# A CMOS Floating Point Multiplier

MASARU UYA, KATSUYUKI KANEKO, AND JURO YASUI

*Abstract*—**This paper describes a 32-bit CMOS floating point multiplier. The chip can perform 32-bit floating point multiplication (based on the proposed IEEE Standard format) and 24-bit fixed point multiplication (two's complement format) in less than 78.7 and 71.1 ns, respectively, and the typical power dissipation is 195 mW at 10 million operations per second.**

**High-speed multiplication techniques—a modified Booth's algorithm, a carry save adder scheme, a high-speed CMOS full adder, and a modified carry select adder—are used to achieve the above high performance.**

**The chip is designed for compatibility with 16-bit microcomputer systems, and is fabricated in 2 μm n-well CMOS technology; it contains about 23 000 transistors of 5.75×5.67 mm² in size.**

## I. Introduction

RECENTLY, the need for high-speed and high-precision computation has been increasing in applications for image processing, computer graphics, robotics, model simulation, CAD/CAM, measurement, and so on. Floating point computation is most suitable for these applications because it keeps the precision of operation high in spite of the wide-dynamic range. For microcomputer systems, not only high-speed but also small-sized floating point devices are in great demand.

The authors have developed a high-speed, low-power 32-bit floating point multiplier employing 2 μm design rule n-well CMOS technology. The data format is based on the proposed IEEE P754 Standard, which is suitable for microcomputer systems.

First, the configuration of the chip is described. Then, several techniques employed to achieve high-speed operation are discussed. The last part of the paper describes the performance and characteristics as well as considerations relating to chip operation.

## II. Chip Configuration

According to the single precision format of the proposed IEEE P754 Standard, the value of a multiplier $X$ (normalized number) is described as

$$X = (-1)^{Sx} \cdot 2^{Ex-127} \cdot (1.Fx)$$

where $Sx$, $Ex$, and $Fx$ are the 1-bit sign, the 8-bit exponent biased by 127, and the 23-bit fraction, respectively $(1.Fx)$ is the 24-bit mantissa consisting of an implied leading "1" and $Fx$. So, the range of the mantissa is $1 \leqslant (1.Fx) < 2$.

Manuscript received March 29, 1984; revised May 22, 1984.
The authors are with the Matsushita Electric Industrial Co., Ltd., Central Research Laboratory, Osaka 570, Japan.

Also, the value of a multiplicand $Y$ has the expression

$$Y = (-1)^{Sy} \cdot 2^{Ey-127} \cdot (1.Fy).$$

Then, the product $P = X \cdot Y$ comes to

$$P = (-1)^{Sx \oplus Sy} \cdot 2^{(Ex+Ey-127+a)-127} \cdot [(1.Fx) \cdot (1.Fy)] \cdot 2^{-a}$$

where "$a$" results from

$$a = 0: \quad 0 \leqslant (1.Fx) \cdot (1.Fy) < 2$$
$$a = 1: \quad 2 \leqslant (1.Fx) \cdot (1.Fy) < 4.$$

The chip is designed to fit 16-bit microcomputer systems. The 32-bit multiplier/multiplicand and product are loaded and taken out by 16 bits on two clocks each. An error signal is output for use as an interrupt to the host microcomputer when a multiplication finishes abnormally, and then overflow/underflow flags can be read if necessary. Aside from 32-bit floating point multiplication, fixed point multiplication can be provided in 24-bit two's complement data format.

A block diagram of the floating point multiplier is shown in Fig. 1. The chip consists of input/mode registers, an exponent adder, a mantissa multiplier, and output/status registers. Multiplier $X$, multiplicand $Y$, and mode control signals ($FIX$, $RND$) are clocked into 16-bit input registers ($XH$, $XL$), ($YH$, $YL$), and a mode register ($FIX$, $RND$), respectively, through a bidirectional 16-bit bus ($D15$-$D0$). The $FIX$ is operation mode control ($FIX = 0$: 32-bit floating point operation, $FIX = 1$: 24-bit fixed point operation). The $RND$ is the rounding mode control [$RND = 0$: round toward zero, $RND = 1$: round to the nearest value (add $\frac{1}{2}$ LSB to the result)]. Selection signals ($X/\overline{Y}$, $H/\overline{L}$, $S/\overline{D}$) and a clock $CLK$ are used for time multiplexing the bus. $\overline{OE}$ is the output enable signal for the bus. $ERR$ is an error signal indicating the occurrence of overflow or underflow of multiplication. Exponent adder block executes the operation of the sign and exponent in floating point multiplication mode. Besides the 8-bit exponent addition, this block detects invalid input data and predetects overflow/underflow. The final overflow/underflow is determined in the overflow/underflow detector ($OVF/UDF\ DET$).

