VLSIシステム向け加算回路および共通回路の高性能化に関する研究

牧野 博之

VLSIシステム向け加算回路および乗算回路の高性能化に関する研究

1997年

牧野博之

<内 容 梗 概>

本論文は、著者が1983年三菱電機株式会社に入社以来、LSI研究所ならびにシステムLSI開発研究所において行ってきたLSI回路設計技術に関する研究のうち、特に多くのVLSIを構成する上で必要不可欠である加算器、乗算器、さらには浮動小数点加算器および浮動小数点乗算器の高性能化に関する研究成果をまとめたもので、本文は以下の6章および付録より構成される。

第1章　序論

本研究を始めるに当たっての歴史的並びに技術的背景について概観し、問題点を抽出するとともに本研究の目的および意義を明らかにする。

第2章　加算器の高速化と小面積化に関する研究

本章では、高速加算器を実現するために行ったアルゴリズムおよびアーキテクチャ技術の最適化に関する研究内容について述べる。まず、種々の加算器アーキテクチャのスピードと占有面積を解析的に求めて比較する手法について述べ、これによって加算器の最適アーキテクチャを決定する。またトランジスタ数が少なく面積低減に有効な新規キャリーセレクト回路を提案する。次にこれらの技術を用いた64ビット整数加算器の試作・評価結果について述べる。

第3章　乗算器の高速化と小面積化に関する研究

本章では高速乗算器に関する研究内容について述べる。まず、トランスミッションゲートを有効に利用した回路のクリティカルパスの高速化手法に関して説明する。次に乗算器の高速化に関して、冗長二進数を用いた高速のアルゴリズムおよびアーキテクチャを提案し、これを有効に実現するCMOS回路の設計について述べる。また、これらの技術を適用した54 x 54ビット整数乗算器の試作・評価を行ったのでその結果について述べる。

第4章　浮動小数点加算器の高性能化に関する研究

本章においては浮動小数加算器の高速化に関する研究内容について述べる。まず浮動小数点加算の際に問題となる桁落ちの処理に関して、加算結果の桁落ちを入力データから容易に予測できる桁落ち予測の手法を提案し、その論理および回路構成を説明する。さらに、この桁落ち予測を64ビット浮動小数点加算器に適用し、試作および評価を行ったのでその結果について述べる。最後に、この64ビット浮動小数点加算器に第2章の結果を適用した場合についての考察を行う。

第５章　浮動小数点乗算器の高性能化に関する研究

本章においては、高速浮動小数乗算器に関する研究内容について述べる。まず、３次元CGの処理に適した機能として「CG乗算機能」を提案し、これを少量のハードウエアでインプリメントする方法について説明する。さらにこのCG乗算機能と第３章の結果とを64ビット浮動小数点乗算器に適用し、試作および評価を行ったのでその結果について述べる。

第６章　結論

本章においては本研究から得られた主要な結論について総括し、さらに第２章～第５章の各章で得られた研究成果を実際のVLSIマイクロプロセッサに適用した場合の性能予測について述べる。さらに、本研究における問題点について述べ、本研究以後の加算器および乗算器の将来展望について述べる。また、マイクロプロセッサを高速に動作させる上でのもう一つの重大なボトルネックとして、メモリのアクセスタイムが重要であることを述べ、筆者らが行ったキャッシュメモリの高速化に関する研究内容を付録として付記する。

付録

付録においては、基板材料として高速性に優れたガリウム砒素（GaAs）を用いた高速メモリに関する研究内容について述べる。まず、GaAsメモリにおける問題点を明らかにし、これらの問題点に対する解決策を示す。さらに、これらの解決策を16Kb GaAsメモリに適用し、その結果動作スピードと信頼性の両面において実用可能なレベルの値が得られたことを示す。

＜目　次＞

第1章　序論

1．1　関連分野の歴史的背景と問題点の概要

近年、コンピュータにおける高性能化は実に目覚ましい発展を遂げている。1951年に最初に商用となったUNIVAC1においては1秒間の加算回数が1,900であったのに対し、1991年に発売されたワークステーションHP9000/model750では50,000,000となっており、40年間で実に約26,000倍の高性能化が実現されている[1]。これによって、複雑な科学技術計算が可能となり、原子力、気象、エレクトロニクスあるいは機械などをはじめとする様々な分野においてシミュレーションによる現象予測やCAD（Computer Aded Design）による技術開発あるいは製品開発に利用されている。また、コンピュータの高速化はコンピュータ・グラフィクス（CG）の分野においても極めて大きな力を発揮しており、以前は2次元の処理までしかできなかったのに対して、近年では3次元の処理が可能となり、いわゆる仮想現実（バーチャルリアリティ）を一部実現するに至っている。これによって、様々な現象の可視化が可能となり、医療をはじめとする各分野で利用されるとともに、ゲームを中心とするエンターテインメント分野を大きく発展させている。特にエンターテインメント分野を通じてパーソナルコンピュータが急速に家庭に浸透し始めており、これに伴ってインターネットを始めとするデジタル情報通信分野も急速な発展を遂げている。このように、コンピュータの高速化は、社会および経済の発展に大きな影響を及ぼしており、今後も人類に様々な恩恵を与え続けることが期待される。

このようなコンピュータの急速な進歩を支えてきた技術として、コンパイラやOSを始めとするソフトウエア技術、あるいは半導体や基板実装などを中心とするハードウエア技術が挙げられるが、これらの中でも特に半導体によるVLSI（Very Large Scale Integration）技術の進歩によるところが極めて大きい。図1－1に代表例としてインテル社のマイクロプロセッサにおけるゲート長と動作周波数の推移を示す[2]。ゲート長とは、半導体素子におけるトランジスタのチャネルの長さを表す量で、微細化の指標とされる数値である。筆者の研究開始当時の1983年においては動作周波数が10MHz程度であったものが、現在では100MHzを超えており、およそ10年で1桁という急激な性能向上が実現されていることが分かる。インテル社のマイクロプロセッサ以外の大半のLSI（Large Scale Integrated Circuits）もほぼこのトレンドに従っている。また、この間にゲート長が1.0μmから0.3μmまで短縮されており、微細化を中心とする半導体技術が、コンピュータの進歩を大きく支えてきたということができる。また、これと同時にLSIにおいては多ビット化や多機能化も進められてきており、インテル社において最初に実現されたマイクロプロセッサ4004ではビット幅が4ビットであったのに対し、PentiumやPentium Proにおいては、64ビットのデータを処理することが可能となっている。このような多ビット化や高機能化を実現するためには、半導体の微細化に加えて回路設計技術においても工夫が必要とされる。しかしながら、上で述べたように筆者の研究開始当時においては、動作周波数が10MHz程度で

ビット幅も 8 ビットが主流であり、高性能化および高機能化のための回路技術に関する研究は未だ途上段階であった。したがって、高速化を実現するためにはVLSIの性能を律速する要因を明らかにしてそれを解決していくことが重要であった。すなわち、マイクロプロセッサを構成する各要素回路のなかでボトルネックとなる部分を明らかにして、これを回路技術の工夫によって効果的に克服していくことが必要であった。

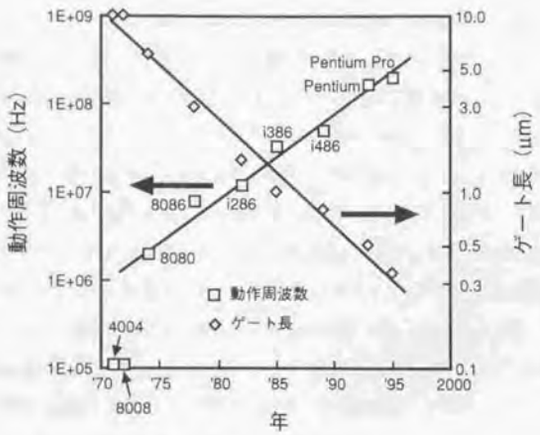


図１－１　インテル社のマイクロプロセッサにおける動作周波数とゲート長の推移

図１－２にコンピュータの構成を表す概念図を示す[1]。コンピュータは基本的にコントロール（Control）、データパス（Datapath）、メモリ（Memory）、入力装置（Input）および出力装置（Output）の５つの部分からなる。いわゆるプロセッサ（Processor）と呼ばれるのはコントロールとデータパスとを含めた部分のことで、現在シリコンによる１チップのVLSIで実現されており、一般に「マイクロプロセッサ」と呼ばれている。実行すべき処理を記述したプログラムはメモリの中にあり、プロセッサはメモリから命令とデータとを受け取り処理を実行する。プロセッサ内のコントロールは命令を解析してその結果に基づいて他の４つの部分を制御するコントロール信号を発生する。データパスは、このコントロール信号に従って例えば足し算や論理演算などの処理を実行する。入力装置および出力装

置もまた、コントロール信号に従って外部とのデータのやりとりを行う。近年では、これらのユニットを並列に並べ処理性能を高めたいわゆる並列コンピュータ等も作られているが[1],[3]、どのようなコンピュータも基本的に図１－２のような構造を有する。

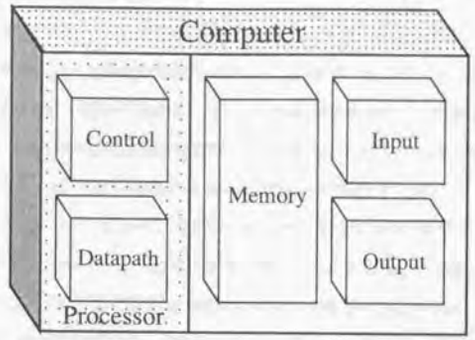


図１－２　計算機システムの基本構成を示す概念図

このような構成を持つコンピュータを高性能化するためには、それぞれのユニットの処理能力および相互の信号伝達能力を高める必要があるが、先にも述べたように全体の中でボトルネックとなっている箇所の性能を高めることが有効である。すなわち全体の中の最も遅い経路となっている、いわゆる「クリティカルパス」を高速化することによって全体の性能を効果的に改善することができる。一般に、コンピュータの性能を大きく支配しているのはマイクロプロセッサのクロック周波数であり、これを高めることでコンピュータ全体の性能を向上させることができる。したがって、マイクロプロセッサ内のクリティカルパスを高速化することによってクロック周波数を向上させることが最も効果的である。

筆者の研究開始時点においてマイクロプロセッサの重大なボトルネックとなっていたもはデータパスであった。データパスでは様々な演算やビット処理が実行されるが、その中でも最も時間を要するのが加算と乗算である。加算と乗算は全ての算術演算の基本となるもので、あらゆるマイクロプロセッサに必要不可欠な機能であるが、特に近年演算すべきデータのビット幅が増加しているために、演算に要する時間が増大している。このため、長ビットの加算や乗算を高速に処理するために、加算や乗算を実行するハードウエア（それぞれ加算器および乗算器と呼ぶ）には様々な工夫が行われている

[4]-[30]。加算器には、もともといわゆる筆算のように最下位桁から順番に計算を行うリップルキャリー方式[1],[4]が用いられていたが、この手法は計算時間がビット長に比例するために、ビット長が増大すると使うことができず、代わりにキャリールックアヘッド方式[5]-[8]が用いられるようになった。キャリールックアヘッド方式は、計算時間がビット長の対数に比例するので高速処理には有効な手法である。またこれらの他に有効な手段として、キャリーセレクト方式[9]-[11]、キャリースキップ方式[12]-[14]等が提案されている。また、乗算についてはビット数が小さいときは加算とシフトで実行されていたが、8ビットあるいは16ビットの時代から専用の乗算器が搭載されるようになっている[15]-[20]。乗算器の実行時間は基本的には加算器と同様にビット長の対数に比例させることができるが[21]、[22]、計算が複雑で比例係数が大くなるために加算器に比べて数倍の実行時間がかかり、最も重大なボトルネックとなっている。これを改善するためにブースのアルゴリズム[23]、ウレストリー法[24]および冗長二進数の使用[47]など様々の手法が提案され、実用化されている[25]-[30]。このように加算器と乗算器には高速化のための様々な手法があるが、それらはいずれもハードウエア量とのトレードオフとなっている。すなわち、処理の並列度を増すことである程度の高速化を実現することは可能であるが、そのために素子数が増加して面積が大きくなると、チップのコストが増大するばかりでなく、消費電力も増大し、配線等による寄生効果のためにかえって高速性が損なわれる場合もある。したがって、高速化を図る上ではハードウエア量を増加させないように考慮する必要がある。しかし、筆者の研究開始当初においては、64ビットの処理が可能な高速かつハードウエアの小さい加算器および乗算器はまだ作られていなかった。したがって、高速かつマイクロプロセッサに搭載可能なのデータパスを実現するためには高速でかつハードウエアの小さい加算器および乗算器を開発する必要があった。

また、コンピュータの重要な用途である3次元CGや各種科学技術計算などの分野では高精度の演算処理を実行するために大量の浮動小数点演算が必要とされる。浮動小数点演算においても加算と乗算が基本となるが、通常の整数用の加算器と乗算器だけでこれを処理しようとすると浮動小数点処理や丸め処理あるいは正規化処理[1]等のために一つの浮動小数点処理の実行に膨大な計算ステップが要求され、この結果処理時間が増大してしまう。このために最近のマイクロプロセッサでは浮動小数点演算器をハードウエアとして内蔵する傾向がある[31]。すなわち、半導体の微細化技術の進歩によって搭載可能なハードウエア量が増加したために通常の整数用データパス以外に浮動小数点処理のためのデータパスもVLSIマイクロプロセッサ内に取り込まれ始めている。筆者の研究開始当初においては、このようなハードウエアの搭載に向けて、高速かつ素子数の少ない浮動小数点加算器および浮動小数点乗算器の開発も重要な課題であった。

以上のように、CG等の用途に適した高速のVLSIマイクロプロセッサを実現するためには、加算器と乗算器の高速化と素子数の削減および浮動小数点処理機能の有効な付加が必要であった。

1．2　本研究の目的と論文の構成

本研究は、以上の問題点を解決するために行われたもので、主として3次元CG向けのVLSIマイクロプロセッサに関し、回路技術面からのアプローチによって高速化を図ることを目的とした。さらに具体的には、マイクロプロセッサの速度を律速するデータパスの主要構成要素である加算器および乗算器を高速化することにより、高速の整数データパスを実現するとともに、浮動小数点演算処理やCG向けの演算処理を高速化することにより、高速の浮動小数点データパスを実現することを目的とした。目標性能としては、研究開始当初の動作周波数である10MHzの10倍の100MHzを最低線として、100MHzを上回る性能の実現を目指した。図1－3に本論文の構成を示す。

第2章においては、高速加算器を実現するために行ったアルゴリズムおよびアーキテクチャ技術の改善に関する研究内容について述べる[32]-[34]。まず、種々の加算器アーキテクチャのスピードと占有面積を解析的に求めて比較する手法について述べ、これによって加算器の最適アーキテクチャを決定する。またトランジスタ数が少なく面積低減に有効な新規キャリーセレクト回路を提案する。さらに、これらの技術を用いた64ビット加算器の試作および評価結果について述べる。

第3章においては、高速乗算器に関する研究内容について述べる[35]-[37]。まず、トランスミッションゲートを有効に利用した回路のクリティカルパスの高速化手法に関して説明する。次に乗算器の高速化に関し、冗長二進数を用いた高速の乗算アルゴリズムおよびアーキテクチャを提案し、これを有効に実現するCMOS回路の設計について述べる。さらに、これらの技術を適用した54×54ビット乗算器の試作および評価結果について述べる。

第4章においては、浮動小数加算器の高速化に関する研究内容について述べる[38]-[40]。まず浮動小数点加算の際に問題となる桁落ちの処理に関して、加算結果の桁落ちを入力データから容易に予測できる桁落ち予測（LZA）の手法を提案し、その論理並びに回路構成を説明する。さらに、この桁落ち予測を64ビット浮動小数点加算器に適用し、試作および評価を行ったのでその結果について述べる。最後に、この64ビット浮動小数点加算器に第2章の結果を適用した場合についての考察を行う。

第5章においては、高速浮動小数乗算器に関する研究内容について述べる[41]-[43]。まず、3次元CGの処理に適した機能として「CG乗算機能」を提案し、これを少量のハードウエアでインプリメントする方法について説明する。さらにこのCG乗算機能と第3章の結果とを64ビット浮動小数点乗算器に適用し、試作および評価を行ったのでその結果について述べる。

第6章においては本研究から得られた主要な結論について総括し、さらに第2章～第5章の各章で得られた研究成果を実際のVLSIマイクロプロセッサに適用した場合の性能予測について述べる。さらに、本研究における問題点に関して述べ、本研究以後の加算器および乗算器の将来展望について述べる。また、マイクロプロセッサを高速に動作させる上でのもう一つの重大なボトルネックとして、メ

モリのアクセスタイムが重要であることを述べ、筆者らが行ったキャッシュメモリの高速化に関する研究内容を付録として付記する。

付録においては、基板材料として高速性に優れたガリウム砒素（GaAs）を用いた高速メモリに関する研究内容について述べる[44]-[62]。まず、GaAsメモリにおける問題点を明らかにし、これらの問題点に対する解決策を示す。さらに、これらの解決策を16Kb GaAsメモリに適用し、その結果動作スピードと信頼性の両面において実用可能なレベルの値が得られたことを示す。

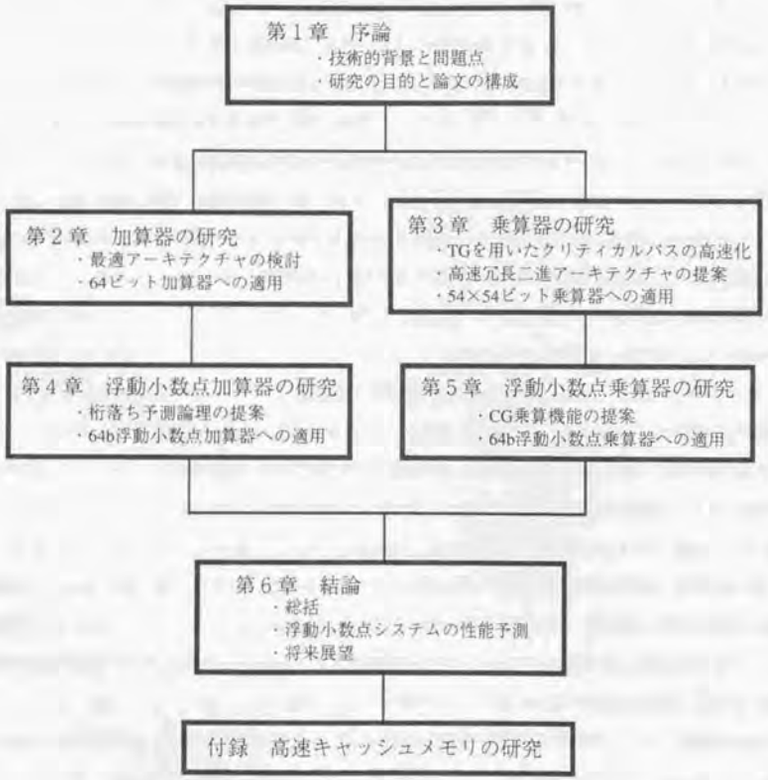


図１－３　本論文の構成

＜参考文献＞

[1] J. L. Hennessy and D. A. Patterson, "Computer Architecture : A Quantitative Approach," Morgan Kaufman Publishers, 1990.

[2] インターネットホームページ：http://www.intel.co.jp/jp/intel/museum/25anniv/html/index.htm

[3] K. Hwang and F. A. Briggs, "Computer Architecture and Paralell Processing," McGraw-Hill Book Company, 1985.

[4] N. H. E. Weste and K. Eshraghian, "Principles of CMOS VLSI Design," Addison-Wesley Publishing Company, 1993.

[5] A. Weinberger and J. L. Smith, "A Logic for High-Speed Addition," Nat. Bur. Stand. Circ., No.591 pp.3-12, 1958.

[6] R.P.Brent and H.T.Kung, "A regular layout for parallel adders," IEEE Tran. on Computers, vol. c-31, No.3, pp. 260-264, Mar. 1982.

[7] M. A. Bayoumi, G. A. Jullien and W. C. Miller, "An Area-time efficient NMOS Adder," Integration, the VLSI Journal, No.1, pp.317-334, 1983.

[8] T. F. Ngai, M. J. Irwin and S. Rawat, "Regular, Area-Time Efficient Carry-Lookahead Adders," IEEE Journal of Paralell and Distributed Computing, No.3, pp.92-105, 1986.

[9] A.Tyagi, "A Reduced Area Scheme for Carry-select Adders(Preliminary Version)," IEEE ICCD Proc., pp. 255-258, 1990.

[10] A.Tyagi, "A reduced Area Scheme for Carry-select Adders," IEEE Tran. on Computers, vol.42, No10, pp. 1163-1170, October 1993.

[11] K.Suzuki, M.Yamashita, J.Goto, T.Inoue, Y.Koseki et al, "A 2.4-ns,16-b,0.5-μm CMOS Arithmetic Logic Unit for Microprogrammable Video Signal Processor LSIs", CICC Proc., pp. 12.4.1-12.4.4, 1993.

[12] V.G.Oklobdzija and E.R.Barnes, "Some Optimal Schemes for ALU Implementation in VLSI Technology," Proc. of 7th Symposium on Computer Arithmetic, 1985, pp. 2-8.

[13] A. Guyot, B. Hochet and J. M. Muller, "A Way to Build Efficient Carry-Skip Adders," IEEE Tran. on Computers, vol. 36, No.10, pp. 1144-1152, Oct. 1987.

[14] P. K. Chan and M. D. F. Schlag, "Analysis and Design of CMOS Manchester Adders with Variable Carry-Skip," IEEE Tran. on Computers, vol. 39, No.8, pp. 983-992, Aug. 1990.

[15] L. Y. Lee, H. L. Garvin and C. W. Slayman, "A 8 x 8b Paralell Multiplier in Submicron Technology," Dig. Tech. Papers of 1985 ISSCC, pp.84-85, Feb. 1985.

[16] M. Hatamian and G. L. Cash, "A 70-Mhz 8-bit x 8-bit Paralell Pipelined Multiplier in 2.5-μm CMOS," IEEE J. Solid-state Circuits, Vol. SC-21, No.4, pp. 505-513, Aug. 1986.

[17] Y. Oowaki, K. Numata, K. Tsuchiya, K. Tsuda, A. Nitayama and S. Watanabe, "A 7.4ns CMOS 16 x 16 Multiplier," Dig. Tech. Papers of 1987 ISSCC, pp.52-53, Feb. 1987.

[18] R. Sharma, A. D. Lopez, J. A. Michejda, S. J. Hillenius, J. M. Andrews and A. J. Studwell, "A 6.75-ns 16 x 16-bit Multiplier in Single-Level-Metal CMOS Technology," IEEE J. Solid-State Circuits, vol. 24, No.4, pp.922-927, Aug. 1989.

[19] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi and A. Shimizu, "A 3.8-ns CMOS 16 x 16-b Multiplier Using Complementary Pass-Transistor Logic," IEEE J. Solid-state Circuits, Vol. 25, No.2, pp. 388-395, Apr. 1990.

[20] S. Roberts, W. Snyder, H. Chin, H. Hingarh, S. Leibiger, R. Lahri, L. Bouknight and M. Biswal, "A 2.5 ns ECL 16 x 16 Multiplier," CICC Proc., pp. 24.7.1-24.7.4, 1990.

[21] 高木、安浦、矢島、「冗長2進加算木を用いたVLSI向き高速乗算器」、電子通信学会論文誌、vol. J66-D, No.6, pp.683-690, June 1983.

[22] 高木、矢島、「算術演算のハードウェアアルゴリズム」、情報処理、vol. 26, No.6, pp.632-639, June 1985.

[23] A. D. Booth, "A Signed Binary Multiplication Technique," Q. J. Mech., Appl. Math. 4 pp.236-240, 1951.

[24] C. S. Wallace, "A Suggestion for Fast Multiplier," IEEE Trans. Electron. Comput., vol. EC-13, pp. 14-17, Feb. 1964.

[25] S. Hiker, N. Phan and D. Rainey, "A 3.4ns 0.8μm BiCMOS 53 x 53b Multiplier Tree," Dig. Tech. Papers of 1994 ISSCC, pp.292-293, Feb. 1994.

[26] M. Nagamatsu, S. Tanaka, J. Mori, T. Noguchi and K. Hatanaka, "A 15 ns 32 x 32-Bit CMOS Multiplier with an Improved Paralell Structure," CICC Proc., pp. 10.3.1-10.3.4, 1989.

[27] J. Mori, M. Nagamatsu, M. Hirano, S. Tanaka, M. Noda, Y. Toyoshima, K. Hashimoto, H. Hayashida and K. Maeguchi, "A 10-ns 54 x 54-b Parallel Structured Full Array Multiplier with 0.5-μm CMOS Technology," IEEE J. Solid-State Circuits, vol. 26, No.4, pp.600-605, Apr. 1991.

[28] G. Goto, T. Sato, M. Nakajima and T. Sukemura, "A 54 x 54-b Regularly Structured Tree Multiplier," IEEE J. Solid-State Circuits, vol. 27, No.9, pp. 1229-1235, Sep. 1992.

[29] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki and Y. Nakagome, "A 4.4 ns CMOS 54 x 54-b Multiplier Using Pass-Transistor Multiplexer," IEEE J. Solid-state Circuits, Vol.30, No.3, pp. 251-257, March 1995.

[30] S. Kuninobu, T. Nishiyama and T. Taniguchi, "High Speed MOS Multiplier and Divider Using Redundant Binary Representation and Their Implimentation in a Microprocessor," IEICE Trans. Electron., vol. E76-C, No.3, pp. 436-445, Mar. 1993.

[31] W. J. Bowhill, R. L. Allmon, S. L. Bell, E. M. Cooper, D. R. Donchin, J. H. Edomondson, T. C. Fischer, P. E. Gronowski, A. K. Jain, P. L. Kroesen, B. J. Loughlin, R. P. Preston, P. I. Rubinfeld, M. J. Smith, S. C. Thierauf and G. M. Wolrich, "A 300Mhz 64b Quad-Issue CMOS RISC Microprocessor," 1995 ISSCC Dig. Tech. Papers, pp.182-183, February 1995.

[32] H. Morinaka, H. Makino, Y. Nakase, H. Suzuki and K. Mashiko, "A 64bit Carry Look-ahead CMOS Adder using Modified Carry Selec," CICC Proc., pp. 585-588, May 1995.

[33] 森中、牧野、中瀬、鈴木、益子、角、「新しいキャリーセレクト方式(MCS)を使った64ビットキャリールックアヘッドCMOS加算器」、電子情報通信学会技術研究報告、ED95-97, pp.1-6, 1995.

[34] H. Morinaka, H. Makino, Y. Nakase, H. Suzuki, K. Mashiko and T. Sumi, "A 2.6-ns 64-b Fast and Small CMOS Adder," IEICE Transactions on Electronics, Vol.E79-C, No.4, pp.530-537, Apr. 1996.

[35] H. Makino, Y. Nakase and H. Shinohara, "A 8.8-ns 54 x 54-bit Multiplier Using New Redundant Binary Architecture," IEEE Proc. of 1993 ICCD., pp.202-205, Oct. 1993.

[36] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, H. Shinohara and K. Mashiko, T. Sumi and Y. Horiba, "A Design of High-Speed 4-2 Compressor for Fast Multiplier," IEICE Transactions on Electronics, Vol.E79-C, No.4, pp.538-548, Apr. 1996.

[37] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara and K. Mashiko, "An 8.8-ns 54×54-bit Multiplier with High Speed Redundant Binary Architecture,"IEEE J. Solid-State Circuits, Vol.31, No.6, pp.773-783, Jun. 1996.

[38] H. Suzuki, Y. Nakase, H. Makino, H. Morinaka and K. Mashiko, "Leading-zero Anticipatory Logic for High-speed Floating Point Addition," CICC Proc., pp. 589-592, May 1995.

[39] 鈴木、森中、牧野、中瀬、益子、角、「高速浮動小数点加算器のための桁落ちシフト量予測回路の提案」、電子情報通信学会技術研究報告、ED95-98, pp.7-12, 1995.

[40] H. Suzuki, H. Makino, H. Morinaka, Y. Nakase, K. Mashiko and T. Sumi," IEEE J. Solid-State Circuits, vol. 31, No.8, pp. 1157-1164, Aug. 1996.

[41] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase and K. Mashiko, "A 286MHz 64-bit Floating Point Multiplier with Enhanced CG Operation," Dig. of Symposium on VLSI Circuit, pp.15-16, June 1995.

[42] 牧野、鈴木、森中、中瀬、益子、角、「CGに適した機能を有する286MHz、64ビット浮動小数点乗算器」、電子情報通信学会技術研究報告、ED95-99, pp.13-20, 1995.

[43] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, K. Mashiko and T. Sumi, "A 286 MHz 64-b Floating Point Multiplier with Enhanced CG Operation," IEEE J. Solid-State Circuits, Vol.31, No.4, pp.504-513, Apr. 1996.

[44] 牧野、高野、谷野、茅野、「プリデコード方式を用いた低消費電力GaAs 4Kbスタティック RAMの設計」昭和60年度電子通信学会半導体・材料部門全国大会予稿集、pp.2-113.

[45] N. Tanino, S. Takano, M. Noda, H. Makino, K. Sumitani, H. Nakano, K. Nishitani and S. Kayano, "A 2.5ns/200mW GaAs 4Kb SRAM," Tech. Dig. of 1986 GaAs IC Symposium, pp.101-104, Oct. 1986.

[46] 牧野、高野、野田、谷野、西谷、茅野、「GaAs 4Kbスタティック RAM」、電子情報通信学会技術研究報告、ED86-135, pp.39-46, 1987.

[47] S. Takano, H. Makino, N. Tanino, M. Noda, K. Nishitani and S. Kayano, "A 16K GaAs SRAM," Dig. Tech. Papers of ISSCC '87, pp.140-141, Feb. 1987.

[48] S. Takano, H. Makino, N. Tanino, M. Noda, K. Nishitani and S. Kayano, "A GaAs 16K SRAM with a Single 1-V Supply," IEEE J. Solid-State Circuits, vol. SC-22, No.5, pp.699-703, Oct. 1987.

[49] H. Makino, S. Matsue, M. Noda, N. Tanino, S. Takano, K. Nishitani and S. Kayano, "A 7ns/850mW GaAs 4Kb SRAM Fully Operative at 75℃," Tech. Dig. of 1988 GaAs IC Symposium, pp.71-74, Nov. 1988.

[50] H. Makino, S. Matsue, M. Noda, N. Tanino, S. Takano, K. Nishitani and S. Kayano, "A 7ns/850mW GaAs 4-Kb SRAM with Little Dependence on Temperature," IEEE J. Solid-State Circuits, vol. SC-25, No.5, pp.1232-1238, Oct. 1990.

[51] 牧野、松江、野田、谷野、高野、西谷、茅野、「75℃で動作する7ns/850mWのGaAs 4Kビット RAM」、電子情報通信学会技術研究報告、ED88-144, pp.59-66, 1989.

[52] M. Noda, K. Hosogi, K. Sumitani, H. Nakano, H. Makino, K. Nishitani and M. Otsubo, "A high-yield 4Kb SRAM process technology using self-aligned gate MESFETs with a partially depleted p-layer," Tech. Dig. of 1988 GaAs IC Symposium, pp.227-230, Nov. 1988.

[53] 野田、細木、住谷、中野、牧野、西谷、大坪、「部分空乏化したp型埋め込み層を有するセルフアラインゲートMESFETを用いたGaAs 4Kb SRAMプロセス技術」、ED88-144, pp.59-66, 1989.

[54] S. Matsue, H. Makino, M. Noda, N. Tanino, S. Takano, K. Nishitani and S. Kayano, "A Soft Error Improved 7ns/2.1W GaAs 16Kb SRAM," Tech. Dig. of 1989 GaAs IC Symposium, pp.41-44, Oct. 1989.

[55] 松江、牧野、野田、谷野、高野、西谷、茅野、「ソフトエラー耐性を改善した7ns/2.1W GaAs 16Kb SRAM」、電子情報通信学会技術研究報告、ED89-137, pp.1-8, 1990.

[56] S. Matsue, H. Makino, M. Noda, H. Nakano, S. Takano, K. Nishitani and S. Kayano, "A 5-ns GaAs 16-kb SRAM," IEEE J. Solid-State Circuits, vol. SC-26, No.10, pp.1399-1406, Oct. 1991.

[57] M. Noda, S. Matsue, M. Sakai, K. Sumitani, H. Nakano, T. Oku, H. Makino, K. Nishitani and M. Otsubo,

"A Triple-Level Interconnection Technology for High Speed 16Kb GaAs SRAM," Extended Abstracts of 1990 Conference on Solid State Devices and Materials (SSDM), pp.71-74, Aug. 1990.

[58] 中野、野田、酒井、松江、奥、住谷、牧野、高野、西谷、「3層配線を用いた4.4ns/2W GaAs 16Kb SRAM」、電子情報通信学会技術研究報告、ED90-150, pp.41-46, 1991.

[59] H. Nakano, M. Noda, M. Sakai, S. Matsue, T. Oku, K. Sumitani, H. Makino, H. Takano and K. Nishitani, " A High-Speed GaAs 16Kb SRAM of 4.4ns/2W Using Triple-Level Metal Interconnection," Tech. Dig. of 1990 GaAs IC Symposium, pp.151-154, Oct. 1990.

[61] M. Noda, S. Matsue, M. Sasaki, K. Sumitani, H. Nakano, T. Oku, H. Makino, K. Nishitani and M. Otsubo, "A High-Speed 16-kb GaAs SRAM of Less than 5 ns Using Triple-Level Metal Interconnection," IEEE Trans. on Electron Dev., vol. ED-39, No.3, pp.494-499, Mar. 1992.

[62] 野田、細木、前村、加藤、中島、牧野、西谷、大坪、「GaAs LSI用サブミクロンゲートp層埋込み型FETのLDD化による均一性、高速性の改善」、電子情報通信学会技術研究報告、ED89-130, pp.7-13, 1990.

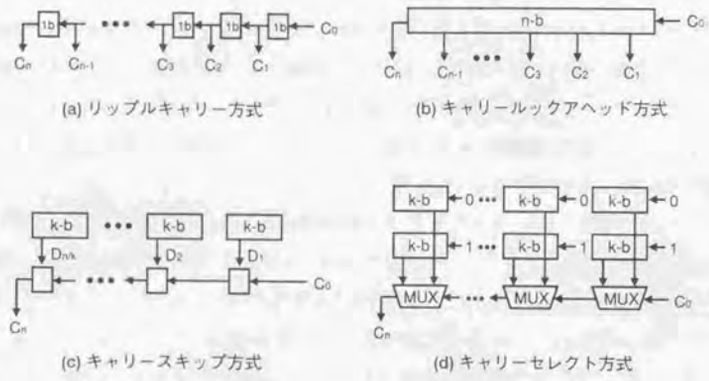
第２章　加算器の高速化と小面積化に関する研究

２．１　緒言

第１章において述べたように、加算器はデータパスの中で最も重要なキーパーツであり、その高速化は LSI 全体を高性能化する上で必要不可欠である。また、携帯機器への応用が進むにしたがって加算器には高速性だけでなく、素子数を極力削減することによる小面積化とこれに伴う低消費電力化が強く要求されるようになっている。このため、加算器の高速化と小面積化に関しては、これまでアルゴリズムやアーキテクチャ等のいわゆる上流設計レベルから、回路やレイアウトといった下流設計レベルまで様々な視点からの研究が行われてきた[1]-[20]。加算器においては、いわゆる筆算から類推されるように、各桁で発生する桁上げ信号（キャリー）を決定することが重要であり、高速加算器を実現する上ではキャリー信号を効率よく高速に伝達する必要がある。このため、これまで行われてきた研究は主にキャリー信号の伝達方式に関するものとなっており、加算器のアーキテクチャはキャリーの伝達方式の違いによって特徴付けられている。

図２－１に代表的な４種類の加算器アーキテクチャの概念図を示す。それぞれｎビットのキャリー出力を求めるための構成を示している。加算器アーキテクチャの中で最も単純なものは、(a)に示すリップルキャリー方式（RCA）であり、これは下位桁から順次キャリーを発生させて計算を行う方式である。その研究の歴史は古く 1955 年に既に真空管を用いた電子回路の試作結果が報告されており[1]、その後現在に至るまでシリコン VLSI の中に広く用いられている。リップルキャリー方式は単純で素子数が少ないという長所を持つ反面、計算時間が入力データのビット数に比例して増大するため、多ビットの計算に長時間を要するという欠点がある。この欠点を改善するために、(b)に示すキャリールックアヘッド方式（CLA）[2], [4], [5], [7]、(c)に示すキャリースキップ方式（CSK）[6], [8], [10], [15]、および(d)に示すキャリーセレクト方式（CSA）[3], [11], [12], [15], [16]等の手法が開発されている。キャリールックアヘッド方式は、多ビットからなるブロックのキャリー出力を論理回路を用いて一度に計算する方式で、一般に計算時間を入力データのビット数の対数に比例させることができるため、リップルキャリー方式に比べて高速の計算が可能となる。その反面論理が複雑で、素子数が多く回路の規則性が乏しいためにレイアウトが困難であるという欠点を持つ。キャリースキップ方式は、全体を例えばｋビットのブロックに分割し、それぞれのブロックから中間信号（D_i）発生させ、その中間信号と下位桁からのキャリー入力から順次キャリー信号をブロック毎に伝搬させる手法である。この例ではブロックの分割を全てｋビットとしているが、必要に応じて各ブロックのビット数を変化させても良い。この方式は、リップルキャリー方式よりも高速で、キャリールックアヘッド方式よりも規則性が高くレイアウトし易いという長所があり、両者の中間的な方式であると言うことができる。キャリ

ーセレクト方式は、キャリースキップ方式と同様に全体を例えばkビットのブロックに分割し、それぞれのブロックにおいてキャリー入力が「0」の場合と「1」の場合の2系統のキャリー伝搬回路を設け、それぞれのキャリー出力を、下位桁からのキャリー入力によって選択する手法である。この方法は、高速性と回路の規則性の両方に優れているが、キャリー伝搬回路を2系統設ける必要があるため、素子数が多く大面積を要し、これに伴って消費電力が大きくなるという欠点がある。このため、キャリーセレクト方式は、主として上位側の一部の桁のキャリー生成に用いられることが多い。



(a) リップルキャリー方式　(b) キャリールックアヘッド方式

(c) キャリースキップ方式　(d) キャリーセレクト方式

図2－1　代表的な加算アーキテクチャ

以上のように、それぞれの加算器アーキテクチャには長所と短所があり、これらを用途に応じて単独あるいは複数のアーキテクチャを組み合わせて用いる必要がある。したがって、高速かつ小面積の加算器を実現するためには、それぞれのアーキテクチャに対して最適化された回路のスピードと面積を正当に評価し、それぞれを比較することにより優れたものを選定する必要がある。また、その比較にはレイアウトに伴う配線容量やゲート容量等の遅延要因が正確に取り入れれなければならない。これまで、ある特定のアーキテクチャに対して高速化と小面積化の両面からの最適化を行った研究は多数行われている[4], [5], [7], [8], [11], [14], [17]。しかしながら、様々なアーキテクチャに対してゲート遅延時間や配線容量の効果などを正確に考慮した正当な比較というものは筆者らの研究以前には行われていなかった。従来の研究では、単純化のために例えばすべての2入力ゲートを同じ遅延時間としたり、あるいは配線遅延を配線の長さに拘わらず一定とするなど、いくつかの現実的でない仮定を基

にしていた。Tyagiらの研究[12], [15]ではこれらの遅延効果も取り入れられているが、これはあくまでもスタンダードセルを基本とした自動レイアウトでの議論であった。筆者らの目指す高性能プロセッサ用の高速加算器を実現するためには、よりフレキシブルで高速化が可能な手書きによるフルカスタムレイアウトを基にした議論が必要である。

そこで筆者らは実際のフルカスタムレイアウトを基に各種アーキテクチャの遅延時間と面積の最適化を行った[21]-[23]。この最適化にはゲート自体の遅延、配線遅延およびファンアウトによる遅延をそれぞれのゲートの面積とともに現実的な形で取り入れた。後に2．2．2節において述べるように、サブミクロン以降のプロセスでは配線遅延とゲート容量遅延の影響が大きくなり、この2つの遅延を正確に見積ることが高速で小面積な加算器を設計するためには不可欠である。

本章では、2．2節において、まず代表的な5つのアーキテクチャに対し、フルカスタムレイアウトに基づく速度と面積のパラメータを導入することによって正当な比較を行い、最適アーキテクチャを決定する。さらに、加算器の遅延時間の要因に関して分析を行い、各アーキテクチャの特徴と今後の展望に関して考察する。2．3節においては、2．2節で決定したアーキテクチャによる64ビット加算器の設計、試作および評価結果について述べる。回路面ではトランジスタ数を低減するために新しいキャリーセレクト方式であるModified Carry Select（MCS）を提案する。最後に2．4節において本章の結論をまとめる。

2．2　加算器アーキテクチャおよび回路の最適化に関する研究

2．2．1　各種方式の比較によるアーキテクチャの最適化

加算器のアーキテクチャを最適化するためには、加算器を構成する各種ゲートの遅延時間と面積を定量化する必要がある。図２－２に例としてインバータの遅延時間の要因を示す。遅延時間の要因は、大きくゲート自体のイントリンシックな遅延、配線による遅延およびファンアウトによる遅延の３つに分けられる。ゲート自体のイントリンシックな遅延は、ゲートのスイッチングに要する遅延とゲート自身の寄生容量を充放電するのに要する時間の和であり、配線による遅延は配線の寄生容量を充放電するのに必要な時間である。また、ファンアウトによる遅延は、次段に接続される回路のゲート容量を充放電するのに要する時間である。したがって、スピードの評価にはこれら３つの要因を考慮しなければならない。特にサブミクロンプロセスでは配線遅延とゲート容量遅延の影響がゲート自体のイントリンシックな遅延よりも大きくなっており、この２つの遅延を正確に取り入れた上で最適化を図る必要がある。

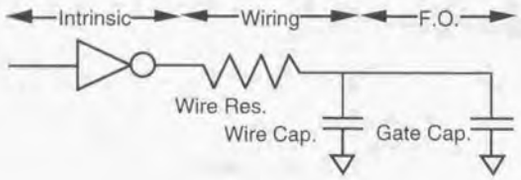
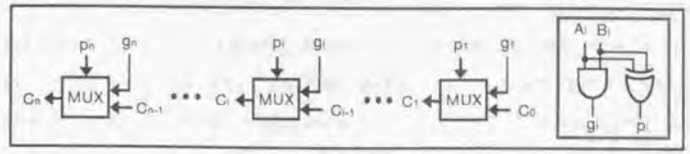
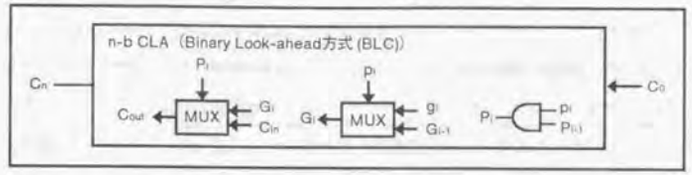


図２－２　ゲート遅延時間の要因

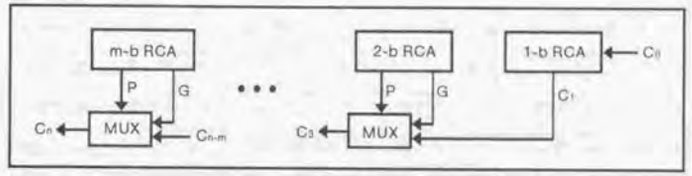
そこで、面積や遅延時間に対して有効な５種類のアーキテクチャに対して、これらの遅延要因をできるだけ正確に考慮した遅延時間と面積の見積もりを行った。図２－３にそれぞれのアーキテクチャを示す。図中(a)はリップルキャリー方式（RCA）で、各桁において入力 A_i と B_i の論理積からいわゆる生成信号 g_i（generate signal）、および排他的論理和から伝搬信号 p_i（propagate signal）を発生させ、これらの信号に基づいて下位桁より順次キャリー信号を決定する。キャリーの決定回路は最も単純で高速なマルチプレクサ回路（MUX）とした。(b)はキャリールックアヘッド方式（CLA）で、各桁の伝搬信号 p_i の論理積から多ビット分の伝搬信号 P_i を生成し、同時に生成信号 g_i と伝搬信号 p_i によって多ビット分の生成信号 G_i を生成する。G_i は g_i と下位桁からの G_{i-1} とを p_i によって選択することにより生成さ



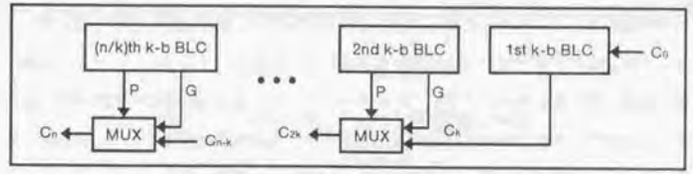
(a) リップルキャリー方式（RCA）



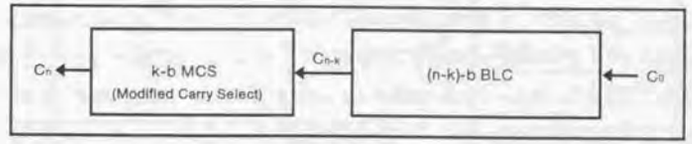
(b) キャリールックアヘッド方式（CLA）



(c) キャリースキップ方式（CSK）



(d) CLAとRCAの組合せ（k-b CLA x (n/k)-b RCA）



(e) CLAとCSAの組合せ（(n-k)-b CLA + k-b CSA）

図２－３　最適化を行った加算器アーキテクチャ

れる。このように生成された多ビット分の生成信号G_iと伝搬信号P_iに基づいて、G_iと下位からのキャリー入力C_nとをP_iで選択することによりキャリー出力C_{out}を生成する。ここでは、CLAを実現する最も高速の方式としてG_i、P_iおよびC_{out}を二進木状に生成するバイナリルックアヘッド方式（BLC）[16], [18]を採用した。(c)はキャリースキップ方式（CSK）で、全体をm個のブロックに分割してそれぞれのブロックからブロックの生成信号Gと伝搬信号Pを生成し、これらの信号に基づいてマルチプレクサ回路によってキャリー信号の伝搬を行う構成とした。各ブロックのビット数を下位側から1、2、…、mと一つづつ増加させ、ブロック内でリップル的に順次G信号とP信号を伝搬させることにより、クリティカルパスの回路段数を最小限としている。(d)はキャリールックアヘッド方式とリップルキャリー方式を組み合わせたもので、全体をkビットのブロックに分割し、各ブロックにおいてキャリールックアヘッド方式による生成信号Gと伝搬信号Pの生成を行い、さらにこれらの信号に基づいてマルチプレクサ回路を用いてリップルキャリー方式によるキャリー信号の伝搬を行う。ここでも各ブロックの信号生成には、バイナリルックアヘッド方式（BLC）を採用した。(e)はキャリールックアヘッド方式とキャリーセレクト方式を組み合わせたもので、下位側の(n−k)ビットはキャリールックアヘッド方式によるキャリー信号C_{n-k}の生成を行い、上位側のkビットはキャリーセレクト方式によって予めキャリー入力が「0」の場合と「1」の場合のキャリーおよび和（サム）を生成しておいて、下位からのキャリー信号C_{n-k}の値によっていずれかを選択する構成となっている。キャリーセレクト回路には、従来の回路を改良して素子数を削減したMCS（Modified Carry Select）回路を採用した。MCSに関しては2．3．2節において説明する。

以上の5種類のアーキテクチャに対して回路の最適化を行い、動作速度と面積の算出を行った。算出に当たっては、各アーキテクチャを構成する基本回路を実際にレイアウトし、それらの動作スピードと面積を求め、その値に基づいてそれぞれのアーキテクチャにおける全体の動作速度と面積を求めた。表2－1に各アーキテクチャを構成する基本ゲートの遅延時間と面積を示す。回路はインバータ（INV）、2入力NAND（2NAND）、2入力NOR（2NOR）、排他的論理和（XOR/XNOR）およびマルチプレクサ（MUX）の5種類で、XOR/XNORおよびMUXはトランスミッションゲート（TG）により構成される。基本インバータはゲート幅が8.4μmのpMOSトランジスタと4.2μmのnMOSトランジスタから構成され、他の基本ゲートはサイズを調整することにより、この基本インバータと負荷駆動能力を等しくしている。表2－1において各ゲートに対して示されている遅延時間は、この際のイントリンシックな遅延時間である。各ゲートの負荷駆動能力を等しくしているために、配線長に対する依存性とファンアウトに対する依存性は、いずれも等しくなっている。配線による遅延時間は配線1mm当たりの遅延時間として表しており、ゲート負荷による遅延時間はファンアウト1当たりの遅延時間として表している。表の値は全て基本サイズのインバータで規格化された値となっており、ゲート

負荷であるファンアウト数もこの基本インバータを単位としている。また、マルチプレクサ回路においては、トランスミッションゲートのゲート信号が切り替わる場合（select）と、ゲート信号が切り替わらずにソース・ドレインを信号が通過する場合（thru）とに分けて遅延時間を示している。遅延時間は0.5μmCMOSプロセスのパラメータによるSPICEシミュレーションの結果として得られた値で、面積は実際にマニュアルレイアウトを行うことにより得られた値である。配線は3層配線としている。

表2－1　加算器を構成する基本ゲートの遅延時間と面積

	Delay	Size
INV(8.4/4.2)	1.0	1.0
2NAND	1.5	1.7
2NOR	2.1	2.1
XOR/XNOR	1.9	2.3
MUX (thru)	0.7	1.9
MUX (select)	1.5	1.9
Wire delay / 1mm	3.9	——
Gate cap. delay / F.O.	0.6	——

表2－2　各アーキテクチャの遅延時間と面積

	Architecture	Delay	Size
(a)	RCA	2.0n + 4.8	8.6n - 2.3
(b)	CLA	7log10n + 0.1n + 6.7	6nlog10n + 6.3n
(c)	CSK	$\sqrt{8n+1}$ + 0.1n + 3.7	11.7n - 1.8$\sqrt{8n+1}$
(d)	k-b CLA × (n/k)-b RCA	$7\log_{10}k + 1.9n/k$ + 0.1n + 4.8	$6n\log_{10}k + 6.3n$ +2.3n/k - 2.3
(e)	(n-k)-b CLA + k-b CSA	$7\log_{10}(n-k)$ + 0.1n + 6.7	$6(n-k)\log_{10}n + 6k\log_{10}k$ + 6.3n + 4.4k

表２－１の値を用いて５種類の各アーキテクチャによるｎビット加算器を実現した場合の、それぞれの遅延時間と面積をｎの関数で表したものを表２－２に示す。それぞれ実際のレイアウトを想定して配線長およびファンアウト数を計算し、そこから遅延の値を算出している。５つのアーキテクチャのうち「k-b CLA×(n/k)-b RCA」と「(n-k)-b CLA＋k-b CSA」の２つは、ビット数ｎの他にさらにブロック分割のための変数ｋを持つ。例えば「k-b CLA×(n/k)-b RCA」はｋがｎのときｎビットCLAとなり、ｋが１のときｎビットRCAとなる。このため、この２つのアーキテクチャでは遅延時間と面積がｎとｋの２つの変数に対する関数として表されている。５種類のアーキテクチャのｎに対する依存性に注目すると、スピード面ではRCAがｎに比例、CSKがｎの平方根に比例、他の３つがｎの対数に比例する。また、面積ではRCAとCSKがｎに比例し、他の３つがｎの対数にｎを乗じたものに比例する。

表２－２をグラフにしたものを図２－４に示す。図ではｎが16ビット、32ビット、64ビットおよび128ビットの４通りの場合に対する遅延時間と面積の関係をプロットしている。

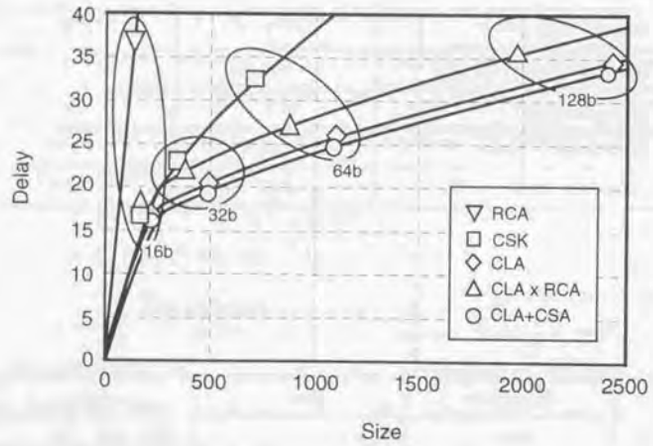


図２－４　各アーキテクチャにおける遅延時間と面積の関係

リップルキャリー方式（RCA）は最も面積が小さいが、遅延時間がｎに比例するため常に最も遅く、高速加算器には適さない。キャリースキップ方式（CSK）は32ビットまではRCA以外の他の３つのアーキテクチャCLA、CLA×RCAおよびCLA＋CSAと同程度の遅延時間であるが、64ビットでは、

この３者と比べそれぞれ27%、20%および29%遅くなる。キャリールックアヘッド方式（CLA）を用いた３つのアーキテクチャCLA、CLA×RCAおよびCLA＋CSAは64ビットでも高速であるが、面積においてCSKに劣る。64ビットでは、CSKに比べてCLA、CLA×RCA、CLA＋CSAはそれぞれ55%、23%、54%面積が大きくなる。速度面では、16ビット以上でCLA＋CSAが最も高速である。

以上の結果より、面積の面ではCSKが優れており、速度面ではCLA＋CSAが優れていることがわかる。したがって、加算器を設計する際の要求に応じて、面積を優先する場合はCSKを採用し、速度が優先される場合はCLA＋CSAを採用すればよく、またその中間のものが欲しい場合はCLA×RCAを用いればよい。すなわち、図２－４に示す結果により、加算器アーキテクチャの選定基準が明確にされたということができる。

本研究ではでは加算器の試作にあたって、面積よりも速度に重点を置き、最も高速なアーキテクチャであるCLA＋CSAを選択した。ただしこのアーキテクチャでは、ｋの値を最適化する必要があるためｋと遅延時間×面積の関係を調べた。図２－５にCLA＋CSAにおける遅延時間×面積のｋに対する依存性を示す。ｋが８ビット、すなわち「56-b CLA＋8-b CSA」のときに遅延時間×面積が最小となり、スピードと面積の点で最も有利となることがわかる。したがって、k=8を採用することとした。

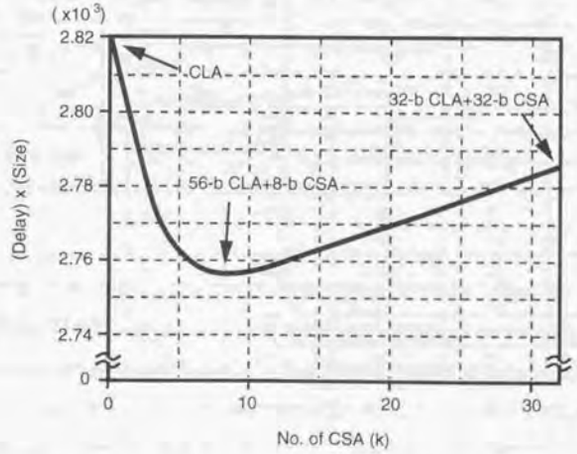


図２－５　CLA+CSA方式における遅延時間×面積のｋに対する依存性

2．2．2　遅延時間に関する考察

先にも述べたように、回路の遅延時間はゲートのイントリンシックな遅延、配線容量による遅延および次段のゲート容量による遅延の3つに分けることができる。表2－3に、それぞれのアーキテクチャに対して3つの遅延時間をnおよびkの関数として式で表したものを示す。いずれのアーキテクチャにおいてもゲートのイントリンシックな遅延と次段のゲート容量による遅延は同様のnに対する依存性を持っている。すなわち、RCAがnに比例、CSKがnの平方根に比例、他の3つがnの対数に比例する。これに対して配線による遅延はすべてnに比例する項が支配的となっている。これは、加算器全体のサイズがビット数nに比例するので、配線の長さもこれに比例するためである。nが増加すればするほど、nが$\sqrt{n}$やLog(n)よりも大きくなるため、RCA以外はビット数の増加に伴い配線による遅延が支配的になる。

