論文 / 著書情報
Article / Book Information

題目(和文)	小規模マルチコア技術および投機実行用リカバリ機構を用いた低電力、高信頼プロセッサの研究
Title(English)	
著者(和文)	安藤壽茂
Author(English)	
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:乙第3870号, 授与年月日:2006年5月31日, 学位の種別:論文博士, 審査員:
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:乙第3870号, Conferred date:2006/5/31, Degree Type:Thesis doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

小規模マルチコア技術および投機実行用リカバリ機構を用いた低電力，高信頼プロセッサの研究

安 藤 壽 茂

目次

第 1 章　序論

1.1. 研究の背景

プロセッサは，半導体の微細化によるトランジスタの動作速度の向上を主因としたクロック周波数の向上と，微細化によるトランジスタ集積度の向上（Moore の法則）を利用して性能を向上してきた。しかし，素子の微細化にともなう消費電力の増大や，誤動作の確率の増大が大きな問題となってきている。

1.1.1. 微細化とそれに伴うプロセッサの性能，電力の推移

プロセッサは，大量のトランジスタを使用し命令実行の並列度を向上させる

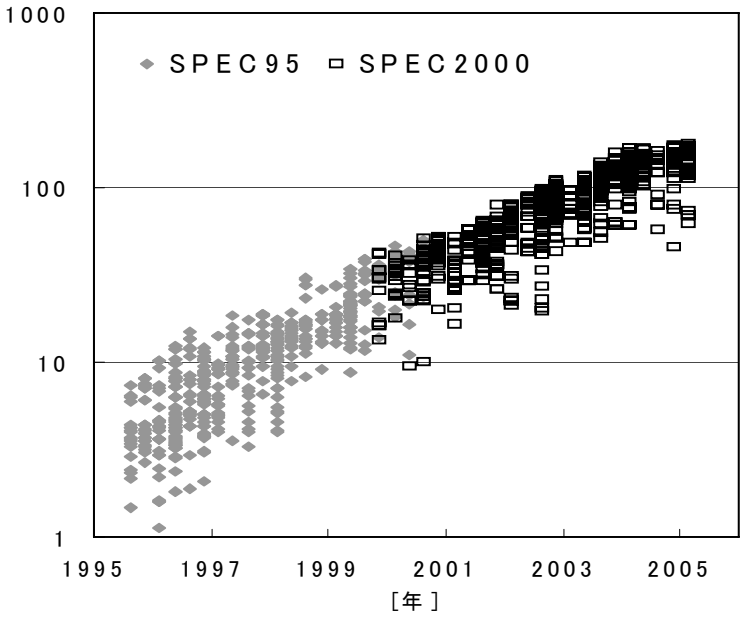


図 1.1 SPECint 性能の年次推移

アーキテクチャを用いることにより性能向上を追及してきた。しかし，クロック周波数の向上と大量のトランジスタ使用の結果として，近年のハイエンドマイクロプロセッサの消費電力は 100W～150W になっており，かつてのバイポーラ ECL LSI の発熱を上回るレベルになっている。このため，今後のマイクロプロセッサは，発熱が制約となり性能を向上させられないという状況になっている。従って，今後のプロセッサの性能向上を図るためには，低消費電力と高性能化を両立できる技術を確立することが必須となっている。

図 1.1 に SPECint ベンチマーク[6]で測定したプロセッサ性能の年次推移を示す。この図に見られるように，最近では若干鈍化傾向が見られるが，過去 10 年間で約 30 倍のペースで性能向上がなされている。なお，SPECmark は 2000 年までは CPU95 ベンチマークが用いられ，それ以降は CPU2000 ベンチマークが用いられている。これらのベンチマークは内容が同一ではないため正確には性能は比例関係にはないが，ここでは年次推移が連続になるように比例係数を決めて一つのグラフに表示している。

ISSCC(International Solid-State Circuit Conference)で発表されたプロセッ

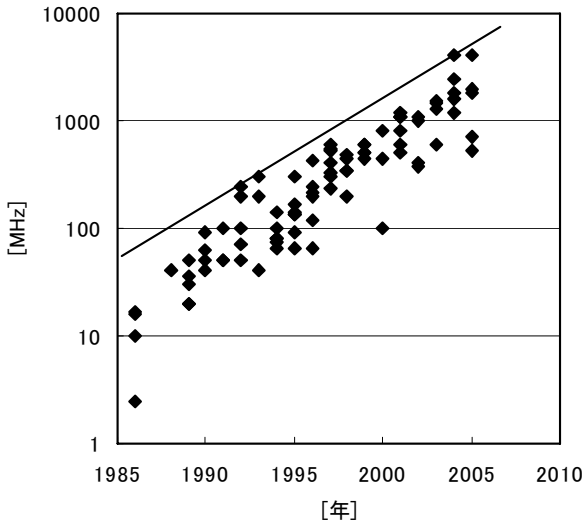


図 1.2 ISSCC で発表されたプロセッサのクロック周波数の年次推移

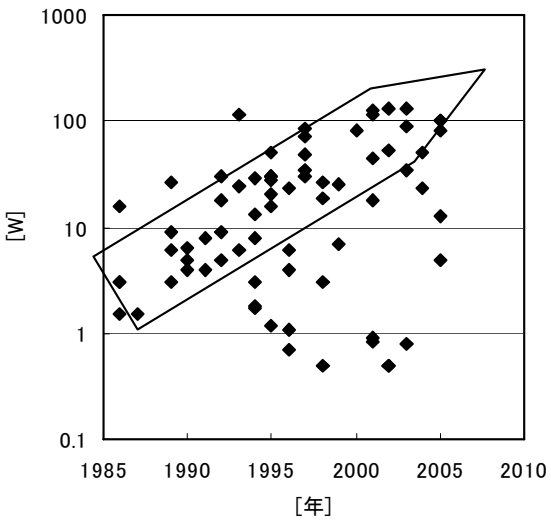


図 1.3 ISSCC で発表されたプロセッサ消費電力の年次推移

サのクロック周波数の推移を図 1.2 に，消費電力の推移を図 1.3 に示す。ISSCC で発表されたプロセッサは，高性能のものだけでは無く小電力の組み込み用プロセッサも含んでいるために消費電力には大きなばらつきがあるが，矢印で囲んだ高性能,高電力プロセッサを見ると 1990 年以前には 30W 以下であったが，2000 年代に入り 100W を超えるプロセッサが作られており，10 年間に約 5 倍のペースで電力が増加している。

図 1.3 に見られるように消費電力が増えているのは,半導体が微細化されるにつれてトランジスタ数が増加し，かつ，クロック周波数が向上したことが主因である。一方，トランジスタ数の増加とクロック周波数の向上がプロセッサの性能向上の主因であり，半導体の微細化を根本原因としてプロセッサの性能向上と消費電力の増加は直接結びついている。

ITRS（International Technology Roadmap for Semiconductors） 2004 Update[7]によると，図 1.4 に示すように，今後も微細化が進展すると想定されている。図 1.3 において微細化に伴う消費電力の推移を単純に延長すると 2010 年の高性能マイクロプロセッサの消費電力は 500W～1KW になると予想される。