Mantissa multiplier is a 24×24-bit asynchronous parallel multiplier which can perform both signed 2's complement and unsigned magnitude multiplication with/without rounding. The 24-bit result taken out of it is sent to the normalizer, where the above result (1-bit sign, 8-bit expo-

0018-9200/84/1000-0697$01.00 

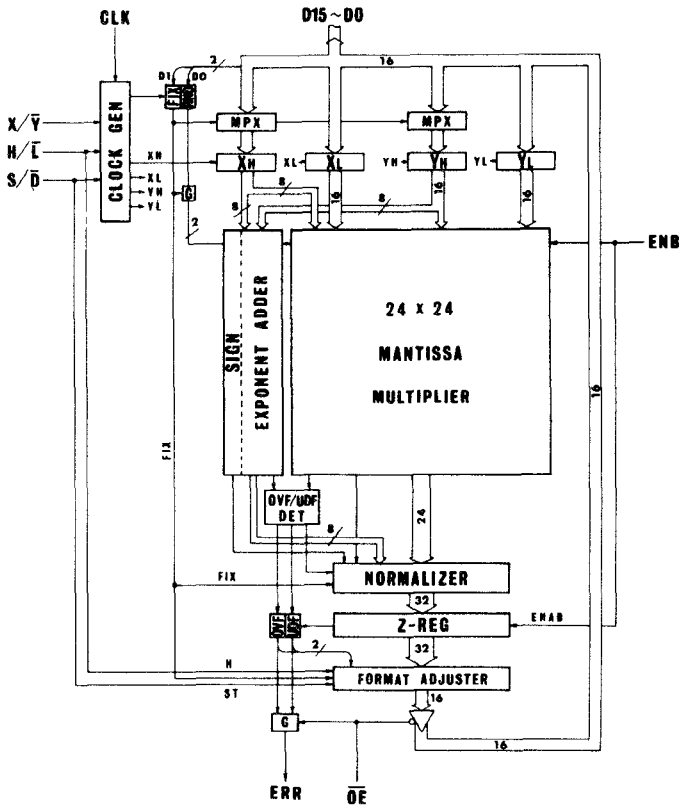



Fig. 1. Block diagram of the floating point multiplier.

nent, and 24-bit mantissa) is formed into IEEE standard format normalized numbers and are sent to the output register $Z$. The normalizer forces the result to the true zero ($Ez = Fz = 0$) when underflow occurs. The 32-bit result and the status are put on the bus 16 bits at a time.

*ENB* is the enable signal for the multiplier and is provided to measure the multiply time. The execution of multiplication is triggered by the rising edge of the *ENB*, and the output/status registers are latched by the falling edge of the *ENB*. Thus, the pulse width of the *ENB* is the multiply time.

As mentioned above, this chip is designed to configure easily with 16-bit microcomputer systems.

## III. Exponent Adder Design

As the result of mantissa multiplication is in the range $1 \leqslant (1.Fz) < 4$, the exponent $Ez$ must be incremented by 1 when $2 \leqslant (1.Fz) < 4$. This normalization (increment or not) of the exponent cannot be done until mantissa multiplication is completed. That is, this operation is on the critical path of the floating point multiplication.

