SURVEY OF ARITHMETIC INTEGRATED CIRCUITS

Shlomo Waser

Monolithic Memories, Inc.
Sunnyvale, California

Abstract

The purpose of this report is to provide the state-of-the-art of high performance arithmetic integrated circuits (ICs). The survey concentrates on arithmetic ICs that are designed to improve execution speed over software techniques, therefore, no calculator chips are surveyed.

In order to understand the difficulties encountered in fabricating high speed arithmetic ICs, we start the article with a discussion on semiconductor technology. Next, we survey the various arithmetic elements that are available in monolithic form: ALUs, Data Slices, Multipliers, Floating Point Processors, and ROMs. Finally, we conclude with a comment on digital signal processing and a discussion of the future trends in arithmetic ICs.

NOTE: The report assumes knowledge of the basic arithmetic algorithms. For those desiring more background information, the following references are recommended: [MAC 61], [WAL 64], [BOO 51], [AND 67].

Technology Background

Three factors limit how much parallel arithmetic can be put into one chip: maximum allowed power dissipation, cost and pin count. In this paper we give only a brief review of these factors, for more details refer to [WAS 78].

Most LSI devices can dissipate a maximum of 1.5 watts without any special cooling. Thus using the popular technology of low-power Schottky TTL (2mW/gate) the maximum number of gates on a single LS/TTL chip is 750 gates (1.5 watt/2mW).

In technologies with low power per gate (MOS, I^2L) the power dissipation is not the limiting factor, but cost is. This is due to the fact that yield is a function of random defects. Thus doubling the number of gates per chips, can increase the cost per chip by a factor of 20. For a more detailed discussion on the cost aspects of producing Integrated Circuits see the excellent article by Noyce [NOY 76].

The state-of-the-art in packages is 64 pin package. This is a severe limitation on high speed arithmetic. For example, single chip 24 bit mantissa multiplication in floating point processors requires a package with at least 98 pins. A good detailed discussion of the pin count problem, and the potential solutions is given by [LYM 77].

Comparing Technologies

The easiest and the most popular way to compare the various technologies is to use gate characteristics, in terms of speed (ns) and power dissipation (mw). A common figure of merit for a technology is the speed-power product of its gate.

However, this popular analysis can be quite misleading for the following reasons: (a) Fan-In is never stated with the above characteristics, but fan-in is a very critical information for arithmetic circuits. Winograd [WIN 67] showed that the lower bound on the speed addition is $\lceil \log_r 2n \rceil$, where r is the fan-in and n is the number of bits in each operand. Thus, for r large enough, all arithmetic operations can be accomplished in one gate delay (analyzing few TTL gates gives a fan-in = 4). (b) Fan-out is not stated either with the above characteristics, but for the popular MOS technology, the gate delay and power dissipation increases substantially with increased fan-out. This explains why the MOS microprocessors still use bipolar devices for Bus interface. (c) to illustrate how misleading the gate characteristics can be we quote from an article in Electronics [PAS 78] the following numbers for H-MOS: gate delay = 1ns, gate power = 1mw. We know that 16 bit ALU (using carry-look-ahead) can be constructed using 8 gate delays; i.e., 8ns. But the fact is that the newest 16 bit microprocessor from Intel can cycle only at 125ns... (d) For speed-power figures to be meaningful they need to be specified along with the amount of integration involved. For example, SSI in the 100K ECL has speed-power product of 30pj, whereas MSI in this family is characterized by 5pj.

Given all the above cautions the reader is now referred to Table 1 which shows only approximate gate characteristics for each technology. To make this table more complete and meaningful, the most complex arithmetic IC available in each technology is also listed. These technologies can be divided into two groups:

a. High speed and high fan-out technologies: ECL, TTL. These technologies are used to implement building blocks for high speed arithmetic processors.

CH1412-6/78/0000-0257$00.75 (c) 1978 IEEE

b. High density technologies: MOS, I^2L. These are just beginning to be used in implementing single chip arithmetic processors.

Technology		Year Introduced	Approximate Gate Characteristics for LSI (Assuming fan-in = fan-out = 4) Delay (ns)	Power (mW)	Speed Power Product (pj)	Density Gates per mm^2	Most Complex Arithmetic IC Available
ECL	100K	1973	0.5	10.0	5	30	Fairchild: F100181 4 bit binary/BCD ALU Addition takes 6ns
	10K	1971	2.0	25.0	50	30	Motorola: MC 10800 4 bit ALU Slice Addition takes 19ns
TTL	EFL	1965	5.0				TRW: MPY-24 24x24 Multiplier (200ns)
	S/TTL	1970	3.0	20.0	60	30	TI: 74S181 4 bit ALU Addition takes 11ns
	LS/TTL	1972	7.0	2.0	14	30	MMI: 67516 16x16 Multiplier/Divider Multiplication takes 800ns
I^2L		1975	10.0	0.1	1	300	TI: 9900 16 bit Microprocessor Fastest operation takes 4.6us
N-MOS		1973	10.0	1.0	10	170	AMD: 9511 32-bit Floating Point Processor Fastest operation takes 4us

TABLE 1: Comparison of the Technologies used in Implementing Arithmetic ICs.

ARITHMETIC ELEMENTS

This section describes major arithmetic building blocks in Integrated Circuit form: Adders (ALU's), Multipliers, Floating Point Processors, Combinatorial Shifters, and ROM Look Up Tables.