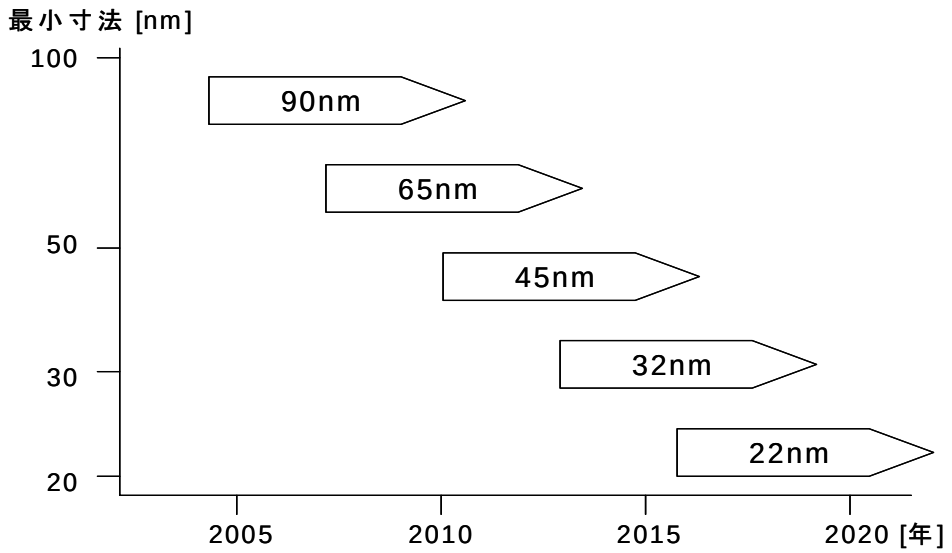


図 1.4 ITRS ロードマップ
微細化世代の製造開始年

しかし，現実的な方法では 200W 程度までしか冷却できず，電力を抑えるために性能向上が制限されるようになってきている。

更に，温暖化の進行を抑えるための消費エネルギー削減の要請や，電気料金抑制の観点からも低電力化について強い要請があり，マイクロプロセッサの性能/電力比の改善は重要課題である。

1.1.2. LSI の誤動作と高信頼化の必要性

半導体の微細化に伴い，トランジスタ内部の電界強度を許容範囲内に抑えるためには電源電圧の低下が必要となる。しかし，一般に CMOS 回路では電源電圧を下げると信号振幅が低下し，雑音マージンも比例して減少する。更に，微細化に伴い相対的に素子の製造ばらつきが増加する傾向にあり，回路特性のばらつきにより雑音マージンは一層減少する。一方，雑音は R x di(電流変化)や L x di/dt（電流変化率）に比例し，低電源電圧化による電流量の増加と回路が高速動作し dt が減少することにより増加傾向にある。

また，半導体チップやパッケージに含まれる微量の放射性同位元素に起因するアルファ線や宇宙線起因の中性子のヒットにより発生する電荷注入による誤動作に関しては，電源電圧の低下と微細化による寄生容量の減少により蓄積電荷が減少することから，微細化に伴い急速に耐量が減少する。このため，マイクロアーキテクチャにより信頼性向上をはかることが重要になってきている。

A. アルファ線や中性子ヒットによる誤動作

1978 年に DRAM メモリがα線のヒットにより誤動作することが報告[1]されて以来，α線や中性子のヒットによる半導体回路の誤動作は信頼性にとって大きな問題として認識されている[2]。

α線や中性子が半導体の原子に衝突すると電離を起こして電子と正孔のペア

が生じる。N チャネルトランジスタのドレイン拡散に正の電圧が印加されていると，電離で発生した電子が半導体内部の電界により移動してドレイン電極に注入される。これは電子回路的には負の電流パルスがドレイン電極に印加されたことと等価であり，ドレイン電圧が低下する。そして，この電圧低下量が大きいと回路の誤動作を引き起こす。一方，P チャネルトランジスタの場合は正孔がドレインに注入され，同様に誤動作を引き起こす。このような誤動作は一過性であり，素子を破壊するものではなく，Single Event Upset (SEU，単発エラー)，あるいはソフトエラーと呼ばれる。

半導体の微細化により，回路を構成するトランジスタや配線が微細化され，ドレインノードの寄生容量が小さくなる。一定量の注入電荷による電圧変動は寄生容量に逆比例するので，微細化にともないより大きなノイズパルス電圧が発生することになる。また，許容できるノイズ電圧は一般に電源電圧に比例し，また，トランジスタの高速応答性が改善するので，微細化により小さなノイズでも誤動作を起こしやすくなる。

一方，微細化により 1bit のメモリセルの面積も減少し，α 線や中性子がヒットする確率が減少するという効果があり，単位面積あたりの照射量が一定の場合のビットあたりのエラー率はほぼ一定で推移[3]してきている。しかし，微細化により単位面積あたりのビット数は増加するので，シリコンチップの単位面積あたりのエラー頻度は集積密度にほぼ比例して増加する傾向にある。

B. マイクロプロセッサ回路の誤動作

近年のマイクロプロセッサは，キャッシュメモリとして大量（数 MB から 10MB 以上）の SRAM（Static Random Access Memory）メモリを搭載しており，エラー頻度としてはこの SRAM のエラーが大部分を占める。しかし，大容量 SRAM に対しては ECC（Error Correction Code）を付加し訂正を行う手法が一般に用いられており，SRAM のランダムな 1 ビット誤りがシステムのエラーに繋がる確率は無視できるようになっている。しかし，マイクロプロセッサ

の中では ECC の付加されていない小規模なメモリも各所に使われており，これらのメモリのエラーに対する対策が必要である。

メモリのエラーが問題視された 1980 年代は，論理回路はサイズが大きく寄生容量が大きいので，その誤動作は問題視されなかったが，最近の 90nm 世代の論理回路は 0.35μm～0.25μm 世代のメモリセルと同じ程度の面積，蓄積電荷量になっており，中性子ヒットによる論理回路（ラッチ，FF）の誤動作の確率はメモリセルと同程度と報告[4]されている。

本研究で取り上げた SPARC64 V プロセッサは約 200K 個の FF（FlipFlop）を使っている。これらの FF は各 2 個のラッチを含むが，一方は記憶保持状態，他方はパス状態でありゲートとして動作している。保持状態のラッチの中性子ヒットによるエラー率は，ラッチの設計に大きく依存するので一概には言えないが，典型的な値として 0.001～0.01fit（1fit : 1 故障/10^9 時間）を想定すると，チップ全体では 200～2000fit となる。

チップ全体のエラー率としては組み合わせ回路のエラー率を加える必要があるが，組み合わせ回路に注入されたノイズは伝搬途中で消滅する場合が多く，また，次のラッチまで到達してもラッチされるタイミングで到着しなければエラーにはならないので，ラッチや FF に比べるとエラー率は低くなる。SPARC64 V マイクロプロセッサでは組み合わせ回路に使用されているトランジスタ数とラッチ，FF に使用されているトランジスタ数は同程度であるが，前記のように組み合わせ回路のエラー率は低く，チップの平均故障間隔はほぼラッチのエラー率で決まると考えられる。

200～2000fit のエラー率は 50 万～500 万時間の平均故障間隔に相当し，約 57 年～570 年に 1 回のエラーである。この時間は一個のプロセッサチップを用いるシステムとしては妥当な水準であるが，数十，数百個のプロセッサチップを用いる大規模システムでは毎年 1 回程度の故障が発生し，このような大規模システムは社会や企業の基幹業務を行うインフラストラクチャであることを考えると許容できない水準である。また，微細化が進むにつれて故障率が増加す

るという問題があり，アーキテクチャ的な研究により，システムとしてのエラー率を低減し信頼度を向上することが必須である。

1.2. 従来の研究

1.2.1. マイクロプロセッサの消費電力

A. 等電界スケーリング

微細化による MOS トランジスタのスケーリングは, 寸法の縮小と同一の比率で動作電圧を低減する等電界スケーリングが基本的な考え方であり，等電界スケーリングにより消費電力がどのように変化するかを考察する。