High-speed operation in the exponent adder part is achieved by selecting one out of two previously prepared values: $Ex + Ey - 127$ and $Ex + Ey - 127 + 1$. The exponent adder consists of an 8-bit adder performing the addition of $Ex$, $Ey$, and $-127$, an incrementer incrementing the output of the 8-bit adder, and an 8-bit 2-to-1 data selector. The output of the incrementer is the exponent of product $Z$ in the case of a product where the mantissa

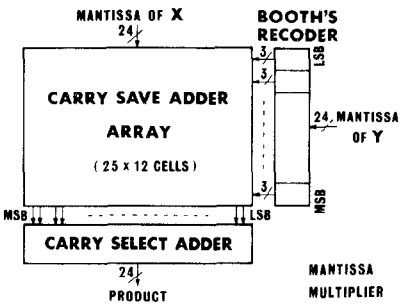



Fig. 2. Block diagram of the mantissa multiplier.

multiplier is equal or greater than 2 (normalizing is required). As soon as the multiplication of the mantissa multiplier is completed, the data selector selects out a true exponent $Ez$ between the output of the 8-bit adder and that of the incrementer. Thus, the chip can perform the normalization in only 5 gate delays.

## IV. Mantissa Multiplier Design

A block diagram of the mantissa multiplier is shown in Fig. 2. The mantissa multiplier performs the $24 \times 24$-bit multiplication of two two's complement binary numbers (in fixed point mode) or two unsigned magnitude binary numbers consisting of 23-bit fraction and implicit leading "1" (in floating point mode). High-speed operation is achieved by the following techniques.

A modified Booth's algorithm reduces the number of partial products to half. The 24-bit mantissa of multiplicand $Y$ is recoded into 12 sets of 3-bit codes by twelve Booth's recoders, so that the $24 \times 24$ multiplication becomes $25 \times 12$ recoded multiplication, and the number of partial products is reduced from 24 to 12. The critical paths of the Booth's recoder and decoder (a partial product generator) are 3 and 3 gate stages, respectively.

A carry save adder (CSA) scheme is employed. The CSA array in Fig. 2 is made up of 25 columns by 12 rows, and performs the high-speed addition of the 12 partial products. The longest path of propagation in the CSA array is 12 CSA's. Each terminal in the CSA array is interconnected so that the propagation delay time on the critical path should be the shortest. The final carry propagation for the lower 24 bits of the product is packed into the array by using twelve high-speed 2-bit carry generators.

A high-speed CMOS full adder is used in the CSA array. Fig. 3 shows the circuit diagram of the CMOS full adder. This full adder consists of 40 transistors and generates the sum and carry output in only 3 and 2 gate delays including the output buffer, respectively. Thus, the full adder can perform addition at about twice the speed of conventional CMOS full adders. By using the full adder, the critical path of the array made up of 12 CSA's is reduced to 36 gate stages.

The physical design of the full adder cell used in the CSA array is one of the most important aspects because its performance greatly influences the operating speed, total power dissipation, and chip size. Full CMOS construction

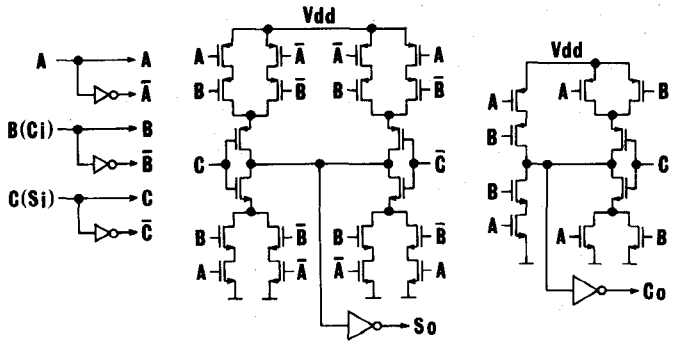



Fig. 3. Circuit diagram of the CMOS full adder.

is employed to realize the low-power dissipation. Wiring resistances and parasitic capacitances are minimized by optimum pattern designing. As a result, the array cell of the CSA array is $139\times275$ $\mu m^2$ in size, including the full adder, the partial product generator (a 2-1 multiplexer and an Exclusive-OR gate), and the interconnection area. A photograph of the array cell is shown in Fig. 4. This array cell can perform the addition in 3.7 ns, typically, as mentioned later.

A modified carry select adder speeds the 24-bit addition in the final adder stage, and is shown in Fig. 5. This adder is made up of five block adders and three block-carry generators (two OR-NAND gates and an AND-OR gate). Each $n$-bit block adder has two $n$-bit ripple adders (one and the other provide addition assuming that the carry input is "1" and "0," respectively), and an $n$-bit 2-to-1 multiplexer.