Adders and ALU's

a. Combinatorial ALU's

In the late fifties and early sixties many papers have been written on speeding-up addition. These papers described both variable time techniques (asynchronous) and fixed time techniques (synchronous). Since most computers are synchronous, the fixed time were considered more attractive. Of the fixed time adders, the two major algorithms were conditional sum invented by Sklansky [SKL 60] and the carry-look-ahead which was described by Weinberger and Smith [WEI 56]. But, of these two techniques, only the carry-look-ahead was implemented as an integrated circuit. However, instead of having just an adder the IC manufacturer incorporated ALU function into the same chip. ALU units are capable of add, subtract, shift, and logic operations (AND, OR, EXOR, etc.). The most popular ALU device is the 74S181 [TI 76] implemented in S/TTL technology which is used in minicomputers such as the DEC PDP-11 and the Data General Nova. This device performs addition using a carry-look-ahead algorithm across four bits at a time. When operating on wider words, a companion device (74S182) receives the generate and propogate terms from a group of four 74S181's. In general, the number of levels of carry-look-ahead is $\log_4 n$, e.g., adding 64 bits with full carry-look-ahead takes 15ns in ECL and 28ns in S/TTL.

The 74S181, and the ECL 10181 [MOT 74], are combinatorial devices, and accumulation of results requires an additional register (accumulator). The 74S281 is an ALU with an accumulator on one chip, which still uses the 74S182 for carry-look-ahead. Adding and storing 64 bit operands take 42ns.

Table 2 lists the speed and power of the popular ALUs. As can be expected, the power is approximately proportional to the gate count of the device. The first three devices of the table are the basic ALU elements. The last two devices

are support chips for the ALU, which provide carry-look-ahead when cascading several ALUs.

PART #	FUNCTION	GATE COUNT	SPEED (ns)	POWER (mW)
74S181	4 Bit ALU	75	11	600
10181	4 Bit ALU	75	7	600
74S281	4 Bit ALU & ACC	100	22	700
74S182	4 Groups CLA	20	7	350
10179	4 Groups CLA	20	4	300

TABLE 2: Comparison of various adders (ALUs). The 74S181 is a popular S/TTL ALU, the 10181 is a similar ALU implemented in ECL. Both of these ALUs have corresponding carry-look-ahead 74S182 and 10179. The 74S281 contain on-chip accumulator but it is slower then the combinatorial ALUs.

b. Bit-slice Processor Elements

The architecture of the bit-slice is similar to the classic ALU, but with multiple accumulators (registers) and a control over the ALU sources and destination. Figure 1 shows the architecture of the AMD-2901 [AMD 76], which is probably the most popular bit-slice processor element. Other such 4-bit slices are MOT10800, MMI 6701, TI 74S481, and TI SBP400.

To determine the speed of these devices, the register to register operation (RA + RB→RB) is examined. The 2901A, which is a faster version of the 2901, performs such an operation in 90ns. In order to compare this speed with the speed of the 74181 type ALU, it is necessary to add registers to the 74181.

Expanding the bit slices to handle more than 4 bits is similar to expanding the 74181 ALUs. In fact, most of the bit-slice vendors recommend using the same carry-look-ahead unit (74S182). To get a more realistic picture of the actual throughput of these devices, it is necessary to assume some overall system of architecture. Figure 2 shows the architecture for a 16-bit system. It is assumed to be microprogrammable with a pipelined microinstruction register, i.e., maximum speed is achieved when no branch decisions are made. Figure 2 also contains a comparison of 16-bit throughput for various building blocks. In computing the addition time using combinatorial ALUs (74S181 and 10181), the 2901 architecture was emulated using ALUs, registers and multiplexors.

Figure 2 (b) shows that the speed of the arithmetic processor elements is about half that of the older ALU speed. This speed ratio applies to both the TTL and ECL technologies. The slower speed of these elements is probably the main reason that many new minicomputers are still using the older combinatorial ALUs.

The pattern of increased integration at the expense of speed, is continuing with the recent introduction of the 2903 by AMD. The 2903 is similar to the 2901, but it has some extra features to simplify arithmetic operations; for example both 2901 and 2903 [AMD 78a] perform multiplication using conditional add and shift algorithm. In the 2901 extra hardware is needed to maintain the sign of partial product during the down shift, but in the 2903 this hardware is incorporated into the chip. Similarly, the 2903 has the extra hardware to implement divide and normalize instructions. However, the added hardware makes the 2903 slower than the 2901. The TI 74S481 [TI 77] is a 4-bit slice similar to the 2903 in its arithmetic capabilities, but, the main difference between them is in their architecture. The 74S481 has only accumulators, while the 2903 has an expandable register file.

Monolithic Multipliers

Multipliers use a variety of algorithms and configurations. These can be categorized into two main classes; parallel and serial.

a. Parallel multipliers

Stenzel [STE 77] indicates that the schemes for parallel multiplication are divisible into two groups - iterative array of cells (AMD 25S05), and generation of matrix of partial product terms (MMI 67558) with subsequent reduction of the matrix by means of psuedo adders. Array scheme, like the AMD 25S05, uses only one IC type to implement any size multiplication, and they are relatively compact solution, but their speed increases linearly with operand length. Matrix generation schemes (like MMI 67558) are much faster for larger operands since their speed of operation increases with log of the operand length.