MOS トランジスタのゲート容量は，次式で表される。

$$C_g = \varepsilon \times L \times \frac{W}{t} \qquad \cdots(1.1)$$

ここで ε はゲート絶縁膜の誘電率，L はゲートの長さ，W はゲート幅，t は絶縁膜の厚みである。ここで L，W， t を全て s 倍にスケールするとゲート容量 Cg も s 倍となる。一方，単位面積あたりに集積できるトランジスタ数は $1/s^2$ 倍であるので，結果として単位面積あたりの総ゲート容量は 1/s 倍になる。

トランジスタの寄生容量としては，ゲート容量以外にソースやドレインの拡散容量があるが，ゲート容量に比べて小さいこと，及び，微細化に伴い不純物濃度を上げる必要があり単位面積あたりの拡散容量も増加する傾向にあることから，トランジスタの寄生容量は 1/s に比例すると近似する。

また，配線についても，配線の幅と厚み，配線間隔，層間の絶縁膜の厚みを全て s 倍に縮小すると，単位長あたりの配線容量は一定である。しかし，一定の面積に収容できる総配線長は 1/s 倍になり，単位面積あたりの配線容量は 1/s 倍となる。配線容量の低減のために低誘電率の絶縁膜が開発され単位長あたりの配線容量を低減する効果があるが，一方，配線の抵抗増加を抑えるためにアスペクト比（金属配線の高さ/幅）を大きくすることによる隣接配線間の容量増加，配線層数の増加という単位面積あたりの配線容量増加要因もある。

従って，寸法を s 倍にする微細化により単位面積あたりの全容量はおおよそ

1/s 倍となるとみなす。

一方，MOS トランジスタのゲート酸化膜やチャネルの電界を一定にする等電界スケーリングでは，寸法を s 倍にすると電圧も s 倍にする必要がある。この等電界スケーリングにより，トランジスタのスイッチ時間はおおよそ s 倍に高速化される。

スイッチングに伴う充放電による電力消費は，よく知られているように次式であらわされる。

$$p = \frac{1}{2} \times C \times V^2 \times f \qquad \cdots(1.2)$$

ここで充放電される C は等価スイッチ容量であり，微細化により 1/s 倍となり，電圧 V は s 倍，f はスイッチ時間の逆数であり 1/s 倍となるので，結果として，等電界スケーリングを行うと単位面積あたりの消費電力はほぼ一定である。

B. 現実のスケーリング

しかし，現実のマイクロプロセッサでは，パイプライン段数の増加などのマイクロアーキテクチャ的な改善や回路設計の工夫によりクロック周波数は年率 25%向上(図 1.2)しており，これはスケール係数 s が半減する 6 年の間に 3.8 倍のペースである。このクロック周波数の向上のペースは等電界スケーリングから予想されるペースの約 2 倍であり，単位面積あたりの消費電力が増加する原因となっている。

また，トランジスタ間の分離（アイソレーション）法の改善などの半導体技術の改善や CAD ツールの改善などによりプロセッサチップのトランジスタ数は 6 年間で 13.9 倍に増加しており，消費電力の増加に拍車をかけている。

更に，電源電圧は二乗で消費電力に影響するため，微細化に伴う電源電圧の低下は消費電力の増加を抑える重要な要因であったが，以下に述べるようにリーク電流を抑える必要から，130nm 以降の世代では電源電圧の低下がスケーリ

ングに追随せず，下げ止まりの傾向にある。

図 1.3 において矢印で囲んだ電力の大きい汎用ハイエンドマイクロプロセッサをみると，1985 年から 2000 年までの 15 年間に消費電力は 20 倍に増加している。この間のチップ面積の増加は 2~3 倍であり，単位面積あたりの消費電力は 10 倍近く増加している。これは等電界スケーリングの単位面積あたりの消費電力は一定という傾向とは乖離している。

微細化を利用して，トランジスタ数を増加させ，クロックを上げるという従来の設計方針を続けると，矢印で示した電力増加傾向が続くことになる。しかし，図 1.3 に見られるように，消費電力の大きいプロセッサでは既に 150W 程度となっており，空冷の限界に近い値である。従って，プロセッサの性能向上を続けるためには，低電力と高性能を両立させる設計手法の確立が必須である。

C. リーク電流の影響

スケーリングにより高速化を達成するには，低下する電源電圧に比例してトランジスタのスレッショルド電圧（Vth）を下げることが必要である。しかし，ゲート電圧を 0V とした時のサブスレッショルドリーク電流は次式で表され，Vth の低下に伴い指数関数的に増加する。

$$I_{leak} = I_0 \times \exp\left(\frac{-V_{th}}{S}\right) \qquad \cdots(1.3)$$

ここで S は，最近の短チャネルトランジスタでは 35mV 程度の値であり，Vth が 90mV 程度減少するとリーク電流は 10 倍となる。また，I_0 はトランジスタの構造と温度などで決まる定数である。

表 1.1 ITRS ロードマップのサブスレッショルドリーク電流と電源電圧の推移

		1999 年	2001 年	2004 年	2007 年
テクノロジ世代		180nm	130nm	90nm	65nm
ITRS2002	リーク電流 [nA/μm]	7*	10	100	1000
	電源電圧 [V]	1.5～1.8*	1.2	1.0	0.7
ITRS2003	リーク電流 [nA/μm]			50	70
	電源電圧 [V]			1.2	1.1

*ITRS2000 による

表 1.1 に ITRS ロードマップのチャネル幅 1μm のトランジスタのリーク電流の推移を示す。2002 年以前の ITRS ロードマップによると 1999 年の 180nm プロセスのトランジスタのサブスレッショルドリーク電流は 7 nA /μm であるが，2004m 年の 90nm プロセスのトランジスタでは 100 nA /μm と 14 倍になり，2007 年の 65nm プロセスのトランジスタでは 1000nA/μm と更に 10 倍になっている。Nowak[8]は過去に設計された LSI の動作電力とリーク電力の推移をプロットし，Lgate=50nm 程度になると動作電力とリーク電力が同程度となり，それ以上微細化が進むとリーク電力の方が大きくなるとリーク電流の問題を指摘している。

このサブスレッショルドリーク電流の増加傾向は過大であり実用上問題であるとの反省から 2003 年の ITRS ロードマップでは方向修正が行われ，2004 年の 90nm プロセスでは 50 nA /μm，2007 年の 65nm プロセスでも 70nA /μm とリーク電流の増大を抑えるトランジスタ設計に路線変更された。しかし，高速化のトレンドを維持するため，電源電圧を 2004 年は 1.0V から 1.2V，2007 年は 0.7V から 1.1V へと変更した。このため，2002 年以前のロードマップで大きな問題であったサブスレッショルドリーク電流の急増はある程度回避されたが，

電源電圧の低下傾向がスローダウンし等電界スケーリングからは更に離れる傾向となり，動作電力が急増するロードマップとなった。

また，微細化にともないトランジスタのゲート酸化膜が薄くなり，1.5nm を下回るようになると，ゲート酸化膜をトンネル効果で突き抜けるゲートリーク電流が急増する。高誘電率材料を用いた絶縁膜を用いることにより， SiO_2 換算の電気的膜厚は減少させるが実際の膜厚は厚く保つことによりトンネル電流を減らす方法が本質的な解決であるが，高性能トランジスタに適用可能になるにはまだ時間が掛かる状況である。

