2006 年 10 月 31 日

サイエンティフック・システム研究会
科学技術計算分科会 2006 年度会合資料

SPARC64 V/VI の高性能，高信頼技術

富士通株式会社
サーバシステム事業本部
井上 愛一郎
inoue.aiichiro@jp.fujitsu.com

[アブストラクト]

富士通は黎明期である 1950 年代からコンピュータハードウェアの開発に一貫して取り組んできた．この長い開発の歴史の中で，富士通では信頼性を常にテーマとして掲げると同時に，最先端の高性能技術を開発してきた．

これらの高性能と高信頼の技術を盛り込んだメインフレームの CPU をベースにして，2000 年から SPARC64 V の開発に着手して，これを 2003 年から出荷した． そして 2004 年には，これを 90nm の最先端半導体に焼きなおした二世代目の SPARC64 V＋を出荷し，高性能と高信頼によって大変好評を頂いている．

また，これに続くプロセサとして，デュアルコア化，コアのエンハンス，フロントサイドバスを高スループットの Jupiter バスとするなど，マイクロアーキを刷新した SPARC64 VI の出荷が間近となっている．本稿では，ここに至る開発の歴史を概観しながら，高性能と高信頼に関わる技術を紹介し，あわせて，将来に向けた方向性を示していきたい．

[キーワード]

SPARC64，高性能，高信頼，デュアルコア，最先端半導体，開発の歴史

1. はじめに

半導体の集積度の低い時代において，1チップに CPU を収める要件は，マイクロプロセサ開発の大きな制約であった．一方，同時代に，メインフレームやサーバ用の CPU 開発に求められた第一の要件は，高性能と信頼性の追及である．この時代のマイクロプロセサとメインフレームの CPU とは，機能に大きな隔たりがあった．

テクノロジの進歩でチップの集積度が上がるに伴い，徐々に，高性能化のための複雑な機能が，マイクロプロセサに実装されていった．やがてチップをまたぐ信号授受のコストが，性能のボトルネックになるに至り，全てのコンピュータの CPU は１チップで作られるようになった．そして，この１チップ化が，かつてのマイクロプロセサを起源とする CPU と，メインフレームを起源とする CPU の競合状態を生んだ．その中で，多くの人が性能差以外に，大きな意味を持たないかのように受け止めている．どの CPU も，本当に，違いがなくなってしまったのだろうか？

半導体の集積度向上はなおも続き，チップあたりのトランジスタ数は CPU 1 個分を超過し，それが大容量キャッシュの内蔵に繋がった．現在は，複数 CPU を１チップに収め，さらにシステム機能も CPU チップに取り込む流れにある．その一方で，向上した周波数や微細化は消費電力を増大させ，電力量削減と熱冷却が，現在の

大きな課題である. そのなかで, システムのありかたまでを含む検討, すなわち, 従来どおり性能向上を追求していくのか, 何によって高性能を実現していくのか, 社会の重要なインフラに求められる要件を現在注目されている機能で満たしていけるのか, などに対する解が求められている. これから数年間はメインフレームやサーバ用の CPU にとっては, 試練の連続であろうことは想像に難くない.

一旦世にでた製品は, 開発当初は先進的で優れたものであっても, やがて寿命がくる. しかし, 製品のなかにある技術は, 注目の多寡があるものの, それは市場の要求や時代のテクノロジとの組み合わせなどに左右されており, 寿命という概念のあてはまらないものである. 技術は, 必然性との出会いで注目をあび, あるいは, 新たな技術が加わることにより変容し, 自ら必然性を獲得するものである.

本論文は, 技術とその必然性の観点から, メインフレームとその流れを受ける UNIX サーバの CPU: SPARC64 について述べる. また, メインフレームやサーバ用の CPU の現在の課題と, 今後の方向性に言及する. 第 2 章で, FACOM シリーズから現在の GS シリーズおよび SPARC64 にいたる, 富士通のコンピュータの開発の歴史と, そこで開発された技術を示す. 第 3 章で, SPARC64 シリーズのマイクロアーキテクチャを概説し, 最新のコアエンハンスを事例に引いて, 現在の技術を述べる. 第 4 章では, 命令実行処理装置, 命令リトライ機能を, メインフレームの流れの観点で整理する. 第 5 章では, 消費電力の問題を, アクティブ電力とリーク電流にわけて述べる. 第6章で, 業界のトレンドとなっている, マルチコアとシステム機能の搭載についてふれ, 最後に, 第 7 章で, 開発手法における技術の継承を考える.

2. 富士通のコンピュータの開発を振り返る

2.1. FACOM シリーズ

富士通は 1950 年代はじめのコンピュータの黎明期から開発に取り組み, 1954 年にリレー式の FACOM100 を稼動させて以来, 一貫して商用コンピュータを開発している.

FACOM128A/B などのリレー式の後, FACOM201 などパラメトロンと主記憶にはコアを使った計算機を経て, 1960 年代にはトランジスタの時代に入った.

1964 年に IBM がシステム/360 を発表し, これに対抗する純国産コンピュータとして, 富士通は FACOM230-10 から FACOM230-70 までの体系化された一連のコンピュータを開発した. この開発の過程でいち早くIC 化を進め, 1968 年に完成した FACOM230-60 は全面的に IC を採用している.

1973 年に完成したFACOM230-75 には, バッファメモリと称したキャッシュの採用や記憶装置にECCコードを採用など, 高性能, 高信頼の両面で, 今に引き継がれる多くの重要な技術が実現されている. 中には, 今では当たり前である技術も多いが, それはFACOM230-75 の先進性を物語っている. その一例として, 命令フェッチと命令実行をデカップリングや, 分岐予測による投機実行がある. 回路規模は大きくなるが, 現在のマイクロプロセッサの高速化機能では主流の技術である[1].

また FACOM230-75 は, RAS に関する設計思想も現在なお模範とすべきものを持っていた. まず, エラーを洩らさずにいち早く見つけ, 記録し, 回復するという基本がすべて押さえられている. データパスでは, バイトごとにパリティビットを設け, 演算回路の入力から出力の間もプレディクションによってパリティを保存し, また, 乗算回路などパリティプレディクションが困難な回路では, レシデュチェックを行っている. このようにして, レジスタから演算器と途中の経路を含めて一貫してデータのインテグリティを保証し, さらには命令の再試行を行って間欠故障を救済している.

これらを綿密に実現しているプロセサは, 現在においてもむしろ例外的な存在である. FACOM230-75 は, 現

[1] 最近になって, この考え方の起源がFCOM230-75[1]にあることを, 三輪修氏のホームページ[3]で知った.

在のコンピュータとは比べ物にならないほど僅かなトランジスタ数で構成されていた. しかし, 基幹業務を担うシステムとして開発されたために, 信頼性に関して妥協をせず, 向上のために多くの回路を費やすことができた. トランジスタ数はあり余るほどになった現在において, この技術がますます価値あるものとなっている.

2.2. Mシリーズ

国際競争力のある国産コンピュータを目指した通産省の後押しもあり, 富士通は日立とともに新シリーズの開発に着手し, 1970 年代初頭からAmdahl[2]と組んで国際標準とも言えるIBM互換機の開発に乗り出した. ここに始まるMシリーズは, M-190, M-380, M-780, M-1800とほぼ5年ごとに世代交代しながら, およそ20年間にわたるメインフレームの全盛時代となった.

Mシリーズでは, 最先端の ECL LSI を使用した. この集積度と速度の向上が切り札となり, コンピュータの性能向上が進んだ. 表1に, M シリーズのゲート数, 実行方式と出荷年を示す. M-190 が発表され出荷された 1974 年, 1975 年の時期, 下位機種の M-160 などをふくめ全部で 6 機種が開発された他, 少し遅れて 1978 年には 4CPU までのマルチプロセサシステムを実現する M-200 が開発された. わが国初のスーパーコンピュータ:FACOM230-75APU の完成が 1977 年であるから, 下表のように世代交代が 5 年毎であっても, 開発は実に矢継ぎ早に進められたようだ.

表 1. Mシリーズ

	ゲート数		ゲート遅延時(ps)	命令制御方式	出荷年
	LSI	CPU 合計			
M-190	100	-	700	2 サイクル・パイプライン	1975 年 11 月
M-200	100	106K	700	2 サイクル・パイプライン	1978 年 12 月
M-380	400	436K	350	2 サイクル・パイプライン	1982 年 5 月
M-780	3,000	638K	180	1 サイクル・パイプライン	1986 年 12 月
M-1800	15,000	1,411K	80	1 サイクル・パイプライン	1991 年 4 月

Mシリーズの時代に発展した, 高性能, 高信頼技術について述べる.