The AMD 25S05 [AMD 77] is an iterative cell that implements X·Y+K, where K is an input fed by the partial product from an earlier stage. In the early and mid 70's this MSI device (which performs only 2x4 multiplication) has been the standard building block for high speed digital signal processing, where the resultant large number of chips was not an objection. The device itself uses the modified Booth algorithm (in combinatorial fashion). A single 25S05 performs 2x4 multiplication at 25ns, but 16x16 multiplication requires 32 devices, and the associated delay is 150ns. Motorola has a similar device (10183) implemented in ECL, where 16x16 multiplication is completed in 100ns.

The MMI 67558 [MMI 78] is an 8x8 multiplier that belongs to the class of matrix reduction scheme. Internally the 67558 uses the modified Booth algorithm to generate five partial products, which are reduced to two operands using carry-save-adders. The resultant two operands are added using carry-look-ahead scheme. The simple black box description of this combinatorial multiplier is shown in figure 3. The device is housed in 40 pin package and its delay is 100ns. The input data to the multiplier is interpreted (according to a mode control lines) as signed two's complement, or unsigned representation. The dual data represen-

tation simplifies expansion of the multiplier to a larger signed multiplication, for example, in multiplying 16 bits operands using 8-bit multipliers, the operand is broken into most significant signed portion, and least significant unsigned portion.

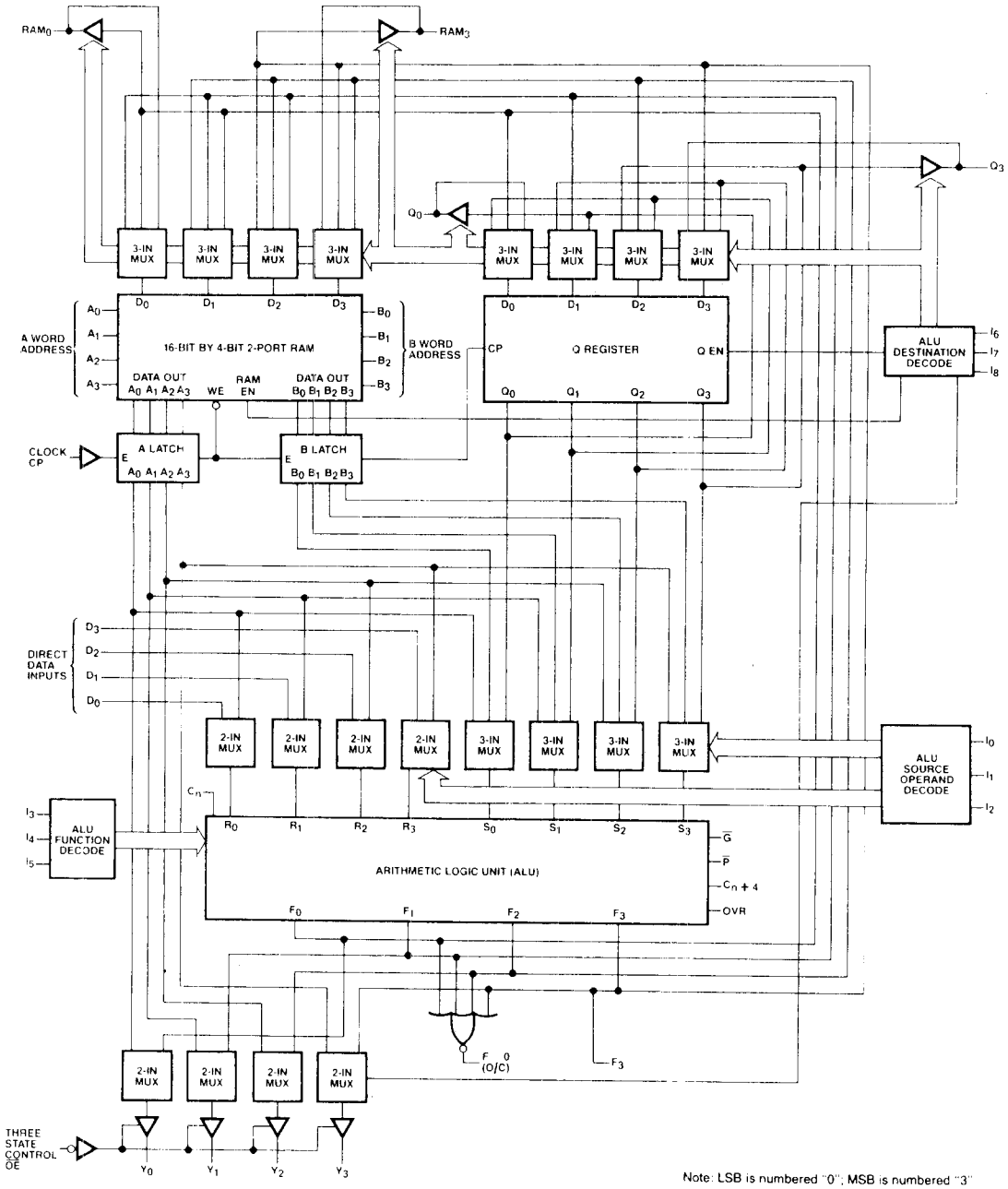

Figure 1

Detailed 2901 Microprocessor block diagram: The main element of the 2901 is the ALU, which performs 8 operations according to 3 instruction lines. The source and destination of the operands and results are determined by the remaining 6 instruction lines. The register file, on the upper left corner, is a 16x4 dual port RAM which is used as 16 registers or accumulators for the ALU. External input and output data bus are also used as source and destination respectively.

