64 Bit Monolithic Floating Point Processors

FREDERICK A. WARE, MEMBER, IEEE, WILLIAM H. McALLISTER, JOHN R. CARLSON, DAN K. SUN, AND RICHARD J. VLACH

Abstract—**This paper describes a set of three processor chips capable of performing 32 and 64 bit floating point add/subtract, multiply, and divide operations. The chips can perform over one million scalar floating point operations per second, and over four million vector operations per second. The set is implemented in a four micron CMOS-on-sapphire process. Each chip has between 30 000 and 60 000 devices, and is about 250 mils on a side. Although asynchronous data paths are used within the chips, their interface to external system buses is synchronous with a maximum data bandwidth of over 70 Mbytes/s. The set has been designed for use in Hewlett-Packard computer and instrument systems.**

I. Introduction

HEWLETT-PACKARD has developed a set of three floating point processor chips (an adder/subtracter, a multiplier, and a divider) that are to be utilized by a variety of computer and instrument systems within the company. This paper provides a description of the design and operation of the individual chips.

The principle design objective was maximum performance of 32 and 64 bit floating point scalar (single element) operations. This was accomplished primarily with the three chip partitioning which allows each fundamentally different operation to be optimized with its own data path. Also, combinational circuitry is provided for minimum execution delay of each operation.

A second objective was to maximize the ease of use of the chip set for the system designer. Static logic is used throughout the design, and all internal registers are edge-triggered with a single clock. All inputs and outputs are TTL compatible, and outputs are tristated. Power consumption is low because the chip set is implemented with static CMOS circuitry.

The third design objective was maximum vector performance. The add/subtract and multiply chips each contain a pipeline storage register that allows the combinational logic delay to be overlapped with loading and unloading operations. Also, it is possible to parallel several chips of one type and interleave their operation, thereby improving the vector (multiple element) bandwidth.

II. Chip Operation

As shown in the block diagrams of Figs. 3(a), 8(a), and 9(a), the chips have similar interface and control circuitry, but the data path logic is unique to each. Each chip has three 16 bit unidirectional data buses (two input and one output) that have maximum bandwidths of over 12 MHz. The 32 and 64 bit operands and results must be time-multiplexed for loading and unloading. The floating and fixed point data formats used are those of the HP1000 minicomputer.

Manuscript received February 23, 1982; revised May 5, 1982.

The authors are with the Computer Systems Division, Hewlett-Packard, Inc., Palo Alto, CA 94304.

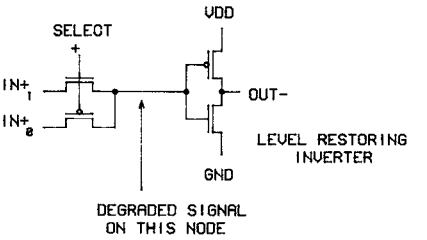


Fig. 1. Two-to-one multiplexer implemented with single-sided pass devices (p devices have bubbles, n devices do not). Half the time the devices pass a degraded signal which is restored by an inverter. Disadvantages of this technique are loss of noise margin and performance and an increase in dc power dissipation in the inverter. The principle advantage of this technique is its high functional density.

Operands and the function opcode are loaded into a chip on two or four clock edges. The data propagate through combinational data path logic, and may be directly unloaded in another two or four clock cycle. Results may alternately be passed through a pipeline register allowing two operations to be overlapped. Unloading this register requires two or four cycles as before.

The divide chip contains a single clocked feedback path within its data path logic, and thus is not purely combinational. Its internal delay is about two to four times the 400 ns-600 ns delay required for add/subtract and multiply. Unlike the add/subtract and multiply chips, it has no pipeline register since the internal delay is far greater than load/unload delays.

III. Circuit Design Techniques

The CMOS-on-sapphire process used for the chip set has four micron feature sizes and metal gate devices. The supply voltage is relatively large (12 V) compared to the threshold voltages (1.5-2.0 V). Also, there is less performance degradation due to the body effect with SOS devices than with bulk CMOS devices. This allows circuit designers limited use of devices operating as source-follower drivers, i.e., p devices passing a low level or n devices passing a high level.

A two-to-one multiplexer can be implemented with two single-sided pass devices, and with a level restoring inverter as in Fig. 1. This requires about half the area of an implementation utilizing double-sided pass devices [Fig. 1(b)]. The performance of the single-sided circuit is less than that of the other implementation, but it is still acceptably fast, particu-

$$C_{i+1} = P_i C_i + \overline{P_i} B_i$$

$$C_{i+2} = P_{i+1} P_i C_i + P_{i+1} \overline{P_i} B_i + \overline{P_{i+1}} B_{i+1}$$

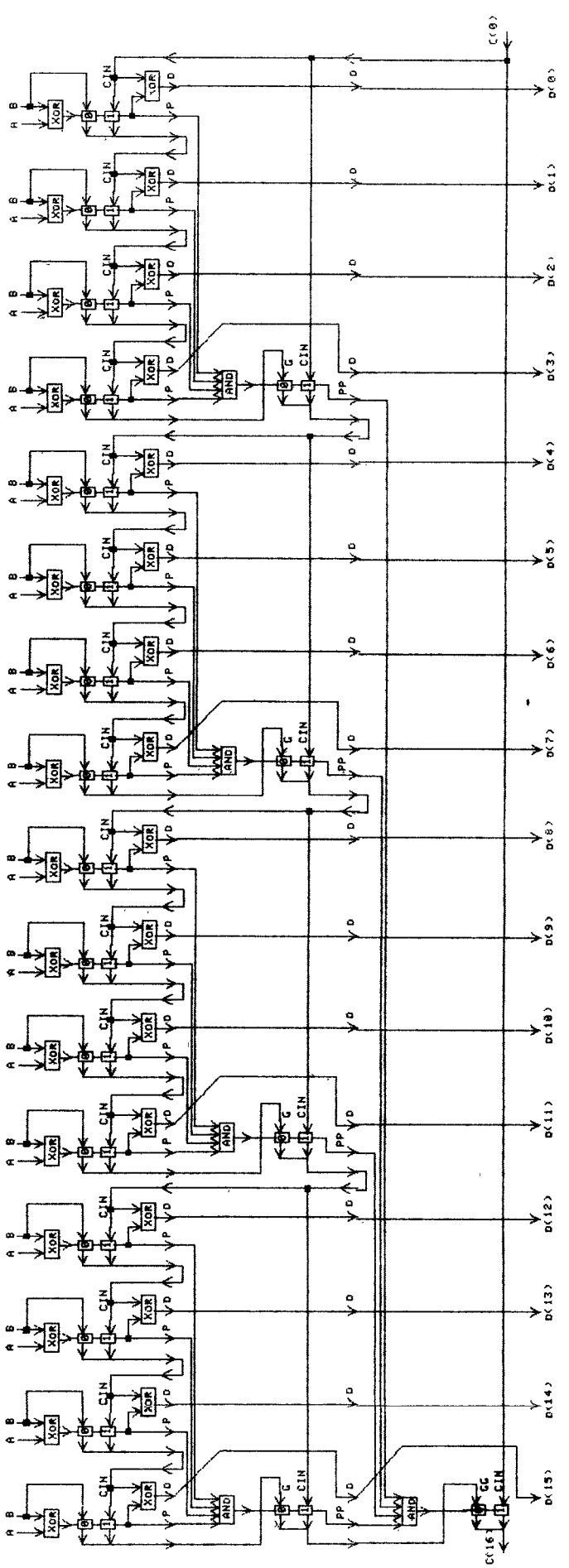
$$C_{i+3} = P_{i+2} P_{i+1} P_i C_i + P_{i+2} P_{i+1} \overline{P_i} B_i + P_{i+2} \overline{P_{i+1}} B_{i+1} + \overline{P_{i+2}} B_{i+2}$$