従って，プロセッサの消費電力を考えるに当たっては，動作電力に加えて，サブスレッショッルドリーク，ゲートリーク電流に起因するリーク電力の低減を考慮する必要がある。

1.2.2. 低電力化の研究

A. 低電源電圧動作

1.2 式に示すように消費電力は電源電圧 V の 2 乗に比例するので, 電源電圧を低下させることが消費電力の削減に有効な手段である。Chandrakasan 等は，1992 年の論文[21]において，電源電圧の低減による消費電力の削減を論じている。この論文では，加算器と比較器を含むデータパスを 2 組設けることにより性能が 2 倍となるので，一定のスループットを得るのに必要なクロック周波数は半分に下げることができ，それにともない電源電圧を 0.58 倍に下げることが可能となる。加算器と比較器が 2 組と付属のマルチプレクサが必要でありスイッチングする容量は 2.16 倍となるが，クロックと電源電圧の低下により全体の消費電力は 0.36 倍に減少することを示している。さらに，加算器と比較器を 2 段のパイプライン構成とすることにより，パイプラインレジスタの追加などによりスイッチングする容量は 1.15 倍になるが，各段の遅延時間がほぼ 2 倍許容できるため電源電圧を 0.58 倍に下げることが可能であり，総消費電力は 0.39

倍にできることを示している。

また，処理負荷が高く高速で動作させる必要がある場合には通常の電源電圧を用いて高速クロックで動作させ，処理負荷が低い場合は，必要最小限にクロック周波数を下げ，それに伴って電源電圧も下げるというように，適応的に電源電圧とクロックを変動させる方式[22]も提案されている。

B. リーク電力の低減

1.3 式で示すように, トランジスタのスレッショルド電圧 Vth を低下させるとリーク電流が増大する。このため，遅延時間に余裕がある部分では Vth の高いトランジスタを使用してリーク電流を減少させる手法が一般的に用いられている。また，バルク CMOS ではボディー電位を変化させ基板バイアスを掛けることにより実効的に Vth を増加させリーク電流を低減する手法も提案されている。この手法では処理負荷にあわせて基板バイアスを変化させ適応的にリーク電力を減らす[23]ことが可能である。また，リーク電流は電源電圧の 1.5～2 乗に比例し，電源電圧を下げることもリーク電力の低減に有効である。

更に，高 Vth トランジスタを電源パスに挿入した回路構成をとり，回路が長時間動作する必要がない場合には電源供給を止めることによりリーク電流を減らすことも行われている。

これらの手法を用いても，90nm 世代の半導体プロセスを使ったプロセッサ[24]では，全消費電力の 40%程度がリーク電流によるものであると報告されており，リーク電力の低減は課題である。

C. 動作率の低減による電力削減

また，回路のアクティブな消費電力は状態変化の回数に比例し，クロックに対して回路の状態変化が起こる比率（動作率）に比例する。プロセッサの各部

は常に動作する必要があるとは限らず，例えば，浮動小数点演算器は浮動小数点演算命令を実行していない状態では動作する必要がない。しかし，浮動小数点演算器にクロックやデータが供給されていると，浮動小数点演算器は不必要な動作を行い電力を消費する。

このような無駄な動作とそれによる電力消費を抑えるため，動作を必要としない回路ブロックへのクロック供給を止めるクロックゲーティングという手法が用いられる。この手法により，論文[25]では約 20%の電力削減が報告されている。

1.2.3. コンピュータハードウェアのエラー検出，訂正技術

コンピュータシステムが社会や企業の基盤を担うようになるにつれ，その誤動作や故障による停止は社会的，経済的な損失が大きくなっており，電気，電話システムなどと同様の高信頼度を要求されるようになってきている。従って，高信頼で故障を起こさない，あるいは，故障が起こっても自己修復して機能を果たしつづけるプロセッサが強く求められるようになってきている。

A. エラー検出，訂正符号

リレーや真空管を論理素子として用いていた時代には，素子の信頼性が低いためにエラー検出は必須の技術であった。コンピュータの初期の事務用計算機には 10 進法による計算が用いられており，10 進数一桁を 5bit の ${}_5C_2$ コードで表現し，加算器は ${}_5C_2$ コードを加算する論理とし，結果が ${}_5C_2$ コードとなっていることを確認する論理を設けるなどの方法がとられた[9]。また，2 進数の計算機では，データビットに 1bit を追加し，全体の 1 の数が奇数になるようにした奇数パリティーチェックなどがエラー検出に用いられた。

しかし，素子がトランジスタや半導体集積回路になるとその信頼性は飛躍的に向上し，小規模なコンピュータではエラーの発生頻度は非常に低くなり十分

な信頼度が達成できるようになった。このため，マイクロプロセッサではエラー検出を持たない設計が一般的となり，エラー検出はメインフレームなどの大型コンピュータだけで限定的に用いられる時代が続いたが，近年になりアルファ線や中性子ヒットによる論理回路のエラーが問題になり始めて，再び，論理回路のエラー検出，訂正が顧みられるようになってきた。

一方，メモリシステムはその回路量が多く故障率が高いこと，比較的微小な読み出し信号を用いノイズマージンが小さいことなどから，パリティーチェックによるエラー検出や Hamming コード[10]などの 1bit エラー訂正コード(Single bit Error Correction Code，SEC コード)，あるいは Hamming コードに対して全ビットのパリティーを 0 とするように 1bit を追加した 1bit エラー訂正 2bit エラー検出コード（Single bit Error Correction Double bit Error Detection Code，SECDED コード）などが用いられている。そして，中性子ヒットによるエラーが懸念されるに至って，マイクロプロセッサチップに内蔵されるキャッシュメモリなどにも SECDED コードが用いられるようになっている。

また, 主記憶では 8 ビット, 16 ビットなどの複数ビットの入出力をもつ DRAM 素子が用いられ，単一チップの故障が複数ビットの故障を引き起こす可能性があるため，同一チップに割り当てられる b ビットを単位として，エラー検出や訂正を行う Reed-Solomon コード[11]や金田-藤原コード[12]などの SbECDbED（Single byte Error Correction Double byte　Error Detection）コードが用いられるようになってきている。

B. ソフトエラー耐性の高い回路設計

回路的な工夫により蓄積電荷を増加[13]させて中性子ヒットによるエラー頻度を低減したり，複数ノードに情報を記憶することにより 1 ノードの情報が失われてもエラーとならない回路形式[14][15]も提案されているが，動作速度が遅くなったり，必要なトランジスタ数が増えるという問題があり，一般的に使用

するには適せず，特にエラーが問題となる部分だけに適用するという使い方がなされている。また，このような回路は，消費電力を増加させるという問題もある。

C. ハードウェア冗長

論理回路のエラーは，データの形を変えず単純に伝送する場合にはパリティーチェックなどで検出可能であるが，加算器などの演算を行う回路には適用できない。このため，前述のように $_5C_2$ コードで演算を行う加算器を用いたり，加算器と独立に入力データから和のパリティー値を計算する回路を追加し，和から求めたパリティーと突き合わせたりする方法などが用いられる。

また，単純に同じ論理ユニットを 2 組用意し，これらに同一のデータを入力し出力同士を比較する方法は，必要なハードウェアは 2 倍となり，かつ，比較回路も必要となるが，高い検出力を持つ点と，和のパリティー計算のような特殊な回路の設計を必要とせず，設計の手間が少ないという利点がある。

1.2.4. チェックストップ-リカバリ

前記のエラー検出ハードウェアで論理回路のエラーを検出した場合の対応としては，その時点で実行を中止し，誤った結果をレジスタに書き込まないという手法がとられる。前述の $_5C_2$ コードを用いるリレー計算機では，加算器の結果が $_5C_2$ コードとなっているかのチェックを行い，誤りがある場合は結果をアキュムレータに書き込まず，加算を再実行することにより単発的なエラーからの回復を行っている。また，IBM の ES/9000 メインフレームプロセッサ[65]では，エラーが検出されると実行をストップし誤った結果がレジスタに書き込まれることを防止し，それまでに誤り無く実行されたレジスタ状態をメモリ経由で別のプロセッサに転送して実行することによりリカバリを行っている。