表2－3　各アーキテクチャにおける遅延時間の内訳

	Intrinsic Delay	Wiring Delay	Gate Cap. Delay
RCA	0.7n + 3.1	0.1n + 0.5	1.2n + 1.2
CLA	$2.3\log_{10}n + 3.8$	$0.7\log_{10}n + 0.1n + 0.5$	$4\log_{10}n + 2.4$
CSK	$0.4\sqrt{8n+1} + 2.6$	0.1n + 0.5	$0.6\sqrt{8n+1} + 0.6$
k-b CLA × (n/k)-b RCA	$2.3\log_{10}k + 0.7n/k + 3.1$	$0.7\log_{10}k + 0.1n + 0.5$	$4\log_{10}k + 1.2n/k + 1.2$
(n-k)-b CLA + k-b CSA	$2.3\log_{10}(n-k) + 3.8$	$0.7\log_{10}(n-k) + 0.1n + 0.5$	$4\log_{10}(n-k) + 2.4$

表2－4に、16ビット、32ビット、64ビットおよび128ビットの4通りのビット幅における各アーキテクチャの遅延時間の前記3種類の遅延による内訳をパーセンテージで示す。kの値は高速性の観点から、「k-b CLA x (n/k)-b RCA」の場合はn/k＝4とし、「(n-k)-b CLA + k-b CSA」はn/k＝8とした。表より、RCAはビット幅によらず3種類の遅延の割合が変化しないことが分かる。これに対して他の4つのアーキテクチャは、ビット幅の増加に伴って配線による遅延の割合が14～26%増加し、その分だけゲートのイントリンシックな遅延と次段のゲート容量による遅延の割合が減少する。加算器の試作に当たって筆者らが採用した「56-b CLA＋8-b CSA」の場合は、ゲートのイントリンシックな遅延が

31%、配線による遅延が32%、次段のゲート容量による遅延が37%と、それぞれの割合がほぼ等しくなっている。

今後トランジスタの微細化がさらに進み、配線間隔が狭くなると配線容量による遅延の割合は一層増大すると考えられる。また、トランジスタが微細化されてもゲート電極とソース・ドレイン電極の間のオーバーラップ容量はあまり小さくならないため、ゲートのイントリンシックな遅延に比べて、次段のゲート容量による遅延の影響が増大する。したがって、今後はここで行ったような配線容量と次段のゲート容量を正確に考慮した加算器の設計が重要となる。このように、微細化が進んで遅延要因の割合が変化することによって加算器の最適アーキテクチャも変化することが予想される。本節で述べた加算器アーキテクチャの最適化手法は、今後いかなる製造プロセス技術を用いた場合に対しても有効であり、かつ必要不可欠なものになると考えられる。

表2－4　各アーキテクチャの遅延時間の内訳の割合

		16 b	32 b	64 b	128 b
RCA	Intri.	39%	37	36	36
	Wiring	6%	5	5	5
	Gate Cap.	55%	58	59	59
CLA	Intri.	40%	36	31	25
	Wiring	17%	24	32	43
	Gate Cap.	43%	40	37	32
CSK	Intri.	43%	39	36	32
	Wiring	13%	16	21	27
	Gate Cap.	44%	45	43	41
k-b CLA × (n/k)-b RCA (n/k = 4)	Intri.	40%	37	32	26
	Wiring	14%	20	28	40
	Gate Cap.	46%	43	40	34
(n-k)-b CLA + k-b CSA (n/k = 8)	Intri.	39%	36	31	25
	Wiring	18%	24	32	43
	Gate Cap.	43%	40	37	32

2．3　64ビット加算器への適用

2．3．1　回路構成

以上の検討結果に基づいて64ビット加算器の設計を行った。図2－6に全体の回路構成を示す。この加算器は大きく8-b Carry Generate Propagate circuit（CGP）、8-b Carry Look-ahead Circuit（CLC）、8-b Conventional Sum Circuit（CSC）および8-b Modified Carry Select Circuit（MCS）の4つのブロックから成る。入力の2つの64ビットデータ A_0～A_{63}およびB_0～B_{63}は、8ビット毎に8個のブロックに分けられ、それぞれ8ビットCGP回路に入力する。CGP回路はこれらの入力信号から、排他的論理和信号X_0～X_{63}、生成信号G_0～G_{63}および伝搬信号P_0～P_{63}を生成する。8ビットCLC回路は、8個のCGP回路の各最上位出力信号であるG_7, G_{15}, …, G_{63}およびP_7, P_{15}, …, P_{63}から、8ビット毎のキャリー信号C_8, C_{16}, …, C_{56}および最上位のキャリー信号C_{out}の生成を行う。このようにCGP回路とCLC回路がキャリールックアヘッド回路を構成する。

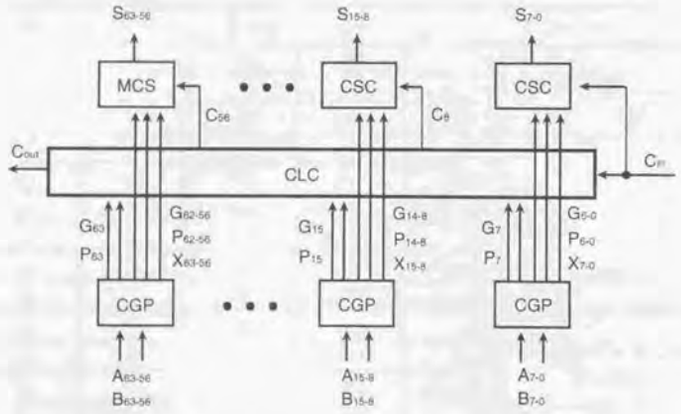


図2－6　64ビット加算器の回路構成

図2－7に例として最下位のCGP回路の構成を示す。各桁に入力する一対の信号は全てAND回路とExclusive-or回路によって論理積と排他的論理和信号（X_7～X_0）に変換され、いわゆるバイナリルックアヘッド方式（BLC）[16], [18]で2進木状に生成信号G_7～G_0および伝搬信号P_7～P_0を生成する。これらの生成はGPAとGPBという2種類の回路で行う。

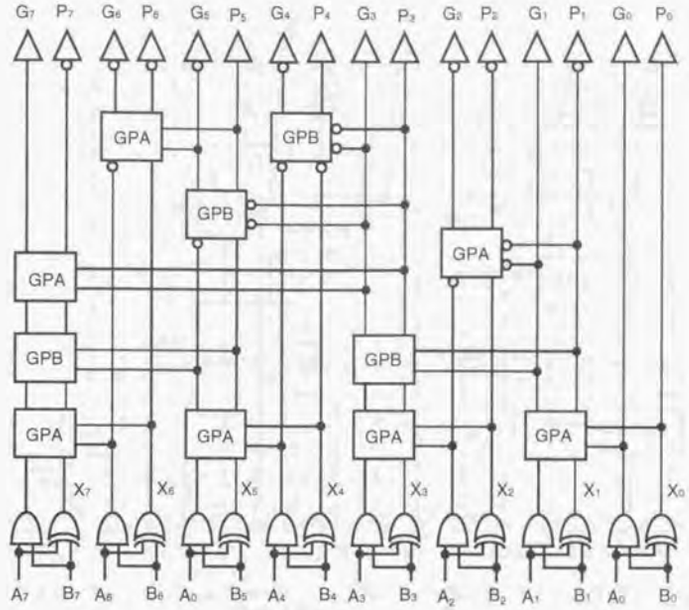


図2－7　最下位のCGP回路の構成

図2－8にCLC回路の構成を示す。入力のG_7, G_{15}, …, G_{63}およびP_7, P_{15}, …, P_{63}からさらにGPA回路とGPB回路によってバイナリルックアヘッドを行い、最後にマルチプレクサ回路（MUX）によってキャリー入力信号C_{in}から8ビット毎のキャリー信号C_8, C_{16}, …, C_{56}, C_{out}を生成する。

図2－9にGPA、GPBおよびMUXの回路構成を示す。GPAとGPBは2つの生成信号（G）と2つの伝搬信号（P）から1つのG信号と1つのP信号を生成する4対2のコンプレッサ回路となっている。また、3種類の回路はいずれもインバータ、トランスミッションゲートおよび2入力NAND/NOR回路という高速の回路から構成されている。

図2－10に下位56桁分の和の値を生成するCSC回路のうちの下位8ビット分の構成を示す。CSC回路も図に示す8ビットの回路を単位として7つのブロックに分割され、それぞれ排他的論理和信号（X）、生成信号（G）、伝搬信号（P）および下位からのキャリー入力C_{in}に基づいて、マルチプレクサ回路（MUX）およびExclusive-or回路によって和の値S_{55}～S_0を発生させる。S_{55}～S_0はクリティカルパス上にはないので、最もトランジスタ数の少ない回路構成としている。

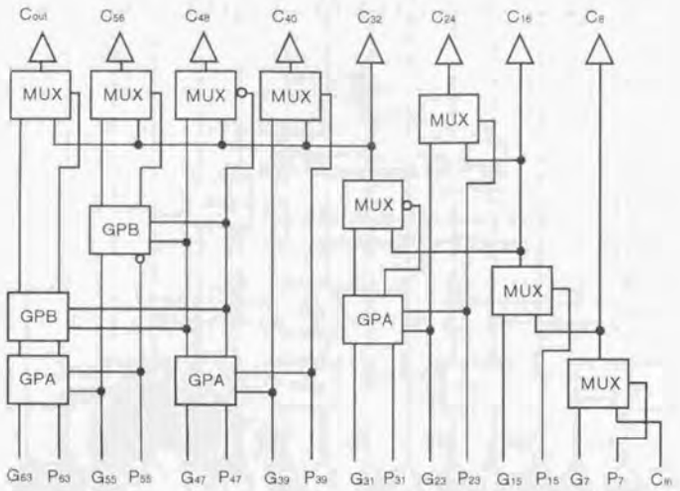


図２－８　CLC 回路の構成

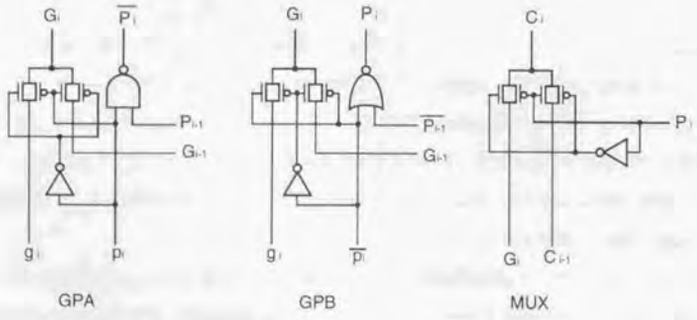


図２－９　GPA, GPB および MUX の回路構成

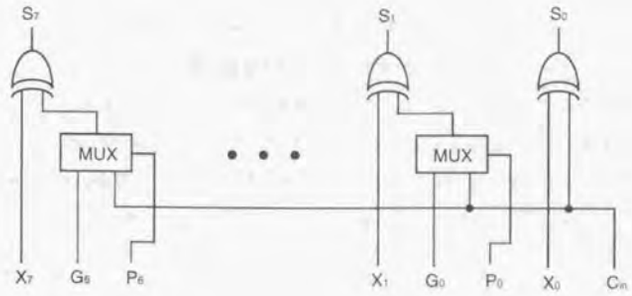


図２－10　８ビット CSC の回路構成

2．3．2　キャリーセレクト回路の改良によるトランジスタの削減

和の値のうち最上位の８桁を決定する回路は、64 ビット加算器のクリティカルパスとなるため、高速性を重視したキャリーセレクト方式（CSA）を採用したが、ここではさらに従来のキャリーセレクト回路よりも素子数を削減した MCS（Modified Carry Select）を提案した。図２－11 に、従来の CSA[11], [12], [15], [16]および MCS の論理構成を示す。

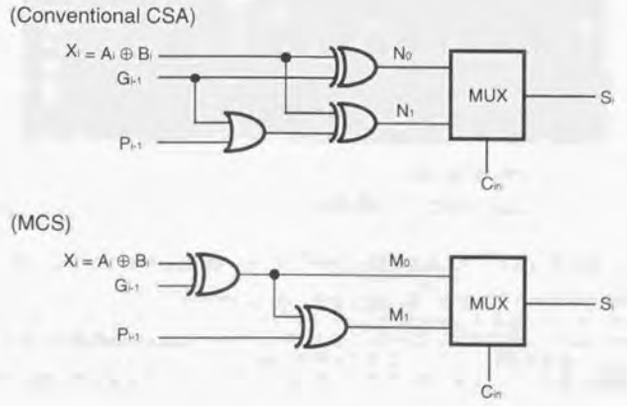


図２－11　従来の CSA と MCS の構成

従来のCSAにおけるN_0とN_1はそれぞれ下位からのキャリー入力C_{55}の値が「0」と「1」の場合のサム（和）の値で、両者のうちのいずれかをマルチプレクサ回路（MUX）でキャリー入力によって選択することで最終的なサムの値を決定する。キャリー信号N_0およびN_1は$X_i = A_i \oplus B_i$、G_{i-1}およびP_{i-1}から１つのOR回路と２つのExclusive-or回路によって生成される。一方、MCSにおけるM0およびM1も同様に$X_i = A_i \oplus B_i$、G_{i-1}およびP_{i-1}から生成されるが、これらは２つのExclusive-or回路のみから生成される。ノードN_0およびN_1とM_0およびM_1との関係は、次式のように表される。

$$N_0 = X_i \oplus G_{i-1} = M_0 \qquad (式2-1)$$

$$N_1 = X_i \oplus (G_{i-1} + P_{i-1}) \qquad (式2-2)$$

ここで、X_i、P_iおよびG_iはそれぞれ、

$$X_i = A_i \oplus B_i \qquad (式2-3)$$

$$P_i = X_0 \bullet X_1 \bullet \cdots \bullet X_i \qquad (式2-4)$$

$$G_i = g_i + X_i \bullet G_{i-1} \qquad (式2-5)$$

で表される。ただしg_iは、

$$g_i = A_i \bullet B_i \qquad (式2-6)$$

である。（式２－３）および（式２－６）より常に$X_i \bullet g_i = 0$となるので（式２－４）および（式２－５）より、

$$P_i \bullet G_i = 0 \qquad (式2-7)$$

となる。したがって、（式２－２）は次のように変形することができる。

$$\begin{aligned} N_1 &= X_i \oplus (G_{i-1} + P_{i-1}) = X_i \oplus \left\{ G_{i-1}(P_{i-1} + \overline{P_{i-1}}) + P_{i-1}(G_{i-1} + \overline{G_{i-1}}) \right\} \\ &= X_i \oplus G_{i-1} \oplus P_{i-1} \\ &= M_1 \end{aligned} \qquad (式2-8)$$

（式２－１）および（式２－８）よりMCSと従来のCSAは論理的に等価となることが分かる。したがって、キャリーセレクト回路として、MCSを用いることができる。

図２－12に64ビット加算器の上位８ビットに適用した８ビットMCSの回路構成を示す。MCSは従来のCSA回路と比べ１ビットあたり１つの２入力ORゲートを少なくできるため、MCSの採用によりキャリーセレクト部の面積を２０％削減することができる。

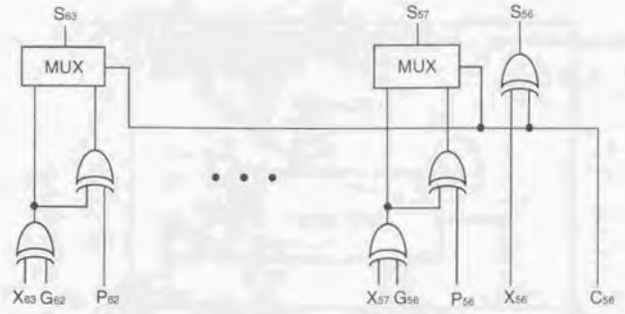


図２－12　８ビットMCS回路

2．3．3　試作および評価結果

設計した64ビット加算器を0.5μmCMOS、３層配線プロセスで試作した。図２－13にチップ写真を示す。チップ面積は1305μm x 207μm（0.27mm²）でトランジスタ数の総和は3044である。

図２－13　64ビット加算器のチップ写真

図２－14に評価に用いたテスト回路を示す。クリティカルパスはB_0からS_{63}で、クリティカルなテストパターンに対して、入力B_0とS_{63}との排他的論理和を取ることにより速度を評価した。この加算器のクリティカルなテストパターンは、

A = FFFFFFFFFFFFFFFF

C_{in} = 0　　（A，B，C_{in}はいずれも16進表示）

として、B=0000000000000000 と B=0000000000000001 とを交互に与えるというもので、これを用いることによってテストを行った。

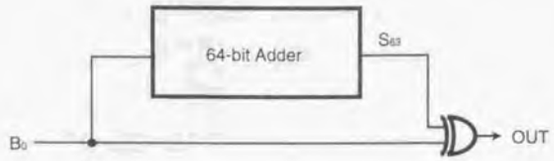


図２－14　テスト回路

図２－15 に測定波形を示す。出力の OUT 信号は B_0 と S_{63} との排他的論理和で生成されるので、まず B_0 が変化すると OUT が変化し、さらに S_{63} が変化するともう一度変化する。したがって、OUT 信号のパルス幅が加算器の遅延時間を表している。図より加算時間は 2.6ns であることが分かる。ただし測定は電源電圧 3.3V、室温の条件で行っている。また、消費電力は 200MHz 動作時において 33mW であった。論理シミュレーションにより、動作時は平均して全体の約 12%のゲートがアクティブ状態となっていることが確認された。 図２－16 に過去に発表された加算器[17]-[19]と本章で試作した加算器の遅延時間と面積の関係を示す。本研究による加算器は最も小面積を実現しており、速度においても 200MHz のマイクロプロセッサで使用可能な高速性を実現している。図２－17 は電源電圧を変化させたときの遅延時間の変化を示している。本加算器が 2.0V まで動作可能であることがわかる。

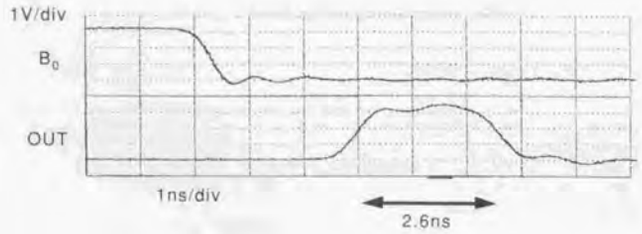


図２－15　入出力波形

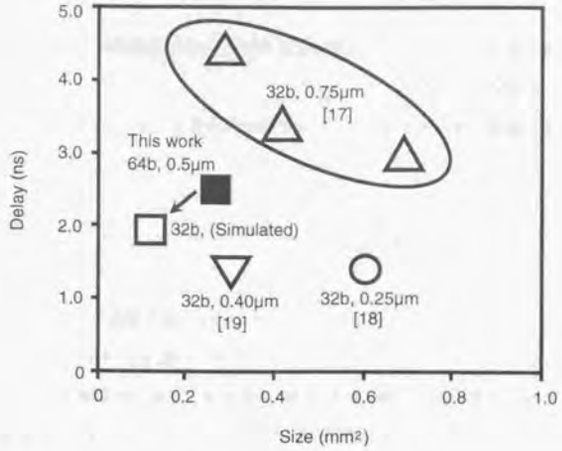


図２－16　加算器の遅延時間と面積の関係

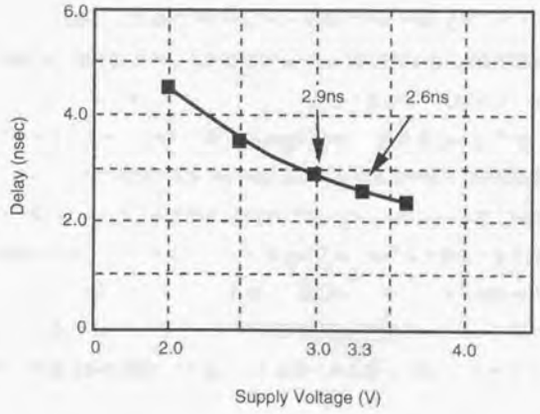


図２－17　遅延時間の電源電圧依存性

2．4　結言

本章においては、高速のデータパスを実現する上で最も重要な構成要素である加算器の高速化および小面積化に関する研究内容に関して述べた。

まず、これまでに提案された加算器アーキテクチャを概括することにより、以下の代表的な5種類のアーキテクチャを抽出した。

(a) リップルキャリー方式

(b) キャリールックアヘッド方式

(c) キャリースキップ方式

(d) キャリールックアヘッド方式とリップルキャリー方式の組み合わせ

(e) キャリールックアヘッド方式とキャリーセレクト方式の組み合わせ

次にこれらのアーキテクチャによる加算器を構成する基本ゲート回路を最適化し、実際にレイアウトすることによってそれぞれの遅延時間と面積を算出した。さらに、この遅延時間と面積の値を用いて、各アーキテクチャによる加算器の遅延時間と面積の式をビット数nの関数として求めた。(d) および (e) のアーキテクチャに関しては、n以外にブロックの分割をあらわすパラメータkも導入してnとkに関する式とした。これらの式から、各アーキテクチャの特徴に関して以下の結果を得た。

(1) リップルキャリー方式は面積は最も小さいが最も遅い。

(2) キャリースキップ方式は、ビット幅が32ビット以下では (c) 、(d) および (e) の各方式と同程度の遅延時間であるが、64ビットになると (c) 、(d) および (e) の各方式よりもそれぞれ27%、20%および29%遅くなる。

(3) 面積は、64ビットでは (c) 、(d) および (e) の3つのアーキテクチャの方が、キャリースキップ方式よりもそれぞれ55%、23%および54%大きくなる。

(4) 16ビット以上では (e) のアーキテクチャが最も高速である。

本研究では高速性を最も重視するため、以上の結果から (e) のアーキテクチャを最適アーキテクチャとして採用することとした。

次に、各アーキテクチャに対して得られた遅延の式に基づいて遅延時間の解析を行い、以下の結論を得た。

(1) ゲートのイントリンシックな遅延と次段のゲート容量による遅延はいずれも、(a) がnに比例，(b) がnの平方根に比例、(c) 、(d) および (e) がnの対数に比例する。

(2) 配線容量にによる遅延は5つのアーキテクチャともnに比例する項が支配的となっている。

(3) nの増加に伴い、配線による遅延が支配的になる。

(4) 最適アーキテクチャでは、ゲートのイントリンシックな遅延、次段のゲート容量による遅延および配線容量にによる遅延の3つの割合がそれぞれ31%、37%および32%とほぼ等しくなっている。

(5) 今後、半導体の微細化がさらに進むと次段のゲート容量による遅延と配線容量による遅延の割合が増大することが予測される。

得られた最適アーキテクチャを用いて64ビット加算器の設計を行った。主な設計の要点は以下の通り。

(1) 全体を8ビット毎に8個のブロックに分割し、各ブロックでバイナリルックアヘッド方式によるキャリーの生成を行う。

(2) 各ブロックから生成されたキャリー信号に基づいて、さらにその上位でバイナリルックアヘッドを行うことにより、上位のキャリー信号を生成する。

(3) 最上位の8ビットは、キャリーセレクト方式によるキャリー信号の生成を行う。この際キャリーセレクト回路を改良して素子数を削減したMCS（Modified Carry Select）回路を提案し、キャリーセレクト部の面積を20%削減した。

設計した64ビット加算器の試作および評価を行い、以下の結果を得た。

(1) 0.5μm CMOS、3層配線プロセスで試作した。チップ面積は1305μm×207μm（≈0.27mm²）トランジスタ数は3044個。

(2) クリティカルなテストパターンによるテストを行い、加算時間2.6nsの動作を確認した。

(3) 本加算器は従来のものと比べて最も小面積であり、速度においても200MHz動作が十分に可能な高速性を実現している。

今後ともプロセスや電源電圧の変化により最適な加算器のアーキテクチャは変化してゆくと考えられるが、本章で行った最適化はどんなプロセスや電源電圧が主流になろうとも有効であり、最適な加算器アーキテクチャを得るのに有効である。

＜参考文献＞

[1] B. Gilchrist, J. H. Pomerene and S. Y. Wong, "Fast Carry Logic for Digital Computers," IRE Trans. Electron. Comput., EC-4, pp.133-136, 1955.

[2] A. Weinberger and J. L. Smith, "A Logic for High-Speed Addition," Nat. Bur. Stand. Circ., No.591 pp.3-12, 1958.

[3] J. Sklansky, "Conditional-Sum Addition Logic," IRE Trans. Electron. Comput., EC-9, pp.226-231, 1960.

[4] R.P.Brent and H.T.Kung, "A regular layout for parallel adders," IEEE Tran. on Computers, vol. c-31, No.3, pp. 260-264, Mar. 1982.

[5] M. A. Bayoumi, G. A. Jullien and W. C. Miller, "An Area-time efficient NMOS Adder," Integration, the VLSI Journal, No.1, pp.317-334, 1983.

[6] V.G.Oklobdzjia and E.R.Barnes, "Some Optimal Schemes for ALU Implementation in VLSI Technology," Proc. of 7th Symposium on Computer Arithmetic, 1985, pp. 2-8.

[7] T. F. Ngai, M. J. Irwin and S. Rawat, "Regular, Area-Time Efficient Carry-Lookahead Adders," IEEE Journal of Paralell and Distributed Computing, No.3, pp.92-105, 1986.

[8] A. Guyot, B. Hochet and J. M. Muller, "A Way to Build Efficient Carry-Skip Adders," IEEE Tran. on Computers, vol. 36, No.10, pp. 1144-1152, Oct. 1987.

[9] A. Rothermel, B. J. Hosticka, G. Troster and J. Arndt, "Realization of Transmission-Gate Conditional-Sum (TGCS) Adders eith Low Latency Time," IEEE J. Solid-State Circuits, vol. 24, No.3, pp.558-561, June 1989.

[10] P. K. Chan and M. D. F. Schlag, "Analysis and Design of CMOS Manchester Adders with Variable Carry-Skip," IEEE Tran. on Computers, vol. 39, No.8, pp. 983-992, Aug. 1990.

[11] J.L.Hennessy and D.A.Patterson, " Computer Architecture : A Quantitative Approach," Morgan Kaufmann Publishers, pp. A31-A39, 1990.

[12] T. Sato, M. Sakate, H. Okada, T. Sukemura and G. Goto, "An 8.5-ns 112-bit Transmission Gate Adder with a Conflict-Free Bypass Circuit," Dig. of Symposium on VLSI Circuit, pp.105-106, May 1991.

[13] N. H. E. Weste and K. Eshraghian, "Principles of CMOS VLSI Design," Addison-Wesley Publishing Company, 1993.

[14] A.Tyagi, "A Reduced Area Scheme for Carry-select Adders(Preliminary Version)," IEEE ICCD Proc., pp. 255-258, 1990.

[15] A.Tyagi, "A reduced Area Scheme for Carry-select Adders," IEEE Tran. on Computers, vol.42, No10, pp. 1163-1170, October 1993.

[16] K.Suzuki, M.Yamashita, J.Goto, T.Inoue and Y.Koseki, "A 2.4-ns,16-b,0.5-μm CMOS Arithmetic Logic Unit for Microprogrammable Video Signal Processor LSIs", CICC Proc., pp. 12.4.1-12.4.4, 1993.

[17] I.S.Hwang and A.L.Fisher, "A 3.1ns 32b CMOS Adder in Multiple Output Domino Logic," Dig. Tech. Papers of ISSCC '88, pp.140-141, Feb. 1988.

[18] M.Suzuki, N.Ohkubo, T.Yamanaka, A.Shimizu and K.Sasaki, "A 1.5ns 32b CMOS ALU in Double Pass-transistor Logic," ISSCC Digest of Technical Papers, pp. 90-91, Feb. 1993.

[19] A. Inoue, Y. kawabe Y. Asada and S. Ando, "A 0.4um 1.4ns 32b Dynamic Adder Using Non-precharge Multiplexers and Reduced Precharge Voltage Technique," Dig. of Symposium on VLSI Circuit, pp. 9-10, June 1995.

[20] T. Yoshida, G. Matsubara, S. Yoshioka, H. Tago, S. Suzuki and N. Goto, "A 500MHz 1-stage 32bit ALU with Self-running Test Circuit," Dig. of Symposium on VLSI Circuit, pp.11-12, June 1995.

[21] H. Morinaka, H. Makino, Y. Nakase, H. Suzuki and K. Mashiko, "A 64bit Carry Look-ahead CMOS Adder using Modified Carry Selec," CICC Proc., pp. 585-588, May 1995.

[22] 森中、牧野、中瀬、鈴木、益子、角、「新しいキャリーセレクト方式(MCS)を使った64ビットキャリールックアヘッドCMOS加算器」、電子情報通信学会技術研究報告、ED95-97, pp.1-6, 1995.

[23] H. Morinaka, H. Makino, Y. Nakase, H. Suzuki, K. Mashiko and T. Sumi, "A 2.6-ns 64-b Fast and Small CMOS Adder," IEICE Transactions on Electronics, Vol.E79-C, No.4, pp.530-537, Apr. 1996.

第３章　乗算器の高速化と小面積化に関する研究

３．１　緒言

乗算器は乗数と被乗数の二つの入力を掛け合わせて結果を出力する演算器で、高速の計算システムにおいては加算器と並んで必要不可欠なハードウエア構成要素である。近年、高速コンピュータはその用途の拡大に伴い、ますます高速かつ高精度の計算能力が要求されており、これを実現するためのキーパーツとして高速かつ多ビットの乗算器が必要とされている。多ビット化に関しては、例えば倍精度の浮動小数点乗算を行うためには、53ビットの仮数部に１ビットの符号を加えた合計54ビットという多ビットの乗算を実施する必要がある（浮動小数点演算に関しては４．１節および５．１節参照）。乗算器は、加算器に比べてはるかに複雑で大きいハードウエアを要求するため、このような多ビットの乗算器を実現する場合、高速設計ばかりでなく、トランジスタ数の少ないシンプルな構造が要求される。このため、乗算器の高速化および素子数の削減に関して、これまで数多くの研究が行われてきた[1]-[61]。

図３－１に従来の高速乗算器の一般的な構成例を示す。乗算器は、大きく分けて乗数と被乗数から部分積を生成する部分積生成部、部分積を足し上げて最後の２つまで減少させる部分積足し上げ部および最後に２つの部分積を足して積の結果を得る最終加算部の３つの部分からなる。

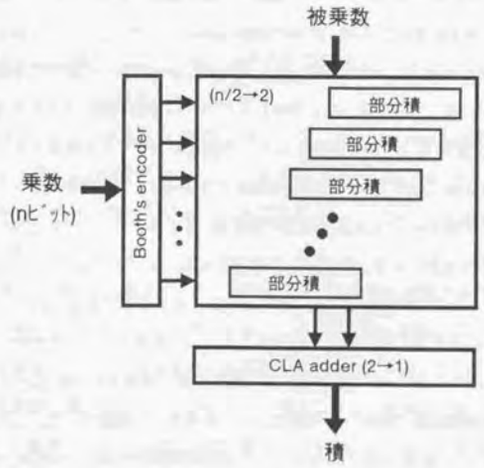


図３－１　従来の高速乗算器の一般的な構成

部分積生成部において有効な手法としてはBoothのアルゴリズム[1],[18]がある。これは、図3－1に示すように入力の乗数を「Booth Encoder」によってエンコードして、これを被乗数に作用させることによって部分積を得る方法で、Boothのアルゴリズムを用いない場合は部分積数が乗数のビット数nと等しくなるのに対して、それよりも部分積数を減少させることができる。エンコードの手法によって2次、3次などのBoothアルゴリズムがあり、一般にn次の場合に部分積数を1/nまで低減することができる。部分積の減少により、トランジスタ数が低減されるだけでなく、部分積の足し上げに要する時間が短縮されるために高速化も同時に実現される。このため、高速かつ小面積の乗算器を実現する上で有効な手法としてこれまで実に多くの乗算器に適用されてきた[6]-[9], [14], [16], [17], [20]-[24], [26], [27], [37]-[39], [41], [43]-[46], [49]。従来の高速乗算器では主に図3－1に示すような2次のBoothアルゴリズムを用いて部分積を半減させる手法が用いられているが、3次以上のものを用いた例も報告されている[37],[43]-[46]。しかし、3次のBoothアルゴリズムは2次のものに比べてエンコードに多くの素子を必要とするため、ハードウエア量の面で不利であり、あまり用いられていない。

次に、生成された部分積を足し上げる部分積足し上げ部においては、ビット数が小さいときは上から順に一つづつ加えていくキャリーセーブ法が用いられてきたが[3]-[6], [13], [14], [19], [20]、多ビットに対してはWallace-tree法[2]が有効であり、主として16ビット以上の乗算器に適用されている[7]-[11], [15]-[17], [21], [22], [23], [24], [26], [27]。Wallace-tree法は、複数の部分積を木状に並列に足し上げる手法で、キャリーセーブ法では遅延時間が入力のビット数nに比例するのに対して、Wallace-tree法では遅延時間をnの対数に比例させることができ、理論的に最も高速な手法とされている。多ビットの乗算器の場合Wallace-tree法は必須となるが、従来のWallace-tree法では、足し上げの基本回路として3入力2出力の全加算器（3-2コンプレッサ）が用いられていたために配線が不規則で複雑になるという問題点があった[11], [16], [17]。このため、チップのレイアウトが困難で設計コストを増大させる要因となっていた。しかも、乗算のビット数が増加するとこの複雑さはますます助長されるという厄介なものであった。Wallace-tree法におけるこの問題点を解決する手法として、従来の3入力2出力の全加算器（3-2コンプレッサ）の代わりに4入力2出力の加算器（4-2コンプレッサ）を用いることが有効である。4-2コンプレッサの使用により、部分積を二進加算木状に足し上げることができるので、配線が飛躍的に単純化される。このため設計コストの増大を防止することができる。4-2コンプレッサ回路は、もともと2つの3-2コンプレッサを組み合わせることにより実現されていた[7]が、その後いくつかの研究機関によって4-2コンプレッサに最適化された回路構成が提案され[16], [22]-[27]、これらによって遅延時間が排他的論理和回路（XOR）4段相当から3段相当へと短縮された。これらの改善された4-2コンプレッサを用いて、近年高速の54×54ビット乗算器が報告されている[23], [24], [26], [27]。

最終加算部においては、通常の多ビット加算を高速に行うために、前章で説明したキャリールックアヘッド方式（CLA）等によって二つの部分積を加算し[6]-[10], [22]-[24], [26] [27]、その結果が積として出力される。

この他に、回路全体をいくつかに分割してレジスタを挿入し、クロック信号によってレジスタ間でデータを送っていくいわゆる「パイプライン化」による高速化[4], [5], [7], [11], [15], [20], [21]や、パイポーラ素子を用いた高速化[12], [21]あるいは多値論理の利用[28]-[32]等が行われている。

高速乗算器を実現するもう一つの有効な手法として、冗長二進数の利用がある。冗長二進数とは、1ビットを表現するのに従来の"1"と"0"の2値ではなく、"1", "0", "-1"の3値を用いるというものである。通常の二進数では加算を行う際に最下位桁から最上位桁に向かってキャリー信号が伝搬してこれが高速化を阻害するのに対し、冗長二進数同士の加算ではこのようなキャリーの伝搬が起こらないため、高速の加算が可能となる。乗算器の場合は、部分積を冗長二進数とすることにより、部分積の足し上げを二進木状に高速に実行することができるので、4-2コンプレッサを使用したときと同じく配線が単純化される。したがって、冗長二進数の使用によって高速かつシンプルな乗算器を構成できる可能性があり、冗長二進数を乗算器に適用するために多くの研究が行われている[28]-[61]。しかしながら、筆者の研究以前には非冗長二進乗算器[23], [24], [26], [27]よりも優れた冗長二進乗算器は報告されていなかった（本論文では部分積を冗長二進数にする乗算器を冗長二進乗算器と呼び、これに対して従来の冗長二進数を用いない乗算器を非冗長二進乗算器と呼ぶ）。それは主に以下の3つの理由によるものである。

1）部分積を通常の二進数から冗長二進数に変換するためのハードウエアが必要なため、ハードウエアが増加し、同時に遅延時間も増大してしまう。

2）冗長二進乗算器において最も重要な構成要素は、2つの冗長二進数を加算して1つの冗長二進数を出力する冗長二進加算器である。このため冗長二進加算器に関して、本研究以前にも種々の回路が提案されている[50]-[54]が、いずれも従来の4-2コンプレッサ[22]-[27]と比較してスピード面で優れたものがなかった。

3）冗長二進乗算器では計算の最後に冗長二進数を通常の二進数に変換する必要があり、これにはキャリーの伝搬の伴う加算操作が必要となる。したがってこの変換には、従来のキャリールックアヘッド加算器をはじめとする高速の加算器が用いられてきた。このため、この部分における冗長二進乗算器の優位性は得られなかった。冗長二進→二進変換に適した回路に関してもいくつかのものが提案されている[55]-[61]が、筆者らの研究時点においてそれらが乗算器に有効に適用された例は報告されていなかった。

以上のように、高速乗算器には冗長二進方式および非冗長二進方式を含めて様々なアーキテクチャが提案されており、従来のものに優る乗算器を実現するためには、優れたものを抽出しそれぞれの長所を最大限に利用する必要がある。しかし、高速乗算器を作る上でもう一つ重要なことはこれを優れたCMOS回路で実現することである。すなわち乗算器は加算器に比べて構成が複雑でトランジスタもはるかに多数となるので、回路設計の際に明確な方針を立てることと、これに基づいて十分な最適化を行うことが重要であり、これを行わなければいくら優れたアーキテクチャを開発しても高速かつ小面積な乗算器を実現することはできない。

また、本研究以前の最高速の54×54ビット乗算器は乗算時間が10nsであったが[23]、通常のマイクロプロセッサにおいては乗算器の前後にレジスタが置かれ、レジスタの遅延時間が加算されるため、この従来の乗算器では本研究の目標性能である100MHzを達成することはできない。これは、パイプライン化を行うことによってさらに動作周波数を上げることが可能ではあるが、その場合計算のレイテンシが増加して場合によっては計算効率の低下を招く。したがって、パイプライン分割を行わずに100MHzを達成することが望ましく、そのためには従来よりも高速な乗算器を実現する必要があった。

筆者らは、以上の観点に基づき、アーキテクチャの最適化とこれを実現するCMOS回路の最適化の両面から、従来よりも高速かつ小面積の乗算器を実現するための研究を行った[62]-[67]。その結果、冗長二進アーキテクチャを用いて通常の二進数による乗算器よりもスピードと面積の両面で優れた乗算器を開発することに成功した。本章では、まず3．2節においてCMOS基本ゲートの遅延時間を解析することにより、高速のクリティカルパスを設計するためのルールを導く。次に、3．3節において冗長二進乗算器のアーキテクチャおよびこれを実現するための最適回路の構成について説明する。さらに3．4節においてこのアーキテクチャおよび回路を用いた54×54ビット乗算器の設計、試作および評価結果について述べ、3．5節においてまとめを行う。

3．2　クリティカルパスの高速化に関する研究[63], [64], [66]

3．2．1　構成ゲートの選定

高速乗算器を実現するためには、回路の中で最も遅いパスであるクルティカルパスを高速化する必要がある。近年、高速化の手法としてトランスミッションゲート（TG）で論理を構築するパストランジスタロジックが注目され種々のLSIに適用されている[10], [25], [26], [27], [68]-[70]。パストランジスタロジックは、インバータとトランスミッションゲート（TG）のみであらゆる種類の回路を構成するため、構造が簡単で排他的論理和回路（XOR）やマルチプレクサ回路（MUX）がゲート1段で構成できるといった長所がある。しかしその反面、TGを直列に多段接続することにより動作スピードが著しく劣化してしまうという欠点がある。この劣化を防ぐためには適当にバッファリング回路を設ける必要があるが、バッファリング回路が多過ぎると回路の段数が多くなるために却って動作が遅くなってしまう。従って、TGの使用には何らかのルールを設ける必要がある。

クリティカルパスを高速化する上で重要なのは、それを高速ゲートで構成することである。CMOS論理ゲートには多種類のゲートが存在するが、その中で高速かつあらゆる論理を構成するのに十分なゲートを選択する必要がある。図3－2にシミュレーションによる各種ゲートの遅延時間のファンアウト依存性を示す。

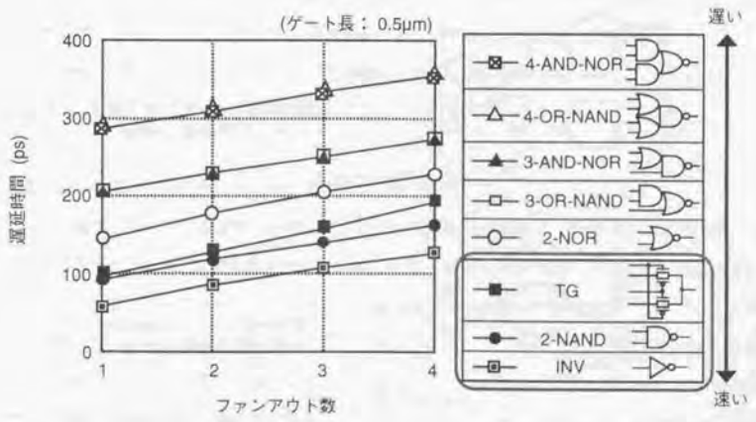
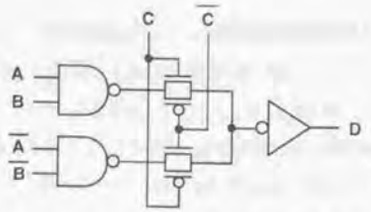
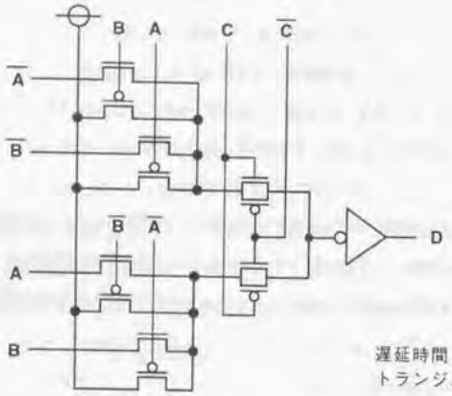


図3－2　各種ゲート遅延時間のファンアウト依存性



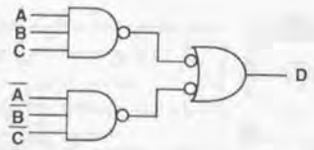
遅延時間　　：310ps
トランジスタ数：14個

(a) 3種類のゲートに最適化された回路



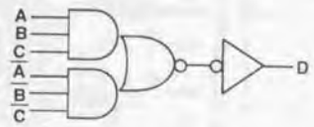
遅延時間　　：390ps
トランジスタ数：14個

(b) パストランジスタロジック



遅延時間　　：370ps
トランジスタ数：16個

(c) 2段のNAND回路



遅延時間　　：500ps
トランジスタ数：14個

(d) 6入力複合ゲート

図3－3　各種回路の比較

ここでは、インバータ、2入力NAND、TG、2入力NOR、3入力OR-NAND 、3入力AND-NOR、4入力OR-NANDおよび4入力AND-NORの8種類のゲートの遅延時間が示されている。シミュレーションには0.5μmCMOS のパラメータを用いた。電源電圧は3.3Vである。この図から分かるように、インバータ、2入力NANDおよびTGの3種類が最も高速である。2入力NORや多入力ゲートは、スピードが遅くクリティカルパスには適さない。2入力NAND回路はTGよりも高速であるため、パストランジスタロジックに2入力NANDを加えることで、さらに高速な回路を構成することができる。パストランジスタロジックでも全てのロジックを実現することができるので、2入力NANDを加えてもパストランジスタロジックと同様にあらゆる回路を構成することができる。言い換えれば、回路をインバータ、2入力NANDおよびTGの3種類に最適化することにより、パストランジスタロジックよりも高速の回路を実現することができる。

図3－3にこの最適化の例を示す。この図ではブール式：$D = A \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C}$を4通りの回路で実現した場合の回路構成と遅延時間およびトランジスタ数を示している。遅延時間については、ファンアウト4の負荷条件でシミュレーションを行った。図中(a)は筆者らの提案する回路で、2入力NAND、TGおよびインバータの3種類に最適化された構成となっている。この回路の遅延時間は310ps、トランジスタ数は14である。(b)はパストランジスタロジックによるもので、TGとインバータのみから構成されている。この回路のトランジスタ数は(a)と同じく14であるが遅延時間は390psと(a)よりも80ps遅い。(c)は、3入力および2入力の2段のNAND回路で実現した場合で、トランジスタ数は(a)、(b)よりも2個多いものの、遅延時間はパストランジスタロジック(b)よりも20ps高速である。(d)は6入力の複合ゲートを用いた場合でトランジスタ数は(a)、(b)と同じであるが、遅延時間は500psと非常に遅く、多入力複合ゲートはクリティカルパスには適さないことが分かる。これらの結果から、インバータ、2入力NANDおよびTGの3種類のゲートに最適化された(a)の回路が遅延時間、トランジスタ数ともに最も優れており、クリティカルパスに最適であることが分かる。

3．2．2　波形劣化の防止による高速化の検討

回路を3種類のゲートに最適化する際に注意しなければならないのは、TGを直列に多段接続した際の速度の劣化である[24]。この劣化を調べるためにトランスミッションゲート（TG）の遅延時間に関するシミュレーションを行った。図3－4にシミュレーションに用いた回路を示す。ゲートとしては最も一般的に用いられる2入力マルチプレクサを想定した。ゲート長は0.5μmである。図中(1)、(2)および(4)はそれぞれTGを1段、2段および3段に直列接続したものであり、(3)および(5)はそれぞれ2段および3段の接続において最終段の前段にインバータを挿入したものである。このインバータはバッファ回路として動作する。「load」としてはファンアウト数を変化させて遅延時間を計算した。

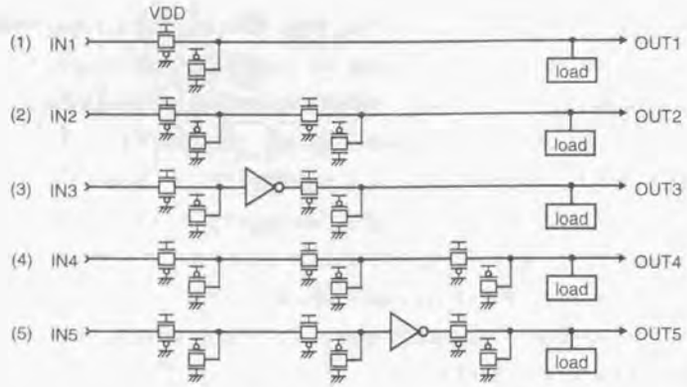


図３－４　シミュレーションに用いた回路

図３－５にシミュレーション結果を示す。これは、図３－４における５種類の回路に対する遅延時間のファンアウト依存性である。TG ２段の接続の場合（(2)と(3)の場合）、直列接続(2)は、バッファを挿入した場合(3)よりもファンアウト９程度まで高速である。しかし、TG ３段の接続（(4)と(5)）になると、グラフの交点はファンアウト４まで低下する。すなわち、負荷がファンアウト４以上では、バッファを挿入した方が高速になる。通常の設計では、クリティカルパスは、ファンアウト４以上となるのが普通であるため、TG を３段以上直列につなぐことは高速化には不利である。

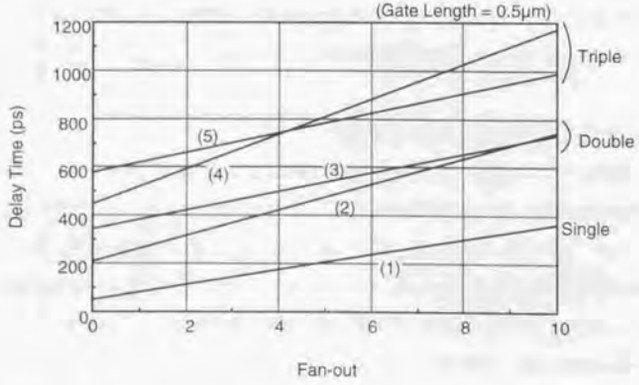


図３－５　各回路の遅延時間のファンアウト依存性

TG の直列接続には、TG 自体が遅くなるという問題点の他に、TG 出力の信号のなまりの問題がある。図３－６に、ファンアウト４に対する図３－４の(1)、(2)および(4)の回路の出力信号波形を示す。TG の段数の増加とともに出力波形が著しくなまって劣化することが分かる。特に３段の場合（(4)の場合）は、出力の遷移時間が、2ns を超えている。このような波形の劣化により、次段の回路の動作が遅くなる。表３－１に、TG の直列接続を１段、２段および３段とした場合の次段の回路の遅延時間 Td と１段の場合に対する遅延の増加量 ΔT を示す。次段の回路としては、ファンアウト４を駆動するインバータを仮定している。表から分かるように、TG の直列接続が２段の場合、次段の動作は１段の時と比べて 32ps(20%)劣化し、直列接続が３段になると劣化は 63ps(40%)に増大する。従って、図３－５より TG の直列接続が３段の場合はファンアウト４でもバッファを挿入したものよりも動作スピードが遅くなる。したがって、ファンアウト４においても TG ３段の直列接続はバッファを挿入したものよりも不利となることが分かる。またこのようなスピード面の劣化に加えて、TG の直列接続による波形劣化のために、例えばトランジスタのしきい値電圧のようなパラメータの変化に対して回路動作が影響されやすくなり、動作が不安定になるという問題も生じる。従って、TG ３段以上の直列接続は避けるべきである。

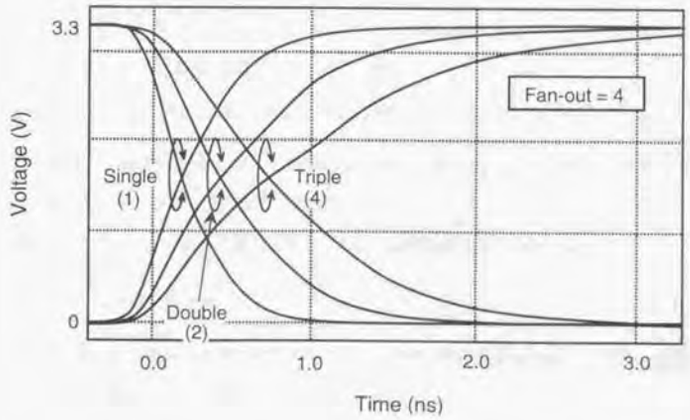


図３－６　ファンアウト４での(1)、(2)、(4)の出力波形

表３－１　次段の回路の遅延時間

	Td (ps)	ΔT (ps)
Single Stage	158	0
Double Stages	190	32 (20%)
Triple Stages	221	63 (40%)

以上の考察により、乗算器におけるクリティカルパスの設計方法として以下の２つのルールを採用することとした。

(i) クリティカルパスはインバータ、２入力NANDおよびTGの３種類のゲートのみで構成する。

(ii) 直列接続のTGは２段までとする。

３．３　冗長二進数を用いた乗算器の高速化と小面積化に関する研究

３．３．１　冗長二進数表現法

最初に、本論文で使用する冗長二進数について説明する。冗長二進数は一つの桁に"1","0","-1"の３値を取らせるため、一つの桁を表現するのに２ビットの二進数を必要とする。この２ビットによる表現のやり方には様々のものある[50]が、本論文では基本的表現として以下のものを用いる。すなわち、１桁を表す２ビットのペア（a_i^+, a_i^-）の値を、

$$(a^+, a^-) \equiv a^+ - a^- \qquad (式３－１)$$

で定義する。従って（0, 1）=-1、(0, 0) = (1, 1) =0, (1, 0) = 1となる。この表現を用いると、二つの二進数の差および和を容易に冗長二進数で表すことができる。PとQをnビットの二進数とすると、それぞれ以下のように表すことができる。

$$P = p_{n-1}\, p_{n-2}\, p_{n-3} \cdots p_0 = \sum_{i=0}^{n-1} p_i \cdot 2^i \qquad (式３－２)$$

$$Q = q_{n-1}\, q_{n-2}\, q_{n-3} \cdots q_0 = \sum_{i=0}^{n-1} q_i \cdot 2^i \qquad (式３－３)$$

PとQの差は、次のように容易に表すことができる。

$$\begin{aligned} P - Q &= \sum_{i=0}^{n-1} (p_i - q_i) \cdot 2^i \\ &= \sum_{i=0}^{n-1} (p_i, q_i) \cdot 2^i \\ &= (p_{n-1}, q_{n-1})(p_{n-2}, q_{n-2})(p_{n-3}, q_{n-3}) \cdots (p_0, q_0) \end{aligned} \qquad (式３－４)$$

例えば、４桁の二進数"0100"(=4)と"1110"(=14)との差は、単に各桁をそれぞれペアとするだけで"(0,1)(1,1)(0,1)(0,0)"(=-8+0-2+0=-10)と表すことができる。

次に加算についても、上で述べた考えを適用することができる。すなわちP+Qは、

$$P + Q = P - (-Q) \qquad (式３－５)$$

と表すことができるが、-Qは２の補数表現を使うとQの各桁の"1"と"0"を反転させて最下位桁に"1"を加えることによって得ることができる。すなわち、

$$-Q = \overline{Q} + 1 \qquad (式３－６)$$

$$= 1\, \overline{q_{n-1}}\, \overline{q_{n-2}}\, \overline{q_{n-3}} \cdots \overline{q_0} + 1 = -1 \cdot 2^n + \sum_{i=0}^{n-1} \overline{q_i} \cdot 2^i + 1. \qquad (式３－７)$$

ただし、$\overline{Q}$はQの各桁を反転させたもので、最上位桁に現れる"1"は負の値を表現するための符号ビットである。PとQの和は、（式３－５）よりPからQを引けばよいので、（式３－４）を当てはめることにより、次式を得る。

$$P+Q=(1,0)\cdot 2^{n}+\sum_{i=0}^{n-1}(p_i,\overline{q_i})\cdot 2^{i}+(0,1) \qquad (式3-8)$$

$$=(1,0)(p_{n-1},\overline{q_{n-1}})(p_{n-2},\overline{q_{n-2}})\cdots(p_0,\overline{q_0})+(0,1) \qquad (式3-9)$$

（式３－９）における第２項の(0, 1)は（式３－６）および（式３－７）において２の補数を作る際に現れた"1"に対応するもので、最下位桁に加えられるものである。

実際には、冗長二進乗算器においては定義（式３－１）から導かれる以下の二つの表現がしばしば用いられる。第１のものは、

$$a^{+}\equiv a^{+}\cdot\overline{a^{-}},\qquad a^{-}\equiv\overline{a^{+}}\cdot a^{-} \qquad (式3-10)$$

である[53]。これは、（1, 1）を（0, 0）に変換するもので、この変換によって後節で述べるように回路構成を単純化することができる。第２のものは、

$$a^{s}\equiv\overline{a^{+}}\cdot a^{-}=a^{-},\quad a^{a}\equiv a^{+}\oplus a^{-}=a^{+}+a^{-} \qquad (式3-11)$$

で、これはa^sとa^aがそれぞれ符号および絶対値と表す符号／絶対値表現である[49]。本論文では、（式３－１），（式３－10）および（式３－11）の表現をそれぞれExp.1, Exp.2およびExp.3呼ぶこととする。表３－２に３つの表現方法をまとめる。

表３－２　冗長二進数の表現方法

Value	Exp. 1		Exp. 2		Exp. 3	
	a^+	a^-	a^+	a^-	a^s	a^a
-1	0	1	0	1	1	1
0	0 1	0 1	0	0	0	0
1	1	0	1	0	0	1

3．3．2　冗長二進数による部分積生成法の提案[62], [65], [67]