$$G_{i+4} = P_{i+3} P_{i+2} P_{i+1} P_i C_i + P_{i+3} P_{i+2} P_{i+1} \overline{P_i} B_i + P_{i+3} P_{i+2} \overline{P_{i+1}} B_{i+1} + P_{i+3} \overline{P_{i+2}} B_{i+2} + \overline{P_{i+3}} B_{i+3}$$

$$PP_i = P_{i+3} P_{i+2} P_{i+1} P_i$$

$$C_{i+4} = PP_i C_i + \overline{PP_i} G_{i+4}$$

(a)



(b)

Fig. 2. (a) Boolean equations for a four bit adder slice with indexes i through $i + 3$. The *Ai*, *Bi*, and *Pi* terms are, as before, for each of the four bits. The $Ci + 3$, $Ci + 2$, and $Ci + 1$ terms are expressed as functions of any of the *Bi* and *Pi* and of only the first carry signal *Ci*. The carry-out of the four bit slice $Ci + 4$ is then generated from two intermediate terms $Ci + 4$ and *PPi*. The equations may be efficiently implemented with two-to-one multiplexers and Exclusive-OR gates as in Fig. 2(b). (b) This is a 16 bit slice of a 64 bit carry propagate adder with three levels of look-ahead logic. Each four bit slice is equivalent to the equations of Fig. 2(a). *Ai* and *Bi* are the bits of two 64 bit numbers to be added. *Pi* is the Exclusive-OR of *Ai* and *Bi*, and is used to control a two-to-one multiplexer to form the carry out $Ci + 1$. $Ci + 1$ is either given the value of *Ci* (PASS) or *Bi* (KILL/GENERATE) if *Pi* is one or zero, respectively. *Pi* is also Exclusive-ORed with *Ci* to form the sum bit *Di*. The ripple carry chain is broken at four and sixteen bit intervals. If all four or sixteen bits have a propagate signal *Pi* equal to one, the carry-in to that slice may skip over it. The above logic is implemented with twenty transistors/bit–sixteen for the full adder and four for the look-ahead circuitry.

larly when compared to equivalent implementations with bulk rather than SOS devices. The density advantages of the single-sided circuit are far more important for the processors than the additional 20 percent or 30 percent overall propagation delay that must be tolerated.

IV. Carry Propagate Adder

The most common digital function required for the chip set was that of fixed point addition with carry propagation. This operation may be implemented in a variety of ways depending upon the exact area-performance tradeoff required. The 64 bit precision of the chip set meant that silicon area for carry propagation should be minimized as far as was practical.

An efficient carry look-ahead addition scheme which requires only 20 transistors/bit is shown in Fig. 2(b) along with the Boolean equations in Fig. 2(a). It has an equivalent delay of 15 to 20 full adder delays, giving three to four times the performance of a ripple adder for about 25 percent additional silicon area (a ripple adder requires 16 transistors/bit).

It is possible to design 64 bit adders with up to five or six times the ripple adder performance by utilizing conditional sum techniques [1]. However, the device count per bit is two to three times that of the ripple adder. Such an approach was not judged to be a good area-performance tradeoff.

V. Floating Point Add/Subtract Chip

A block diagram and microphotograph of the floating point add/subtract chip are shown in Fig. 3(a) and (b). There are about 30 000 devices on a die that is 230 X 255 mils in size. Most of the chip area is devoted to the fraction data path in the center and right. The major hardware structures in this path are operand registers, right shifter, adder and priority encoder, left shifter, and incrementer and output register. The shifters make the distinctive slanting patterns.

The exponent data path is in the upper left and control logic in the lower left. Operand input pads are along the top and right sides, result pads are along the bottom, and control pads are along the left side.

When the two operands are first loaded, their exponents are compared and the larger one passed on. The fraction with the smaller exponent is right shifted by N bits, where N is the difference of the two exponents. The right shift logic consists of six two-to-one multiplexers/bit, and the wiring to perform power-of-two shifts.

Next, the addition or subtraction of fractions is performed exactly as with fixed point operands, except that at least three extra bits of precision must be included to provide accuracy of one half of the LSB (least significant bit). When the two fractions have the same sign, a fraction overflow may occur. If this happens, the fraction is right shifted once to put it back into the range, and the exponent is incremented.

If the operand fractions have different signs, there may be cancellation of leading significant bits, yielding an unnormalized result. Normalization is performed with a priority encoder and an N bit left shifter. The priority encode logic uses a look-ahead circuit (similar to that of the carry propagate adder) to detect the position of the first significant bit from the left. This position is encoded into a six bit value that is subtracted from the exponent, and is also used as the shift amount by the left shifter.

The extra bits of precision that were carried are now used to perform rounding of the normalized result fraction. An incrementer (with a look ahead unit identical to that of the carry propagate adder) conditionally adds one LSB to the fraction. If the fraction overflows, the exponent is incremented and the fraction right shifted one place.

The final step is checking the exponent for overflow or underflow. If either occurs the appropriate constants are generated for the exponent and fraction of the result, and the status bits are set.