このようなレジスタへの書き込みに先立ってエラーの検出状況をチェックす

る方式は高い信頼度が得られるが，結果の生成から書き込みの間にエラーチェックが入りサイクルタイムが増加したり，エラーチェックのためにパイプラインを追加すると，後続の命令での結果の使用が 1 サイクル遅くなったりするという問題がある。

また，処理が固定のパイプラインである場合は，パイプラインの各段で処理されている命令と各段でのエラー検出の対応が明確であるが，最近の Out-of-Order 実行を行うスーパスカラプロセッサではエラーの検出と命令の対応づけを行うことが難しいという問題がある。

1.2.5. チェックポイント-リカバリ

より粒度の大きいエラー訂正手法として，Hunt 等により Cache Aided Rollback Error Recovery[16]が提案されている。

この手法は，図 1.5 に示すように，整合の取れたプロセッサ状態とメモリの状態のセットをチェックポイントとしてメモリに格納し，その後の状態の変更はアクティブ状態としてプロセッサ内部とキャッシュに格納しメモリには影響を与えないようにする。そして，エラーが検出された場合にはアクティブ状態を破棄してチェックポイントに戻ることにより回復（リカバリ）を可能としてい

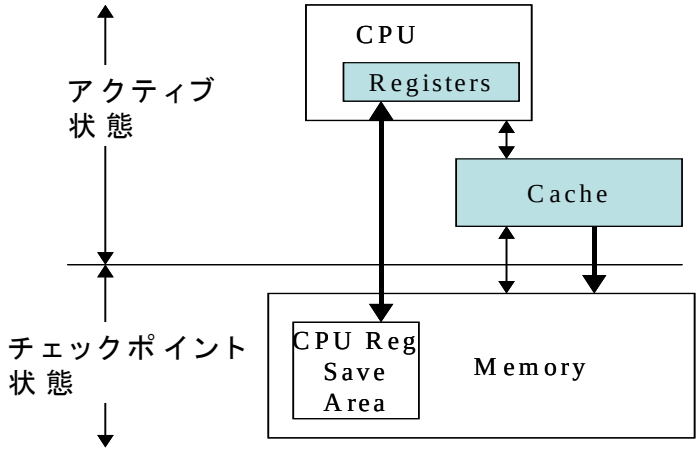


図 1.5 Cache Aided Rollback Error Recovery による状態回復

る。この方式では複数の命令の実行結果を取り消すことが可能であり，チェックストップ方式よりも長い投機実行が可能である。

これを実現するため，キャッシュメモリの書きかえられた(Dirty)ブロックをメインメモリに書き戻す必要が出た時点で，プロセッサ状態とメモリイメージをチェックポイントとして保存する。チェックポイントの作成は，実行の再開に必要なプロセッサ状態（プログラムカウンタ，汎用レジスタ等のアーキテクチャレジスタ）をメモリの特別な領域に保存し，また，キャッシュの全ての Dirty ブロックをメモリに書き出し，それらのキャッシュブロックの状態を Clean に変更するという手順で行われる。また，この手法をキャッシュコヒーレントなマルチプロセッサへ拡張する方法[18]も提案されている。

しかし，この方法では，チェックポイントの作成にあたってキャッシュに存在する全ての Dirty ブロックのメモリへの書き出しが必要であり，これに時間がかかるという問題がある。また，書き出しを行わずに Dirty なブロックを unchangeable という読み出しは可能であるが内容を変更する書き込みは不可とする状態に変更し，このブロックに書き込みが必要となった時点で，メモリへの書き込みを実施するという改良案も提案されている。これにより Dirty ブロックの書き出し頻度は減少するが，Ahmed 等の結果[17]では，この改良によっても書き出しのオーバヘッドは性能的に無視できない影響を与えていることが見られる。

また，メインフレームプロセッサでは，プロセッサのアーキテクチャ的状態を保持する全てのレジスタの状態を，チェックポイントとして R-unit と呼ぶ機構に格納し，保存している[19][20]。しかし，保存のためのレジスタファイルが必要であることに加え，毎サイクルごとに変更が発生した全てのレジスタの状態を R-unit に転送して格納する必要があり，多数の命令を並列に実行するスーパスカラプロセッサでは負担が重いと言う問題がある。

1.2.6. まとめ

以上のように，発熱が性能を制約する状況であるので，各種の低電力化技術が研究，提案されている。しかし，これらの手法を用いても消費電力の増加傾向を止めるには到っていない。低電力化には電源電圧の低減が有効な方法であるが，性能が低下することと，回路の動作マージンが減少し，中性子ヒットなどによる誤動作の確率が増加し，信頼性が低下するという問題がある。

また， プロセッサのエラーを検出する手法については多くの研究があり，また，SRAM などのエラーを訂正する ECC コードは広く実用化されているが，論理回路のエラーを検出した場合のリカバリについて提案されている手法は実装上の負担が大きく，満足すべきものではない。

このため，実行性能を低下させることなく消費電力を減少させ，同時に信頼度についても向上させることが可能な技術の研究が必要とされている。

1.3. 本研究の目的と概要

第 1 章では，本研究を行った背景と従来の研究について述べる。1.1 の背景では，プロセッサの消費電力の増大により，今後の性能向上が困難になっており，また，微細化に伴い内部雑音の増加や中性子ヒットによる誤動作の問題が深刻になりつつある状況について述べ，これらの問題に対してマイクロアーキテクチャの観点から消費電力の低減と高信頼化を行うことが必要であると述べた。

1.2 の従来の研究では，プロセッサの消費電力急増の原因を分析し，微細化に伴い等電界スケーリングを行えば単位面積あたりの消費電力は一定であるが，現実にはトランジスタ密度やクロック周波数の向上は微細化よりも早いペースで進展しており，消費電力が急増していることを述べた。そして，消費電力を低減するための電源電圧の低減や動作率の低減などの各種の研究について概観した。また，コンピュータのエラー検出，訂正技術を概観し，論理回路のエラーを訂正するチェックポイント-リカバリ方式としてこれまでに提案されたり，メインフレームで用いられたりしている方法は，多くの追加ハードウェアを必

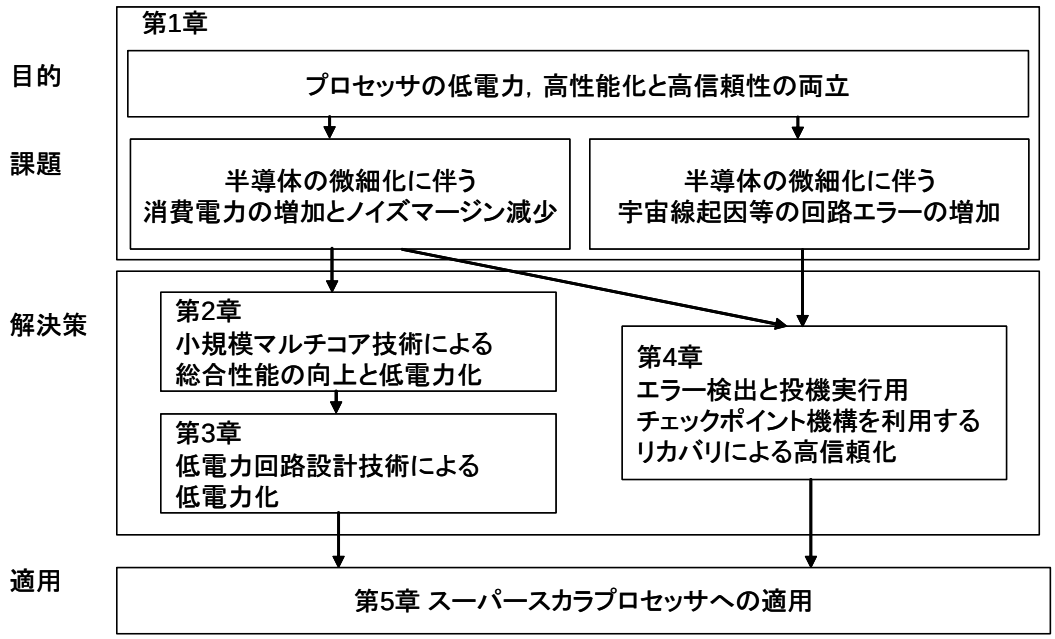


図 1.6 本論文の構成

要としたり，実行時間のオーバヘッドが大きかったりするという問題があることを述べた。従って，これらの問題を解決する高性能，低電力と高信頼化を両立する研究が必要な状況である。