命令制御方式は, M-380 で 2 サイクルパイプラインとなり, M-780 で 1 サイクルパイプラインとなった. この後, スーパースカラ方式に切り替わるまでのおよそ 10 年の間, M-780 のパイプラインが数世代にわたる基本となった. この M-780 は, 私が設計にたずさわった最初のシステムでもある. 命令制御方式のほかには, ハーバードアーキテクチャ(命令とデータに分かれたキャッシュを装備), 仮想計算機機構など現在のコンピュータの骨組みとなる機能が, Mシリーズの時代にかたち作られた.

GBS と呼ぶ CPU 間共有の 2 次キャッシュを備えた 3 階層記憶装置は M-380 で実現された後しばらく姿を消していた. 最近の CPU チップにある CPU コア間共有の 2 次キャッシュは GBS と同じ発想であり, 興味深い.

また, M-190 で既に, サービスプロセッサ(SVP)によるリモートメンテナンスが実現されている. 自動修復やメモリパトロール, 交代メモリ, 3.3m 秒ごとの演算器のパトロールなどの RAS 技術は, ゲート規模の拡大と合わせて充実していった.

ベクトル計算機を概説すると, FACOM230-75 からは APU(Array Processing Unit)が派生し, これは日本初のスパコンであった. VP-400 は M-380 に, VP-2000 は M-1800 に結合され, より使いやすい汎用性のあるスパコンであった. VPP-500 から VPP-5000 は VLIW 形式の命令セットを持つ Tachyon アーキテクチャのスカラ機にベクトル計算機を結合し多数並列化したものであった.

M-1800 の後に, ECL の超大型汎用コンピュータの開発が行なわれた. ECL の素子は, バイポーラトランジスタで電流を常に流して動作させるため, 速度は早いが発熱が大きい. それで M-780, M-1800 と水冷方式を実装した. M-780 や M-1800 の, SSC や MLA と呼ばれる CPU ボードを挟み込む, 伝導冷却モジュール CCM (Contact Cooling Module)は巨大で重かった. この頃は, 集積度も速度も頭打ちとなりつつあった. ECL の発熱に苦労している頃, CMOS の半導体が進歩してきていた.

2.3. ECL から CMOS へ

CMOS 素子はスイッチングの際の充放電でしか電流を流さないために電力を喰わないにも関わらず, 集積度の高さにおいて抜きん出ていた. 初期の弱点であった速度は世代ごとに大きく改善し, 1990 年ころは, ECL より遅いもののやがて追い越すのは時間の問題という状態にあった.

その状況下で, M-1800 の次世代機のプロジェクトの論理マスタファイルは既に出来ており, シミュレーションが走り始めていたが, 1991 年にキャンセルされた. そして, 新たにCMOSを使った超大型汎用コンピュータの開発が始まった. この CMOS のシステムが現在に続くメインフレーム製品群であり, ECL 時代の M シリーズに対し, GS シリーズと命名された.

2.4. GS シリーズと SPARC64

GS シリーズ最初の製品 GS8600 では, 2 階層キャッシュシステム, ブランチヒストリ, ストアの突き放しや, プリフェッチ機構を導入した. CPU のコア部の回路規模は, M-780 とほぼ同等であり, 命令制御部, 演算器, 1 次キャッシュ制御部, 命令用 1 次キャッシュ, オペランド用 1 次キャッシュの 5 つのチップで構成された. さらに 2 次キャッシュの制御やディレクトリで 3 チップを費やし, 2 次キャッシュのデータ部は SIMM 形状の専用のボードで CPU あたり 2 枚ずつ搭載した. この当時は, メインフレームの CPU コアと1チップで作られるマイクロプロセッサでは, どちらも CMOS 半導体ではあっても回路規模にかなり大きな差があった.

CMOS 半導体の世代交代は早く, それに合わせて私たちも毎年のように新機種を開発した. CMOS 半導体はスケーリング則によって, 世代交代のたびに集積度をおよそ 2 倍向上し, スピードをおよそ 1.5 倍向上してきた. その上, その集積度とスピードの利益を全て使い切る設計をしても, 電源電圧の低下と相まって消費電力は殆ど変わらないという, とてつもなく恵まれた状況にあった. GS8600 では, LSI あたり 1 千万トランジスタであり, CPU コアを 5 チップで構成したが, その 2 年後に出荷した GS8800 では, 2 次キャッシュを除く CPU コアを 1 チップに収めた.

GS8800 の時代に, 1チップでメインフレームクラスの CPU が作れるようになったことを機に, マイクロプロセッサとメインフレームで, 回路規模に関する差が無くなった. トランジスタ数の増加は凄まじく, 現在では, 5 億トランジスタ超のチップが市場でみられ, 10 億トランジスタ超の製品が発表されているが, この爆発的な増加がマイクロプロセッサにもたらしたものは, 1チップで作るが為に回路規模を小さくせざるを得なかった制約からの, 完全な解放である. 制約から離れた後は, 増加するトランジスタを高性能化機能の実装に注ぎこみ, 積極的な性能向上が展開されている.

GS8800B は, この性能競争の激化の中で開発した. CPU の制御方式を M-780 以来続いてきたロックステップパイプライン方式から, アウト・オブ・オーダ制御のスーパースカラ方式へと大きく変えた. ここで開発された制御方式は, 現在も GS と SPARC64 V/VI に受け継がれて, 高性能と高信頼を両立する技術の核となっている. この詳細は, 4 章で述べる.

---

[2] IBMでシステム 360 の開発を行っていたGene M. Amdahl博士が, 1970 年に設立.

GS8900 の後，西暦 2000 年を機にシリーズ名称は GS21 となった．

米国HAL Computer Systemsで開発していたSPARCプロセサがキャンセルになり，GSのCPUを開発してきた設計者が，SPARCアーキテクチャ[4]のCPUも担当することになった．GSのCPUをベースにした初代のSPARCチップがSPARC64 V[5]である．図 1は，GSのCPUと，SPARC64 Vのチップ実装図であり，両者の類似度の高さが現れている．GSベースの設計を決めるにあたり，ソフトウェア性能評価シミュレータを用いて，基本的な構造は維持しながらGSのCPUをSPARC64 のCPUに作り変えるアプローチでの性能確保を確認し，性能向上につながる構成を検討・評価した[6][7]．このシミュレータ上での検討を数ヶ月で終え，設計開始から 14 ヶ月目にはテープアウトしていた．設計でむしろ梃子摺ったのは，2 次キャッシュをチップに搭載するために新規設計した部分だった．

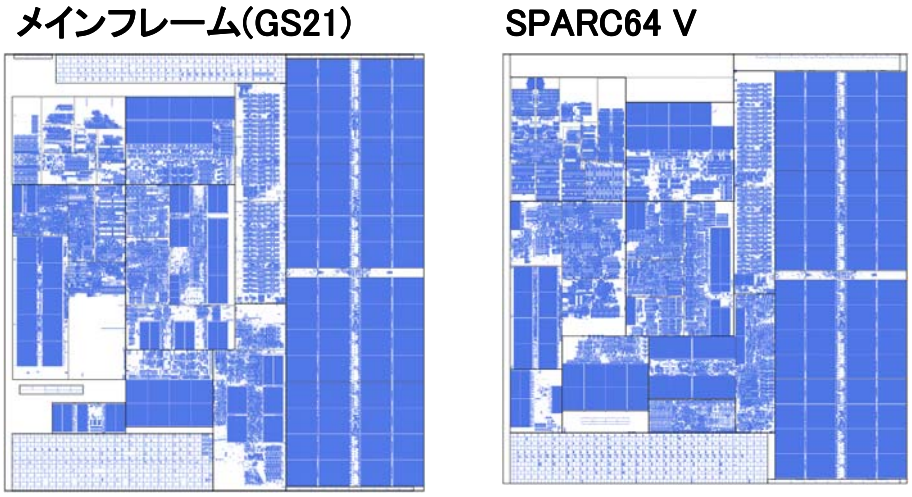

図 1. チップ実装図の比較

その後は GS も SPARC64 も，同じ設計者が開発している．90nm の半導体を使用した SPARC64 V+を経て，デュアルコア×マルチスレッドで 4 スレッド並列の SPARC64 VI，さらにその次の 4 コアの SPARC64 VII を開発中であり，GS も GS21 モデル 900 の CPU(2006 年 5 月発表)の開発を行った．