VI. Multiplication Techniques

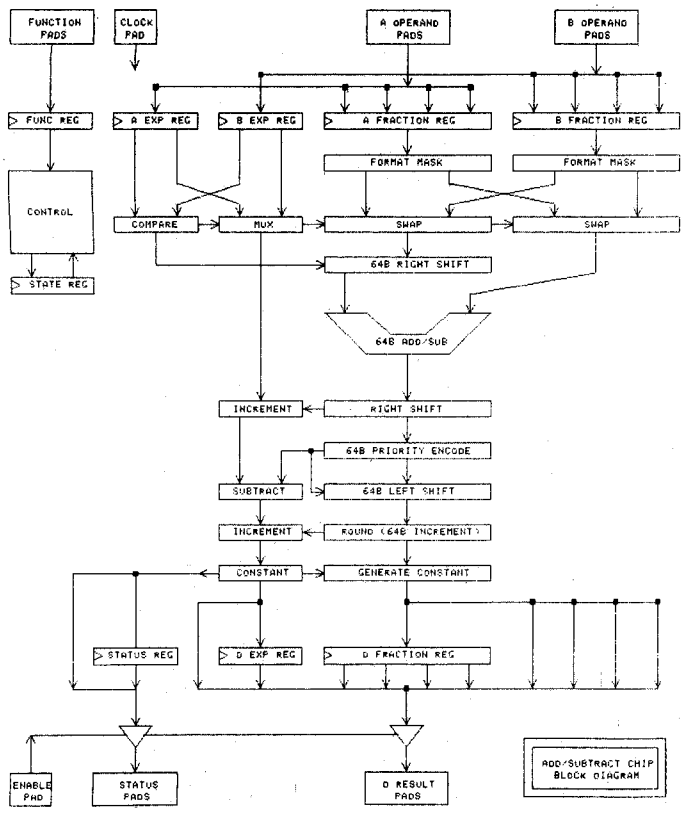
Fig. 4 illustrates the application of signed digit encoding to an 8 X 8 two's complement multiplier. The technique allows the number of partial products to be reduced by a factor of two, and requires only a small amount of shifting and multiplexing logic.

The Boolean truth table for a commonly used encoding scheme (modified Booth encoding) is shown in Fig. 5. The original Booth encoding technique [2] is able to skip over arbitrarily long strings of ones or strings of zeros; the modified technique more rigidly examines only pairs of multiplier bits and converts them into the set of five signed digits [3]-[5].

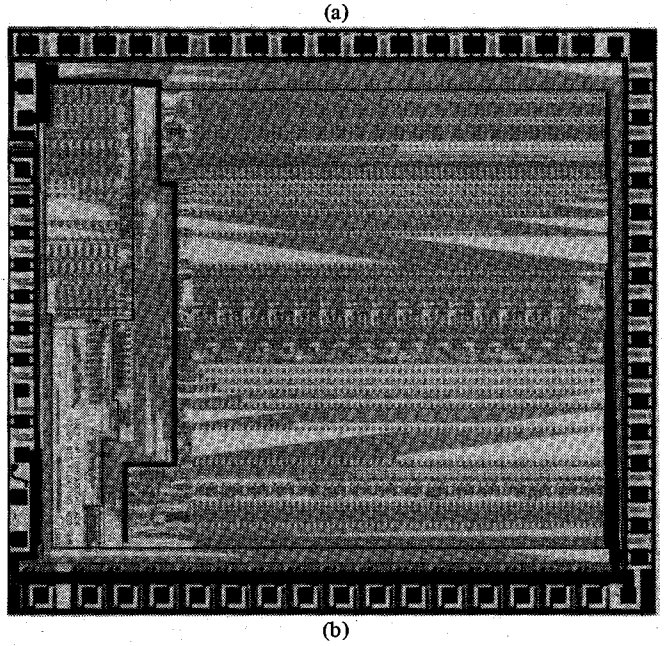
An alternate encoding scheme not previously reported is possible. It uses the reduced signed digit set +2, +1, 0, -1 by modifying line five of the truth table to give a multiplier of $XP2$ and a signed-digit-carry-out of zero. The disadvantage of this is that the signed-digit-carry-out is now a function of the signed-digit-carry-in, unlike the original scheme. It might be thought that this would impact the multiplier performance since the signed-digit-carries must ripple down the multiplier bit pairs. However, since the partial products must ripple down rows of carry save adders in the array as they accumulate, there is in fact no performance penalty. The advantage of the technique is that no *XM2(J)* version of the multiplicand needs to be generated in the multiply array, allowing an efficient four-to-one multiplexer to perform the encoding operation in each array cell.

A complete 8 X 8 recoded multiplier is shown in Fig. 6(a) along with a summary of Boolean equations for its logic blocks in Fig. 6(b) [the equations work for either encoding technique]. The 8 bit operands are A(7-0) and B(7-0). The inputs *TCA* and *TCB* indicate whether the A and B operands are two's complement (=1) or unsigned (=0). The 16 bit result of the multiplication is D(15-0), and the F_{ADD} and H_{ADD} blocks are conventional full adders and half adders.

The schematic for the critically important array cell for the multiplier is shown in Fig. 7(a). A microphotograph of the array is in Fig. 7(b). The metal interconnect layer runs horizontally and is used for power, ground, and the encoded B operand lines. The silicon island layer runs vertically and is used for the A operand lines. There is significant series resistance in these vertical silicon lines, but because they are



(a)



(b)

Fig. 3. (a) Floating point add/subtract chip block diagram. (b) Floating point add/subtract chip microphotograph.

driven from the top of the chip and because the rows of full adder cells ripple from top to bottom, the full *RC* time penalty is not paid.

Note that a four-to-one multiplexer is used to select the *XP*2, *XP*1, *X*0, or *XM*1 version of the *A* operand. Because single sided transfer gates are used (along with a level restoring inverter), only six devices and two metal control lines are required. This compact circuit could not be used if the conventional encoding scheme with five signed digits were utilized.

