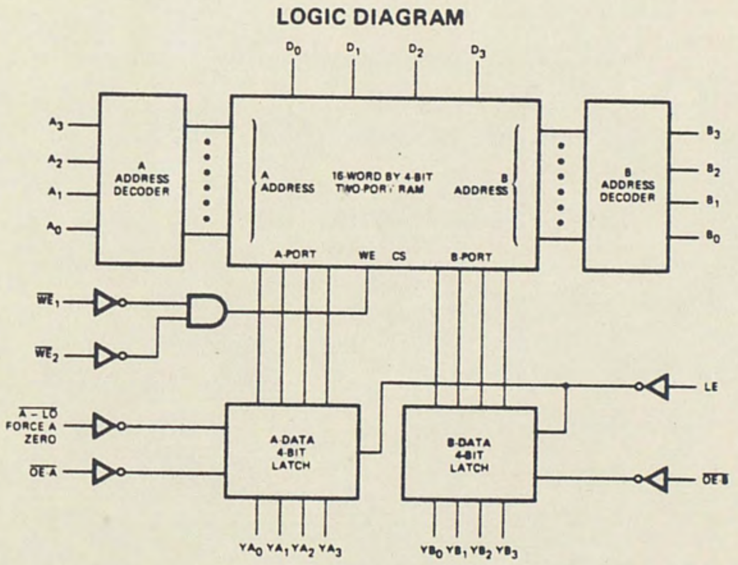


Figure 4. 16 Word by 4 Bit Dual Port RAM
(Advanced Micro Devices, 1982)

output ports such that any two 4 bit words can be read from these outputs simultaneously. In sequential memory, adjacent bits appear at the output in sequence. Sequential memories take the form of first-in-first-out (FIFO) memories and shift registers. FIFOs may be utilized whenever two variable speed processes must be interfaced or a number of samples must be taken before processing may begin (FFT processing). Figure 5 shows the block diagram for the C5/C67401/A FIFO memory. Data enters when the shift in (SI) input goes high. Data exits when the shift out (SO) input goes high. The outputs input ready (IR) and output ready (OR) are low when the memory is full or empty respectively.

2. Addition

Figure 6 shows a ripple carry adder. Arithmetic logic units (ALU) which utilize the ripple carry technique make up the simplest and slowest designs. Consider four AM25LS2517s used in such an ALU shown in Figure 7. The worst case delay is computed using the following three steps. First, select the longest combinatorial delay in the least significant device from any input to the carry output C_{n+4}. This delay is usually from A or B inputs to the carry output. Second, add the carry input to the carry output propagation as many times as required to represent each of the intermediate 4 bit ALUs. Third, take the propagation delay from the carry input to the ALU adder outputs. Table 1 indicates the worst case delay for the ripple carry adder is 103 ns.

Block Diagrams

C5/C67401/A 64x4

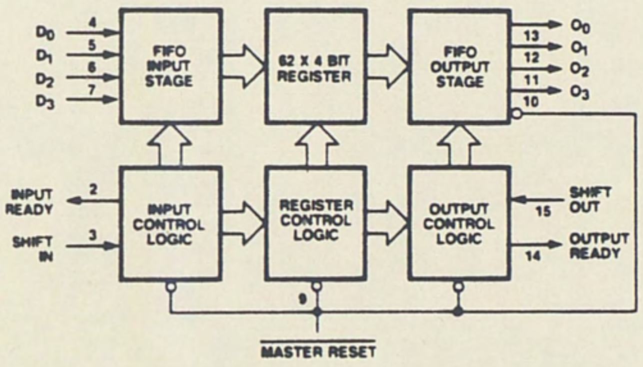


Figure 5. First-In-First-Out
(FIFO) Memory
(Monolithic Memories, 1981)

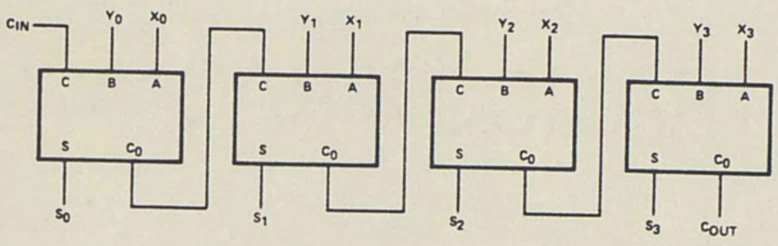


Figure 6. Cascaded Full Adder Cells Connected as a 4 Bit Ripple Carry Adder (Advanced Micro Devices, 1982)

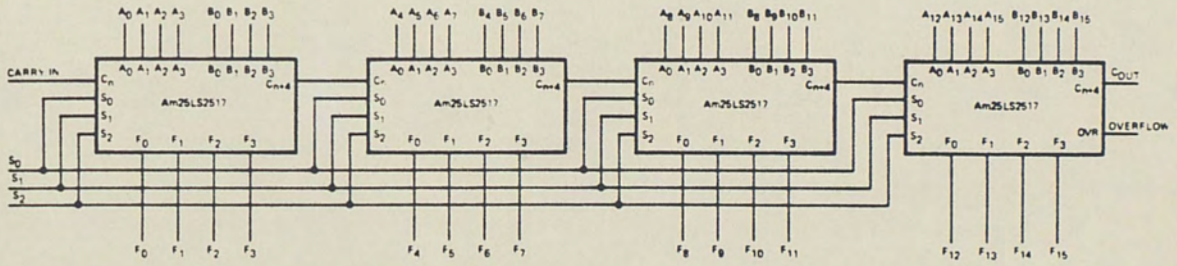


Figure 7. Connection for a 16 Bit ALU Using Ripple Carry (Advanced Micro Devices, 1982)

Figure 8 shows a full lookahead carry 16 bit adder. The worst case delay is computed by summing three delay paths. The path delay for A_i or B_i to G or P is 26 ns (AM25LS381). Path delay for C_n to F_3 is 19 ns. Note the sum delay of 55 ns is almost double the speed of the ripple carry adder.

3. Multiplication

Multipliers can be broken down into two categories-- clocked multipliers and array multipliers. Figure 9 illustrates the structure of a clocked add-shift multiplier. Clock 1 shifts the y bits into the AND gates so that the x bits or 0s are applied to the adders. After the adders have stabilized, clock 2 strobes the adder outputs into the accumulator flip-flops, which then are shifted one bit to the right. This adder, having no lookahead logic, has a worst case delay for an n bit adder of $(n-1)T_C+T_S+T_R$ where T_C is the carry propagation time, T_S is the sum propagation delay, and T_R is the y-shift register settling time. Thus, if n bits are multiplied by m bits the delay time for the multiplication is $m(n-1)T_C+mT_S+T_R$.

The array multipliers are considerably faster than clocked multipliers. The array multipliers consist of a two-dimensional array of one bit adders and are completely memoryless. The settling time for the array of Figure 10 requires m sum delays and n carry delays to reach p_7; but to reach p_{13} n additional carry delays are required. Thus, the total delay time is $2nT_C+mT_S$.