表 2. GS シリーズと SPARC64 シリーズ

	使用半導体 （全て CMOS）	周波数	出荷年
GS8600	350nm －	非公開	1996 年 10 月
GS8800	250nm －	非公開	1998 年 9 月
GS8800B	220nm －	非公開	1999 年 4 月
GS8900	180nm －	非公開	2000 年 4 月
GS21-600	130nm 銅配線 8 層	≧1GHz	2003 年 3 月
SPARC64 V	130nm 銅配線 8 層	1.35GHz	2003 年 4 月
SPARC64 V+	90nm 銅配線 10 層	2.16GHz	2004 年 8 月
SPARC64 VI	90nm 銅配線 10 層	2.40GHz	T.B.D.
SPARC64 VII	65nm 銅配線 11 層	～2.70GHz	T.B.D.
GS21-900	90nm 銅配線 10 層	非公開	2007 年 2 月(予定)

現在，CPU の単体性能をメインフレームに期待することはほとんどなくなった．しかし，GS シリーズはオペレーティングシステムを含めてシステムとしてみた場合に，基幹業務分野で優位性を保っている．そして，高い信頼度とホットスタンバイなどの独自機能ゆえにオープン系の装置で簡単に置き換えることが出来ない．

3. SPARC64 のマイクロアーキテクチャ

SPARC64 V[8][10]は，アウト・オブ・オーダ実行機構を備えたスーパースカラ・プロセッサである．SPARC64 VI/VII [9][11][12]は，SPARC64 V からの性能向上を目的に，マルチコア化，Jupiter バスの採用，マルチスレッド化と演算機能の強化を行った．CPU コアのマイクロアーキテクチャは基本的に同じで，コア数が SPARC64 V は1コア，SPARC64 VI は 2 コア，SPARC64 VII は 4 コアである．マルチコア化については後述する事として，ここでは SPARC64 V のマイクロアーキテクチャを概説し，また SPARC64 VI/VII における Jupiter バスとコアのエンハンスについて述べる．

3.1. SPARC64 V のマイクロアーキテクチャ

キャッシュは，命令とオペランドに対する個々の 1 次キャッシュと，命令とオペランドで共有の 2 次キャッシュの2階層構造である．(2コアを備える SPARC64 VI，4コアを備える SPARC64 VII では，いずれもコア間で2次キャッシュを共有している．)

命令フェッチは，命令バッファを介して実行部とデカップリングされている．ブランチヒストリを備え，命令実行とは非同期的に命令を先取りして命令バッファに充填する．この制御を，以降，先行制御と呼ぶ．命令のフェッチ幅は 32 バイトであり，8 命令を同時にフェッチできる．

サイクルあたり最大 4 命令を，デコード＆イシューする．分散型のリザベーションステーションは，アドレス生成用(2 つ)，固定小数点演算用(2 つ)，浮動小数点(2 つ)，および分岐処理用(1つ)の合計 7 つ有り，各リザベーションステーションから，サイクルごとに最大1命令をディスパッチできる．個々のリザベーションステーションで，実行準備の出来たものからアウト･オブ･オーダでディスパッチする実行機構を備える．真のデータ依存の待ち合わせは投機的に処理する．すなわち，投機的にディスパッチし，演算開始のタイミングでデータが利用できなければ，ディスパッチを無効化する．レジスタのリネーミングはレジスタ・アップデート・バッファ方式で，命令デコード時に割り当て，コミットで開放する．

オペランドのロードとストアは，フェッチポートとストアポートで管理する．キャッシュミスの場合にはフェッチポートからリサイクルし，ストアはストアポートに対応するストアバッファにデータを格納して，それ以降は突き放し処理を行う．

オペランド用の 1 次キャッシュは 8 バンクに分割されており，干渉がなければ 2 オペランド同時に読出せる．従ってサイクルあたり最大で 16 バイト(8 バイト×2)のスループット持つ．

ウィンドウレジスタのアクセスを高速化するために，Joint Window Register(JWR)と称する機構をもつ．全部で 8 ウィンドウある内の，現ウィンドウを含む前後 3 ウィンドウ分のコピーを JWR に置き，演算実行時にはここからデータを読み出す．

3.2. Jupiter バス

Jupiterバスは，システムスループットの大幅な改善を狙い開発した．PRIMEPOWER[3]は，2CPUから 128CPUまでを 250/450/650/850/900/1500/2500 の 7 つの性能レンジの機種でカバーする．その開発当初はSPARC64GP[13]を搭載して，良好なリニアリティを実現していた．その後，SPARC64GPをSPARC64 Vに置き換え，CPU性能を大幅に向上させてシステムのポテンシャルを引き出し，大きな性能向上を図ってきた．しかし，アプリケーションによっては，少数のCPUでシステムスループットを使い切ってしまうことがあり，これはCPUの数を増やしたときの性能伸び悩みに繋がる．

[3] SPARC64 プロセサが組み込まれるシステム

スループット改善のほかに，キャッシュコヒーレンシ制御にも工夫を行い，キャッシュの状態遷移に伴うトランザクション発生を減らした．さらにRASの観点から，アドレスとデータの何れもECCによってエラー修復可能とした．

3.3. コアのエンハンス：マルチスレッド化と演算機能強化

SPARC64 VIはVertical Multi-Treading(VMT)をサポートし，これに対応してウィンドウレジスタを8ウィンドウ×2 セットに倍増した．またデータパスの実装に着目して，SPARC64 V の JWR にかえて，Current window Replace Buffer(CRB)と Current Window Register(CWR)から演算器への読出しを行う方式とした．

そして SPARC64 VII ではマルチスレッドを 2 つのスレッドを混在して実行する Simultaneous Multi-Threading(SMT)方式にエンハンスし，これに対応して CRB と CWR もそれぞれ 2 セットを装備した．さらにレジスタリネーミング用の General register Update Buffer(GUB)を 32 本から 48 本に強化した．

SPARC64 VI で，FMA の演算のレイテンシを短縮して，Linpack などの FMA を高密度で実行するアプリケーションで，FMA を稠密に実行できるようにした．これにあわせてレジスタの書き込みポートを 4 つに増強し，レイテンシ短縮にバランスさせた他，レジスタリネーミング用の Floating point register Update Buffer(FUB)を 48 本に強化した．

SPARC64VI/VII は，マルチスレッドのサポートのために Floating Point Register(FPR)本体も 2 セットに強化した．

4. 技術と必然性

メインフレームの技術は過去のものだろうか．メインフレームの CPU をベースに開発したことは，SPARC64 V/VI/VII をどのように特徴づけるだろうか．マイクロプロセサから発展したチップ搭載のシステムが，性能でメインフレームを凌駕した現在，メインフレームの技術を開発した当時に立ち返って概説し，これらの問いを考える．

メインフレームの技術開発を牽引してきたものは，高性能・高信頼性からの要件であった．現在ミッションクリティカルな分野のシステムに，同じ要件が求められている．技術のドライバ(要件)が同じ場合，課題解決に共通のアプローチがとれることを，実例に基づき述べる．ここにみられる必然性は，要件の一致である．

最初に，命令実行処理装置を紹介する．この技術は，SPARC64 V に引き継がれている．ドライバとなる要件は，“演算部への高い命令供給能力”である．要件の背景として，メインフレームは高い 1 次キャッシュミス率があり，SPARC64 V の場合は演算処理能力の向上がある．最近 Intel は，命令フェッチの方式を従来のトレースキャッシュから分岐予測の方式に変更した．限られた情報からの推測であるが，ここで紹介するものと同様の方式と見ている．

次に，命令リトライについて述べる．まず，命令リトライを実現させる前提にある，同期一括更新方式を概説する．これは，GS8800B の新規アイデアであり，この技術によって明確なチェックポイントを形成でき，メインフレームで従来からあった命令リトライを，アウト・オブ・オーダのシステムで実現することができた．命令リトライは，現在の GS のみならず SPARC64 V/VI に受け継がれて，高性能と高信頼を両立する技術の核となっている．そして，GS21 と SPARC64 V/VI は，アウト・オブ・オーダ実行機構を備えると同時に命令リトライが出来る，世界で唯一の CPU である．

4.1. 命令実行処理装置

ブランチヒストリを用いた動的な分岐予測方式を考案したのは，ECL の超大型コンピュータを開発していた

1990 年頃だったが，プロジェクトがキャンセルされたため，実現できたのは GS8600 からである．

このころ，マイクロプロセッサの分野でも分岐予測が実用化されつつあった．しかし命令フェッチは命令処理パイプラインの先頭の1ステージにすぎず，先行制御の仕組みを備えていないのが普通であった．したがって，分岐予測は命令処理の頭で方向予測するだけの簡単な機構が一般的であった．このような機構では，命令フェッチでキャッシュミスが発生すると，パイプライン全体が停止していた．