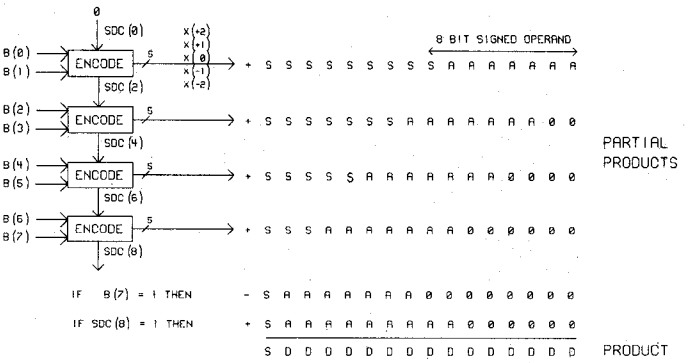


Fig. 4. Two's complement 8 × 8 multiplication with encoding. Pairs of multiplier bits are encoded into the signed digit set +2, +1, 0, −1, −2. A signed-digit-carry is passed between pairs of *B* multiplier bits to allow the bit pair "11" to be represented (it is represented by a −1 signed digit and a carry-out of one). The Boolean table for the encoding logic is given in Fig. 5. The signed digit from the encoding process multiplies the left-shifted, sign-extended *A* operand and accumulates it. If the *B* operand is negative ($B(7) = 1$), then the indicated correction term must be subtracted. If $SDC(8) = 1$, then the same term must be added. These two corrections are performed by a single addition, a single subtraction, or by no action.

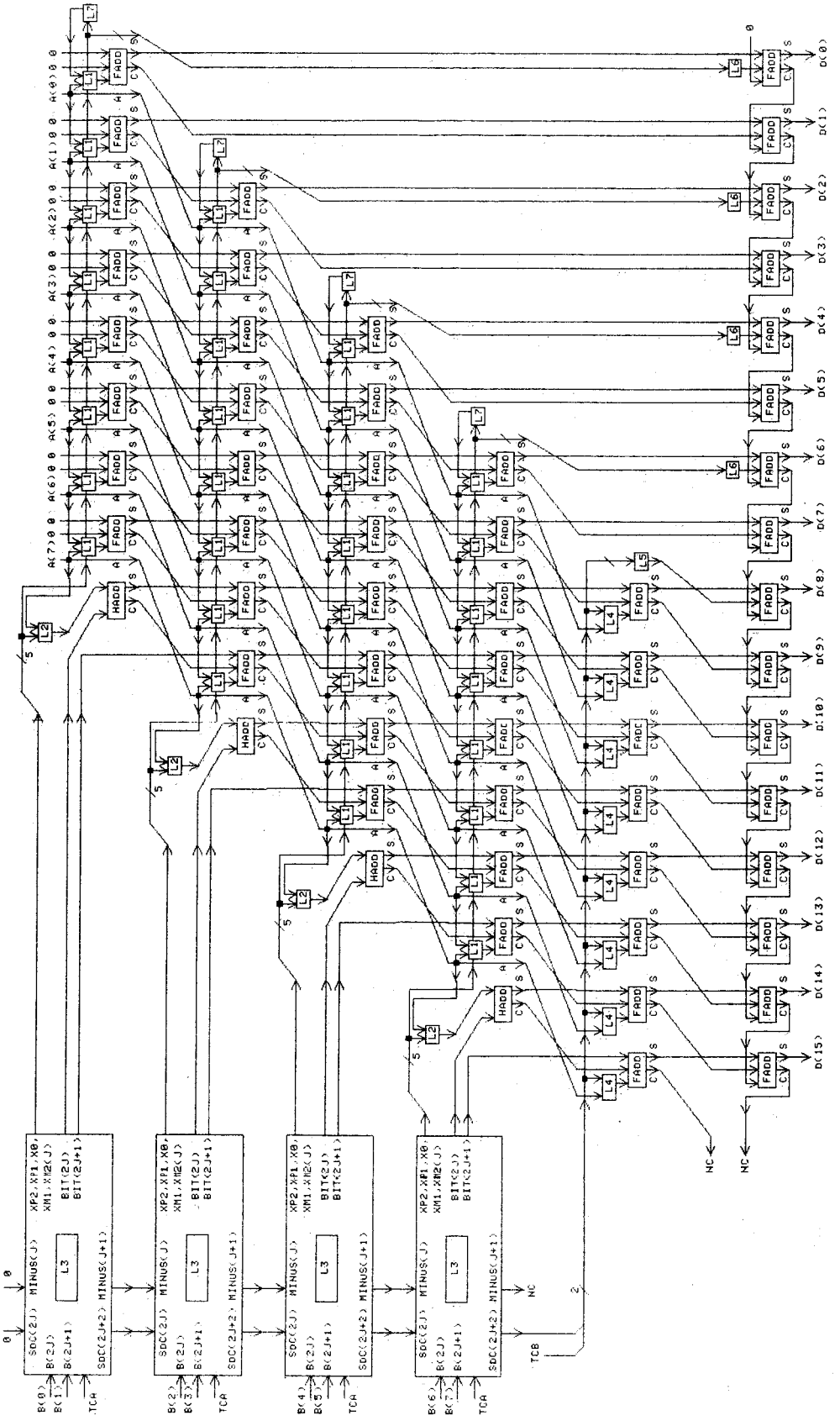
B(2J+1)	B(2J)	SDC(2J)	MULTIPLIER	SDC(2J+2)
0	0	0	X0	0
0	0	1	XP1	0
0	1	0	XP1	0
0	1	1	XP2	0
1	0	0	XM2	1
1	0	1	XM1	1
1	1	0	XM1	1
1	1	1	X0	1

Fig. 5. Boolean table for the two bit encoding scheme utilizing the full set of signed digits: +2, +1, 0, −1, −2. The signed-digit multiplier (*XP*2, *XP*1, *X*0, *XM*1, *XM*2) and signed-digit-carry-out are functions of the signed-digit-carry-in and the two *B* operand bits. The two operand bits have relative weights of $x2$ and $x1$, and the carry-in has a weight of $x1$. The weight of the signed digits +2, +1, 0, −1, −2 is $x1$, and the weight of the carry-out is $x4$. In each of the eight lines of the table, the sum of the three weighted input terms $B(2J+1)$, $B(2J)$, and $SDC(2J)$ equals the sum of the two weighted outputs. The index J runs from 0 through 3 for an 8 × 8 multiplier.

VII. Floating Point Multiply Chip

Floating point multiplication is relatively simple compared to floating point add and subtract. Referring to Fig. 8(a) and (b), it is seen that most of the silicon is used to perform integer multiplication of the fraction parts (center and right). The circuitry at the lower left of the chip contains the exponent data path, which occupies a very small portion of the chip compared to the exponent path of the floating point adder. The multiplier is roughly 245 × 280 mils, and contains about 60 000 devices.