第２章では，小規模マルチコア化による低消費電力，高性能化について述べる。マイクロプロセッサの性能や消費電力の年次推移を分析すると，性能は年率 40%のペースで向上しているが，クロック周波数の向上を除いたトランジスタ数あたりの性能は低下する傾向である。従って，1 クロックあたりの処理命令数を向上する複雑なマイクロアーキテクチャのプロセッサに較べて，簡素なマイクロアーキテクチャのプロセッサの方がエネルギー効率が高く，この簡素で小規模なプロセッサコアを複数集積するプロセッサチップの方が高い性能が得られるとの見通しに基づき，小規模コア化による低電力，高性能化を検討する。

トランジスタ数と性能，電源電圧とクロック周波数などの関係から，プロセッサコアの大きさをパラメタとしてチップマルチプロセッサのスループット性能や消費電力を計算するモデルを構築した。このモデルにより，複雑なプロセッサと比較して 1/4 の大きさの小規模プロセッサコアを 4 個集積する方式により，同一チップ面積で 1.5 倍のスループット性能が得られ，また，電源電圧とクロック周波数を下げ，消費電力を 1/2 以下にするとともに，複雑なプロセッサと比較して高いスループット性能を実現できることを示す。

第３章では，回路設計技術による低電力化について述べる。第 2 章で提案した小規模コアによる低電力化には電源電圧の低減が必要であり，低電源電圧で安定に動作し，かつ，所望の処理速度をもつ回路が設計できなければ低電力化は実現できない。そこで，富士通の 90nm 半導体プロセスのトランジスタモデルを用いて，スタンダードセルライブラリを設計し，電力性能指標である MIPS/W，$MIPS^2/W$ を最適化する動作条件として電源電圧 0.8V を導いた。

また，SRAM やレジスタファイルなどのマクロは通常のスタンダードセルのゲート回路より動作マージンが小さく，通常の設計では低電源電圧での動作が困難である。本研究では，これらのマクロの回路設計を行い，回路的な工夫に

より0.8Vの電源電圧での動作が可能なフルカスタムマクロの設計が可能であることを示した。

更に，ゲート長の若干長いLg=45nmのトランジスタの多用により，リーク電流を大幅に削減可能であることを示した。

第4章では，筆者らが開発した投機実行を行うスーパスカラプロセッサのチェックポイント-リカバリ機構について述べ，この機構を利用してハードウェアエラーからのリカバリ機能を追加する方法について述べる。

投機実行を行う高性能プロセッサでは，チェックポイント-リカバリ機構は投機実行にともなう分岐予測外れの修正やプレサイズ割り込み（Precise Interrupt)を実現するために必須の機構である。本研究では，投機実行用に既に存在するチェックポイント-リカバリ機構をハードウェアエラーの訂正にも用いる方式を提案する。ハードウェアのエラーを検出すると実行中の仕掛り状態にある命令を中断してクリアし，投機実行用チェックポイントに戻って再実行を行うことにより単発故障からのリカバリを行う。この方式は，従来の方式と比較して，追加のハードウェア量，実行時間ともに殆ど負担がなく，投機実行を行うプロセッサに対して容易に実装することが出来るものである。

第5章では，富士通の高性能unixサーバ用プロセッサであるSPARC64 Vプロセッサ[37][38][39][40]を取り上げ，小規模マルチコア化と回路技術による低電力化の組み合わせによる消費電力の低減，性能/電力の改善に関するフィージビリティースタディーについて述べる。

SPARC64 Vプロセッサの設計をベースラインとし，基本的な構造を保ちながら，チップ面積と性能のトレードオフを考慮してハードウェア量の削減を行い，ベースラインの約1/2の面積で70%程度の性能を持つ小規模コアが作れることを示す。また，第3章で設計したカスタムマクロを適用することにより，プロセッサコアの面積はベースラインの40%に減少し，ベースラインプロセッサより1世代微細化された90nm半導体プロセスの採用により6コアを集積したプロセッサチップが実現できると見積もられる。

この小規模コアによるマルチプロセッサの性能スケーラビリティーを検討し，DRAM アクセスのボトルネックなどを解消した構造を設計し，同じ微細化を適用したベースラインプロセッサを含む各種の設計との性能や消費電力の比較を行った。小規模マルチコア化と低電力回路の組み合わたチップマルチプロセッサは，90nm 化したベースラインプロセッサと比較して，40%の消費電力で 24%高いスループット性能が実現可能であることを示す。

この低電力化は，小規模マルチコア化による性能向上分をクロックと電源電圧の低減に廻すことにより達成されており，一般的には，電源電圧化を低下させることによる回路の動作マージンの低下のため誤動作が増加するという問題がある。しかし，投機実行用チェックポイントを利用するハードウェアエラーリカバリを組み合わせることにより信頼度の改善を行い，結果としてハードウェアエラーリカバリを行わないプロセッサを元の電源電圧で動作させた場合に比べて高い信頼度を実現している。

また，第 5 章の後半においては，投機実行用チェックポイント-リカバリ機構を用いるハードウェアエラー訂正機構の実装について述べる。SPARC64 V プロセッサのエラー検出機構について述べ，続いて投機実行とそのチェックポイント-リカバリ機構について述べる。そして，投機実行用のチェックポイント-リカバリ機構をハードウェアエラーからのリカバリ機能にも共用する実装について述べ，このエラー検出とリカバリによる信頼度の向上について述べる。

第 6 章では，纏めとして，低消費電力化と信頼度向上を両立することを可能とする小規模マルチコア技術と投機実行用チェックポイント-リカバリ機構を利用したハードウェアエラーからのリカバリ技術を確立したことを述べる。

1.4. 本研究の解決方法の特徴

背景において低電力化と高信頼性の達成の必要性を述べたが，マイクロプロセッサの設計においてこれらは独立の課題ではなく，両方をバランス良く達成する必要がある。このためには，両方の解決案がシナジー効果をもつことが必要である。

本研究で提案する低電力化は，小規模マルチコア化と低電力回路技術の組み合わせによるものである。また，高信頼化については，投機実行用のチェックポイント-リカバリ機構をハードウェアエラー訂正にも共用する方式を提案している。