一方，FACOM230-75 に起源を持つ先行制御 の命令フェッチは，M-780 では富士通の追永勇次の手によって，最大 48 バイト情報を格納可能な命令バッファを備え，実行中の命令列と，最大 2 系統までの分岐先ターゲットを保持する形に進化していた．メインフレームを用いる基幹業務処理は動作する命令域が広大であり，このため，命令用に 64KB，128KB等の大きなキャッシュを用意しても命令キャッシュは 5%を超えるミス率となるケースが散見した．その状況においては，巨大な命令バッファを備えて命令フェッチを先行させて先取りし，分岐先アドレスが確定した後に分岐先の命令列を即座にフェッチ可能とする作りに必然性があった．巨大な命令バッファを持ったため，命令フェッチに費やす回路規模としては当時としてはとびぬけて大きく，超大型といわれたコンピュータであればこそ実現できたものだった．しかしこれは同時に，マイクロプロセッサの分野で実現しつつあったような分岐予測の機構をそのままでは適用出来ないことを意味した．そして命令キャッシュミス率の悪化など，当てにならない予測による弊害の方が，分岐予測機構を装備することによる効果を上回ることは容易に想像された．その上メインフレームの命令セットアーキテクチャでは， レジスタ相対[4]で分岐先アドレスが指定されるため，ますますマイクロプロセッサで行われつつある機構は適用が困難であった． GS8600 以降で装備した分岐予測機構は，M-780 で装備した巨大な命令バッファを備える先行制御の仕組みの上に動的な予測機構を組み込んだ．その前提は，上記の必然を覆す機能の実現であり，すなわち命令フェッチのスループットや命令キャッシュの効率を低下させないこと，予測ミスのペナルティーを充分小さくすることであった．

考案したしくみ[14]を，図 2を使って説明し，効果を図 3 に示す．この方式の特徴は，分岐処理で非連続となった命令シーケンスに対しても，デコーダへ連続して命令を供給できることにある．その実現には，複数命令バッファに，近い将来実行されるであろう命令流を保持しておき，命令バッファからデコーダへの命令供給を，タグ部からの情報で制御する方式をとった．図 2 では，3つの命令シーケンスの同時保持の仕組みを例示しており，命令アドレス生成回路，命令バッファ，タグに，A,B,Cを付し，組として扱われることを示している．現在処理中の命令列が，図中命令アドレス生成回路Aに保持されたアドレスの指す場所にあるとする．生成回路Aのアドレスでフェッチされた命令列は，命令バッファAに格納される．命令キャッシュアクセスは，生成回路A～Cのいずれか一つを選択して行われる．選択アルゴリズムの説明は，ここでは割愛する．命令フェッチアドレスは，命令キャッシュへの送付と同時に，ブランチヒストリへ送られる．ブランチヒストリへの登録は，分岐命令実行時に，命令アドレスと分岐先アドレスを対にして実施されており，従って，過去に実行された命令で，リプレースされていない場合，ブランチヒストリでヒットし，分岐先のアドレスが得られる．ヒット時に，取り出した分岐先命令アドレスは，選択回路（図 2）に送られ，次のサイクルで命令フェッチ要求の対象となる．それと同時に命令アドレス生成回路Bにも送られる．図 3 の上部は，分岐予測を備えない処理のフローを表す．分岐命令(A)の分岐先アドレスが確定した時点で，分岐先命令のフェッチ(図 3 'IA')が開始され，分岐確定以降に分岐先命令のデコードが実施されている．下部の図は，命令フェッチ時にブランチヒストリがヒットし，予測された分岐先命令が，分岐命令のデコードに先行してフェッチされる様子を表す．先行フェッチにより，分岐命令（A）のデコードの次サイクルに分岐先命令のデコードが可能となっている．上述のように，複数の命令バッファに，異なった命令流の命令列が格納されており，デコーダへの命令供給元のバッファ選択は，タグA,B,Cで管理する．巨大な命令バッファに

[4]分岐先アドレスを，命令のオペランドに指定されたレジスタにあるデータと，オペコード内の即値の和で求める

よって命令フェッチを命令実行とデカップリングし，分岐予測結果に従って，これから実行されるであろう命令シーケンスをたどりながら，命令バッファが一杯になるまで命令を先取りしていく仕組みは，キャッシュミスの弊害を少なくし，高い命令供給能力を持った非常に先進的な作りといえる．

また，命令のフェッチアドレスを比較に用いて，ブランチヒストリのヒット・ミスを厳密に判断する仕組みとしたことで，過去に実行されていない命令で誤ってヒット判定する誤りを抑制した．ターゲットアドレスの正誤判定は，分岐先アドレスが判明した時点で行う．そして，分岐が成立するか否かの予測を誤ったために不要な命令が実行されている場合には，先行制御の仕組みを利用して，実行部での分岐命令およびそこに至る命令列の実行完了を待たずに，直ちに命令フェッチのやり直しを開始する．

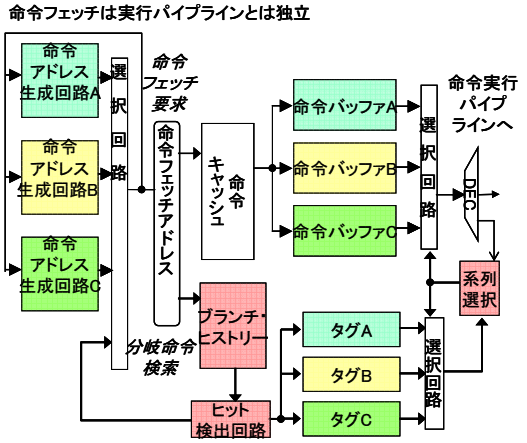

図 2. 分岐予測機構

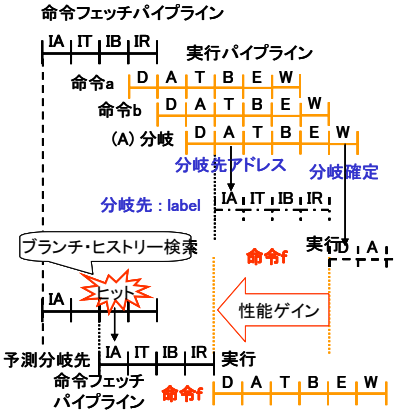

図 3. 分岐予測の効果

この方式をはじめて適用した際にはブランチヒストリの容量はわずか 256 エントリだったが期待通りの効果であり，キャッシュのミス率をいくらか低下させるというおまけ付きだった．

その後，世代ごとにブランチヒストリの容量を増加した．またサブルーチンからの戻りアドレスが，呼びだし元によって異なることに対処するためにリターンアドレススタックを追加した．

SPARC64 V/VI でも，基本的な仕組みは考案当時のものから変わっていない．しかし，ブランチヒストリは 16 K エントリであって 1 次キャッシュの大きさに匹敵し，命令バッファ容量は 192 バイト(命令フェッチ巾 : 32 バイト × 6)である．そしてリターンアドレススタックのほかに，複雑な挙動を示す高頻度ルーチンの予測精度の向上のために Write-cycle-driven Global History Table(WGHT)といった補助機構も追加している．

SPARC64 V/VI に限らず最近のマイクロプロセッサでは，命令フェッチがかなり大きな回路となっている．これは実行部の高速化と並列度の向上により，命令フェッチが性能に与える影響が相対的に大きくなったため，必然的に機能が強化されたためである．結果的に，M-780 で命令フェッチにつぎ込んだ物量は今では珍しくはなくなっている．こうした中で振り返ると，GS8600 で装備した命令フェッチの方式は，ある程度物量を豊富に使える前提では，効率の良い方式であったことが判る．

4.2. 命令リトライ

GS8800B は初めての，動的スケジューリングに基づくアウト・オブ・オーダ制御のスーパースカラ方式のシステムである．M-780 以来続いてきたロックステップパイプライン方式の変更を伴うこの開発は，富士通の清水和之が Amdahl 社で行われていた Chaos プロジェクトを私に紹介してくれたことがきっかけとなって始まった．Chaos プロジェクトにはロバート・トマスロー，ボブ・マイヤー，マーク・ニールセンらがメンバーとして加わっていた．

Chaos は究極のアウト・オブ・オーダのマシンを目指すものだったが，当初のアイデアは全くのペーパーマシンであり，1サイクルの論理段数が 30 段を超えるようなものであった．さらに，CISC 型の複雑な命令の処理に必要なマルチフロー展開やマイクロプログラム制御，そして RAS については，ほとんど検討されていなかった．さらに，従来のメインフレームで実現していた全ての機能を，レベルを低下させずにサポートできるのか，何が本質的な問題であるかも不明であった．