従来の冗長二進乗算器においては、冗長二進数の部分積（冗長二進部分積と呼ぶ）を生成するのに１つの二進部分積から１つの冗長二進部分積を発生させていた[33]-[36]。しかし、冗長二進数は１桁を表現するのに２ビットの信号を必要とするため、この方法では実質的にビット数が増加してハードウエアを増大させるという欠点を伴っていた。これに対して、冗長二進数による部分積の生成を、２つの二進部分積から効率よく行う手法も提案されている[38], [40], [49]。しかし、これらの研究では冗長二進数の表現法として（式３－１）の表現Exp.1ではなく（式３－11）の符号／絶対値表現Exp.3が用いられていたため、各ビットはORゲートや排他的論理和（XOR）ゲートを用いて符号ビットと絶対値ビットに変換する必要があり、このために依然としてハードウエアと遅延時間の両方の増加を伴っていた。そこで、本節ではさらにこれを改良して、特別なハードウエアなしに変換を実現する以下の手法を提案する。

ここで提案する手法は、冗長二進数として（式３－１）で表される表現Exp.1を用い、２つの二進部分積から（式３－９）にしたがって、１つの冗長二進部分積を得るというものである。すなわち、通常の二進数で表わされた２つの部分積PとQを考えると、前節で説明したように２つの部分積の和P+Qは（式３－９）で表される。この式は、２つ二進数の部分積の和P+Qが、Qの各桁を反転してそれぞれをPの各桁とペアにして、さらに最上位桁の一桁上に(1, 0)を加え、最下位桁に(0, 1)を加えるだけで容易に１つの冗長二進数として表現できることを示している。CMOS回路でこれを実現する場合、単にペアを作るという操作には特別なハードウエアは全く要求されない。また、Qを反転させる操作も、Qを生成する際に反転出力を得る論理を用いればよいので特にハードウエアを増加させない。(1, 0)や(0, 1)といった定数を加える操作も最上位桁や最下位桁の論理に少し手を加えるだけで実現することができるのでハードウエアは増加しない。以上のことから、本方式で冗長二進数の部分積を生成することにより、ハードウエアの増加を全く伴うことなく二つの二進部分積から効率よく冗長二進部分積を得ることができる。

図３－７に一例として２つの部分積をそれぞれA=10100110(=-90)およびB=01101101(=109)とした場合の冗長二進部分積の生成法を示す。ただし、AおよびBは２の補数で表示されているものとする。まず、Bの符号を反転させるために各桁の"1"と"0"を反転して最下位桁に"1"を加える。その結果、Bは"10010010"と"1"に変換される。次に、Aと反転されたBとでペアを作ると、そのペアは冗長二進部分積(1,1)(0,0)(1,0)(0,1)(0,0)(1,0)(1,1)(0,0) (=20)と付加的な冗長二進ビット(0,1) (= -1)になる。この冗長二進部分積(=20)と付加的な冗長二進ビット(=-1)との和は19で、これは非冗長二進の２つの部分積A(=-90)およびB(=109)の和に等しい。このように、２つの非冗長二進部分積の和の値を持つ冗長二進部分積を、一方を反転して他方とペアとするだけで容易に生成することができる。

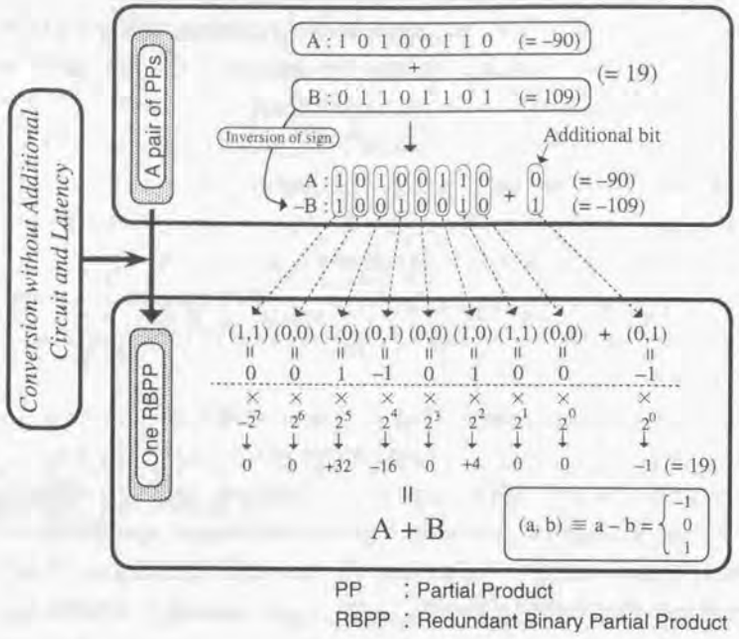


図３－７　冗長二進部分積の生成例

３．３．３　冗長二進加算器の高速化の研究[62], [65], [67]

３．３．３．１　従来の4-2コンプレッサとその問題点

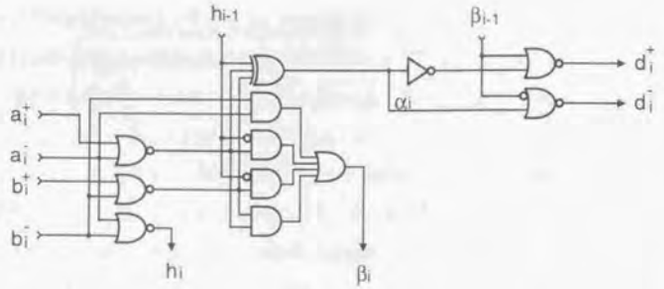
生成された冗長二進部分積は、冗長二進加算器（RBA：Redundant Binary Adder）を基本回路とする加算器のアレイで足し上げられる。高速乗算器の場合、この足し上げはWallace-tree 法によって行われるが、それでもWallace-tree 部が乗算器の遅延時間のうちの大きな部分を占めるため、高速乗算器を実現するためには、冗長二進加算器の高速化が重要となる。このため、本研究以前にも冗長二進加算器に関してはいくつかののアルゴリズムや回路が研究されていた[50]-[54]。冗長二進加算器は、２個の冗長二進数を加算して１個の冗長二進数として出力する加算器で４入力２出力を有するため、4-2 コンプレッサとして働く。したがって、これは従来の非冗長二進乗算器における4-2 コンプレッサに相当するが、

非冗長二進方式においてはいくつかのスピード面で優れた4-2 コンプレッサが提案されているのに対して[22]-[27]、３．１節で述べたようにこれらよりも優れた冗長二進加算器は本研究以前には作られていなかった。そこで本節では、高速の冗長二進加算器を開発するに当たり、まず従来の冗長二進乗算器および非冗長二進乗算器における4-2 コンプレッサを概括し、問題点を分析する。

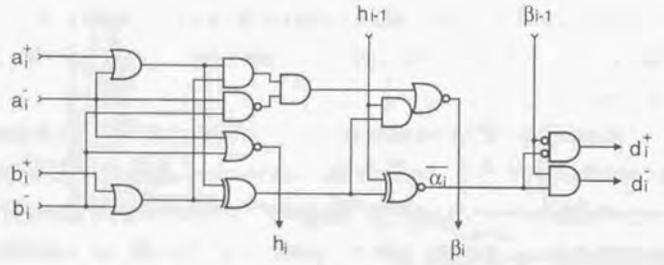
図３－８に従来の代表的な冗長二進加算器を示す。図には３種類の回路が示されており、(a)[46]をRBA1、(b)[53]をRBA2、(c)[49]をRBA3と呼ぶこととする。RBA1とRBA2は入出力の冗長二進数（(a_i^+, a_i^-)、(b_i^+, b_i^-)および(d_i^+, d_i^-)）としてExp.2 の表現を用いる。従来の冗長二進乗算器ではこの表現が単純で最もよく用いられている。これに対してRBA3では入出力信号に対し、Exp.3 の符号／絶対値表現が用いられる。a_i^s、b_i^sおよびd_i^sが符号を表し、a_i^a、b_i^aおよびd_i^aが絶対値を表す。各冗長二進加算器とも一つ上位桁へ入力する２種類の中間信号h_iとβ_iとを生成するが、隣り合う２桁の信号、h_iとh_{i+1}およびβ_iとβ_{i+1}は、いずれも互いに独立な信号なので、最下位桁から最上位桁へ向かうキャリー信号の伝搬は生じない。このために、冗長二進数どうしの加算は通常の二進数の加算に比べて高速に行うことができる。

図３－９に従来の非冗長二進方式の乗算器に用いられている代表的な4-2 コンプレッサを示す。図には３種類の回路構成が示されており、(a)[23]をNBA1、(b)[24]をNBA2、(c)[25]-[27]をNBA3と呼ぶこととする。いずれの回路も４つの入力信号a_i^1、a_i^2、a_i^3およびa_i^4と２つの出力信号s_iとc_i^1とを持つ。また、冗長二進加算器におけるh_i、β_iと同様の中間信号c_i^2を生成するが、これも隣り合う２桁の信号、c_i^2とc_{i+1}^2が互いに独立なので最下位桁から最上位桁へ伝搬することはない。非冗長二進乗算器の4-2 コンプレッサにおいては、s_iはサム、c_i^1とc_i^2はキャリー信号として働く。4-2 コンプレッサが次段にも直列に接続される場合、c_i^1は次段において１桁上位の4-2 コンプレッサに入力する。

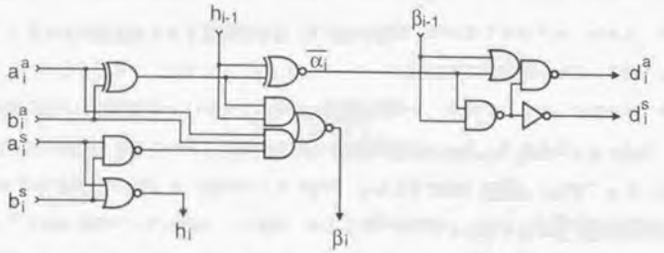
上に示した従来の4-2 コンプレッサは、入力から出力までのゲート段数が比較的少なく高速動作に適したものであるが、以下のような２つの問題点がある。第１の問題点は、３入力以上の多入力ゲートを持つ点である。NBA3を除く５つの4-2 コンプレッサは、いずれもそのような多入力ゲートを持つ。３．２節で述べたように、一般にゲートの遅延時間は入力のファンイン数の増加とともに急速に増大する。したがってこれらの多入力ゲートが速度を劣化させる原因となっている。第２の問題点は、トランスミッションゲート（TG）の使い方である。RBA1およびRBA3は、複数段接続するとTGの直列接続が３段以上続くので、スピードの劣化を防ぐために適所にバッファリング回路を挿入する必要がある。また、RBA3および３つのNBAは、初段あるいは最終段（あるいはその両方）にTGが使われているが、このようなTG の使用は高速動作には適さない。３．２節で示したように、TGは従来の4-2 コンプレッサに使用されている他のゲートと比較して一般に負荷容量に対する遅延時間の依存性が大きく、またWallce-tree 法によって部分積を足し上げる乗算器では4-2 コンプレッサは大きな配線容量を



(a) RBA1 [46]

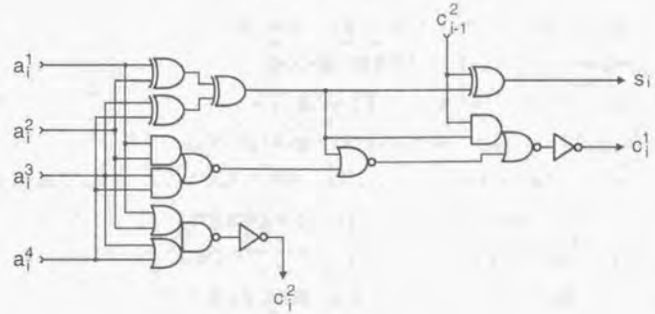


(b) RBA2 [53]

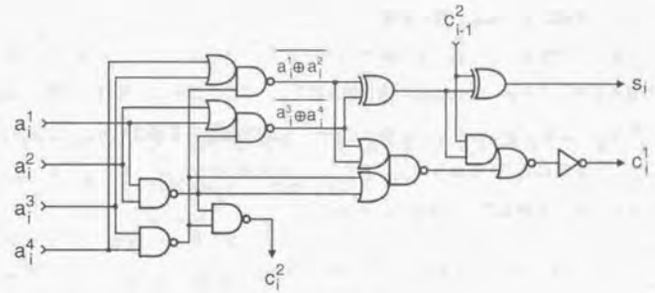


(c) RBA3 [49]

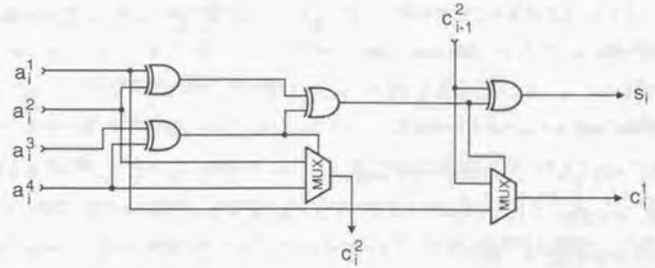
図３－８　従来の代表的な冗長二進加算器



(a) NBA1 [23]



(b) NBA2 [24]



(c) NBA3 [25]-[27]

図３－９　従来の非冗長二進方式の乗算器に用いられている代表的な 4-2 コンプレッサ

駆動する必要があるため，出力のTGはインバータなどによってバッファリングする必要が生じる。また，TGはON時には抵抗として働くため，TGにおいては出力信号だけでなく入力信号にもなまりが生じる[24]。このなまりが大きいと信号の遅延時間に重大な劣化を生じるため，TGの前段もバッファリングが必要である。すなわち，高速動作の必要な4-2コンプレッサの入力段や出力段にTGを適用する際には入力段の前段および出力段の後段にバッファを設ける必要がある。このように，TGを用いた従来の4-2コンプレッサでは，高速動作を実現するために多数のバッファリング回路が必要で，逆にこのようなバッファの付加のために，回路の段数が増加して遅延時間が増大してしまう。

これらの問題点を解決して従来よりも高速の4-2コンプレッサを得るためには，入出力部でのTGの使用を避け，3．2節で定めたルールに従った回路の最適化が必要である。

3．3．3．2　高速冗長二進加算器の提案

本節では，筆者らが提案する冗長二進加算器（4-2コンプレッサ）をプール式に基づいて導出する。また，この導出を通じて冗長二進加算器の定式化を行う。ここで提案する回路は，冗長二進加算器のアルゴリズムを3．2節で示したルールに従って高速のCMOS回路に最適化することにより，NOR回路や多入力ゲートを使用せずに実現される。まず，2つの冗長二進数（a_i^+, a_i^-）と（b_i^+, b_i^-）を足し合わせて和（d_i^+, d_i^-）を得ることを考える。すなわち，

$$(a_i^+,\ a_i^-) + (b_i^+,\ b_i^-) \rightarrow (d_i^+,\ d_i^-) \qquad (式3-12)$$

一般的な議論をするために，入力の冗長二進数（a_i^+, a_i^-）および（b_i^+, b_i^-）は，3．3．1節で述べた3種類の冗長二進表現Exp.1, Exp.2およびExp.3のうち最も一般性の高いExp.1で表わされているとする。したがって，入力の組み合わせは表3－3に示すように全部で16となる。これら16通りの入力は加算結果の値によって5つの場合case1～case5に分けることができる。表3－3にそれぞれの場合における中間和（s_i^+, s_i^-）および中間桁上げ信号（c_i^+, c_i^-）を示す。中間桁上げ信号（c_i^+, c_i^-）は1つ上位桁の中間和に加算されるべきものである。5つの場合の内case2とcase3とは1桁下位から生成されるh_{i-1}の値によってさらに2つの場合に分けられる。h_iは，中間和（s_i^+, s_i^-）と1桁下位から来る中間桁上げ信号（c_{i-1}^+, c_{i-1}^-）との衝突を避けるために導入された信号で，中間桁上げ信号の値が1か0の時はh_i=1となり，中間桁上げ信号の値が-1か0の時はh_i=0となるように設定される。このようなh_iの定義には，h_i=0の場合の分け方によって何通りかの方法が考えられ[50]，冗長二進加算器の構成はこのh_iの定義の仕方に依存して決定される。したがって高速化に有効なh_iの定義を採用する必要がある。

表3－3　冗長二進加算における入出力信号の値

Case	Value	(a_i^+, a_i^-) (b_i^+, b_i^-)	h_{i-1}	Intermediate carry sum (c_i^+, c_i^-) (s_i^+, s_i^-)
1	0	(0, 0) (1, 0) (0, 1) (0, 0) (1, 1) (1, 1) (0, 0) (0, 1) (1, 0) (1, 1) (0, 0) (1, 1)	any	(0, 0) (0, 0)
2	-1	(0, 1) (0, 0) (0, 1) (1, 1)	1	(0, 0) (0, 1)
		(0, 0) (0, 1) (1, 1) (0, 1)	0	(0, 1) (1, 0)
3	1	(1, 0) (0, 0) (1, 0) (1, 1)	1	(1, 0) (0, 1)
		(0, 0) (1, 0) (1, 1) (1, 0)	0	(0, 0) (1, 0)
4	-2	(0, 1) (0, 1)	any	(0, 1) (0, 0)
5	2	(1, 0) (1, 0)	any	(1, 0) (0, 0)

ここでは単純なCMOS回路で実現でき，高速の冗長二進加算器を構成できるという観点から以下の3つのh_iに対する定義Def.1～Def.3について考察する。

Def. 1:　$h_i = \overline{\overline{a_i^+ \cdot a_i^-} + \overline{b_i^+ \cdot b_i^-}}$　（式3－13）

Def. 2:　$h_i = a_i^+ \cdot \overline{a_i^-} + b_i^+ \cdot \overline{b_i^-}$　（式3－14）

Def. 3:　$h_i = \overline{a_i^-} \cdot (a_i^+ \oplus a_i^-) + \overline{b_i^-} \cdot \overline{(a_i^+ \oplus a_i^-)}$　（式3－15）

表3－4に各定義に対するh_iの真理値表を後に用いる他の変数l_i，m_iおよびn_iの値とともに示す。

最終的な和の値（d_i^+, d_i^-）は，以下の式で定義する。

$$d_i^+ \equiv (s_i^+ + c_{i-1}^+) \cdot \overline{(s_i^- + c_{i-1}^-)} \qquad (式3-16)$$

$$d_i^- \equiv \overline{(s_i^+ + c_{i-1}^+)} \cdot (s_i^- + c_{i-1}^-) \qquad (式3-17)$$

この定義によって（d_i^+, d_i^-）が（1, 1）となる場合を除外することができる。すなわち，（d_i^+, d_i^-）は（1, 0），（0, 0）および（0, 1）の3値に限られ，したがって（式3－10）で定義されるExp.2の表現となる。後に示すように（d_i^+, d_i^-）をExp.2とすることによって冗長二進加算器の構成を単純化することができる。以上の定義を踏まえ，以下でDef.1～Def.3の各定義に対して高速CMOS回路に適した(d_i^+, d_i^-)の式を導出する。

表３－４　h_i, l_i, m_iおよびn_iの真理値表

(a_i^+, a_i^-)	(b_i^+, b_i^-)	Value	h_i			l_i	m_i			n_i		
			Def.1	Def.2	Def.3		Def.1	Def.2	Def.3	Def.1	Def.2	Def.3
(0, 0)	(0, 0)	0	1	0	1	0	0	1	0	1	0	1
(0, 1)	(0, 0)	-1	0	0	0	1	0	0	0	0	0	0
(1, 0)	(0, 0)	1	1	1	1	1	0	0	0	0	0	0
(1, 1)	(0, 0)	0	1	0	1	0	0	1	0	1	0	1
(0, 0)	(0, 1)	-1	0	0	0	1	0	0	0	0	0	0
(0, 1)	(0, 1)	-2	0	0	0	0	0	0	0	1	1	1
(1, 0)	(0, 1)	0	0	1	1	0	1	0	0	0	1	1
(1, 1)	(0, 1)	-1	0	0	0	1	0	0	0	0	0	0
(0, 0)	(1, 0)	1	1	1	1	1	0	0	0	0	0	0
(0, 1)	(1, 0)	0	0	1	0	0	1	0	1	0	1	0
(1, 0)	(1, 0)	2	1	1	1	0	1	1	1	0	0	0
(1, 1)	(1, 0)	1	1	1	1	1	0	0	0	0	0	0
(0, 0)	(1, 1)	0	1	0	0	0	0	1	1	1	0	0
(0, 1)	(1, 1)	-1	0	0	0	1	0	0	0	0	0	0
(1, 0)	(1, 1)	1	1	1	1	1	0	0	0	0	0	0
(1, 1)	(1, 1)	0	1	0	0	0	0	1	1	1	0	0

(a) Def.1 の場合

表３－３のルールから、中間和（s_i^+, s_i^-）と中間桁上げ信号（c_i^+, c_i^-）に対して以下の式を得る。

$$s_i^+ = l_i \cdot \overline{h_{i-1}} \qquad (式3-18)$$

$$s_i^- = \overline{l_i} \cdot h_{i-1} \qquad (式3-19)$$

$$c_i^+ = m_i \cdot h_i + l_i \cdot h_i \cdot h_{i-1} \qquad (式3-20)$$

$$c_i^- = n_i \cdot \overline{h_i} + l_i \cdot \overline{h_i} \cdot \overline{h_{i-1}} \qquad (式3-21)$$

表３－４にl_i, m_iおよびn_iの真理値表を示す。l_iは和の値が1か-1のときに真となり、m_iは和の値が2の場合と0となるいくつかの場合に真となる。また、n_iは和の値が-2の場合と0となる場合のうちm_iが真とならない場合に真となる。また、ここではm_iとn_iは表の中のDef.1に対応する値をとる。l_i, m_iおよびn_iは次の式を満たす。

$$l_i + m_i + n_i = 1 \qquad (式3-22)$$

（式３－18）～（式３－21）を式（式３－16）と（式３－17）に代入して（式３－22）を用いるとd_i^+とd_i^-に対して次の式が得られる。

$$d_i^+ = (l_i \oplus h_{i-1}) \cdot (m_{i-1} + l_{i-1} \cdot h_{i-2}) \qquad (式3-23)$$

$$d_i^- = \overline{(l_i \oplus h_{i-1})} \cdot \overline{(m_{i-1} + l_{i-1} \cdot h_{i-2})} \qquad (式3-24)$$

l_iとm_iは表３－４より次の式で表わされる。

$$l_i = a_i^+ \oplus a_i^- \oplus b_i^+ \oplus b_i^- \qquad (式3-25)$$

$$m_i = k_i \cdot \overline{l_i} \qquad (式3-26)$$

ただし、

$$k_i = a_i^+ \cdot \overline{a_i^-} + b_i^+ \cdot \overline{b_i^-} \qquad (式3-27)$$

（式３－23）～（式３－27）をまとめることによって、冗長二進数の加算に関して以下の関係式を得ることができる。

$$d_i^+ = \alpha_i \cdot \overline{\beta_{i-1}} \qquad (式3-28)$$

$$d_i^- = \overline{\alpha_i} \cdot \beta_{i-1} \qquad (式3-29)$$

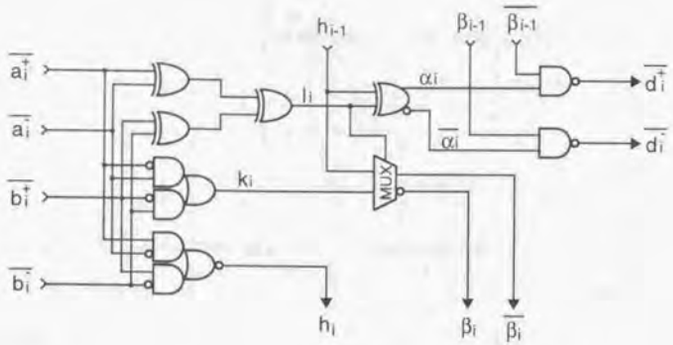
ただし、

$$\alpha_i = l_i \oplus h_{i-1} \qquad (式3-30)$$

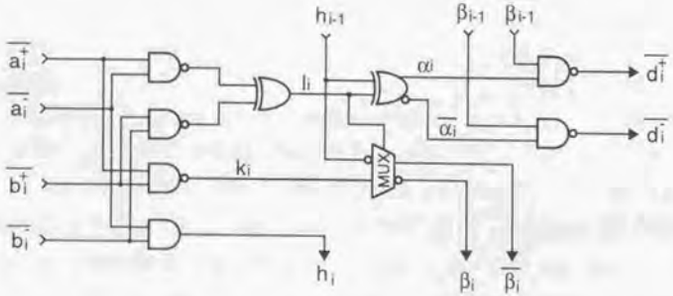
$$\beta_i = \overline{l_i} \cdot k_i + l_i \cdot h_{i-1} \qquad (式3-31)$$

さらに、h_i、l_iおよびk_iは入力信号（a_i^+, a_i^-）および（b_i^+, b_i^-）から（式３－13），（式３－25）および（式３－27）によって生成される。（式３－25）および（式３－28）～（式３－31）は、高速のCMOS回路に適した表現となっている。なぜなら、（式３－28）と（式３－29）は２入力NANDで実現することができ、また（式３－25）と（式３－30）に現われる排他的論理和回路（XOR）は、TGによるマルチプレクサ回路で実現できるからである。（式３－31）もl_iの値によってk_iとh_{i-1}のいずれか一方を選ぶ形になっているのでTGによるマルチプレクサ回路で実現することができる。図３－10(a)に（式３－13），（式３－25）および（式３－27）から（式３－31）の計７つの式に基づいて構成した冗長二進加算器の論理図を示す。この冗長二進加算器をHSRBA1（High Speed Redundant Binary Adder 1）と呼ぶ。入出力信号はいずれも反転論理となっており、これによって回路の段数を削減している。この回路は、４つのXOR/XNORゲート、２つの４入力ANDNORゲート、２つの２入力NAND、１個のMUXおよびいくつかのインバータによって構成される。ただし、この構成では依然として前節で説明した２つの問題点のうち、（式３－13）および（式３－27）を満たすために３入力以上の多入力ゲー

トを使用している点および（式３－25）を満たすために入力部にTGが用いられている点が改善されていない。



(a) HSRBA1



(b) HSRBA2

図３－10　Def.1から導かれる冗長二進加算器

この問題点は、入力の冗長二進数（a_i^+, a_i^-）および（b_i^+, b_i^-）の表現法をExp.2とすることで解決することができる。すなわち、（a_i^+, a_i^-）と（b_i^+, b_i^-）の表現法をExp.2とすると（1, 1）という値をとらなくなるので、（式３－13）、（式３－25）および（式３－27）は次のように簡略化することができる。

$$h_i = \overline{a_i^- + b_i^-} \qquad (式３－32)$$

$$l_i = (a_i^+ + a_i^-) \oplus (b_i^+ + b_i^-) \qquad (式３－33)$$

$$k_i = a_i^+ + b_i^+ \qquad (式３－34)$$

図３－10(b)に、（式３－13）、（式３－25）および（式３－27）の代わりに（式３－32）～（式３－34）を用いて簡略化された冗長二進加算器の論理図を示す。この冗長二進加算器をHSRBA2と呼ぶ。この回路は、６個の２入力NANDゲート、２個のXOR/XNORゲート、１個のMUX回路およびいくつかのインバータから構成される。HSRBA1に見られた４入力ゲートおよび入力部のTGはここでは用いられなくなり、３．２節で定めたルールに従った構成となっている。従って、従来の4-2コンプレッサに見られた問題点は解決されている。

(b) Def.2の場合

h_iが式（式３－14）で表されるDef.2の場合は、中間和（s_i^+, s_i^-）と中間桁上げ信号（c_i^+, c_i^-）に対する（式３－18）～（式３－21）はDef.1の場合と同じで、m_iとn_iが、表３－４のDef.2のものに変わるだけである。また、l_iも（式３－25）と同じである。k_iは次式のようになる。

$$k_i = \overline{\overline{a_i^+} \cdot a_i^- + \overline{b_i^+} \cdot b_i^-} \qquad (式３－35)$$

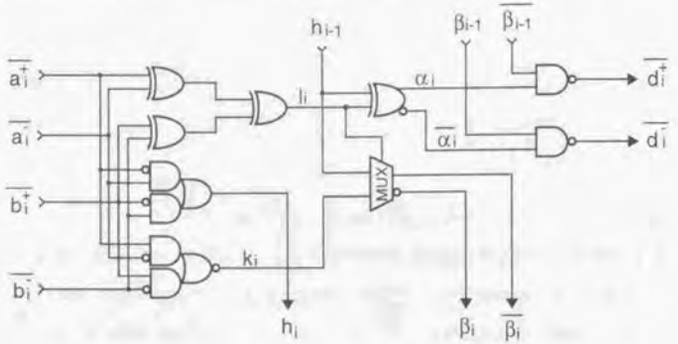
h_iは（式３－14）で表されるので、k_iとh_iに注目するとDef.1の場合に比べk_iとh_iが単に逆になっているだけであることが分かる。したがって、（式３－28）～（式３－31）において単にk_iとh_iを入れ替えるだけでDef.2におけるd_i^+, d_i^-, α_iおよびβ_iに対する式を得ることができる。図３－11(a)にこの冗長二進加算器の論理図を示す。この冗長二進加算器をHSRBA3と呼ぶ。

入力信号（a_i^+, a_i^-）および（b_i^+, b_i^-）がExp.2で表される場合は、k_iおよびh_iは以下のように簡略化される。

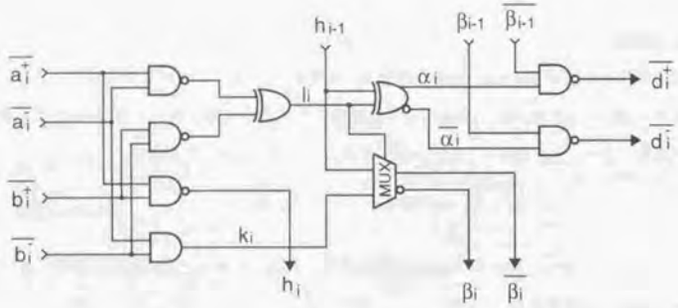
$$h_i = a_i^+ + b_i^+ \qquad (式３－36)$$

$$k_i = \overline{a_i^- + b_i^-} \qquad (式３－37)$$

l_iは、（式３－33）と同様となる。図３－11 (b)にこの簡略化された場合の冗長二進加算器の論理図を示す。この冗長二進加算器をHSRBA4と呼ぶ。これは、HSRBA2においてk_iとh_iとを入れ替えたものと等しくなっている。



(a) HSRBA3



(b) HSRBA4

図３－11　Def.2 から導かれる冗長二進加算器

(c) Def.3 の場合

Def.2 の場合と同様に、（式３－18）～（式３－21）は Def.1 と同じで、m_i と n_i が、表３－４の Def.3 のものに変わるだけである。ただし、k_i は次式のように簡単に表される。

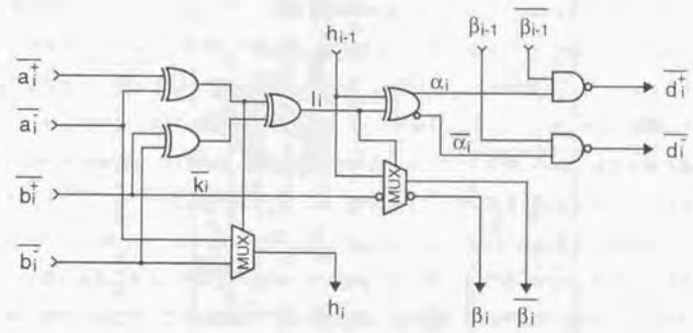
$$k_i = b_i^+ \qquad \text{（式３－38）}$$

h_i は（式３－15）で表される。Def.1 の時と同様の考え方で計算を行うことにより、l_i、d_i^+、d_i^-、α_i お

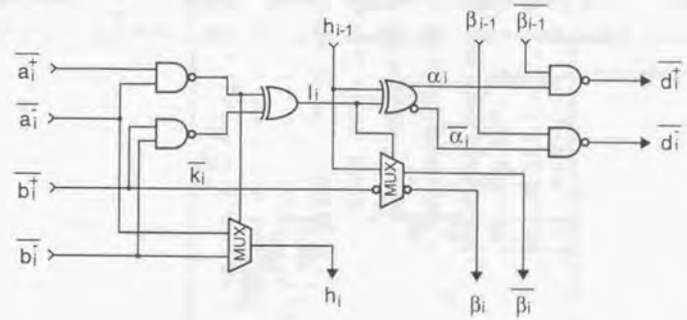
よび β_i に対して（式３－25）および（式３－28）～（式３－31）と同様の式が得られる。図３－12(a) にこの冗長二進加算器の論理図を示す。この冗長二進加算器を HSRBA5 と呼ぶ。これは４個の XOR/XNOR ゲート、２個の２入力 NAND ゲート、２個の MUX 回路およびいくつかのインバータ回路によって構成される。ここでは３入力以上の多入力ゲートは存在しないが、入出力部に TG を持つ。

入力信号（a_i^+, a_i^-）および（b_i^+, b_i^-）が Exp.2 で表される場合は、（式３－15）は以下のように簡略化される。

$$h_i = \overline{a_i^-} \cdot (a_i^+ + a_i^-) + \overline{b_i^-} \cdot \overline{(a_i^+ + a_i^-)} \qquad \text{（式３－39）}$$



(a) HSRBA5



(b) HSRBA6

図３－12　Def.3 から導かれる冗長二進加算器

l_iは、（式３－33）と同様となる。したがってこの冗長二進加算器の論理図は図３－12(b)のようになる。この冗長二進加算器をHSRBA6と呼ぶ。この回路は、入力部のマルチプレクサ回路（MUX）において入力をバッファリングしないTG回路を持つが、この部分がクリティカルパスとはなっていないので動作速度には影響を与えない。

表３－５に６つの冗長二進加算器HSRBA1～HSRBA6の関係式をまとめる。一般に乗算器においては、部分積を足し上げるために２段以上の4-2コンプレッサが直列に接続されるため、HSRAB1, HSRBA3およびHSRBA5のいずれかが初段に用いられれば、２段目以降はHSRAB2, HSRBA4およびHSRBA6のいずれかでよい。なぜなら、HSRAB1, HSRBA3およびHSRBA5は（式３－16）および（式３－17）に示すようにExp.1の入力に対しExp.2の出力を生成するため、２段目以降はExp.2の入力に対応したHSRAB2, HSRBA4およびHSRBA6が使用できるからである。もしも、初段にHSRAB2, HSRBA4およびHSRBA6のいずれかを使用する場合は、さらにその前段において入力信号をExp.1からExp.2に変換しておく必要がある。この場合、HSRBA2, HSRBA4, HSRBA6の３つは２入力NANDとTGの３種類のみからなっており、上で定めたルールを満たす高速動作に適した構成となっているため、高速の部分積足し上げが実現できる。ここで注目すべき点は、HSRBA1～HSRBA6の全てに現れる中間信号α_iとβ_{i-1}から作られるExp.1の冗長二進数（α_i, β_{i-1}）が、Exp.2の（d_i^+, d_i^-）と同じ値を持つという点である。すなわち、（式３－28）および（式３－29）により、（α_i, β_{i-1}）から（d_i^+, d_i^-）が生成されているが、これは単に(1, 1)を(0, 0)に変換しているだけなので、冗長二進数の値としては同じものである。この点に注目すると、HSRBA1, HSRBA3およびHSRBA5のいずれかが次段に来る場合、前段の冗長二進加算器の最終段の２つの２入力NANDゲートは省略することができることが分かる。なぜならば、これらの２入力NANDゲートがExp.1からExp.2への変換を行っており、次段がHSRAB1, HSRBA3およびHSRBA5のいずれかの場合はExp.1のまま出力すればよいので、これらの２入力NANDゲートは不要となるからである。

表３－５　６つの冗長二進加算器における変数の関係式

		HSRBA1	HSRBA2	HSRBA3	HSRBA4	HSRBA5	HSRBA6
Def.		Def.1		Def.2		Def.3	
RB Exp.	Input	Exp.1	Exp.2	Exp.1	Exp.2	Exp.1	Exp.2
	Output	Exp.2		Exp.2			
Equations	d_i^+	$\alpha_i \cdot \overline{\beta_{i-1}}$					
	d_i^-	$\overline{\alpha_i} \cdot \beta_{i-1}$					
	α_i	$l_i \oplus h_{i-1}$					
	β_i	$l_i \cdot k_{i+1} + l_{i+1} \cdot h_{i-1}$					
	h_i	$\overline{a_i^+ \cdot a_i^+ + b_i^+ \cdot b_i^+}$	$\overline{a_i^+ + b_i^+}$	$\overline{a_i^+ \cdot a_i^+ + b_i^+ \cdot b_i^+}$	$a_i^+ + b_i^+$	$\overline{a_i \cdot (a_i^+ \oplus a_i^-) + b_i \cdot (a_i^+ \oplus a_i^-)}$	$\overline{a_i \cdot (a_i^+ + a_i^-) + b_i \cdot (a_i^+ + a_i^-)}$
	k_i	$\overline{a_i^+ \cdot a_i^- + b_i^+ \cdot b_i^-}$	$a_i^+ + b_i^+$	$\overline{a_i^+ \cdot a_i^+ + b_i^+ \cdot b_i^-}$	$a_i^- + b_i^-$	b_i^+	b_i^+
	l_i	$a_i^+ \oplus a_i^- \oplus b_i^+ \oplus b_i^-$	$(a_i^+ + a_i^-) \oplus (b_i^+ + b_i^-)$	$a_i^+ \oplus a_i^- \oplus b_i^+ \oplus b_i^-$	$(a_i^+ + a_i^-) \oplus (b_i^+ + b_i^-)$	$a_i^+ \oplus a_i^- \oplus b_i^+ \oplus b_i^-$	$(a_i^+ + a_i^-) \oplus (b_i^+ + b_i^-)$

3．3．3．3　従来回路との比較

本節では、前節で導いた6つの冗長二進加算器HSRBA1～HSRBA6と従来の4-2コンプレッサとの関係をそれぞれのブール式を解析することにより明らかにした上で、さらにシミュレーションによる比較を行う。以下で、まず従来の4-2コンプレッサRBA1～RBA3およびNBA1～NBA3のそれぞれに対してHSRBA1～HSRBA6との論理構造の関係を明らかにする。

(a) RBA1

RBA1に関しては、図3－8(a)の論理図よりh_i，α_iおよびβ_iは以下の式で表わされる。

$$h_i = \overline{a_i^- + b_i^-} \qquad (式3-40)$$

$$\alpha_i = \overline{(a_i^+ + a_i^-)} \oplus \overline{(b_i^+ + b_i^-)} \oplus h_{i-1} = (a_i^+ + a_i^-) \oplus (b_i^+ + b_i^-) \oplus h_{i-1} \qquad (式3-41)$$

$$\beta_i = a_i^- \cdot b_i^- + \overline{(a_i^+ + a_i^-)} \cdot \overline{h_{i-1}} + \overline{(b_i^+ + b_i^-)} \cdot \overline{h_{i-1}} + \overline{(a_i^+ + a_i^-)} \cdot \overline{(b_i^+ + b_i^-)} \qquad (式3-42)$$

これらの式と（式3－30），（式3－32）および（式3－33）とを比較すると、h_iとα_iは明らかにHSRBA2のものと同じであることが分かる。また、RBA1が（式3－10）で定義されるExp.2の冗長二進数を入力とする点を考慮すると、（式3－42）は以下のように書き直すことができる。

$$\beta_i = \overline{\overline{(a_i^+ + a_i^-) \oplus (b_i^+ + b_i^-)} \cdot (a_i^- + b_i^+) + \{(a_i^+ + a_i^-) \oplus (b_i^+ + b_i^-)\} \cdot h_{i-1}} \qquad (式3-43)$$

これは、（式3－31），（式3－33）および（式3－34）より、HSRBA2における式（式3－31）と等価な式となっている。さらに、RBA1の出力信号d_i^+およびd_i^-は式（式3－28）および（式3－29）に従ってα_iとβ_{i-1}から生成される。以上より、RBA1はDef.1から導かれる冗長二進加算器HSRBA2と同じ論理構造を持つことが分かる。

(b) RBA2

RBA2に関しては、図3－8(b)よりh_iおよびα_iは明らかに（式3－40）および（式3－41）と同じである。また、β_iは次の式となる。

$$\beta_i = \overline{(a_i^+ + a_i^-) \cdot (b_i^+ + b_i^-) \cdot \overline{a_i^-} \cdot \overline{b_i^-} + \{(a_i^+ + a_i^-) \oplus (b_i^+ + b_i^-)\} \cdot h_{i-1}} \qquad (式3-44)$$

ここで、RBA2もまたExp.2を入力とすることを考慮すると、（式3－44）は（式3－43）に変換することができる。出力信号d_i^+およびd_i^-はRBA1と同様に（式3－28）および（式3－29）によってα_iとβ_{i-1}から生成される。したがって、RBA2もまたHSRBA2と同じ論理構造を持つことがわかる。

(c) RBA3

RBA3に関しては、図3－8(c)よりh_i，α_iおよびβ_iに対して次式を得る。

$$h_i = \overline{a_i^S + b_i^S}, \qquad (式3-45)$$

$$\alpha_i = a_i^a \oplus b_i^a \oplus h_{i-1}, \qquad (式3-46)$$

$$\beta_i = a_i^a \cdot b_i^a \cdot \overline{a_i^S \cdot b_i^S} + (a_i^a \oplus b_i^a) \cdot h_{i-1}. \qquad (式3-47)$$

RBA3においては、（式3－11）で与えられるExp.3の符号／絶対値表現が用いられているため、入力a_i^s，a_i^a，b_i^sおよびb_i^aは、Exp.2で表されたa_i^+，a_i^-，b_i^-およびb_i^-によって以下のように表される。

$$a_i^s = a_i^-, \quad a_i^a = a_i^+ + a_i^-, \quad b_i^s = b_i^-, \quad b_i^a = b_i^+ + b_i^- \qquad (式3-48)$$

これらを（式3－45）～（式3－47）に代入することにより、（式3－40），（式3－41）および（式3－44）を得ることができる。したがって、h_i，α_iおよびβ_iはHSRBA2と同じ信号であることが分かる。また、出力のd_i^sとd_i^aは図3－8(c)および式（式3－28），（式3－29）より以下のように表される。

$$d_i^s = \overline{\alpha_i} \cdot \beta_{i-1} = d_i^- \qquad (式3-49)$$

$$d_i^a = \alpha_i \cdot \overline{\beta_{i-1}} + \overline{\alpha_i} \cdot \beta_{i-1} = d_i^+ + d_i^- \qquad (式3-50)$$

この二つの式は、出力信号d_i^sとd_i^aが定義式（式3－11）によって（d_i^+，d_i^-）から生成されることを意味する。したがって、RBA3は冗長二進数の表現法が違うにも関わらず、HSRBA2と同じ論理構造を持つことが分かる。

(d) NBA1

NBA1に関しては、図3－9(a)の論理図よりs_i，c_i^1およびc_i^2に対して以下の式を得る。

$$s_i = a_i^1 \oplus a_i^2 \oplus a_i^3 \oplus a_i^4 \oplus c_{i-1}^2 \qquad (式3-51)$$

$$c_i^1 = (a_i^1 \oplus a_i^2 \oplus a_i^3 \oplus a_i^4) + \overline{(a_i^1 \cdot a_i^2 + a_i^3 \cdot a_i^4)} + (a_i^1 \oplus a_i^2 \oplus a_i^3 \oplus a_i^4) \cdot c_{i-1}^2 \qquad (式3-52)$$

$$c_i^2 = (a_i^1 + a_i^2) \cdot (a_i^3 + a_i^4) \qquad (式3-53)$$

ここでa_i^1，a_i^2，a_i^3およびa_i^4の代わりにa_i^+，$\overline{a_i^-}$，b_i^+および$\overline{b_i^-}$をそれぞれ代入すると、（式3－13），（式3－25），（式3－27），（式3－30）および（式3－31）より、（式3－51）～（式3－53）はHSRBA1におけるα_i，β_iおよびh_iを用いて次のように表すことができる。

$$s_i = \alpha_i, \quad c_i^1 = \overline{\beta_i}, \quad c_i^2 = h_i \quad （\alpha_i，\beta_i\text{および}h_i\text{はHSRBA1のもの}） \qquad (式3-54)$$

これらの関係式は、4 個の二進数a_i^1, a_i^2, a_i^3およびa_i^4の加算は、Exp.1 で表現された 2 つの冗長二進数（a_i^1, $\overline{a_i^2}$）と（a_i^3, $\overline{a_i^4}$）との加算に等しいことを意味している。なぜなら、3．3．1 節で説明したように 2 つの二進数の和は一方を反転、他方を非反転としてペアとすることによって 1 つの冗長二進数として表現されるからである。また、加算結果であるs_iおよびc_i^1は、（s_i, $\overline{c_{i-1}^1}$）というペアを作ることによって Exp.1 で表現された冗長二進数とみなすことができる。なぜなら、（式 3 −54）より（s_i, $\overline{c_{i-1}^1}$）は（α_i, β_{i-1}）に等しく、これは前節で述べたように Exp.1 で表現された冗長二進数とみなすことができるからである。したがって、この場合 HSRBA1 には Exp.1 を Exp.2 に変換してd_i^+とd_i^-を得るための 2 入力 NAND 回路は不要となる。以上より、NBA1 は HSRBA1 と同じ論理構造を持つことが分かる。

(e) NBA2

NBA2 においては、図 3 − 9 (b)の論理図よりs_i, c_i^1およびc_i^2に対して以下の式を得る。

$$s_i = a_i^1 \oplus a_i^2 \oplus a_i^3 \oplus a_i^4 \oplus c_{i-1}^2 \qquad (式3-55)$$

$$c_i^1 = \overline{(a_i^1 \oplus a_i^2) + (a_i^3 \oplus a_i^4)} \cdot \overline{(a_i^1 \cdot a_i^2) + (a_i^3 \cdot a_i^4)} + (a_i^1 \oplus a_i^2 \oplus a_i^3 \oplus a_i^4) \cdot c_{i-1}^2 \qquad (式3-56)$$

$$c_i^2 = a_i^1 \cdot a_i^2 + a_i^3 \cdot a_i^4 \qquad (式3-57)$$

ここでa_i^1, a_i^2, a_i^3およびa_i^4の代わりにa_i^+, $\overline{a_i^-}$, b_i^+および$\overline{b_i^-}$をそれぞれ用いると、（式 3 −14），（式 3 −25），（式 3 −30），（式 3 −31）および（式 3 −35）より、（式 3 −55）〜（式 3 −57）は HSRBA3 におけるα_i, β_iおよびh_iを用いて次のように表すことができる。

$$s_i = \alpha_i, \quad c_i^1 = \overline{\beta_i}, \quad c_i^2 = h_i \quad （\alpha_i, \beta_i および h_i は HSRBA3 のもの） \qquad (式3-58)$$

したがって、NBA2 は HSRBA3 と同じ論理構造を持つ。

(f) NBA3

NBA3 においては、図 3 − 9 (c)の論理図と図 3 −12(a)に示した HSRBA5 の論理図とを単純に比較するだけで容易に以下の関係式が成り立つことが分かる。

$$s_i = \alpha_i, \quad c_i^1 = \overline{\beta_i}, \quad c_i^2 = h_i \quad （\alpha_i, \beta_i および h_i は HSRBA5 のもの） \qquad (式3-59)$$

この場合、a_i^1, a_i^2, a_i^3およびa_i^4はそれぞれa_i^+, $\overline{a_i^-}$, b_i^+および$\overline{b_i^-}$に対応する。したがって、NBA3 は HSRBA5 と同じ論理構造を持つ。

このように、HSRBA1〜HSRBA6 と従来の主な 4-2 コンプレッサとの関係を考察することによって、従来の 3 つの冗長二進加算器 RBA1〜RBA3 は、いずれも HSRBA2 と同じ論理構造を持ち、また従来の 3 つの非冗長二進方式による 4-2 コンプレッサ NBA1〜NBA3 はそれぞれ HSRBA1, HSRBA3 および HSRBA5 と同じ論理構造を持つことが明らかになった。言い換えれば、従来の主な 4-2 コンプレッサは全て 3 種類の定義 Def.1〜Def.3 のいずれかから導くことができる。つまり、これまで個々に研究・開発されてきた 4-2 コンプレッサ回路はそれぞれ一見別のもののように思われるが、実はいずれも同様の論理構造を持った回路を異なる CMOS 回路で実現していただけである、ということができる。

HSRBA1〜 HSRBA6 を従来の 4-2 コンプレッサと比較するために SPICE2 によるシミュレーションを行った。用いたパラメータは 0.5μm CMOS のもので電源電圧は 3.3V である。回路の負荷条件としては、Wallace-tree を適用する場合の平均的な条件として、ファンアウト 1 と 0.5mm の配線容量とを仮定した。シミュレーションに当たっては、この負荷条件に対してスピードが最も高速になるようにそれぞれの回路のトランジスタサイズを最適化した。表 3 − 6 (a)に従来の 4-2 コンプレッサ RBA1〜RBA3 および NBA1〜NBA3 の遅延時間を示す。従来の回路の中では NBA3 が 1.04ns と最も高速である。これは、NBA3 が 3 入力以上の多入力ゲートを持たず、またクリティカルパスの回路段数が最も小さいためである。表 3 − 6 (b)に HSRBA1〜HSRBA6 の遅延時間を示す。このうち、HSRBA1, HSRBA3 および HSRBA5 に対しては、入力からノードα_iおよびβ_iまでの遅延時間を記入している。すなわち、前章で説明したようにα_iおよびβ_iはそのまま次段に入力することができるため、最終段でd_i^+およびd_i^+に変換する必要はなく、したがって変換のための 2 入力 NAND 回路の遅延を除外している。HSRBA1, HSRBA3 および HSRBA5 はいずれも遅延時間 1.04ns と同じ値となっている。これは、クリティカルパスが全て NBA3 と同じ経路になるためである。これに対し、 HSRBA2, HSRBA4 および HSRBA6 はいずれも入力から出力のd_i^+およびd_i^+までの遅延時間が示されている。これら 3 つはいずれも遅延時間が 0.89ns と NBA3 や HSRBA1, HSRBA3 および HSRBA5 よりも 150ps 速く、全ての 4-2 コンプレッサの中で最も高速である。これは、 HSRBA2, HSRBA4 および HSRBA6 が、3 入力以上の多入力ゲートを一切持たず、入力段や出力段に TG を用いないためである。したがって、HSRBA2, HSRBA4 および HSRBA6 が高速乗算器に適用する 4-2 コンプレッサとして最も適していると言うことができる。

ここで注意しなければならないのは、 HSRBA2, HSRBA4 および HSRBA6 は（式 3 −10）にしたがって加算前に Exp.1 から Exp.2 への変換を行わなければならない点である。しかし、この変換は 1 段の 2 入力 NAND 回路で実現できるため、これによる遅延時間の増加は小さい。実際、この場合の 2 入力 NAND の遅延は 150ps 程度であり、 HSRBA2, HSRBA4 および HSRBA6 の高速性によってこの遅延時間は隠蔽されてしまう。さらに、他の 4-2 コンプレッサにおいても初段の TG を高速に駆動するためにバッファ回路が必要となるため同程度のオーバーヘッドが生じる。したがって、このオーバーヘッド

は、 HSRBA2, HSRBA4 および HSRBA6 にとって不利な条件とはならない。

表３－６　シミュレーションによる遅延時間

(a) 従来の4-2コンプレッサ

Type	RBA1	RBA2	RBA3	NBA1	NBA2	NBA3
Delay Time (ns)	1.26	1.36	1.11	1.20	1.18	1.04

(b) 本研究による4-2コンプレッサ

Type	HSRBA1	HSRBA2	HSRBA3	HSRBA4	HSRBA5	HSRBA6
Delay Time (ns)	1.04	0.89	1.04	0.89	1.04	0.89

３．３．４　冗長二進数→二進数変換器の高速化と小面積化の研究[62], [65], [67]

Wallace-tree 部においては上で述べた冗長二進加算器のアレイによって最終の１個の冗長二進数（F^+, F^-）になるまで足し上げられるが、最終的に二進数の答えを得るためには冗長二進数から通常の二進数Ｚへの変換が必要となる。この変換は定義（式３－１）からF^+と$-F^-$とを加えることによって実現できる。すなわち、

$$Z = F^+ - F^- = F^+ + (-F^-) = F^+ + \overline{F^-} + 1 \qquad (式3-60)$$

本論文で提案する変換器は以下の考えに基づいてクリティカルパスであるキャリーの伝搬回路を単純かつ高速な回路で実現するものである。最終の冗長二進数（F^+, F^-）のi桁目を（f_i^+, f_i^-）、下位桁からのキャリー入力をc_{i-1}とすると、キャリー出力c_iは（式３－60）より以下の式で表わされる。

$$c_i = f_i^+ \cdot \overline{f_i^-} + f_i^+ \cdot c_{i-1} + \overline{f_i^-} \cdot c_{i-1} \qquad (式3-61)$$

$$c_0 = 1 \qquad (式3-62)$$

ここで、（式３－16）および（式３－17）より（f_i^+, f_i^-）は（0, 1）, (0, 0) および (1, 0) のうちのいずれかの値しか取らないことを用いると（式３－61）は次のように書き直すことができる。

$$c_i = f_i^+ \cdot \overline{c_{i-1}} + \overline{f_i^-} \cdot c_{i-1} \qquad (式3-63)$$

この式は、キャリー出力c_iがf_i^+と$\overline{f_i^-}$の一方をキャリー入力c_{i-1}で選択することによって生成されるこ

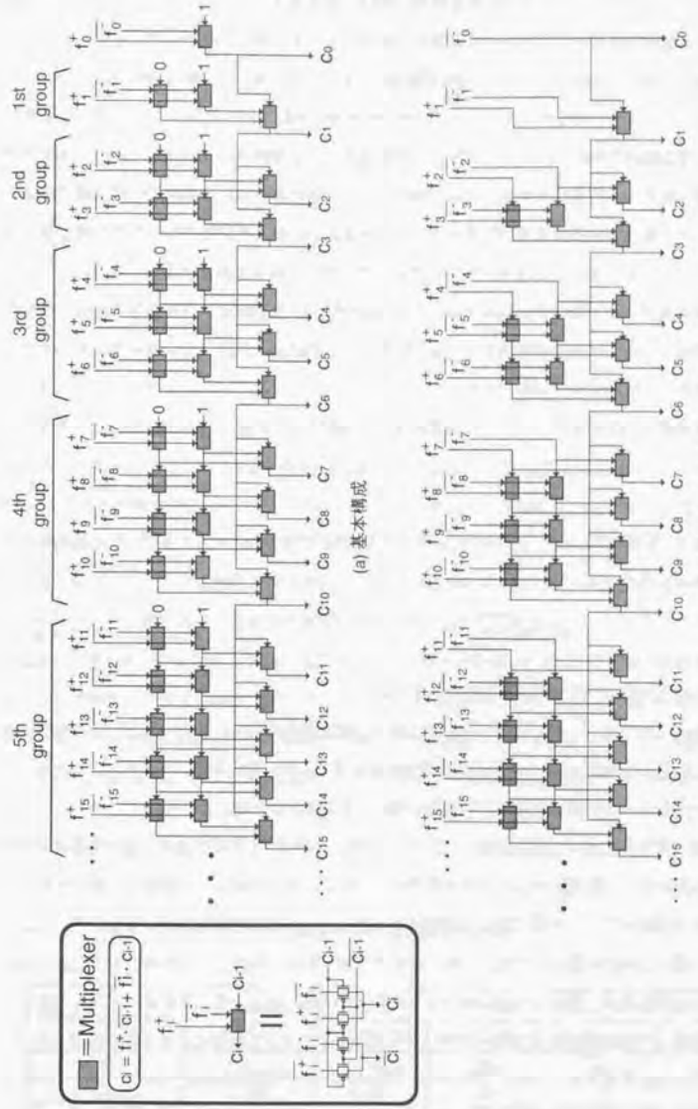


図３－13　冗長二進→二進変換器のキャリー伝搬回路

とを示しており、したがってマルチプレクサ回路（MUX）で実現できる。図３－13(a)にこの考えに基づくキャリーの伝搬回路を示す。図中の各四角はマルチプレクサ回路を表わす。最下位桁へのキャリー入力は（式３－62）より常に"1"である。第２桁目以上は、下位桁側から１桁、２桁、３桁、… という具合にグループ化され、それぞれのグループにおいてキャリー入力が"1"の場合と"0"の場合の２通りのキャリー伝搬回路が作られている。各グループで生成された２種類のキャリー信号は、１つ下位のグループの最上位桁で生成されたキャリー出力信号によって選択される。この方式は第２章で述べた一種のキャリーセレクト方式であるが、各グループに含まれる桁数を下位側から１つずつ増加させることにより、クリティカルパスの段数を全てのグループに対して揃えて高速化している。このキャリー伝搬回路は、各グループの初段の回路への入力が「0」と「1」と予め決まっていることに注目すると、さらに図３－13(b)のように簡略化することができる。これによって、クリティカルパスのマルチプレクサ回路を１段減少させている。