(b)

ECL		S/TTL	LS/TTL	I^2L
10181	10800	74S181	2901A	SBP 400
32 ns	62 ns	68 ns	143 ns	300 ns

(a)

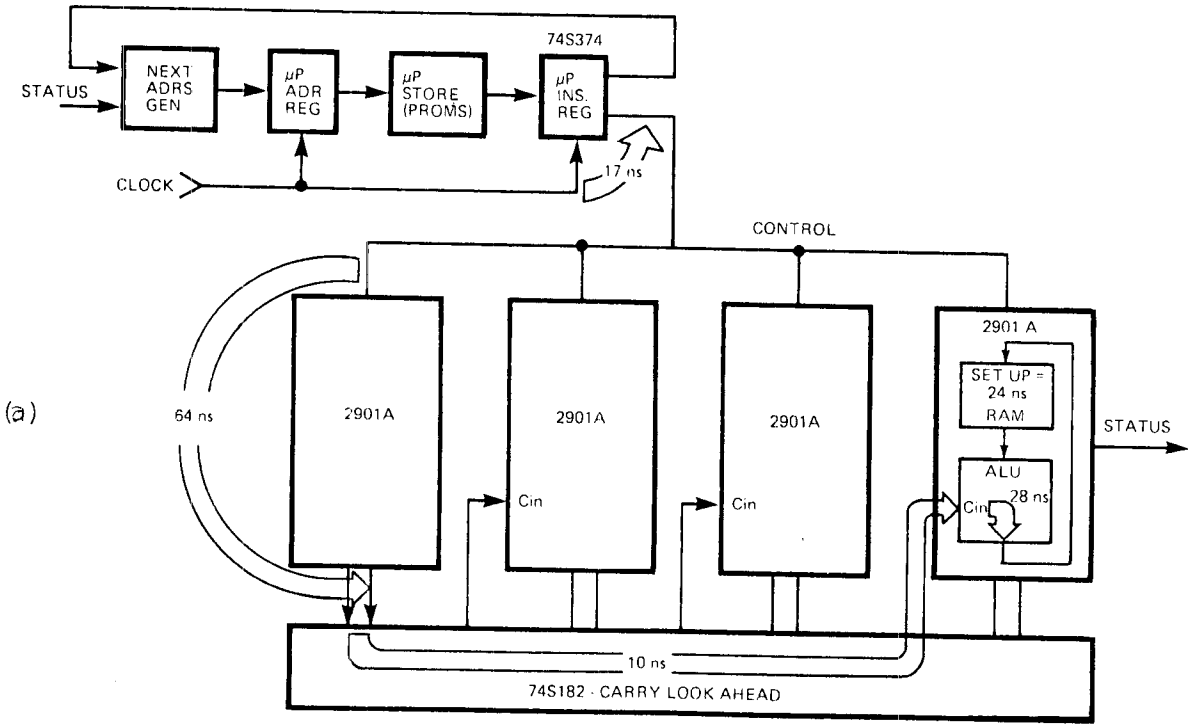

Figure 2

Architecture and performance of a microprogrammable system.

(a) Microprogrammable system (16 bits) made of 2901's. For simplicity the data bus is now shown.

(b) Comparison of min time to perform RA + RB⟶RB, for 16 bit systems made of different building blocks.

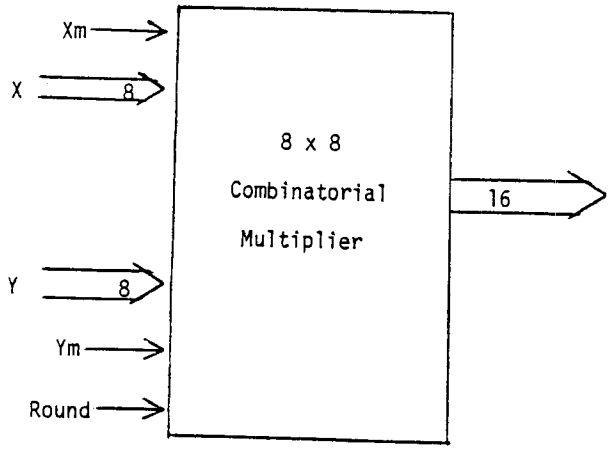

Figure 3: Block Diagram of the 67558

So far, we have discussed expandable multipliers, however, TRW manufactures a series of non-expandable multipliers that are intended specifically for digital signal processing. In such applications pipelining is an acceptable solution to obtain more bandwidth, therefore, the TRW multipliers have input and output registers, which can be clocked simultaneously. TRW solved the non-expandability problem by offering a series of multipliers at 8, 12, 16 and 24 bits. These multipliers are capable of operating at 100 to 200ns pipeline rate. Recently TRW has extended its multiplier family to perform also accumulation. Figure 4 shows the block diagram for a 12x12 multiplier accumulator [TRW 78] . Note, that the accumulator has 27 bits, while only 24 bits are required for double length product. These extra bits enable accumulation of at least 8 products before overflow can occur. The multiplier/accumulator which performs sum of products is an important building block for digital filters, and FFT systems.

The internal logic design of the TRW multiplier array is very straight forward. Partial products are generated by AND gates, and then they are added using carry-save-adders. This logic design does not provide the best possible speed, but it has the improtant advantage of being very cellular, which shortens the LSI design cycle and cuts initial design costs.