本当に「ものになる」のかをはっきりとさせるために，ほどなくマーク・ニールセンが富士通(川崎)の職場に合流し，浅川岳夫，本車田強，森田國樹らと一緒に勉強会を行い，検討を重ねた．Amdahl の Chaos のメンバーと，週を置かずにテレビ会議を実施する中で，問題が明確になり，解決の糸口が見えてきて，次第に具体的な回路イメージが浮かび上がった．このような準備を経て，基本検討開始からおよそ２年で設計を完了した．GS8800B は，Chaos でもたらされた概念を基盤に，過去から継承してきた技術と，多くの新規アイデアを加えたものとなった．

メインフレームでは，高信頼性の要件は絶対であり，性能向上と引き換えに高信頼を失うことはあってはならなかった．それを踏まえた高性能の追及は設計者として挑戦しがいのある壁であり，その克服を通して確かな手ごたえを味わった貴重な経験である．この技術は 1998 年に特許出願した[15]．

4.2.1. 同期一括更新方式

演算など，命令で陽に指定された処理を制御する Reservation Station for Execution(RSE)などとは別に，コミット・スタック・エントリ(CSE : Commit Stack Entry)と呼ばれるバッファが存在する．CSE は，命令ごと(マルチフロー展開では1フローごと)に1エントリが割り当てられ，実行中の命令の進捗状況の監視に用いる．CSE のエントリは，プログラムのオーダに従いイン・オーダで実施される命令コミット時に，無効化される．このコミット処理で，レジスタ(GPR,FPR など)の更新，ストアバッファにデータを格納した後の突き放された状態のストア処理に対するメモリ反映指示(ストア・イン方式のためキャッシュに書き込む)，プログラムカウンタの更新指示が，一括して出される．これを同期一括更新と呼んでいる．

アウト・オブ・オーダで実行した結果はすべて，コミットまでソフトウェアから見えない作業用レジスタ(GUB，FUB)に格納され，ソフトウェアから見えるプログラマブルな資源には反映されない．CSE の更新指示はイン・オーダで行われるために，いつ実行を中断しても，その時点でプログラムカウンタが指す位置で，プログラマブルな資源の一貫性が保証できる．一方，作業レジスタや演算器の入出力などに残っているものは全て破棄することが出来る．これは，コミットのたびにチェックポイントを形成していることになっており，M-780 などのパイプライン制御のW(Write)サイクルの機能を実現している．

この機構は，分岐の予測ミスずれのリカバリや，割り込み発生時にプログラマブルな資源の一貫性を保つことに使用される．しかし，最たる特徴は，従来からの命令リトライを可能にすることにある．

4.2.2. 命令リトライの機能

命令フェッチ，命令実行に関するあらゆる部分のどこかで，エラーが検出された場合，割り込み同様に CPU のプロセススイッチを起動して命令実行を中断する．前述の CSE が形成したチェックポイントで停止することになる．そこで命令フェッチや実行に関する資源をフラッシュし，あらためてプログラムカウンタの位置から再開すれば，データインテグリティを損なうことなく命令リトライを行うことが出来る．GS8800B では，CSE のエントリ数は 16 で，同時にコミットできる命令は最大 3 命令だった．SPARC64 V/VI では CSE は 64 エントリで，同時に最大 4 命令までコミットできる．この様に数の拡張はしているが，同期一括更新方式や命令リトライ機構自身は全く変わっていない．

FACOM230-75 を振り返ると，CPU 部の IC 数はおよそ 24,000 個である．当時のコンピュータは，1日に 10 回を超えるダウンも発生した中で，命令リトライはなくてはならない機能で，それゆえ，よく磨かれた機能でもあった．時をへて，命令リトライをフィールドで観測することは稀になったが，その状況は近年になって変わりつつある．高度な半導体の微細化技術でゲート酸化膜は原子数個のレベルとなり，トランジスタはゆらぎともいえる特性変動を示すようになっており，今後は，RAM 部分に限らずロジック部でも，ソフトエラーの影響が無視できなくなる．また，回路規模が拡大すると，ソフトエラーに遭遇する確率は確実に上がる．

この背景の元，性能向上用途の回路規模拡大を，エラー発生頻度とのトレードオフで判断する時代がくる．その中にあり，前述の命令リトライ機能は，トレードオフ回避の効果を発揮する．それは，プログラマブルなリソースの回路規模は命令セットアーキテクチャによってほぼ決まるためである．プログラマブルなリソースを使って命令リトライを掛けるため，ここの回路規模が変化せず，結果として信頼性が変化しないことが，命令リトライの成功率を一定に保つ．性能向上目的で回路が拡大しても，随所にエラーチェッカを設けて実行途上のエラーを漏らさず捕まえて命令リトライを実施すれば，エラーに遭遇した場合にも，ソフトウェアへの影響は皆無となる．

現在コンピュータは，社会のインフラとして定着しており，コンピュータの引き起こすデータ化けやダウンは，社会に非常に大きなダメージを与える．これが，コンピュータの信頼性が改めて注目される所以であり，その重要性は増していく．社会基盤に位置づけられるサーバは，信頼性への要求に十分答えるものでなければならない．今ふたたび，命令リトライ機能は，欠くことのできない機能となりつつある．

4.2.4. 他社のとりくみ

SPARC64 V/VI と富士通のメインフレームの CPU 以外に，演算器のエラー検出と命令リトライが可能なプロセッサとして，IBM メインフレームの Z シリーズがあげられる．

IBM の場合，Power4，Power5 では，SPARC64 V/VI や GS21 同様にアオトオブオーダの実効機構を備えているが，Z シリーズでは古典的な単一命令発行のロックステップパイプライン制御となっている．これはメインフレームの使われ方から演算性能に対する要求が高くないこともあるが，信頼性を求めるために，そうせざるを得なかったと言う面もある．

IBMのZシリーズでは，演算部分にパリティプレディクションなどの入力データのパリティを保存する機構を設けないかわりに，演算部分を 2 重化している．そして演算結果をレジスタに書き込む前に 2 つの演算結果の照合チェックを行い，不一致の場合には，これを破棄して命令リトライを行っている．この方法は，2 重化ではどちらの結果が正しいのかが分からない短所を，結果を破棄することによって補い，データインテグリティを保証することが出来るとともに，ソフトエラーからの回復を実現している．しかし一旦，照合チェックでエラーを見逃すと，その痕跡は残らず[5]，照合回路そのものがALUに匹敵する回路となるためエラーの危険にさらされており，万全とはいえない．

加えて，演算器を 2 重に使用し，照合のためにパイプラインの 1 ステージを費やし一致チェックを行うことから，同じ仕事に対する消費電力としては，非常に不利である．さらにロックステップパイプラインでは，演算器の実

[5] SPARC64 V/VIのEXECレジスタのパリティをチェックする方式では，チェッカに故障があっても，伝播されたパリティは，またいずれかのチェッカによってデータインテグリティを保証される．例えば，演算途中でエラーが発生した場合に，演算結果のチェッカが壊れてエラーの検出が出来なくても，そのパリティがアーキテクチャレジスタに書き戻されて保存され，次にこのレジスタを使用したときに，今度は演算入力のパリティチェッカによってエラーが検出される．

行効率はどうしても低くなるから，高い演算性能を求める場合には，より多くのハードウェアが必要となる．従って，高い演算性能が必要なプロセッサでは，この方法は好ましくなく，消費電力の観点でも上述と合わせて不利となる．

5. 消費電力の問題

5.1. アクティブ電力の低減

SPARC64 Vの特徴の一つは消費電力あたりの性能に優れている(高い)事である．平均では約 40W，最悪ケースで 65Wの消費電力はサーバ用のハイエンドプロセサでは群を抜いて低い．無駄な電力を食わせずに無理なく高い動作周波数を達成するために，クロックをHツリーで分配してパルスラッチを使用し，食い込みとスキューを積極的に利用してディレーを稼いでいる．さらに，スタンダードセルベースで設計して論理と実装をハンドチューンで最適化している．また，チップ面積の多くを占める 2 次キャッシュはブロック分割して，これを構成するSRAMマクロを必要なときだけ活性化した他，演算器の入出力レジスタやパイプラインのタグ類などのラッチを，内部のクロック伝播をInhibit(IH)ピンによって制御して，不要なときは止めている．ている．このIHピンの効果は大きく図 4に示すようにラッチ単体で 70%～80%の電力を削減できる．ラッチのIHピンを上げると出力も固定されるので，出力に繋がっている論理の動作も抑えられて電力削減効果が更に大きくなる．