最終的な積の値Ｚは各桁のキャリー出力信号を用いて次式により求めることができる。

$$z_i = f_i^+ \oplus \overline{f_i^-} \oplus c_{i-1} \qquad \text{（式３－64）}$$

ここでz_iはＺのｉ桁目の値である。この式はf_i^+とf_i^-が同時に"1"にならないことを用いると次式のように簡略化することができる。

$$z_i = \overline{(f_i^+ + f_i^-)} \oplus c_{i-1} \qquad \text{（式３－65）}$$

これは、２入力NANDとTGで実現することができる。

図３－13(b)および（式３－65）より、本冗長二進→二進変換回路は、インバータ、２入力NANDおよびTGの３種類から構成でき、しかもTGの直列接続を最大２段に抑えることができる。したがって３．２節で定めたルールに従った高速な変換器を構成することができる。

本節で開発した冗長二進→二進変換器（ここではCONV1と呼ぶ）を従来の冗長二進乗算器および非冗長二進乗算器のものと比較した。正当な比較を行うために、従来回路として高速性に優れたものを選定し、それを異なったビット幅に対して最適化することにより遅延時間を最小となるようにした。従来の冗長二進→二進変換器としては、CONV2[59]とCONV3[60]を選定した。CONV2は、冗長二進→二進変換用に改良された一種のキャリールックアヘッド加算器（CLA）で、第２章で説明した生成信号Ｇと伝搬信号Ｐの生成回路を単なるインバータとすることによって高速化と素子数の削減を行っている。キャリールックアヘッド部には下で述べるCLA2のタイプの回路を用いている。CONV3は冗長二進→二進変換専用に作られた回路で、トランスミッションゲート（TG）の直列接続によるキャリー信号の伝搬経路を持つ。TGの直列が多段になると速度が劣化するため、３．２節で定めたルールに従ってTG２段毎にインバータによるバッファを挿入した。次に、従来の非冗長二進乗算器において冗長

二進→二進変換器に相当するのは最終加算部における加算器で、この回路としてはCLA1[23]およびCLA2[27]を選定した。CLA1は、キャリールックアヘッド方式とキャリーセレクト方式とを組み合わせたものでスタティックな電流を流すことによって高速化を図っている。CLA2はバイナリルックアヘッド方式とキャリーセレクト方式とを組み合わせたものでパストランジスタロジックの採用により高速化を図っている。CLA1とCLA2はともに実際に乗算器に適用され、高速動作を実現している。

以上４種類の従来回路CONV2, CONV3, CLA1およびCLA2と本研究により得られた冗長二進→二進変換器CONV1とを、それぞれビット幅8, 16, 32および64に対して最適化し、トランジスタ数とシミュレーションによる遅延時間を求めた。シミュレーションに用いたパラメータはゲート長0.5μmのCMOSプロセスのもので電源電圧が3.3Vである。表３－７および表３－８にそれぞれ遅延時間およびトランジスタ数の比較を示す。CLA1とCLA2に注目すると、この二つはスピードおよびトランジスタ数ともにほとんど同じであるが、64ビットの際にわずかにCLA2の方が優れている。CONV2は、CLA2よりもさらにスピードおよびトランジスタ数の両面で優れている。これは、CONV2がCLA2をさらに単純化した構成を持つためである。CONV3は他のものに比べて最も遅い。これはキャリー伝搬回路のTGにバッファ回路を挿入したためにクリティカルパスの段数が増加してしまったためである。筆者らが本章で開発した冗長二進→二進変換器CONV1は、16ビット以上のビット幅では、これらの中で最も高速でかつトランジスタ数が少なく優れていることが分かる。これは、キャリー信号の伝搬回路をマルチプレクサ回路のみによる極めて単純な回路で実現したためである。

高速かつ素子数が少ないこと以外にCONV1にはもう一つの長所がある。それは、構造が規則的でレイアウトが容易という点である。キャリールックアヘッド方式などではトリー状のキャリー伝搬が必要となり、このために回路の配置や配線が複雑になって設計コストが増大するという問題があるが、CONV1では、図３－13で示したようにトリー状の構成は不要であり、設計コストを低減することができる。

表３－７　遅延時間の比較

Word Length (W)	Our Converter CONV1	Conventional Converter		Conventional CLA adder	
		CONV2 [59]	CONV3 [60]	CLA1 [23]	CLA2 [27]
8	1.21ns	1.18ns	1.90 ns	1.06 ns	1.27ns
16	1.56	1.63	2.08	1.72	1.72
32	1.90	1.95	2.80	2.22	2.04
64	2.42	2.58	2.94	2.70	2.67

表３－８　トランジスタ数の比較

Word Length (W)	Our Converter CONV1	Conventional Converter		Conventional CLA adder	
		CONV2	CONV3	CLA1	CLA2
8	264	228	262	274	276
16	536	564	604	548	660
32	1048	1268	1312	1424	1460
64	1840	2676	2936	3192	3060

３．４　54×54 ビット乗算器への適用

３．４．１　回路構成

前節で説明したアーキテクチャにより、54×54 ビット整数乗算器の設計を行った。図３－14 に全体構成を示す。全体は冗長二進部分積（RBPP）発生部、Wallace-tree 部および冗長二進→二進変換部の３つの部分に分かれる。

冗長二進部分積発生部においては２次の Booth アルゴリズム[1]により部分積数を半減し、その結果 27 個の通常の二進数による部分積を発生する。このために 27 個の Booth エンコーダ／セレクタ回路 BE1～BE27 があり、それぞれ部分積 PP1～PP27 を発生する。これらのうち偶数番目の Booth エンコーダ／セレクタ回路（BE2, BE4, BE6, …）から生成される部分積 PP2, PP4, PP6, … は符号が反転される。このように生成された部分積は、隣り合う奇数番目と偶数番目をペア（PP1, PP2），（PP3, PP4），…，（PP25, PP26）とすることによって、冗長二進部分積 RBPP1～RBPP13 とみなすことができる。RBPP14 のみは PP27 単独から生成される。また、RBPP15 は偶数番目の部分積の符号反転の際に最下位桁に現れる付加ビット AB1～AB14 から生成される。このようにして合計 15 個の冗長二進部分積が生成される。

Wallace-tree 部においては、15 個の冗長二進部分積が高速冗長二進加算器（RBA）のアレイによって並列に足し上げられる。本乗算器においては，冗長二進加算器として前節で導いた６つの高速冗長二進加算器 HSRBA1～HSRBA6 のうち高速性と素子数の面で最も優れた HSRBA2 を採用した。図３－15 に HSRBA2 の詳細な回路図を示す。RBPP15 は最下位桁の位置が RBPP1 と同じであるため、両者の加算を初段で実行することにより、ハードウエアの増大を避けている。全ての冗長二進部分積を足し上げて１つにするのに４段の冗長二進加算器を必要とする。

冗長二進→二進変換部においては、108 ビットの冗長二進数から同じビット幅の二進数への変換が行われる。この変換は、前節で説明した変換器によって実行される。Wallace-tree 部において最終の冗長二進部分積は下位側から 4, 8, 16, 80 桁の順に決定されるため、変換もこの順で行われる。始めの 4, 8 桁の変換と次の 16 桁のキャリー生成とは、Wallace-tree 部における足し上げが行われている際に実行できるので、冗長二進→二進変換部における動作速度は最後の 80 ビットの変換速度で決まる。この 80 ビットの変換回路におけるクリティカルパスのマルチプレクサ回路の段数は 12 である。

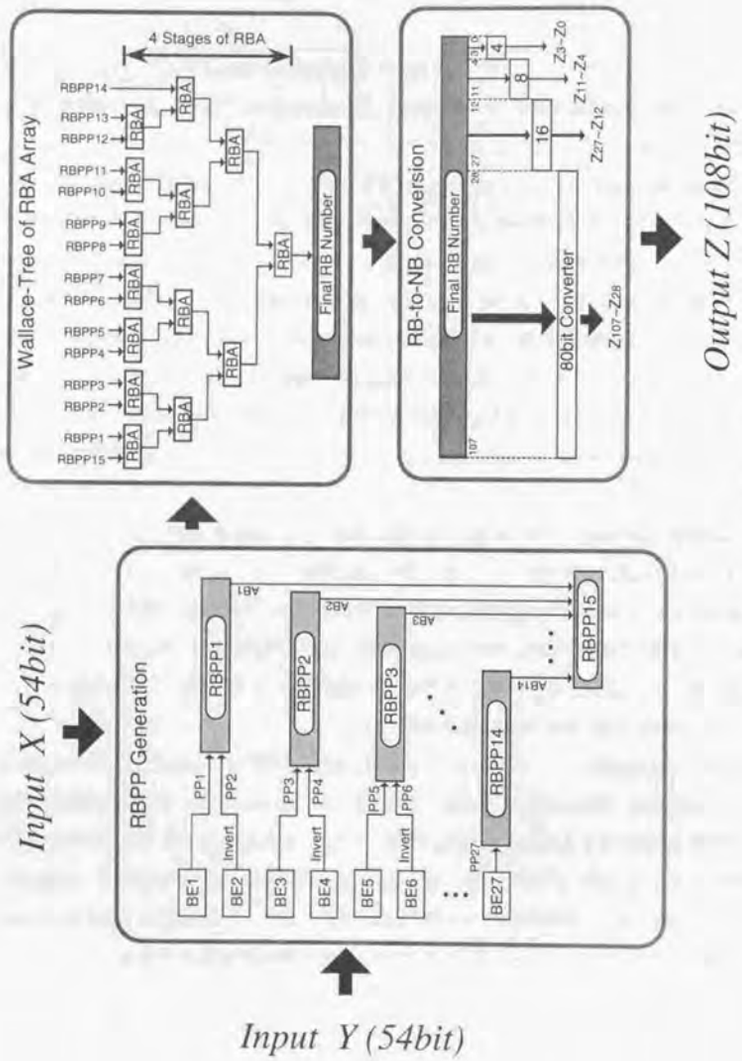


図３－14　54×54ビット乗算器の全体構成

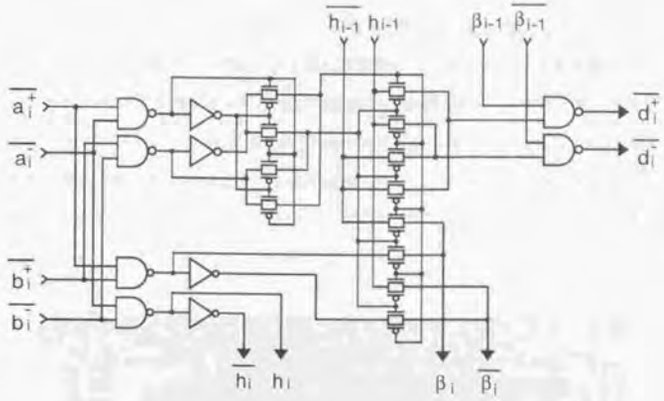


図３－15　HSRBA2の回路構成

全体の動作速度を見積もるためにSPICE2によるシミュレーションを行った。用いたパラメータは0.5μm CMOSプロセスのもので、電源電圧は3.3Vである。シミュレーションは全体回路からクリティカルパスを抽出することにより行った。結果を図３－16に示す。全体で8.8nsの乗算時間が得られた。全体の遅延時間のうち冗長二進部分積発生部の遅延時間は2.3nsで、ここには入力バッファ、Boothエンコーダおよびセレクタの遅延時間が含まれる。Wallace-tree部の遅延時間は3.8nsでこれはRBA1の4段分の遅延時間である。冗長二進→二進変換部の遅延は2.7nsでこれは80ビットの変換を行うための、マルチプレクサ12段分の遅延時間である。

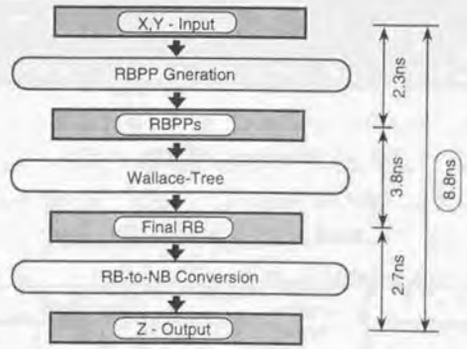


図３－15　シミュレーションによる各部の遅延時間

3．4．2　試作および評価結果

54×54 ビット乗算器を 0.5μm CMOS、3 層配線プロセスで試作した。図３－17 にチップ写真を示す。パッド領域を除くチップの有効面積は 3.05×3.08mm² でトランジスタ数は 78,800 である。トランジスタ密度は 8.4k/mm² となる。表３－９に従来の 54×54 ビット乗算器[23], [24], [27]とのトランジスタ数の比較を示す。本試作のものがトランジスタ数が最も少なく、提案した冗長二進アーキテクチャが素子数の低減に有効であることがわかる。

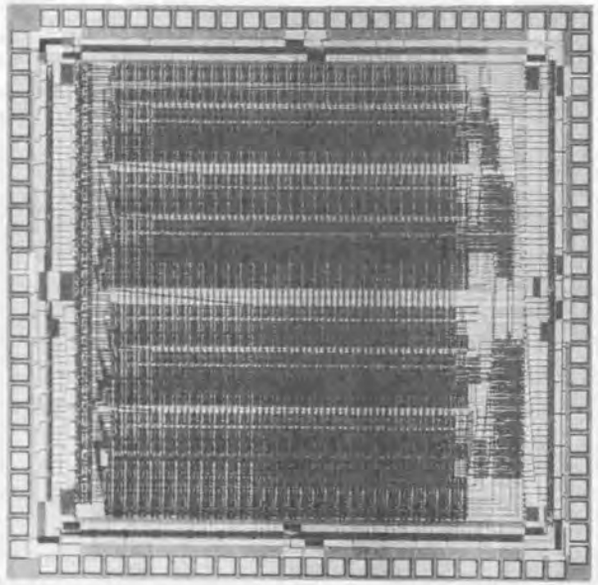


図３－17　54×54 ビット乗算器チップ写真

表３－９　トランジスタ数の比較

MultiplierType	Present Work	Conventional		
		[23]	[24]	[27]
Transistor Count	78,800	81,600	82,500	100,200

試作した 54×54 ビット乗算器の動作速度を評価した。評価は試作した乗算器にクリティカルなテストパターンを入力して出力の正／誤を調べることにより行った。図３－18 に乗算器のクリティカルなテストパターンを示す。入力を X および Y、出力を Z とした場合、まず最初に X=Y=0 を入力し、次に X=-1, Y=1 とし、再び X=Y=0 を入力する。すると出力は、最初は全ビットが「0」、次に全ビットが「1」となりさらに全ビットが「0」となる。すなわち出力が全ビット変化するため、これがクリティカルなテストパターンとなる。実際の評価では、動作の正当性も確認するためにこのクリティカルパターンを含む 10,000 パターン以上のテストパターンで評価を行った。

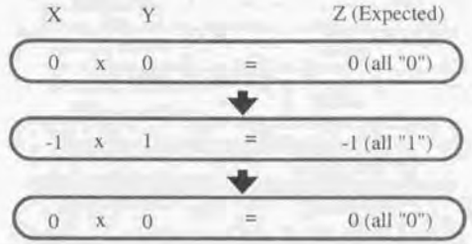


図３－18　乗算器のクリティカルなテストパターン

図３－19 に、評価結果として乗算時間と電源電圧に対するシュムープロットを示す。図中'P'で示されている領域がパスした領域である。電源電圧 3.3V で乗算時間 8.8ns の高速動作が得られている。これは、従来の 0.5μm CMOS で作られた 54×54 ビット乗算器と比較して 12%速い値である[23]。図３－20 に試作した乗算器の消費電力の動作周波数に対する依存性を示す。測定は、電源電圧（VDD）を 4.2V, 3.9V, 3.6V, 3.3V, 3.0V, 2.7V および 2.4V と７通りに変化させて、動作周波数 10MHz, 20MHz, 30MHz および 40MHz の４点に対して消費電力の測定を行った。電源電圧 3.3V の条件では動作周波数 40MHz のときに 216mW の値が得られており、これを 100MHz に換算すると 540mW に相当する。この値は、従来の 54×54 ビット乗算器[23]と比較すると 38%低い値である。これは、トランジスタ数の削減と３層配線の使用によるチップ面積の低減により全体の寄生容量が低減されたことによると考えられる。表３－10 に本 54×54 ビット乗算器の諸元を示す。

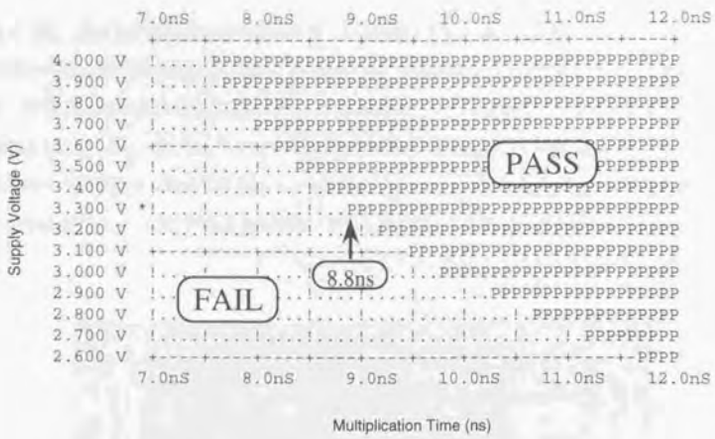


図３－19　乗算時間と電源電圧に対するシュムープロット

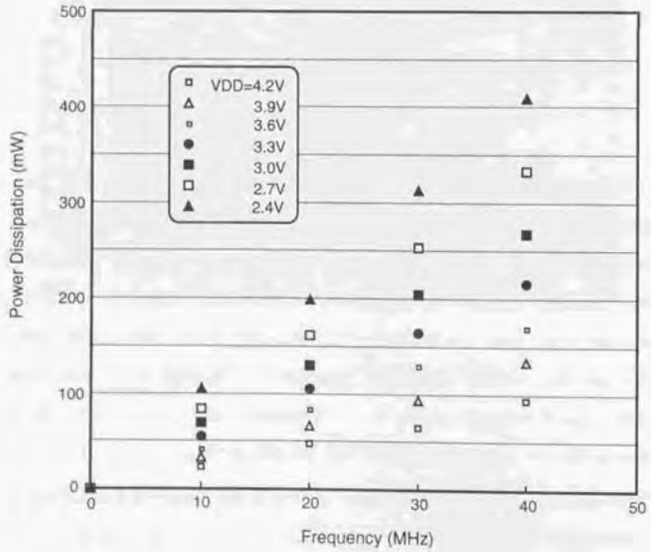


図３－20　消費電力の動作周波数に対する依存性

表３－10　54×54ビット乗算器諸元

Multiplier, Multiplicand	54b (two's complement)
Product	108b (two's complement)
Supply voltage	3.3V
Multiplication time	8.8ns
Power dissipation (estimated)	540mW (at 100MHz)
Active area size	3.05 x 3.08 mm^2
Transistor count	78,800
Density of transistors (without pad area)	8.4k/mm^2
Process	0.5-μm CMOS Triple metal

3．5　結言

本章においては、高速のデータパスを実現する上で加算器と並んで重要な構成要素である乗算器の高速化および小面積化に関する研究内容に関して述べた。

まず、高速のクリティカルパスを設計するための研究を行い、以下の結果を得た。

(1) 各種基本ゲートの遅延時間を比較することにより、インバータ、2入力NANDおよびTGの3種類のゲートが最も高速であり、クリティカルパスを構成するのに適している。すなわちパストランジスタロジックに2入力NANDを加えることでさらに高速なクリティカルパスを得ることができる。

(2) TGの直列接続の段数に関する検討を行い、3段以上の直列接続は高速化の上で不利となることを示した。したがって、TGの直列接続は2段以内とし、それ以上になる場合は間にバッファリング回路を挿入した方が高速になる。

(3) 以上の結果からクリティカルパス設計のルールとして以下の二つを定めた。

- クリティカルパスはインバータ、2入力NANDおよびTGの3種類のゲートのみで構成する。
- TGの直列接続は2段以内とする。

次に冗長二進数を用いた高速乗算アーキテクチャに関する研究を行い、以下の提案ならびに回路の最適化を行った。

(1) 二つの部分積の一方を反転してペアとすることにより冗長二進部分積を生成する手法を提案した。これによって余分なハードウエアおよび遅延時間を伴わずに冗長二進部分積が生成可能となった。

(2) これまで提案されている冗長二進加算器および非冗長二進乗算器における4-2コンプレッサに関して分析を行い、高速化を阻害する要因として以下の2点があることを明らかにした。

- 3入力以上の多入力ゲートが使用されている。
- 入出力部にTGが使用されている。

(3) 冗長二進加算器のプール式をCMOS回路に最適化して上の問題点を解決することにより、6種類の高速冗長二進加算器HSRBA1～HSRBA6を導出した。これによって従来よりも14%高速の冗長二進加算器を実現した。

(4) 従来の4-2コンプレッサの論理構造を解析することにより、従来の4-2コンプレッサがいずれもHSRBA1～HSRBA6のうちのどれかと同一の論理からなることを明らかにした。これによって冗長二進加算器と非冗長二進方式における4-2コンプレッサとの関係が明らかになり、これらを正当に評価および比較することが可能となった。

(5) 冗長二進→二進変換器に関して、パストランジスタのみで構成された高速キャリー伝搬回路を

提案した。これによって、従来の冗長二進→二進変換器およびCLA加算器よりも高速でトランジスタ数の少ない変換が実現できた。

このアーキテクチャによる54×54ビット乗算器の設計を行った。主な要点は以下とおり。

(1) 54ビットの入力から2次のブースアルゴリズムによって27個の部分積を発生させ、これらのうち偶数番目の部分積を符号反転し、それぞれを奇数番目の部分積とペアとすることによって冗長二進部分積を生成する。冗長二進部分積数は、符号処理ビットも含めて合計15となる。

(2) 冗長二進部分積を高速冗長二進加算器HSRBA2のアレイからなるワレストリーで高速に足し上げることにより、1つの冗長二進部分積まで減少させる。これに要する冗長二進加算器の段数は4段となる。

(3) パストランジスタからなる冗長二進→二進変換器によって、最終の冗長二進部分積を通常に二進数に変換する。

(4) トランジスタ数は78,800でこれは従来のものに比べて世界最少である。

設計した54×54ビット乗算器の試作および評価を行い以下の結果を得た。

(1) 0.5μmCMOS、3層配線プロセスで試作した。チップ面積は3.05×3.08mm^{2}で、0.5μmCMOSプロセスのものでは世界最小である。

(2) 電源電圧3.3Vで遅延時間8.8nsが得られた。これは、従来に比べて12%速く、世界最高速の値である。

以上のように、冗長二進乗算器のアーキテクチャをプール式に基づいてCMOS回路に最適化することにより従来に比べて高速でトランジスタ数の少ない乗算器を実現することができた。

<参考文献>

[1] A. D. Booth, "A Signed Binary Multiplication Technique," Q. J. Mech., Appl. Math. 4 pp.236-240, 1951.

[2] C. S. Wallace, "A Suggestion for Fast Multiplier," IEEE Trans. Electron. Comput., vol. EC-13, pp. 14-17, Feb. 1964.

[3] L. Y. Lee, H. L. Garvin and C. W. Slayman, "A 8 x 8b Paralell Multiplier in Submicron Technology," Dig. Tech. Papers of 1985 ISSCC, pp.84-85, Feb. 1985.

[4] T. G. Noll, D. S. -Landsiedel, H. Klar and G. Enders, "A Pipelined 330-MHz Multiplier," IEEE J. Solid-state Circuits, Vol. SC-21, No.3, pp.411-416, June 1986.

[5] M. Hatamian and G. L. Cash, "A 70-Mhz 8-bit x 8-bit Paralell Pipelined Multiplier in 2.5-μm CMOS," IEEE J. Solid-state Circuits, Vol. SC-21, No.4, pp. 505-513, Aug. 1986.

[6] Y. Oowaki, K. Numata, K. Tsuchiya, K. Tsuda, A. Nitayama and S. Watanabe, "A 7.4ns CMOS 16 x 16 Multiplier," Dig. Tech. Papers of 1987 ISSCC, pp.52-53, Feb. 1987.

[7] M. R. Santoro and M. A. Horowitz, "A Pipelined 64 x 64-bit Iterative Multiplier," IEEE J. Solid-State Circuits, vol. 24, No.2, pp.487-493, Apr. 1989.

[8] R. Sharma, A. D. Lopez, J. A. Michejda, S. J. Hillenius, J. M. Andrews and A. J. Studwell, "A 6.75-ns 16 x 16-bit Multiplier in Single-Level-Metal CMOS Technology," IEEE J. Solid-State Circuits, vol. 24, No.4, pp.922-927, Aug. 1989.

[9] R. K. Montoye, P. W. Cook and E. Hokenek, "An 18ns 56-Bit Multiply-Adder Circuit," Dig. Tech. Papers of 1990 ISSCC, pp.46-47, Feb. 1990.

[10] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi and A. Shimizu, "A 3.8-ns CMOS 16 x 16-b Multiplier Using Complementary Pass-Transistor Logic," IEEE J. Solid-state Circuits, Vol. 25, No.2, pp. 388-395, Apr. 1990.

[11] C. C. Stearns and P. H. Ang, "Yet Another Multiplier Architecture," CICC Proc., pp. 24.6.1-24.6.4, 1990.

[12] S. Roberts, W. Snyder, H. Chin, H. Hingarh, S. Leibiger, R. Lahri, L. Bouknight and M. Biswal, "A 2.5 ns ECL 16 x 16 Multiplier," CICC Proc., pp. 24.7.1-24.7.4, 1990.

[13] F. Lu and H. Samueli, "A 140-Mhz CMOS Bit-Level Pipelined Multiplier-Accumulator Using a New Dynamic Full-Adder Cell Design," Dig. of Symposium on VLSI Circuit, pp. 123-124, May 1990.

[14] O. Salomon, J. -M. Green and H. Klar, "General Algorithms for a Simplified Addition of 2's Complement Numbers," IEEE J. Solid-State Circuits, vol.30, No.7, pp. 839-844, July 1995.

[15] K. F. Pang, "Architectures for Pipelined Wallace Tree Multiplier-Accumulators," IEEE ICCD Proc., pp. 247-250, 1990.

[16] P. J. Song and G. D. Micheli, "Circuit and Architecture Trade-offs for High-Speed Multiplication," IEEE J. Solid-state Circuits, Vol. 26, No.9, pp. 1184-1198, Sept. 1991.

[17] J. F. -Ardekani, "M x N Booth Encoded Multiplier Generator Using Optimized Wallace Trees," IEEE ICCD Proc., pp. 114-117, 1992.

[18] P. E. Madrid, B. Millar and E. E. Schwartzlander Jr., "Modified Booth Algorithm for High Radix Multiplication," IEEE ICCD Proc., pp. 118-121, 1992.

[19] R. G. Burford, X. Fan and N. W. Rergmann, "An 180 MHz 16 bit Multiplier Using Asynchronous Logic Design Techniques," CICC Proc., pp. 215-218, 1994.

[20] C. Heikes, "A 4.5mm^2 Multiplier Array for a 200MFLOP Pipelined Coprocessor," Dig. Tech. Papers of 1994 ISSCC, pp.290-291, Feb. 1994.

[21] S. Hiker, N. Phan and D. Rainey, "A 3.4ns 0.8μm BiCMOS 53 x 53b Multiplier Tree," Dig. Tech. Papers of 1994 ISSCC, pp.292-293, Feb. 1994.

[22] M. Nagamatsu, S. Tanaka, J. Mori, T. Noguchi and K. Hatanaka, "A 15 ns 32 x 32-Bit CMOS Multiplier with an Improved Paralell Structure," CICC Proc., pp. 10.3.1-10.3.4, 1989.

[23] J. Mori, M. Nagamatsu, M. Hirano, S. Tanaka, M. Noda, Y. Toyoshima, K. Hashimoto, H. Hayashida and K. Maeguchi, "A 10-ns 54 x 54-b Parallel Structured Full Array Multiplier with 0.5-μm CMOS Technology," IEEE J. Solid-State Circuits, vol. 26, No.4, pp.600-605, Apr. 1991.

[24] G. Goto, T. Sato, M. Nakajima and T. Sukemura, "A 54 x 54-b Regularly Structured Tree Multiplier," IEEE J. Solid-State Circuits, vol. 27, No.9, pp. 1229-1235, Sep. 1992.

[25] Y. Kanie, Y. Kubota, S. Toyoyama, Y. Iwase and S. Tsuchimoto, "4-2 Compressor with Complementary Pass-Transistor Logic," IEICE Trans. Electron., vol. E77-C, No.4, pp.647-650, Apr. 1994.

[26] N. Okubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, Y. Nakagome, "A 4.4-ns CMOS 54x54-bit Multiplier Using Pass-transistor Multiplier," IEEE Proc. 1994 CICC, pp. 599-602, May 1994

[27] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki and Y. Nakagome, "A 4.4 ns CMOS 54 x 54-b Multiplier Using Pass-Transistor Multiplexer," IEEE J. Solid-state Circuits, Vol.30, No.3, pp. 251-257, March 1995.

[28] M. Kameyama, S. Kawahito and T. Higuchi, "A Multiplier Chip with Multiple-Valued Bidirectional Current-Mode Logic Circuits," Computer, pp.43-56, Apr. 1988.

[29] S. Kawahito, M. Kameyama and T. Higuchi, "High-Performance Multiple-Valued Radix-2 Signed-Digit Multiplier and Its Application," Dig. of Symposium on VLSI Circuit, pp. 125-126, May 1989.

[30] S. Kawahito, M. Kameyama and T. Higuchi, "Multiple-Valued Radix-2 Signed-Digit Arithmetic Circuits for High-Performance VLSI Systems," IEEE J. Solid-state Circuits, Vol. 25, No.1, pp. 125-131, Feb. 1990.

[31] S. Kawahito, M. Ishida, T. Nakamura, M. Kameyama and T. Higuchi, "High-Speed Area-Efficient Multiplier Design Using Multiple-Valued Current-Mode Circuits," IEEE Trans. on Computers, vol. 43, No.1, pp. 894-897, Jan. 1994.

[32] T. Hanyu, A. Mochizuki and M. Kameyama, "A 1.5V-Supply 200MHz Pipelined Multiplier Using Multiple-Valued Current-Mode MOS Differential Logic Circuits," Dig. Tech. Papers of 1995 ISSCC, pp.314-315, Feb. 1995.

[33] 高木、安浦、矢島、「冗長2進加算木を用いたVLSI向き高速乗算器」、電子通信学会論文誌、vol. J66-D, No.6, pp.683-690, June 1983.

[34] 高木、矢島、「算術演算のハードウェアアルゴリズム」、情報処理、vol. 26, No.6, pp.632-639, June 1985.

[35] N. Takagi, H. Yasuura and S. Yajima, "High-Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree," IEEE Trans. on Computers, vol. c-34, No.9, pp. 789-796, Sept. 1985.

[36] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa and N. Takagi, "A High-Speed Multiplier Using a Redundant Binary Adder Tree," IEEE J. Solid-State Circuits, vol. sc-22, No.1, pp. 28-34, Feb. 1987.

[37] O. Kal and T. N. Rajashekhara, "High Speed Multiplier Design Using Redundant Signed digit Number," IEEE ICCD Proc., pp. 414-417, 1987.

[38] S. Kuninobu, T. Nishiyama, H. Edamatsu, T. Taniguchi and N. Takagi, "Design Of High Speed MOS Multiplier And Divider Using Redundant Binary Representation," IEEE Proc. of the 8th Symp. on Computer Arithmetic (ARITH8), pp. 80-86, May 1987.

[39] 谷口、枝松、西山、國信、「冗長２進表現を用いた高速浮動小数点プロセッサ」、電子情報通信学会技術研究報告、ICD87-100, pp. 43-48, 1987.

[40] H. Edamatsu, T. Taniguchi, T. Nishiyama and S. Kuninobu, "A 33 MFLOPS Floting Point Processor using Redundant Binary Representation," Dig. Tech. Papers of 1988 ISSCC, pp.152-153, Feb. 1988.

[41] 谷口、枝松、西山、國信、高木、「冗長２進表現を用いた高速乗除算器」、電子情報通信学会技術研究報告、ICD88-39, pp. 1-6, 1988.

[42] M. Yamashita, J. Goto, F. Okamoto, H. Yamada, T. Horiuchi, K. Nakamura and T. Enomoto, "200Mhz 16-bit BiCMOS Signal Processor," Dig. Tech. Papers of 1989 ISSCC, pp.172-173, Feb. 1989.

[43] M. Yamashita, J. Goto, F. Okamoto, K. Ando, H. Yamada, T. Horiuchi, K. Nakamura and T. Enomoto, "A 200-Mhz 16-bit Super High-Speed Signal Processor (SSSP) LSI," IEEE J. Solid-state Circuits, Vol. 24, No.6, pp. 1668-1674, Dec. 1989.

[44] 岡本、山品、後藤、山田、榎本、「高速サブミクロンBiCMOS駆動回路の設計」、電子情報通信学会技術研究報告、SDM89-61, pp. 79-86, 1989.

[45] 外村、「冗長２進数表現による繰返し乗算方式」、情報処理学会論文誌、vol. 31, No.6, pp.861-869, June 1990.

[46] T. N. Rajashekhara and O. Kal, "Fast Multiplier Design Using Redundant Signed-digit Number," Int. J. Electronics, vol. 69, No.3, pp. 359-368, 1990.

[47] H. -Y. Lo, "High-Speed Singned Digital Multipliers for VLSI," Microprocessing and Microprogramming, No.29, pp.205-215, 1990.

[48] J. Goto, K. Ando, T. Inoue, M. Ishida, M. Yamashina, H. Yamada and T. Enomoto, "A 250Mhz 16b 1-Million Transistor BiCMOS Super-High-Speed Video Signal Processor," Dig. Tech. Papers of 1991 ISSCC, pp.254-255, Feb. 1991.

[49] S. Kuninobu, T. Nishiyama and T. Taniguchi, "High Speed MOS Multiplier and Divider Using Redundant Binary Representation and Their Implimentation in a Microprocessor," IEICE Trans. Electron., vol. E76-C, No.3, pp. 436-445, Mar. 1993.

[50] C. Y. Chow and J. E. Robertson, "Logical Disign of a Redundant Binary Adder," Proc. of the 4th Symp. on Computer Arithmetic (ARITH4), pp. 109-115, Oct. 1978.

[51] T. Nakanishi, H. Yamauchi and H. Yoshimura, "CMOS Radix-2 Signed-Digit Adder by Binary Code Representation," Trans. of the IECE of Japan, vol.E69, No.4, Apr. 1986.

[52] 原田、服部、長瀬、瀧川、「冗長二進加算セルの回路構成の検討」、電子情報通信学会全国大会、No.278, pp.2-82, 1987.

[53] M. Tonomura, "High-Speed Digital Circuit of Discrete Cosine Transform," Technical Report of IEICE Japan, SP94-41, DSP94-66, pp.39-46, 1994.

[54] 外村、「高速離散コサイン変換ディジタル回路」、電子情報通信学会技術研究報告、DSP94-66, pp.39-46, 1994.

[55] M. D. Ercegovac and T. Lang, "On-the-Fly Conversion of Redundant into Conventional Representations," IEEE Trans. on Computers, vol. C-36, No.7, pp. 894-897, July 1987.

[56] M. D. Ercegovac and T. Lang, "Radix-4 Multiplication Without Carry-Propagate Addition," IEEE ICCD Proc., pp. 654-658, 1987.

[57] T. N. Rajashekhara and A. S. Nale, "Conversion from signed-digit to radix compliment representation," Int. J. Electronics, vol. 69, No.6, pp. 717-721, 1990.

[58] M. D. Ercegovac and T. Lang, "Fast Multiplication Without Carry-Propagate Addition," IEEE Trans. on Computers, vol. 39, No.11, pp. 1385-1390, Nov. 1990.

[59] S. M. Yen, C. S. Laih, C. H. Chen and J. Y. Lee, "An Efficient Redundant-Binary Number to Binary Number Converter," IEEE J. Solid-State Circuits, vol. 27, No.1, pp. 109-112, Jan. 1992.

[60] H. R. Srinivas and K. K. Parhi, "A Fast VLSI Adder Architecture," IEEE J. Solid-State Circuits, vol. 27, No.5, pp. 761-767, May, 1992.

[61] M. Tonomura, "Simple Quotient-Digit-Selection Radix-4 Divider with Scaling Operation," IEICE Trans. Fundamentals, vol. E76-A, No.4, pp.593-602, Apr. 1993.

[62] H. Makino, Y. Nakase and H. Shinohara, "A 8.8-ns 54 x 54-bit Multiplier Using New Redundant Binary Architecture," IEEE Proc. of 1993 ICCD., pp.202-205, Oct. 1993.

[63] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase and K. Mashiko, "A 286MHz 64-bit Floating Point Multiplier with Enhanced CG Operation," Dig. of Symposium on VLSI Circuit, pp.15-16, June 1995.

[64] 牧野、鈴木、森中、中瀬、益子、角、「CGに適した機能を有する286MHz、64ビット浮動小数点乗算器」、電子情報通信学会技術研究報告、ED95-99, pp.13-20, 1995.

[65] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, H. Shinohara and K. Mashiko, T. Sumi and Y. Horiba, "A Design of High-Speed 4-2 Compressor for Fast Multiplier," IEICE Transactions on Electronics, Vol.E79-C, No.4, pp.538-548, Apr. 1996.

[66] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, K. Mashiko and T. Sumi, "A 286 MHz 64-b Floating Point Multiplier with Enhanced CG Operation," IEEE J. Solid-State Circuits, Vol.31, No.4, pp.504-513, Apr. 1996.

[67] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara and K. Mashiko, "An 8.8-ns 54×54-bit Multiplier with High Speed Redundant Binary Architecture," IEEE J. Solid-State Circuits, Vol.31, No.6, pp.773-783, Jun. 1996.

[68] A. Rothermel, B. J. Hosticka, G. Troster and J. Arndt, "Realization of Transmission-Gate Conditional-Sum (TGCS) Adders eith Low Latency Time," IEEE J. Solid-State Circuits, vol. 24, No.3, pp.558-561, June 1989.

[69] T. Sato, M. Sakate, H. Okada, T. Sukemura and G. Goto, "An 8.5-ns 112-bit Transmission Gate Adder with a Conflict-Free Bypass Circuit," Dig. of Symposium on VLSI Circuit, pp.105-106, May 1991.

[70] M.Suzuki, N.Ohkubo, T.Yamanaka, A.Shimizu and K.Sasaki, "A 1.5ns 32b CMOS ALU in Double Pass-transistor Logic," ISSCC Digest of Technical Papers, pp. 90-91, Feb. 1993.

第4章　浮動小数点加算器の高性能化に関する研究

4．1　緒言

コンピュータにおける様々な用途の中で、大規模科学技術計算や3次元コンピュータグラフィクス（CG）など高い演算精度の求められるものには、浮動小数点表示されたデータによる演算が行われる。浮動小数点データは、ビット列の中に整数部と指数部を持ち、同じビット数の整数データと比べてはるかに広範囲の数値を表現することができる[1]。このため、整数表示では計算できないような大きさの非常に異なる大小の数の間の演算が可能となり、高精度の演算結果を得ることができる。浮動小数点の表示形式については、もともとコンピュータメーカーによって種々の異なるものが用いられていたが、統一化の必要性から近年になってIEEE（The Institute of Electrical and Electronics Engineers）よりIEEE-754規格が提案され[2]、その後急速にこれを採用する動きが広まり、現在ではほぼこれに統一されている。

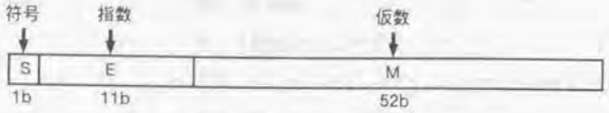
図4－1に、IEEE-754規格に基づく浮動小数点数のフォーマットを示す。規格には，大きく分けて(a)単精度と(b)倍精度があり、それぞれ図に示すように符号S（Sign）、指数部E（Exponent）および仮数部M（MantissaあるいはSignificand）の3つのフィールドを持つ。それぞれのトータルのビット幅は単精度が32ビット、倍精度が64ビットである。符号ビットは、単精度および倍精度ともに1ビットからなり、数が正の場合は「0」、負の場合は「1」の値をとる。指数部は単精度が8ビット、倍精度が11ビットで仮数部は単精度が23ビット、倍精度が52ビットとなる。IEEE-754規格では、指数部を調整することによって仮数部は必ず「1.xx …」となるように表現される。すなわち、整数部分は必ず「1」となるため、このあらかじめ決まっている「1」は仮数部には含まず、仮数部には仮数のうちの小数点以下の部分だけが示される。この手法は「けち表現」と呼ばれるもので、これによって仮数の表現できる数の領域を1ビット分広げている。また、指数部は正／負両方の値をとるが、見かけ上の値を正にするために、バイアス値Bを加えている。これは「げた履き表示」と呼ばれる手法で、Bの値は、単精度の場合が「1111111（＝127）」、倍精度の場合が「1111111111（＝1023）」となる。

以上より、IEEE-754規格による浮動小数点数Fの値は次式で計算される。

$$F=(-1)^{S}\times(1.M)\times 2^{(E-B)} \qquad (式4-1)$$

このほかに、IEEE-754規格では、各種の浮動小数点演算に関する規則や例外が生じた場合の取り扱い等が詳細に定められている。

(a) 単精度浮動小数点フォーマット



(b) 倍精度浮動小数点フォーマット

図４－１　IEEE-754 規格に基づく浮動小数点数のフォーマット

浮動小数点演算の中で最も基本となる演算は浮動小数点加算である。整数の場合と同様に浮動小数点加算ができれば基本的に他の演算はこれを元にして行うことができる。ただし、浮動小数点フォーマットが図４－１のような構造となっているために、その加算は整数の加算の場合と比べてはるかに複雑で多くの計算ステップが必要となる。図４－２に浮動小数点加算の実行手順を示す[1]。２つの浮動小数点数が入力すると、まず桁合わせのために２数の指数部の差を求め、次にその差の値に応じて、一方の仮数部をシフトすることにより指数部の値を等しくして小数点の位置を揃える。その後桁合わせされた２つの仮数部の加算を行う。ただし、２数の符号が異なる場合は減算となる。この加算は基本的に整数加算と同様に考えられるので、第２章で研究した加算器を適用することができる。加算結果によっては、上位ビットに「0」が並ぶいわゆる「桁落ち」が起きる場合があるので、次にこの「0」の数を数え、さらにその数に応じて加算結果を左方向へシフトし、整数部分を「1」にする。これと同時に指数部からこの「0」の数を引く。このように、仮数部と指数部を調整することによって演算結果を IEEE-754 規格に応じた形にすることを「正規化」という。そして、最後に加算結果に対して丸めと例外処理を行って最終的な加算結果として出力する。IEEE-754 規格では、丸めに関しては「最近値への丸め」、「０方向への丸め」、「＋∞方向への丸め」および「－∞方向への丸め」の４種類の丸めモードがサポートされており、モードの切り替えによりそれぞれの丸めを選択できるようにする必要がある。また、例外処理に関しても、加算結果が数の表現範囲外になってしまういわゆるオーバーフローやアンダーフローに関して厳格な規則が定められており、これに則った出力を生成する必要がある。

図４－２　浮動小数点加算の実行手順

このように、浮動小数点加算には多く計算ステップが要求され、極めて複雑な処理が必要となるためこれを実現する回路は大規模なものとなる。このため、1980 年代半ば頃までは、主として「スーパーコンピュータ」と呼ばれる数値計算専用の大型コンピュータ[3], [4]にのみ浮動小数点演算機能が大きなハードウエアを用いて搭載されていた。コンピュータの浮動小数点性能としては、１秒間に計算できる浮動小数点演算の回数として FLOPS（Floating Point Operation Per Second）という単位が用いられており、当時のスーパーコンピュータでは、数百メガ FLOPS 程度の演算性能が実現されていた[3], [4]。ただし、その頃はまだ半導体の微細化技術が進んでおらず、汎用のマイクロプロセッサに浮動小数点機能をハードウエアとして搭載することは不可能であった。しかし、基本的には整数加算器と論理演算器およびビット処理機能を持つ従来のいわゆる ALU（Arithmetical Logic Unit）回路があれば、特別なハードウエアを用いずにソフトウエアによって浮動小数点加算を実行することもできる。したがって当時のパーソナルコンピュータにおいては、汎用のマイクロプロセッサの ALU を用いて主としてソフトウエアによって浮動小数点加算を行っていた。しかし、そのような方法では簡単な計算にも膨大

な時間がかかるため、パーソナルコンピュータをそのような用途に用いることはあまり行われていなかった。その後1980年台の終わりから1990年代始めにかけて半導体の微細化レベルが1μm以下にまで進むと、浮動小数点演算を専用に行うVLSIプロセッサが作られるように作られるようになった[5]。これらは、いわゆる「コ・プロセッサ」と呼ばれ、パーソナルコンピュータにハードウエアとして追加することによって演算性能を向上させるというものであった。これによってパーソナルコンピュータでも3次元CGがある程度可能となったが、まだ仮想現実にはほど遠いものであった。さらに、1990年代の半ば頃になると半導体の微細化レベルが0.5μm以下になり、これによってVLSIの1チップに搭載できるトランジスタ数が飛躍的に大きくなり、浮動小数点を計算するためのデータパスを搭載した数百メガFLOPSの浮動小数点演算性能を持つ汎用マイクロプロセッサが作られるようになった[6]-[10],[26]。これによって家庭用のパーソナルコンピュータにおいても過去のスーパーコンピュータ並の処理が可能となり、パーソナルコンピュータの用途を著しく拡大した。特に3次元CGにおいては現在バーチャルリアリティ（仮想現実）を一部実現するに至っており、科学技術計算結果の可視化やエンターテインメントを中心とする幅広い分野に用いられている。また、このような用途の拡大はコンピュータの普及を急速に促進しており、大きな経済効果をもたらしている。

以上加算を中心とする浮動小数点演算の概要と歴史について述べたが、このような進歩を実現するために、浮動小数点加算器の高性能化に関して多くの研究が行われてきた。筆者らが浮動小数点加算器に関する研究を開始したのは1990年であるが、それ以前から研究中にかけても、浮動小数点加算器をシリコンLSIで実現するための研究は数多く報告されている[11]-[25]。これらのうち初期のものは、非同期式で長いサイクルタイムをかけて計算を行うものであったが[11]、その後ステップ毎に全体をいくつかに分割してレジスタを挿入し、クロック信号によってレジスタ間でデータを送っていくいわゆる「パイプライン方式」が用いられるようになっている[12]-[25]。この方式は、浮動小数点加算器のようなステップの多い演算器のスループットを向上させて演算性能を上げるのに有効である。パイプライン方式の中では、スループットを上げるためにパイプラインをビット単位で細かく分割することによってクロック周波数を上げる試みも行われている[12]。しかし、この方法はパイプラインの段数が数十段と多くなるために、演算が終了するまでのレイテンシが長くなり、演算結果を再利用したい場合に待ち時間が長くなって、計算の効率が落ちるため、限られた用途にしか用いられていない。したがって、通常は3～7段程度のパイプライン分割が行われれている。この他に、データ駆動型のアーキテクチャで処理効率を高める試みや[16]、バイポーラトランジスタを用いて高速化を図る試み[20]などが報告されている。このように、これまでいろいろな試みが行われていたが、いずれも基本的には図4－2に示したアルゴリズムに基づくものであった。筆者らが本研究を開始した1990年当初は、サブミクロン技術によって倍精度用の浮動小数点演算器全体をインプリメントした高速RISC（Reduced

Instruction Set Computer）プロセッサが発表され始めた段階であった。高速RISCプロセッサでは、DEC社のアルファチップが1992年の段階で既に200MHzの高速動作を実現しており[26]、その後も現在に至るまでマイクロプロセッサとしての最高性能を維持し続けている[27]-[29]が、その浮動小数点演算部はフルカスタム設計により莫大な人手をかけて実現された従来方式による究極の浮動小数点演算器であるということができる。しかし、3次元CG等の応用においては、現在の性能でもまだ不足であり、今後ともさらなる浮動小数点処理の高性能化が要求される。

浮動小数点加算器のさらなる高速化を図るためには、アルファチップのように多くの人手をかけた最適化を行う以外に、アーキテクチャの工夫によって計算ステップそのものを減少させて高速化を図ることが考えられる。これが実現できればあまり人手をかけずにCAD（Computer Aded Disign）ツールによる自動設計で現状並の性能が得られるばかりでなく、さらに人手をかけることで現状以上の性能を実現することができる。アーキテクチャの工夫を行うためには、浮動小数点加算器の高速化を阻害する要因に注目し、それを改善する必要がある。筆者らは、浮動小数点加算器においては桁落ちのために仮数部の加算結果の上位に並ぶ「0」のカウントやこれに基づく多ビットシフトが高速動作を阻害していることに着目し、これを改善するための研究を行った。本章では、4．2節において桁落ち予測の論理の改良による高速浮動小数点加算器のアルゴリズムを提案する。次に4．3節において、この手法を64ビット高速浮動小数点加算器に適用し、その試作および評価結果について述べる。さらに4．4節においては、この浮動小数点加算器に第2章で述べた加算器を適用した場合の性能向上について考察する。そして最後に4．5節においてまとめを行う。

4．2　桁落ち予測論理による浮動小数点処理の高速化に関する研究[30]-[32]

4．2．1　桁落ち予測（LZA）の考え方

前節で述べたように浮動小数点の加算では、異符号の２数の加算を行う際に仮数部の減算が必要となるが、その際２数の差が小さいと計算結果の上位に０が並ぶことがある[1]。これを「桁落ち」というが、この場合、仮数部は正規化のためにその分だけ左にシフトを行い、必ず先頭の１の位を「1」にしなければならない。倍精度浮動小数点加算において、もしも入力の２数が異符号でかつ非常に近い数である場合は、この先頭の「0」の数が最大 52 個になる可能性があり、これを数えてさらに仮数部の加算結果を 52 ビットシフトするためには膨大な処理時間を要する。浮動小数点加算器においては、仮数部の加算に多ビットの加算器が必要でありこれが１つのクリティカルパスとなっているが、この「桁落ち」に対する処理も仮数部の加算と並んで重大なクリティカルパスとなる。しかも従来この処理は仮数部の加算終了後にしか行うことができないため、仮数部加算と桁落ち処理という２つのクリティカルパスがシリアルに続くことになり、極めて長時間を要してしまう。従来の浮動小数点加算器では、このように加算結果から先行する０の数を求めこれに基づいてシフトを行う方式が用いられており[11]-[25]、全体の遅延時間を増大させていた。これに対し、もしもこの先行０の数が仮数加算の前にわかれば、正規化処理を仮数加算と並列に実行でき、遅延時間を大幅に低減することができる。これが桁落ちシフト量予測 LZA（Leading Zero Anticipation）の基本的な考え方である。

図４－３(a)に LZA を用いた場合の浮動小数点加減算器の仮数部処理系の基本構成図を示す。二つの入力 A および B の仮数部（Significand of input A および Significand of input B）は、加減算部（Add/Sub）において加減算されると同時に、これと並行して桁落ち予測回路（LZA）に入力し、ここで桁落ち予測信号が生成される。さらに正規化シフタ（Shifter for normalization）において、桁落ち予測信号に基づいて仮数部の加減算結果に対して正規化のためのシフトを行う。これによって、正規化された仮数部（Significand of output）が生成される。図４－３(b)に LZA を用いない従来の仮数部処理系の構成図を示す。この場合は上で説明したように、加減算部（Add/Sub）において仮数部の加減算が行われた後、加減算結果の先頭に並ぶ「0」をカウンタ（LZ counter）において数え、その結果に基づいて正規化シフタ（Shifter for normalization）によって正規化された仮数部（Significand of output）が生成される。この場合，LZA を用いた場合に比べて LZ counter の分だけ余分に遅延時間がかかる。これは、0.5μmCMOS で 3.5ns となる。したがって、LZA を用いることによって仮数部処理系の遅延時間を 3.5ns 高速化することができる。

このような背景から、これまで LZA の実現に向けての研究もわずかながら行われていたが[33], [34]、従来の手法は極めて複雑な論理から構成されており、CMOS 回路で実現する場合、後に述べるように

大面積を必要とする上に、スピード面でのメリットも少なかった。

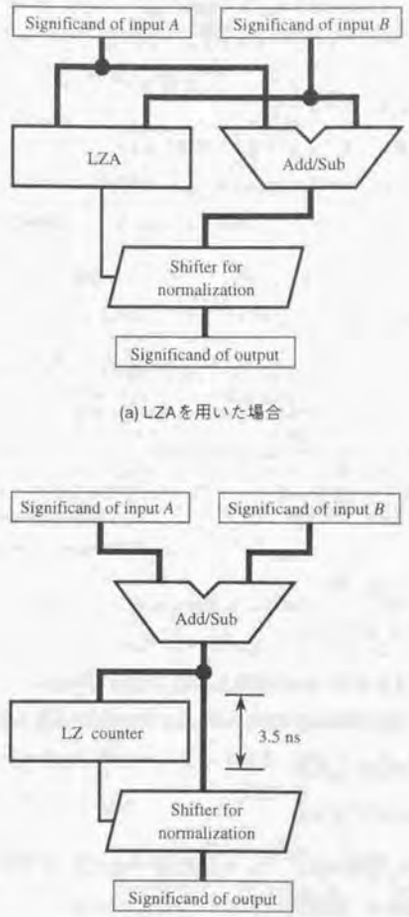


図４－３　仮数部処理系の基本構成図

4．2．2　新たなLZA論理の提案

本節では，従来のLZAにおける問題点を解決し、単純な回路構成で高速な桁落ち予測を可能とする新たなLZA論理を提案する。提案するLZA論理においては、正規化のための仮数部のシフト量を決定するために、まず入力の仮数部の値に基づいて、加算結果の先頭から最初に「1」となる桁の位置を予測する信号を生成し、次にこの信号をデコードすることによって先頭に並ぶ「0」の数をバイナリデータとして出力するという2段階の処理を行う。

LZA論理について述べる前に、まず本LZA論理の前提となるルールについて説明する。すなわち、本浮動小数点加算器においては、異符号の加算の際に2つの仮数部のうち常に大きい方から小さい方を減ずるものとする。すなわち、2つの入力の仮数部をAおよびB、仮数部の減算結果をSとすると、

$$\begin{aligned} &A \geqq B\text{のとき：} S = A - B = A + (-B) \\ &A < B\text{のとき：} S = B - A = (-A) + B \end{aligned} \qquad (式4-2)$$

であるとする。ただし、AおよびBはいずれも互いに桁合わせが行われた後の仮数部である。このようなルールを導入することによって、減算結果Sを常に正の値とすることができる。次に、減算の実行について考えると、Aあるいは Bの符号の反転は、各桁を「0」と「1」とを全て反転させて、最下位桁に「1」を加えるいわゆる2の補数変換によって行うことができるため、（式4－2）は次式のように書くことができる。

$$\begin{aligned} &A \geqq B\text{のとき：} S = A + \overline{B} + 1 \\ &A < B\text{のとき：} S = \overline{A} + B + 1 \end{aligned} \qquad (式4-3)$$