When the operands are first loaded, the two exponents are added, and the fractions are masked to the proper precision. One operand is encoded as previously described, and conventional integer multiplication is then performed. The integer product (in carry propagate form) is normalized with a zero or one bit left shift, and then rounded to the final precision. The result exponent is incremented if there was a zero bit shift or if the fraction overflowed after rounding. The exponent is checked for exponent overflow or underflow and if either oc-



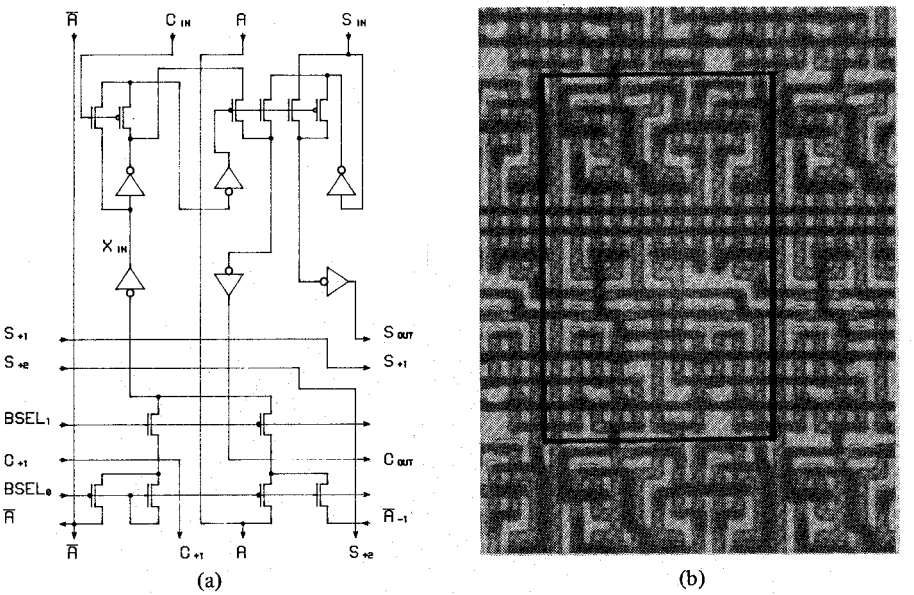
(a)

Fig. 6. (a) Complete 8×8 encoded multiplier. The J index runs from 0 through 3 and is used for the (vertical) multiplier $B(7\text{-}0)$. The K index runs from 0 through 7 and is used for the (horizontal) multiplicand $A(7\text{-}0)$. The inputs *TCA* and *TCB* indicate whether each operand is two's complement (=1) or unsigned (=0). The output result is $D(15\text{-}0)$. Logic block $L3$ accepts pairs of multiplier bits $B(2J)$ and $B(2J+1)$ and a signed-digit-carry-in $SCD(2J)$, and produces signed-digit-carry-out $SCD\ (2J+2)$ and asserts one of the five signed-digit multiplier lines $XP2, XP1, X0, XM1$, or $XM2$. This logic block also uses the minus(J) signal to generate two signals bit($2J$) and bit($2J+1$) that perform in effective sign extension of each partial product to 16 bits. Logic blocks $L1, L2, L6$, and $L7$ perform the shifting and multiplexing operations that multiply the multiplicand $A(7\text{-}0)$ by one of the signed digits +2, +1, 0, −1, −2. The F_{ADD} and H_{ADD} blocks are conventional full adders and half adders. Logic blocks $L4$ and $L5$ perform sign correction for the B operand, and correct for a signed-digit-carry-out of the last $L3$ block [$SDC(8)$]. The final carry propagation of the product $D(15\text{-}0)$ is shown using ripple adders, although a real circuit would use a look-ahead technique for the upper 8 bits.

```
L1  LOGIC
    L1(K,J)    = [XP2(J) • A(K-1)] + [XP1(J) • A(K)]
                 + [XM1(J) • Ā(K)] + [XM2(J) • Ā(K-1)]
L2  LOGIC
    L2(2J)     = [XP2(J) • A(7)] + [XM2(J) • Ā(7)]
L3  LOGIC
    NEGA       = TCA • A(7)
    P          = [XM2(J) • NEGA] + [XP2(J) • NEGA]
    Q          = [XM1(J) • NEGA] + [XP1(J) • NEGA]
    BIT(2J+1)  = [MINUS(J) XOR P] + [MINUS(J) • Q]
    BIT(2J)    = MINUS(J) XOR Q
    MINUS(J+1) = MINUS(J) + P + Q
L4,L5,L6,L7  LOGIC
    NEGB       = TCB • B 7
    L4(K)      = [NEGB XOR SDC(8)] • [NEGB XOR A(K)]
    L5         = NEGB • SDC(8)
    L6(J)      = XM1(J) + XM2(J)
    L7(J)      = A(-1) = 0
```

