A GaAs IEEE Floating Point Standard Single Precision Multiplier

S. Cui, N. Burgess, M. Liebelt & K. Eshraghian†

Dept. of Electrical and Electronic Eng. & †Dept of Electronics, Computer and Communication Eng.
The University of Adelaide, SA 5005, Australia & †Edith Cowan University, WA 6027, Australia

Abstract

This paper presents a GaAs IEEE floating point standard single precision multiplier. A modified carry save array is used in conjunction with Booth's algorithm to reduce the partial product addition and interconnection. A special rounding technique called Trailing-1's Predictor is used to speed up the final addition and rounding. The combination of the fast arithmetic architecture and compact layout style achieves 4ns multiplication time with 3.5W power dissipation at 75°C giving 14 mW/MHz. The area is 2.43mm by 3.77mm (excluding pads) and uses 28,000 transistors to give a density of 3056 transistors/mm² for 0.8-μm GaAs technology.

1 Introduction

High-speed and high-precision computation are required for many digital signal processing, computer graphics, model simulation and image processing applications. Floating point computation is most suitable for these applications because it maintains high precision operation over a wide dynamic range. For microcomputer systems, not only high-speed but also small-sized floating point devices are in great demand.

Usually floating point chips are designed using silicon-based technology [1]-[3]. Advances in fabrication processes have significantly improved the performance of silicon devices [4, 5]. However, GaAs has inherent superiority in electron mobility and saturation velocity, high-temperature operation, and radiation hardness. In a research and development environment, GaAs digital circuits have clearly a better power-delay performance than silicon circuits [6]. In recent years GaAs ICs have become increasingly attractive due to their high speed. Also, the yield of GaAs ICs has increased dramatically. However, the relatively low layout density compared to CMOS has tended to limit the utilization of GaAs VLSI circuits.

In this paper, we describe a fast, compact 32-bit GaAs floating point multiplier using a new design and layout strategy which takes advantage of the superior performance of GaAs and improves the layout density.

A modified carry save array is used in conjunction with Booth's algorithm to reduce the partial product addition and interconnection. A special rounding technique called Trailing-1's Predictor is used to speed up the final addition and rounding. The chip supports IEEE standard 754 single precision format and provides overflow/underflow flags. The chip measures *2.43×3.77mm* and contains *28,000* transistors. The simulated result of the multiplication time for the chip is *4ns*, with *3.5W* power dissipation at *75°C*.

2 Chip Architecture

A block diagram of the floating point multiplier is shown in Figure 1. The chip consists of an exponent block and a mantissa multiplier block. The exponent block evaluates the sign and exponent in floating point multiplication mode, and detects overflow/underflow.

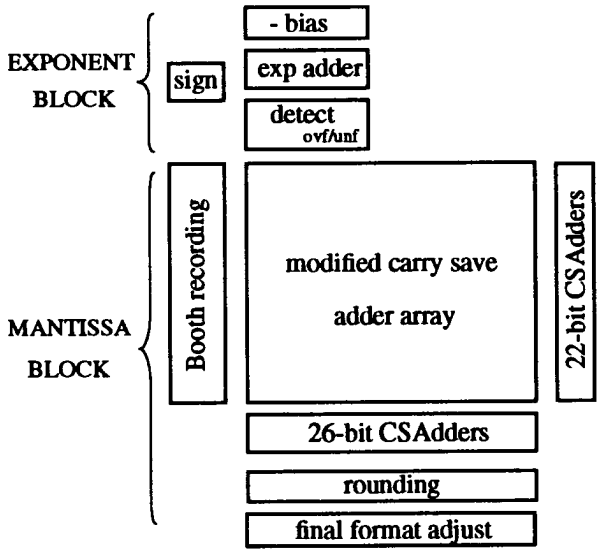

Figure 1. Architecture of the floating point multiplier.

2.1 The Multiplier Array

The mantissa multiplier is a 24×24-bit parallel multiplier which consists of two parts: an array and a final adder. The array uses the radix-4 modified Booth's algorithm to reduce the number of partial products from 24 to 13.