SPARC64 V では，全ラッチの 50%以上に IH ピンの制御を適用している．これらの結果として，SPARC64 V のアクティブ電力は，数%の動作率に相当する程度まで低減されている．しかしまだ改善の余地が残っており，今後は更に徹底していく．

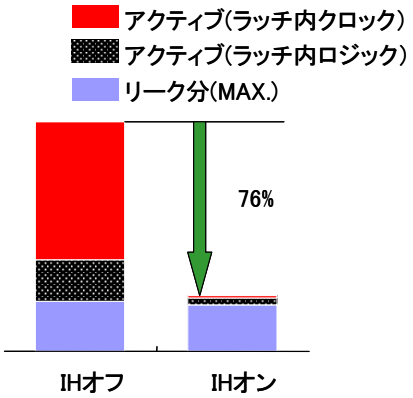

図 4. 電力削減効果

5.2. リーク電流の低減

半導体の微細化により電力に対してリークの占める割合が大きくなってきた．図 5に示すように，SPARC64 V(90nm)では消費電力の平均 40Wと最悪 65Wの差分 25Wのほとんどがリークであり，消費電力全体に占める割合はおよそ 40%に達している．SPARC64 VIではアクティブ電力はSPARC64 Vより改善が進んでおり，デュアルコアでも単コアのSPARC64 Vより少し多い 55W程度である．平均電力は，リークも入れて 80W程度であるが，最悪の場合の電力は 120Wに達し，その内訳はリークが 60%以上を占める．

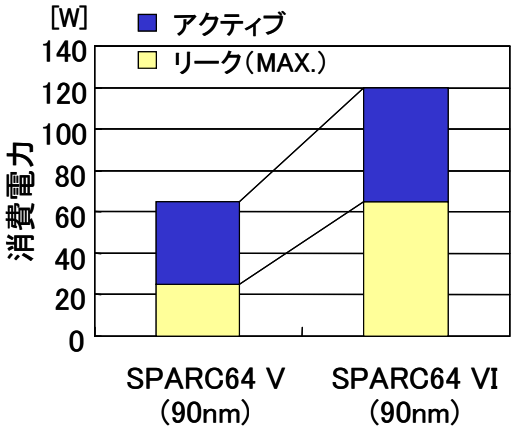

図 5. 消費電力の推移

リーク電力の削減は，背景により複数手段を使い分けて削減する課題である．これを，90nm 版の SPARC64 V を用いた実験結果に基づいて示す．

実験から，リーク電流は，個体差,温度,電圧の関数であることを示す．個体差はテクノロジの微細化に伴い拡大する．ばらつきの縮小は，微細化技術の課題である．温度と電流は，設計技術に関連する課題である．

実験は，スタティック電力（図 5左）と消費電力(図 6右：スタティック電力+ダイナミック電力)を，電源電圧(Vdd)と温度を振って測定した．図中，横軸に温度，縦軸に消費電力をとり，温度を振ったときのチップごとの消費電力値を示した．スタティック電力は，クロックを停止した状態の測定であり，ダイナミック電力は，同じサンプルでクロックを入れて行った測定である．実験対象のサンプルは，出荷基準による選別前の全チップであり，温度をフィールドではありえない値まで振って測定した．

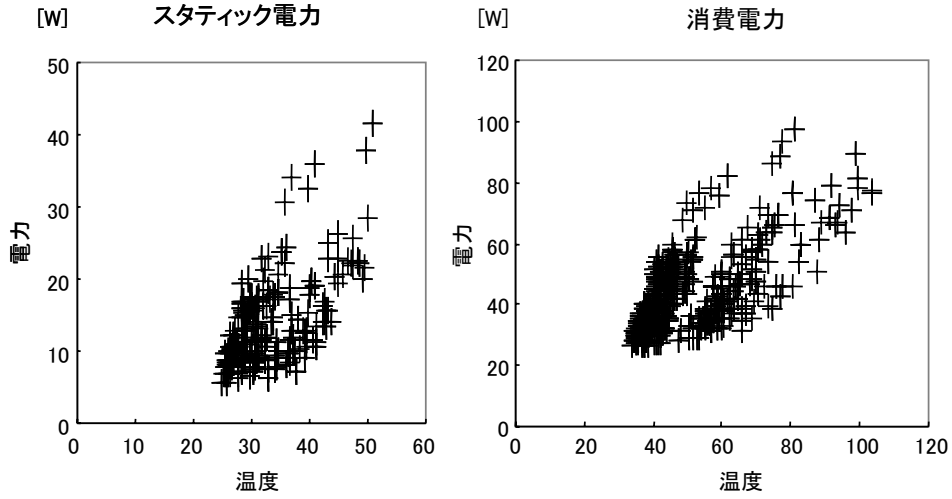

図 6. 電力のばらつき （スタティック電力，消費電力）

経験から，リーク電流 Ileak は定性的には電源電圧 Vdd と温度 t の両方に対して指数関数的に増加することが分かっている．その特性に基づき，リーク電流を，実験式(1)であらわす．

$$Ileak = Iddq \times e^{\{\alpha(Vdd-1.0)+\beta(t-85)\}} \quad \text{———} \quad \text{実験式(1)}$$

Ileak: リーク電流

Iddq : 85℃，1.0V で測ったリーク電流

Vdd ：電源電圧

式(2)に変形し，測定データをつかって回帰分析して，係数αとβの値を求め，最終式(1')となる.

ln(Ileak)−ln(Iddq)= α (Vdd−1.0) + β (t−85) ———— 式(2)

α =3.06, β =0.0157

Ileak = Iddq × e^{3.06(Vdd−1.0)+0.0157(t−85)} ———— 式(1')

Ileak: リーク電流

Iddq : 85℃, 1.0V で測ったリーク電流

Vdd : 電源電圧

式(3)から，クロックをいれた場合の電力のアクティブ分(以降 Pactive)を，消費電力（以降 Ptotal）の実測値と，実験式(1')で算出したリーク電流(Ileak)を用い，式(3)の形で算出できる.

Pactive=Ptotal−Vdd × Ileak ————式(3)

電力は周波数fに比例し，電源電圧 Vdd のほぼ 2 乗に比例するため，Pactive を，実験式(4)であらわす.

Pactive=k × f × Vdd$^{\gamma}$ ————式(4)

変形すると，式(5)となる.

ln(Pactive) − ln(f) = ln(k) + γ × ln(Vdd) ————式(5)

ここで ln(k)とγを測定データから算出されるアクティブ分の電力 Pactive を使って回帰分析することで，k とγが得られる.

k=10.3, γ =2.03

アクティブ分の電力 Pactive が電源電圧 Vdd の 2 乗に比例することが確かめられたので，先に求めたリーク電流 Ileak の実験式(1)も妥当であると考えられる.

図 7は，実験式の検証である. 図の左軸は電力であり，実験式求めたスタティック分Pstatic，アクティブ分Pactiveと，その合計値の値を示す. 右軸は比を表す. 実測した電力Ptotalと，実験式でもとめたPtotalの比を，破線で示した. 誤差はおよそ±10%である.

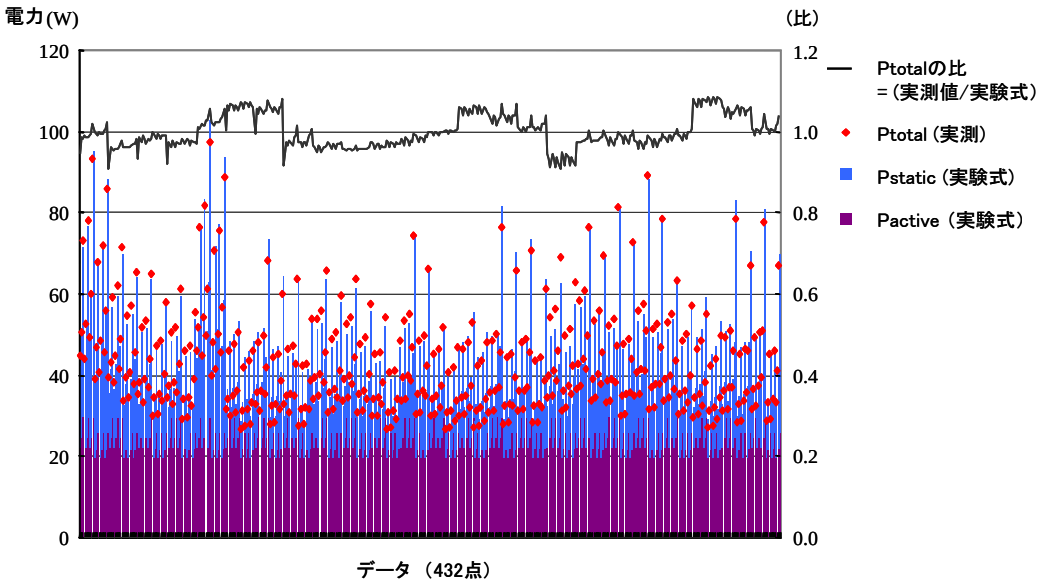

図 7. 実験式による値とその検証

以上の実験で立てた実験式は，チップレベルの特性を外側から見たものである．チップごとのばらつき，電源電圧，ジャンクション温度，動作周波数による，電力の変化をよく表している．アクティブ電力はチップごとのばらつきが小さく，電源電圧と動作周波数が決まればほぼ決まるのに対して，リーク電流はチップごとにばらつきが大きく，電源電圧とジャンクション温度に依存して指数関数的に増大する．