Table 1. Data Path Delay 16 Bit Ripple Carry Adder (Advanced Micro Devices, 1982)

Path	Output			Units
	F_i	C_{n+4}	OVR	
A_i or B_i to C_{n+4}	36	36	36	ns
C_n to C_{n+4}	22	22	22	ns
C_n to C_{n+4}	22	22	22	ns
C_n to F_i	23	–	–	ns
C_n to C_{n+4} or OVR	–	22	22	ns
TOTAL 16-bit delay	103	102	102	ns

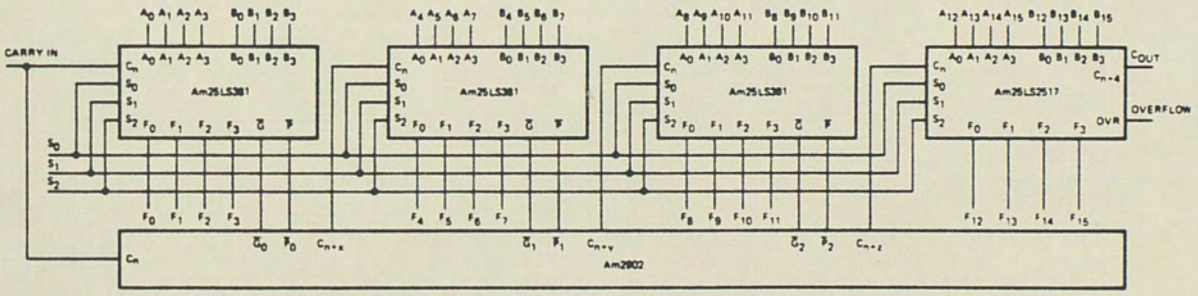


Figure 8. Full Lookahead Carry 16 Bit Adder (Advanced Micro Devices, 1982)

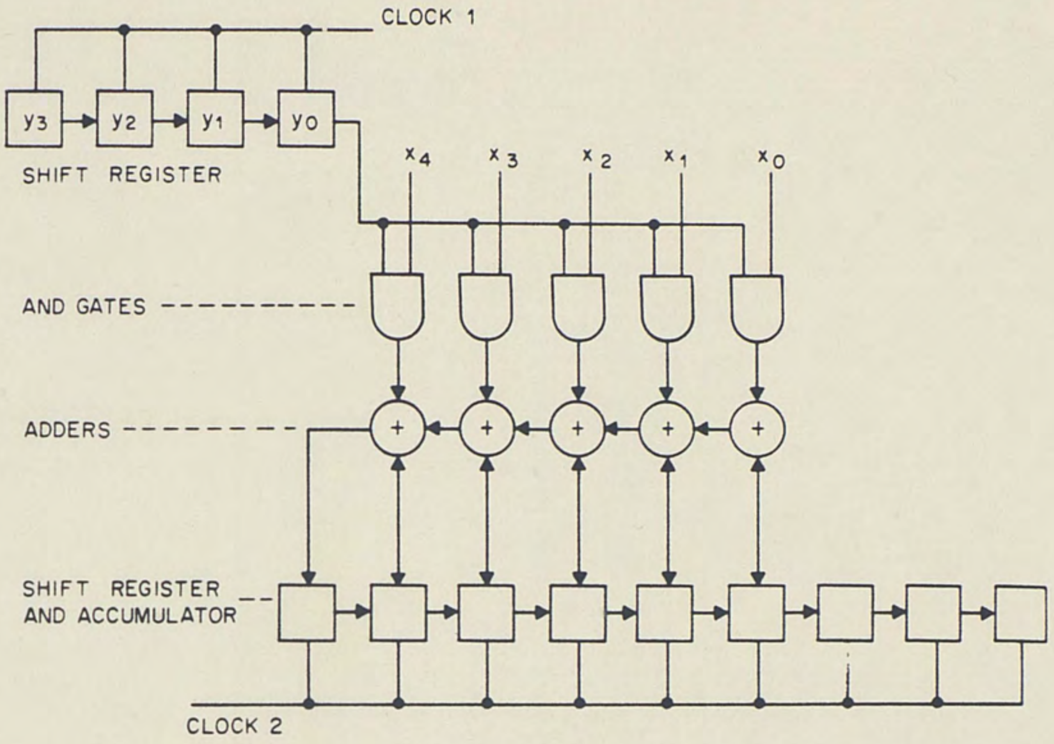


Figure 9. Clocked Add-shift Multiplier (Rabiner-Gold, 1975)

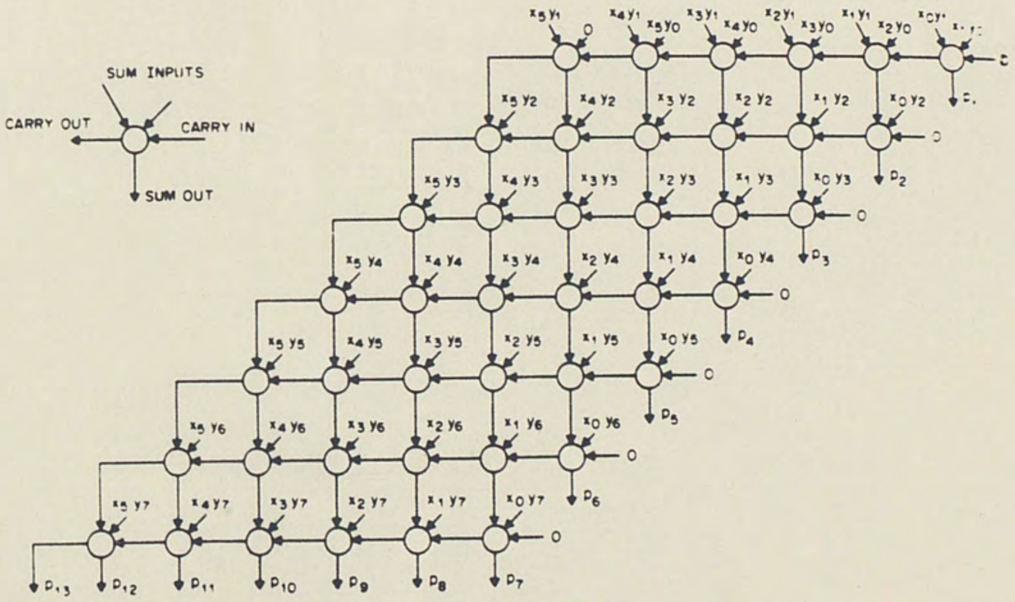


Figure 10. Structure for an Array Multiplier (Rabiner-Gold, 1975)

4. Division

Division is much more difficult to realize than multiplication. The logical nature of division is such that it does not lend itself to speed-up as well as multiplication and, therefore, takes about four times longer than multiplication. The simplest division scheme is subsequent subtraction. First, subtract the divisor from the dividend and increment a counter. Continue as long as the remainder is positive. When the remainder becomes negative, cancel the last step. The counter will contain the quotient and the remainder will be correct.