The TRW 16x16 multiplier [TRW 77] had to solve two problems - pin limitation and power dissipation. The pin count problem arises because 16x16 multiplier require more than 64 pins, but TRW wanted to use the standard 64 pin package, therefore, they had to multiplex the least significant portion of the product with one of the operands. This is quite acceptable since many times (in signal processing) a single length product is sufficient. The excessive power dissipation (5 watts) was solved by mounting a heat sink on the package.

b. Serial-parallel multipliers

In this section we describe three devices that multiply by sequentially computing and accumulating partial products. Obviously, these devices are slower than the parallel multiplier, however, more integration is possible for same silicon real estate. For example, 100ns 8x8 multiplier has about the same chip size as a 1usec. 16x16 multiplier/divider.

The 25LS14 from AMD [AMD 77] was the first available serial-multiplier. This 8x1 multiplier has one serial input, and 8 parallel inputs. The most attractive feature of this device is its easy expandability. Figure 5 shows how to expand it to do 24x24 multiplication. In general, to do nxn multiplication requires only n/8 multipliers, but the number of clock cycles increases to 2n. Thus, if larger multiplication is required and speed is not critical, this expandable approach is the best available. The author was told about encryption application that requires 400x400 multiplication; using fifty 25LS14's it was

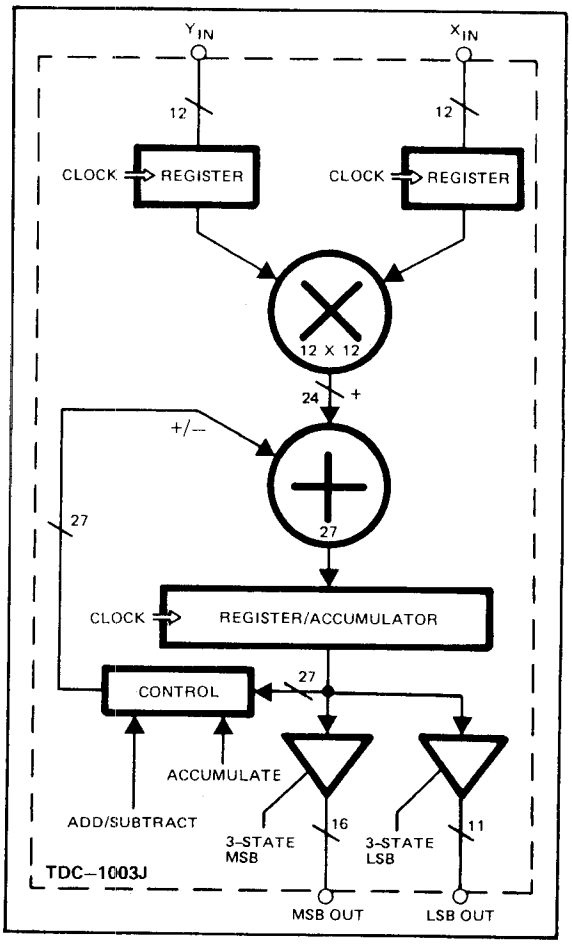

Figure 4: 12x12 multiplier-accumulator

accomplished in 32us. Another application which lends itself to a serial parallel multiplier is in single-line communication, where incoming information has to be processed by digital filter. In this case the serial data is fed into the serial port and the known coefficient is loaded into the parallel port.

The 25LS2516 [AMD 77] is an 8x8 expandable multiplier, which requires a 40 pin package to accomodate all the expansion interconnection. Its structure is an integration of three chips; a) 8x1 serial multiplier, b) a parallel-in serial out shift register, and c) serial adder. Its overall performance is similar to that obtained with the 25LS14.

The 67516 is a non-expandable 16x16 multiplier/divider [MMI 78] . A similar 8 bit version is available as 67508. The expandability of device was sacrificed for the sake of smaller package (24 pin), and more capabilities (e.g. divide). The logic design can be internally expanded to 24 and even 32 bits, but presently the limitation is power dissipation. Figure 6 shows the block diagram of the 67516, which uses the modified Booth algorithm to perform 16 bit multiplication

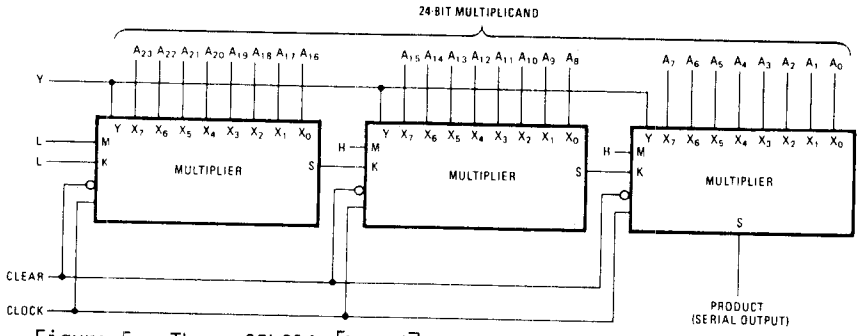

Figure 5: Three 25LS14 [AMD 77] cascaded to make 1x24 multiplier.

in 8 clock cycles (100ns each). The 67516 can execute 23 different micro instructions; examples are: multiply by a constant, multiply and accumulate, divide, etc. Division is accomplished using non-restoring algorithm, which takes 20 clock cycles (2us total).