After the partial products have been generated, there are other techniques which can further improve the speed of multipliers, of which Wallace tree reduction is one of the fastest. In general, the number of stages required to reduce n partial products to 2 is [7]:

$$T = \log_{1.5}(n/2) = 2.4664 \ln(n/2) \quad \text{(EQ. 1)}$$

For our case, $n=13$, $T=4.6$. Therefore, at least 5 stages are needed. For Wallace tree reduction, each stage has one full adder. This means that for IEEE single precision format with modified Booth's encoding, the partial-product tree height reduces from 11 to 5 full adders. However, the total delay depends not only on the number of stages T, but also on the delay associated with the interconnection wiring capacitance. Since Wallace tree reduction is not regular, the delay of the interconnection is significant and the method is not suitable for VLSI implementation. There are other reduction schemes which use (n, m) compressors with n inputs and m outputs [7]. A full adder is a (3, 2) compressor. Other useful compressors are (4, 2) and (7, 3) [8, 9]. Figure 2 shows a 24×24-bit modified Booth multiplier using (4, 2) compressors. It can be seen that it needs 3 stages i.e. it is 6 full adders deep, but the layout regularity is still compromised, which is a disadvantage in ultra high-speed GaAs technology.

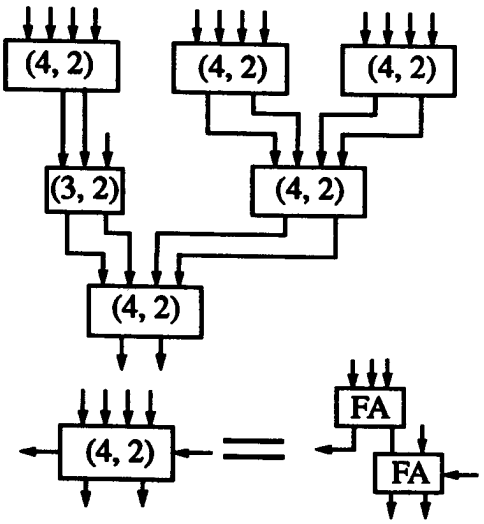

Figure 2. A 24×24-bit multiplier using (4, 2) compressors.

In order to maintain both regularity and high speed, a modified carry save array [10] was used for our multiplier. Figure 3 shows the structure of this method which adds even rows with even rows, odd rows with odd rows and at the end adds the two rows together. The partial product reduction tree is 7full adders deep for 24×24-bit multiplication, which is comparable to a multiplier using (4,2) compressors (6 full adders deep) while using a simple and regular structure, necessary for minimizing delays due to interconnect in GaAs technology. Figure 4 shows the HSPICE simulated results of the critical path of these two multipliers. The modified carry save method is slightly faster than (4, 2) compressor method because of shorter interconnection wires. The (4, 2) compressor also has larger power dissipation due to larger buffers required to drive longer wires. Therefore, mCS is preferred due to faster, easier layout and lower power. However, for double precision multiplication, the (4, 2) multiplier is faster than the modified carry save multiplier, because the former is 8 full adders deep, whereas the latter is 14 full adders deep.

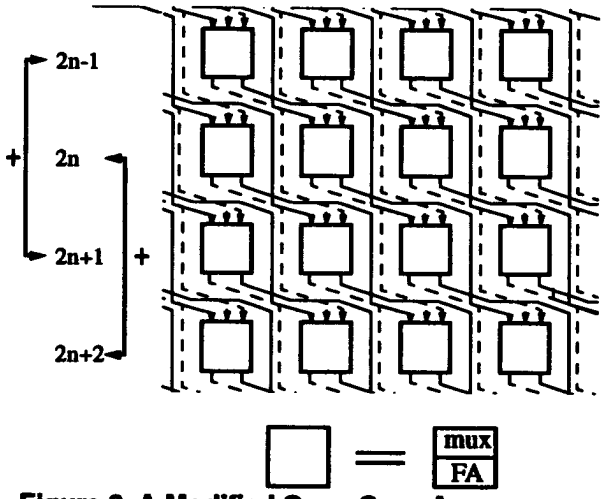

Figure 3. A Modified Carry Save Array.