A more rapid division can be realized by calculating the quotient digits instead of counting them. First, the divisor is subtracted from the most significant part of the dividend. If the remainder is positive the quotient digit is "1"; otherwise the subtraction is cancelled by adding the divisor to the remainder, and the quotient digit will be "0". Now shift the remainder one place to the right and repeat until all the quotient digits have been calculated. Figure 11 shows an example of this binary division.

```
               0.101110
     0.011010)0.010011
               -011010      Divisor larger than dividend, shift
               -------
                001100      Subtract, enter 1
                 011010     Shift divisor, enter 0
                 -011010    Shift divisor
                 -------
                  010110    Subtract, enter 1
                  -011010   Shift divisor
                  -------
                   010010   Subtract, enter 1
                   -011010  Shift divisor
                   -------
                    001010  Subtract, enter 1
                     011010 Shift divisor, enter 0
```

Figure 11. Fixed Point Binary Division
(Hill-Peterson, 1978)

IV. IMPLEMENTATIONS

1. Correlation/Convolution

The correlation between two functions is a measure of their similarity. Digital signal processing requires functions to be represented in discrete form where time scale and amplitude are quantized into discrete steps. Correlation in discrete form is given below.

$$R(n) = \sum_{k=-\infty}^{\infty} v_1(k)\ v_2(n+k)$$

$R(n)$ is the correlation between two signals, $v_1(k)$ and $v_2(k)$, where k and n are digital indexes representing their common independent variable, generally time. It is determined by multiplying one signal $v_1(k)$, by the other signal shifted in time, $v_2(n+k)$, and then taking the sum of the products. Convolution in discrete form is given below.

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)\ x(n-k)$$

Correlation handles the two functions forward in time, while convolution treats one forward $h(k)$ and one backward $x(k)$. The filter output is represented by $y(n)$. Figure 12 shows the basic structure for an all digital correlator. Each bit of an input shift register is compared with a reference shift register through an exclusive-NOR gate. The outputs are then summed. Figure 13 shows a sample output.

There are five steps to convolve two functions using a correlator. First, load register M with function $h(k)$ "backward" as shown in

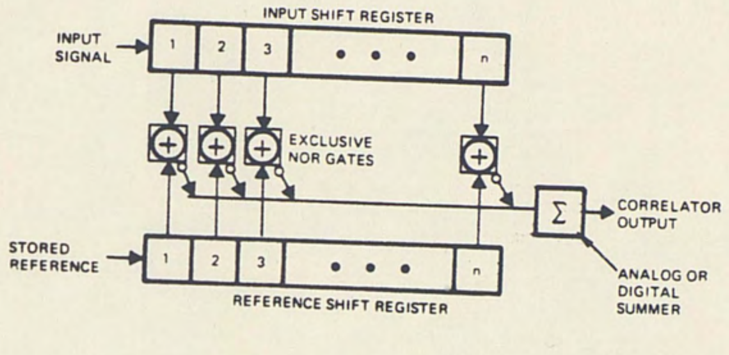


Figure 12. Digital Correlator (TRW, 1981)

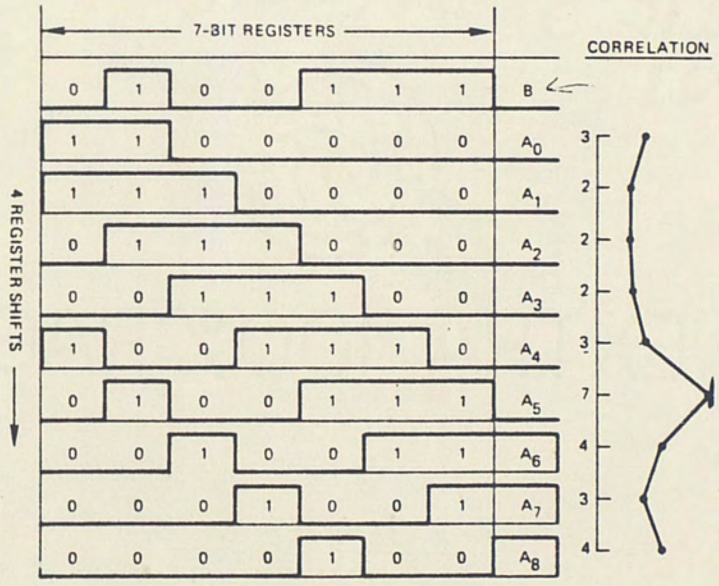


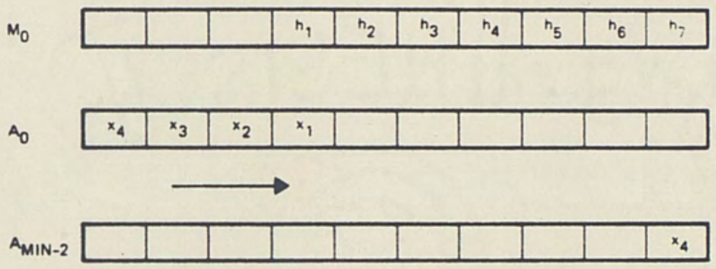
Figure 13. Sample Output of a Digital Correlator (TRW, 1981)

Figure 14. Second, clock function $x(k)$ forward through A until x_1 and h_1 align. Third, compute the first convolution point, $y_1=x_1 \cdot h_1$. Fourth, advance "A" register contents one bit, and compute $y_2=h_1x_2+h_2x_1$. Fifth, continue until the sequence is completed. The TRW TDC1023J has a maximum bit rate of 20 MHz.

2. Fixed Instruction Set General Purpose Microprocessor

Most general purpose microprocessors either have 8 bit or 16 bit word lengths. The microprocessor sacrifices bandwidth and word length for board space and design ease. In digital signal processing, the throughput of the microprocessor system can usually be improved by using a coprocessor for multiplication. Figure 15 shows an 8085-based system using a 74S508 to provide a hardware multiply. To compare throughput, consider the software multiply of Figure 16. Also consider the routine for the two-chip 8085/74S508 combination which is shown in Figure 17. The multiplication results are stored in registers B and E. Worst case execution time drops from 181 μs to 7.5 μs by using the hardware multiply. The 8086 can do a firmware multiply in around 80 μs.

One way to dramatically increase the dynamic range of the microprocessor-based system is to use a floating point coprocessor. The 8232 can provide 64 bits of precision for the 8085/8086 families. The single precision format is shown in Figure 18. Bit 31 is the sign bit where "0" represents positive. Bits 23-30 represent the exponent. Bits 0-22 are the mantissa which represent the fractional magnitude of the number.



Step 1: *Load register* M *with function "*h*" (backward) to state "*M_0*."*

Step 2: *Clock function "*x*" forward through register* A *until* x_1 *and* h_1 *align.*

Step 3: *Compute first convolution point,* $y_1 = x_1 \cdot h_1$.

Step 4: *Advance "*A*" register contents one bit, compute* $y_2 = h_1x_2 + h_2x_1$.