Floating Point Processors

IC implementation of floating point processor can be either by a single dedicated chip, or by a chip set. The first approach has the convenience of integration at the expense of format flexibility and performance. The multichip implementation can obtain more speed, more performance and format flexibility. The need for a flexible format is due to the great variety of floating point formats coupled with unsuccessful attempts to standardize on a format. Currently, there is an attempt by IEEE standard committee to establish such a standard. If a standard is adopted it is likely that many semiconductor manufacturers will supply IC's to implement this standard floating point format in a single chip.

a. Single Chip Floating Point Processor

Currently the 9511 [AMD 78b] is the only available single chip floating point processor, excluding slow calculator chips. Unfortunately, its format is different from the already established large number of existing floating point format. The 32 bit format consists of (from left to right) a Manitssa sign, 7 bits two's complement exponent and 24 bits magnitude of the Manitssa. The use of two's complement for the exponent rather than using excess code, is quite unusual. The lack of a longer format for more precision is a severe limitation, and so is the restricted dynamic range (= 10^{-20} to 10^{20}). In spite of its limitation the 9511 is as an important milestone in the evolution of arithmetic ICs. The device was designed to operate with the 8080 microprocessor, therefore, it has 8 bit bidirectural data bus, and control lines that are unique to the 8080. In addition to the four basic operations, the 9511 can implement transcendental function (sin, log, etc.), and it can also perform conversion from floating point to fixed point and vice versa. It can also perform operations in 16 or 32 bit fixed point two's complement integer representation.

The speed of the 9511 is a function of the executed instruction, and in some instruction it is also data dependent. The fastest instruction is 16 bit fixed point addition which takes 4 microsec., the slowest instruction is arc-tangent which takes almost 2 milisec., 32 bit floating point addition takes 14-87 microsec. These instruction execution times are about twenty times slower than those of a state of the art minicomputer with floating point processor option. For example, the PDP11/60 executes similar floating point multiply in 1.5 microse., but, of course, the 9511 is a single chip a 24 pin package, and the PDP 11/60 uses more than hundred chips to get its performance.

The internal data paths and the internal ALU are 16 bits wide, even though some operations are on 32 bits operands (this fact indicates that extending the 9511 to handle double precision format could have been implemented internally quite easily). The 9511 has an internal stack for the operands and results which simplifies the arithmetic of long expressions. The transcendental functions are evaluated using Chebyshev Polynomials which provide an even distribution of errors within the selected data representation.

b. Building Blocks for Floating Point Processors

The single chip floating point processor is not useful to many computer manufacturers mostly because of its format and its speed. And it is quite likely that there will never be a single chip capable of handling both IBM floating format (hexidecimal base) and PDP 11 format (binery base and hidden bit). Therefore, the next best thing is to identify building blocks that are common to most floating point processors. A secondary benefit from the building block approach is the ability to trade-off speed vs. chip count (cost); e.g., if an 8x8 multiplier is such a building block then one could implement high speed parallel 56x56 Mantissa multiplication by 49 multipliers in 500ns, or he could build only 8x56 multiplier (from seven 8x8 multiplier) and iterate on it in 1.4 microsec.

We can identify four major building blocks; adders (ALU), multipliers, leading zeros (ones) detection and combinatorial shifters. The adders (ALUs) and multipliers were discussed earlier. The leading zero detector is obviously a simple priority encoder such as TI 74LS348,

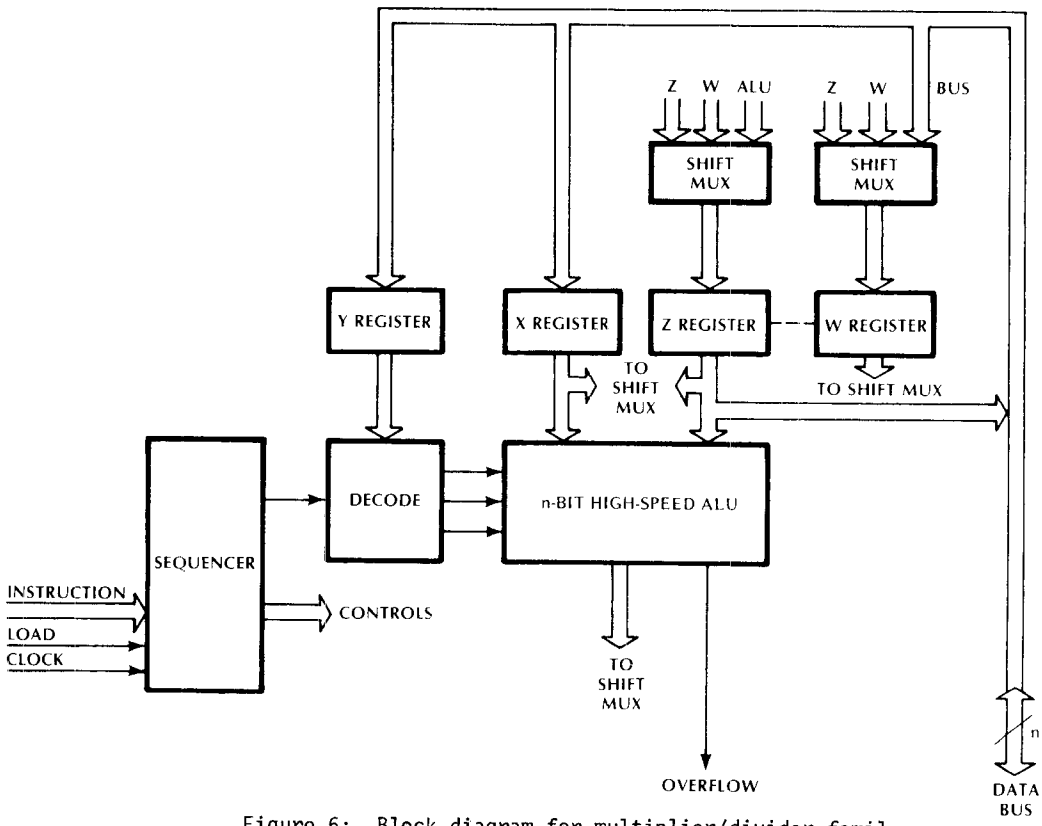

Figure 6: Block diagram for multiplier/divider family
For the 67516 n = 16
For the 67508 n = 8

which can detect up to eight leading zeros in a single IC. To detect leading zeros in a long format (56 bits) nine such ICs are required, with a combinatorial delay of 70ns. Of course, some additional circuitry is required to perform leading ones detection.