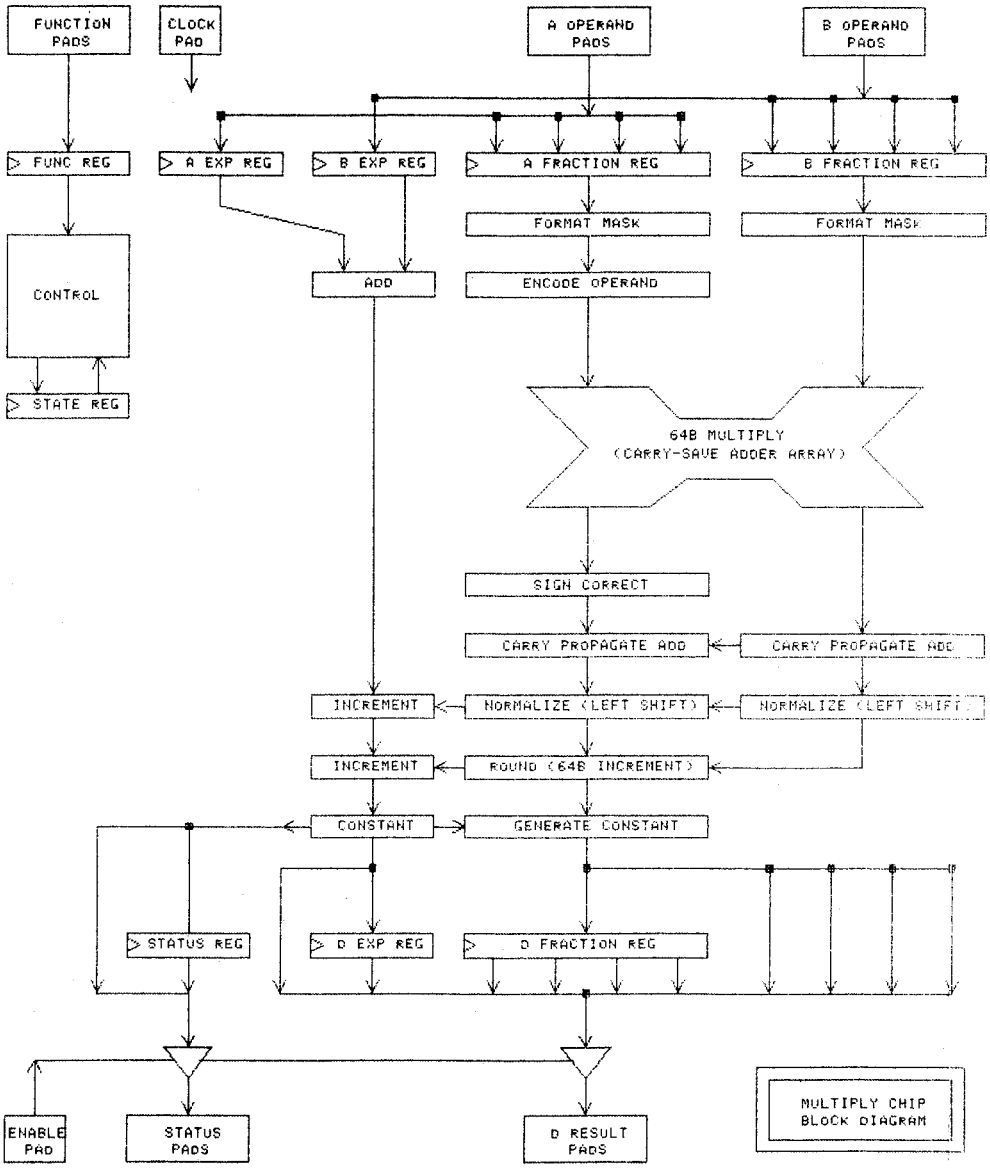
(b)

Fig. 6. (*Continued.*) (b) Summary of Boolean equations for the logic blocks of the 8 × 8 multiplier of Fig. 6(a). The index J runs from 0 through 3, and the index K runs from 0 through 7. The variables *NEGA*, *NEGP*, *P*, and *Q* are used within a single logic block, and therefore do not appear in Fig. 6(a).

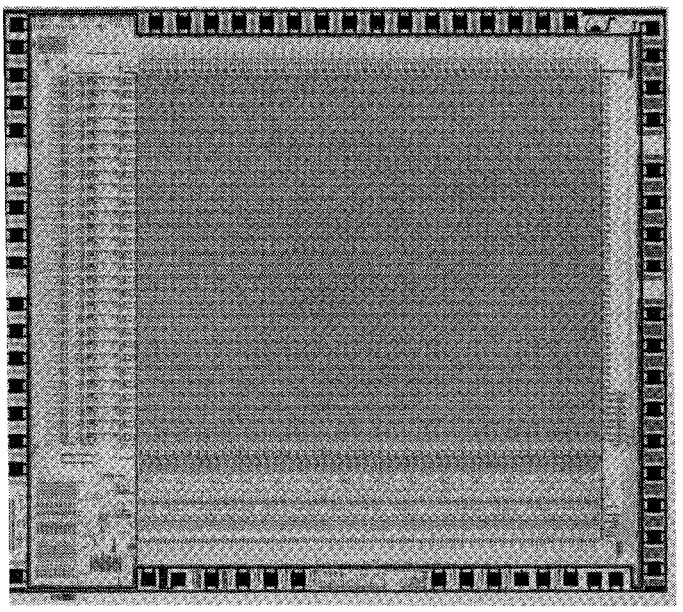


(a) (b)

Fig. 7. (a) Schematic of 2 × 1 multiply cell. This shows the 24 NMOS and PMOS devices used to implement the $L1$ logic block and FADD block of Fig. 6(b) (p devices have a bubble on the gate). Two select lines are encoded from each pair of multiplier bits, and indicate whether +2, +1, 0, or −1 is to multiply the operand A to form a partial product. The true and complement of the appropriate A bit pass vertically through the cell and are used by the four-to-one multiplexer at the bottom. A +2 multiplier implies using $A(-1)$, +1 implies $A(0)$, 0 implies zero, and −1 implies complemented $A(0)$. The multiplexer output X_{in} is accumulated in the full adder with C_{in} and S_{in} to produce S_{out} and C_{out}. The S_{out} (sum-out) line is passed two cells to the right before being used by the next row, and the C_{out} (carry-out) line is passed one cell. (b) Microphotograph of 2 × 1 multiply cell. The schematic of Fig. 7(a) is quite similar topologically to this photo. Metal lines are dark and run horizontally, and island silicon is lighter and runs vertically. The cell is outlined by a heavy black line. Power and ground run through the center of the cell in metal, powering the six inverters. Three two-to-one multiplexers are above the inverters, and one four-to-one multiplexer is below. The cell is 88 × 144 μm (11 island pitches by 16 metal pitches).