ただし、$\overline{A}$および$\overline{B}$はそれぞれAおよびBの各桁を反転したものである。

ここで，仮数部を減算する際にLZA論理への入力として次式で定義されるA′およびB′を導入する。

$$\begin{aligned} &A \geqq B\text{のとき：} A' = A,\ B' = \overline{B} \\ &A < B\text{のとき：} A' = \overline{A},\ B' = B \end{aligned} \qquad (式4-4)$$

このA′およびB′は仮数部の加算器（図4－3(a)のAdd/Sub）への入力としても用いられる。

同符号の入力の加算の場合は、仮数部同士の加算が必要となるため

$$\text{同符号の加算の場合：} A' = A,\ B' = B \qquad (式4-5)$$

とする。

筆者らの提案するLZA論理は、A′およびB′の各桁の信号A_i'およびB_i'を用いて次のブール式で定義される。

$$\begin{aligned} E_i &= \overline{A_i' \oplus B_i'} \bullet (A_{i-1}' + B_{i-1}') \\ &= (A_i' \bullet B_i' + \overline{A_i' + B_i'}) \bullet (A_{i-1}' + B_{i-1}') \end{aligned} \qquad (式4-6)$$

ここでE_iは減算結果における最上位の「1」の位置を予測する信号で，最上位の「1」のところに「1」が立ち、それよりも上位では全て「0」となる信号である。この式は、i桁目の予測信号が、i桁目の入力の排他的論理和と（i-1）桁目の入力の論理和によって非常に単純に表現できることを示している。しかも、予測信号E_iは各桁が互いに独立となるため，加算器のキャリー信号に見られるような全桁を伝搬する信号は存在しない。したがって、単純かつ高速な回路で実現することができる。

4．2．3　桁落ち予測論理の動作

本節では、（式4－6）で表される桁落ち予測論理の動作について説明する。図4－4に例として(a)～(d)の4つの場合における桁落ち予測信号E（（式4－6）で得られるE_iを各桁に持つ）と実際の加減算結果Sを示す。ただし、これらの例では2つの仮数部AおよびBはいずれも16ビットの数とし，減算の場合は第17ビット目を符号ビットとして2の補数変換によって一方を符号反転している。2数の大小に関してはA≧B，すなわち（式4－4）よりA′＝Aかつ$B' = \overline{B}$であることを仮定している。また、E_0は（式4－6）では定義されないので、図では「＿」で表している。以下、4つの各場合について、提案したLZA論理の動作を説明する。

(a) AとBが互いに近い数でA′とB′の上位側の複数ビットが等しくなる場合

この例では、A′とB′とも16ビット目が「1」で、15ビット目から9ビット目までが「0」となるので、これらが上位側の「0」の並びを発生させる。この結果，EおよびSとも10ビット目から上位が「0」となり，9ビット目で初めて「1」となる。すなわち、桁落ち予測信号Eは，正確に減算結果Sの最初の「1」の位置を予測していることが分かる。なお，Eの最下位桁E_0はこの予測に全く影響を与えないので不定のままで差し支えない。

(b) AとBが互いに近い数でAとBの上位側の複数ビットが等しくなる場合

この例では，16ビット目から7ビット目までが$A_i \oplus B_i = 1$で、6ビット目が$A_i = B_i = 1$となるのでこれらのビットによって減算結果に上位側の「0」の並びが発生する。この場合は、Eは6ビット目に初めての「1」が現れるのに対して、実際の減算結果であるSでは5ビット目に初めての「1」が現れる。すなわち、この場合はEによる予測には1ビットの誤差が含まれる。種々の入力に対して試してみると容易に分かるように，この誤差は常に予測が実際よりも1ビット上位側にずれる。したがって、この場合は、Sの先頭に並ぶ「0」の数に関して、Eは常に実際よりも1ビット少なく予測し，このため

に1ビット分の補正のためのシフト動作が必要となる。なお、(a)の場合と同様に、Eの最下位桁E_0はこの予測に全く影響を与えないので不定のままで差し支えない。

```
A :  +    1000 0000 1010 0001
B :  -    0111 1111 0110 1001
          ⇩
A' : + 0  1000 0000 1010 0001
B' : + 1  1000 0000 1001 0110 +1
-------------------------------
E :    0  0000 0001 0100 100_
S :    0  0000 0001 0011 1000
```

(a)

```
A :  +    1001 1100 1010 0001
B :  -    1001 1100 1000 1001
          ⇩
A' : + 0  1001 1100 1010 0001
B' : - 1  0110 0011 0111 0110 +1
-------------------------------
E :    0  0000 0000 0010 100_
S :    0  0000 0000 0001 1000
```

(b)

```
A :  +    1001 1100 1010 1000
B :  +    1001 1100 1010 0111
          ⇩
A' : + 0  1001 1100 1010 1000
B' : + 1  0110 0011 0101 1000 +1
-------------------------------
E :    0  0000 0000 0000 000_
S :    0  0000 0000 0000 0001
```

(c)

```
A :  +    1001 1100 0010 0001
B :  +    0011 1011 1100 1001
          ⇩
A' : + 0  1001 1100 0010 0001
B' : + 0  0011 1011 1100 1001 +0
-------------------------------
E :    1  0101 1000 0011 001_
S :    0  1101 0111 1110 1010
```

(d)

図4－4　提案したLZA論理の動作

(c) AとBの差が「1」の場合

これは、2つの入力における仮数部の差が最小となる場合で、このときは「S＝00…001」および「E＝00…00_」となる。したがって、この場合のみはEの最下位桁E_0の値を考慮しなければならない。もしも常に「E_0＝1」とすると、この例ではEが先行する「0」の数を正確に予測していることになる。しかし、「S＝00…000」となる場合すなわち入力のAとBが等しい場合にも「E_0＝1」となり、Eは先行する「0」の数を実際よりも1ビット少なく予測してしまうことになる。この状況は上で述べた(b)の場合と同様であり、(b)で述べたものと同じ補正用シフタを用いて補正することができる。これに対して、常に「E_0＝0」とすると、今度はEが実際の減算結果Sよりも先行する「0」の数を1つ多く予測することになり、これを補正するためには(b)で述べた補正用シフタとは逆方向のシフタが必要となるため、ハードウエアが増大して構成が複雑になってしまう。したがって、「E_0＝1」とした方が補正が容易なため、これを採用することとした。

(d) AとBの和を求める場合

4番目の例として、入力の2数が同符号で仮数部の加算を行う際にもEが正しく動作するかどうかを調べる。この場合はA′およびB′が（式4－5）によって与えられるため、上の3つの場合とは異なってEの17ビット目が「1」となる。このような加算の場合は、元々「桁落ち」は起こり得ないので、シフト動作は不要である。したがってEの17ビット目を「非シフト信号」として、これが「1」のときは正規化のためのシフト動作は行わせないようにすればよく、この方式を採用することとした。このようにすることで、加算の場合にもEは正しくシフト量の予測をする。

以上の4つの場合の説明で分かるように、（式4－6）で与えられるLZA論理によって仮数部の加減算結果の上位に並ぶ「0」の数を予測することができることが分かる。入力が例と異なる場合でも、上の4つの場合のどれかに当てはめることができる。またA＜Bであってもa′とB′とを入れ替えれば同様の議論が成立する。ただし、(b)の場合と二つの入力が等しい場合のみEは実際の上位に並ぶ「0」の数よりも1つ少ない値を予測するため、1ビット分の補正が必要となる。このような1ビット分のシフトを実現する回路は小さくて済むので、この手法における大きな欠点とはならない。しかも、後に述べるようにこのシフタ回路は丸めのために用いられるシフタ回路と共通化することができるため、この場合全くハードウエアを増加させない。このように、提案したLZA方式は、1ビットのずれをうまく補正することにより、単純な構成で浮動小数点加算器の高速化を実現することができる。

4．2．4　桁落ち予測回路の構成

（式4－6）で与えられるLZA論理は図4－5に示すような非常にシンプルなCMOS回路で実現できる。この回路はゲート段数が3段、1ビット当たりのトランジスタ数は16個である。（i-1）桁目のブロックにおけるNORゲートをi桁目のブロックと共有することによって素子数を削減している。上でも述べたように加算器のキャリー信号に見られるような多ビットにまたがる長い信号経路が存在しないため、遅延時間は加算器に比べて十分小さくなる。また、図4－4(c)で説明したように、E_0は1としており、$\overline{A_0 + B_0}$はE_1の生成のみに用いられている。

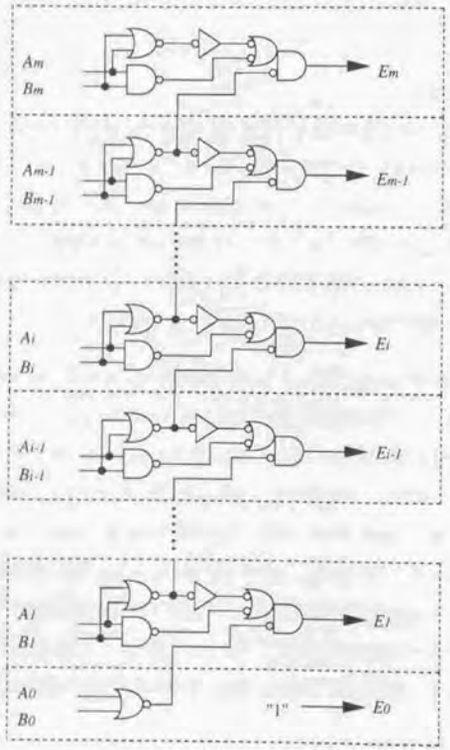


図4－5　LZA論理を実現するCMOS回路

また、本LZAを用いると出力の先行0を数える必要があるが、これを高速に行うために図4－6に示す先行「0」カウンタ（Leading-zero counter）回路を用いた。この回路は入力E_iをトリー状にエンコードして先行0の個数を二進値$D8_{i,j}$として高速に得るもので、図では8ビットに対するブロックが示されている。このブロックはさらに2つの4ビットブロックから成っている。図の左上の4ビットブロックは、入力$E_{<m:m-3>}$から二進数にエンコードされた先行「0」の数$D4_{m,<1:0>}$と入力の4ビットのOR出力である$OR4_m$を生成する。左下の4ビットブロックでは4ビットの入力$E_{<m-4:m-7>}$から$D4_{m-4,<1:0>}$と$OR4_{m-4}$を生成する。灰色の線で囲んだ右側のブロックはコンプレッサ回路で$D4_{m,<1:0>}$、$OR4_m$、$D4_{m-4,<1:0>}$および$OR4_{m-4}$から、さらに上位のエンコードされた先行「0」の数$D8_{m,<2:0>}$および入力8ビットのOR出力である$OR8_m$を生成する。これらのうち$D8_{m,2}$は$OR4_m$から直接生成され、$D8_{m,<1:0>}$は$D4_{m,<1:0>}$と$D4_{m-4,<1:0>}$を$OR4_m$によって選択することによって生成される。このようにして、この8ビットのビット列に対する先行「0」の数を$D8_{m,<2:0>}$および$OR8_m$によって二進数の形で得ることができる。この8ビットブロックを同様の方法で組み合わせることによりさらに大きな入力をエンコードすることができる。

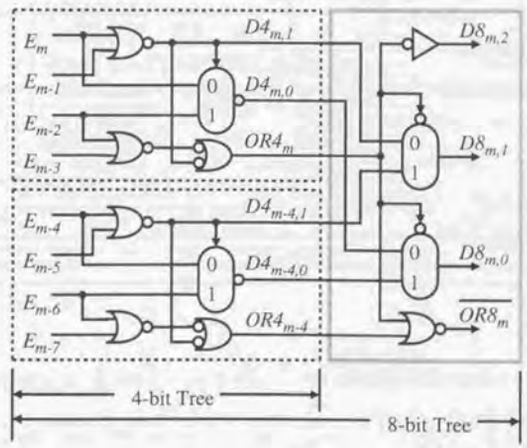


図4－6　LZカウンタ回路

4．3　64ビット浮動小数点加算器への適用

4．3．1　回路構成

提案したLZA方式を用いた64ビット浮動小数点加算器の設計を行った。表4－1に、本64ビット浮動小数点加算器の機能の一覧を示す。入出力のフォーマットとしてはIEEE-754準拠の単精度および倍精度浮動小数点フォーマットの両方が扱えるようにした。算術演算としては単精度及び倍精度の加算、減算および大小比較の機能をサポートした。また、フォーマット変換機能として単精度から倍精度および倍精度から単精度への相互の変換を可能とした。さらに、整数との変換機能として単精度から整数への変換と、倍精度から整数および整数から倍精度への相互の変換機能をサポートした。丸め機能としては、IEEE-754規格で定められた4種類の丸めである「最近値への丸め」、「0方向への丸め」、「＋∞方向への丸め」および「－∞方向への丸め」のすべてをインプリメントした。

表4－1　64ビット浮動小数点加算器の機能の一覧

フォーマット	IEEE-754規格準拠の単精度及び倍精度 浮動小数点フォーマット
算術演算	加算、減算、比較 （単精度、倍精度の両方をサポート）
フォーマット変換	単精度 ⇄ 倍精度 単精度 → 整数、　倍精度 ⇄ 整数
丸め	最近値への丸め、0方向への丸め ＋∞方向への丸め、－∞方向への丸め

図4－7に64ビット浮動小数点加算器のブロック図を示す。サイクルタイムを低減して動作周波数を上げるために、全体を5段のパイプラインに分割した。また、提案したLZA論理を含む正規化のための回路はパイプラインの3段目と4段目にインプリメントした。

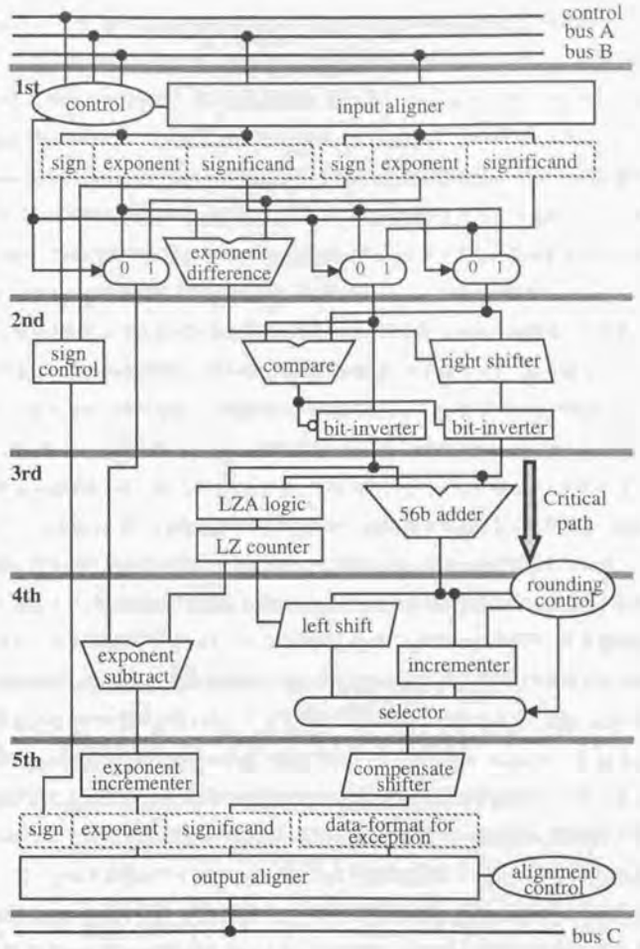


図4－7　浮動小数点加算器の構成

パイプラインの第1ステージにおいては、2つの入力オペランドに対し、与えられた命令に従って後の処理に適するようにデータの並べ替えが行われる。制御部（control）では、フォーマットの種類、算術演算やフォーマット変換などのオペレーションの種類、および丸めモードなどを指定する信号が外部から与えられ、それに応じて浮動小数点加算器全体を制御する信号を生成する。ここでは、さら

に入力オペランドの大小比較も行い、どちらが大きいかを示す信号を生成する。この比較は、入力オペランドの符号ビット以外の部分を整数とみなして比較することにより、入力フォーマットに無関係に高速に行うことができる。「input aligner」では、制御部から入力する単精度・倍精度を示す信号に従って、2つの入力オペランドを「符号（sign）」、「指数部（exponent）」および「仮数部（significand）」の3つに分割し、それぞれの回路系に送出する。2つの指数部は、減算器（exponent difference）によって差が計算されると同時に、先に生成したオペランドの大小比較結果の信号によってマルチプレクサにおいて大きい方の指数値が選択される。この指数値は第4ステージの指数減算器（exponent subtract）に送られる。また、2つの仮数部は、同じくオペランドの大小比較結果の信号によって2つのマルチプレクサにおいて、位置の入れ替えが行われる。すなわち、この2つのマルチプレクサは2個1組でスワッパーとして動作し、その結果小さい指数部を持つオペランドの仮数部が右側に出力される。

パイプラインの第2ステージにおいては、符号ビットの処理と（式4－4）に従った加減算処理の準備が行われる。まず、2つの入力オペランドの符号ビットから符号コントロール回路（sign control）によって出力の符号ビットが生成され第5ステージに送られる。また、小さい指数部を持つオペランドの仮数部が、第1ステージで求めた指数部の差の値に従って右方向シフタ（right shifter）によってシフトされ、これによって仮数部の桁合わせが完了する。次に、二つの仮数部を比較器（compare）で比較して小さい方を反転器（bit-inverter）において反転する。これによって（式4－4）における符号の反転が行われる。

第3ステージにおいては仮数部の加算とLZAの処理が並列に行われる。仮数部の加算は56ビットの整数加算器（56b adder）によって行われる。この加算器は，7つの8ビットブロックから成り、それぞれのブロックは8ビットのキャリールックアヘッド回路（CLA）で構成される。さらに，各ブロックから生成されたキャリー信号はリップルキャリー方式によって上位に伝えられる。この方式は、第2章の2．2．1節で述べた加算器アーキテクチャのうちの(d)の方式に相当するが、本章の研究は第2章の研究とは個別に行ったため，第2章で行ったような最適化はここでは行っていない。この最適化を行った場合については4．4節において考察する。LZA処理部は、前節で述べたLZA論理部（LZA logic）と先行「0」カウンタ（LZ counter）で構成される。LZA論理の出力は上位側に1ビット拡張され、4．2．3節の(d)で述べた加算の場合の「非シフト信号」を発生させている。

第4ステージでは、まず第3ステージの先行「0」カウンタの出力に従って指数減算器（exponent subtract）において指数部の値が補正される。これと同時に、左方向シフタ（left shift）によって先行「0」の値の分だけ仮数部を左方向にシフトする。このシフト操作は入力容量を低減するために2段階に分けて行っており、1段目は8ビットを単位とする粗いバイトシフトで、2段目が8ビットレンジの細かいシフトとしている。また、図には示されていないが左方向シフタの入り口において先行「0」カウンタの出力は、（shift decoder）によってデコードされている。また，丸めによる切り上げの場合のためにインクリメンタ（incrementer）によって仮数部の有効最下位桁に「1」が加えられたものも生成される。第3ステージで生成された「非シフト信号」は、非シフトの際には左方向シフトを行わないでそのまま仮数部を出力させる。丸め制御回路（rounding control）は、丸めモードにしたがって、選択回路（selector）を制御し、切り捨ての場合は左方向シフタの出力を選択し、切り上げの際にはインクリメンタ回路の出力を選択する。

第5ステージでは、まず補正シフタ（compensation shifter）によって仮数部の補正が行われる。この補正は、以下の2つの場合に対して行われる。第1の場合は、4．2．3節で述べたようにLZAによる予測が1ビット分の誤差を含む場合である。この誤差は常に先行「0」を1個少なく予測するというものであるため、第4ステージから送られてくる仮数部の最上位桁が「0」の場合に1ビット左方向へシフトする必要がある。第2の場合は、丸めの結果、仮数部の最上位に桁あふれが生じる場合である。この場合は第4ステージから送られてくる仮数部の最上位桁に桁あふれがあれば，右方向に1ビットのシフトを行う必要がある。第5ステージにおける補正シフタは、これら2つの場合の両方に対応するように作られている。第2の場合に対する補正シフトはLZAとは無関係に必要なので、LZAの予測誤差のための補正シフタは本64ビット浮動小数点加算器においては、ハードウエア上のオーバーヘッドとはなっていない。指数部補正回路（exponent incrementer）では、仮数部の補正シフトに従って，その分の指数部に対する補正が行われる。以上の処理によって、演算結果の符号、指数部および仮数部が生成されたので、最後に（output aligner）によって必要な出力フォーマットに変換されて答えが出力される。また、結果が浮動小数点フォーマットの表現範囲を超えてしまうオーバーフローやアンダーフローの場合に対する処理もこの第5ステージで行われる。

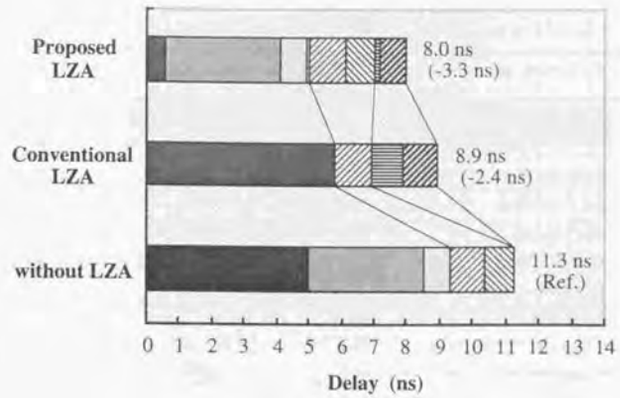
回路設計に当たっては、パイプラインの各ステージにおいてSPICE2によるシミュレーションを行った。用いたパラメータは0.5μmCMOSのもので、電源電圧は3.3Vである。最も遅延時間が長いのが第3ステージで、これが全体のサイクルタイムを決定している。第3ステージの遅延時間を決定しているのは56ビット加算器であり、これが5.0nsの遅延時間を持っている。このため、次節でも示すようにLZA論理（LZA logic）と先行「0」カウンタ（LZ counter）における遅延時間は完全に加算器の遅延時間の中に隠蔽されている。ただし、第2章の研究成果をこの56ビット加算器に適用することにより、さらに性能向上が可能であり、これに関しては4．4節において考察する。トランジスタ数の合計は約54,000である。本章で提案したLZA論理のインプリメントに当たっては極力従来の回路との共有化を図っているため、従来の浮動小数点加算器と比べると、ハードウエアの増加はLZA論理回路（LZA logic）の部分だけとなっている。LZA論理回路のトランジスタ数は884でありこれは全体の1.6%と小さい。

パターンレイアウトは 0.5μmCMOS の 3 層配線プロセスで行った。レイアウトに当たっては、回路ブロック毎にセルを CAD ツールを用いて自動生成し、これをマニュアルで配置および配線するという手法を用いた。これにより、全てを人手でレイアウトする場合に比べて期間が 1/3 以下に短縮できた。

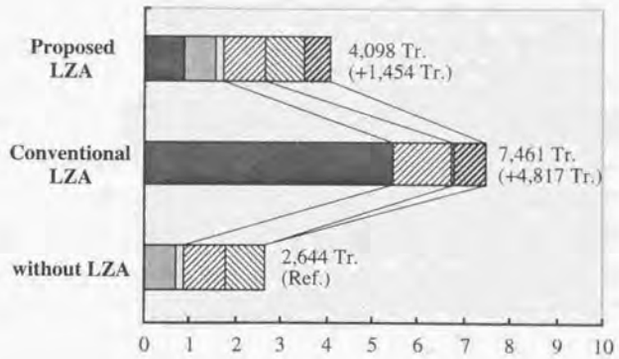
4．3．2　従来方式との比較

本節では、提案した LZA 方式と従来方式との遅延時間およびトランジスタ数の比較について述べる。4．2．1 節でも述べたように、浮動小数点加算器の正規化処理に関しては、これまで高速化のために LZA を用いたものもあったが[33], [34]、この方法は複雑で大面積を必要としたため、その他のものは面積の増加を避けて LZA を用いていなかった。そこで、筆者らが提案した新しい LZA 論理の有効性を示すために、この方式を従来の LZA 論理および LZA を用いない場合の両方と比較した。図 4－8 (a) および(b)に遅延時間およびトランジスタ数の比較をそれぞれ示す。この比較は、64 ビット浮動小数点加算器の正規化処理部を 0.5μmCMOS で実現した場合の遅延時間およびトランジスタ数を求めることによって行っている。表 4－2 に得られた値の一覧表を示す。

まず、提案した LZA 方式において正規化処理部は、LZA 論理部（LZA logic）、先行「0」カウンタ（LZ counter）、シフトデコーダ（Shift decoder）、2 段階から成る左方向シフタ（Multiple shifter (1st) および(2nd)）、予測誤差補正信号発生器（Compensate decoder）および補正シフタ（Compensate shifter）からなり、遅延時間の合計は 8.0ns、トランジスタ数の合計は 4,098 である。なお、この遅延時間には仮数部の加算の終了を待つ時間（Waiting delay）の 0.1ns が含まれている。これに対して従来の LZA 方式[33], [34]は、筆者らが提案した方式と異なり LZA 論理（LZA logic）、4 ビットを単位とするシフタ（Multiple shifter (hex)）、4 ビットを単位とする先行「0」デコーダ（4-bit LZ decoder）および 4 ビットレンジのシフタ（Multiple shifter (binary)）の 4 つの部分から構成され、遅延時間の合計は 8.9ns、トランジスタ数の合計は 7,461 となる。従来の LZA 方式は、先行「0」の数を予測するために非常に複雑な回路を必要とするため、遅延時間が長く、トランジスタ数もはるかに多くなることが分かる。また、LZA を用いない場合、正規化のための回路は先行「0」カウンタ（LZ counter）、シフトデコーダ（Shift decoder）および 2 段階から成る左方向シフタ（Multiple shifter）から構成される。これは本章で提案した LZA 方式から、LZ 論理部と補正のための回路を取り去ったものとなる。その代わりに、先行「0」を数える前に仮数部加算器（Significand adder）によって仮数部の加算が実行されなければならないために、遅延時間にはこの加算による遅延時間 5.0ns が加えられる。その結果、遅延時間の合計は 11.3ns、トランジスタ数は 2,644 となる。仮数部加算器はどの方法においても存在するのでトランジスタ数の計算からは除外している。このように、LZA 方式を用いない場合は遅延時間が大きく高速動作には不利であることが分かる。



（a）遅延時間



（b）トランジスタ数

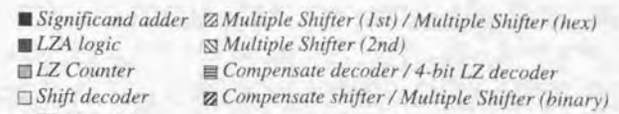


図 4－8　桁落ち処理系回路の遅延時間およびトランジスタ数の比較図

表４－２　桁落ち処理系回路の遅延時間およびトランジスタ数の比較表

	Delay (ns)	Transistor count
Proposed LZA	8.0	4,098
- *LZA logic*	0.6	884
- *LZ counter*	3.5	696
- *Shift decoder*	0.8	152
- *Waiting delay*	0.1	-
- *Multiple shifter (1st)*	1.1	928
- *Multiple shifter (2nd)*	0.9	868
- *Compensate decoder*	0.2	2
- *Compensate shifter*	0.8	568
Conventional LZA	8.9	7,461
- *LZA*	5.8	5,446
- *Multiple shifter (hex)*	1.1	1,288
- *4-bit LZ decoder*	1.0	39
- *Multiple shifter (binary)*	1.0	688
Without LZA	11.3	2,644
- *Significand adder*	5.0	-
- *LZ counter*	3.5	696
- *Shift decoder*	0.8	152
- *Multiple shifter (1st)*	1.1	928
- *Multiple shifter (2nd)*	0.9	868

以上の比較により、提案したLZA方式は従来のLZA方式に比べて高速かつトランジスタ数が少なくなることが分かる。遅延時間では、従来のLZA方式よりも0.9ns速く、LZA方式を用いない場合と比べると3.3ns高速である。また、LZA論理部（LZA logic）と先行「0」カウンタ（LZ counter）との遅延時間の合計は4.9nsで、これは仮数部加算の5.0nsよりも小さい。したがってこの２つの処理は仮数部加算処理の中に完全に隠蔽されてしまう。先行「0」のカウント後に0.1nsの待ち時間（Waiting delay）ができるのはこのためである。これに対して従来のLZA方式ではLZA論理部の遅延時間が5.8nsと仮数部加算よりも0.8ns遅くなるため、LZA論理部の遅延の方がクリティカルパスとなっている。また、

提案したLZA方式におけるシフト操作（左方向シフトおよび補正シフト）の遅延時間は3.0nsで、従来のLZA方式におけるシフト操作（Multiple shifter (hex)、4-bit LZ decoderおよびMultiple shifter (binary)）の遅延時間の合計である3.1nsよりも高速である。これは、提案したLZA方式が予測誤差による補正シフトを必要とするにも関わらず、この誤差が１ビットと小さく抑えられているために、従来のLZA方式における（4-bit LZ decoder）と同等の遅延時間で補正シフトが実現できたためである。また、提案したLZA方式における（multiple shifter (2nd)）の遅延時間が0.9nsで、これに対応する従来のLZA方式における（multiple shifter (binary)）の遅延時間1.0nsよりも速いのは、（multiple shifter (2nd)）の入力ドライバを前段の（multiple shifter (1st)）の動作時にこれと並行して動作させているためである。

トランジスタ数に関しては、提案したLZA方式は、LZAを使用しない場合と比べて1,454個増加する。これは主にLZA論理部と補正シフト部の付加に起因するものである。これに対して従来のLZA方式では、LZAを使用しない場合に比べて4,817個の増加であり、提案したLZA方式よりも３倍以上トランジスタ数が増加する。

以上のように、LZA方式の使用は高速化には有利であるがトランジスタ数が増加するために面積の点では不利である。提案したLZA方式では、遅延時間が3.3ns短縮され、トランジスタ数が1,454個増加するのに対して、従来のLZA方式では遅延時間が2.4ns短縮され、トランジスタ数が4,817個増加する。すなわち、提案したLZA方式は従来のLZA方式よりもスピードと面積の両面で優れており、LZA使用のメリットを拡大している。また、パイプライン方式の適用という観点から考えると、パイプライン方式の場合サイクルタイムは全体を構成するブロックの最も遅いもので決定されてしまうため、従来のLZA方式ではLZA論理部（LZA logic）の5.8nsでサイクルタイムが律速される。これに対して提案したLZA方式では、最も遅いのは仮数部加算器の5.0nsであり、この方式がパイプライン化において14%高速化に有利であることが分かる。しかも、仮数部加算器に第２章の成果を適用して高速化することにより、さらなる高速化も可能である。

４．３．３　試作および評価結果

設計した64ビット浮動小数点加算器を0.5μm CMOSプロセス技術を用いて試作した。図４－９にチップ写真を示す。パッドまで含めた試作チップのダイサイズは3.5mm×3.6mm、周辺のテスト回路を除いたアクティブエリアのみのサイズは2.5mm×3.5mmである。配線には３層配線を用いており、１層、２層および３層配線の幅/間隔はそれぞれ1.2/0.8μm、1.2/1.3μm、2.0/1.0μmである。１層配線と２層配線は主にそれぞれ横方向と縦方向の配線として用いられ、３層配線は横方向の電源およびGND配線とクロック配線に用いられている。これによって内部のセル内の１層あるいは２層配線による太

い電源配線を極力なくしている。

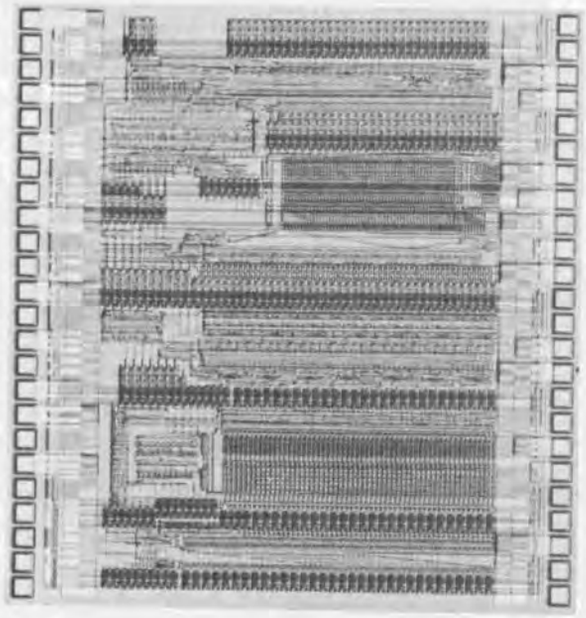


図４－９　浮動小数点加算器のチップ写真

試作した64ビット浮動小数点加算器には、テスト用回路としてスキャンパスを設けた。スキャンパスは、各パイプラインステージの入出力に設けられたパイプラインレジスタを横方向にシフトレジスタとして動作させるもので、クロックの与え方によって、パイプラインレジスタとして動作させることもスキャンパスとして動作させることもできるように設計した。64ビット浮動小数点加算器のテストでは、クロックとしてメインクロック、インプットクロックおよびアウトプットクロックの３種類のクロック信号を用いた。メインクロックはパイプラインを動かして浮動小数点加算器の本体を動作させるクロックである。インプットクロックとアウトプットクロックはそれぞれ入力と出力のシフトレジスタを動作させるためのクロックである。入力するテストパターンは、まずインプットクロック

によって入力シフトレジスタに蓄えられ、次に５サイクルのメインクロックによって浮動小数点加算器に入力して演算が実行され、出力シフトレジスタに蓄えられる。最後にアウトプットクロックによって計算結果が外部に取り出され、期待値データと比較される。動作の評価には10万パターン以上のテストパターンを用いた。これらのうち、1,000パターンは人手で作成したクルティカルパターンで、残りは自動生成したランダムパターンである。

図４－10はサイクルタイムと電源電圧に対するシュムープロットである。測定は室温で行った。電源電圧3.3Vの際のサイクルタイムは6.1nsである。これは、動作周波数164MHzに相当し、100MHzを大きく上回る動作速度が達成されていることが分かる。サイクルタイムは３段目のパイプラインステージで律速されており、上で述べたように仮数部加算器による遅延時間が5.0nsで残りの1.1nsがパイプラインレジスタによる遅延時間である。消費電力は3.24mW/MHzである。動作する電源電圧の下限は2.2Vで、これはnMOSのパストランジスタから成る左方向シフタの動作劣化によるものである。チップの諸元を表４－３にまとめる。

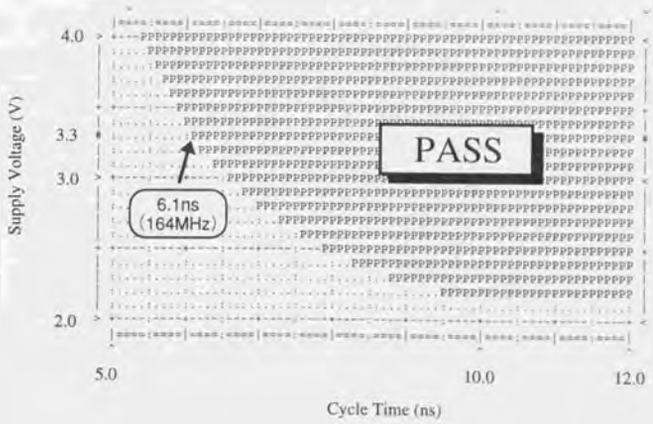


図４－10　サイクルタイムと電源電圧に対するシュムープロット

表４－３　64ビット浮動小数点加算器の諸元

Process technology	Triple metal CMOS
Design rule	0.5 μm
Transistor count	54 k
Die size	3.5 mm × 3.6 mm
Active area	2.5 mm × 3.5 mm
Clock rate	164 MHz
Power dissipation	3.24 mW/MHz at 3.3 V

４．４　課題および考察

本章で述べた研究内容は、第２章で述べた整数加算器の研究とは個別に行ったもので、本章の成果には、第２章の成果は十分には取り入れられていない。その理由は本章で設計・試作した64ビット浮動小数点加算器では、基本セルの生成を自動で行ったため、完全にマニュアルで設計された第２章の加算器とはセル幅などが異なり、そのまま使うことができなかったためである。また、第２章における加算器アーキテクチャの検討と本章の浮動小数点加算器設計とをほぼ同時に行ったため、前者の成果を後者に十分に取り入れる時間が無かったこともう一つの理由である。その結果、本章の64ビット浮動小数点加算器では仮数部の加算器として第２章の２．２．１節で述べた加算器アーキテクチャのうちの(d)の方式を十分に最適化されない形で用いており、遅延時間も5.0nsと第２章のものと比較して大きかった。このために仮数部加算器が全体の動作速度を律速してクリティカルパスとなっていた。したがって、第２章の成果を本章の64ビット浮動小数点加算器の仮数部加算器に適用することによってさらに高速動作を実現することができる。第２章の成果を実際に浮動小数点加算器に適用して更なる高速化を図ることは本研究以後の課題であるが、適用した場合の性能予測を行うことは可能である。そこで、本節ではその性能予測について述べる。

第２章ではアーキテクチャおよび回路の最適化によって加算時間が2.6nsの64ビット加算器が得られたが、これを56ビット加算器に適用した場合の加算時間について考える。これは第２章の表２－２を用いることにより容易に計算できる。すなわち、表２－２よりアーキテクチャ（(n-k)-b CLA + k-b CSA）の遅延時間$T_{n,k}$は

$$T_{n,k} = 7\log_{10}k + 0.1n + 6.7 \qquad (式４－７)$$

で与えられるので、64ビットに対する値（n=64かつk=8）と56ビットに対する値（n=56かつk=8）を代入すると、

$$T_{64,8} = 25.3 \qquad (式４－８)$$

$$T_{56,8} = 24.1 \qquad (式４－９)$$

となる。したがって、56ビット加算器の加算時間は、以下のように見積もることができる。

$$2.6\times (T_{64,8}/T_{56,8}) = 2.6\times (24.1/25.3) = 2.48\ (\mathrm{ns}) \qquad (式４－10)$$

実際の56ビット加算器の加算時間を求めるためには、さらに加算器のサイズの違いの効果を考慮する必要がある。これは第２章の表２－３を用いて計算することができる。すなわち、第２章で設計した加算器の横方向の幅は1305μmであったが、これは１ビット分に換算するとほぼ20μmである。これに対して本章で設計した64ビット浮動小数点加算器においては、仮数部加算器の１ビット分の幅は25μm

で設計されている。したがって、加算器の遅延時間のうちの配線容量の分だけが25/20＝1.25倍になる。表2－3において、アーキテクチャ（(n-k)-b CLA＋k-b CSA）のイントリンシック、配線およびゲート容量による遅延時間の式にn＝56とk＝8を代入すると、

イントリンシックな遅延　；7.67
配線遅延　；7.28　（式4－11）
ゲート容量遅延　；9.12

となる。したがって、配線遅延のみ1.25倍にすることにより、幅25μmの基本セルからなる56ビット加算器の遅延時間は（式4－10）の値から以下の式で計算できる。

$$2.48\times(7.67+7.28\times1.25+9.12)/(7.67+7.28+9.12)=2.7\ (\mathrm{ns}) \qquad (式4-12)$$

以上の計算を第2章で検討した他の4つの加算器アーキテクチャにも当てはめると、各アーキテクチャで幅25μmの基本セルからなる56ビット加算器を実現した場合の加算時間は、表4－4に示す通りとなる。すなわち、RCAは12.3nsと遅いが、CLAでも（48-b CLA＋8-b CSA）と同様の2.7nsとなり、CSKと（8-b CLA×7-b RCA）とはともに3.3nsとなる。

表4－4　各アーキテクチャによる56ビット加算器の遅延時間

アーキテクチャ	遅延時間
RCA	12.3 ns
CLA	2.7 ns
CSK	3.3 ns
8-b CLA × 7-b RCA	3.3 ns
48-b CLA + 8-b CSA	2.7 ns

ここで、64ビット浮動小数点加算器の3段目のパイプラインステージに注目すると、56ビット加算器の次に遅いパスは、図4－7から分かるように、LZA論理（LZA logic）と先行「0」カウンタ（LZ counter）からなるLZA処理回路で、この2つの遅延時間の合計は表4－2より、0.6＋3.5＝4.1nsとなる。したがって56ビット加算器をこれよりも高速にしても効果は得られない。すなわち、第2章で最適化された回路を使用すれば、RCA以外の4種類のアーキテクチャのどれを用いても同様の効果を得ることができる。これらの中で最も面積が小さくなるのは図2－4よりCSKの場合であり、本64ビット浮動小数点加算器には最適な仮数部用加算器であるということができる。

このように仮数部加算器が高速化されると3段目のパイプラインステージは、LZA処理回路の4.1nsのパスがクリティカルパスとなるので、パイプラインレジスタの遅延時間も含めて実測値の6.1nsから0.9ns高速化されて5.2nsとなる。これは、動作周波数に換算すると192MHzに相当する。ただし、この際考慮しなければならないのは、他のパイプラインステージも同様に速くなるかどうかという点である。3段目のパイプラインステージ以外で5.2nsよりも遅いステージはシミュレーションから1段目と4段目であることが分かっている。しかし、4．3．1節で説明したように第1ステージのクリティカルパスは入力オペランドの大小比較を行う比較器であり、第3ステージのクリティカルパスは丸めの結果を得るためのインクリメンタであって、いずれも加算器に基づいた回路となっている。したがってこれらの回路に対しても第2章の成果を適用することによって3段目と同様かそれ以上の効果を得ることができる。また、2段目と5段目は他のステージに比べて処理が単純なのでクリティカルパスとはならない。すなわち、第2章の成果を適用することによって、全てのパイプラインステージの遅延を5.2ns以下に高速化することができ、その結果動作周波数を192MHzまで上げることができる。

4．5　結言

本章では、高速浮動小数点プロセッサに必要不可欠な構成要素である浮動小数点加算器の高速化に関する研究内容について述べた。

まず、浮動小数点加算器におけるクリティカルパスを調べるために、浮動小数点加算の手順を分析し、その結果以下の点を明らかにした。

(1) 異符号の加算を実行する際、2 数の差が小さいと仮数部の加算結果の上位に「0」が並ぶ「桁落ち」が起き、その場合この「0」の数を数えてその分だけ加算結果を左方向にシフトする必要があり、これが重大なクリティカルパスとなっている。

(2) この桁落ち処理を仮数部の加算と並列に行うことができれば、高速化を実現することができる。

(3) 桁落ち処理と仮数部加算とを並列に行うためには、入力オペランドから仮数部加算結果の先頭に並ぶ「0」の数を直接予測する LZA（Leading Zero Anticipation）機能の搭載が必要である。

この結果に基づき、高速でハードウエア量の少ない LZA 論理に関する研究を行い、以下の結果を得た。

(1) 次式で与えられる LZA 論理を提案した。

$$E_i = \overline{A_i' \oplus B_i'} \bullet (A_{i-1}' + B_{i-1}')$$
$$= (A_i' \bullet B_i' + \overline{A_i' + B_i'}) \bullet (A_{i-1}' + B_{i-1}')$$

ただし、E_i は先頭の「1」の位置を予測する信号で、A_i' および B_i' は入力オペランドである。

(2) 種々の入力に対して動作を調べることにより、E_i が 1 ビット分の誤差を含んで先頭の「1」の位置を予測できることを示した。

(3) E_i を生成する回路として 1 ビット当たり 16 個のトランジスタで構成されるシンプルな回路を提案した。

(4) E_i に基づいて先頭の「0」の数を求める回路をトリー状の高速な回路で構成した。

提案した LZA 論理を用いた 64 ビット浮動小数点加算器の設計を行った。設計の主な要点は以下の通り。

(1) IEEE-754 準拠による単精度および倍精度の加算、減算および比較機能を搭載した。さらに、単精度と倍精度間の相互の変換機能および浮動小数点数と整数との相互の変換機能を搭載した。

(2) 動作周波数を上げて演算のスループットを高めるために、5 段のパイプライン分割とした。桁落ち処理系の回路は、パイプラインの第 3 ステージと第 4 ステージに割り付け、第 3 ステージにおいて LZA と仮数部加算を並列に処理し、第 4 ステージにおいて正規化シフトを行う構成とした。

(3) シミュレーションの結果、最も遅延時間が長いのがパイプラインの第 3 ステージで、仮数部加算器がクリティカルパスとなっている。LZA 処理系回路の遅延は仮数部加算器の遅延よりも小さく、仮数部加算に完全に隠蔽されている。

(4) トランジスタ数は約 54,000 で、そのうち LZA 処理系回路のトランジスタ数は 884 であり全体の 1.6%と小さい。

(5) パターンレイアウトは、CAD ツールによって回路ブロックごとにセルを自動生成し、これをマニュアルで配置及び配線することにより行った。この方法によってすべてをマニュアルで行う場合に比べてレイアウト期間を 1/3 以下に短縮することができた。

以上の設計に基づき、提案した LZA 方式を従来方式と比較し、以下の結果を得た。

(1) 従来提案されていた LZA 方式では、LZA を用いない場合と比較して正規化処理系回路における遅延時間が 2.4ns 短縮され、トランジスタ数が 4,817 増加する。これに対して筆者らが提案した LZA 方式では、遅延時間の短縮が 3.3ns でトランジスタ数の増加は 1,454 個となる。したがって、提案した LZA 方式の方が少ないハードウエアで大きな効果を得ることができる。

(2) 従来の LZA 方式では浮動小数点加算器のサイクルタイムが LZA 処理回路における 5.8ns で律速されるのに対して、提案した LZA 方式では 5.0ns と 0.8ns 高速化される。したがって、その分動作周波数を増加させることができる。

設計した 64 ビット浮動小数点加算器の試作および評価を行い、以下の結果を得た。

(1) 0.5μm CMOS 3 層配線プロセスで試作した。チップサイズは 2.5mm×3.5mm。1 層、2 層および 3 層配線の幅/間隔はそれぞれ 1.2/0.8μm、1.2/1.3μm および 2.0/1.0μm である。

(2) スキャンパスによるテストの結果、電源電圧 3.3V の下でサイクルタイム 6.1ns の動作が確認された。これは、動作周波数に換算すると 164MHz に相当する。

本章で得られた 64 ビット浮動小数点加算器に第 2 章の成果を適用した場合の効果について考察を行い以下の結果を得た。

(1) 第 2 章で開発した 2.6ns の 64 ビット整数加算器を、セル幅の増加も考慮して 64 ビット浮動小数点加算器の 56 ビット整数加算器に適用すると遅延時間は 2.7ns となる。

(2) この加算器を用いることにより、64 ビット浮動小数点加算器のサイクルタイムを 5.2ns に短縮

することができる。これによって、動作周波数を192MHzまで高めることができる。

(3) 64 ビット浮動小数点加算器には第2章で検討した5種類のアーキテクチャのうち、リップルキャリー方式以外の4つのアーキテクチャがいずれも適用可能であり、動作周波数192MHzを達成することができる。面積の面ではキャリースキップ方式が最も小さく、本64 ビット浮動小数点加算器に最適なアーキテクチャであるということができる。

以上のように、浮動小数点加算器に関してアーキテクチャの改良による高速化を検討し、新たなLZA方式を提案することにより高速動作が達成されることを確認した。ここで得られた性能は、CADツールによる自動レイアウトを一部用いているにも関わらず、人手によって最適化された高速浮動小数点演算器と同等のものであり、提案したLZA方式が浮動小数点加算器の高速化に有効であることが確認された。さらに人手をかけることで従来のものを上回る性能を実現することが可能であり、今後3次元CGを中心とする浮動小数点演算器の用途に適用されていくことが期待される。

＜参考文献＞

[1] J. L. Hennessy and D. A. Patterson, "Computer Architecture : A Quantitative Approach," Morgan Kaufman Publishers, 1990.

[2] "A Proposed Standard for Binery Floating-Point Arithmetic -Draft 8.0 of IEEE Task P754," IEEE Computer, pp. 51-62, Mar 1981.

[3] 村田、小国、唐木、「スーパーコンピュータ　－科学技術計算への適用－」、丸善、1985年

[4] 有馬、唐木、土方、日本物理学会編、「スーパーコンピュータ」、培風館、1986年

[5] J. Shipnes, "Graphics Processing with the 88110 RISC Microprocessor," Digest of Papers, IEEE Proc. COMPCON '92, pp.169-174, 1992.

[6] インターネットホームページ：http://www.intel.co.jp/jp/intel/museum/25anniv/html/index.htm

[7] F. Abu-Nofal, R. Avra, K.Bhabuthmal, R. Bhamidipaty, G. Blanck, A. Charnas, P. DelVecchio, J. Grass, J. Grinberg, N. Hayes, G. Haber, J. Hunt, G. Kizhepat, A. Malamy, A. Marston, K. Mehta, S. Nandi, H. V. Nguyen, R. Patel, A. Ray, J. Reaves, A. Rogers, S. Rusu, T. Shay, I. Sidharta, T. Tham, P. Tong, R. Trauben, A. Wong, D. Yee, N. Maan, D. Steiss and L. Youngs, "A Three-Million-Transistor Microprocessor," Dig. Tech. Papers of ISSCC '92, pp. 108-109, Feb. 1992.

[8] E. DeLano, W. Walker, J. Yetter and M. Forsyth, "A High Speed Superscalar PA-RISC Processor," Digest of Papers, IEEE Proc. COMPCON '92, pp.116-121, 1992.

[9] N. K. Yeung, Y.-H. Sutu, T. Y.-F. Su, E. T. Pak, C.-C. Chao, S. Akki, D. D. Yau and R. Lodenquai, "The Design of a 55SPECint92 RISC Processor under 2W," Dig. Tech. Papers of ISSCC '94, pp. 206-207, Feb. 1994.

[10] D. Pham, M. Alexander, A. Arizpe, B. Burgess, C. Dietz, L. Eisen, R. El-Kareh, J. Eno, S. Gary, G. Gerosa, B. Goins, J. Golab, R. Golla, R. Harris, B. Ho, Y.-w. Ho, K. Hoover, C. Hunter, P. Ippolito, R. Jessani, J. Kahle, K. Kishore, B. Kuttanna, S. Litch, S. Mallick, T. Ngo, D. Ogden, C. Olson, S.-H. Park, R. Patel, M. Pham, J. Prado, S. Reeve, R. Reininger, H. Sanchez, M. Shiffli, J. Slaton, G. Thuraisingham, K. Torku, C. Tran, N. Vanderchaaf, P. Voldstad and G. R. Zenhari, "A 3.0W 75SPECint92 Superscalar RISC Processor," Dig. Tech. Papers of ISSCC '94, pp. 212-213, Feb. 1994.

[11] F. A. Ware, W. H. McAllister, J. R. Carlson, D. K. Sun and R. J. Vlach, "64 Bit Monolithic Floating Point Precessors," IEEE J. Solid-State Circuits, vol. SC-17, No.5, pp.898-907, Oct. 1982.

[12] G. Wolrich, E. McLellan, L. Harada, J. Montanaro and R. A. J. Yodlowski, "A High Performance Floating Point Coprocessor," IEEE J. Solid-State Circuits, vol. SC-19, No.5, pp.690-696, Oct. 1984.

[13] K. Takeda, F. Ishino, Y. Ito, R. Kasai and T. Nakashima, "A Single-Chip 80-bit Floating Point Processor," IEEE J. Solid-State Circuits, vol. SC-20, No.5, pp.986-992, Oct. 1985.

[14] J. Fandrianto, B. Y. Woo, "VLSI Floating-Point Processors," Proc. of the 7th Symp. on Computer Arithmetic (ARITH7), pp. 93-100, 1985.

[15] Y. Shimazu, T. Kengaku, T. Fujiyama, E. Teraoka, T. Ohno, T. Tokuda, O. Tomisawa and S. Tsujimichi, "A 50MHz 24b Floating-Point DSP", Dig. Tech. Papers of ISSCC '89, pp. 44-45, Feb. 1989.

[16] S. Komori, H. Takata, T. Tamura, F. Asai, T. Ohno, O. Tomisawa, T. Yamasaki, K. Shima, H. Nishikawa and H. Terada, "A 40MFLOPS 32-bit Floating-Point Processor", Dig. Tech. Papers of ISSCC '89, pp. 46-47, Feb. 1989.

[17] K. Molnar, C. Ho, D. Staver, B. Davis and R. Jerdonek, "A 40 MHz 64-Bit Floating-Point Co-Processor", in ISSCC Dig. Tech. Papers, pp. 48-49, Feb. 1989.

[18] B. Benschneider, W. Bowhill, E. Cooper, M. Gavrielov, P. Gronowski, V. Maheshwari, V. Peng, J. Pickholtz and S. Samudrala, "A 50MHz Uniformly Pipelined 64b Floating-Point Arithmetic Processor", Dig. Tech. Papers of ISSCC '89, pp. 50-51, Feb. 1989.

[19] L. Kohn and S. Fu, "A 1,000,000 Transistor Microprocessor", Dig. Tech. Papers of ISSCC '89, pp. 54-55, Feb. 1989.

[20] G. Taylor, A. Rekow, J. Radke and G. Thompson, "A 100MHz Floating Point/Integer Processor," CICC Proc., pp. 24.5.1-24.5.4, May 1990.

[21] D. Steiss, S. Mangelsdorf, P. Groves, D. Bural, M. Gill, R. Gratias, M. Jassowski, R. Luebs, B. Naas, A. Reynolds, H. Rothermel, W. Walker and T. Wolf, , "A 65MHz Floating-Point Coprocessor for a RISC Processor", Dig. Tech. Papers of ISSCC '91, pp. 94-95, Feb. 1991.

[22] F. Okamoto, Y. Hagihara, C. Ohkubo, N. Nishi, H. Yamada and T. Enomoto, "A 200-MFLOPS 100-MHz 64-b BiCMOS Vector-Pipelined Processor (VPP) ULSI", IEEE J. Solid-State Circuits, vol. 26, pp. 1885-1892, Dec. 1991.

[23] H. Nakano, M. Nakajima, Y. Nakamura, T. Yoshida, Y. Goi, Y. Nakai, R. Segawa, T. Kishida and H. Kadota, "An 80-MFLOPS (Peak) 64-b Microprocessor for Parallel Computer", IEEE J. Solid-State Circuits, vol. 27, pp. 365-372, Mar. 1992.

[24] H. Fujii, C. Hori, T. Takada, N. Hatanaka, T. Demura and G. Ootomo, "A Floating-Point Cell Library and a 100-MFLOPS Image Signal Processor", IEEE J. Solid-State Circuits, vol. 27, pp. 1080-1087, Jul. 1992.

[25] N. Ide, H. Fukuhisa, Y. Kondo, T. Yoshida, M. Nagamatsu, J. Mori, I. Yamazaki and K. Ueno, "A 320-MFLOPS CMOS Floating-Point Processing Unit for Superscalar Processors", IEEE J. Solid-State Circuits, vol. 28, pp. 352-361, Mar. 1993.

[26] D. Dobberpuhl, R. Witek, R. Allmon, R. Anglin, D. Bertucci, S. Britton, L. Chao, R. Conrad, D. Dever, B. Gieseke, S. Hassoun, G. Hoeppner, K. Kuchler, M. Ladd, B. Leary, L. Madden, E. McLellan, D. Meyer, J. Montanaro, D. Priore, V. Rajagopalan, S. Samudrala and S. Santhanam, "A 200-MHz 64-b Dual-Issue CMOS Microprocessor", IEEE J. Solid-State Circuits, vol. 27, No.11, No.11, pp. 1555-1567, Nov. 1992.

[27] W. J. Bowhill, R. L. Allmon, S. L. Bell, E. M. Cooper, D. R. Donchin, J. H. Edomondson, T. C. Fischer, P. E. Gronowski, A. K. Jain, P. L. Kroesen, B. J. Loughlin, R. P. Preston, P. I. Rubinfeld, M. J. Smith, S. C. Thierauf and G. M. Wolrich, "A 300Mhz 64b Quad-Issue CMOS RISC Microprocessor," 1995 ISSCC Dig. Tech. Papers, pp.182-183, February 1995.

[28] P. E. Gronowshi, P. J. Bannon, M. S. Bertone, R. P Blake-Campos, G. A. Bouchard, W. J. Bowhill, D. A. Carlson, R. W. Castelino, D. R. Donchin, R. M. Fromm, M. K. Gowan, A. K. Jain, B. J. Loughlin, S. Mehta, J. E. Meyer, R. O. Mueller, A.Olesin, T. N. Pham, R. P. Preston and P. I. Rubinfeld, "A433MHz 64b Quad-Issue RISC Microprocessor," 1996 ISSCC Dig. Tech. Papers, pp.222-223, February 1996.