Step 5: *Continue until last step,* x_4h_7 *in this case.*

Figure 14. Convolution Using a Digital Correlator (TRW, 1981)

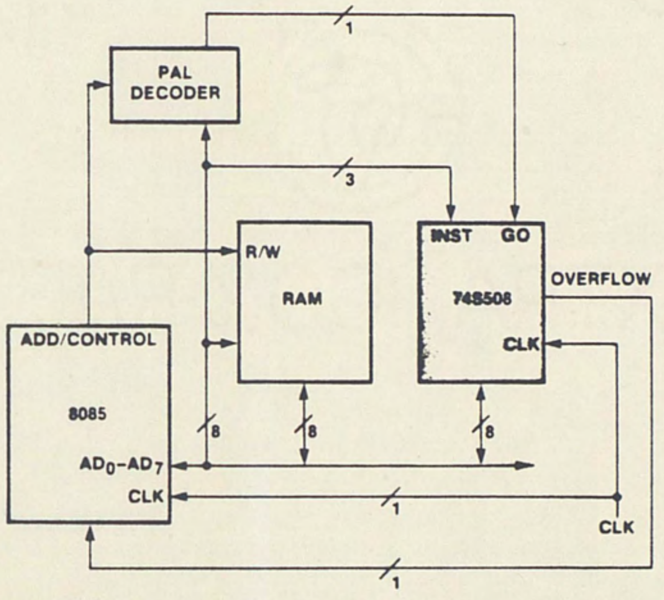


Figure 15. 8085 Microprocessor System Using 74S508 to Provide a Hardware Multiply (Gordon, 1982)

```
MULTY LXI  H, 0  CLEAR H AND L BYTES
                 FOR RESULT
      MVI  D, 0  CLEAR THE HIGH PART
                 OF THE SHIFT REGISTER
      MVI  C, 7  INITIALIZE COUNTER
MULT  MOV  A, B
      RRC        ROTATE B RIGHT, SAVE
                 THE CARRY BIT
      JNC  SFT   JUMP TO SFT IF CARRY=0
      DAD  D     ADD H AND L TO D AND E
SFT   DEC  C     DECREMENT COUNTER
      JZ   EXIT  JUMP TO EXIT IF COUNT-
                 ER=0
      STC
      CMC        CLEAR CARRY
      MOV  A, E
      RAL        ROTATE E LEFT, SAVE
                 CARRY BIT
      MOV  E, A  STORE NEW E
      MOV  A, D
      RAL        ROTATE D LEFT THROUGH
                 THE
                 CARRY BIT
      MOV  D, A  STORE NEW D
      JMP  MULT
EXIT  RET
```

Figure 16. Routine for 8085 Software Multiply (Gordon, 1982)

```
MOV   106, E   LOAD X, INSTRUCTION 6
MOV   100, B   LOAD Y, INSTRUCTION 0
NOP            MULTIPLY
MOV   B, 107   READ AND STORE MSB OR
               RESULT, INST 7
MOV   E, 107   READ AND STORE LSB OR
               RESULT, INST 7
```

Figure 17. Routine for 8085/74S508 Multiply (Gordon, 1982)

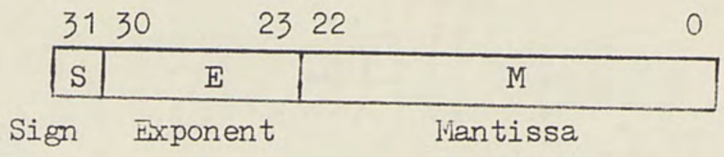


Figure 18. Single Precision Format for an 8232 Floating Point Number

A number of applications utilize the microprocessor for offline signal processing. These applications no longer require the high speed real-time multipliers, adders and assorted processing hardware. The real-time problem now is to sample and store on secondary memory as rapidly as possible. Thus, high speed data handling devices such as direct memory access (DMA) controllers combined with multiple buffering techniques become the primary real-time ingredients.

3. Bit Slice Processor

Figure 19 shows a simple microprogrammed architecture built around the AM2901 bit slice ALU. The instruction lines, addresses, and the data inputs normally will come from registers latched with the same clock as the AM2901. The system is driven through the pipeline register from a PROM containing the stored microprogram. This memory contains microinstructions which apply the proper control signals to both the ALU and other control circuitry. The address lines are generated by the AM2901 microprogram sequencer. The start address plus control signals from the previous instruction aid this device in incrementing an address, jumping to an address, storing an address or linking to subroutines. Note that as one instruction is executed the next instruction is being read from program memory. Any number of 4 bit slice

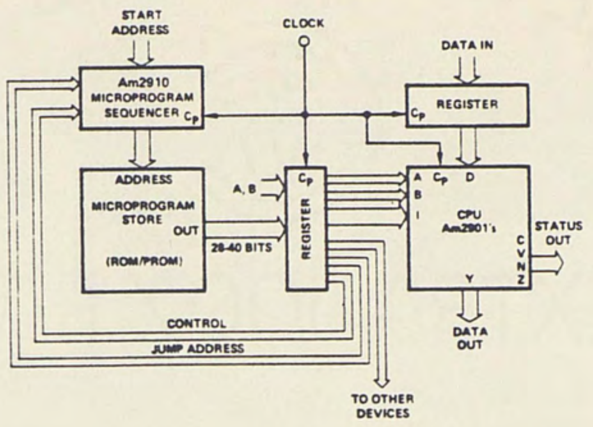
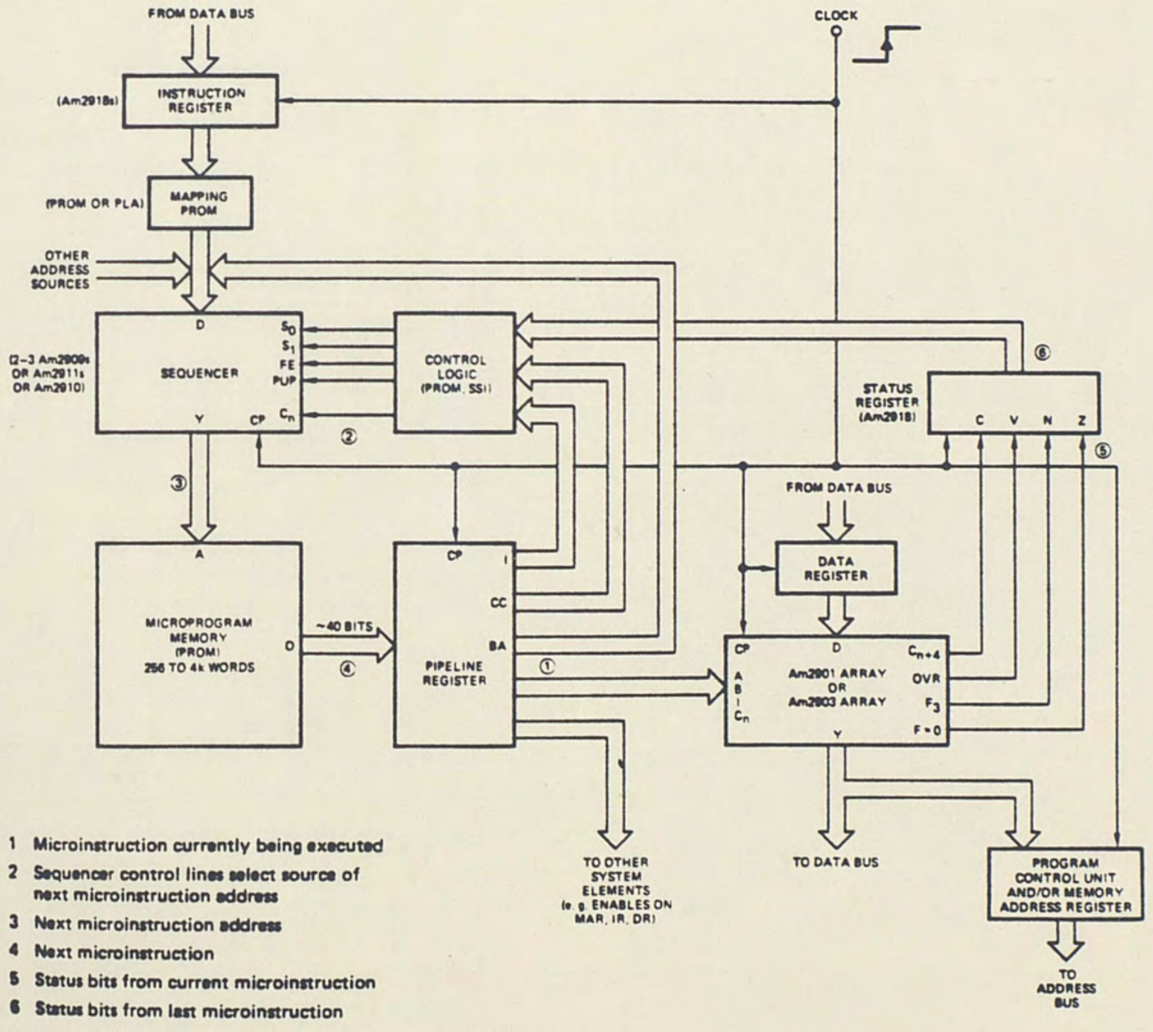