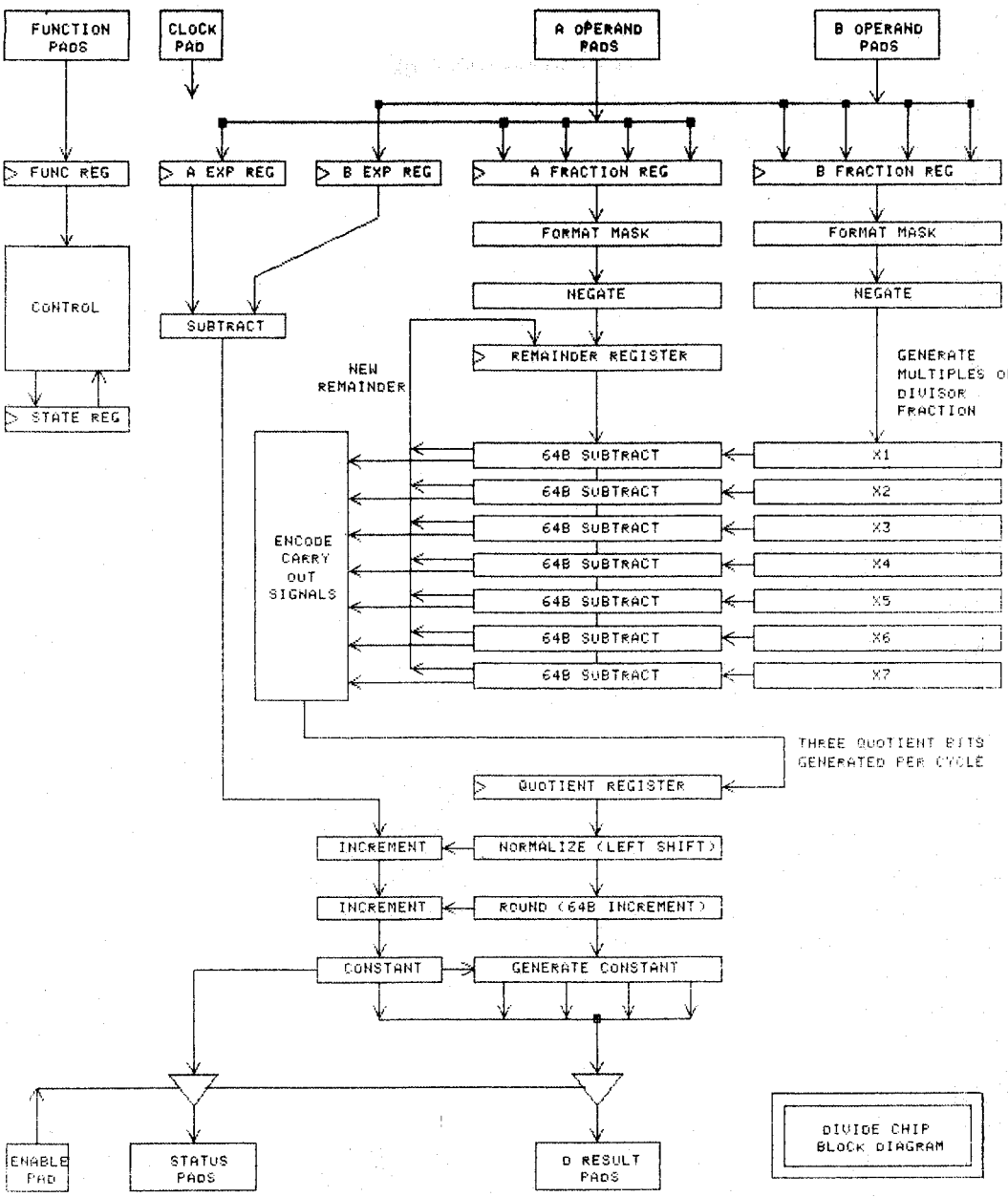


(a)

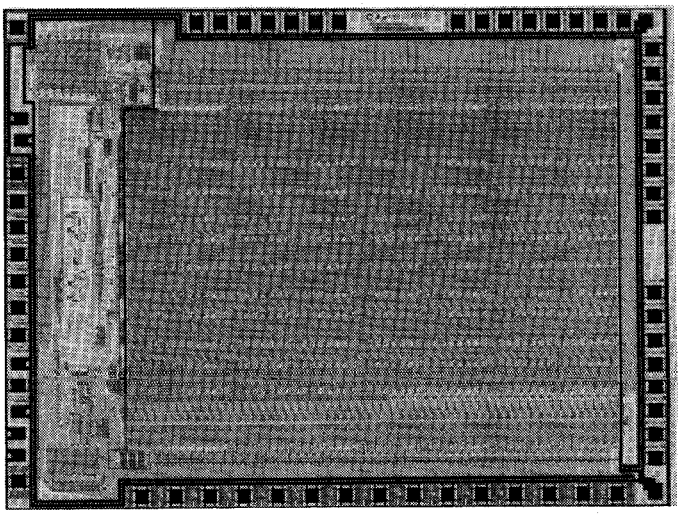


(b)

Fig. 8(a) Floating point multiply chip block diagram. (b) Floating point multiply chip microphotograph.



(a)



(b)

Fig. 9. (a) Floating point divide chip block diagram. (b) Floating point divide chip microphotograph.

curs the appropriate constants are forced and the status outputs set.

VIII. Floating Point Divide Chip

Floating point division, like multiplication, is not much harder than its fixed point counterpart. In Fig. 9(a) and (b) it is seen that the divide chip has a small exponent path in the upper and lower left. Most of the silicon is devoted to the fraction data path in the center and right of the chip. The chip is about 210 X 290 mils in size, and contains about 35 000 devices.

When the operands are first loaded, the two exponents are subtracted and the fractions masked to the proper precision. Conditional two's complement negation is performed so the dividend fraction is positive and the divisor fraction negative. The first seven integral multiples of the divisor are generated combinationally with three carry propagate adders and four left shifters. An extra load cycle is required at this point to transfer the dividend fraction into the remainder register.

The following sequence of events is repeated anywhere from nine to twenty times depending upon the fraction precision. The remainder fraction is passed to an array of seven carry propagate adders which add it to the seven negative divisor multiples to produce seven new remainders. The smallest positive remainder is passed back to the remainder register, and its encoded three bit position is shifted into the quotient register.

This technique is inherently faster than the method of using a full combinational array of 64 64-bit carry propagate adders [6]-[8]. The first technique generates three bits in the time to perform one carry propagation, to drive horizontal enable lines, and to drive the remainder into and out of the array. The combinational technique generates one bit in the time to perform one carry propagation and drive horizontal enable lines. Technique one is thus about two to three times faster.

When the quotient is completely loaded, it is normalized with a zero or one bit left shift, and then rounded to its final precision. The result exponent is incremented if there was a zero bit shift or if the fraction overflowed after rounding. The exponent is checked for exponent overflow or underflow and if either occurs the appropriate constants are forced and the status outputs set.

IX. Summary

This chip set will provide a powerful (up to one million floating point operations per second), yet inexpensive floating point execution unit for Hewlett-Packard minicomputer systems. In addition, multiple chip sets may be interleaved to implement vector processing units (four MFLOPS per chip set) in technically oriented minicomputers [9], [10].

The chips may also be used as building blocks to construct processing systems for specific applications. Such applications include signal and image processing, graphics processing, and the numerical solution of linear systems. Virtually any structured algorithm can benefit from the chip set in the implementation of high performance floating point systems.