[29] A. K. Jain, R. P. Preston, P. J. Bannon, M. S. Bertone, R. P Blake-Campos, G. A. Bouchard, D. S. Brasili, D. A. Carlson, R. W. Castelino, K. M. Clark, S. Kobayashi, B. P. Lilly, S. Mehta, B. S. Miller, R. O. Mueller, A. Olesin and Y. Saito, "1.38cm^2 550Mhz Microprocessor with Multimedia Extensions," 1997 ISSCC Dig. Tech. Papers, pp.174-175, February 1997.

[30] H. Suzuki, Y. Nakase, H. Makino, H. Morinaka and K. Mashiko, "Leading-zero Anticipatory Logic for High-speed Floating Point Addition," CICC Proc., pp. 589-592, May 1995.

[31] 鈴木、森中、牧野、中瀬、益子、角、「高速浮動小数点加算器のための桁落ちシフト量予測回路の提案」、電子情報通信学会技術研究報告、ED95-98, pp.7-12, 1995.

[32] H. Suzuki, H. Makino, H. Morinaka, Y. Nakase, K. Mashiko and T. Sumi, " Leading-zero Anticipatory Logic for High-speed Floating Point Addition," IEEE J. Solid-State Circuits, vol. 31, No.8, pp. 1157-1164, Aug. 1996.

[33] E. Hokenek, R. K. Montoye and P. W. Cook: "Second-generation RISC floating point with multiply-add fused," IEEE J. Solid-State Circuits, vol. 25, pp. 1207-1213, Oct. 1990.

[34] E. Hokenek and R. Montoye, "Leading-zero anticipator (LZA) in the IBM RISC System/6000 floating-point execution unit", IBM J. Res. Develop., vol. 34, pp. 71-77, Jan. 1990.

第５章　浮動小数点乗算器の高性能化に関する研究

５．１　緒言

浮動小数点乗算は、浮動小数点加算とともに浮動小数点演算の中で最も基本となるもので、浮動小数点計算システムには必要不可欠な演算である。一般に全ての演算は多項式で近似すれば、加算と乗算の組み合わせで表すことができるため、浮動小数点加算器と浮動小数点乗算器の２種類の演算器があればあらゆる浮動小数点演算を実行することができる。浮動小数点乗算も、浮動小数点加算の場合と同様に通常のALUを用いても実行できるが、浮動小数点加算よりもさらに多くの計算ステップが必要とされ、膨大な時間を必要とする。このため浮動小数点演算器は浮動小数点加算器と浮動小数点乗算器の両方を備えているのが一般的である。浮動小数点データのフォーマットについても浮動小数点加算器と同様に４．１節で示したIEEE-754のフォーマット[1]が用いられる。

図５－１に浮動小数点乗算の一般的な手順を示す[2]。２つの浮動小数点データが入力されると、まず２数の指数部の和を求め、さらに仮数部の乗算を行う。この仮数部の乗算は整数乗算となるので、第３章で研究した整数乗算器を適用することができる。仮数部を乗算すると１ビットの桁あふれが生じる場合があるので、これを処理するために桁あふれの有無をチェックする。桁あふれがあれば仮数部を１ビット右方向へシフトし、同時に指数部に「1」を加える。これが浮動小数点乗算における正規化である。そして、最後に丸めと例外処理を行って最終的な乗算結果として出力する。丸めに関しても浮動小数点加算器の場合と同様に「最近値への丸め」、「０方向への丸め」、「＋∞方向への丸め」および「－∞方向への丸め」の４種類の丸めモードをサポートして、モードの切り替えによってそれぞれの丸めを選択できるようにする必要がある。また、例外処理に関しても、オーバーフローやアンダーフロー検出し、これに則った出力を生成する必要がある。

浮動小数点乗算では、浮動小数点加算において見られた「桁落ち」の現象は起きないため、これに伴う複雑な処理は必要とされない。また、入力の桁合わせも行う必要がないので多ビットのシフトも不要である。その代わりに、仮数の乗算部が入力ビット数の２乗の規模となるために、トランジスタ数が大きく大面積を要し、同時に計算時間がかかるという問題点がある。特に、倍精度浮動小数点データの乗算を実行するためには54×54ビットの仮数部乗算が必要となり、この仮数部乗算が浮動小数点乗算器の遅延時間および面積のうちの大きな部分を占めるようになる。したがって、高性能の浮動小数点乗算器を実現するためには、この仮数部乗算の部分をいかに高速化し、かつトランジスタ数を低減するかが重要となる。

図５－１　浮動小数点乗算の実行手順

筆者らが研究を開始した1990年よりも以前から現在にかけて，浮動小数点乗算器に関する研究は数多く報告されている[3]-[19]。浮動小数点加算器と同様に，初期のものは長いサイクルタイムをかけて計算を行うものであったが[3], [5]、その後ステップ毎に全体をいくつかに分割してレジスタを挿入し、クロック信号によってレジスタ間でデータを送る「パイプライン方式」が用いられるようになっている[4], [6]-[19]。さらに，データ駆動型のアーキテクチャで処理効率を高める試みや[10]、バイポーラトランジスタを用いて高速化を図る試み[14]なども報告されている。仮数の乗算部に関しては，はじめはハードウエア量の制限から８ビット程度の小さな乗算器を何回も繰り返し用いることによって実現されていたが[3], [4], [8]、その後半導体の微細化技術の進歩に伴って32×32ビット程度の整数乗算器が内蔵されるようになり、単精度の乗算の場合は１回、倍精度の場合は２回の乗算で実行できるようになった。これによって浮動小数点乗算性能が飛躍的に向上した[5]-[7], [10]-[13], [15], [18]。さらに微細化技術が進歩して近年では54×54ビット程度の整数乗算器が内蔵されるようになり、仮数部の乗算を１回の整数乗算で実行可能としている[16], [17], [19]。上で示した研究例はいずれも浮動小数点演算専用

のプロセッサに関するものであったが、汎用のマイクロプロセッサにおいても倍精度の浮動小数点乗算が可能な浮動小数点データパスを内蔵したものが最近発表され[20]-[27]，既にワークステーションやパーソナルコンピュータをはじめとする様々なコンピュータに搭載されている。また、整数乗算器単体の研究においても最近54×54ビット乗算器の高速化や小面積化に関するものが発表されているが[28]-[32]，これらはいずれも浮動小数点乗算器の仮数部の乗算用に作られたものである。第３章で述べた筆者らによる整数乗算器に関する研究成果も、そのまま浮動小数点乗算器の仮数部の乗算に適用可能なものである。

前章までにおいても何度か触れてきたように本研究の大きな用途の一つにいわゆるバーチャルリアリティを目指した３次元CGがあり、これを実現するためには３次元CGにおける演算を高速に処理する必要がある。３次元CGにおける演算のうちで最も大きな部分を占めるのが浮動小数点による座表計算である[33]-[35]。この座表計算には膨大な量の浮動小数点乗算が要求される。したがってこれを高速化するためには、浮動小数点乗算器の高速化が必要不可欠である。先にも述べたように浮動小数点乗算器の性能と面積を決定するのは仮数部の乗算を行う整数乗算器であり、この点においては第３章の研究成果である54×54ビット整数乗算器を適用することによって、他のものよりも高速かつ小面積の浮動小数点乗算器を実現することができる。しかし、今日の３次元CGはバーチャルリアリティを一部実現しているものの、さらなる高画質の画像を求める要求が強く、これに応えるためにはCGに対する演算性能をさらに高める必要がある。

CGに対する演算性能を高める方法としては、半導体の微細化や回路の工夫によって浮動小数点演算器自体を高速化する以外にCGに有効な機能を付加することが考えられる。筆者らはこの点に着目して，回路技術による高速化とCGに効果的な機能の付加の両面から、浮動小数点乗算器の高速化ならびに高機能化に関する研究を行った[36]-[38]。本章では、まず５．２節において，CGに適した機能として「CG乗算」機能を提案し、それを効率よく浮動小数点乗算器にインプリメントする方法について述べる。さらに、５．３節において「CG乗算」機能を搭載した64ビット高速浮動小数点乗算器の設計、試作および評価結果について述べる。クリティカルパスの高速化を中心とする回路技術には第３章の成果を適用している。そして最後に５．４節においてまとめを行う。

5．2　CG乗算機能の搭載による浮動小数点乗算器の高性能化に関する研究

5．2．1　CG乗算機能の提案

浮動小数点計算システムの大きな用途である3次元CGにおいては、それぞれ8ビットからなるRGB（赤、緑および青）の3つの色データが一組になったピクセルデータ（画素データ）と浮動小数点数からなる係数との乗算を実行することがしばしば求められる。これまで様々な浮動小数点乗算器が報告されているが[3]-[19]、これらはいずれも通常の浮動小数点乗算機能しか持たないため、このようなCGに特有の乗算は、浮動小数点数を整数に変換した後に整数演算ユニットで多数のステップをかけて処理する必要があった。CG用の機能を一部搭載した報告もあるが[39]、これも演算の前処理や後処理に複数の演算ステップを必要としている。したがって、この処理をハードウエアによって1ステップで実行することができればCGの処理を大幅に短縮することができる。そこで、筆者らは「CG乗算」と呼ばれるCGに適した機能を提案した。図5－1にこの機能を示す。CG乗算とは、RGBおよびα（他の属性）の4つの8ビットデータが一組になった32ビットピクセルデータと浮動小数点数との乗算を1ステップで実行する機能である。出力もピクセルデータであり、その各フィールドは、入力ピクセルデータの各フィールドにそれぞれ浮動小数点数βが乗じられた値となる。CG乗算機能は、CGにおいてαブレンディングなどのピクセル操作に便利であるほか、グーローシェーディング、フォンシェーディングおよびレイトレーシングといった各種シェーディングにおける輝度計算にも用いることができ、効率の高い処理を実現することが可能となる[33]-[35]。図5－2に従来方式との処理の比較を示す。(a)は通常の浮動小数点乗算器で処理する場合である。この場合は、まず浮動小数点数βを整数β'に変換する。次にピクセルデータの4つのフィールドをそれぞれ取り出して順にβ'を乗じ、最後に4つの乗算結果を8ビットづつ組み合わせて結果を得る。この処理には少なくとも7個以上のステップが必要とされる。(b)はCG用の機能を持つ従来の浮動小数点乗算器[39]で処理する場合である。この浮動小数点乗算器には、「pack」および「unpack」というCG用の命令が搭載されている。「unpack」命令は、32ビットのピクセルデータの4つのフィールドの間および左側に8ビットの「0」を挿入して64ビットのデータに変換する命令であり、「pack」命令は、「unpack」命令の逆で64ビットデータを32ビットデータに圧縮する命令である。この場合には、まず(a)の場合と同様に浮動小数点数βを整数β'に変換し、次に「unpack」命令によって64ビットに拡張する。さらにこの拡張されたデータにβ'を掛けることにより、4つの16ビットデータβR, βG, βBおよびβαからなる64ビットデータを得ることができる。この場合、予めフィールド間に挿入した8ビットの「0」のために計算後の4つの16ビットデータは互いに重なり合わずに独立に出力される。次に、この64ビットデータを「pack」命令によって32ビットに圧縮して結果を得る。この処理には少なくとも4個以上のステップが必要とされる。これに対し(c)が提案したCG乗算機能の場合で、1ステップで同様の処理を行うものである。これによって同じ処理を従来の少なくとも4倍以上のスピードで実現することができる。

図5－2　CG乗算機能

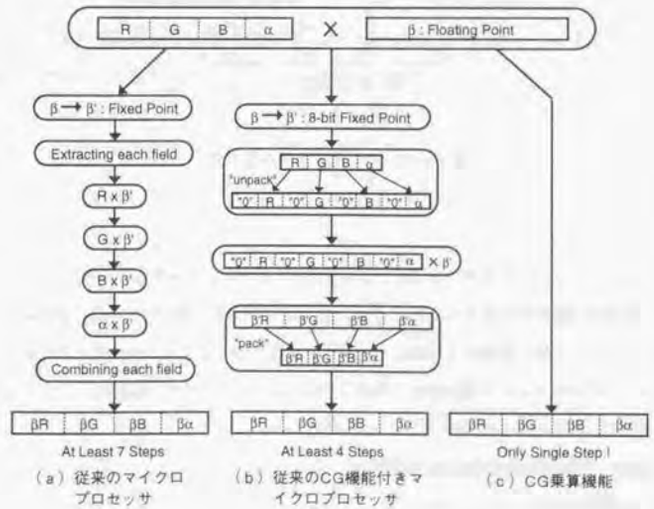


図5－3　CG乗算機能の従来方式との比較

5．2．2　CG 乗算機能の実現方法の検討

CG 乗算機能を少量のハードウエアで浮動小数点乗算器に搭載するために、アルゴリズムの最適化を行った。図５－４に最適化されたアルゴリズムを示す。

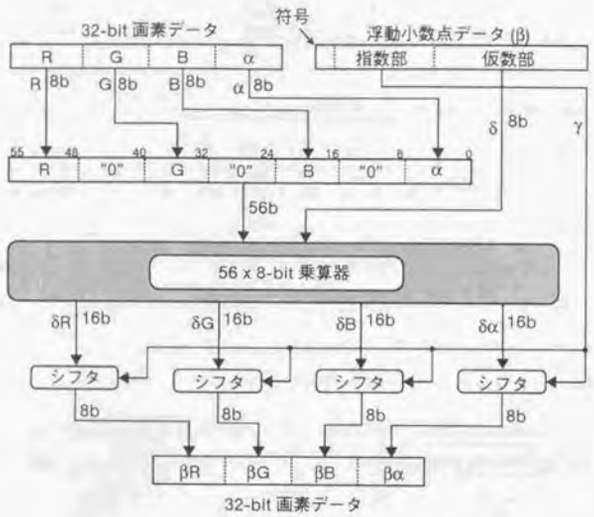


図５－４　CG 乗算のアルゴリズム

入力される２つのソースオペランドは、一方はR、G、Bおよびαからなる32ビットピクセルデータであり、他方は浮動小数点数βである。βは、IEEE-754 の標準に基づいて符号、指数部および仮数部の３つのフィールドから構成される[1]。本アルゴリズムでは、まずピクセルデータの各フィールドの間に８ビットのゼロデータ「00000000」を挿入することにより、ビット幅を56ビットに拡張する。例えば、入力するピクセルデータを、

00001111 00001010 00001110 00001101

とすると、拡張後の56ビットデータは

00001111 00000000 00001010 00000000 00001110 00000000 00001101

となる。次にこの56ビットデータと浮動小数点数βの仮数部の上位８ビット（δ）との乗算を行う。

従ってこの乗算には 56×8 ビット以上の規模の整数乗算器が必要となる。整数乗算の結果は、それぞれδR、δG、δBおよびδαの値を持つ４つの16ビットデータとなる。例えば、

δ＝1.0001000

とすると乗算結果は、

000011111.1110000、000010101.0100000、000011101.1100000、000011011.1010000

となる。それぞれの16ビットデータは、予め挿入された８ビットのゼロデータのために互いに重なり合わず、別々に出力される。次に４つの16ビットデータは入力の浮動小数点数の指数部の値（γ）によってシフトが行われ、正しい値に補正される。例えば、指数部（γ）を十進数で、

γ＝３

とすると、それぞれの16ビットデータは３ビットづつ左方向へシフトされ、

0000111111111.0000、000010101010.0000、000011101110.0000、000011011101.0000

となる。最後にそれぞれの整数部の下位８ビットずつを取り出してこれをつなぎ合わせることにより結果の32ビットピクセルデータが出力される。上の例では、

11111111 10101010 11101110 11011101

となり、これは入力の各フィールドに浮動小数点数βが乗じられた値となっている。

このアルゴリズムは浮動小数点乗算器に搭載するのに適している。すなわち、56×8 ビット整数乗算器は仮数部用の整数乗算器と容易に共用できるほか、この機能にに必要な４つのシフタは、整数乗算器の規模と比較すると少量のトランジスタで構成できるためである。

5．3　64ビット浮動小数点乗算器への適用[36]-[38]

5．3．1　回路構成

以上の提案に基づき、CG 乗算機能を搭載した 64 ビット浮動小数点乗算器を設計した。表５－１に本 64 ビット浮動小数点乗算器の機能の一覧を示す。入出力のフォーマットとしては IEEE-754 準拠の単精度および倍精度浮動小数点フォーマットの両方が扱えるようにした。算術演算としては単精度及び倍精度の乗算機能を持つ。また、CG 乗算機能として単精度および倍精度の浮動小数点数と 32 ビットピクセルデータとの乗算機能をサポートした。丸め機能としては、IEEE-754 規格で定められた４種類の丸めである「最近値への丸め」、「０方向への丸め」、「＋∞方向への丸め」および「－∞方向への丸め」のすべてをインプリメントした。

表５－１　64 ビット浮動小数点乗算器の機能一覧

フォーマット	IEEE-754規格準拠の単精度及び倍精度 浮動小数点フォーマット
算術演算	乗算 （単精度、倍精度の両方をサポート）
CG乗算	単精度浮動小数点数×32ビットピクセルデータ 倍精度浮動小数点数×32ビットピクセルデータ
丸め	最近値への丸め、０方向への丸め ＋∞方向への丸め、－∞方向への丸め

図４－14 に全体のブロック図を示す。浮動小数点加算器の場合と同様に、サイクルタイムを短縮するために全体を５段のパイプライン構成とした。パイプラインの第１ステージでは（フォーマット）において、オペレーションに応じた入力オペランドのデータの並べ替えが行われる。入力オペランドはＡとＢの２つで、浮動小数点乗算時には単精度あるいは倍精度浮動小数点数が入力する。CG 乗算時のピクセルデータ入力はＡのみに許される。ピクセルデータに対する８ビットのゼロデータの挿入もこのステージで行われる。

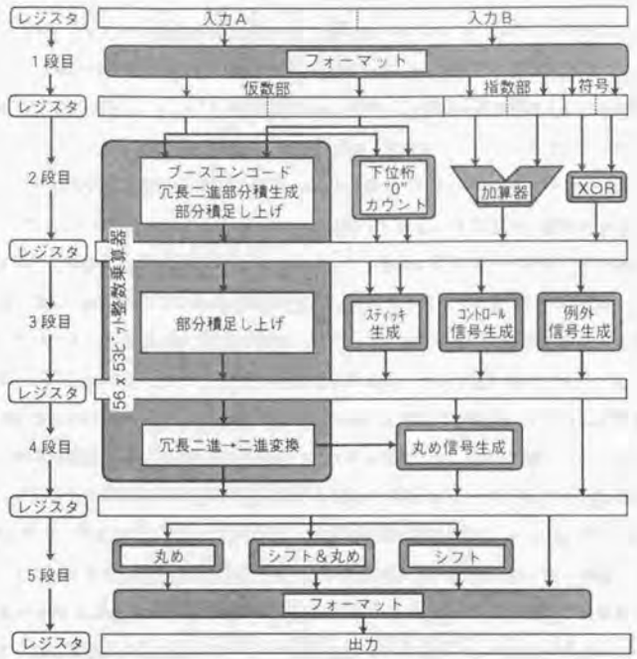


図５－５　浮動小数点乗算器のブロック図

浮動小数点乗算器の遅延時間および面積のうち最も大きな部分を占める仮数部乗算は、パイプラインステージの２段目から４段目にかけて実行される。これは、56×53 ビット整数乗算器によって実行される。この整数乗算器には第３章の研究成果である高速冗長二進アーキテクチャ[39]-[41]が適用され、高速の仮数部乗算を実現している。仮数部乗算は３段階のステップに分けられ、それぞれがパイプラインステージの２－４段目に割り付けられている。パイプラインステージの第２段目においては、２次のブースエンコーディング[42]、冗長二進部分積の生成およびワレストリー[43]による部分積の足し上げの一部が実行され、３段目のパイプラインステージにおいて部分積の足し上げの残りの部分が実行される。最後に４段目のパイプラインステージにおいて冗長二進数から通常の二進への変換が行われる。通常の浮動小数点乗算器においては、仮数部の乗算に 53×53 ビットの整数乗算器が用いられるのに対し、本設計で 56×53 ビット乗算器が用いられるのは、前節で述べたように CG 乗算においては

ピクセルデータのデータ幅がゼロデータの挿入によって56ビットになるためである。

この他、パイプラインの第2ステージでは符号ビットと指数部の計算が行われる。符号ビットは、二つの入力オペランドが同符号の際には正、異符号の際には負の値を出力するために、二つの入力オペランドの符号ビットの排他的論理和を（XOR）においてとることによって求められる。指数部は、二つの指数部を（加算器）において加算することによって求められる。

また、丸めに用いるスティッキビットに対しては、その生成を仮数部乗算の結果からではなく、入力オペランドから直接求めることによって、高速の生成を実現している。スティッキビットとは、IEEE-754 規格の丸めを実行するために必要なビットで、仮数部の演算結果の有効最下位桁よりも以下の桁の値が、最下位桁の2分の1以上かあるいは以下であるかを示すビットである。最近値への丸めノードの際にスティッキビットをチェックすることによって最下位桁の2分の1以上のときは切り上げを行い、以下の場合は切り捨てを行うことができる。スティッキビットを求めるためには、通常仮数部の乗算結果において、有効最下位桁よりも2桁下位のビットから最下位のビットまでの全てのビットの論理和を取る必要があり、パイプラインステージの4段目で生成される仮数部乗算の結果から多ビットの論理和を生成すると、この部分の遅延のために高速動作は困難となってしまう。これを解決するために、本 64 ビット浮動小数点乗算器では、まず入力オペランドの仮数部の下位に並ぶ「0」の数を数え、この「0」の数に基づいてスティッキビットを発生させている。すなわち、2つの入力オペランドの仮数部の下位に並ぶ「0」の数の合計が仮数部の乗算結果の下位に並ぶ「0」の数と等しくなるので、この数が一定の値（単精度の場合と倍精度の場合で異なる）よりも大きいか小さいかを判別することによってスティッキビットを生成している。このような手法によって、スティッキビットの発生を仮数部乗算と並行して行うことができるのでスティッキビットの発生に伴う遅延時間の増大をなくすことができる。パイプラインの第2ステージでは入力オペランドの仮数部の下位に並ぶ「0」の数を（下位桁"0"カウント）において数えている。

パイプラインの第3ステージでは、スティッキビット発生のために第2ステージで求めた入力オペランドの仮数部の下位に並ぶ「0」の数に基づいてスティッキ発生回路（スティッキ生成）においてスティッキビットを発生させる。また、この他に第3ステージでは、制御信号発生器（コントロール信号生成）において丸めモードを制御する信号が生成され、例外信号発生器（例外信号生成）において計算結果のオーバーフロー、アンダーフローおよび不正演算等を示すフラグ信号が生成される。

パイプラインの第4ステージでは、丸め信号発生器（丸め信号生成）において仮数部の計算結果と上記丸めモード制御信号から丸め信号が生成される。この丸め信号は丸めモードに応じて最近値への丸め、ゼロへの丸め、＋∞への丸めおよび－∞への丸めの4通りのうちのいずれかを実行するための信号として生成され、第5ステージに送られる。

第5ステージでは，第4ステージで生成した丸め信号に従って仮数部の丸めを丸め回路（丸め）において行い、同時に仮数部が桁あふれした際の右方向シフトおよび丸めを（シフト＆丸め）において行う。またCG乗算の際に各フィールドを整数化するためのシフト操作もシフト回路（シフト）において行う。最後に選択回路（フォーマット）において、これら3つのうちの必要なものが選択され、出力のフォーマットに従った形に修正されて結果が出力される。

本64ビット浮動小数点乗算器は第3章の3．2節で定めたルールに従ってクリティカルパスを全てインバータ、2入力NANDおよびトランスミッションゲート（TG）の三種類のゲートで構成し、TGの直列接続は最大2段までとしている。例えば、パイプラインの第2および第3ステージのクリティカルパスは、冗長二進加算器であるが、これは3．2節のルールに従って図3－15で示した回路で構成されている。パイプラインの第4ステージのクリティカルパスは、冗長二進数から二進数への変換器であるが、これも3．2節のルールに従って図3－13(b)で示した回路で構成されている。なお、この冗長二進数→二進数変換器と同様の構成の回路が、5段目のパイプラインステージのクリティカルパスである丸め回路の中のインクリメント回路にも用いられている。

表5－2に各パイプラインステージのクリティカルパスにおけるインバータ、2入力NANDおよびTGの段数と、シミュレーションによるトータルの遅延時間を示す。シミュレーションパラメータは0.5μm CMOSプロセスのものを用い、電源電圧は3.3Vとした。ゲート数のトータルでは，1段目と2段目が10と比較的小さく、3段目が15、4段目が16、5段目が最も多く18となっている。遅延時間では3段目、4段目および5段目がほぼ同じでこの3段がサイクルタイムを決定している。5段目のゲート段数が最も多いにも関わらず3段目と遅延時間が同じであるのは、3段目の冗長二進加算器の出力がトリー状の加算を行うために大きな配線容量が付き、ここでの遅延が大きいためである。

本浮動小数点乗算器のトランジスタ数は138,000で、このうちCG乗算機能の付加によるトランジスタ数の増加は5,500と全体の約4％に過ぎない。この増加の大半は仮数部乗算用の整数乗算器を53×53ビット構成から56×53ビット構成へと広げたことに起因している。また、CG乗算機能の付加によるクリティカルパスの遅延の増加は全くない。

パターンレイアウトは第4章で述べた浮動小数点加算器の場合と同様で0.5μmCMOSの3層配線プロセスで行った。レイアウト方式も浮動小数点加算器の場合と同様で，回路ブロック毎にセルをCADツールを用いて自動生成し、これをマニュアルで配置および配線するという手法を用いることにより、人手でレイアウトする場合に比べて期間を1/3以下に短縮している。

表５－２　各段のクリティカルパスの構成

ﾊﾟｲﾌﾟﾗｲﾝ ｽﾃｰｼﾞ	ｲﾝﾊﾞｰﾀ	2-NAND	TG	計	遅延時間 (ns)
1段目	7	1	2	10	2.5
2段目	3	4	4	10	2.9
3段目	3	6	6	15	3.5
4段目	3	0	13	16	3.4
5段目	5	1	12	18	3.5

５．３．２　試作および評価結果

設計した64ビット浮動小数点乗算器をゲート長0.5μm CMOSプロセスで試作した。図５－６にチップ写真を示す。パッド領域を除くチップの有効面積は4.2 x 5.1mm²である。浮動小数点加算器と同様に配線には３層配線を用いており、１層、２層および３層配線の幅/間隔はそれぞれ1.2/0.8μm、1.2/1.3μm、2.0/1.0μmである。１層配線と２層配線は主にそれぞれ横方向と縦方向の配線として用いられ、３層配線は横方向の電源およびGND配線とクロック配線に用いられている。

試作した64ビット浮動小数点乗算器には、浮動小数点加算器と同様にスキャンパスによるテスト回路が設けられており、同じくメインクロック、インプットクロックおよびアウトプットクロックの３種類のクロック信号によるテストを行った。すなわち、入力オペランドのデータは、まずインプットクロックによって入力シフトレジスタに蓄えられ、次に５サイクルのメインクロックによって浮動小数点乗算器に入力および計算され、出力シフトレジスタに蓄えられる。最後にアウトプットクロックによって計算結果が外部に取り出され、期待値データと比較される。動作の評価には10万パターン以上のテストパターンを用いた。これらのうち、70パターンは人手で作成したクルティカルパターンで、残りは自動生成したランダムパターンである。

図５－７にテスト結果として、サイクルタイムと電源電圧に対するシュムープロットを示す。「P」と書かれている領域が正常動作した領域である。電源電圧3.3Vに対する動作可能な最小サイクルタイムは3.5nsである。これは、動作周波数286MHzに相当する。この動作周波数は、CG機能の搭載にもかかわらず研究当時の最高速のマイクロプロセッサ[21]と同程度であり、CG機能の搭載による速度劣化が無いことを示している。表５－３にチップの諸元を示す。

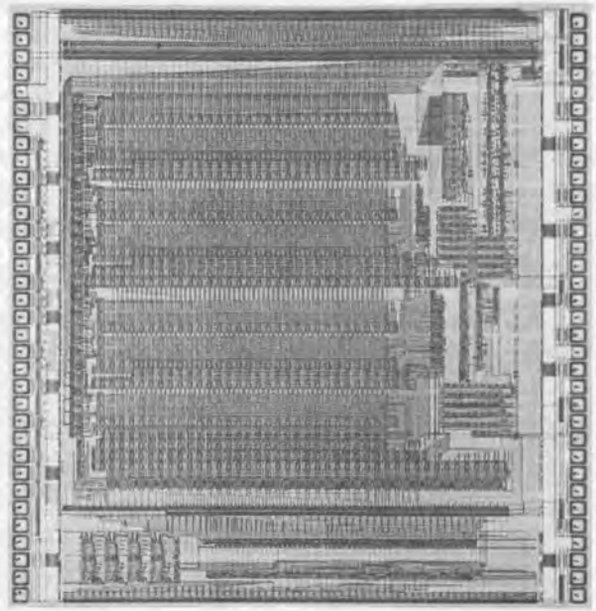


図５－６　64ビット浮動小数点乗算器チップ写真

図5－7　サイクルタイムと電源電圧に対するシュムープロット

表5－4　浮動小数点乗算器のチップ諸元

Function	・FP multiplication ・CG multiplication (IEEE Single/Double Precision Format)
Process	0.5μm CMOS with Triple Metal
Line/Space of Metal	1st ： 1.2/0.8μm 2nd ： 1.2/1.3μm 3rd ： 2.0/1.0μm
Active Area	4.2 x 5.1 mm^2
Transistor Count	138,000
Power Supply	3.3 V
Frequency	286 MHz
Power Dissipation	5.1mW/MHz

5．4　結言

本章では、浮動小数点加算器と並ぶ高速浮動小数点計算プロセッサの主要構成要素として，64 ビット浮動小数点乗算器に関する研究内容について述べた。

まず、CG に対する性能を向上させるために CG に対して有効な機能に関する研究を行い、以下の結果を得た。

(1) CG におけるピクセルデータ（画素データ）と浮動小数点数とを直接乗算する「CG 乗算機能」を提案した。この機能は、CG において α ブレンディングなどのピクセル操作に便利であるほか、グーローシェーディング、フォンシェーディングおよびレイトレーシングといった各種シェーディングにおける輝度計算にも用いることができ、効率の高い処理を実現することができる。

(2) 提案した CG 乗算機能を用いた場合の演算ステップを CG 機能を全く持たない場合および従来の CG 用命令を用いた場合の演算ステップと比較することにより、CG 乗算機能にって CG 機能を持たない場合の 7 倍以上、および従来の CG 用命令を用いた場合の 4 倍以上の性能向上が可能であることを示した。

CG 乗算機能を浮動小数点乗算器に有効にインプリメントする方法に関する検討を行い以下の結果を得た。

(1) ピクセルデータの各フィールド間に 8 ビットの「0」データを挿入することにより、従来の浮動小数点乗算器のハードウエアをほとんど変更せずに CG 乗算機能をインプリメントするアルゴリズムを提案した。

(2) このアルゴリズムには、56×53 ビットの整数乗算器が必要となる。従来の CG 乗算機能を持たない浮動小数点乗算器では、53×53 ビットのものが使用されるため、3 ビット分ビット幅が大きくなり、その分だけハードウエアが増大する。しかし、これによるトランジスタ数の増加量は全体の 4% と小さい。

以上の提案に基づき、CG 乗算機能を搭載した 64 ビット浮動小数点乗算器を設計した。設計の主な要点は以下の通り。

(1) IEEE-754 準拠の単精度および倍精度浮動小数点乗算機能と CG 乗算機能を搭載した。CG 乗算機能としては単精度および倍精度の浮動小数点数と 32 ビットピクセルデータとの乗算機能をサポートした。

(2) サイクルタイムを低減して動作周波数を高めるために 5 段のパイプライン構成とした。

(3) 浮動小数点乗算器の性能を律速する仮数部乗算には、第3章の研究成果である高速冗長二進アーキテクチャによる56×53ビット整数乗算器を用い、これによって高性能化を図った。

(4) スティッキビットを、乗算結果からではなく入力データから直接求めることによって高速のスティッキ生成を実現した。

(5) 第4章で述べた浮動小数点加算器と同様に、レイアウト方式は回路ブロック毎にセルをCADツールを用いて自動生成し、これをマニュアルで配置および配線するという手法を用い、これによって、人手でレイアウトする場合に比べて期間を1/3以下に短縮した。

(6) トランジスタ数は全部で138,000で、このうちCG乗算機能の追加による増加分が5,500（4%）である。

設計した64ビット浮動小数点乗算器の試作および評価を行い、以下の結果を得た。

(1) 0.5μm CMOS 3層配線プロセス技術を用いて試作した。チップ面積は4.2×5.1mm^2。1層、2層および3層配線の幅/間隔はそれぞれ1.2/0.8μm、1.2/1.3μmおよび2.0/1.0μmである。

(2) スキャンパスによるテストの結果、電源電圧3.3Vの下でサイクルタイム3.5nsの動作が確認された。これは、動作周波数に換算すると286MHzに相当する。

このように、浮動小数点乗算器に関しては、第3章の研究成果を適用することによって回路面での高速化を図るとともに、CGに適した機能をハードウエアの増加をほとんど伴わずにインプリメントすることができた。これによってCGに適した機能を搭載したにもかかわらず、浮動小数点乗算器単体としては、最高速の動作周波数を達成することができた。浮動小数点加算器の場合と同様に、CADツールによる自動レイアウトを一部用いているにも関わらず、人手によって最適化された高速浮動小数点演算器と同等の性能を達成しており、提案したアーキテクチャが浮動小数点乗算器の高速化に有効であることが確認された。さらに人手をかけることで従来のものを上回る性能を実現することが可能であり、今後浮動小数点加算器とともに3次元CGを中心とする浮動小数点演算器の用途に適用されていくことが期待される。

<参考文献>

[1] "A Proposed Standard for Binery Floating-Point Arithmetic -Draft 8.0 of IEEE Task P754," IEEE Computer, pp. 51-62, Mar 1981.

[2] J. L. Hennessy and D. A. Patterson, "Computer Architecture : A Quantitative Approach," Morgan Kaufman Publishers, 1990.

[3] F. A. Ware, W. H. McAllister, J. R. Carlson, D. K. Sun and R. J. Vlach, "64 Bit Monolithic Floating Point Precessors," IEEE J. Solid-State Circuits, vol. SC-17, No.5, pp.898-907, Oct. 1982.

[4] G. Wolrich, E. McLellan, L. Harada, J. Montanaro and R. A. J. Yodlowski,"A High Performance Floating Point Coprocessor," IEEE J. Solid-State Circuits, vol. SC-19, No.5, pp.690-696, Oct. 1984.

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

[6] K. Takeda, F. Ishino, Y. Ito, R. Kasai and T. Nakashima, "A Single-Chip 80-bit Floating Point Processor," IEEE J. Solid-State Circuits, vol. SC-20, No.5, pp.986-992, Oct. 1985.

[7] J. Fandrianto, B. Y. Woo, "VLSI Floating-Point Processors," Proc. of the 7th Symp. on Computer Arithmetic (ARITH7), pp. 93-100, 1985.

[8] B. K. Bose, L. Pei, G. S. Taylor and D. A. Patterson, "Fast Multiply and Divide for a VLSI Floating-Point Unit," Proc. of the 8th Symp. on Computer Arithmetic (ARITH8), pp. 87-94, 1987.

[9] Y. Shimazu, T. Kengaku, T. Fujiyama, E. Teraoka, T. Ohno, T. Tokuda, O. Tomisawa and S. Tsujimuchi, "A 50MHz 24b Floating-Point DSP", Dig. Tech. Papers of ISSCC '89, pp. 44-45, Feb. 1989.

[10] S. Komori, H. Takata, T. Tamura, F. Asai, T. Ohno, O. Tomisawa, T. Yamasaki, K. Shima, H. Nishikawa and H. Terada, "A 40MFLOPS 32-bit Floating-Point Processor", Dig. Tech. Papers of ISSCC '89, pp. 46-47, Feb. 1989.

[11] K. Molnar, C. Ho, D. Staver, B. Davis and R. Jerdonek, "A 40 MHz 64-Bit Floating-Point Co-Processor", in ISSCC Dig. Tech. Papers, pp. 48-49, Feb. 1989.

[12] B. Benschneider, W. Bowhill, E. Cooper, M. Gavrielov, P. Gronowski, V. Maheshwari, V. Peng, J. Pickholtz and S. Samudrala, "A 50MHz Uniformly Pipelined 64b Floating-Point Arithmetic Processor", Dig. Tech. Papers of ISSCC '89, pp. 50-51, Feb. 1989.

[13] L. Kohn and S. Fu, "A 1,000,000 Transistor Microprocessor", Dig. Tech. Papers of ISSCC '89, pp. 54-55, Feb. 1989.

[14] G. Taylor, A. Rekow, J. Radke and G. Thompson, "A 100MHz Floating Point/Integer Processor," CICC Proc., pp. 24.5.1-24.5.4, May 1990.

[15] D. Steiss, S. Mangelsdorf, P. Groves, D. Bural, M. Gill, R. Gratias, M. Jassowski, R. Luebs, B. Naas, A. Reynolds, H. Rothermel, W. Walker and T. Wolf, , "A 65MHz Floating-Point Coprocessor for a RISC Processor", Dig. Tech. Papers of ISSCC '91, pp. 94-95, Feb. 1991.

[16] F. Okamoto, Y. Hagihara, C. Ohkubo, N. Nishi, H. Yamada and T. Enomoto, "A 200-MFLOPS 100-MHz 64-b BiCMOS Vector-Pipelined Processor (VPP) ULSI", IEEE J. Solid-State Circuits, vol. 26, pp. 1885-1892, Dec. 1991.

[17] H. Nakano, M. Nakajima, Y. Nakamura, T. Yoshida, Y. Goi, Y. Nakai, R. Segawa, T. Kishida and H. Kadota, "An 80-MFLOPS (Peak) 64-b Microprocessor for Parallel Computer", IEEE J. Solid-State Circuits, vol. 27, pp. 365-372, Mar. 1992.

[18] H. Fujii, C. Hori, T. Takada, N. Hatanaka, T. Demura and G. Ootomo, "A Floating-Point Cell Library and a 100-MFLOPS Image Signal Processor", IEEE J. Solid-State Circuits, vol. 27, pp. 1080-1087, Jul. 1992.

[19] N. Ide, H. Fukuhisa, Y. Kondo, T. Yoshida, M. Nagamatsu, J. Mori, I. Yamazaki and K. Ueno, "A 320-MFLOPS CMOS Floating-Point Processing Unit for Superscalar Processors", IEEE J. Solid-State Circuits, vol. 28, pp. 352-361, Mar. 1993.

[20] D. Dobberpuhl, R. Witek, R. Allmon, R. Anglin, D. Bertucci, S. Britton, L. Chao, R. Conrad, D. Dever, B. Gieseke, S. Hassoun, G. Hoeppner, K. Kuchler, M. Ladd, B. Leary, L. Madden, E. McLellan, D. Meyer, J. Montanaro, D. Priore, V. Rajagopalan, S. Samudrala and S. Santhanam, "A 200-MHz 64-b Dual-Issue CMOS Microprocessor", IEEE J. Solid-State Circuits, vol. 27, pp. 1555-1567, Nov. 1992.

[21] W. J. Bowhill, R. L. Allmon, S. L. Bell, E. M. Cooper, D. R. Donchin, J. H. Edomondson, T. C. Fischer, P. E. Gronowski, A. K. Jain, P. L. Kroesen, B. J. Loughlin, R. P. Preston, P. I. Rubinfeld, M. J. Smith, S. C. Thierauf and G. M. Wolrich, "A 300Mhz 64b Quad-Issue CMOS RISC Microprocessor," 1995 ISSCC Dig. Tech. Papers, pp.182-183, February 1995.

[22] F. Abu-Nofal, R. Avra, K.Bhabuthmal, R. Bhamidipaty, G. Blanck, A. Charnas, P. DelVecchio, J. Grass, J. Grinberg, N. Hayes, G. Haber, J. Hunt, G. Kizhepat, A. Malamy, A. Marston, K. Mehta, S. Nanda, H. V. Nguyen, R. Patel, A. Ray, J. Reaves, A. Rogers, S. Rusu, T. Shay, I. Sidharta, T. Tham, P. Tong, R. Trauben, A. Wong, D. Yee, N. Maan, D. Steiss and L. Youngs, "A Three-Million-Transistor Microprocessor," Dig. Tech. Papers of ISSCC '92, pp. 108-109, Feb. 1992.

[23] E. DeLano, W. Walker, J. Yetter and M. Forsyth, "A High Speed Superscalar PA-RISC Processor," Digest of Papers, IEEE Proc. COMPCON '92, pp.116-121, 1992.

[24] N. K. Yeung, Y.-H. Sutu, T. Y.-F. Su, E. T. Pak, C.-C. Chao, S. Akki, D. D. Yau and R. Lodenquai, "The Design of a 55SPECint92 RISC Processor under 2W," Dig. Tech. Papers of ISSCC '94, pp. 206-207, Feb. 1994.

[25] D. Pham, M. Alexander, A. Arizpe, B. Burgess, C. Dietz, L. Eisen, R. El-Kareh, J. Eno, S. Gary, G. Gerosa, B. Goins, J. Golab, R. Golla, R. Harris, B. Ho, Y.-w. Ho, K. Hoover, C. Hunter, P. Ippolito, R. Jessani, J. Kahle, K. Kishore, B. Kuttanna, S. Litch, S. Mallick, T. Ngo, D. Ogden, C. Olson, S.-H. Park, R. Patel, M. Pham, J. Prado, S. Reeve, R. Reininger, H. Sanchez, M. Shiffli, J. Slaton, G. Thuraisingham, K. Torku, C. Tran, N. Vanderchaaf, P. Voldstad and G. R. Zenhari, "A 3.0W 75SPECint92 Superscalar RISC Processor," Dig. Tech. Papers of ISSCC '94, pp. 212-213, Feb. 1994.

[26] P. E. Gronowshi, P. J. Bannon, M. S. Bertone, R. P Blake-Campos, G. A. Bouchard, W. J. Bowhill, D. A. Carlson, R. W. Castelino, D. R. Donchin, R. M. Fromm, M. K. Gowan, A. K. Jain, B. J. Loughlin, S. Mehta, J. E. Meyer, R. O. Mueller, A.Olesin, T. N. Pham, R. P. Preston and P. I. Rubinfeld, "A433MHz 64b Quad-Issue RISC Microprocessor," 1996 ISSCC Dig. Tech. Papers, pp.222-223, February 1996.

[27] A. K. Jain, R. P. Preston, P. J. Bannon, M. S. Bertone, R. P Blake-Campos, G. A. Bouchard, D. S. Brasili, D. A. Carlson, R. W. Castelino, K. M. Clark, S. Kobayashi, B. P. Lilly, S. Mehta, B. S. Miller, R. O. Mueller, A. Olesin and Y. Saito, "1.38cm^2 550Mhz Microprocessor with Multimedia Extensions," 1997 ISSCC Dig. Tech. Papers, pp.174-175, February 1997.

[28] S. Hiker, N. Phan and D. Rainey, "A 3.4ns 0.8μm BiCMOS 53 x 53b Multiplier Tree," Dig. Tech. Papers of 1994 ISSCC, pp.292-293, Feb. 1994.

[29] J. Mori, M. Nagamatsu, M. Hirano, S. Tanaka, M. Noda, Y. Toyoshima, K. Hashimoto, H. Hayashida and K. Maeguchi, "A 10-ns 54 x 54-b Parallel Structured Full Array Multiplier with 0.5-μm CMOS Technology," IEEE J. Solid-State Circuits, vol. 26, No.4, pp.600-605, Apr. 1991.

[30] G. Goto, T. Sato, M. Nakajima and T. Sukemura, "A 54 x 54-b Regularly Structured Tree Multiplier," IEEE J. Solid-State Circuits, vol. 27, No.9, pp. 1229-1235, Sep. 1992.

[31] N. Okubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, Y. Nakagome, "A 4.4-ns CMOS 54x54-bit Multiplier Using Pass-transistor Multiplier," IEEE Proc. 1994 CICC, pp. 599-602, May 1994.

[32] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki and Y. Nakagome, "A 4.4 ns CMOS 54 x 54-b Multiplier Using Pass-Transistor Multiplexer," IEEE J. Solid-state Circuits, Vol.30, No.3, pp. 251-257, March 1995.

[33] W. M. Newman and R. F. Sproull, "Principles of Interactive Computer Graphics : 2nd. Edition," McGraw-Hill Book Company, 1979.

[34] D. F. Rogers, "Procedural Elements for Computer Graphics," McGraw-Hill Book Company, 1985.

[35] J. D. Foley, A. V. Dam, S. K. Feiner and J. F. Hughes, "Computer Graphics: Principles and Practice, Second Edition," Addison-Wesley Publishing Company, 1993.

[36] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase and K. Mashiko," A 286MHz 64-bit Floating Point Multiplier with Enhanced CG Operation," Dig. of Symposium on VLSI Circuit, pp. 15-16, June 1995.

[37] 牧野、鈴木、森中、中瀬、益子、角、「CGに適した機能を有する286MHz、64ビット浮動小数点乗算器」、電子情報通信学会技術研究報告、ED95-99, pp.13-20, 1995.

[38] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, K. Mashiko and T. Sumi, "A 286 MHz 64-b Floating Point Multiplier with Enhanced CG Operation," IEEE J. Solid-State Circuits, vol. vol. 31, No.4, pp. 504-513, Apr. 1996.

[39] J. Shipnes, "Graphics Processing with the 88110 RISC Microprocessor," Digest of Papers, IEEE Proc. COMPCON '92, pp.169-174, 1992.

[40] H. Makino, Y. Nakase and H. Shinohara, "A 8.8-ns 54 x 54-bit Multiplier Using New Redundant Binary Architecture," IEEE Proc. of 1993 ICCD., pp.202-205, Oct. 1993.

[41] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, H. Shinohara and K. Mashiko, T. Sumi and Y. Horiba, "A Design of High-Speed 4-2 Compressor for Fast Multiplier," IEICE Transactions on Electronics, Vol.E79-C, No.4, pp.538-548, Apr. 1996.

[41] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara and K. Mashiko, "An 8.8-ns 54×54-bit Multiplier with High Speed Redundant Binary Architecture,"IEEE J. Solid-State Circuits, Vol.31, No.6, pp.773-783, Jun. 1996.

[42] A. D. Booth, "A Signed Binary Multiplication Technique," Q. J. Mech., Appl. Math. 4 pp.236-240, 1951.

[43] C. S. Wallace, "A Suggestion for Fast Multiplier," IEEE Trans. Electron. Comput., vol. EC-13, pp. 14-17, Feb. 1964.

第6章　結論

6．1　総括

本論文においては、3次元CG等への用途に適した100MHzを超える高速浮動小数点プロセッサを実現することを目的として筆者らが行った、加算器、乗算器、浮動小数点加算器および浮動小数点乗算器の高性能化に関する研究内容について述べた。以下に第2章から第5章における研究内容ならびに得られた結果を総括する。

第2章においては、高速のデータパスを実現する上で最も重要な構成要素である加算器の高速化および小面積化に関する研究内容について述べた。まず、種々の加算器アーキテクチャから代表的な5種類のアーキテクチャを抽出し、遅延時間および面積の比較を行った。比較に当たっては、加算器を構成する基本ゲート回路に対して実際のレイアウトに基づいてそれぞれの遅延時間と面積を算出し、この遅延時間と面積の値を用いて、各アーキテクチャによる加算器の遅延時間と面積の式をビット数の関数として求めた。その結果、最適アーキテクチャとして、キャリールックアヘッド方式とキャリーセレクト方式を組み合わせたもの選定した。さらに、得られた式から遅延時間の解析を行い、ビット数の増加および半導体の微細化の進歩によって、ゲート自体の遅延時間よりも配線容量や次段のゲート容量による遅延の割合が増大することを示した。次に、得られた最適アーキテクチャによる64ビット加算器の設計を行った。構成は下位56ビットをキャリールックアヘッド回路とし、上位8ビットをキャリーセレクト回路とした。キャリールックアヘッド回路においてはバイナリルックアヘッド方式によってキャリー生成の高速化を図るとともに、キャリーセレクト回路としては従来の回路を改良して素子数を削減したMCS（Modified Carry Select）回路を提案し、キャリーセレクト部の面積を20%削減した。設計した64ビット加算器を0.5μm CMOS、3層配線プロセスで試作した。トランジスタ数は3044個で、チップ面積は1305×207μm^{2}であった。さらにテストを行うことにより、加算時間2.6nsの動作を確認した。得られた加算器は従来のものと比べて最も小面積であり、速度においても200MHz動作が十分に可能な高速性を実現している。また、今後ともプロセスや電源電圧の変化により最適な加算器のアーキテクチャは変化してゆくと考えられるが、第2章で行った最適化はどんなプロセスや電源電圧が主流になろうとも有効であり、最適な加算器アーキテクチャを得ることができる。

第3章においては、高速のデータパスを実現する上で加算器と並んで重要な構成要素である乗算器の高速化および小面積化に関する研究内容を述べた。まず、高速のクリティカルパスを実現するために各種ゲートの遅延時間を比較し、またTGの性質を分析することにより、クリティカルパスの設計ルールとして以下の二つを定めた。

・クリティカルパスはインバータ、2入力NANDおよびTGの3種類のゲートのみで構成する。

・TGの直列接続は2段以内とする。

次に冗長二進数を用いた高速乗算アーキテクチャに関する研究を行い、高速かつ小面積を実現するアーキテクチャを提案した。まず二つの部分積の一方を反転してペアとする冗長二進部分積の生成法を提案し、これによってハードウエアおよび遅延時間び増加を伴わずに冗長二進部分積の生成を可能とした。次に冗長二進加算器に関して、従来回路における問題点を明らかにするとともに、プール式をCMOS回路に最適化することにより、6種類の高速冗長二進加算器HSRBA1～ HSRBA1を導出し、これによって従来よりも14%高速の冗長二進加算器を実現した。さらに、論理構造を解析することにより、従来の4-2コンプレッサがいずれもHSRBA1～ HSRBA1のうちのどれかと同一の論理構造を有することを明らかにした。次に、冗長二進→二進変換器に関して、パストランジスタのみで構成された高速キャリー伝搬回路を提案し、これによって従来の冗長二進→二進変換器およびCLA加算器よりも高速でトランジスタ数の少ない変換器を実現した。上で定めたクリティカルパス設計ルールと提案した冗長二進アーキテクチャに基づいて54×54ビット乗算器の設計を行った。トランジスタ数は78,800でこれは従来のものに比べて世界最少である。設計した54×54ビット乗算器を0.5μm CMOS、3層配線プロセスで試作した。チップ面積は3.05×3.08mm^{2}で、0.5μmCMOSプロセスのものでは世界最小である。評価の結果、電源電圧3.3Vで遅延時間8.8nsが得られた。この値は従来に比べて12%速く、世界最高速の値である。このように、冗長二進乗算器のアーキテクチャをプール式に基づいてCMOS回路に最適化することにより従来に比べて高速でトランジスタ数の少ない乗算器を実現した。

第4章では、高速浮動小数点プロセッサに必要不可欠な構成要素である浮動小数点加算器の高速化に関する研究内容を述べた。まず、浮動小数点加算器におけるクリティカルパスを調べるために浮動小数点加算の手順を分析し、その結果「桁落ち」に対する処理が重大なクリティカルパスとなっており、これを改善することによって高速化が実現できるという結論を得た。そこで、入力オペランドから仮数部加算結果の先頭に並ぶ「0」の数を直接予測するLZA機能に関する研究を行い、従来よりも高速でハードウエア量の少ないLZA論理を提案した。この方式によって、従来のLZA方式と比較して遅延時間を0.9ns短縮し、トランジスタ数を3,363個低減することができた。提案したLZA論理を用いた64ビット浮動小数点加算器の設計を行った。搭載した機能はIEEE-754準拠による単精度および倍精度の加算、減算および比較機能、単精度と倍精度間の相互の変換機能および浮動小数点数と整数との相互の変換機能で、動作周波数を上げて演算のスループットを高めるために、5段のパイプライン分割とした。トランジスタ数は約54,000で、そのうちLZA処理系回路のトランジスタ数は884であり全体の1.6%と小さい。レイアウト設計は一部CADによる自動ツールを用いることによって設計期間を短縮した。設計した64ビット浮動小数点加算器を0.5μm CMOS、3層配線プロセスで試作した。チップサイズは2.5×3.5mm^{2}であった。スキャンパスによるテストの結果、電源電圧3.3Vの下でサイクルタイム

6.1nsの動作が確認された。これは、動作周波数に換算すると164MHzに相当する。第4章ではさらに、得られた64ビット浮動小数点加算器に第2章の成果を適用した場合の効果について考察を行った。その結果、第2章の加算器を用いることにより、64ビット浮動小数点加算器のサイクルタイムを5.2nsに短縮することができ、これによって動作周波数を192MHzまで高めることができるという結論を得た。ここで得られた性能は、CADツールによる自動レイアウトを一部用いているにも関わらず、人手によって最適化された高速浮動小数点演算器と同等のものであり、提案したLZA方式が浮動小数点加算器の高速化に有効であることが確認された。

第5章では、浮動小数点加算器と並ぶ高速浮動小数点プロセッサの主要構成要素として、64ビット浮動小数点乗算器に関する研究内容を述べた。まず、CGに対する性能を向上させるために、ピクセルデータと浮動小数点数とを直接乗算する「CG乗算機能」を提案した。この機能は、CGにおける様々な計算に用いることができ、CG機能を持たない場合の7倍以上、また従来のCG用命令を用いた場合の4倍以上の性能向上が可能である。次に、CG乗算機能を浮動小数点乗算器に有効にインプリメントする方法に関する検討を行い、ピクセルデータの各フィールド間に8ビットの「0」データを挿入することにより、従来の浮動小数点乗算器のハードウエアをほとんど変更せずにCG乗算機能をインプリメントするアルゴリズムを提案した。これによって、CG乗算機能の搭載に伴うトランジスタ数の増加量を全体の4%に抑えた。以上の提案に基づき、CG乗算機能を搭載した64ビット浮動小数点乗算器を設計した。搭載した機能は、IEEE-754準拠の単精度および倍精度浮動小数点乗算機能とCG乗算機能で、CG乗算機能においては単精度および倍精度の浮動小数点数と32ビットピクセルデータとの乗算を可能とした。サイクルタイムを低減して動作周波数を高めるために5段のパイプライン構成とし、仮数部乗算には、第3章の研究成果である高速冗長二進アーキテクチャによる56×53ビット整数乗算器を用いた。また、レイアウト設計は一部CADによる自動ツールを用いることによって設計期間を短縮した。トランジスタ数は全部で138,000で、このうちCG乗算機能の追加による増加分が5,500（4%）と小さい。設計した64ビット浮動小数点乗算器を0.5μm CMOS 、3層配線プロセス技術で試作した。チップ面積は4.2×5.1mm^{2}であった。さらに、スキャンパスによるテストの結果、電源電圧3.3Vの下でサイクルタイム3.5nsの動作が確認された。これは、動作周波数に換算すると286MHzに相当する。このように、CG乗算機能を搭載したにもかかわらず、浮動小数点乗算器単体としては最高速の動作周波数が得られ、提案したアーキテクチャが浮動小数点乗算器の高速化に有効であることが確認された。

6．2　VLSI浮動小数点プロセッサの性能予測

本節では第2章から第5章において得られた整数加算器、整数乗算器、浮動小数点加算器および浮動小数点乗算器を浮動小数点機能を持つVLSIプロセッサに適用した場合の性能予測について述べる。