Figure 19. Simple Microprogrammed Architecture (Advanced Micro Devices, 1982)



1 Microinstruction currently being executed
2 Sequencer control lines select source of next microinstruction address
3 Next microinstruction address
4 Next microinstruction
5 Status bits from current microinstruction
6 Status bits from last microinstruction

Figure 20. Pipelining Bit Slice Architecture (Advanced Micro Devices, 1982)

AM2901s can be connected to increase the dynamic range of the CPU. Thus, the CPU can be 4, 8, 12, 16, 32 or any increment of 4 bits wide.

Figure 20 illustrates a more complex pipelining architecture. The same clock pulse loads the instruction register, updates the sequencer, latches the next instruction out of the pipeline register, and transfers the accumulator status to the sequencer. The AM2903 ALU is shown in Figure 21. It contains an instruction decoder, scratch pad RAM, ALU and multiply hardware for a 4 bit slice. The instruction lines I_0-I_8 can be broken down as follows: I_5-I_8 identify the destination, I_1-I_4 identify the ALU function, and I_0 is a multiplexer select. Figure 22 shows the interconnections and microcode algorithm for a 16 bit ripple carry multiply.

The AM29500 family make up an 8 bit slice processor which is specifically targeted for digital signal processing applications. The AM29501 is the processing unit. Its pipelined architecture allows any combination of register instructions, ALU instructions, and I/O instructions that can be microprogrammed to occur in the same cycle. This architecture allows overlap of external multiplication, ALU operations and memory I/O as shown in Figure 23. The AM29516/AM29517 are 16 by 16 bit parallel multipliers. The array multiplier provides a 40 ns multiply time. The family also features a programmable FFT address sequencer-- the AM29540. It provides the RAM and ROM addresses necessary to perform the repetitive butterfly operations of the FFT. The FFT sequencer provides a maximum transform length of 65,536 points.