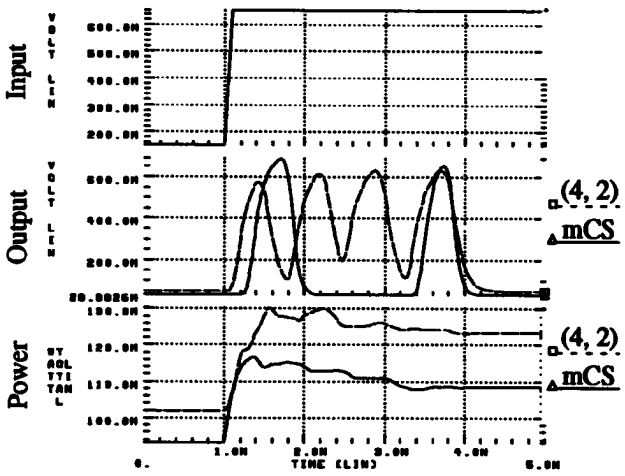

Figure 4. The HSPICE simulated results of two multipliers.

2.2 The Final Adder

To obtain maximum speed, both the array and the final adder part must operate at the highest possible speed. In GaAs technology all of the logic functions have to be implemented by inverters and NOR gates because of noise margin problems [11]. Pass transistor logic is also restricted in GaAs. Sarmiento *et al.* studied a variety of adders and concluded that the Binary Carry Look-ahead adder (also called Brent and Kung adder) and carry-select adder are the fastest adders in GaAs [12]. However, the Brent and Kung adder contains long wires which require buffing.

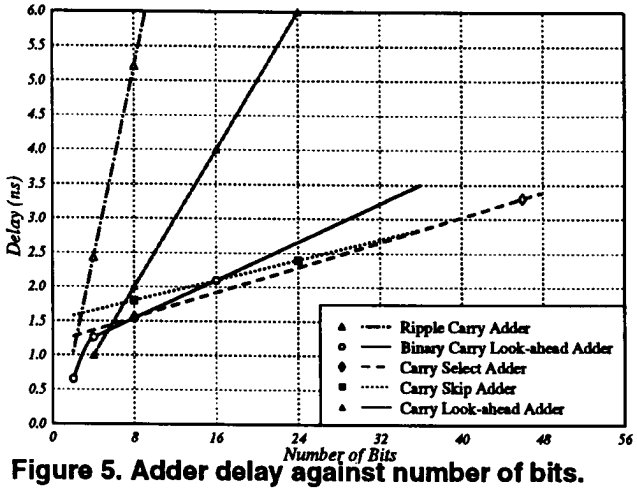

Figure 5. Adder delay against number of bits.

Figure 5 shows our simulated result of adder delay against number of bits. It confirms Sarmiento's results and shows that for adders exceeding 8 bits in width, the carry select adder as shown in Figure 6(a) is the fastest adder with moderate area. Each block consists of a pair of ripple carry adders and successive blocks increase in length by one bit. However the number of *m-bit* ripple carry adders must be chosen carefully. For *n-bit* addition, the total delay of the adder of Figure 6(a) can be calculated roughly as follows:

$$d_{total} = m \cdot d_{carry} + \left(\frac{n}{m} - 1\right) \cdot d_{mux} \qquad \text{(EQ. 2)}$$

where d_{carry} is the carry in to carry out delay of a full adder, d_{mux} is the delay of a multiplexer. Our simulated results show that $d_{carry} = 0.25ns$ and $d_{mux} = 0.3ns$. For *24×24-bit* multiplication, $n=48$. m may be found by differentiating EQ. 2 with respect to m:

$$\frac{\partial}{\partial m}(d_{total}) = d_{carry} - \frac{n}{m^2} \cdot d_{mux} = 0 \qquad \text{(EQ. 3)}$$

$$m = \sqrt{\frac{d_{mux}}{d_{carry}} \cdot n} \approx 7.6 \qquad \text{(EQ. 4)}$$

$$d_{total} = d_{min} = 3.5ns \qquad \text{(EQ. 5)}$$

The relationship between d_{total} and m is shown in Figure 7. Figure 6(b) shows the final adder partitioning used in our 32-bit floating point multiplier