図 1.7 に本研究の目的と提案する解決方法の関係を示す。実線の矢印は改善効果，破線の矢印は目的の達成を妨げる方向に働く悪化効果を示す。

図 1.7 に示すように高性能，低電力化は，第 2，第 3 章で提案する小規模マル

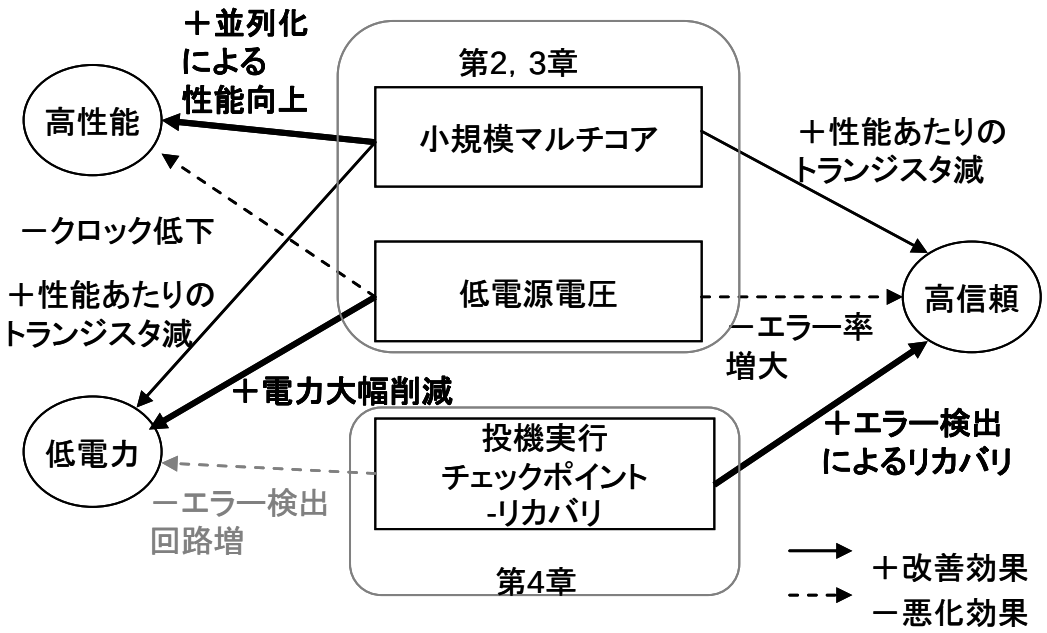


図 1.7 本研究の解決方法のシナジー効果

チコア化と低電力回路技術で達成されている。大幅な低電力化には電源電圧の低下が必須であるが，電源電圧を低下させると回路の動作速度が低下し，クロック周波数を低下させる必要があり，性能が下がる。また，信頼度に関しては，電源電圧 1.0V での動作と比較して 0.8V での動作では，中性子ヒットによる回路のエラー率は 1.5～2 倍に増加すると報告されている。従って，低電源電圧化による消費電力の低減だけを単独で適用すると性能が低下し，かつ，信頼度も低下するという問題がある。

小規模マルチコア化は，小規模化によりトランジスタ数あたりの性能が高いプロセッサコアを作ることが可能であることから，同一面積により多くのプロセッサコアを集積し並列化により高性能化を行うことが出来る技術である。小規模マルチコア化による高性能化と低電力化のトレードオフにより電源電圧をどこまで低下させるかを決定することにより，低電源電圧によるクロック低下の影響を含めても，チップあたりの総合性能を向上させ高性能化を実現する方法を提供している。

また，小規模マルチコア化は性能あたり必要な回路量が小さいので，性能あたりで見ると中性子ヒットなどによるエラーが低下し高信頼化にも貢献する技術である。

本研究では，電源電圧の低下による回路マージンの減少，中性子ヒットなどによるエラー率を増加の問題については，投機実行用チェックポイントを用いるエラーリカバリによる信頼度の向上によって補完する方法を提案している。ハードウェアエラーを検出し，チェックポイント-リカバリにより信頼度を改善する方法は有効な方法であるが，従来の研究の方式では，性能に大きな負担があったり，追加の回路量が大きいという問題がある。性能の低下は，取りも直さず，性能あたりの電力の増加であり，また，回路の追加は電力を増加させる。低電力化を阻害しないためには追加の回路量が少なく，かつ，性能の低下も小さいエラーリカバリ方式が必要である。

本研究の第二の提案である投機実行用チェックポイント-リカバリ機構をハー

ドウェアエラーからのリカバリにも共用する方式は，エラー検出回路の追加は必要であるが，チェックポイントの作成とリカバリは投機実行を行うプロセッサでは既存の機構を流用しており，追加のハードウェアはほぼゼロであり，また，性能低下も無い方式である。第 5 章に述べる SPARC64 V プロセッサの例では信頼度を 3 倍以上に改善することが可能であり，本技術を併用することにより，低電源電圧化によるエラー率の増大の影響を補うだけでなく，当初の信頼度に比べて 2 倍程度の改善が可能となる。

SPARC64 V プロセッサの例では，エラー検出回路に使われるハードウェアは全体の 10%程度であるが，キャッシュメモリなどにはパリティーチェックや ECC の付加が一般的になっており，このようなマイクロプロセッサと比較するとエラー検出のための追加の回路を必要とするのはプロセッサコアの論理回路部分だけであり，チップ全体としてのトランジスタ数の増加は 3〜4%である。従って，エラー検出回路の追加を含めても，消費電力の増加は少なく，低電力化をほとんど阻害しない高信頼化手法である。

また，追加の回路量が少ないことは，小規模コア化を妨げないという点で望ましい信頼度改善技術である。

以上のように，本研究の第 2 章，第 3 章で提案する小規模マルチコア化と低電力回路技術と第 4 章で提案する投機実行用チェックポイント-リカバリ機構をハードウェアエラーからのリカバリにも共用する方式の併用はシナジー効果が高く， 両者の組み合わせにより，高性能，低電力化と高信頼化を同時に達成することを可能としている点が特徴である。

第 2 章 小規模マルチコア技術による低電力，高性能化

2.1. 小規模マルチコア化の検討

従来，プロセッサの開発は命令並列度の追求による IPC（Instruction Per Cycle）の向上と，クロック周波数の向上により，単一命令列の実行性能を高めることに努力してきた。しかし，プロセッサのトランジスタ数の増加とクロック周波数の向上により消費電力が急増しており，利用可能な命令並列度の飽和と相俟って，この方法でのプロセッサの性能向上は困難になってきている。

命令並列度の飽和を回避する手段として，豊富なトランジスタを活かし，一つの半導体チップ上に複数のプロセッサを集積するチップマルチプロセッサが 1996 年に Nayfeh 等により提案[29]され，以来，多くの研究発表[30][31][32]が行われている。また，チップマルチプロセッサは 2001 年に商用化されており，[33]として論文発表されている。

マルチプロセッサは，複数の処理が並行して実行され，複数命令列実行の利点が活きる用途でなければ性能向上が得られないが，最近ではインタネットのホームページや各種のオンライントランザクション処理などの，多数のユーザの処理を同時並列的に実行するため多数の命令列（スレッド）を並行的に実行する使い方が増加している。Alameldeen 等の論文[34]ではオンライントランザクション処理に加えて SPECjbb，Apatch，Slashcode などのアプリケーションは高いスレッド並列性を持つことが示されている。これらの処理では，個々のユーザの処理は大きな計算能力を必要としないが，同時ユーザ数が大きいため，サーバ全体としては大きな処理能力が要求される。

そこで，命令並列度の改善にトランジスタを注ぎ込むのではなく，複数の小規模なプロセッサを作りスレッド並列度を有効に利用することにより，従来の単一命令列を高速で実行することに主眼を置いたプロセッサよりも高いスルー

プットを実現し，また，同一スループットならば低電力で動作するプロセッサについて研究を行った。このような考え方は Barroso 等により発表[35]されている。しかし，この論文では 1 命令発行の初期のαプロセッサを複数集積するケースと新設計の 4 命令並列発行プロセッサ 1 個との性能比較を行っているが，電力やエネルギー効率については考察されていない。また，Cordrescu 等[36]は命令発行数や演算パイプラインの数を変えてチップマルチプロセッサの性能トレードオフを考察しているが，ハイレベルのモデリングであり，実設計に基づく詳細な比較にはなっていない。本研究では富士通のハイエンド unix サーバ向けに開発された SPARC64 V プロセッサをベースとし，これを簡単化し小規模なプロセッサコアを作る。そして，これを複数個集積することに加えて，低電力化手法と組み合わせ，よりエネルギー効率の高いチップマルチプロセッサを作るというアプローチ[62][63]をとった。