BLOCK DIAGRAM

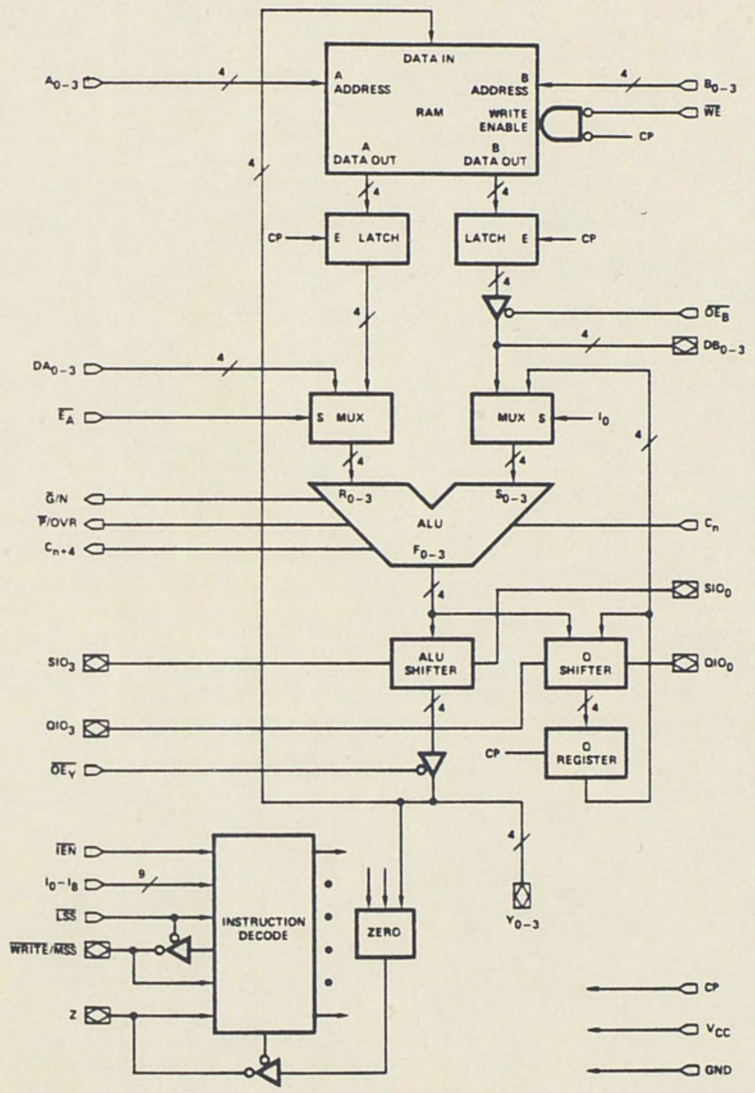
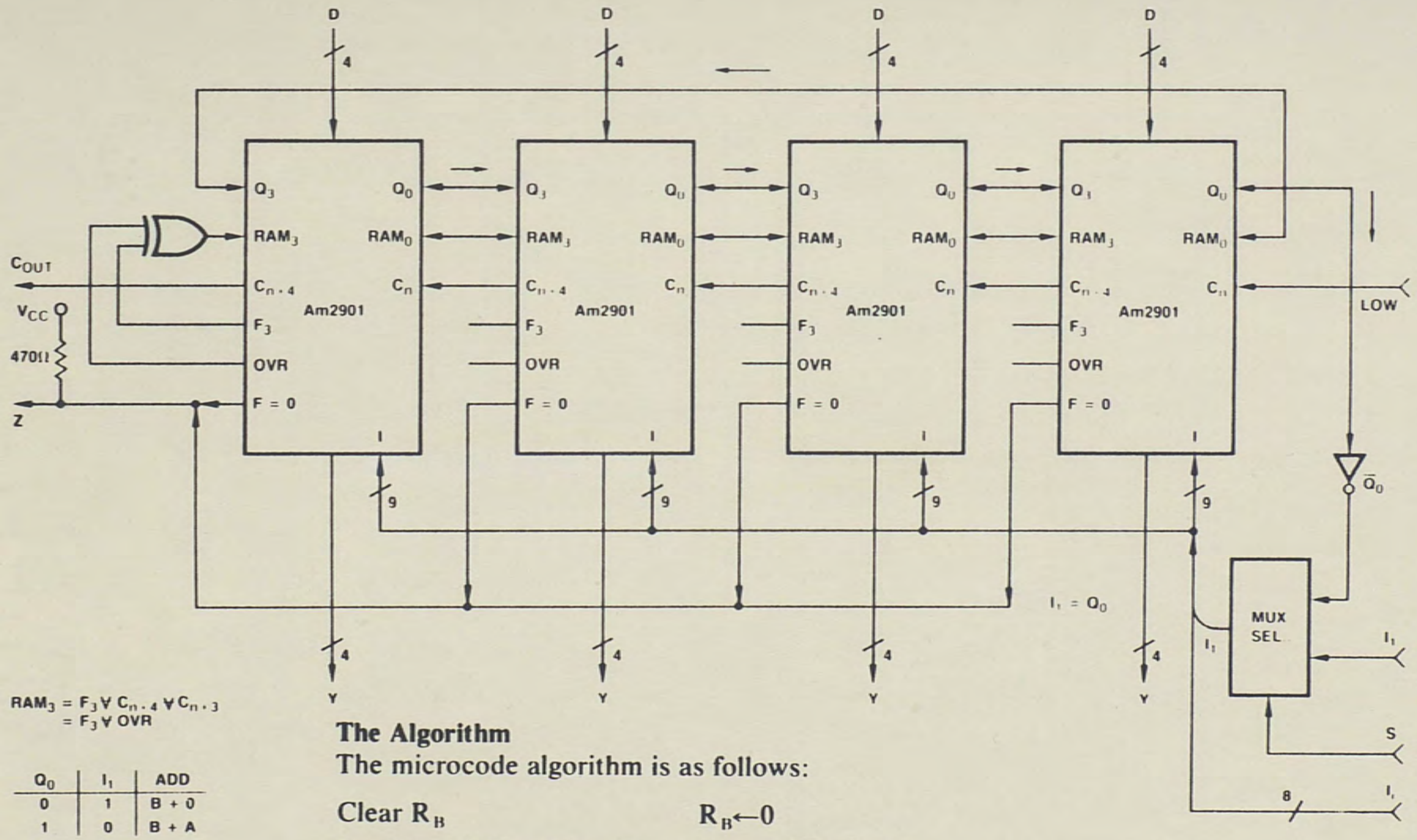


Figure 21. AM2903 Arithmetic Logic Unit (ALU) (Advanced Micro Devices, 1982)



$RAM_3 = F_3 \vee C_{n+4} \vee C_{n+3}$
$= F_3 \vee OVR$

Q_0	I_1	ADD
0	1	B + 0
1	0	B + A

The Algorithm
The microcode algorithm is as follows:

Clear R_B	$R_B \leftarrow 0$
Load multiplicand into R_A	$R_A \leftarrow$multiplicand
Load multiplier into Q	$Q \leftarrow$multiplier (Q_0 can be sensed)
If not done, add, then shift Q down and shift R_B down	If $Q_0 = 0$, $R_B \leftarrow (R_B + 0)/2$: $Q \leftarrow Q/2$ If $Q_0 = 1$, $R_B \leftarrow (R_B + R_A)/2$: $Q \leftarrow Q/2$ Repeat loop until counter = 0
If done, add, then shift Q down and shift R_B down	If $Q_0 = 0$, $R_B \leftarrow R_B/2$: $Q \leftarrow Q/2$ If $Q_0 = 1$, $R_B \leftarrow (R_B - R_A)/2$: $Q \leftarrow Q/2$

Figure 22. Interconnections and Algorithm for the 16 Bit Ripple Carry Multiply (White, 1981)