When the partial product addition in the CSA array is completed, all of the ripple adders start operating simultaneously. The carry output of the $n$-bit ripple adder is generated after $n$ gate delays. In our circuit design, we made the carry propagation time of the block-carry generator equal to that of a "one-bit" ripple adder (1 gate delay). By designing an $n+1$-bit block adder next to the $n$-bit block adder, two carry outputs of the ripple adders in the $n+1$-bit block adder are generated at the same time as an output of the block-carry generator in the $n$-bit block adder. For example, in Fig. 5, $C_{19}^{1}$ and $C_{19}^{0}$, $\overline{C_{13}^{1}}$, and $\overline{C_{13}^{0}}$ are generated at the same time as $C_{13}$ and $\overline{C_8}$, respectively. That is, the higher the order of a block adder, the larger the number of bits of it can be made. For this reason, 4-bit, 5-bit, and 6-bit ripple adders are used for the second, third, and fourth block adders, respectively, so that every data path becomes critical. For instance, $A_0-C_4-\overline{C_8}-C_{13}-\overline{C_{19}}-S_{23}$, $B_8-C_{13}-\overline{C_{19}}-S_{23}$, and $A_{13}-\overline{C_{19}}-S_{23}$ are all critical.

As mentioned above, optimizing the distribution of bits to each block adder makes it possible to reduce the number of block adders and minimize the critical path. This adder reduces the 24-bit addition time to 10 gate delays including the propagation delay time (3 gate delays) of the 2-1 multiplexer.

Besides these high-speed techniques, careful attention was paid to the arrival sequence of input signals in interconnecting gates.

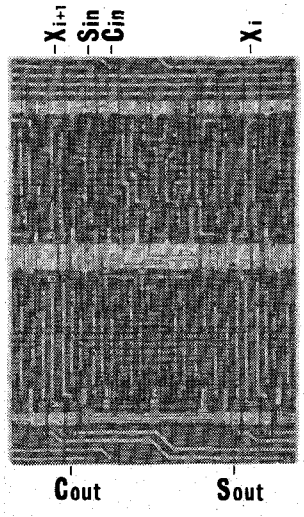



Fig. 4. Photograph of the array cell in the carry save adder array. The cell contains the CMOS full adder and the partial product generator consisting of a 2-1 multiplexer and an Exclusive-OR gate. The cell size is $139\times275$ $\mu m^2$ (repeatable).

Consequently, the propagation delay on the critical path of the mantissa multiplier is reduced to 52 gate delays.

## V. Characteristics

A photograph of the chip is shown in Fig. 6. The $24\times24$-bit mantissa multiplier has an effective area of $3.8\times3.8$ $mm^2$. The twelve Booth's recoders are at the right side of the chip. The exponent adder and other control logic are to the left of that.

The chip was fabricated using a conventional 2 $\mu$m n-well CMOS process with single level poly-Si and Al. The gate length is 2.0 $\mu$m for the n-channel, and 2.25 $\mu$m for the p-channel transistor. The gate oxide thickness is 400 Å. The threshold voltage is 0.6 V for the n-channel, and $-0.6$ V for the p-channel transistor.

Fig. 7 shows the multiply time as the sum of individually measured propagation delay times on the critical path.

The addition time of the CMOS full adder measured 3.7 ns under normal operating conditions ($V_{DD}=5$ V, $T=25$°C). This value coincided well with a simulated value of 3.66 ns. Therefore, the critical path of the carry save adder array required 44.1 ns using the CMOS full adders.

A typical addition time for the modified carry select adder measured 18 ns.

As a result, we obtained a typical value of 71.1 ns as the 24-bit two's complement fixed multiply time, and a value of 78.7 ns as the 32-bit floating point multiply time.

We can measure the multiply time including the propagation delay time of an output register and two gates by using the enable control *ENB*. The propagation delay time is equivalent to about 4 gate delays, namely, about 9 ns. In the 24-bit two's complement fixed point multiplication, 80 ns is obtained as the measured value by using *ENB* in the typical condition. The characteristics of multiply time versus supply voltage at room temperature ($T=25$°C) are shown in Fig. 8.