図6－1に構築可能なプロセッサシステムの構成図を示す。この図は第1章で述べた基本構成を示す図1－2におけるプロセッサの部分に相当するが、浮動小数点処理をハードウエアで実行可能とするためにマイクロプロセッサのデータパスが整数データパスと浮動小数点データパスに分けられている。整数データパスには第2章で得られた64ビット加算器および第3章で得られた64ビット乗算器が用いられ、浮動小数点データパスには第4章で得られた64ビット浮動小数点加算器および第5章で得られた64ビット浮動小数点乗算器が用いられる。予測する各演算器の構成および性能は以下の通りである。

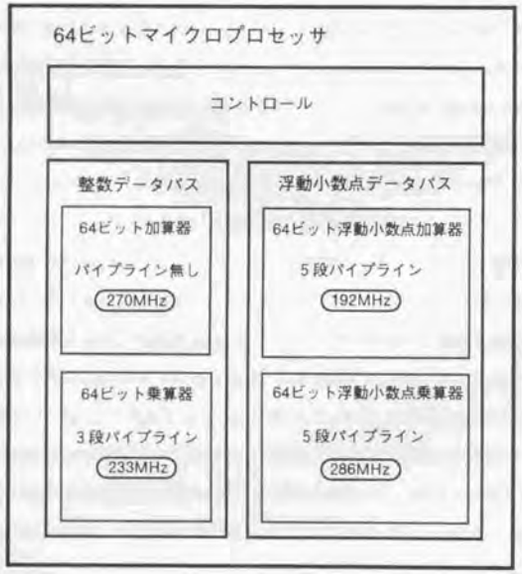


図6－1　研究成果を用いた浮動小数点計算プロセッサの構成

(1) 64ビット整数加算器

第2章で開発した2.6nsの64ビット整数加算器をそのまま使用する。ただし、入出力部にはレジスタが挿入されるため、レジスタの遅延が加算される。レジスタの遅延時間としては、第4章の64ビット浮動小数点加算器の評価の際に得られた値が1.1nsであったのでこの値を用いることができる。したがって、遅延時間の合計は2.6+1.2=3.7nsとなる。これは動作周波数に換算すると270MHzに相当する。

(2) 64ビット整数乗算器

第3章で開発した8.8nsの54×54ビット乗算器を64×64ビットに拡張して用いる。この拡張に際して、冗長二進部分積発生回路は、クリティカルパスの回路段数が変化しないので遅延時間は54×54ビットのものと同じである。また、冗長二進部分積の数は16個になるので、ワレストリーによる部分積足し上げ部は、54×54ビットの際と同様に冗長二進加算器4段で実行でき、この部分の遅延時間も54×54ビットのものと同じとなる。これに対して、冗長二進→二進変換部は、54×54ビットの際は80ビットのキャリー伝搬回路が必要であったのに対して、64×64ビットの場合は20ビット拡張されて100ビットのキャリー伝搬回路が必要とされる。キャリー伝搬回路は図3－13に示した構成となるため、この20ビット分の拡張によって、クリティカルパスはマルチプレクサ回路2段分長くなる。マルチプレクサ1段の遅延時間は150psであったため、遅延時間の増加は0.3nsである。したがって64×64ビット乗算器の遅延時間は8.8+0.3=9.1nsとなる。

この64×64ビット乗算器をVLSIプロセッサに搭載する際には、高速化のために以下の3段のパイプラインに分割する。

・1段目：冗長二進部分積発生部とワレストリーの1段目まで
・2段目：ワレストリーの2段目から4段目まで
・3段目：冗長二進→二進変換部

図3－15に54×54ビット乗算器の遅延時間の内訳を示したが、これに基づいて考えると、ワレストリーにおける冗長二進加算器の初段の遅延時間は0.9nsであったので、パイプラインの1段目の遅延時間は2.3+0.9=3.2ns、2段目の遅延時間は3.8−0.9=2.9nsおよび3段目の遅延時間は2.7+0.3=3.0nsとなる。したがって1段目の3.2nsがクリティカルなパイプラインとなり、この部分の遅延にパイプラインレジスタの遅延時間1.1nsを加えると、合計の遅延時間は4.3nsとなる。これは動作周波数に換算すると233MHzに相当する。

(3) 64ビット浮動小数点加算器

第4章で得られた動作周波数164MHzの64ビット浮動小数点加算器を適用するが、4．4節で行っ

た考察に基づいて第２章の成果を適用することとする。したがって，動作周波数は４．４節で述べたように 192MHz となる。

(4) 64 ビット浮動小数点乗算器

第５章で得られた動作周波数 286MHz の 64 ビット浮動小数点乗算器をそのまま使用する。

以上の演算器によって整数データパスおよび浮動小数点データパスを構成すると、最も遅くクリティカルとなる演算器は 64 ビット浮動小数点加算器の 192MHz である。したがってこの VLSI プロセッサの動作周波数は 192MHz となる。192MHz で動作させることにより、最大整数演算性能 192MIPS（Mega Instruction Per Second）、最大浮動小数点演算性能 192MFLOPS（Mega Floating Point Operation Per Second）が達成できる。また、整数処理と浮動小数点処理を同時実行可能とすることによって、最大２倍の 384MIPS が達成可能である。さらに、64 ビットマイクロプロセッサを並列に複数個配置し、いわゆるマルチプロセッサ構成で並列処理することによって、さらに性能を高めることができる。たとえば、メインメモリを共有する「メモリ共有型マルチプロセッサ」構成にすると、プロセッサ数が 30 程度までほぼリニアに性能を高めることができる[1]ので、例えば３プロセッサで約 1152MIPS、20 プロセッサで 7680MIPS の高性能を得ることができる。

このように、主としてアーキテクチャと回路の工夫により、目標としていた 100MHz を大きく上回り約２倍の性能を達成することが可能な演算器を実現することができた。第４章において述べたように、さらに人手をかけることによって、クリティカルとなっている 64 ビット浮動小数点加算器の性能をさらに向上させることが可能であり、200MHz を超える高性能を達成することも可能である。

６．３　残された問題と将来の展望

以上のように、本研究を通じて 10 年余りで約 20 倍の高速化を実現する演算器技術を得ることができたが、このような急激な性能の進歩によって、これまで性能を律速していた演算器以外の部分がクリティカルパスとなり始め、現在でも既に新たな問題点が顕在化し始めている。筆者らの研究中あるいはその後において問題となっているものとして以下のものが挙げられる。

第１はメモリのアクセスタイムである。コンピュータシステムにおいて、マイクロプロセッサは常にメモリとのデータのやりとりを行うため、プロセッサがメモリ内のデータをアクセスするのに必要とする時間が長いと、この部分がボトルネックとなり全体の性能を劣化させてしまう。近年ではメモリのアクセスタイムを低減するために、メモリとプロセッサとの間に比較的小規模で高速なキャッシュメモリを置く、いわゆる「メモリの階層化」が行われている[1]。したがってコンピュータの性能を向上させるためには、演算器の高速化によってデータパスを高性能化することと並んでキャッシュメモリを高速化することが重要である。筆者らが研究を開始した 1983 年当時からこの点は認識されており、これを解決するために筆者らは基板材料としてガリウム砒素（GaAs）を用いた超高速メモリに関する研究を行った。この研究の詳細を本論文の付録において述べる。この研究によってアクセスタイム 5ns の実用可能な 16K ビット SRAM（Static Random Access Memory）を得ることができた。本研究で得られた演算器技術とこの GaAs SRAM とを用いて構築可能なコンピュータシステムを図６－２に示す。この図は第１章で述べた基本構成図１－２に基づいて描かれているが、マイクロプロセッサとメインメモリおよび I/O との間にキャッシュメモリユニットが接続されている。キャッシュメモリには GaAs 16K ビット SRAM が必要個数用いられる。例えばキャッシュメモリサイズが 16K バイトのときは、8 個の GaAs 16K ビット SRAM が用いられる。64 ビットマイクロプロセッサとキャッシュメモリユニットとは高速バスで接続され、キャッシュメモリユニットとメインメモリおよびインプット／アウトプットとの間は、比較的低速のバス（メモリ－I/O バス）で接続される。動作スピードに関しては、64 ビットマイクロプロセッサが最大動作周波数 192MHz であるが、GaAs 16K ビット SRAM のアクセスタイムが 5ns なので、マイクロプロセッサとキャッシュメモリユニットとを結ぶ高速バスの動作周波数は最大 200MHz となる。キャッシュメモリユニットと DRAM（Dynamic Random Access Memory）からなるメインメモリおよびインプット／アウトプットとを結ぶメモリ－I/O バスの速度は 192MHz よりも遅くても良い。このように、得られた演算器と GaAs 16K ビット SRAM を適用することによって最大 192MHz のコンピュータシステムを実現することができる。しかしながら、GaAs によるキャッシュメモリはシリコンよりも高速化が可能である反面、消費電力の面で 16K ビット以上の高集積化が困難であるという問題がある。これは付録で述べたリーク電流低減の工夫にもかかわらず、シリコンに比べて依然としてリーク電流が大きいことによる。さらに、GaAs メモリは基板材料がシリコンと異なるた

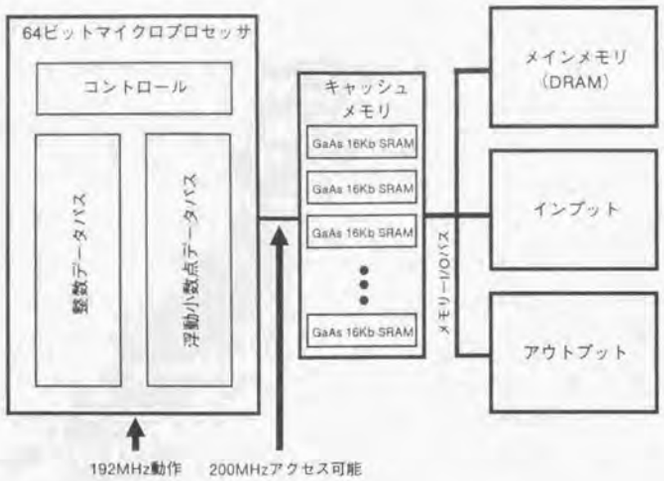


図６－２　研究成果を用いて構築可能なコンピュータシステムの構成

めに、シリコンによるVLSIマイクロプロセッサ上に取り込むことができないという問題もある。近年、半導体の微細化技術が進歩して１チップ上に搭載可能なトランジスタ数が増加しており、これに伴って汎用のマイクロプロセッサにはシリコンSRAMによる数十Kビットレベルのキャッシュメモリが搭載され始めている。マイクロプロセッサ上にキャッシュメモリを搭載することによって、チップ外のキャッシュメモリにアクセスする場合と比べて高速のアクセスが可能となるため、これによって現在では200MHzを超える高速のキャッシュアクセスが実現されており、キャッシュメモリの高速化はシリコンでも十分に実現可能であることが実証されている。このように、キャッシュメモリに関しては、現在では高速化と高集積化の両面においてシリコンが適しており、今後ともシリコンが主流となると考えられる。

第２の問題点としてクロックのスキューが挙げられる。パイプラインを用いた同期式のマイクロプロセッサにおいては、クロック信号をチップ全体に分配して、これでパイプラインレジスタを動作させることによってチップ全体を動作させるが、チップ内でのクロック信号の遅延によって、チップ内の各部分でクロックのずれによるスキューが発生する。このスキューは、サイクルタイムにオフセットとなって付加され、動作周波数を劣化させる。しかも、この劣化は動作周波数が高くなるほど、より顕著になる。したがって、高速のマイクロプロセッサを実現するためには、チップ内でクロックの遅延時間をそろえてスキューの低減を図るための注意深い設計が要求される。本研究において全体の

クリティカルパスとなっている64ビット浮動小数点加算器では、パイプラインレジスタの遅延時間が1.1nsであったが、これはレジスタ回路自体の遅延時間と比較すると約2倍の値となっており、パイプラインレジスタの遅延時間の約半分が、クロックスキューに起因するものであると考えられる。64ビット浮動小数点加算器の設計においては、特にクロックスキューに対する考慮は行われていなかったため、今後更なる高速化を実現するためには、スキューを低減するための設計が必要である。また、クロックスキューの問題は、演算器に限らずマイクロプロセッサ上の全ての回路に共通する問題であり、高速化の上で今後ますます重要となると考えられる。スキューを低減するためにシミュレーションによってチップ内のスキューの様子を調べる手法はDEC社のアルファチップにおいて既に行われている[2]。また、さらに将来の方向として、グローバルなクロック信号を用いないマイクロパイプライン方式[3]やウエーブパイプライン方式[4]などによる非同期設計手法が提案されており、今後高速演算器の設計にあたり、これらの技術も検討していく必要がある。

第３の問題点としてVLSIチップ間を結ぶI/O信号の速度が挙げられる。本研究で述べたようにチップ内の動作周波数はアーキテクチャや回路を工夫することによってある程度上げることができるが、チップ外との信号のやりとりは容易に高速化することができない。チップ外にはパッケージやプリント基板による大きな寄生容量が存在し、これを駆動するためにはチップ内の出力回路に大きなサイズのトランジスタを用いる必要がある。しかしながら、出力回路を大きくして大電流を流すと、電流値の時間に対する変化率が大きくなり、配線のインダクタンスによる電磁誘導ノイズが問題となる。このノイズは出力波形を著しく劣化させるため、ノイズが大きい場合には動作不能となってしまう。また、インダクタンスとしては、パッケージとチップを接続するワイヤボンディングとパッケージ内の配線によるものが支配的とであり、これは半導体の微細化によっても改善されない。すなわち、出力回路の電流値には限界があり、それ以上に電流を流して高速化することができない。この問題は動作周波数が100MHzを超えた頃から既に問題となり始めており、現在ではVLSIの高性能化上重大な課題となっている。これを改善する方法としては、入出力信号の電圧振幅を低下させることによって電流値を少なく保ったまま信号の遷移時間を短縮することが有効であり、GTL（Gunning Transceiver Logic）およびCTT（Center Tapped Termination）等を始めとする多くの小振幅の電圧レベルが提案され、実用され始めている[5]。今後は、チップ内の動作周波数の向上に際して、このような入出力信号レベルの最適化も同様に考慮していく必要がある。

第４の問題点として消費電力の増大が挙げられる。一般に動作周波数が向上するとこれにほぼ比例して消費電力が増大する。消費電力が増大すると、発熱による動作劣化を引き起こすため、チップを実装する際に高価なパッケージが必要になったり、あるいは使用時に送風が必要になるなど、システム構築に必要なコストが増大する。近年、ハイエンドのコンピュータにおいて動作周波数が200MHz

と高速化されているが、これに伴って消費電力の問題が既に顕在化している。またこれとは別に、最近の携帯機器の発展とともに、電池を少しでも長持ちさせるための技術が重要となっており、あまり高速化が要求されない分野においても低消費電力化が要求され始めている。このように、ハイエンドとローエンドの両面において低消費電力化が重要な技術となっているが、基本的には半導体を微細化することによってチップサイズが小さくなり寄生容量が低減されるため、低消費電力化を実現することができる。しかし、実際にはチップサイズが小さくなると、余った領域に回路を作って高機能化を図るということが一般に行われるため、現実にはあまり消費電力は低減されない。したがって、低消費電力化を実現するためには微細化に加えて消費電力を低減するための技術が必要とされる。消費電力を低減するためには、チップ内の使用しない回路を止めることによって無駄な信号遷移による電力消費を防ぐことが有効である。これは、クロックをチップ内の回路ブロック毎に系統分けして、命令の種類によって使用しないブロックのクロックを止めることによって実現することができ、既にいくつかのVLSIプロセッサに適用されている[6]。さらに、最近ではトランジスタのしきい値電圧を低下させて、回路を1.0～1.5V程度の低電圧で動作させることによって、高速性を維持しつつ従来よりも1桁以上低消費電力化した研究が報告されている[7]。ただし、しきい値電圧を低下させるとトランジスタのOFF時のリーク電流が大きくなるという問題があり、これに対する対策も報告されている[8]。また、デバイス面からのアプローチとしてSOI（Silicon on Insulator）技術による低消費電力化も注目されている[9]。SOIはシリコン基板上に酸化膜を形成し、さらにその上にトランジスタを形成するもので、トランジスタのソースおよびドレインの寄生容量が従来よりも飛躍的に低減されるため低消費電力化が可能となる。以上のように、低消費電力化には様々な手法が検討されており、今後はこれらの技術が組み合わされて用いられていくと考えられる。

以上述べた問題点は、演算器に限らず全てのLSIに共通するものであるが、演算器の設計に関しても、さらなる高速化が図られていくと考えられる。演算器のアーキテクチャによる高速化に関しては、本研究を含めてはほぼ研究段階は終わったと考えられ、今後アーキテクチャ面での大きな進展はないであろう。これからは、主に回路面での工夫とプロセス技術に適した回路の最適化が中心となると考えられる。今後の有力な回路設計技術としてはダイナミック回路の使用による高速化が挙げられる。ダイナミック回路は、従来のスタティック型の回路動作と異なり、回路の信号ノードにある寄生容量（場合によっては容量を作ることもある）に電荷を蓄え、その後電源とそのノードを遮断することによって信号の電位を保持するもので、信号遷移時の貫通電流が流れないために、高速化と低消費電力化の両方を実現することができる。現在、マイクロプロセッサにおいては、これがパイプラインレジスタに一部適用されており、また加算器や乗算器全体をダイナミック回路によって構成する研究も報告されている[10], [11]。ただし、ダイナミック回路は時間の経過とともに信号の情報が失われるため、

逆に低速での動作ができなくなるという問題があり、今後ハイエンドのプロセッサ向け演算器を中心に適用が進められていくと考えられる。また、この他に上で述べたウエーブパイプライン方式による高速化などもパイプラインレジスタが不要になり高速化が可能となるため、今後研究が進められていくと考えられる。プロセス技術に適した回路の最適化に関しては、本論文の第2章および第3章で述べた手法が今後とも有効であり、高速の演算器を得るために用いられると考えられる。

今後半導体の微細化の進展と相まって、以上のような回路方式および最適化技術が適用されることにより、動作周波数は1GHz程度まで高められると考えられる。ただし、そのような高周波で動作するようになると回路設計はデジタル設計よりもむしろアナログ設計に近いものとなり、今後アナログ的な精密な設計手法がより重要性を増すであろう。演算器の動作周波数が1GHzよりもさらに高められるかどうかは、このようなアナログ的な高精度の設計が大規模回路に対して適用できるかどうかにかかっており、今後のCADシステムを中心とする設計手法の進歩が期待される。

また、本研究では加算器および乗算器の主な用途として3次元CGを想定しているが、今後CGは益々発展し、更なる高速化と画質の高品質化が要求され続けていくと考えられる。また、現在CGにおいては主に物体表面のみを扱っているが、医療分野などを中心として物体の中身までを含めて表示しようとするいわゆる「ソリッドモデル」が注目されている[12]。これは、従来のCGよりもはるかに多量の浮動小数点演算が要求されるため、演算器の高速化への要求は益々強くなるものと考えられる。

本論文で述べた研究内容は、今後の製造プロセス技術の進歩、動作速度の進展および用途の変化に対して常に適用可能なものであり、高性能の加算器、乗算器、浮動小数点加算器および浮動小数点乗算器を得る上で有効なものであると考えられる。

＜参考文献＞

[1] J. L. Hennessy and D. A. Patterson, "Computer Architecture : A Quantitative Approach," Morgan Kaufman Publishers, 1990.

[2] D. Dobberpuhl, R. Witek, R. Allmon, R. Anglin, D. Bertucci, S. Britton, L. Chao, R. Conrad, D. Dever, B. Gieseke, S. Hassoun, G. Hoeppner, K. Kuchler, M. Ladd, B. Leary, L. Madden, E. McLellan, D. Meyer, J. Montanaro, D. Priore, V. Rajagopalan, S. Samudrala and S. Santhanam, "A 200-MHz 64-b Dual-Issue CMOS Microprocessor" , IEEE J. Solid-State Circuits, vol. 27, No.11, pp. 1555-1567, Nov. 1992.

[3] I. E. Sutherland, "Micropipelines," Communication of the ACM, Vol.32, No.6, pp.720-738, June 1989.

[4] D. Ghosh and S. K. Nandy, "A 400MHz Wavw-Pipelined 8×8-bit Multiplier in CMOS Technology," IEEE Proc. of 1993 ICCD., pp.198-199, Oct. 1993.

[5] 山田、「インタフェースの高速化」、電子情報通信学会誌、Vol.76, No.7, pp.721-725, 1993年7月

[6] J. Montanaro, R. T. Witek, K. Anne, A. J. Black, E. M. Cooper, D. W. Dobberpuhl, P. M. Donahue, J. Eno, A. Farell, G. W. Hoeppner, D. Kruckemyer, T. H. Lee, P. Lin, L. Madden, D. Murray, M. Pearce, S. Santhanam, K. J. Snyder, R. Stephany and S. C. Thierauf, "A 160MHz 32b 0.5W CMOS RISC Microprocessor," Dig. Tech. Papers of ISSCC '92, pp. 214-215, Feb. 1996.

[7] A. P. Chandrakasan, S. Sheng and R. W. Brodersen, "Low-Power CMOS Digital Design" , IEEE J. Solid-State Circuits, vol. 27, No.4, pp. 473-484, April. 1992.

[8] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. S. and J. Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," IEEE J. Solid-State Circuits, vol. 30, No.8, pp. 847-854, August. 1995.

[9] Masayuki Ino, Hirotoshi Sawada, Kazuyoshi Nishimura, Masami Urano, Hiroki Suto, Shigeru Date, Takako Ishihara, Tadao Takeda, Yuichi Kado, Hiroshi Inokawa, Toshiaki Tsuchiya, Yutaka Sakakibara, Yoshinobu Arita, Katsutoshi Izumi, Ken Takeya and Tetsushi Sakai, "0.25μm CMOS/SIMOX Gate Array LSI," Dig. Tech. Papers of ISSCC '96, pp. 86-87, Feb. 1996.

[10] A. Inoue, Y. Kawabe, Y. Asada and S. Ando, "A 0.4μm 1.4ns 32b Dynamic Adder using Non-precharge Multiplexer and Reduced Precharge Voltage Technique," Dig. of Symposium on VLSI Circuit, pp. 9-10, June 1995.

[11] M. Hanawa, K. Kaneko, T. Kawashimo and H. Maruyama, "A 4.3ns 0.3μm CMOS 54×54b Multiplier Using Precharged Pass-Transistor Logic," Dig. Tech. Papers of ISSCC '96, pp. 364-365, Feb. 1996.

[12] J. D. Foley, A. V. Dam, S. K. Feiner and J. F. Hughes, "Computer Graphics: Principles and Practice, Second Edition," Addison-Wesley Publishing Company, 1993.

付録　キャッシュメモリに関する関連研究

付．1　研究の背景

本論文では、マイクロプロセッサにおける重大なボトルネックとして加算器および乗算器を挙げ、これらを高速化するための研究内容に関して述べてきたが、マイクロプロセッサを高速に動作させる上でのもう一つの重大なボトルネックとしてメモリのアクセススピードの問題がある。すなわち、マイクロプロセッサは命令およびデータをメモリから読み出してその後処理を実行するが、通常メモリは大容量化に適したダイナミック・ランダムアクセスメモリ（DRAM）で構成されており、データのアクセスには数十～百 ns を要する[1]。したがってプロセッサが DRAM に直接アクセスする方法では、100MHz を超えるような高速な動作周波数で動作させることは不可能である。この問題を解決する方法として、キャッシュメモリを用いたメモリの階層化が行われている[2]-[4]。すなわち、メモリ（以後メインメモリと呼ぶ）とプロセッサとの間に比較的小規模で高速なキャッシュメモリを置き、そこにメインメモリのデータの一部をコピーしてアクセスすることにより、アクセス時間を高速化するというものである。一般にコンピュータにおいては、メモリのアクセスに時間的・空間的な局所性があり[2]-[4]、このためにメモリの階層化が有効となっている。キャッシュメモリとしては、通常スタティック・ランダムアクセスメモリ（SRAM）が用いられており、筆者の研究開始当時の 1980 年代前半においても 10～20ns の高速のアクセスタイムがシリコン（Si）によって達成されていた[5]。しかし、100MHz を超えるさらなる高速化を実現するためには従来の SRAM でも不十分であり、さらに高速のメモリを実現する必要があった。

従来よりも高速のメモリを実現する方法として基板材料にガリウム砒素（GaAs）を用いることが考えられる。GaAs は Si に比べ約 5 倍の電子移動度を有し、基板容量が小さく、しかも室温で動作が可能なことから[6]、Si を上回る高速 LSI を実現する可能性を有している。したがって、GaAs 技術をメモリに適用することによって Si では達成不可能な高速メモリを実現できる可能性がある。 GaAs このように有望な材料ではあったが、同時に次のような問題点もあった。すなわち、GaAs 素子はシリコンと類似してはいるものの、安定な酸化膜が形成できないために、Si のような MOS 構造を作ることができず、金属－半導体のショットキ接触を利用した MES（Metal-Semiconductor）構造からなる MESFET（Metal-Semiconductor Field Effect Transistor）が基本素子となる[6]。したがって、酸化膜を利用するダイナミック RAM を作ることができず、スタティック RAM しか形成できないため、高集積化よりも比較的集積度の低い高速のメモリに適した素子であると言える。また、MESFET においては、ゲートとソース・ドレイン間にゲートをアノードとする寄生ショットキダイオードが形成され、ゲート入力は MOS のように高インピーダンスとはならず、容易に電流が流れる構造となる。このような GaAs 特有の性質

のためにGaAsメモリの設計には独自の設計手法が要求される。また、動作マージンや信頼性などの面で未知の部分も多かった。したがって高速でかつ実用可能なメモリを得るためには、それら未知の部分を研究することによって問題点を解決していく必要があった。このような背景から、GaAsメモリの高速・高集積化と実用化に向けて数多くの研究が報告されていた[7]-[62]。しかし、筆者らの研究段階においては、いずれも未だ十分な成果は得られていなかった。

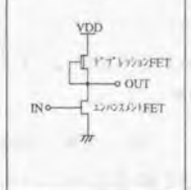
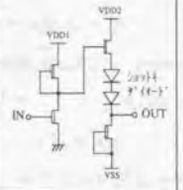
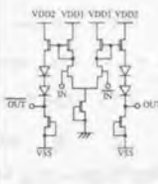
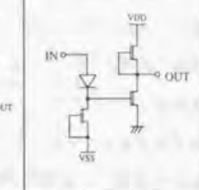
この付録においては、GaAs SRAMの高速・高集積化および高信頼性化のために行った研究内容について述べる[63]-[80]。まず付．2節において従来最もよく用いられるE/D型DCFL回路による高速設計手法とその評価結果について述べ、次に付．3節において問題点の抽出および解析を行う。付．4節において主として設計技術面からの解決策を述べ、付．5節においてこれらをGaAs 16Kビット SRAMに適用した結果を示す。

付．2　GaAs DCFL（Direct Coupled FET Logic）回路による高速化の検討

付．2．1　各種回路方式の比較・検討

GaAs素子による回路方式にはいくつかのものが提案されているが[31]，[84]-[94]、その中で代表的なものは、DCFL（Direct Coupled FET Logic）[82]，[83]，[86]，[90]、BFL（Buffered FET Logic）[81]，[82]，[85]-[87]，[89]、SCFL（Source Coupled FET Logic）[31]およびSDFL（Shottky Diode FET Logic）[82]，[85]，[91]の4つである。表付－1に4種類の回路の基本構成とその特徴を示す。GaAsの回路は、しきい値電圧が正の「エンハンスメントMESFET」としきい値電圧が負の「デプレッションMESFET」および「ショットキダイオード」から構成される。

表付－1　GaAs基本構成と特徴

名称	DCFL	BFL	SCFL	SDFL
基本回路				
長所	・1電源で構成できる ・素子数が少なく高集積化に有利 ・低消費電力 ・高速 ・負荷の電流が飽和するので電源電圧依存性が良い	・電圧振幅が広くとれる ・高速 ・構成素子数が比較的少ない ・しきい値電圧を1種類で構成できる	・電圧振幅が広くとれる ・差動型なのでしきい値電圧や入出力レベルのばらつきに強い ・出力に相補信号が得られる ・負荷の電流が飽和するので電源電圧依存性が良い	・電圧振幅が広くとれる ・ワイヤードORが構成できる ・構成素子数が比較的少ない ・しきい値電圧を1種類で構成できる
短所	・電圧振幅が狭くノイズマージンが小さい ・Lowレベルが浮くためにドライバFETの電流リークが大きい ・温度特性が悪い	・2電源以上を必要とする ・貫通電流が飽和しないため、電源電圧依存性が悪い ・消費電力が大きい	・2電源以上を必要とする ・構成素子数が多く高集積化には適さない ・消費電力が大きい ・低速	・2電源以上を必要とする ・消費電力が大きい ・あまり高速でない ・貫通電流が飽和しないため、電源電圧依存性が悪い
種類	ノーマリオフ型	ノーマリオン型	ノーマリオン型	ノーマリオン型

DCFL：Direct Coupled FET Logic
BFL：Buffered FET Logic
SCFL：Source Coupled FET Logic
SDFL：Shotky Diode FET Logic

DCFL回路は、シリコン技術におけるNMOS回路に相当するもので，正のしきい値電圧を持つエンハンスメント型MESFETのドライバ素子と負のしきい値を持つデプレッション型MESFETの負荷素子との1対からなるE/Dインバータを基本回路とするため、E/D型DCFL回路とも呼ばれる。これはGaAs

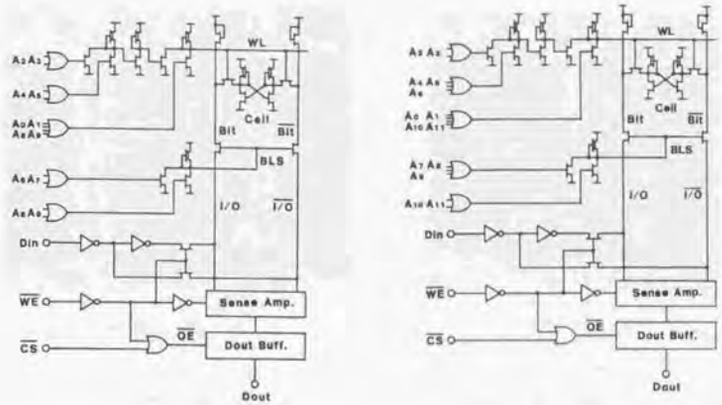
で構成可能な回路の中で最も単純で素子数が少なく、高速かつ低消費電力であるという長所を持つ。ただし、出力を次段の回路に接続した場合に、次段のドライバ素子のゲート・ソース間に寄生的に存在するショットキダイオードによってHighレベルが0.6V程度にクランプされてしまうため、信号の振幅が狭くノイズマージンや温度特性が悪いという短所がある。BFLは、2つのデプレッション型MESFETによるインバータの出力をバッファ回路によってレベルシフトし、Lowレベルを GNDレベル以下まで下げることによって次段への入力を可能としたものである。これは、電圧振幅を広く取ることができ、比較的高速でかつ1種類のしきい値電圧で構成できるという長所がある反面、2電源以上を必要とし消費電力が大きいという短所がある。SCFLは、ソースを共通とした1対の入力トランジスタに相補信号を入力し、その出力をバッファ回路によってレベルシフトして次段に入力する回路である。これは、電圧振幅が広く、出力に相補信号が得られるなどの長所があるが、やはり2電源以上を必要とし消費電力が大きく、しかも素子数が多く高集積化に適さないという短所がある。SDFLは入力部にショットキダイオードを用いたもので、多数の入力に対するダイオード出力をつなぐことによってOR回路を構成するワイヤードORが可能であるほか、電圧振幅を広く取ることができ、しきい値電圧も1種類で済むという長所がある。しかし、これも2電源以上を必要として消費電力が大きいという短所がある。また、DCFLのみは入力がLowレベルの際にDC電流が流れないためにノーマリオフ型の回路と呼ばれ、他の3つはデプレッション型MESFETによる貫通電流のパスが存在するため常にDC電流が流れ、このためにノーマリオン型の回路と呼ばれる。

本研究ではキャッシュメモリとして4K～16Kビット（1Kは1024ビットを表す）程度の規模のGaAs SRAMを想定しており、これを構成する場合 10^4～10^5程度のトランジスタ数が要求されるため、基本回路には単純さが要求される。上記の4つの中では、DCFLのみが1電源で構成でき、素子数が最も少なく、高速性・低消費電力性に優れ、LSI化に適している。したがって SRAMの基本回路としてDCFL回路を採用することとした。ただし、DCFL回路は電圧振幅がゲート・ソース間のショットキ電圧である0.6Vで制限されるため、他の3つの回路に比べてノイズマージンが小さく、設計の際には動作マージンの確保に注意する必要がある。

付．2．2　4Kビットおよび16Kビット SRAMの設計および試作

DCFL回路により、4Kビットおよび16Kビット SRAMの設計および試作を行った[63]-[67]。それぞれの回路構成を図付－1(a)および(b)に示す。 DCFL回路で集積回路を構成する場合、無駄な電力消費を避け、回路を高速かつ低消費電力で動作させるためには、回路の各出力部における次段のゲート数（ファンアウト数）の最適化を行う必要がある。そこで、シミュレーションによるファンアウト数の最適化を行い、最適値としてファンアウト数を4とした。したがって、それぞれの回路の各部分は全

てファンアウト4で設計されている。また、アドレス系回路の巨大化による消費電力の増加を抑えるために、アドレスの選択を複数段に分けて行うアドレスプリデコード方式を採用している。さらに電源電圧を最適化する検討を行い、電源電圧を0.7Vとした。



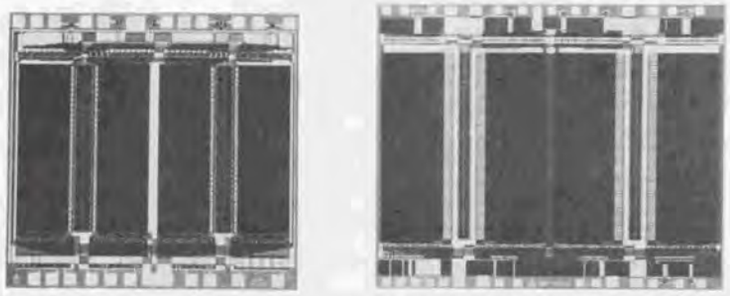
（a）4Kb SRAM　　（b）16Kb SRAM

図付－1　4Kビットおよび16Kビット SRAMの回路構成

試作に用いたFETは、ゲート長1.0μmのタングステン・シリサイド（WSix）耐熱性セルフアラインゲートFETで、しきい値電圧とそのばらつきはエンハンスメント型が0.10±0.05V、デプレッション型が-0.40±0.10Vである。配線は2層配線で、第2層金属がゲート金属及びソース・ドレインのオーミック金属と直接コンタクト可能としてチップサイズの低減を図っている。試作した4Kビットおよび16Kビット SRAMのチップ写真を図付－2(a)および(b)に示す。チップサイズはそれぞれ3.40×3.24mm^{2}および5.79×4.73mm^{2}である。

評価の結果、4Kビット SRAMでは電源電圧0.7Vで最小アクセスタイム2.5ns、消費電力200mWの性能が得られた。既発表のGaAs4Kビット SRAMと比較して同程度のアクセスタイムで消費電力が1/4～1/9程度に低減されている[22],[28],[30]-[32],[34]。これは、回路及び電源電圧の最適化によって無駄な電流を極力低減した効果を示している。これに対し16Kビット SRAMは、電源電圧0.7Vでは良好な

動作が得られず、電源電圧 1.0V で最小アクセスタイム 5.0ns、消費電力 1.0W の性能が得られた。消費電力は従来の 2/5 に低減されている[21]。図付－3 (a)および(b)にそれぞれのアドレス入力及びデータアウトの波形写真を示す。



（a） 4Kb SRAM　　（b） 16Kb SRAM

図付－2　4K ビットおよび 16K ビット SRAM のチップ写真

（a）4Kb SRAM　　（b）16Kb SRAM

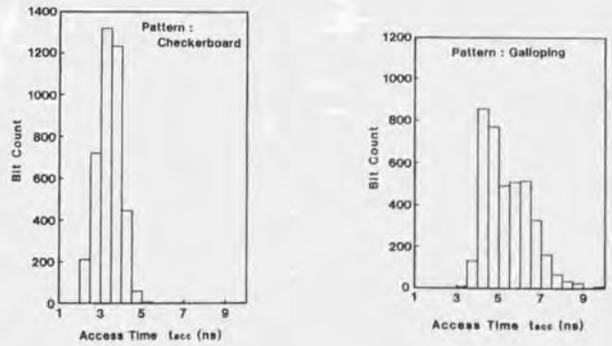
図付－3　4K ビットおよび 16K ビット SRAM の入出力波形

付．3　GaAs DCFL 回路における問題点の抽出と分析

以上のように、DCFL 回路の最適化により 4K ビットおよび 16K ビット SRAM の高速かつ低消費電力の動作が確認されたが、実際に使用する場合は最小アクセスタイムがいくら高速であっても、チップ内のアクセスタイムにばらつきがあればそのうちの最も遅いものがチップの性能を決定してしまう。また、温度特性や放射線に対する耐性など GaAs には未知の部分が多く、これらを解析することによって以下の問題点およびその要因を明らかにした。

付．3．1　アクセスタイムのばらつきとその要因

図付－4 に試作した 4 K ビット SRAM における 2 種類のテストパターンに対する室温でのアクセスタイムの分布を示す[68]-[70]。テストパターンは、(a)チェッカーボードおよび(b)ギャロッピングである。チェッカーボードパターンは、メモリ空間にアドレスの最下位から最上位まで順番に市松模様状に「0」と「1」のデータを書き込み次にこれを読み出すもので測定の条件としては最も緩いものの一つである。これに対してギャロッピングパターンは、あらゆるアドレスの遷移の組み合わせに対して「0」と「1」の順番を換えて読み出すもので、メモリのテストパターンとしては最も厳しいものである。チェッカーボードパターンの場合は、最小アクセスタイムが 2.5ns となっているが最大アクセスタイムは 5ns とほぼ 2 倍のばらつきがある。これに対してギャロッピングパターンでは最小アクセスタイムは 3.5ns と遅くなり、最大アクセスタイムは 10ns まで劣化してしまう。したがって、このメモリの実用上のアクセスタイムは 10ns となり、シリコンの SRAM と同程度になってしまう。

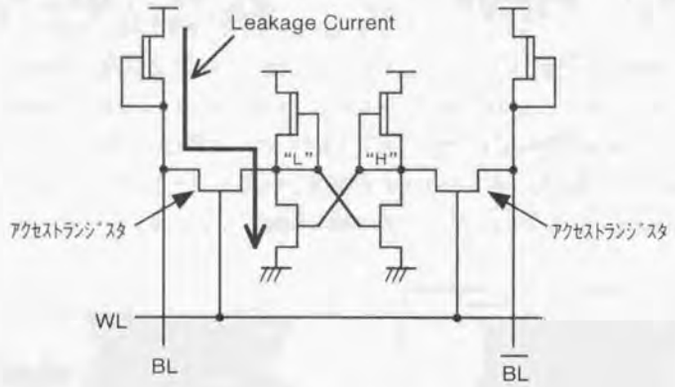


（a） チェッカーボード　　（b） ギャロッピング

図付－4　2 種類のテストパターンに対するアクセスタイムの分布

このようなアクセスタイムのばらつきの原因を解析した結果、以下の二つの要因が明らかになった。まず第1は、トランジスタ特性のばらつきである。上でも述べたようにDCFL回路は電圧振幅が0.6Vとシリコンの5Vあるいは3.3Vに比べて小さいにもかかわらず、トランジスタ特性のばらつきがこの時点ではシリコンよりも寧ろ大きかったため、トランジスタ特性ばらつきが回路動作に及ぼす影響が大きく、その結果アクセスタイムが大きくばらついていると考えられる。これを改善するためには、プロセス技術の改善によるFET特性のばらつきの低減、及びFETの一層の性能向上によってアクセスタイムの均一性を高める必要がある。また、回路面においても特に読み出し系の回路においてトランジスタ特性のばらつきの影響を抑えるための工夫が必要である。

第2の要因は、メモリセルのアクセストランジスタにおけるリーク電流がビット線電位に影響を与える点である。図付－5にリーク電流の経路を示す。



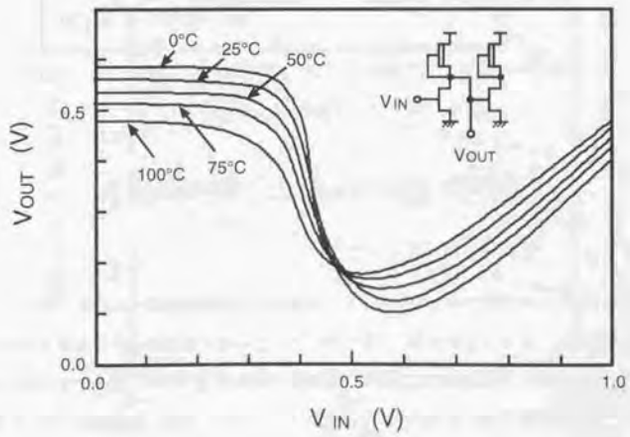
図付－5　リーク電流の経路

図は1ビットのメモリセルの回路で、ワード線WLがLowレベルの時は非選択状態となり、アクセストランジスタがOFFして「high」と「low」の一対のデータが保持される。また、WLがHighレベルになると選択状態になりアクセストランジスタがONしてビット線BLおよび$\overline{BL}$からのデータの書き込みあるいはビット線へのデータの読み出しが行われる。ここで問題となるのは、非選択状態の際に図の矢印で示す経路、すなわちLowレベルのデータが蓄えられている側のアクセストランジスタにリーク電流が流れる点である。このリーク電流は、アクセストランジスタのしきい値電圧以下で流れるためサブスレショルドリーク電流と呼ばれる。Highレベル側のアクセストランジスタは、High側の内部ノードの電位が高いために逆バイアスとなり、リーク電流は殆ど流れない。一対のビット線には多数のメモリセルが接続されており、サブスレショルドリーク電流がLowレベル側のみに流れるため、1本のビット線におけるサブスレショルドリーク電流の総和は、そのカラム内におけるデータの組み合せに依存する。すまわち、テストパターンによってカラム内のデータの組み合わせが変わるためにテストパターン依存性が生じている。これを改善するためには、アクセストランジスタのサブスレショルドリーク電流を低減する必要がある。

付．3．2　温度上昇に伴う特性劣化とその要因

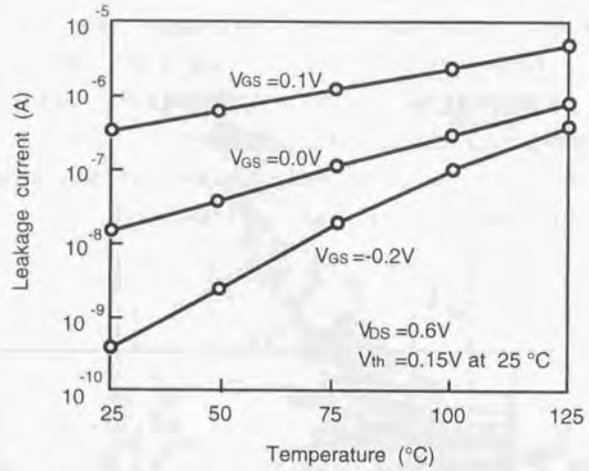
試作した4Kビット SRAMは、室温では上で示した動作を実現したが、温度を上昇させるとアクセスタイムのばらつきがさらに増大し、75℃では不良ビットが生じて全ビット動作が得られなかった[68]-[70]。SRAMを実用化する上では、少なくとも75℃での動作を保証することは重要である。そこで、温度特性の改善を図るために、以下のように要因の分析を行った。

温度が上昇すると、GaAs MESFETの特性が変化し、回路の特性が変化する[43], [92]-[94]。図付－6に0, 25, 50, 75, 100℃のE/D型DCFL回路におけるインバータ特性を示す。



図付－6　0, 25, 50, 75, 100℃のインバータ特性

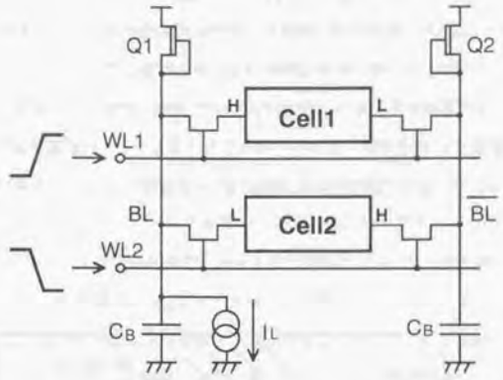
温度の上昇と共にインバータ出力のHighレベルは低下し、Lowレベルは上昇するため、電圧振幅が減少してノイズマージンが低下し、良好な動作を維持することが困難となる。図付－7にアクセストランジスタにおけるサブスレショルドリーク電流の温度依存性を示す[68]-[70]。測定したトランジスタのしきい値電圧は25℃のとき0.15Vで、測定条件はドレイン電圧が0.6V、ゲート電圧が-0.2, 0.0, 0.2Vの3種類である。サブスレショルドリーク電流は温度あるいはゲート電圧の上昇によって指数関数的に急速に増加することが分かる。



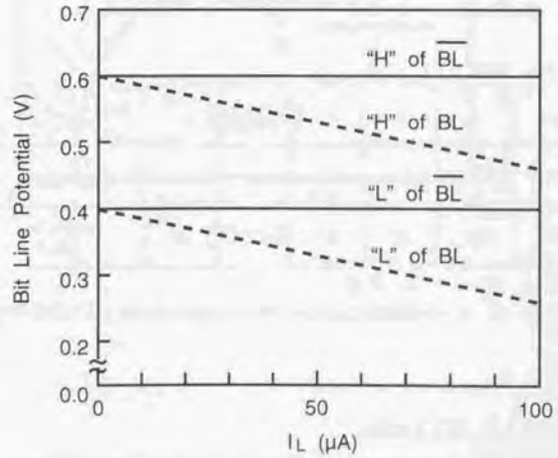
図付－7　サブスレショルドリーク電流の温度依存性

アクセストランジスタのサブスレショルドリーク電流による影響を調べるために図付－8に示す回路を用いてシミュレーションを行った。メモリセルcell1とcell2には逆のデータが蓄えられているとし、Q1及びQ2は1対のビット線負荷、C_Bはビット線容量を表わす。I_Lはcell1とcell2以外のメモリセルにおけるリーク電流の総和を表わしており、このシミュレーションでは、最悪の場合として他のメモリセルは全て左側にLowレベルのデータを蓄えていることを仮定した。図付－9にcell1とcell2を交互に選択した際のI_Lとビット線読み出しレベルとの関係を示す。I_Lが増加すると、ビット線BLは電位がほ

ぼ直線的に低下する。すなわち、温度の上昇によってリーク電流が増加すると、ビット線電位が影響を受け、アクセスタイムのばらつきおよびテストパターンに対する依存性ともに悪化する。

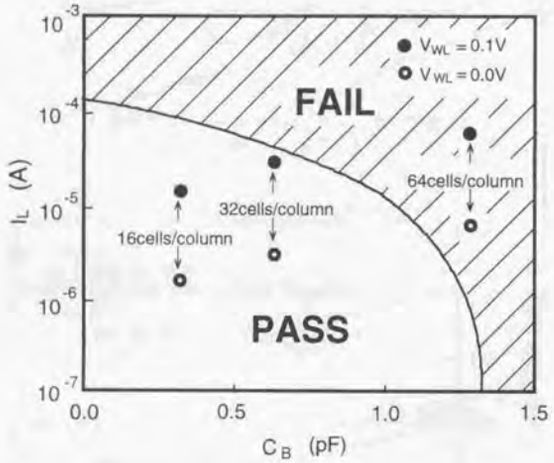


図付－8　シミュレーションに用いた回路



図付－9　ビット線電位のリーク電流に対する依存性

図付－10にビット線容量C_Bとリーク電流I_Lに対するメモリセルの安定性を示す。図中PASS領域は読み出し動作が可能な領域であり、これに対してFAIL領域は、cell1のデータが選択直後に反転して読み出し不能となる領域を示す。C_BあるいはI_Lが増加するとメモリセルが不安定となりデータの反転が起り易くなることが分かる。図中3個の○印は、ワード線電位が0.0Vのときの、1カラムあたり16、32及び64セルに対するC_BとI_Lをそれぞれ表わす。また、3個の●印はワード線電位が0.1Vのときの、同様の点を表わす。ここで、1メモリセルあたりのビット線容量を20fF、温度を75℃と仮定している。DCFL回路の場合はワード線電位が0.1V程度となるため、●印に対応し、I_Lが多くメモリセルの安定性も悪い。ワード線電位を0.0Vまで下げると○印となり、メモリセルの安定性はかなり改善される。したがって、メモリセルを安定に動作させるためにはワード線のLowレベルを下げるなどの対策が必要である。



図付－10　ビット線容量C_Bとリーク電流I_Lに対するメモリセルの安定性

付．3．3　ソフトエラー率とその要因

一般に、半導体素子にα線が入射すると、α線のエネルギーによって半導体基板内で電離が起こり、電子と正孔の対が発生する。このうち電子が回路の中の電位の高いノードに集められると、このノードの電位を低下させる。メモリ素子においてこれが起きるとメモリセル内の記憶ノードのHighレベルの電位がLowレベルの電位よりも低くなり、データが反転してしまう場合がある。一度データが反転してしまうとこれは回復不可能なエラーとなり、コンピュータを誤動作させる要因となる。このようなエラーはハードウエアのエラーと区別して「ソフトエラー」と呼ばれる。したがって、GaAsメモリを実用化する上では、ソフトエラー耐性を保証することが必要不可欠である。

そこで、試作した4KビットSRAMのソフトエラー耐性の測定を行った。表付－2に試作した4KビットSRAMと市販の1KビットECL RAMのα線に対するソフトエラーの頻度を示す。線源としてはAm^{241}を用い、測定時間20秒に対する5回のエラーカウントと平均値を示している。試作した4KビットSRAMは市販の1KビットECL RAMに比べて約3桁ソフトエラー耐性が悪いことが分かる。一般にGaAsは、シリコンに比べてバンドギャップが広いため放射線耐性に優れているといわれているが[6]、[16]、SRAMでは、記憶ノードの内部振幅がショットキ障壁高さϕ_B程度の値に制限されるため、シリコンSRAMに比べ、アルファ線入射に起因するノイズの影響をより強く受け易い。したがって、GaAs SRAMの実用化を図る上ではα線耐性の改善が重要な技術的課題である。ソフトエラーの機構や改善策に関してはいくつかの報告があるが[95]、[96]、4Kビット以上の高集積RAMに効果的に用いられた例は本研究以前にはなかった。

表付－2　ソフトエラーカウント

線源：Am^{241}

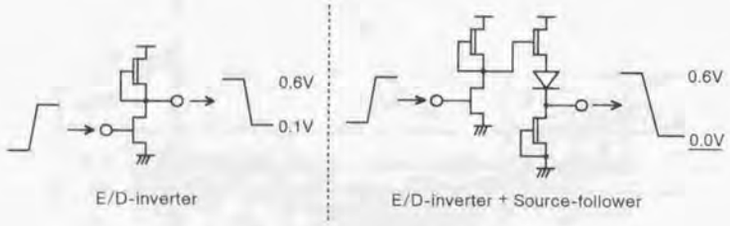
サンプル＼回数	1	2	3	4	5	平均
4Kbit RAM	～17000	～18000	～17000	～17000	～16000	17000
1Kbit ECL RAM	18	20	15	18	20	18.2

単位：エラーカウント/20秒

付．4　回路技術の改善による解決策の検討

付．4．1　ソースフォロワ回路による動作安定化の検討

前節での原因分析の結果、E/D インバータの出力の Low レベルに浮きがあり、しかもこの浮きが温度の上昇とともに増大する点が GaAs SRAM の動作を阻害する大きな要因となっていることが判明した。このような GaAs DCFL 回路の Low レベルの上昇は、ソースフォロワ回路を用いることによって低減することができるが[85],[87]、従来のソースフォロワ回路は表付－1 の BFL や SCFL のバッファ回路において用いられるような、常に電流が流れるノーマリオン型のものであったため、消費電力が大きかった。そこで筆者らは回路の最適化を行うことによって消費電力を増大させないソースフォロワ回路を開発した[68]-[70]。図付－11 に E/D インバータ回路と開発したソースフォロワ回路の構成と出力電圧レベルを示す。E/D インバータは出力の Low レベルが 0.1V 程度浮くのに対し、ソースフォロワ回路は出力の Low レベルが GND レベルまで下がる。この回路は 1 電源で動作するだけでなく、入力が Low レベルの際にショットキダイオードが OFF となるため、DC 電流は流れない。従ってこのソースフォロワ回路を適所に使用することにより、消費電力をあまり増加させることなく、必要な箇所の電圧振幅を拡大し、次段のリーク電流も抑制することができ、高温時において回路を安定に動作させることができる。



図付－11　E/D インバータ回路とソースフォロワ回路の出力電圧

付．4．2　センスアンプ回路の改善

センスアンプは電圧振幅の小さいビット線の信号を増幅する回路で、メモリの読み出し系回路の中で最も重要な回路である。前節での分析の結果、アクセスタイムのばらつきを低減するためには読み出し系回路の改善が必要であることを述べたが、動作を安定化させアクセスタイムのばらつきを抑えるためにはセンスアンプの特性を改善する必要がある。付．2 節において試作した SRAM ではセンスアンプとして単なる E/D インバータを 4 カラム毎に配置する方式をとっていたが、E/D インバータ回路は図付－6 に示すように温度特性が悪く、アクセスタイムのばらつきの原因となっていた。また、4 カラム毎に 1 個のセンスアンプが置かれていたために入力部の寄生容量が大きく入力信号の遷移時間が長かったために、さらにアクセスタイムがばらつき易かった。センスアンプの特性としては、入力の広い電圧範囲に対する感度が良いことと、出力の駆動能力が高いことが要求される。E/D インバータは明確な入力しきい値を持つため、入力のしきい値電圧付近では感度が良いが、入力電圧がしきい値からはずれると感度が著しく劣化する。また出力の駆動能力も低い。筆者らはこれらの問題点を改善する方法として図付－12 のような回路を各カラムに置くカラムセンス方式を採用することとした[68]-[70]。この回路はデプレッション型 FET とエンハンスメント型 FET からなる一対のプッシュプル回路を構成し（これを E/D プッシュプル回路と呼ぶ）、ビット線 BL 及び $\overline{BL}$ を入力とし、データ線 DATA 及び $\overline{DATA}$ を出力とする。この回路を通常の E/D インバータと比較すると、デプレッション型 FET とエンハンスメント型 FET にそれぞれ逆相の信号が入力するために電流駆動能力が大きく、また明確な入力しきい値を持たないため、微小入力や入力電圧の変動に対しても動作が可能である。また、カラムセンス方式とすることによってセンスアンプの入力部の寄生容量は低減され、入力信号が高速化される。ただし、カラムセンス方式によってセンスアンプの数が増加するために消費電力が増加する問題がある。これを解決するためにエンハンスメント型 FET(Q5)が設けられており、センスアンプ選択のための入力信号 SE によって選択・非選択が制御される。非選択状態のカラムセンスアンプでは Q5 が OFF となり電流が流ないため、常にセンスアンプは 1 個しか動かず、消費電力の増加は全くない。

図付－12　センスアンプ回路

付．4．3　メモリセルの安定化の検討

前節での要因分析の結果，GaAs SRAMの温度特性を改善するためには，メモリセルの安定度を増す必要があることが明らかになった。メモリセルの安定化を図るための手法として，非選択メモリセルのアクセストランジスタにおけるサブスレショルドリーク電流を低減することが重要である。上で述べたようにワード線駆動回路をソースフォロワ回路としてワード線のLowレベルをGNDレベルまで下げることにより，図付－10に示すようにメモリセルの安定性を大幅に改善できる。筆者らは，これを適用した4Kビット SRAMの設計および試作を行い，この手法によって温度特性に対しては十分な安定性が得られることを確認した[68]-[70]。また，この試作にはp型埋め込み層を持つトランジスタ BP-FET（Buried P-Channel FET）を採用することにより，素子特性のばらつきの低減とソフトエラー耐性の向上を図った[71], [72]。これによってばらつきに関しては改善が見られたが[68]-[72]、ソフトエラーに関しては十分な結果が得られなかった。表付－3にその際のソフトエラーの評価結果を示す[73]-[75]。評価条件は表付－2のものと同じで、p型埋め込み層を持つトランジスタを採用した4Kビット SRAM、付－2節で試作した4Kビット SRAMおよび市販の1Kビット ECL RAMのものの値が示されている。p型埋込み層によって従来のものよりも2桁弱ソフトエラー率が低減されているが、市販のECL RAMに比べると1桁強悪く、p型埋込み層の使用だけでは不十分であり、ソフトエラー耐性についてはさらに回路面での改善が必要であることが分かる。