リーク電流の問題は，半導体の微細化とともに一般的に大きな問題となっている．Intel や AMD もその例外ではない．リーク電流は，電源を入れただけで何も働かなくても漏れ流れてしまう電流であるため，回路量の増大に伴ってリーク電流は大きくなる．リーク電流が消費電力に占める割合が無視できないレベルになったのは，富士通の場合は 90nm 世代からだが，他社でも情況はあまり変わらないようである．

リークはチップの個体差と使用条件によって大きく違ってくる．このため電源，冷却といった面で最悪の場合に備えた設計が必要になる．結果的に通常の使用では過剰なスペックとなり，筐体サイズやコストの面から好ましくない．

この点もあわせて，マイクロプロセッサが直面する最大の問題は消費電力が大きくなりすぎたことであるといえる．そのことが次に述べるマルチコア化などの，各社 CPU のマイクロアーキテクチャの方向性に大きく影響している．品種によって消費電力の違いはあり，また，搭載する装置によっても許容できる消費電力は異なるため，問題の程度には違いがある．しかし，性能に重点を置いたハイエンドシステムでは，例外なく消費電力を問題にせざるを得ない状況であり，多くはトレードオフとして周波数向上を緩める方策をとっている．

例外は IBM であり，IBM だけが次世代の Power6 で 4GHz 以上を達成するとしている．Power5 の 2.2GHz から 2 倍程度の周波数向上である．IBM の場合には，チップを Multi Chip Module(MCM)や Dual Chip Module(DCM)に高密度で搭載しており，周波数を上げるには強力な冷却が必要である．しかし，強力な冷却を前提としてジャンクション温度を低く保つことが出来るのであれば，リーク電流を小さくすることが出来る．これは，リークの，温度に対して指数関数的に増大する性質による．そしてリークが抑圧可能ならば，周波数を高くすることに伴い絶対値として電力が大きくなったとしても，電力あたりの性能が低下する割合は大きくはならない．低い性能の CPU を並べることは，オーバヘッドが大きくなって効率が悪くなる事を考えるならば，あくまでも周波数を上げるという方向性は，強力な冷却を実現できることを前提とするならば理にかなっている．今後さらに半導体の微細化が進んでいくときに，今後長期にわたって同じ戦略をとり続けることが出来るのかは興味深

い.

6. マルチコアとシステム機能の搭載

周波数向上より，集積度で性能や機能を稼ぐ方法は，業界のトレンドとして，具現化されつつある．マルチコア化とシステム機能の取り込みが，それにあたる．

6.1. マルチコア

もっとも早くマルチコア化を行ったのはIBMである．2001年末に出荷開始されたIBMのPOWER4では，MCMボードにチップを搭載して，16チップの密な結合を実現した．2CPUを1チップに搭載して，装置最大構成32プロセサを実現した．しかしIBMでのマルチコアのドライバは，消費電力の削減ではなく，絶対性能向上のために，システムを高密度に実装するマルチコア化を採ったと考える．

その他の各社がいっせいにマルチコアを採用したのは，先に述べた消費電力の問題が顕在化した 2004 年後半から本年にかけてである．

表 3. プロセサ

プロセサ	コア数 × スレッド数	周波数(GHz) (エンハンス)	電力 (W)	出荷時期
Power4	2×1	1.3 (1.9)	–	2001 年 12 月
Power5	2×2	1.9 (2.3)	120	2004 年 11 月
UltraSPARC IV	2×1	1.2	108	2004 年 10 月
UltraSPARC IV+ (Panther)	2×1	1.5	90	2005 年 10 月
UltraSPARC T1 (Niagara)	8×4	1.2	73	2005 年 12 月
Opteron 280	2×1	2.4	95	2005 年 9 月
Xeon 7040 (Paxville)	2×2	3.0	165	2005 年 11 月
Xeon 5160 (Woodcrest)	2×1	3.0	80	2006 年 6 月
Itanium2 9050 (Montecito)	2×2	1.6	104	2006 年 7 月
SPARC64 V	1×1	1.35	50	2003 年 4 月
SPARC64 V+	1×1	1.89 (2.16)	65	2004 年 8 月
SPARC64 VI	2×2	2.4	120	T.B.D.

これらの中でSunのNiagara[6]以外は，全て2コアを1チップに搭載した，いわゆるデュアルコアである．いずれも動作周波数の向上が頭打ちになり，また，性能向上技術も出尽くして，CPUの単体性能の向上が難しくなったことに関係した動きである．そのような動きのなかで，Niagaraを搭載するサーバ：Sun Fire T1000/T2000は，エコサーバという点をアピールしている．Niagaraの特徴は，マルチスレッド，マルチコアだけではなく，メモリを直付けして，1 チップでシステムが出来てしまう点にある．Niagaraそのものは単一コア性能が低く，特に浮動小数点演算はほとんど使い物にならないレベルであるから，汎用性を損ない用途が限定されるものの，今後のトレンドを考える上では大きな意味があると考えられる．

AMD や Intel がクワッドコアについて発表するなど，今後さらに 4～8 コアにマルチコアを推し進める動きもあるものの，まだしばらくは各社ともロードマップを書き換えながら，ニーズとコストの最適点を探る動きが続くと考

[6] Niagaraは，マルチコアとマルチスレッド技術と組み合わせて，32 スレッド(4 スレッド×8 コア)並列．

える. ソフトウェアのライセンスの問題も, コアに対して性能に見合った比率をかけて課金する方向が Oracle などから打ち出されて一応の決着を見ているが, 今後の動向を追いかけていく必要がある.

我々は, ビジネスアプリケーションの性能向上のために, SPARC64 VI で Vertical Multi Threading(VMT)×デュアルコア, SPARC64 VII では Simultaneous Multi Threading(SMT)×4 コアと, マルチスレッド技術と組み合わせたマルチコア化を進めている. SPARC64 VI は Sun との共通プロダクト機に搭載される.

このように私たちもマルチコア化は無くてはならないと考えている. しかし, これは単一スレッド性能を犠牲にするものではない. なぜならばハードウェアは買い換えればよいが, これまでに蓄積されたソフトウェア資産は大きく, 新たなソフトウェアを開発することは容易ではない. マルチコアがあらゆる場合に有効となるためには, ソフトウェアの革命が必要であり, さらにその革命の成果が行き渡る必要がある. 我々は, 顧客の資産を守るということを主眼とし, マルチコア化を堅実に進めていこうと考えている.

6.2. システム機能の搭載

メモリウォール問題は, CPU 性能とメモリシステムの性能のギャップの拡大に伴い, 従来から課題となっている. マルチコア化はこの問題の解決にはならないどころか, スループットの観点ではますます問題が大きくなる. メモリの大容量化に対する要求がなくなることはなく, 高密度化と併せて, 素子そのものの高速化を図ることは限界がある. そのうえ, チップ間の伝送は基板やソケットの物理的な性質と距離による制約があるため, メモリアクセスの高速化は難しい.

そこで性能向上を, システムコントローラやメモリインターフェイスを CPU チップに内蔵してチップ間伝送を減らして実現することが, 有効な手段となる. これはシステム全体の部品数削減にもつながり, コスト削減にも貢献する. 更に, システム全体の消費電力も削減できる.

先に説明した IBM は, プロセサのチップにデュアルコアとともにシステムの結合機構やメモリコントローラを内蔵して高密度, 高スループットのシステムを実現している. また, AMD の Opteron は Dual Inline Memory Module(DIMM)インターフェイスと汎用の結合機構であるハイパートランスポートを内蔵し, 4 チップないし 8 チップのコンパクトな構成で高性能のシステムを実現している. Niagara が 1 チップでシステムが構成できるのは先に述べた. これらのプロセサを搭載したシステムは, 性能ないしは消費電力, あるいはその両方で優れている.