Bidirectional combinatorial shifters are used for two purposes: fast alignment in floating point addition/subtraction (right shift based on exponent difference). Fast normalization of final result in any operation (left shift based on the leading zeros/ones detector).

The most widely used TTL combinatorial shifter is the 25S10 [AMD 77], which handles only 4 bits, thus, quite a few chips are required to operate across 56 bits. The only available LSI shifter is the 10808 from Motorola. This shifter is implemented in ECL and it handles 16 bits in a single 48 pin IC. The chip has eight different shift types such as: arithmetic shift, logic shift, rotate, etc. Two different arrays can be built. One array requires only two package delay for a shifter requiring up to 256 bits. The other array requires only one package delay but more packages. A 64 bit shifter requires 10 devices with two package delays, or 16 devices with one package delay (12ns).

ROMs - As Arithmetic Elements

As ROMs are increasing in density they become more and more attractive as arithmetic look-up-table. The last decade has seen the ROM density increase from 256 bit in 1968 to 64K in 1978. Presently, the 64K MOS ROM at 250ns is available from Mostek, AMI and other MOS vendors. The current trend of increased density will make one mega bit ROM a reality by the mid 80's. To see the potential in such a ROM let us review the present arithmetic application of ROMs.

Trigonometric look-up-table in ROM is a very attractive application, where speed is essential. For example, the MMI 6086 is a 100ns 1024x10 look-up-table for storing the sine values for 0^{o} to 90^{o}. The input angle resolution is 0.0879^{o} (90/1024). The values of larger angles can be readily derived using this basic ROM as well as for cosine and other trigonometric functions [MMI 78].

ROM can also be a direct look-up-table for multiplication, the TI 74S274 is a 4x4 multiplier implemented in a 256x8 ROM. However, if 8x8 multiplication were to be implemented in ROM, it would require a 64Kx16 bit ROM which is highly inefficient and impractical, when compared to a dedicated multiplier.

ROMs can be used as look-up tables for logorithms, which in turn can be used to evaluate multiplication. Brubaker and Becker [BRU 76] show that for a given acceptable error, it is possible to reduce considerably the number of the required ROM bits. Their discussion is based on multiplication using logarithms, which is described by the following relationship: XY - antilog (log X + log Y). By expressing the multiplicand and the multiplier in a type of floating-point format, they are able to reduce the number of bits by more than an order of magnitude. For example, for an error of 0.5 percent on 8x8 multiplication, only 6144 bits are required compared with 114,688 using direct multiplication. Their paper gives design curves and selection tables for various multiplication lengths. However, using the logarithmic multiplication requires two ROM access times and addition time.

It was mentioned previously that one of the schemes for parallel multipliers is a generation of matrix of partial products with subsequent reduction of the matrix by means of psuedo-adders. Stenzel et al in their excellent paper [STE 77] illustrate the use of ROMs (or PROMs) to reduce n partial products to two operands. They show how to select efficiently the proper size ROM based on the number of the partial products. For example, 1Kx4 ROM may be programmed to treat the 10 address lines as two adjacent columns of 5 bits each and perform a table look-up on the sum. Note that the maximum sum is 15 which requires exactly 4 bits for its binary representation. A similar approach is taken by the 74S275 [TI 76], which is advertised by TI as "7 bit Wallace-Tree Slice" but in reality it is - simply a programmed 512x4 ROM...

Another arithmetic ROM from TI is the 74185, a 32x8 ROM programmed to convert a 6 bit binary to BCD in 40ns. The 74184 is a similar ROM programmed to convert BCD to binary. Both of these ROMs are not too attractive since so many of them are needed to perform conversion on useful data-paths; e.g., to convert 16 binary to BCD takes sixteen ROMs and the speed is 320ns [TI 76].

Residue number system has been known for the last two decades, but no commercial hardware has been implemented so far. This situation, according to a recent article [JUL 78], may be changing now with the availability of large and fast ROMs. This excellent article illustrates the implementation of a 55ns 16x16 residue multiplication with a package count of seven 8K ROM, and the implementation of a second order digital filter. It also illustrates the implementation of the difficult scaling and division using ROMs in residue number system (addition, subtraction and multiplication are relatively easy to implement in such a number system).

Other arithmetic applications for ROMs include reciprocal tables (useful to start divide of the IBM 360/91 style AND 67), table to decode rounding decisions, FFT coefficient storage and many others.