表付－3　ソフトエラー耐性

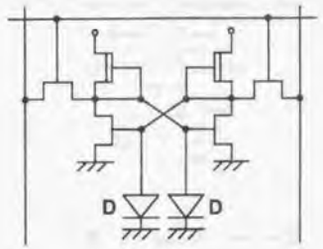
線源：Am^{241}

サンプル＼回数	1	2	3	4	5	平均
4Kbit RAM（p層無し）	~17000	~18000	~17000	~17000	~16000	17000
1Kbit ECL RAM	18	20	15	18	20	18.2
4Kbit RAM（p層有り）	298	301	283	324	259	293.0

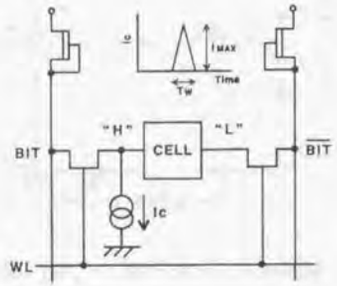
単位：エラーカウント/20秒

ソフトエラー耐性を高めるためには，アルファ線が入射した際の電位ノイズに対してメモリのデータが反転しないようにメモリセルの安定性を高める必要がある。そこで，メモリセルの記憶ノードに容量を付加することによりメモリセルの安定度を増すための研究を行った[73]-[75]。メモリセルに単に

容量を付加だけでは面積が増大するため，容量付加による面積の増加を極力抑え，効果的な容量付加を実現する方法を検討した。図付－13に容量素子としてダイオードを記憶ノードに付加したメモリセル（Type－1）の回路図を示す。このメモリセルのソフトエラー耐性を調べるためにSPICE2によるシミュレーションを行った。図付－14にシミュレーションに用いた回路を示す。メモリセルのHighレベルの蓄えられたノードから三角波の電流Icを強制的に流すことによってノード電位の低下を生じさせ，メモリセルのデータの反転の有無を調べた。そして，データの反転が起きるぎりぎりの電荷量を臨界電荷量Qcとした。電荷量は三角波の面積で与えられるが，三角波の幅Twは過去に報告されている測定結果から1.0nsと仮定し[95], [96]、高さI_{MAX}を変化させることによって臨界電荷量を求めた。



図付－13　Type-1 セルの構成

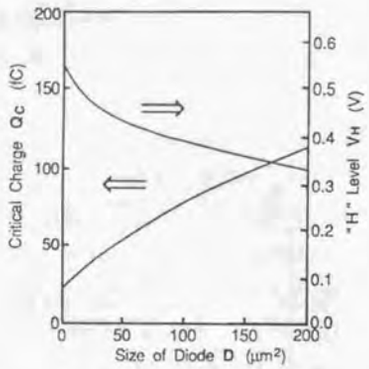


図付－14　シミュレーション回路

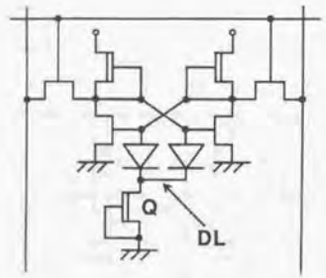
図付－15にダイオードサイズDの値に対するQcとHigh側のノードのDC的なレベルV_Hの値を示す。ダイオードサイズが増加すると、ダイオードを流れる電流が増加するためにV_Hが低下し，その結果ソフトエラーによるデータの反転が起こりやすくなり、Qcの増加が鈍化することが分かる。これはダイオードの付加が効果的に行われていないことを示している。

これを改善してダイオードの付加による効果をさらに高めるために，図付－16に示す回路（Type－2）を提案した。この回路ではダイオードのカソードとGNDとの間にゲート・ソース共通のデプレッションFET(Q)が電流源として挿入されている。図付－17にこの場合におけるダイオードサイズDに対するQcおよびV_Hの関係を示す。図中の破線は比較のためにType－2に対する値を再び示している。電流源Qの挿入によってHighレベルの低下が抑えられており，その結果臨界電荷量がほぼ直線状に増加していることが分かる。すなわち，電流源Qの存在によってダイオードの付加が効果的に作用して

いる。ダイオードのサイズに関しては、通常p型埋め込み層がある場合α線入射時の記憶ノードへの収集電荷量は 100fC 以下なので[95], [96]、臨界電荷量 100fC を満たす値としてダイオードサイズを 100μm²とした。なお、電流源Qは複数のメモリセルに対して1個あればよく、しかもメモリセル領域の外側に形成できるのでこれによる面積の増加はほとんどない。

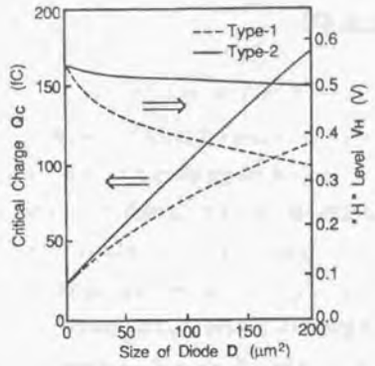


図付ー15　ダイオードサイズに対する臨界電荷量とHighレベルの依存性



図付ー16　Type-2セルの構成

ソフトエラーに対する改善策としては、メモリセルの安定性を高める方法以外に、ソフトエラーをECC（Error Correction Code）を用いて修正する方法も考えられる。この方法に関しては、パリティチェックの導入によってその有効性を示すいくつかの報告が行われているが[97]-[99]、ECCを搭載することによって面積が20%程度増加している。また、ECCはエラーを検出した際にこれを修正するためのサイクルが必要となるため、キャッシュメモリのように常に高速のアクセスが頻繁に行われる場合には、そのようなサイクルは性能を劣化させる要因となる。したがって、面積の増加が20%以下に抑えられれば、ソフトエラー自体を防止できるメモリセル安定化の方法がECCよりも望ましい。後に示すようにダイオードの付加による面積の増加は高々10%と小さく、メモリセルを安定化させる方法が適している。

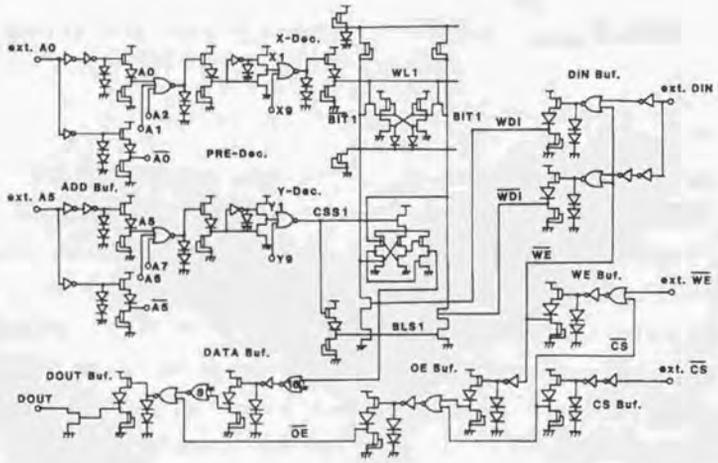


図付ー17　ダイオードサイズに対する臨界電荷量とHighレベルの依存性

付．5　16K ビット SRAM への適用

付．5．1　回路構成

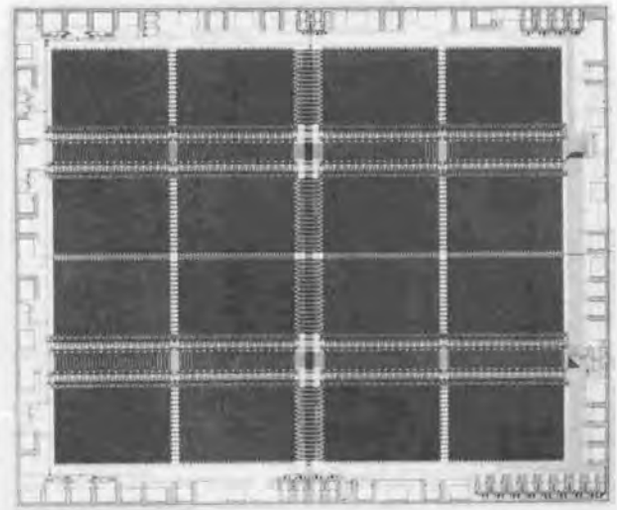
以上の改善策に基づいて 16K ビット SRAM の設計を行った[73]-[75]。図付－18 に回路図を示す。回路の適所には最適化されたソースフォロワ回路を用いることによって、出力の Low レベルを GND レベルまで低下させ、次段の回路動作を安定化させている。特にワード線ドライバの出力の Low レベルを GND レベルとすることによって、非選択メモリセルのアクセストランジスタを流れるリーク電流を低減しメモリセルの安定化を図っている。メモリセルの一対の記憶ノードにはそれぞれ 100um^{2}のショットキダイオードが付加され、各 DL ノードはロウ方向の 32 個のメモリセルで共有される。電流源となるデプレッション型 FET（Q）は、各 DL ノードと GND との間に接続されている。センスアンプには、前節で述べた E/D プッシュプル型の回路が用いられ、これが各カラムに配置されている。一対の E/D プッシュプル回路の出力は、さらにもう 1 段の E/D プッシュプル回路に入力させることよって出力をさらに増幅している。アドレス選択方式には付．2 節で述べたプリデコード方式が用いられ、アドレス選択系の回路のサイズを低減している。



図付－18　16b SRAM の回路構成

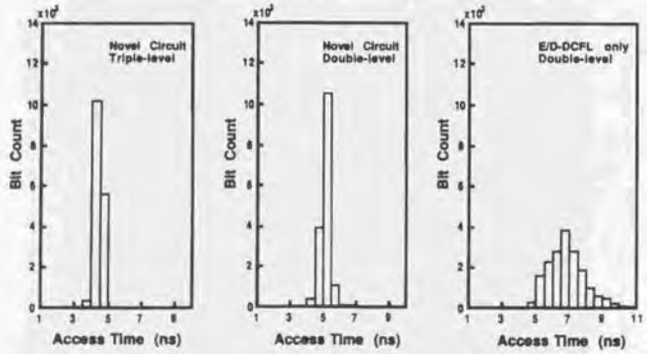
付．5．2　試作および評価結果

設計した 16K ビット SRAM の試作を行った。試作に当たっては、動作スピードを改善するためにトランジスタの最適化を行い、その結果ソース・ドレインとゲートとの間にやや濃度の薄い n 型活性層を有する BP-LDD-FET（Buried P-Channel Light Doped Drain FET）を開発し、これを用いることとした[73]-[80]。ゲート長は 0.7μm である。また、配線はこれまでの 2 層配線から 1 層多い 3 層配線とすることによりチップ面積の低減を図った[76]-[79]。図付－19 にチップ写真を示す。チップサイズは、6.00×5.05mm^{2}である。同様の回路を 2 層配線によっても試作しており[73]、[74]、その場合の面積は 7.41×7.16mm^{2}であり、面積が 58％にまで低減されている。第 1 層配線は、配線幅、間隔とも、最小 1.5μm であり、第 2 層配線では、最小 1.8μm である。これに対し第 3 層配線は電源線にのみ使用するため、配線幅、間隔とも、最小 10.0μm としている。また、最小コンタクトサイズは、1.6×1.6μm^{2}である。メモリセルサイズは、36.0×23.0μm^{2}である。セルの各記憶ノードに付加した 100μm^{2}のショットキダイオードによって 10％面積が増加しているが、付．4．3 で述べたように、ECC を用いるより面積の増加は半分に抑えられている。



図付－19　16K ビット SRAM チップ写真

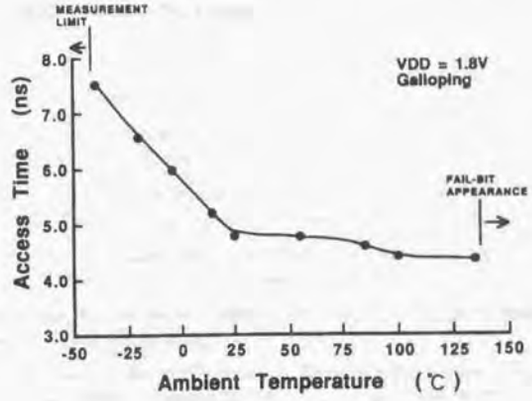
試作した16K ビットSRAMに対し評価を行った。図付－20にチップ内のアクセスタイムのばらつきを示す。(a)は本試作によるもの、(b)は同じ回路を２層配線で試作したもの[73]，[74]、(c)は付．２節で述べたE/D 型DCFL 回路によるもの[66]、[67]である。また、(a)および(b)は最も厳しいギャロッピングパターンによるもので、(c)はチェッカーボードパターンによるものである。DCFLのみによって構成された(c)では、比較的条件の緩いチェッカーボードパターンでも6ns以上の大きなばらつきが見られたいたが、新規回路方式で２層配線を用いた(b)では最も条件の厳しいギャロッピングパターンでもばらつきが2nsまで低減され、さらに３層配線を用いた(a)では、ばらつきが1.5ns以下まで低減されている。すなわち、ソースファロワ回路の適用やセンスアンプ回路の改良によりアクセスタイムの高速化と均一性の向上が図られ、アクセスタイムのばらつきが大幅に改善されている。この結果、最大アクセスタイムは11.0n sから5.0nsまで改善されている。また、３層配線化の効果によって２層配線の場合と比較して1nsの高速化が実現されている。最大アクセスタイム5.0nsは既発表の16K ビットSRAMとしては最も速い値である。また、テストパターン依存性も改善されており、本16K ビットSRAMではテストパターンによる最大アクセスタイムの変化は0.5ns以内に抑えられている。



（a）本設計のチップ　（b）２層配線のチップ　（c）E/D DCFLのチップ

図付－20　３種類の16K ビットSRAMにおけるアクセスタイムの分布

図付－21に、アクセスタイムの温度依存性を示す[75]。25℃から135℃にかけての広範囲にわたって5.0ns以下の安定な動作が実現されていることが分かる。一般にGaAs MESFETは温度の上昇とともにしきい値電圧が低下して電流値が増加する性質があるため[6]、温度の上昇とともにアクセスタイムは速くなる傾向がある。25℃以下でアクセスタイムが遅くなるのはこの効果によるもので、電流値が低下して回路動作が遅くなっていると考えられる。また、135℃以上で不良ビットが現れるのは、付．３，２節で述べた用にアクセストランジスタのリータ電流の増加のためにセルの安定性が損なわれるためであると考えられる。通常、最も厳しい使用条件でも温度は最大125℃程度なので、本SRAMは十分に使用に耐える温度特性を実現していると言うことができる。



図付－21　アクセスタイムの温度依存性

表付－４に、ソフトエラー評価の結果を示す。評価に用いたSRAMは，本試作によるものと、付．４．３節の表付－３の各試料である。評価条件も付．４．３節のときと同様である。本試作のソフトエラー率は、16K ビットとビット数が増加しているにも関わらず，最初に試作した4K ビットSRAMによりも４桁程度低く、また、p 型埋め込み層を有する4K ビットSRAMと比べても、２桁程度低い値となっているのがわかる。さらに、この値は，市販ECL SRAMよりも小さい。従って，本16K ビットSRAMにおいては十分実用化に耐えうるソフトエラー耐性が実現されているということができる。

以上の結果から、高速で実用可能なGaAs 16K SRAMを実現されており、本研究の成果が高速キャッシュメモリの実現に有効であることが示された。

表付－4　ソフトエラーカウント

線源：Am 241

サンプル＼回数	1	2	3	4	5	平均
4Kbit RAM（p層無し）	～17000	～18000	～17000	～17000	～16000	17000
1Kbit ECL RAM	18	20	15	18	20	18.2
4Kbit RAM（p層有り）	298	301	283	324	259	293.0
16Kbit RAM（ﾀﾞｲｵｰﾄﾞ付加）	4	7	5	0	4	4.0

単位：エラーカウント/20秒

＜参考文献＞

[1] 青木 正和，「大容量DRAM」，電子情報通信学会誌，Vol.73，No.4，pp.369-376，1990年4月．

[2] J. L. Hennessy and D. A. Patterson, "Computer Architecture : A Quantitative Approach," Morgan Kaufman Publishers, 1990.

[3] 馬場 敬信，「コンピュータアーキテクチャ」，オーム社，1994年．

[4] 南 宗宏，「32ビット・マイクロプロセッサ」，CQ出版，1986年．

[5] 大谷 孝之，「高速SRAM」，電子情報通信学会誌，Vol.73，No.4，pp.377-384，1990年4月．

[6] S. M. Sze, "Physics of Semiconductor Devices -Second Edition-," A Wiley-interscience Publication, pp. 245-311, 1981.

[7] K. Ohwada, M. Ino, T. Mizutani and K. Asai, "GaAs 256-Bit Static RAM," Electronics Letters, vol.18, No.7, pp.299-300, Apr. 1982.

[8] M. Ino, M. Hirayama, K. Kurumada and M. Ohmori, "Estimation of GaAs Static RAM Performance," IEEE Trans. on Electron Dev., vol. ED-29, No.7, pp.1130-1135, July 1982.

[9] G. Bert, J. Morin, G. Nuzillat and C. Arnodo, "High-Speed GaAs Static Random-Access Memory," IEEE Trans. on Microwave Theory and Techniques, vol. MTT-30, No.7, pp.1014-1019, July 1982.

[10] M. Ino, M. Hirayama, K. Ohwada and K. Kurumada, "GaAs 1Kb Static RAM with E/D MESFET DCFL," Tech. Dig. of 1982 GaAs IC Symposium, pp.2-5, Nov. 1982.

[11] Y. Nakayama, K. Suyama, H. Shimizu, S. Yokogawa and A. Shibatomi, "An LSI GaAs DCFL Using Self-aligned MESFET Technology," Tech. Dig. of 1982 GaAs IC Symposium, pp.6-9, Nov. 1982.

[12] W. V. McLevige, C. T. M. Chang and W. M. Duncan, "GaAs Enhancement/Depletion MESFET Memory Technology," Tech. Dig. of 1982 GaAs IC Symposium, pp.127-130, Nov. 1982.

[13] K. Asai, K. Kurumada, M. Hirayama and M. Ohmori, "1Kb Static RAM using Self-Aligned FET Technology," Dig. Tech. Papers of ISSCC '83, pp.46-47, Feb. 1983.

[14] N. Yokoyama, T. Ohnishi, H. Onodera, T. Shinoki, A. Shibatomi and H. Ishikawa, "A GaAs 1K Static RAM using Tunsten-Silicide Gate Self-Alignment Technology," Dig. Tech. Papers of ISSCC '83, pp.44-45, Feb. 1983.

[15] S. J. Lee, R. P. Vahrenkamp, G. R. Kaelin, L. D. Hou, R. Zucca, C.P. Lee and C. G. Kirkpatrick, "Ultra-Low Power, High Speed GaAs 256-Bit Static RAM," Tech. Dig. of 1983 GaAs IC Symposium, pp.74-77, Oct. 1983.

[16] G. L. Troeger and J. K. Notthoff, "A Radiation-Hard Low-Power GaAs Static RAM Using E-JFET DCFL," Tech. Dig. of 1983 GaAs IC Symposium, pp.78-81, Oct. 1983.

[17] S. Takano, T. Yoshihara, N. Tanino, Y. Mitsui and K. Nishitani, "A High Performance GaAs FET with Shallow N+ Implantation and Recessed Gate for Fast Static RAM," Tech. Dig. of 1983 GaAs IC Symposium, pp.82-85, Oct. 1983.

[18] N. Toyoda, K. Kanazawa, T. Terada, M. Mochizuki, M. Hirose, Y. Ikawa and A. Hojo, "A 256 x 4 Bit GaAs Static RAM," Tech. Dig. of 1983 GaAs IC Symposium, pp.86-89, Oct. 1983.

[19] F. Katano, K. Takahashi, K. Uetake, K. Ueda, R. Yamamoto and A. Higashisaka, "Fully Decoded GaAs 1Kb Static RAM using Closely Spaced Electrode FETs," Tech. Dig. of 1983 IEDM, pp.336-339, Dec. 1983.

[20] T. T. Vu, P. C. T. Roberts, R. nelson, G. M. Lee, B. hanzal, K. W. Lee, N. Zafar, D. R. Lamb, M. J. Helix, S. A. Jamison, S. A. Hanka, J. C. Brown Jr. and M. S. Shur, "A Gallium Arsenide SDFL Gate Array with

On-chip RAM," IEEE Trans. on Electron Dev., vol. ED-31, No.2, pp.144-156, Feb. 1984

[21] N. Yokoyama, H. Onodera, T. Shinoki, H. Ohnishi, H. Nishi and A. Shibatomi, "A 3ns GaAs 4K x 4 SRAM," Dig. Tech. Papers of ISSCC '84, pp.44-45, Feb. 1984.

[22] M. Hirayama, M. Ino, Y. Matsuoka and M. Suzuki, "A GaAs 4Kb SRAM with Direct Coupled FET Logic," Dig. Tech. Papers of ISSCC '84, pp.46-47, Feb. 1984.

[23] K. Nishiuchi, N. Kobayashi, S. Kuroda, S. Notomi, T. Nimura, M. Abe and M. Kobayashi, "A Subnanosecond HEMT 1Kb SRAM," Dig. Tech. Papers of ISSCC '84, pp.48-49, Feb. 1984.

[24] S. J. Lee, C. P. Lee, D. L. Hou, R. J. Anderson and D. L. Miller, "Static Random Access Memory Using High Electron Mobility Transistors," IEEE Electron Dev. Lett., vol. EDL-5, No.4, pp.115-117, Apr. 1984.

[25] K. Asai, K. Kurumada, M. Hirayama and M. Ohmori, "GaAs 1Kb Static RAM with Self-Aligned FET Technology," IEEE J. Solid-State Circuits, vol. SC-19, No.2, pp.260-262, Apr. 1984.

[26] M. Ino, M Togashi, M. Hirayama, K. Kurumada and M. Ohmori, "Design of GaAs 1k Bit Static RAM," IEEE Trans. on Electron Dev., vol. ED-31, No.9, pp.1139-1144, Sept. 1984.

[27] T. Hayashi, A. Masaki, H. Tanaka, H. Yamashita, N. Masuda, T. Doi, N. Hashimoto, N. Kotera, J. Shigeta, T. Kohashi and S. Takahashi, "ECL-Compatible GaAs SRAM Circuit Technology for High Performance Computer Application," Tech. Dig. of 1984 GaAs IC Symposium, pp.111-114, Oct. 1984.

[28] T. Mizoguchi, N. Toyoda, K. Kanazawa, Y. Ikawa, T. Terada, M. Mochizuki and A. Hojo, "A GaAs 4K Bit Static RAM with Normally-On and -Off Combination Circuit," Tech. Dig. of 1984 GaAs IC Symposium, pp.117-120, Oct. 1984.

[29] Y. Ishii, M. Ino, M. Idda, M. Hirayama and M. Ohmori, "Processing Technologies for GaAs Memory LSIs," Tech. Dig. of 1984 GaAs IC Symposium, pp.121-124, Oct. 1984.

[30] S. Kuroda, T. Mimura, M. Suzuki, N. Kobayashi, K. Nishiuchi, A. Shibatomi and M. Abe, "New Device Structure for 4Kb HEMT SRAM," Tech. Dig. of 1984 GaAs IC Symposium, pp.125-128, Oct. 1984.

[31] M. Hirayama, M. Ino, Y. Matsuoka and M. Suzuki, "A GaAs 4Kbit SRAM with Direct Coupled FET Logic," IEEE J. Solid-State Circuits, vol. SC-19, No.5, pp.716-720, Oct. 1984.

[32] K. Takahashi, T. Maeda, F. Katano, T. Furutsuka and A. Higashisaka, "A CML GaAs 4Kb SRAM," Dig. Tech. Papers of ISSCC '85, pp.68-69, Feb. 1985.

[33] F. Katano, K. Takahashi, K. Uetake, K. Ueda, R. Yamamoto and A. Higashisaka, "Fully Decoded GaAs 1-Kbit Static RAM using Closely Spaced Electrode FET's," IEEE J. Solid-State Circuits, vol. SC-20, No.3, pp.810-815, June 1985.

[34] N. Yokotama, H. Onodera, T. Shinoki, H. Ohnishi and H. Nishi, "A 3-ns GaAs 4K x 1-bit Static RAM," IEEE Trans. on Electron Dev., vol. ED-32, No.9, pp.1797-1801, Sept. 1985.

[35] P. O'conner, P. G. Flahive and B. J. Roman, "A High-Speed GaAs 1K Static Random Access Memory," IEEE J. Solid-State Circuits, vol. SC-20, No.5, pp.1080-1082, Oct. 1985.

[36] T. Hayashi, H. Tanaka, H. Yamashita, N. Masuda, T. Doi, J. Shigeta, N. Kotera, A. Masaki and N. Hashimoto, "Small Access Time Scattering GaAs SRAM Technology Using Bootstrap Circuits," Tech. Dig. of 1985 GaAs IC Symposium, pp.199-202, Oct. 1985.

[37] W. V. McLevige and C. T. M. Chang, "An ECL-Compatible GaAs E/D MESFET 1K-Bit Static RAM," Tech. Dig. of 1985 GaAs IC Symposium, pp.203-206, Oct. 1985.

[38] N. Kobayashi, S. Notomi, M. Suzuki, T. Tsuchiya, K. Nishiuchi, K. Odani, A. Shibatomi, T. Mimura and M. Abe, "A Fully Operational 1Kb HEMT Static RAM," Tech. Dig. of 1985 GaAs IC Symposium, pp.207-210, Oct. 1985.

[39] M. Hirayama, M. Togashi, N. Kato, M. Suzuki, Y. Matsuoka and Y. Kawasaki, "A GaAs 16-kbit Static RAM Using Dislocation-Free Crystal," IEEE Trans. on Electron Dev., vol. ED-33, No.1, pp.104-110, Jan. 1986.

[40] N. Kobayashi, S. Notomi, M. Suzuki, T. Tsuchiya, K. Nishiuchi, K. Odani, A. Shibatomi, T. Mimura and M. Abe, "A Fully Operational 1-kbit HEMT Static RAM," IEEE Trans. on Electron Dev., vol. ED-33, No.5, pp.548-553, May 1986.

[41] K. Nishiuchi, N. Kobayashi, S. Kuroda, S. Notomi, T. Mimura, M. Abe and M. Kobayashi, "A Subnanosecond HEMT 1-kbit Static RAM," IEEE J. Solid-State Circuits, vol. SC-21, No.5, pp.869-874, Oct. 1986.

[42] A. Fielder, J. Chun, R. Eden and D. Kang, "A GaAs 256 x 4 Static Self-Timed Random Access Memory," Tech. Dig. of 1986 GaAs IC Symposium, pp.89-92, Oct. 1986.

[43] B. L. Grung, H. K. Chung, W. Walters, G. Y. Lee, R. I. Mactaggart, S. Schuldt and K. L. Tan, "A High-Speed GaAs 256 x 4-Bit RAM," Tech. Dig. of 1986 GaAs IC Symposium, pp.93-96, Oct. 1986.

[44] N. H. Sheng, H. T. Wang, S. J. Lee, C. P. Lee, G. J. Sullivan and D. L. Miller, "A High-Speed 1K-Bit High Electron Mobility Transistors Static RAM," Tech. Dig. of 1986 GaAs IC Symposium, pp.97-100, Oct. 1986.

[45] D. K. Arch, J. K. Abrokwah, P. J. Vold, A. M. Fraasch, B. L. Grung and N. C. Cirillo, "Static Random-Access Memory Based on Self-Aligned-Gate (Al, Ga)As/n+-GaAs Superlattice Modulation-Doped FETs," Electronics Letters, vol.23, No.2, pp.87-88, Jan. 1987.

[46] B. Gabillard, C. Rocher, T. Ducourant and M. Prost, "A GaAs 1K SRAM with 2ns Cycle Time," Dig. Tech. Papers of ISSCC '87, pp.136-137, Feb. 1987.

[47] H. Tanaka, H. Yamashita, N. Masuda, N. Matsunaga, M. Miyazaki, H. Yanazawa, A. Masaki and N. Hashimoto, "A 4K GaAs SRAM with 1ns Access Time," Dig. Tech. Papers of ISSCC '87, pp.138-139, Feb. 1987.

[48] N. H. Sheng, H. T. Wang, C. P. Lee, G. J. Sullivan and D. L. Miller, "A High-Speed 1K-Bit High Electron Mobility Transistors Static RAM," IEEE Trans. on Electron Dev., vol. ED-34, No.8, pp.1670-1675, Sept. 1987.

[49] S. Notomi, Y. Awano, M. Kosugi, T. Nagata, K. Kosemura, M. Ono, N. Kobayashi, H. Ishiwari, K. Odani, T. Mimura and M. Abe, "A High Speed 1K x 4-bit Static RAM Using 0.5μm-gate HEMT," Tech. Dig. of 1987 GaAs IC Symposium, pp.177-180, Oct. 1987.

[50] C. T. Tsen, S. Kuwahara, K. Elliot, L. Salmon, A. Cappon, E. V. Korpinen, E. R. Walton, S. J. Ross, W. Klainhans and R. Kezer, "A Manufacturable Low-power 16K-bit GaAs SRAM," Tech. Dig. of 1987 GaAs IC Symposium, pp.181-184, Oct. 1987.

[51] J. K. Notthoff, R. B. Krein, J. S. Stephens, G. L. Troeger, C. H. Vogelsang and C. H. Hyun, "A 4K x 1 Bit Complementary E-JFET Static RAM," Tech. Dig. of 1987 GaAs IC Symposium, pp.185-188, Oct. 1987.

[52] M. Ino, H. Suto, N. Kato and H. Yamazaki, "A 1.2 ns GaAs 4kb Read-only-memory Fabricated by 0.5μm-gate BP-SAINT," Tech. Dig. of 1987 GaAs IC Symposium, pp.189-192, Oct. 1987.

[53] Y. Watanabe, S. Saitoh, N. Kobayashi, M. Suzuki, T. Yokoyama, E. Mitani, K. Odani, T. Mimura and M. Abe, "A HEMT LSI for a Multibit Data Register," Dig. Tech. Papers of ISSCC '88, pp.86-87, Feb. 1988.

[54] A. Fiedler, J. Chun and D. Kang, "A 3 ns 1K x 4 Static Self-Timed GaAs RAM," Tech. Dig. of 1988 GaAs IC Symposium, pp.67-70, Nov. 1988.

[55] C. H. Vogelsang, J. A. Castro, J. K. Notthoff, G. L. Troeger, J. S. Stephens and R. B. Krein, "Complementary GaAs JFET 16K SRAM," Tech. Dig. of 1988 GaAs IC Symposium, pp.75-78, Nov. 1988.

[56] W. C. terrell, C. L. Ho and R. Hinds, "Direct Replacement of Silicon ECL and TTL SRAMs with High

Performance GaAs Devices," Tech. Dig. of 1988 GaAs IC Symposium, pp.79-82, Nov. 1988.

[57] J. Chun, R. Eden, A. Fiedler, D. Kang and L. Yeung, "A 1.2 ns GaAs 4K Read Only Memory," Tech. Dig. of 1988 GaAs IC Symposium, pp.83-86, Nov. 1988.

[58] W. White, A Taddiken, H. Shichijo and M. Vernon, "Integration of GaAs 4 Kbit Memory with 750-gate Logic for Digital RF Memory Applications," Tech. Dig. of 1989 GaAs IC Symposium, pp.37-40, Oct. 1989.

[59] J. Chun, S. Enam, D. Kang and B. Remund, "A Pipelined 650 MHz GaAs 8K ROM with Translation Logic," Tech. Dig. of 1990 GaAs IC Symposium, pp.139-142, Oct. 1990.

[60] D. E. Grider, A.I. Akinwande, R. Mactaggart, P. P. Ruden, J. C. Nohava, T. E. Nohava, J. E. Breezley, P. Joslyn and D. Tetzlaff, "Development of Static Random Access Memories Using Complementary Heterostructure Insulated Gate Field Effect Transistor Technology," Tech. Dig. of 1990 GaAs IC Symposium, pp.143-146, Oct. 1990.

[61] A. Fielder and D. Kang, "A GaAs Pin-for-Pin Compatible Replacement for the ECL 100474 4K SRAM," Tech. Dig. of 1990 GaAs IC Symposium, pp.147-150, Oct. 1990.

[62] T. Tsen, J. Kwiat, E. Walton, K. Elliott, S. Tiku and A. Cappon, "A Low Power 16K GaAs HMESFET Static RAM with Built-in Redundancy," Tech. Dig. of 1990 GaAs IC Symposium, pp.155-158, Oct. 1990.

[63] 牧野、高野、谷野、茅野、「プリデコード方式を用いた低消費電力 GaAs 4Kb スタティック RAM の設計」昭和 60 年度電子通信学会半導体・材料部門全国大会予稿集、pp.2-113.

[64] N. Tanino, S. Takano, M. Noda, H. Makino, K. Sumitani, H. Nakano, K. Nishitani and S. Kayano, "A 2.5ns/200mW GaAs 4Kb SRAM," Tech. Dig. of 1986 GaAs IC Symposium, pp.101-104, Oct. 1986.

[68] 牧野、高野、野田、谷野、西谷、茅野、「GaAs 4Kb スタティック RAM」、電子情報通信学会技術研究報告、ED86-135, pp.39-46, 1987.

[66] S. Takano, H. Makino, N. Tanino, M. Noda, K. Nishitani and S. Kayano, "A 16K GaAs SRAM," Dig. Tech. Papers of ISSCC '87, pp.140-141, Feb. 1987.

[67] S. Takano, H. Makino, N. Tanino, M. Noda, K. Nishitani and S. Kayano, "A GaAs 16K SRAM with a Single 1-V Supply," IEEE J. Solid-State Circuits, vol. SC-22, No.5, pp.699-703, Oct. 1987.

[68] H. Makino, S. Matsue, M. Noda, N. Tanino, S. Takano, K. Nishitani and S. Kayano, "A 7ns/850mW GaAs 4Kb SRAM Fully Operative at 75℃," Tech. Dig. of 1988 GaAs IC Symposium, pp.71-74, Nov. 1988.

[69] H. Makino, S. Matsue, M. Noda, N. Tanino, S. Takano, K. Nishitani and S. Kayano, "A 7ns/850mW GaAs 4-Kb SRAM with Little Dependence on Temperature," IEEE J. Solid-State Circuits, vol. SC-25, No.5, pp.1232-1238, Oct. 1990.

[70] 牧野、松江、野田、谷野、高野、西谷、茅野、「75℃で動作する 7ns/850mW の GaAs 4K ビット RAM」、電子情報通信学会技術研究報告、ED88-144, pp.59-66, 1989.

[71] M. Noda, K. Hosogi, K. Sumitani, H. Nakano, H. Makino, K. Nishitani and M. Otsubo, "A high-yield 4Kb SRAM process technology using self-aligned gate MESFETs with a partially depleted p-layer," Tech. Dig. of 1988 GaAs IC Symposium, pp.227-230, Nov. 1988.

[72] 野田、細木、住谷、中野、牧野、西谷、大坪、「部分空乏化した p 型埋め込み層を有するセルフアラインゲート MESFET を用いた GaAs 4Kb SRAM プロセス技術」、ED88-144, pp.59-66, 1989.

[73] S. Matsue, H. Makino, M. Noda, N. Tanino, S. Takano, K. Nishitani and S. Kayano, "A Soft Error Improved 7ns/2.1W GaAs 16Kb SRAM," Tech. Dig. of 1989 GaAs IC Symposium, pp.41-44, Oct. 1989.

[74] 松江、牧野、野田、谷野、高野、西谷、茅野、「ソフトエラー耐性を改善した 7ns/2.1W GaAs 16Kb SRAM」、電子情報通信学会技術研究報告、ED89-137, pp.1-8, 1990.

[75] S. Matsue, H. Makino, M. Noda, H. Nakano, S. Takano, K. Nishitani and S. Kayano, "A 5-ns GaAs 16-kb SRAM," IEEE J. Solid-State Circuits, vol. SC-26, No.10, pp.1399-1406, Oct. 1991.

[76] M. Noda, S. Matsue, M. Sakai, K. Sumitani, H. Nakano, T. Oku, H. Makino, K. Nishitani and M. Otsubo, "A Triple-Level Interconnection Technology for High Speed 16Kb GaAs SRAM," Extended Abstracts of 1990 Conference on Solid State Devices and Materials (SSDM), pp.71-74, Aug. 1990.

[77] 中野、野田、酒井、松江、奥、住谷、牧野、高野、西谷、「3 層配線を用いた 4.4ns/2W GaAs 16Kb SRAM」、電子情報通信学会技術研究報告、ED90-150, pp.41-46, 1991.

[78] H. Nakano, M. Noda, M. Sakai, S. Matsue, T. Oku, K. Sumitani, H. Makino, H. Takano and K. Nishitani, " A High-Speed GaAs 16Kb SRAM of 4.4ns/2W Using Triple-Level Metal Interconnection," Tech. Dig. of 1990 GaAs IC Symposium, pp.151-154, Oct. 1990.

[79] M. Noda, S. Matsue, M. Sasaki, K. Sumitani, H. Nakano, T. Oku, H. Makino, K. Nishitani and M. Otsubo, "A High-Speed 16-kb GaAs SRAM of Less than 5 ns Using Triple-Level Metal Interconnection," IEEE Trans. on Electron Dev., vol. ED-39, No.3, pp.494-499, Mar. 1992.

[80] 野田、細木、前村、加藤、中島、牧野、西谷、大坪、「GaAs LSI 用サブミクロンゲート p 層埋込み型 FET の LDD 化による均一性、高速性の改善」、電子情報通信学会技術研究報告、ED89-130, pp.7-13, 1990.

[81] R. L. V. Tuyl and C. A. Liechti, "High-Speed Integrated Logic with GaAs MESFET's," IEEE J. Solid-State Circuits, vol. SC-9, No.5, pp.269-276, Oct. 1974.

[82] R. C. Eden, B. M. Welch, R. Zucca and S. I. Long, "The Prospects for Ultrahigh-Speed VLSI GaAs Digital Logic," IEEE J. Solid-State Circuits, vol. SC-14, No.2, pp.221-239, Apr. 1979.

[83] K. Lehovec and R. Zuleeg, "Analysis of GaAs FET's for Integrated Logic," IEEE Trans. on Electron Dev., vol. ED-27, No.6, pp.1074-1091, June 1980.

[84] K. Suyama, H. Kusakawa and M. Fukuta, "Design and Performance of GaAs Normally-Off MESFET Integrated Circuits," IEEE Trans. on Electron Dev., vol. ED-27, No.6, pp.1092-1097, June 1980.

[85] G. Nuzillat, G. Bert, T. P. Gnu and M. Gloanec, "Quasi-Normally-Off MESFET Logic for High-Performance GaAs IC's," IEEE Trans. on Electron Dev., vol. ED-27, No.6, pp.1102-1109, June 1980.

[86] M. Ino, K. Kurumada and M. Ohmori, "Threshold Voltage Margin of Normally-off GaAs MESFET in DCFL Circuit," IEEE Electron Dev. Lett., vol. EDL-2, No.6, pp.144-146, June. 1981.

[87] G. Nuzillat, G. Bert, F. Damay-kavala and C. Arnold, "High-Speed Low-Power Logic IC's Using Quasi-Normally-Off GaAs MESFET's," IEEE J. Solid-State Circuits, vol. SC-16, No.3, pp.226-232, June 1981.

[88] T. Mizutani, N. Kato, K. Osafune and M. Ohmori, "Gigabit Logic Operation with Enhancement-Mode GaAs MESFET IC's," IEEE Trans. on Electron Dev., vol. ED-29, No.2, pp.199-204, Feb. 1982.

[89] M. R. Namordi and W. M. Duncan, "The Effect of Logic Cell Configuration, Gatelength, and Fan-Out on the Propagation Delays of GaAs MESFET Logic Gates," IEEE Trans. on Electron Dev., vol. ED-29, No.3, pp.402-410, Mar. 1982.

[90] C. P. Lee, B. M. Welch and R. Zucca, "Saturated Resistor Load for GaAs Integrated Circuits," IEEE Trans. on Electron Dev., vol. ED-29, No.7, pp.1103-1109, July 1982.

[91] E. R. Walton Jr., E. K. Shen, F. S. Lee, R. Zucca, Y. Shen, B. M. Welch and R. Dikshit, "High-Speed GaAs SDFL Circuit," IEEE Trans. on Microwave Theory and Techniques, vol. MTT-30, No.7, pp.1020-1026, July 1982.

[92] K. L. Tan, H. K. Chung, G. Y. Lee, S. M. Baier, J. D. Skogen and S. M. Shin, "The Mechanism of Subthreshold Leakeage Current in Self-aligned Gate GaAs MESFET's," IEEE Electron Dev. Lett., vol. EDL-

7, No.10, pp.580-582, Oct. 1986.

[93] C. H. Chen, M. Shur and A. Peczalski, "Trapping-enhanced Temperature Variation of The Threshold Voltage of GaAs MESFET's," IEEE Trans. on Electron Dev., vol. ED-33, No.6, pp.792-798, June 1986.

[94] C. H. Hyun, C. H. Vogelsang and J. K. Notthoff, "Performance Analysis of GaAs JFET Integrated Circuits Operating in The Temperature Range of -55℃ to 125℃," Tech. Dig. of 1986 GaAs IC Symposium, pp.119-122, Nov. 1986.

[95] Y. Umemoto, N. Masuda and K. Mitsusada, "Effects of Buried p-layer on Alpha-particle Immunity of MESFET's Fabricated on Semi-insulating GaAs Substrates," IEEE Electron Dev. Lett., vol. EDL-7, No.6, pp.396-398, Apr. 1984.

[96] Y. Umemoto, N. Masuda, J. Shigeta and K. Mitsusada, "Improvement of Alpha-Particle-Induced Soft-Error Immunity in a GaAs SRAM by a Buried p-Layer," IEEE Trans. on Electron Dev., vol. ED-35, No.3, pp.268-274, Mar. 1988.

[97] K. Furutani, K. Arimoto, H. Miyamoto, T. Kobayashi, K. Yasuda and K. Mashiko, "A Built-In Hamming Code ECC Circuit for DRAM's," IEEE J. Solid-State Circuits, vol. SC-24, No.1, pp.50-56, Feb. 1989.

[98] M. Asakura, Y. Matsuda, H. Hidaka, Y. Tanaka and K. Fujishima," An Experimental 1-Mbit Cache DRAM with ECC," IEEE J. Solid-State Circuits, vol. SC-25, No.1, pp.5-10, Feb. 1990.

[99] K. Arimoto, Y. Matsuda, K. Furutani, M. Tsukude, T. Ooishi, K. Mashiko and K. Fujishima," IEEE J. Solid-State Circuits, vol. SC-25, No.1, pp.11-17, Feb. 1990.

謝　辞

本論文をまとめるにあたり、終始ご親切なる御指導と御鞭撻を賜った東京大学大規模集積システム設計教育研究センター 浅田邦博教授に衷心より御礼申し上げます。

また、本論文の作成にあたり御懇篤なる御検討と御教示を頂いた東京大学大規模集積システム設計教育研究センター 鳳紘一郎教授ならびに平本俊郎助教授、東京大学工学部電気工学科 田中英彦教授、東京大学生産技術研究所 喜連川優教授、ならびに東京大学国際・産学共同研究センター 岡部洋一教授に厚く御礼申し上げます。

本研究の遂行にあたり、終始御懇切なる御教示と御鞭撻を賜り、また本論文作成の機会を与えて頂いた米国VSIS Inc.社長 堀場康孝博士、三菱電機株式会社システムLSI開発研究所所長 松本平八博士、同光・マイクロ波デバイス開発研究所長 松川隆行博士、同ULSI開発研究所 安岡晶彦所長に厚く御礼申し上げます。

本研究の遂行、および論文の作成にあたり、直接御指導頂き、数々の御教示を頂いた三菱電機株式会社本社半導体営業企画部グループマネージャー 茅野晋平博士、同システムLSI開発研究所設計技術開発第三部 浜野尚徳部長、同グループマネージャー 益子耕一郎博士、同光・マイクロ波デバイス開発研究所μ波デバイス開発部 大坪睦之博士、同北伊丹事業所高周波光素子設計部長 西谷和雄博士、同システムLSI開発部 角正参事、同ULSI開発研究所計画部 高野聡博士、同液晶事業推進部開発部 篠原尋史課長に厚く御礼申し上げます。

また、本論文における数々の実験とその分析、解析にご協力頂き、有益な御討論と御指摘をしていただいた大阪大学基礎工学研究科 野田実助教授、深見特許事務所 酒井將行氏、三菱電機株式会社システムLSI開発研究所 中瀬泰伸氏、森中浩之氏、鈴木弘明氏、同光・マイクロ波デバイス開発研究所谷野憲之グループマネージャー、奥友希氏、住谷光一氏、中野博文氏、ならびに松江秀一氏に心から感謝致します。

末筆ながら、本研究の期間中終始有益な御討論と御協力をを頂いた三菱電機株式会社システムLSI開発研究所、同光・マイクロ波デバイス開発研究所、同ULSI開発研究所ならびに同北伊丹事業所の各位に心から感謝致します。

＜研究業績目録＞

（）内は本論文の関連する章を表す

（1）発表論文（査読有り）

1．「A GaAs 16K SRAM with a Single 1-V Supply」
S. Takano, H. Makino, N. Tanino, M. Noda, K. Nishitani, S. Kayano
IEEE Journal of Solid-State Circuits, Vol.SC-22, No.5, Oct. 1987, pp.699-703. （付録）

2．「A 7-ns/850mW GaAs 4-kb SRAM with Little Dependence on Temperature」
H. Makino, S. Matsue, M. Noda, N. Tanino, S. Takano, K. Nishitani, S. Kayano
IEEE Journal of Solid-State Circuits, Vol.25, No.5, Oct. 1990, pp.1232-1238. （付録）

3．「A 5-ns GaAs 16-kb SRAM」
S. Matsue, H. Makino, M. Noda, H. Nakano, S. Takano, K. Nishitani, S. Kayano
IEEE Journal of Solid-State Circuits, Vol.26, No.10, Oct. 1991, pp.1399-1406. （付録）

4．「A High-Speed 16-kb GaAs SRAM of Less than 5 ns Using Triple-Level Metal Interconnection」
M. Noda, S. Matsue, M. Sakai, K. Sumitani, H. Nakano, T. Oku, H. Makino,
K. Nishitani, M. Otsubo
IEEE Transactions on Electron Devices, Vol.39, No.3, Mar. 1992, pp.494-499. （付録）

5．「An Application of Air-Bridge Metal Interconnections to High Speed GaAs LSI's」
M. Noda, H. Matsuoka, N. Higashisaka, M. Shimada, H. Makino, S. Matsue,
Y. Mitsui, K. Nishitani, A. Tada
IEICE Transactions on Electronics, Vol.E75-C, No.10, Oct. 1992, pp.1146-1153.

6．「A BiCMOS Wired-OR Logic」
Y. Nakase, H. Suzuki, H. Makino, H. Shinohara, K. Mashiko
IEEE Journal of Solid-State Circuits, Vol.30, No.6, Jun. 1995, pp.622-628.

7．「A 2.6-ns 64-b Fast and Small CMOS Adder」
H. Morinaka, H. Makino, Y. Nakase, H. Suzuki, K. Mashiko, T. Sumi
IEICE Transactions on Electronics, Vol.E79-C, No.4, Apr. 1996, pp.530-537. （第2章）

8．「A Design of High-Speed 4-2 Compressor for Fast Multiplier」
H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, H. Shinohara, K. Mashiko,
T. Sumi, Y. Horiba
IEICE Transactions on Electronics, Vol.E79-C, No.4, Apr. 1996, pp.538-548. （第3章）

9．「A 286 MHz 64-b Floating Point Multiplier with Enhanced CG Operation」
H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, K. Mashiko, T. Sumi
IEEE Journal of Solid-State Circuits, Vol.31, No.4, Apr. 1996, pp.504-513. （第5章）

10．「An 8.8-ns 54×54-bit Multiplier with High Speed Redundant Binary Architecture」
H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, K. Mashiko
IEEE Journal of Solid-State Circuits, Vol.31, No.6, Jun. 1996, pp.773-783. （第3章）

11．「Leading-Zero Anticipation Logic for High-Speed Floating Point Addition」
H. Suzuki, H. Makino, H. Morinaka, Y. Nakase, K. Mashiko, T. Sumi
IEEE Journal of Solid-State Circuits, Vol.31, No.8, Aug. 1996, pp.1157-1164. （第4章）

（2）講演　国際学会（審査有り）

1．「A 2.5ns/200mW GaAs 4Kb SRAM」
N. Tanino, S. Takano, M. Noda, H. Makino, K. Sumitani, H. Nakano,
K. Nishitani, S. Kayano
Technical Digest of 1986 GaAs IC Symposium, pp.101-104.
於 Grenelefe, Florida, Oct. 28-30, 1986.　（付録）

2．「A 16K GaAs SRAM」
S. Takano, H. Makino, N. Tanino, M. Noda, K. Nishitani, S. Kayano
1987 ISSCC Digest of Tech. Papers, pp.140-141.
於 New York, Feb. 25-27, 1987.　（付録）

3．「A 7ns/850mW GaAs 4Kb SRAM Fully Operative at 75℃」
H. Makino, S. Matsue, M. Noda, N. Tanino, S. Takano, K. Nishitani,
S. Kayano
Technical Digest of 1988 GaAs IC Symposium, pp.71-74.
於 Nashville, Tennessee, Nov. 6-9, 1988.　（付録）

4．「A high-yield 4Kb SRAM process technology using self-aligned gate MESFETs with a partially depleted p-layer」
M. Noda, K. Hosogi, K. Sumitani, H. Nakano, H. Makino, K. Nishitani,
M. Otsubo
Technical Digest of 1988 GaAs IC Symposium, pp.227-230.
於 Nashville, Tennessee, Nov. 6-9, 1988.　（付録）

5．「A Soft Error Improved 7ns/2.1W GaAs 16Kb SRAM」
S. Matsue, H. Makino, M. Noda, N. Tanino, S. Takano, K. Nishitani,
S. Kayano
Technical Digest of 1989 GaAs IC Symposium, pp.41-44.
於 San Diego, California, Oct. 22-25, 1989.　（付録）

6．「A Triple-Level Interconnection Technology for High Speed 16Kb GaAs SRAM」
M. Noda, S. Matsue, M. Sakai, K. Sumitani, H. Nakano, T. Oku,
H. Makino, K. Nishitani, M. Otsubo
Extended Abstracts of 1990 Conference on Solid State Devices and Materials (SSDM), pp.71-74.
於 仙台, Aug. 22-24, 1990.　（付録）

7．「A High-Speed GaAs 16Kb SRAM of 4.4ns/2W Using Triple-Level Metal Interconnection」
H. Nakano, M. Noda, M. Sakai, S. Matsue, T. Oku, K. Sumitani,
H. Makino, H. Takano, K. Nishitani
Technical Digest of 1990 GaAs IC Symposium, pp.151-154.
於 New Orleans, Louisiana, Oct. 7-10, 1990.　（付録）

8．「A 8.8-ns 54 x 54-bit Multiplier Using New Redundant Binary Architecture」
H. Makino, Y. Nakase, H. Shinohara
Preceedings of 1993 International Conference on Computer Design (ICCD), pp.202-205.
於 Cambridge, Massachusetts, Oct. 3-6, 1993.　（第3章）

9．「A 64bit Carry Look-ahead CMOS Adder using Modified Carry Select」
H. Morinaka, H. Makino, Y. Nakase, H. Suzuki, K. Mashiko
Preceedings of 1995 Custom Integrated Circuits Conf. (CICC), pp.585-588.
於 Santa Clara, California, May 1-4, 1995　（第2章）

10．「Leading-zero Anticipatory Logic for High-speed Floating Point Addition」
H. Suzuki, Y. Nakase, H. Makino, H. Morinaka, K. Mashiko
Preceedings of 1995 Custom Integrated Circuits Conf. (CICC), pp.589-592
於 Santa Clara, California, May 1-4, 1995　（第4章）

11．「A 286MHz 64-bit Floating Point Multiplier with Enhanced CG Operation」
H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, K. Mashiko
1995 Sympo. on VLSI Circuits Digest of Tech. Papers, pp.15-16.
於 京都, Jun. 8-10, 1995.　（第5章）

（3）講演　国内学会（審査無し）

1.「プリデコード方式を用いた低消費電力GaAs 4Kbスタティック RAMの設計」
牧野、高野、谷野、茅野
昭和60年度電子通信学会半導体・材料部門全国大会、pp.2-113　（付録）

2.「GaAs 4Kbスタティック RAM」
牧野、高野、野田、谷野、西谷、茅野
1986年度電子情報通信学会電子デバイス研究会、ED86-135、pp.39-46　（付録）

3.「部分空乏化したp型埋め込み層を有するセルフアラインゲート MESFETを用いたGaAs 4Kb SRAMプロセス技術」
野田、細木、住谷、中野、牧野、西谷、大坪
1988年度電子情報通信学会電子デバイス研究会、ED88-144、pp.59-66　（付録）

4.「75℃で動作する7ns/850mWのGaAs 4Kビット RAM」
牧野、松江、野田、谷野、高野、西谷、茅野
1988年度電子情報通信学会電子デバイス研究会、ED88-150、pp.103-110　（付録）

5.「GaAs LSI用サブミクロンゲートp層埋込み型FETのLDD化による均一性、高速性の改善」
野田、細木、前村、加藤、中島、牧野、西谷、大坪
1989年度電子情報通信学会電子デバイス研究会、ED89-130、pp.7-13　（付録）

6.「ソフトエラー耐性を改善した7ns/2.1W GaAs 16Kb SRAM」
松江、牧野、野田、谷野、高野、西谷、茅野
1989年度電子情報通信学会電子デバイス研究会、ED89-137、pp.1-8　（付録）

7.「3層配線を用いた4.4ns/2W GaAs 16Kb SRAM」
中野、野田、酒井、松江、奥、住谷、牧野、高野、西谷
1990年度電子情報通信学会電子デバイス研究会、ED90-150、pp.41-46　（付録）

8.「Auエアブリッジ配線によるGaAs LSIの高速化検討」
野田、松岡、東坂、島田、牧野、三井
1991年度電子情報通信学会電子デバイス研究会、ED91-178、pp.87-92

9.「BiCMOS ワイヤドOR論理」
中瀬、須田、鈴木、牧野、篠原
1993年度電子情報通信学会電子デバイス研究会、ED93-38、pp.25-33

10.「新しいキャリーセレクト方式(MCS)を使った64ビットキャリールックアヘッドCMOS加算器」
森中、牧野、中瀬、鈴木、益子、角
1995年度電子情報通信学会集積回路研究会、ED95-97、pp.1-6　（第2章）

11.「高速浮動小数点加算器のための桁落ちシフト量予測回路の提案」
鈴木、森中、牧野、中瀬、益子、角
1995年度電子情報通信学会集積回路研究会、ED95-98、pp.7-12　（第4章）

12.「CGに適した機能を有する286MHz、64ビット浮動小数点乗算器」
牧野、鈴木、森中、中瀬、益子、角
1995年度電子情報通信学会集積回路研究会、ED95-99、pp.13-20　（第5章）

（4）その他発表（審査無し）

1.「GaAs 4Kビットスタティック RAM」
高野、牧野、野田、谷野、西谷
三菱電機技報　Vol.60, No.8, 1986, pp.19-23

2.「高速ASIC設計技術」
斉藤、佐伯、加藤、山岸、牧野
三菱電機技報　Vol.70, No.2, 1996, pp.22-27