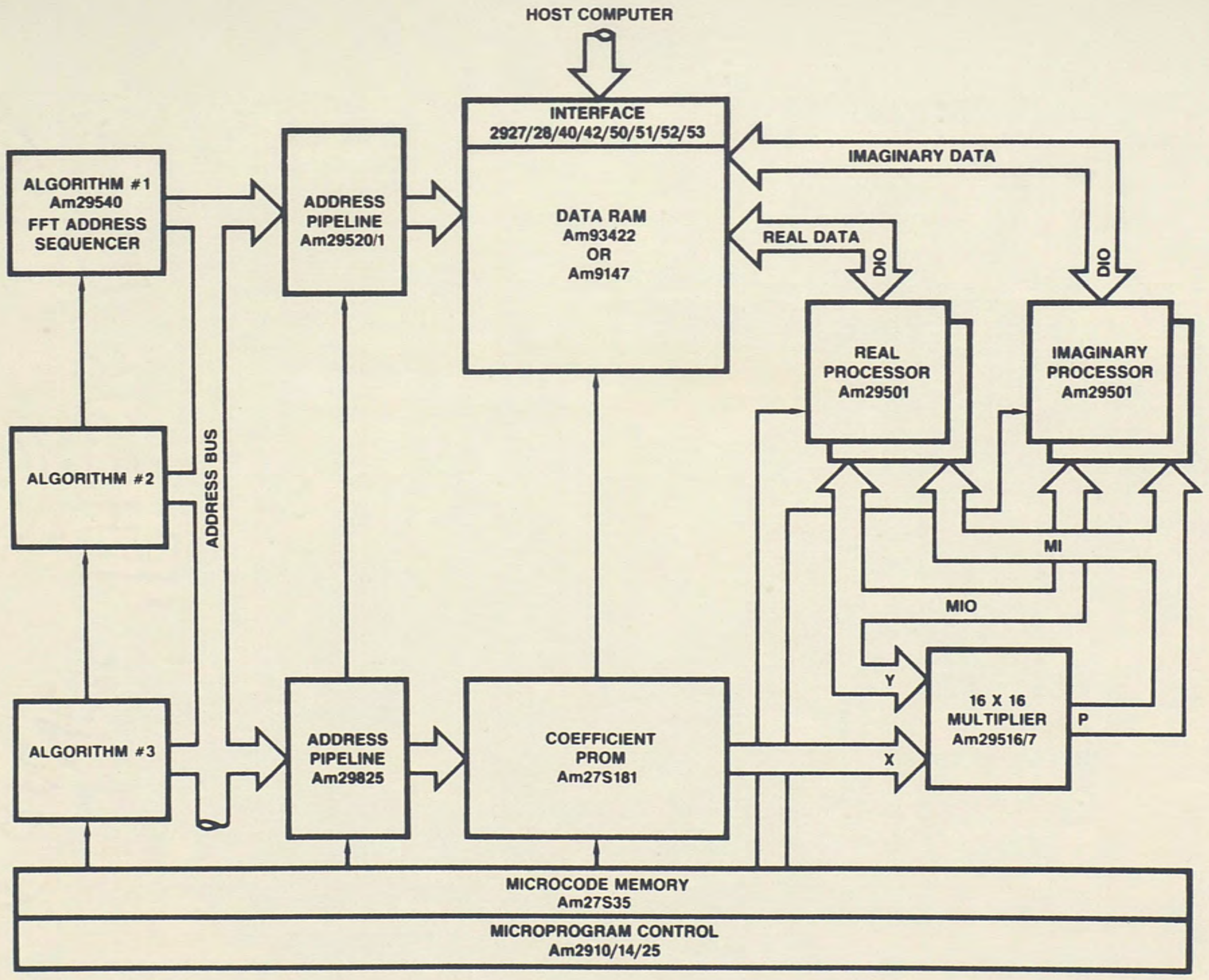

HOST COMPUTER
INTERFACE 2927/28/40/42/50/51/52/53
DATA RAM Am93422 OR Am9147
IMAGINARY DATA
REAL DATA
DIO
DIO
ALGORITHM #1 Am29540 FFT ADDRESS SEQUENCER
ADDRESS PIPELINE Am29520/1
ALGORITHM #2
ADDRESS BUS
REAL PROCESSOR Am29501
IMAGINARY PROCESSOR Am29501
MI
MIO
Y
X
P
16 X 16 MULTIPLIER Am29516/7
ALGORITHM #3
ADDRESS PIPELINE Am29825
COEFFICIENT PROM Am27S181
MICROCODE MEMORY Am27S35
MICROPROGRAM CONTROL Am2910/14/25

Figure 23. Array Processor (Advanced Micro Devices, 1982)

4. Single Chip Signal Processors

The Intel 2920 is a programmable single chip signal processor. It contains an input multiplexer, a sample hold, an analog to digital converter, a scratch pad RAM, an EPROM, an ALU, a digital to analog converter and an output multiplexer on a single chip. The 2920 is designed to replace analog subsystems such as filters, threshold detectors, limiters, oscillators, and waveform generators to name a few. The 2920 has a dynamic range limited to nine bits and a band width limited to 13 KHz. Figure 24 shows a block diagram for the processor.

The TMS320 is a single chip microcomputer aimed at giving both the design ease of a general purpose microprocessor and the high throughput of a multichip design. The TMS320 is based on a modified Harvard architecture which supports separate program and data memory spaces. This architecture allows for the overlap of fetch and execute cycles. The chip also features a 16 X 16 multiply at 200 ns and gives a full 32 bit result. Figure 25 shows a TMS320 FIR filter implementation and corresponding speed of execution.

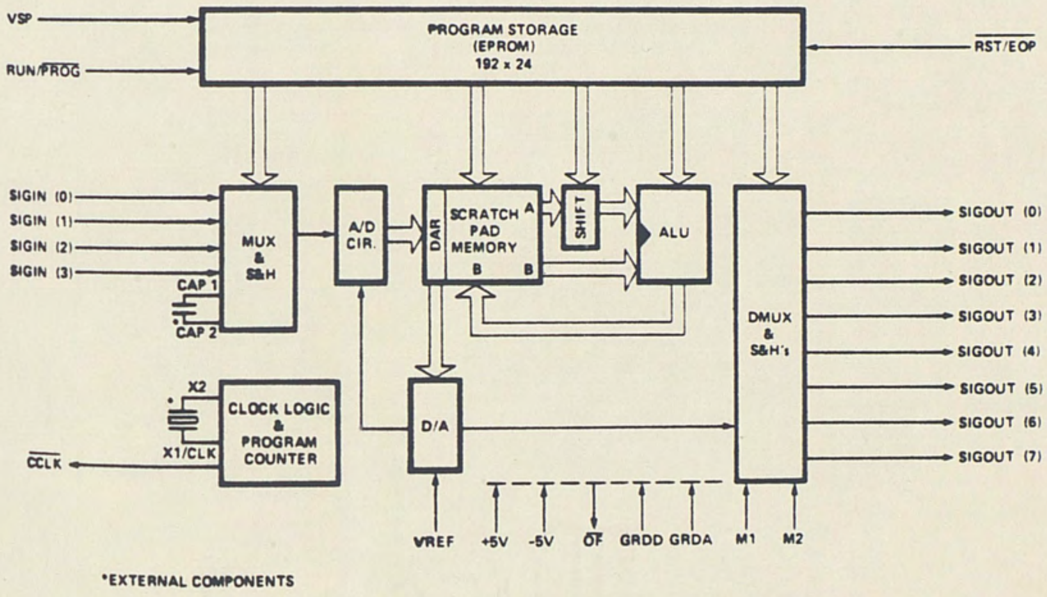


Figure 24. Block Diagram 2920 Signal Processor (Intel Corporation, 1981)

The following equation is used to implement a filter model:

$$y(n) = Ax(n)+Bx(n-1)+Cx(n-2)+Dx(n-3)$$

```
Clock Cycles
    2          START    IN      X1,PA0
    1                   ZAC
    1                   LT      X4
    1                   MPY     D
    1                   LTD     X3
    1                   MPY     C
    1                   LTD     X2
    1                   MPY     B
    1                   LTD     X1
    1                   MPY     A
    1                   APAC
    1                   SACH    Y
    2                   OUT     Y,PA1
    2                   B       START
   ---
   17
```

T_S = sampling period

$$T_S = 17(200\ \text{ns}) = 3.4\ \mu\text{s}$$

$$f_S = \frac{1}{T_S} = 294\ \text{KHz}$$

Figure 25. TMS 320 Implementation for FIR Filter

5. Comparison of Implementations

Figure 26 is a graph showing the complexity vs. throughput for various implementations. The fast Fourier transform ranges from 32 point transform used in instrumentation to the very complex transforms used in pattern recognition. The digital filtering applications range from voice to video. The digital dynamic controllers require both lower bandwidths and less complexity.

Figure 26 also shows the general technology requirements for particular applications. For example, almost all control applications can be handled without requiring a bit slice processor. Also very few digital filtering applications can utilize an 8 bit microprocessor.

The TMS320 filter implementation of Figure 25 has a sampling period of T_S equal to 294 KHz. Thus, if the input analog signal requires at least 20 samples per period then the maximum bandwidth is 14.7 KHz.

$$BW = \frac{1}{20T_S}$$

Table 2 shows the design and cost tradeoffs for the various implementations. The 8 bit processor is the easiest to design and the SSI/MSI discrete logic is the most difficult. The TMS 320 provides an excellent tradeoff of throughput and design time.