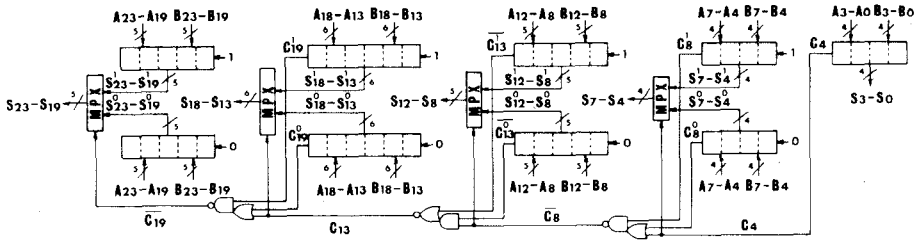



Fig. 5. Circuit diagram of the modified carry select adder. The adder can perform 24-bit addition in the final addition stage in only 10 gate delays. The typical addition time measured 18 ns.

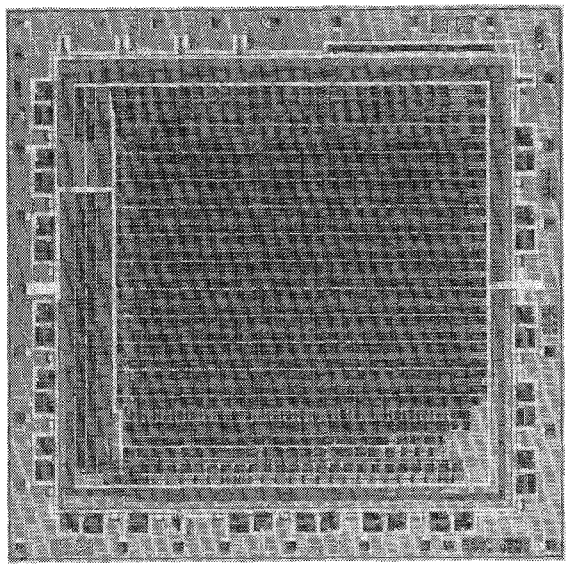


Fig. 6. Photograph of the multiplier chip. The chip size is $5.75 \times 5.67$ mm$^2$.

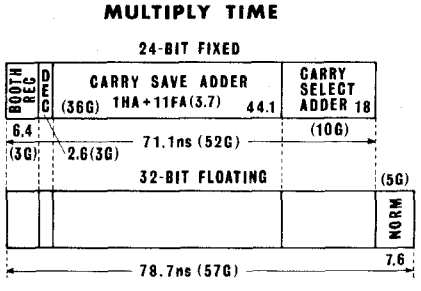



Fig. 7. Multiply time as the sum of individually measured propagation delay times on the critical path.

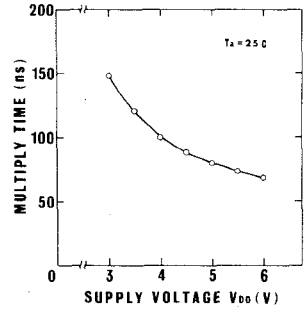



Fig. 8. Characteristics of multiply time versus supply voltage at room temperature ($T = 25$°C).

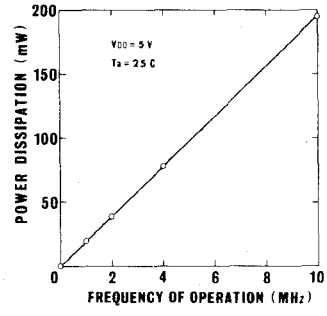



Fig. 9. Characteristics of power dissipation versus frequency of operation in the typical condition ($V_{DD} = 5$ V, $T = 25$°C).

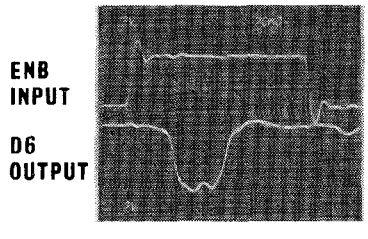



Fig. 10. Oscillograph of the waveforms of *ENB* input and *D*6 (MSB of a fraction *Fz*) output. An interval time from a falling edge to a rising edge of the *D*6 output waveform represents the multiply time.