DIGITAL SIGNAL PROCESSING - as a driving force for arithmetic ICs

Many of the arithmetic ICs described in this report were originated by the need in digital signal processing for fast computations. These observations bring up the question: why doesn't the classic computer industry push for devices more ideally suited for their arithmetic units (e.g. high speed floating-point processors)? The main reason is cost. Minicomputer manufacturers who sell hardware for about $10,000 prefer to implement multiplication in microprograms that have almost zero incremental cost, Microprogramming implementation has a lower throughput, but this is acceptable to most users since in a typical instruction mix, only 6% are multiply/divide instructions. In contrast to the relatively low price of the minicomputer, some end applications in digital signal processing are sold at higher cost. For example, a tomographic scanner, which is used in hospitals to obtain a cross-sectional body picture of a patient, costs about half-million dollars. This scanner generates a digitized image on a CRT by analyzing x-ray results using equations that require as many multiplications as additions. Thus, multiplication implemented in a microprogram is unacceptable, since the picture generation time (for 16 bits) will be increased by an order of magnitude. Another typical area for digital signal processing is for airborne military use where performance and reduced chip count are more important than cost.

It can be argued that the fastest mainframe computers could easily tolerate the high cost of monolithic arithmetic units. However, supercomputers need the performance obtainable only with a lower level of integration. For example, the Amdahl 470 computer has a microcyle time of 32ns, and the Cray-1 computer microcycles at 12ns.

SUMMARY

In this article we have surveyed the available Arithmetic Integrated Circuits. These ICs range from a simple but fast 4-bit ALU to a complex, but slow, floating point processor. These two directions are likely to continue for the next decade. For the low end microcomputers there will be a variety of single chip arithmetic processors capable of doing floating point and triginometric functions in about 10 microseconds. While medium and high end computers will implement high-speed (100ns) floating point operations using combinatorial building blocks such as multipliers, shifters and adders. With the continuing decrease in the price of ICs it can be expected that the computers of the 80's will have floating point processors as a standard option rather than an expensive option.

REFERENCES

[AMD 76] Advanced Micro Devices, Inc., The AM2900 Family Data Book, Sunnyvale, CA, 1976.

[AMD 77] Advanced Micro Devices, Inc., Schottky and Low-Power Schottky Data Book, Sunnyvale, CA 1977.

[AMD 78a] Advanced Micro Devices, Inc., "2903 Data Sheet", Sunnyvale, CA, 1978.

[AMD 78b] Advanced Micro Devices, Inc., "Algorithm Details for the Am9511 Arithmetic Processing Unit", Sunnyvale, CA, 1978.

[AND 67] Anderson, F. S., et al, "The IBM System 360/91 Floating Point Execution Unit", IBM JRD, Vol. 11 #1, (Jan. 1967) pp. 34-53.

[BOO 51] Booth, A. D., "A Signed Binary Multiplication Technique", Quart. Journal Mech. Appl. Math., Volume 4, Part 2, 1951.

[BRU 76] Brubaker, T. A. and Becker, J. C., "Multiplication Using Logarithms Implemented with Read Only Memories", IEEE Trans. Comput., Volume C-24, August, 1975, pp. 761-765.

[HOD 76] Hodges, D.A., "Trends in Computer Hardware Technology", Computer Design, February, 1976, pp. 77-85.

[JUL 78] Jullien, G. A., "Residue Number Scaling and Other Operations Using ROM Arrays", IEEE Transactions on Computers, Vol. C-27, No. 4, April 1978, pp. 325-336.

[LYM 77] Lyman, J., "Growing Pin Count is Forcing LSI Package Changes", Electronics, March 17, 1977, pp. 81-91.

[MAC 61] MacSorley, O.L., "High-Speed Arithmetic in Binary Computers", proc. IRE, Volume 49, January, 1961, pp. 67-91.

[MMI 78] Monolithic Memories, Inc. Bipolar LSI Data Book, Sunnyvale, CA, July 1978.

[MOT 74] Motorola, MECL Data Book, Phoenix, Arizona, 1974.

[NOY 76] Noyce, R. N., "From Relays to MPU's", Computer, December, 1976, pp. 26-29.

[PAS 78] Pashley, R., et al, "H-MOS Scates Traditional Devices to High Performance Level", Electronics, August 18, 1977, p. 95.

[SKL 60] Sklansky, J., "Conditional-Sum Addition Logic", IRE Transactions on Electronic Computers, Vol. EC-9, No. 2, June 1960, pp. 226-231.

[STE 77] Stenzel, W.J., et al, "A Compact High-Speed Multiplication Scheme", IEEE Transactions on Computers, Vol. C-26, No. 10, October 1977, pp. 948-957.

[TI 76] Texas Instruments, Inc., The TTL Data Book for Design Engineers, 2nd Ed., Dallas, Texas, 1976.

[TI 77] Texas Instruments, Inc., Bipolar Microcomputer Components Data Book, Dallas, Texas, 1977.

[TRW 77] TRW, "MPY-Series Multipliers", Redondo Beach, CA, December, 1976.

[TRW 78] TRW, "12 Bit Parallel Multiplier-Accumulator-TDJ1003J", Redondo Beach, CA, August 1977.

[WAL 64] Wallace, C. S., "A Suggestion for a Fast Multiplier", IEEE Trans. Electron. Comput., Volume EC-13, February 1964, pp. 14-17.

[WAS 78] Waser, S., "State-of-the Art in High Speed Arithmetic Integrated Circuits", Computer Design, July 1978.

[WEI 56] Weinberger, Al, Smith, J. L., "A One Microsecond Adder Using One Megacycle Circuitry", Trans. IRE, EC-5 #2 (June 1956).

[WIN 67] Winograd S., "On the Time Required to Perform Addition", J. Ass. Comput. Mach., Vol. 12, no. 2, pp. 277-285, 1965.