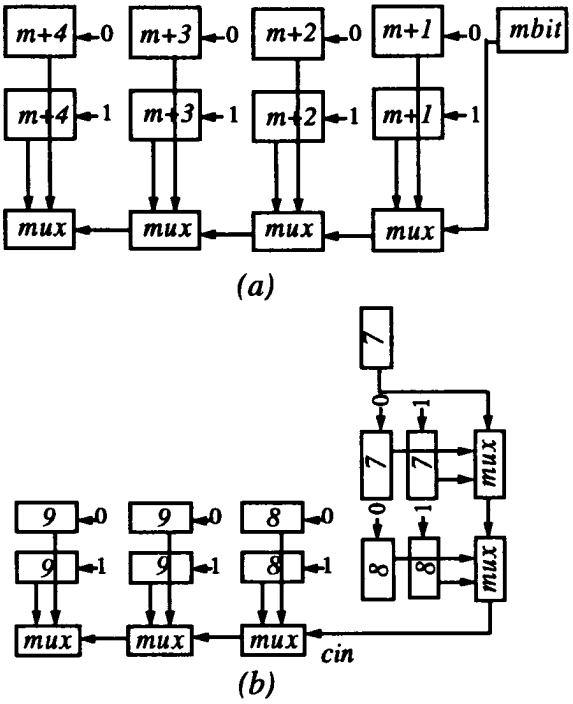

Figure 6. The structure of the final adder.

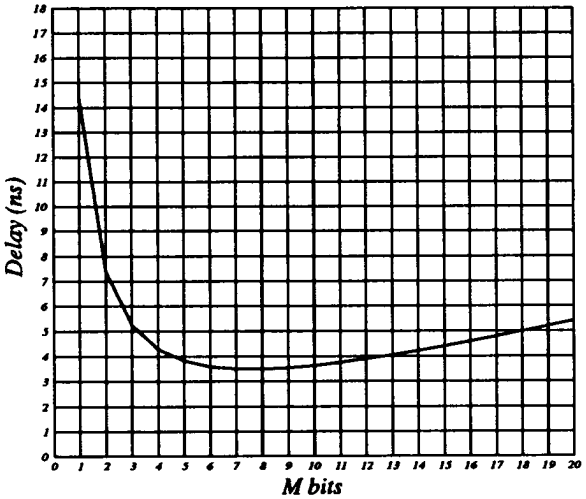

Figure 7. The total delay vs. *m* bit.

3 Rounding

Much delay in a floating-point multiplier may be taken up by rounding the product in accordance with the IEEE standard. Methods for speeding-up the rounding process

centre on using a pair of full-length adders to supply the unrounded product, P, and an incremented product, $P + 2^{-23}$. Then the correctly rounded result may be chosen by interpolation across the two results [13]. In GaAs technology, introduction of a second full-length adder to achieve rounding is overly wasteful of the limited available chip area and so alternative techniques are required that use less circuitry (and consume less power) but which are still capable of accelerating the rounding process.

A simple algorithm for incrementing a number consists of inverting bits in the number from the l.s.b. up to and including the first zero. Hence, instead of employing a second adder to compute $P+2^{-23}$, a trailing-1's predictor would examine the pair of addends to generate a set of flag bits indicating those bits that should be inverted in order to increment P. A simple T1P, based on a carry-ripple principle, is shown in Figure 8.

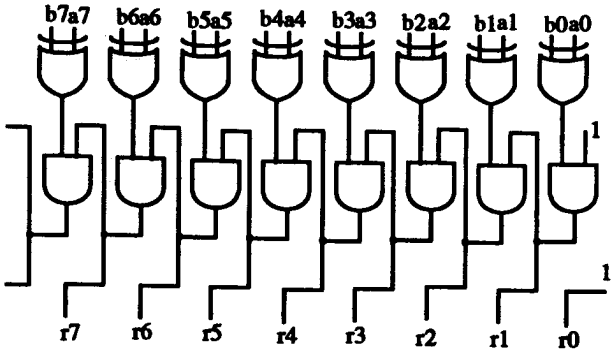

Figure 8. A simple 8-bit carry-ripple T1P.

Conceptually, the structure of the Trailing-1's Predictor rounding technique is illustrated in Figure 9, where the output of T1P, r_i, is a set of flag bits indicating that those sum bits that should be inverted. The control logic activates the T1P by allowing the r_i bits onto the XOR gate inputs if P is to be incremented. If P is not to be incremented, all the r_i bits are forced to zero, so that $p_i = s_i$.