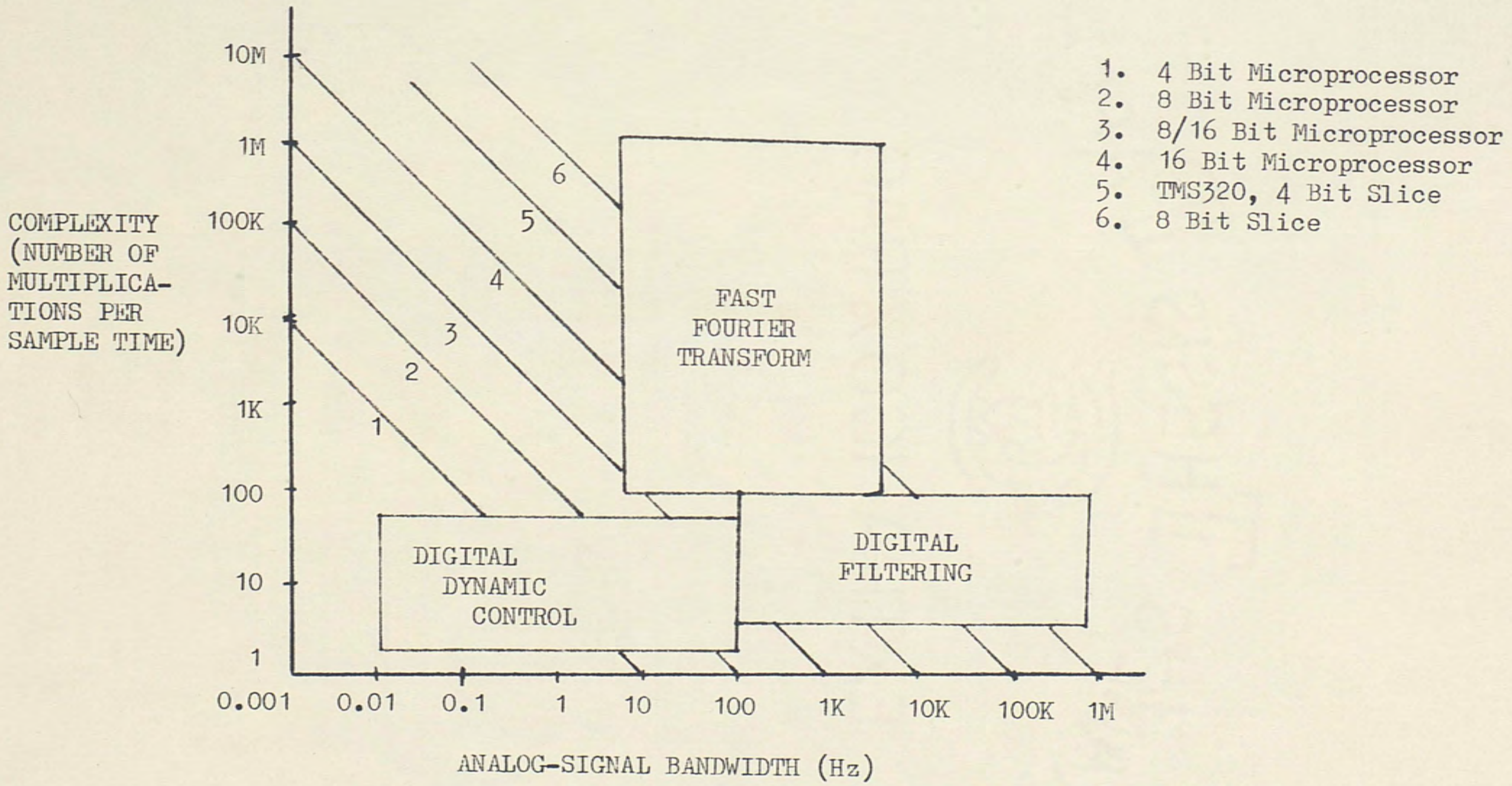


Figure 26. Complexity vs. Throughput

Table 2. Design Cost and Time Considerations for a 10th Order Digital Filter

Implementations	Design Time (Days)	Design Complexity (# Chips)	Electronic Component Cost (Dollars)
8 Bit Microprocessor	4	5	45
16 Bit Microprocessor	7	8	100
INTEL 2920	4	1	100
TMS320	6	6	300
4 Bit Slice	20	35	700
8 Bit Slice	15	20	1000
SSI/MSI	60	150	200

Note: The above figures should only be used in a relative sense. Design time is dependent on the experience of the designer. The cost is dependent upon the environment in which the system is used and parts cost at that time.

X. CONCLUSION

Digital signal processing requires high speed arithmetic and control sequencing capabilities for both digital filtering and spectral analysis. The fundamental hardware elements are memories, adders, multipliers and dividers. Memory can be classified as either static or dynamic. It can also be classified by either writability or accessibility. ALUs can utilize the slower ripple carry adders or the more complex lookahead carry adders. Division tends to be slower and more complex than multiplication.

There is a range of possible solutions available to implement digital signal processing applications. Correlation is a measure of similarity between functions. The digital correlator may be used to convolve two functions to realize a desired filter response. The general purpose microprocessors are desirable in applications requiring limited dynamic range and limited bandwidth. The bit slice processor can attain greater dynamic range and greater speed than the microprocessor-based system. It will usually, however, lead to a more complex circuit design. The present trend is to develop single chip signal processors to replace the more complex multichip bit slice designs.

REFERENCES

Advanced Micro Devices. Array and Digital Signal Processing Products. Sunnyvale, CA: Advanced Micro Devices, 1982.

Cushman, Robert H. "Digitization Is on the Way for FFT Designs." Electronic Design News (August 1981): 99-106.

Eldon, John. Correlation-- A Powerful Technique for Digital Signal Processing. La Jolla, CA: TRW, 1981.

Gordon Ehud. "Speed µP Arithmetic Functions While Using Less Software." Electronics Design News (May 1982): 167-171.

Hill, Fredrick J., and Peterson, Gerald R. Digital Systems-- Hardware Organization and Design. New York: John Wiley, 1978.

Intel Corporation. Intel Component Data Catalog. Santa Clara, CA: Intel Corporation, 1981.

Mick, John R. "Understanding the AM25LS2517 and AM25LS381." AMD Bipolar Microprocessor Logic and Interface Data Book. Sunnyvale, CA: Advanced Micro Devices, 1982.

Monolithic Memories Inc. Monolithic Memories Bipolar LSI Databook. Sunnyvale, CA: Monolithic Memories Inc., 1981.

Motorola Inc. Motorola Memory Data Manual. Austin, TX: Motorola Inc., 1980.

Rabiner, Lawrence R., and Gold, Bernard. Theory and Application of Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1975.

White, Donnamaie E. Bit Slice Design: Controllers and ALUs. New York: Garland STPM Press, 1981.