我々は, これまで小規模から 128CPU までの超大規模サーバまでを, 一品種のプロセサでまかなってきた. しかし今後は, システム構成を限定してでもシステム機能の搭載は必須と考える.

7. 開発手法について

私たちのチームは過去から継承し発展させてきた設計手法を持ち, その構成メンバーの一人一人が高いモチベーションを持って自律的に仕事を進めていく点に特徴があり, このことが確実に開発をやり遂げる上で大きな力となっている.

新機種の開発では, 方式や仕様について検討している期間には, 第三者の目に見える形でのアウトプットはない. しかし, この期間は, 設計者の頭の中では構想が形作られる熟成されていく. そして, 論理入力の初期は, 前機種の回路を用いるか, 新規に論理を入力するかのいずれにしても, ゲートレベルで, 頭の中で考えたことが回路としてまともなものになるかの感触を確かめる. いくつかの論理ブロックをいじりながら, 構想に戻っては考え, ゲートレベルで確認することを繰り返す. ゲートレベルで実装との関連も把握しながら進めるが, 場合によっては, 捨てるつもりで実際にゲートの配置までやってみる. こうして初期に非常に大きな工数をかけてフィージビリティを確認すると同時に, 詳細設計が既に始まっている. 泥臭いようだが, 設計者が, 開発の初期にフィージビリティの感触をつかむことで, 設計完了間際の破綻が避けられる.

論理設計にかける工数に比べて，実装や検証にかかる工数は遥かに大きい．もちろん実装をやっている間もずっとディレーを改善したり，レーシングを取るために，回路を組み替えたり，パワー調整やバッファ挿入を行っていく．実装は何度も手直しを繰り返しながら追い込んで仕上げていく．

構想の段階で積み上げた資料や，あるいは一般的には有効な Hardware Description Language(HDL)でハイレベルの記述を行ったとしても，そこでできたものは，まだペーパーマシンに過ぎない．実装して使える回路となって，はじめて論理が確定するのであり，論理入力の終了時点では，大概の場合，初めの構想に対し相当な手直しが入っている．それを構想段階の資料や HDL に反映するのは，2度手間 3 度手間となり設計効率を著しく損なう．

反対に，ゲートレベルで読みやすくてきれいな図面を作ることを徹底してやる事は，大きな価値があり重要である．回路には文章では表現しきれない細かい情報があり，前機種からコピーするばかりでなく，新たな機能の追加や改善を企てるための最高の情報源となる．論理図面が残っていれば場合によっては過去に捨て去られたものを現在に甦らせることも可能である．

それぞれの設計者が回路図を見て考え，設計を進めていくことから，自律的な仕事のスタイルが定着する．ルーチンワークやツール化には程遠く，評価尺度をうまく表現できない種類の優劣がある．いきおい設計者は職人気質ともいえるような発言が多い．本人の担当部分の周囲のこれと関連する部分の担当者とは日常的に密なやりとりが繰り返されるが，議論というよりは阿吽という雰囲気である．

このようにスタセルベースの設計に最大の特徴があるが，SRAM マクロなどのカスタム設計についてもスキルを向上させてきた．特に微細化した半導体で相対的に大きくなるばらつきに対して動作マージンを確保するためには，カスタム設計の高いスキルもなくてはならない．

半導体の微細化が進展する中で，ハイエンド設計では設計量が膨大となり開発のリスクは高まる一方である．この中で私たちの設計手法やこれを支える設計者の高いスキルがますます重要になっていると考える．

8. 謝辞

過去の開発の歴史を調べるうち大先輩である三輪 修氏のホームページにたどり着いた。三輪氏からは応援のメッセージまで頂き、多くの情報をまとめておいて下さったこととあわせて感謝いたします。

論文の執筆にあたり、助言や文章の手直しを行ってくれた富士通の坂本 真理子氏に感謝します。

最後に、今もプロセサの開発に全力を傾けて頑張っているエンジニアたちに感謝します。

9. おわりに

我々は，UNIX サーバの CPU：SPARC64 V を，メインフレームの CPU をベースに開発した．そしてメインフレームから継承してきた技術によって達成したデータインテグリティや高い信頼性は，オープンサーバ製品の中で，差別化要素の源泉となっている．集積度が低い時代に，1 チップに CPU を入れるという回路量の強い制約を受けていたマイクロプロセサにとって，半導体集積度の急速な伸びは大きな恩恵であり，高性能を実現するための技術を取り入れ，メインフレームとの差を見えにくくした．一方で，高度な微細化や高い集積度は，ソフトエラーの確率を高めており，社会の重要なインフラを担うシステムの信頼性技術の重要性が，あらためて注目されてきている．この技術は同じように基幹業務システムの重責を担ってきたメインフレームで培われたものであり，演算器のエラー検出と命令リトライを実装しているのは，SPARC64 V/VI と，富士通のメインフレーム，IBMのメインフレーム（Zシリーズ）だけである．

今はまさにコンピュータは，CMOS 半導体の微細化によってもたらされた恵まれた状態から一転し，危機に直面していると言って良いだろう．だから近々の課題ばかりを追いかけていても，将来へ繋がる方向は見えな

い．また，急進的な方向付けは却って必然性を阻害する．なぜなら，ブレークスルーは製品としての進化の速さを外から見て結果論を言っているに過ぎず，その技術は，人間の経験と努力という時間を積み重ねで生み出されるからである．

これまでにもそうであったように，長い時間をかけて経験と反省を繰り返しながら練り上げられてきた技術には，たやすく覆せないものがある．その一方で，薄れてしまった必然性には，新たな技術との結びつきが求められる．将来の方向性について具体的な姿を描くことは容易ではない．しかし，必然性と出会いを求める旅は苦しくとも必ず喜びをもたらしてくれると考えている．

参考文献

[1] 雑誌 FUJITSU 1974 年　VOL. 25 NO. 7(FACOM230-75 特集号)

[2] 雑誌 FUJITSU 1976 年　VOL. 27 NO. 4(FACOM M シリーズ特集号)

[3] 三輪　修(http://homepage2.nifty.com/Miwa/)

[4] David L. Weaver, Tom Germond, “The SPARC Architecture Manual Version 9,” PTR Prentice Hall, ISBN0-13-099337-5, 1994.

[5] 井上 愛一郎 “UNIX サーバ用プロセッサ：SPARC64 V,” 雑誌 FUJITSU 2002 年 VOL.53 NO.6(サーバ：特集)

[6] Mariko Sakamoto, Akira Katsuno, Aiichiro Inoue, Takeo Asakawa, Haruhiko Ueno, Kuniki Morita, and Yasunori Kimura, ″Microarchitecture and Performance Analysis of a SPARC-V9 Microprocessor for Enterprise Server Systems”, In Proceedings of HPCA9, pp. 141-152, Feb. 2003.

[7] Mariko SAKAMOTO, Akira KATSUNO, Aiichiro INOUE, Takeo ASAKAWA, Kuniki MORITA, Tsuyoshi MOTOKURUMADA, and Yasunori KIMURA, ″Design Development of SPARC64 V Microprocessor,″ In IEICE TRANS. INF. & SYST., VOL.E64-D, NO.10 OCTOBER 2003.

[8] Aiichiro Inoue, “Fujitsu’s new SPARC64™ V for Mission Critical Servers,” Microprocessor Forum, October 15, 2002.

[9] Takumi Maruyama, “SPARC ™ VI: Fujitsu’s Next Generation Processor,” Microprocessor Forum, October 14, 2003.

[10] Aiichiro Inoue, “SPARC64™ for Mission-Critical Servers,” Fall Processor Forum, October 5, 2004.

[11] Takumi Maruyama, “SPARC64™ VI/VI+ Next Generation Processor,” Fall Microprocessor Forum, October 25, 2005.

[12] Aiichiro Inoue, “SPARC64™ VI: A State of the Art Dual-Core Processor,” Fall Microprocessor Forum, October 10, 2006.

[13] 引地 徹，加藤 哲，大田 秀信，嘉喜村 靖，“64 ビット RISC プロセッサ：SPARC64 GP,”雑誌 FUJITSU 2000 年 VOL.51 NO.4.

[14] “ブランチヒストリを持つ命令実行処理装置,” 特許第 3335379 号. 1992 年 9 月 9 日 出願.

[15] “情報処理装置,”特許第 3469469 号. 1998 年 7 月 7 日出願