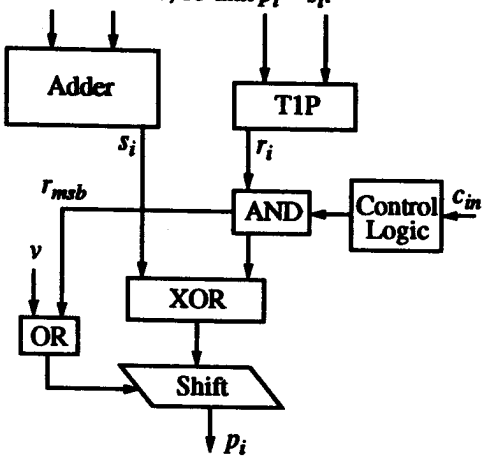

Figure 9. The structure of T1P rounding technique.

In order to discuss the rounding procedure clearly, Figure 10 shows the top 26 bits of the product, produced in this multiplier by the three most significant blocks of the carry-select adder. The square brackets "[]" indicate the bits to be occupied by the correctly-rounded result.

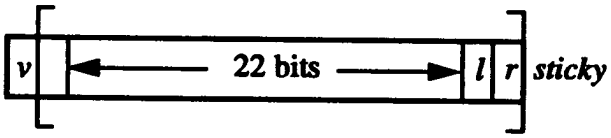

Figure 10. Labelling of bits in product P.

In the case of no overflow, 2^{-24} will be added at the r bit position. If an overflow occurs, 2^{-23} will be added at the l bit position. The T1P is designed to operate from bit position l (at significance 2^{-23}) upwards to accommodate rounding by adding 2^{-23}. Rounding by adding 2^{-24} is then achieved by activating the T1P if $\bar{v}\cdot\{c_{in}\cdot\bar{r} + \bar{c}_{in}\cdot r\cdot s + l\cdot r\cdot\bar{s}\}$ and r are both high, and inverting r in any case. Finally, the act of rounding up may itself cause an overflow, but this event would be detected by the most significant flag bit produced by the T1P going high, which could then provide a right shift signal in concert with the overflow bit, v.

Other faster, more sophisticated T1P's are available based on other accelerated addition techniques. We have found that a carry-select version of the T1P is preferred in GaAs technology for the same reasons as were discussed in relation to the choice of adder. This structure is illustrated in Figure 11.

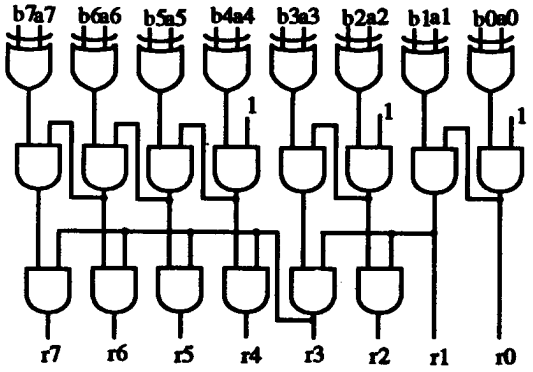

Figure 11. An 8-bit carry-select T1P.

Detailed layout and simulation showed that the T1P was 25% smaller and 30% quicker than using one row of half adders and a second carry-select adder to perform rounding. The T1P can share its *xor* gates within the CSAdder. Thus, the T1P method occupies less area than CSA method. Furthermore the T1P method implements round to nearest/even directly, thus no round to nearest/up to round to nearest/even conversion needed.

4 Circuit Design

Direct Coupled FET Logic (DCFL) was used to implement the logic because it is the simplest logic class in GaAs with reasonable speed and power dissipation [15]. Source-Follower Direct Coupled FET Logic (SDCFL) was used to drive large loads and long wires. Figure 12(a) and (b) show a DCFL inverter and a SDCFL 2-input nor gate, respectively.

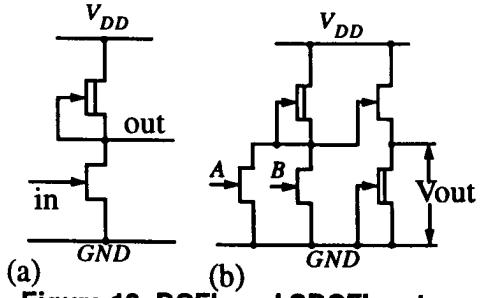

Figure 12. DCFL and SDCFL gates.