The characteristics of power dissipation versus frequency of operation at $V_{DD} = 5$ V and $T = 25$°C are shown in Fig. 9, and represent the typical power dissipation of 195 mW at 10 MHz (10 million operations per second).

Waveforms of *ENB* input and *D*6 (MSB of a fraction *Fz*) output are shown in Fig. 10. This oscillograph is an example of 32-bit floating point multiplication of the case that the multiplier $X = 44400001H$ ($Sx = 0$, $Ex = 88H$, $Fx = 400001H$), the multiplicand $Y = 83FFFFFFH$ ($Sy = 1$, $Ey = 07H$, $Fy = 7FFFFFH$), $RND = 0$, and the product $Z = C8400000H$ ($Sz = 1$, $Ez = 90H$, $Fz = 400000H$).

When the *ENB* input is low, an input of the output register (this is an output of the mantissa multiplier) is low. As the *ENB* input goes high, the "low" signal propagates from the input of the output register to the *D*6 output through the output register, the format adjuster, an internal output bus, and an output driver. At the same time, the mantissa multiplier starts the multiplication. After the operation, the MSB of the fraction of the product *Z* goes high. This "high" signal propagates to the *D*6 output as well as the "low" signal. Therefore, the time interval from a falling edge to a rising edge of the *D*6 output waveform in Fig. 10 represents the multiply time including the propagation delay time of the output register and two gates. In this case, we can read a value of about 40 ns out of the Fig. 10 oscillograph.

```
*********  00000000
*********  00FFFF80
30.00NS    00000000001100000000001000110100
31.00NS    00000000011010100000001000111000
32.00NS    00000000001010100000001000111100
33.00NS    00000000001000100101010111100
34.00NS    00000000100100010010111010111101
35.00NS    00000000100100010111111010111101
36.00NS    00000000100100011111111010001101
37.00NS    00000000001000000101000100011101
38.00NS    00000000001000000101000100011101
39.00NS    00000000001000000000000000001111
40.00NS    00000000001000000000000000001111
41.00NS    00000000001000000000000000001111
42.00NS    00000000001000000000000000001111
43.00NS    00000000000000000000000000001111
44.00NS    00000000000000000000000000001111
45.00NS    00000000000000010000000000011111
46.00NS    00000000000000010000000000011111
47.00NS    00000000000000010000000000011111
48.00NS    00000000000000010000000111111111
49.00NS    00000000000000010000000111111111
50.00NS    00000000000000011000000111111111
51.00NS    00000000000000011100000111111111
52.00NS    00000000000000011111000011111111
53.00NS    00000000000000011111000011111111
54.00NS    00000000000000011111000011111111
55.00NS    00000000000000000000001111111110
56.00NS    00000000000000000000001111111110
57.00NS    00000000000000000000001111111100
58.00NS    00000000000000000000001111111100
59.00NS    00000000000000000000001111111000
60.00NS    00000000000000000000001011111000
61.00NS    00000000000000000000011111110000
62.00NS    00000000000000000001111111100000
63.00NS    00000000000000000001111111100000
64.00NS    00000000000000000011111111100000
65.00NS    00000000000000000011111111100000
66.00NS    00000000000000000111110000000000
67.00NS    00000000000001111111111000000000
68.00NS    00000000000011111111111000000000
69.00NS    00000000111111111110000000000000
70.00NS    00000000111110000000000000000000
71.00NS    00000000000000000000000000000000
72.00NS    00000000000000000000000000000000
73.00NS    00000000000000000000000000000000
74.00NS    00000000000000000000000000000000
75.00NS    00000000000000000000000000000000
76.00NS    00000000000000000000000000000000
77.00NS    00000000000000000000000000000000
78.00NS    00000000000000000000000000000000
79.00NS    00000000000000000000000000000000
80.00NS    00000000000000000000000000000000
                Bit-31—      Bit-23—      Bit-0—
Enable Time
```

Fig. 11. 32-bit output patterns versus enable times.