Acknowledgment

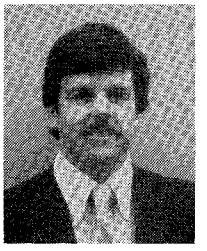
The authors wish to acknowledge the efforts of T. Shillingburg, J. Pelayo, H. Shishido, V. Wilson, L. Andres, T. Corsick, S. Boschma, B. Ames, and D. Zuras.

References

[1] J. Sklansky, "Conditional sum addition logic," *IRE Trans. Electron. Comput.*, vol. EC-9, pp. 226-231, June 1960.
[2] A. D. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. 4, pp. 236-240, 1951.
[3] S. Waser, "High speed monolithic multipliers," *Computer*, pp. 19-29, Oct. 1978.
[4] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.*, pp. 14-17, Feb. 1964.
[5] A. Habibi and P. A. Wintz, "Fast multipliers," *IEEE Trans. Comput.*, pp. 153-157, Feb. 1970.
[6] D. P. Agrawal, "High speed arithmetic arrays," *IEEE Trans. Comput.*, vol. C-28, pp. 215-224, Mar. 1979.
[7] I. Flores, *The Logic of Computer Arithmetic*. Englewood Cliffs, NJ: Prentice-Hall, 1963.
[8] K. Hwang, *Computer Arithmetic*. New York: Wiley, 1979.
[9] F. Ware, "64b monolithic floating point processors," in *ISSCC, Conf. Dig.*, vol. 25, Feb. 1982, pp. 24-25.
[10] F. Ware and W. McAllister, "CMOS floating point chip set," *Electron.*, vol. 55, no. 3, pp. 149-153, Feb. 1982.

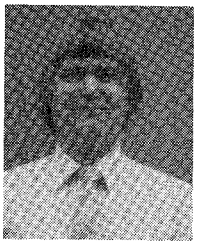
Frederick A. Ware (M'77) was born in Omaha, NE, on April 14, 1952. He received the B.S. degree in applied physics from the California Institute of Technology, Pasadena, in 1974, and the M.S. degree in electrical engineering from the University of California, Berkeley, in 1975.

He joined Hewlett-Packard's Data Systems Division in 1975, where he worked on custom integrated circuits for the HP300 central processing unit. In 1978 he began the development of CMOS/SOS floating point processor circuits. He is currently managing the design of a second set of floating point processors that will be compatible with the new IEEE standard.



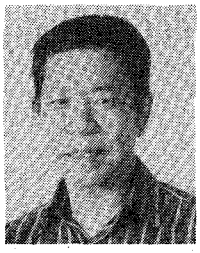
William H. McAllister received the B.S.E.E. degree from the University of California, Santa Barbara, in 1974, and the M.S.E.E. from Stanford University, Stanford, CA, in 1976.

After graduating he joined TRW, Inc., in Redondo Beach, CA, where he worked on the design and analysis of LSI for special-purpose signal processors. In 1980 he joined Hewlett-Packard, where he is currently working on custom VLSI chip design for arithmetic processors. His special interest is in the area of VLSI design methodologies.



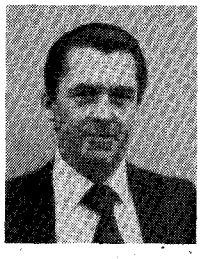
John R. Carlson received the B.S. degree in mathematics from California State University, Hayward, in 1973, the M.S. in computer science from the University of California, Davis, in 1975, and he is currently studying for the M.S. in electrical engineering at Stanford University, Stanford, CA.

He joined Hewlett-Packard in 1975, working on microcode. He has worked on the development of a microcode compiler and is currently working on NMOS floating point chips.



Dan K. Sun was born in China. He received the B.S. degree in computer science from the University of San Francisco, San Francisco, CA, in 1969.

Since joining Hewlett-Packard in 1977, he has been participating in SOS-CMOS process VLSI device testing.



Richard J. Vlach, a native of Ohio and graduate of CEI, moved to California in 1962. He worked for Lockheed Missile and Space Corporation helping in the development of a thin film process. He then worked in the areas of bipolar mask design, industrial engineering, and mask development at Stewart Warner Microcircuits. He later moved to National Semiconductor to design masks in bipolar, linear, and CMOS technologies, also making contributions in automated assembly and teaching mask design. He joined Hewlett-Packard in 1980 to work on the floating point processor family.

A 6K-Gate CMOS Gate Array

HARUYUKI TAGO, TERUO KOBAYASHI, MASARU KOBAYASHI, TAKAHIKO MORIYA, AND SHIN'ICHIRO YAMAMOTO

Abstract—**Combining advanced 2 μm CMOS technology with a newly developed double layer metallization technology, a high-performance 6K-gate CMOS gate array has been developed, featuring an inverter propagation delay time of 0.4 ns with a power dissipation of 10 μW/MHz/stage. As a demonstration vehicle of the high-performance gate array, a 16 bit × 16 bit parallel multiplier has been designed and fabricated in which 3365 basic cells are used. Typical multiplying time has been measured to be 130 ns at a 5 MHz clock rate with a power dissipation of 275 mW.**

I. Introduction

With the proliferation of LSI usage in industry in general, the trend toward the customization of LSI's for obtaining the optimum cost/performance at the system level is becoming clear. Although customization of LSI's is attractive from the performance point of view, the development cost is becoming prohibitive with the increase of integration density and circuit complexity because of the huge amount of resources and long development time required. To contain the development cost to a reasonable level as well as to shorten the development time, master slice or gate array approaches have become popular because of the ease of adopting design automation and uniquely short processing steps. Previously, bipolar technology [1] had been used for obtaining subnanosecond delay time, but it dissipated a lot of power. CMOS technology, on the other hand, showed low power dissipation characteristics and high packing density, but it was relatively slow. The recent emergence of high-performance CMOS device technology prompted the introduction of high-speed and high-density memories [2]-[4]. Combining advanced CMOS technology with double layer metallization technology incorporating a newly developed low-temperature planarization technique [5], a 6K-gate CMOS gate array chip has been developed. The first section of this paper briefly describes the fabrication processes. Then the basic cell configuration, the design considerations of the macrocell, as well as the basic performance of the array are discussed. Finally, the circuit as well as the performance aspects of the 16 bit × 16 bit multiplier as a demonstration vehicle of a 6K-gate array are described.

Manuscript received March 22, 1982; revised May 24, 1982.

The authors are with the Toshiba R and D Center, Toshiba Corporation, Kanagawa, Japan.

0018-9200/82/1000-0907$00.75