Because all the logic functions have to be implemented using inverters and NOR gates, the size of the layout is quite large, especially the multiplexer. In order to reduce the area of the chip, we used a "Ring Notation" approach to implement the circuits. The "Ring Notation" approach (Figure 12(b)) showed that it has better performance and is easier to implement than the nMOS-style (Figure 13(a)) layout for high-speed VLSI circuits [15].

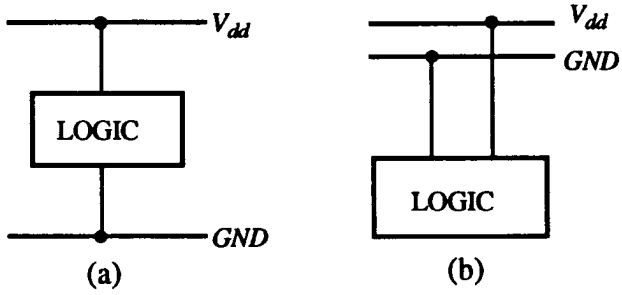

Figure 13. (a) nMOS-style and (b) Ring-style.

5 Simulated Results and Conclusion

The multiplier was designed using Vitesse H-GaAs-II 0.8μm Technology. The SPICE simulated result of the 32-bit multiplier for worst case is shown in Figure 14. The layout of the chip (without pads) corresponding to Figure 1 is shown in Figure 15. The chip measures *2.43×3.77mm²* and contains *28,000* transistors. The simulated result of the worst case multiplication time for the chip is *4ns* at *typical-typical* parameters with *100fF* load. The power dissipation is 3.5W at *75°C* with *2V* power supply giving *14mW/MHz*.

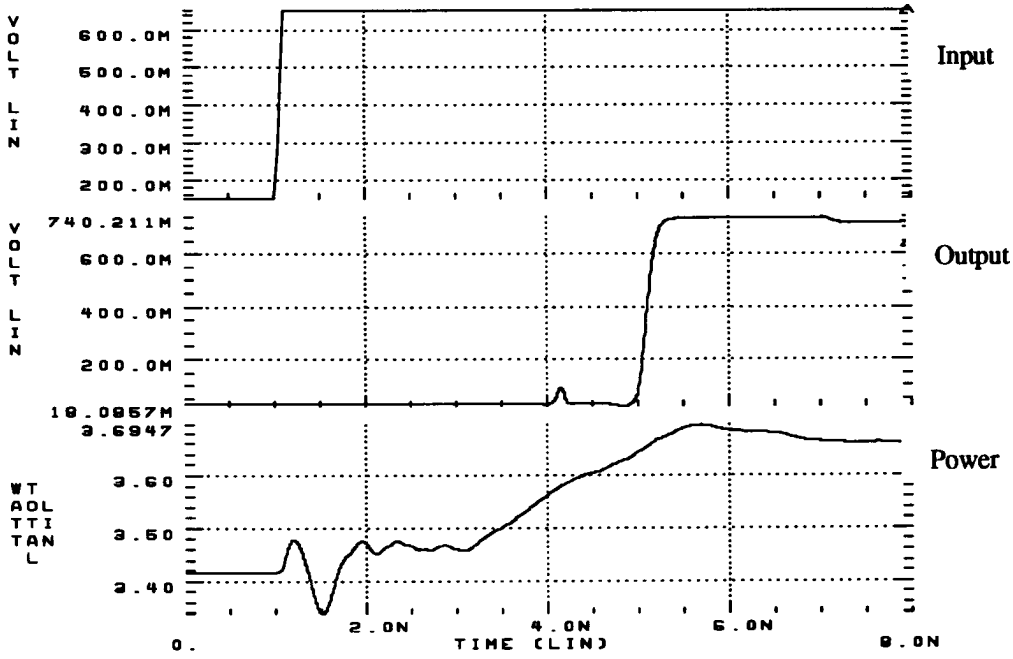

Figure 14. The HSPICE simulated results of the floating point multiplier.