Fig. 11 shows 32-bit output patterns versus enable times. This is measured and printed out by an LSI tester. The enable time signifies that the time during the enable signal *ENB* is high, namely, an allotted time for multiplication. The 32-bit output patterns are the data latched into the output register $Z$ by the falling edge of the *ENB* signal, that is, the halfway result of the multiplication for the enable time. This is the case of $00000000_H \times 00FFFF80_H = 00000000_H$ in 24-bit two's complement fixed point mode. The higher 8 bits (bit-31–bit-24) are always "0"'s in the fixed point format. "1" and "0" in the output patterns represent the wrong and right answer, respectively.

Fig. 11 shows that the first right answer of bit-0 (LSB) is obtained at 55 ns, and that of bit-23 (MSB of 24-bit two's complement result) at 71 ns. The first, second, third, fourth, and fifth block adder of the modified carry select adder put the first right answers at 62, 66, 69, 70, and 71 ns, respectively. As a result, the modified carry select adder stage takes 16 ns (71–55 ns), and the multiply time (including the propagation delay time of the output register and two gates) is 71 ns.

TABLE I
SUMMARY OF THE CHIP FEATURES

| | |
|---|---|
| PROCESS | 2 μm N-well CMOS |
| GATE LENGTH | 2.0 μm (N-ch)<br>2.25 μm (P-ch) |
| TRANSISTOR COUNT | 23,000 Tr |
| CHIP SIZE | 5.75 x 5.67 mm² |
| PACKAGE | 28-pin DIP |
| INTERFACE | 16-bit BUS |
| LOGIC LEVEL | TTL Compatible |
| SUPPLY VOLTAGE | 5 V |
| MULTIPLY TIME | 71.1 ns (24x24 FIXED)<br>78.7 ns (32x32 FLOATING) |
| POWER DISSIPATION | 195 mW (10 MFLOPS) |

## VI. SUMMARY

A high-speed, low-power CMOS floating point multiplier chip has been developed. The chip can perform 32-bit floating point and 24-bit two's complement fixed multiplication in typically less than 78.7 and 71.1 ns, respectively. The above high performance is obtained by high-speed multiplication techniques including the modified Booth's algorithm, a carry save adder scheme, a high-speed CMOS full adder, and a modified carry select adder. The typical power dissipation is 195 mW at 10 million operations per second.

The chip is designed for 16-bit bus compatibility and can be used as a floating point accelerator for microcomputer systems. Furthermore, the design techniques used are applicable to on-chip multipliers and other arithmetic units.

The features of the chip are summarized in Table I.

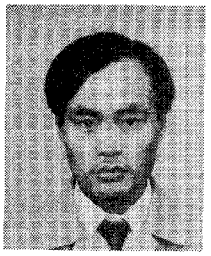
## REFERENCES


[1] A. D. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. 4, part 2, pp. 236–240, 1951.
[2] O. J. Bedrij, "Carry select adders," *IRE Trans. Electron. Comput.*, vol. EC-11, no. 3, pp. 340–346, June 1962.
[3] K. Hwang, *Computer Arithmetic.* New York: Wiley, 1979.
[4] D. Stevenson, "A proposed standard for binary floating point arithmetic," *IEEE Trans. Comput.*, vol. C-14, no. 3, pp. 51–62, Mar. 1981.
[5] F. Ware, W. McAllister, J. Carlson, D. Sun, and R. Vlach, "64 bit monolithic floating point processors," *IEEE J. Solid-State Circuits*, vol. SC-17, no. 5, pp. 898–907, Oct. 1982.
[6] B. Woo, L. Lin, and R. Owen, "ALU, multiplier chips zip through IEEE floating-point operations," *Electronics*, vol. 56, no. 10, pp. 121–126, May 1983.
[7] M. Uya, K. Kaneko, and J. Yasui, "A CMOS floating point multiplier," in *ISSCC Conf. Dig.*, Feb. 1984, vol. 27, pp. 90–91.



**Masaru Uya** was born in Shimane, Japan, on January 15, 1950. He received the B.S. degree in electrical engineering from Osaka University, Osaka, Japan, in 1972.

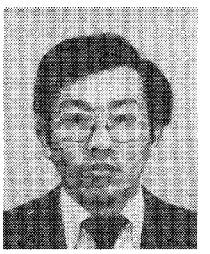
In 1972 he joined the Matsushita Electric Industrial Co., Ltd., Osaka, Japan. From 1972 to 1980 he was engaged in the research and development of an electronic music synthesizer in the Wireless Research Laboratory at Matsushita. Since 1980 he has been involved in the research of high-speed CMOS circuit design, and the development of high-speed CMOS LSI circuits such as 16-bit CMOS microprocessor at Matsushita's Central Research Laboratory. His research interests include computer arithmetic, computer architecture, high-speed numerical computation techniques, and LSI circuit design.



**Katsuyuki Kaneko** was born in Nagano, Japan, on July 21, 1955. He received the B.S. and M.S. degrees in communication engineering from Tohoku University, Sendai, Japan, in 1978 and 1980, respectively.

He joined the Matsushita Electric Industrial Co., Ltd., Osaka, in 1980. He has been working on the design of MOS LSI circuits.

Mr. Kenko is a member of the Institute of Electronics and Communication Engineers of Japan.



**Juro Yasui** received the B.S. and M.S. degrees in solid-state engineering from Osaka University, Osaka, Japan, in 1969 and 1971, respectively.

In 1974 he joined the Semiconductor Research Laboratory, Matsushita Electric Industrial Co., Ltd., Moriguchi, Osaka, Japan, where he has been working on the development of integrated circuit processing.

# A Radix 4 Delay Commutator for Fast Fourier Transform Processor Implementation

EARL E. SWARTZLANDER, JR., SENIOR MEMBER, IEEE, WENDELL K. W. YOUNG, MEMBER, IEEE, AND SAUL J. JOSEPH, MEMBER, IEEE

*Abstract* — **This paper describes the development of a semicustom delay commutator circuit to support the implementation of high speed fast Fourier transform processors based on the McCellan and Purdy radix 4 pipeline FFT algorithm. The delay commutator is a 108000 transistor circuit comprising 12288 shift register stages and approximately 2000 gates of random logic realized with 2.5 micrometer design rule CMOS standard cell technology. It operates at a 10 MHz clock rate which processes data at a 40 MHz rate. The delay commutator is suitable for implementing processors that compute transforms of 16, 64, 256, 1024, and 4096 (complex) points. It is implemented as a 4 bit wide data slice to facilitate concatenation to accomodate common data word sizes and to use a standard 48 pin dual-in-line package.**

## I. INTRODUCTION

ALTHOUGH the Cooley–Tukey FFT algorithm [1] developed nearly two decades ago has made it possible to apply digital signal analysis techniques to many applications, many others (e.g., radar and sonar beam forming, adaptive filtering, communications spectrum analysis, etc.) require both flexibility and speed that exceeds the present state of the art. Currently, there are three approaches for signal processing: software implemented on general purpose computers, software implemented on a general purpose computer augmented with a Programmable Signal Processor (PSP), and custom hardware development. Software only and Software-PSP implementations are adequate when the spectral bandwidth is under 10 MHz. Custom processors achieve analysis bandwidths of 10–50 MHz, but most are optimized for a specific application and would require extensive (and expensive) redesign to modify them to suit other applications. Thus general purpose computers with or without PSP augmentation are too slow while custom processors lack the required flexibility.

Current signal processing systems require many diverse functions: transform computation, time and frequency domain vector processing, and general purpose computing. We are developing a growing family of building block modules to facilitate the development and implementation of such systems on a semicustom basis. The result is the ability to quickly develop high performance signal processing systems for a wide variety of algorithms. The use of predesigned and precharacterized modules reduces cost, development time, and most importantly, risk. The initial set of modules was described in 1983 [2]. The modules defined include a data acquisition module, building block elements that are replicated to realize pipeline FFT and inverse FFT modules, a frequency domain filter module, a power spectral density computational module, and an output interface module.

The modules all have separate data and control interfaces. The separation of the data and control is analogous to the Harvard mainframe computer architecture which uses separate data and instruction memories to eliminate

Manuscript received April 11, 1984; revised June 14, 1984.

E. E. Swartzlander, Jr. and W. K. W. Young are with TRW Defense Systems Group, Redondo Beach, CA 90278.

S. J. Joseph is with AT&T Bell Laboratories, Allentown, PA 18103.

0018-9200/84/1000-0702$01.00 