Table 1. The performance of 32-bit multipliers

	Technology	Format	Method	Delay *ns*	Area mm^2	Transistors	Density $/mm^2$	Power *W*	mW/ MHz
AT&T	GaAsHFET 1μm	IEEE32	(4, 2)	9.25	8×9.5	49,700	654	7	74.75
U of A	GaAsMESFET 0.8μm	IEEE32	m. CS	4	2.43×3.77	28,000	3056	3.5	14

Table 2. The Performance of 16×16 Multipliers

Authors	Technology	Method	Delay *ns*	Area mm^2	FETs	Density $/mm^2$	Power *W*	mW/MHz
Sekiguchi Sumitomo(1990)	GaAs (0.7μm)	m. Booth, WT multi-chip	7.6	4 X 3.75 X 3.75	4 X 3906	278	4 X 1.1	33.4
Singh et al ITT(1990)	GaAs (0.7μm)	CS	4.75	3.2 X 2.8	11708	1307	2.61	12.4
Kajii et al (1988)	GaAs HEMT		4.1	6.3 X 4.8	15000	496	6.2	25.4
Cui et al U of A (1994)	GaAs (0.8μm)	m. Booth & m. CS	2.93	1.4 X 2.5	11562	3300	1.44	4.2

Table 1 shows the comparison of our 32-bit multipliers (U of A) with the AT&T GaAs heterojunction FET 32-bit multiplier [16] which is the only other reference in this field. In order to compare with other GaAs technologies, a 16×16-bit fixed point multiplier has also been designed. Table 2 shows the comparison of our 16×16-bit multiplier with other 16×16-bit multipliers. It can be seen that our chip has better performance in terms of delay, area and power dissipation. It suggests that the modified carry save array and the trailing-1's rounding technique work very well for a GaAs multiplier. To confirm this architecture, the 16×16-bit parallel multiplier is being fabricated to verify our simulation results. If successful, the floating point multiplier will also be fabricated.

6 Acknowledgements

The work described in this paper has been supported by the Australian Research Council.

References

[1] C. Huntsman and D. Cawthron, "The MC68881 Floating-point Coprocessor", *IEEE MICRO*, Dec. 1983, pp. 44-54

[2] M. Uya, K. Kaneko and J. Yasui, "A CMOS Floating Point Multiplier", *IEEE J. Solid-State Circuits*, V. SC-19, No. 5, 1984, pp. 697-702

[3] K. Takeda *et al.* "A Single-Chip 80-bit Floating Point Processor", *IEEE J. Solid-State Circuits*, vol. SC-20, No. 5, 1985, pp. 986-992

[4] J. Fandianto and B. Y. Woo, "VLSI Floating-point Processors", *Proceedings of Seventh Symposium on Computer Arithmetic*, 1985, pp. 93-100

[5] T. Asprey *et al.* "Performance Features of the PA7100 Microprocessor", *IEEE Micro*, 1993, pp. 22-35

[6] I. Deyhimy, "Gallium Arsenide Joins the Giants", *IEEE Spectrum*, Feb., 1995, pp. 33-40

[7] S. Waser and M. Flynn, "Introduction to Arithmetic for Digital Systems Designers", New York: Holt, Rinehart and Winston, 1982

[8] M. R. Santoro and M. A. Horowitz, "SPIE: A Pipelined 64 × 64-bit Iterative Multiplier", *IEEE Journal of Solid-State Circuits*, Vol. 24, 1989, pp. 3594-3597

[9] R. K. Montoye, E. Hokenek and S. L. Runyon, "Design of the IBM RISC System/6000 Floating-Point Execution Unit", *IBM Journal of Research and Development*, Vol. 34, 1990, pp. 59-70

[10] Y. Oowaki, *et al.* "A Sub-10-ns 16×16 Multiplier Using 0.6-mm CMOS Technology", *IEEE J. of Solid-State Circuits*, vol.sc-22, No. 5, 1987, pp. 762-767

[11] Long, Stephen I. & Butner, Steven E. "Gallium Arsenide Digital Integrated Circuit Design". McGraw-Hill, Inc. 1990

[12] R. Sarmiento, P. P. Carballo and A. Núñez, "High Speed Primitives of Hardware Accelerators for DSP in GaAs Technology", *IEE Proceedings-G*, Vol. 139, 1992, pp. 205-216

[13] Knowles, S. "Arithmetic processor design for the T9000 Transputer", *ASPAAI-2, SPIE*, 1991

[14] M. R. Santoro, G. Bewick and M. A. Horowitz,"Rounding Algorithms for IEEE Multipliers", *Proceedings of 9th Symposium on Computer Arithmetic*, 1989, pp. 176-183

[15] K. Eshraghian, R. Sarmiento et al, "Speed-Area-Power Optimization for DCFL and SDCFL Class of Logic Using Ring Notation", *Microprocessing and Microprogramming 32*, 1991, NORTH-HOLLAND pp. 75-82

[16] Larry R. Tate et al. "32 Bit GaAs HFET IEEE Floating Point Multiplier", *GaAs IC Symposium 1992*, pp. 85-88

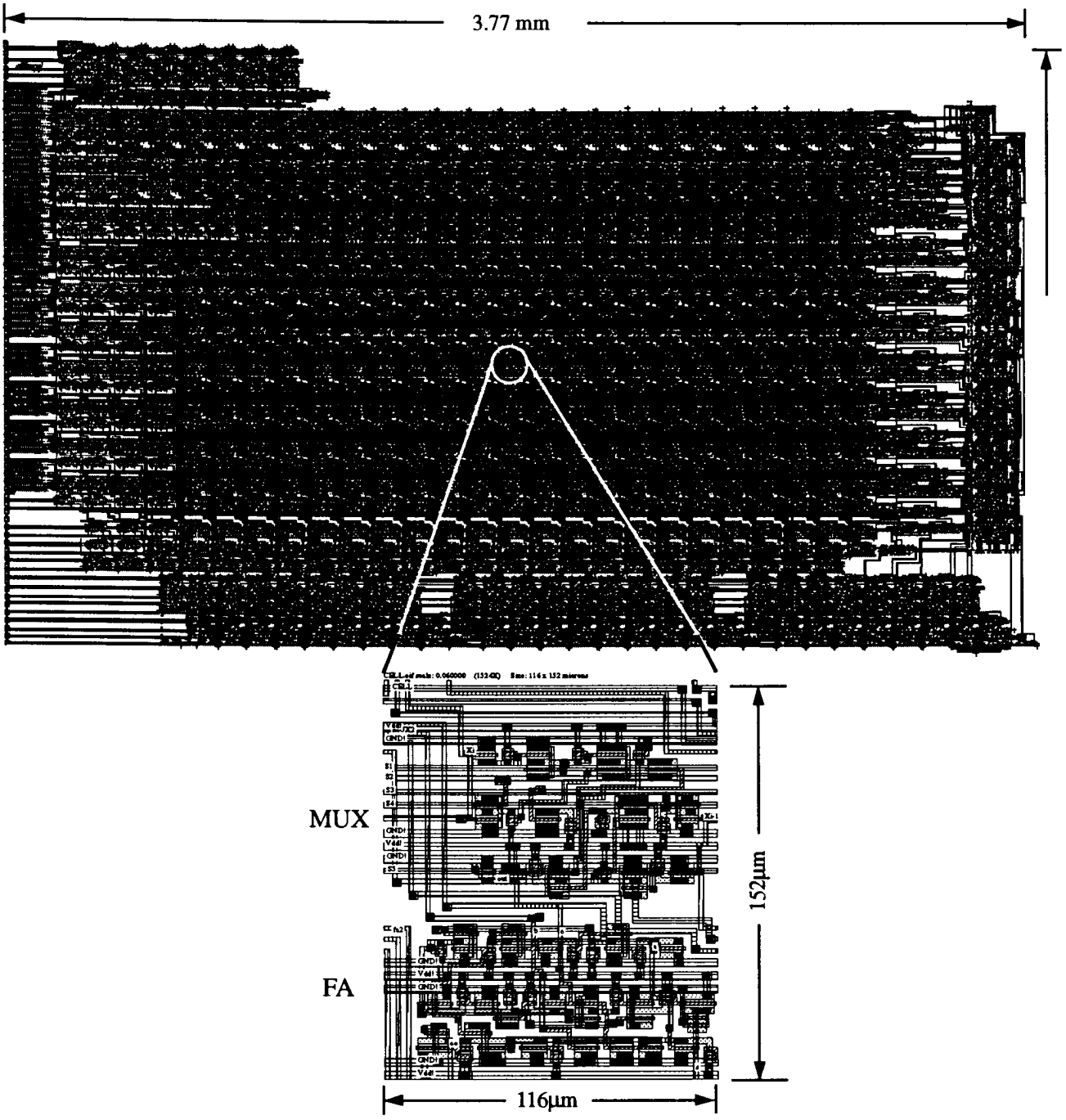

Figure 15. The layout of the 32-bit multiplier.