2.1.1. IPC 性能

図 2.1 に示すように SPECint ベンチマーク性能は年率 1.40 倍で向上している。一方, 図 2.2 に示すようにクロック周波数の向上は年率 1.25 倍である。従って，IPC(Instruction Per Cycle)の性能向上は年率 12%である。

IPC 性能の向上は，多数のトランジスタを使用してより高度な並列処理を実現し 1 サイクルあたりの命令処処理能力を向上させることと，コンパイラを改善し効率の良いコードを生成することにより実現されてきている。コンパイラの改善による性能向上は，ハードウェアに依存せず最適化により実行する命令数を削減するものと，増加したハードウェア資源を利用して実行効率を上げるものがある。レジスタ数の増加や大容量の命令キャッシュを利用してループアンローリングを行ったり，大容量のノンブロッキングキャッシュを利用してプリフェッチを行ったりするなどにより性能の高いコードを生成するのは後者の例であり，トランジスタ数の増加と不可分な性能改善である。

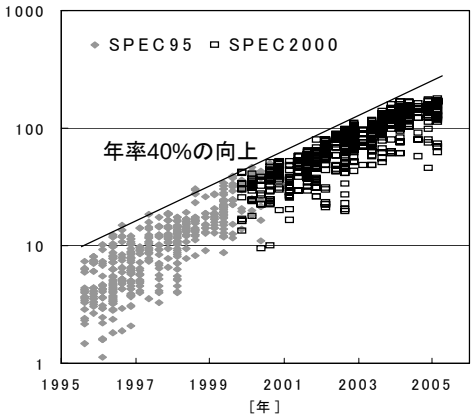


図 2.1 SPECint 性能トレンド

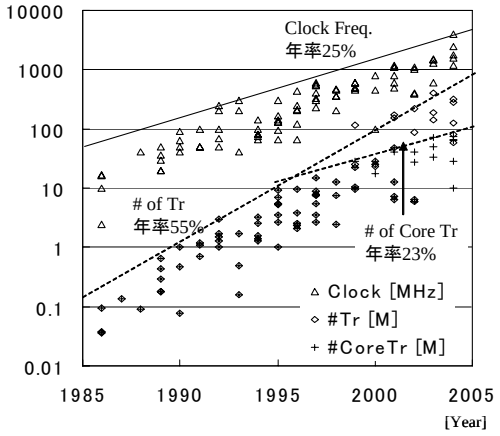


図 2.2 クロック周波数とトランジスタ数

一方，プロセッサチップの総トランジスタ数は年率 1.55 倍で増加しており，トランジスタ数の増加と性能向上の関係を付き合わせると，プロセッサの性能はトランジスタ数の 0.259 乗に比例している。但し，トランジスタ数の増加だけが性能の向上の原因ではないので，トランジスタ数を現在より増加させた場合にこの関係が維持されるかどうかは確実ではない。しかし，トランジスタ数を現在より削減する場合は，過去の状態に戻ることになり，この関係は成立すると考えられる。また，過去の時点からコンパイラの改善があることを含めると，トランジスタ数を削減した場合にはこの関係より高い性能が得られる可能性が大きい。

最近のプロセッサの総トランジスタ数の増加は主に大容量の 2 次/3 次キャッシュ容量の増大によっており，プロセッサコア（1 次キャッシュを含む）のトランジスタ数の増加は年率 1.23 倍に留まっている。こちらを用いると，プロセッサ性能はプロセッサコアトランジスタ数の 0.547 乗に比例している。

Intel 社の Pollack は，同一半導体テクノロジで作られた同社の連続する世代のプロセッサ間でトランジスタ数と性能を比較し，性能はトランジスタ数の 1/3～1/2 乗に比例する[41]と述べており，コンパイラの改善を含む性能改善としてトランジスタ数の 0.547 乗という結果は，Pollock の結果と矛盾しないと考えら

れる。

また，このようなプロセッサの複雑度の増加が電力あたりの性能を低下させている傾向は，Horowitz 等の論文[42]においても既に指摘されている。

歴史的な SPECmark の性能推移はハードウェアの改善だけでなくコンパイラの改善を含むものであるが，ここでは 3.1 式に示すように，一時点での SPECmark の IPC（Instruction Per Cycle）性能はトランジスタ数（＃Tr）の β 乗に比例すると見做す。前述のように β は 0.25～0.55 程度の値であるが，コンパイラの改善を差し引くと，これより小さい値になりうる。

$$IPC = Ki \times \left(\#Tr\right)^{\beta} \qquad \cdots(2.1)$$

2.1.2. 消費電力

トランジスタのドレイン飽和電流　Id は 2.2 式で表される。ここで K_{vt} はトランジスタのスレッショルド電圧 Vth の Vdd に対する比率であり，高性能マイクロプロセッサ用のトランジスタの場合は 0.1 程度の値である。理想的な MOS トランジスタでは α は 2 であるが，最近の短チャネルトランジスタでは α は 1.5 程度の値を取る。

$$Id = Id_0 \times \left[\left(1 - K_{vt}\right) \times Vdd\right]^{\alpha} \qquad \cdots\ (2.2)$$

相補型 CMOS 論理回路の遅延時間は，ドレイン電流と電源電圧の比に比例し，設計スタイル（パイプラン 1 段あたりの論理回路段数や Fan-out で決まる回路の負荷容量など）を一定とするとクロック周波数　f は回路の遅延時間の逆数となり 2.3 式のように表される。

$$f = K_{f0} \times \frac{Id}{c \times Vdd} = K_f \times Vdd^{(\alpha - 1)} \qquad \cdots\ (2.3)$$

ここで Id_0 はトランジスタの構造から決まる定数，c は回路の平均負荷容量，K_f は Id_0，K_{vt}，α，c 及びパイプライン 1 段あたりの論理回路段数などで決まる比例係数である。

2.3 式を整理して Vdd を f で表すと，2.4 式が得られる

$$Vdd = \left(\frac{f}{K_f} \right)^{\frac{1}{\alpha - 1}} \qquad \cdots (2.4)$$

プロセッサの消費電力　P は，全信号ノードの容量 C，信号ノードのスイッチ確率を s_f とすると，つぎのように表される。

$$P = s_f \times C \times Vdd^2 \times f = s_f \times C \times \left(\frac{f}{K_f} \right)^{\frac{2}{\alpha - 1}} \times f$$

$$= s_f \times C \times K_f^{\frac{-2}{\alpha - 1}} \times f^{\left(1 + \frac{2}{\alpha - 1}\right)} \qquad \cdots (2.5)$$

2.1.3. 小規模コアプロセッサの消費電力

一定のトランジスタ数で一つの大きなプロセッサコアを作る場合と，これを分割して複数の小規模なプロセッサコアを作る場合について消費電力と性能あたりの消費電力について考察を行う。

複数の小規模プロセッサコアを用いる場合のスループット性能は，スレッド並列度に依存する。図 2.3 は第 5 章で述べるプロセッサコア数と性能の関係を示すグラフであるが，この図に見られるように，スレッド並列度の高いアプリケーションの場合でも，2 次キャッシュの容量やメモリバスなどの共通資源の競合が発生するため，プロセッサ個数の増加により飽和傾向を示す。これらの効果を含めて IPC 性能を 2.6 式で近似する。

$$IPC(N) = IPC(1) \times \left(\frac{1}{N} \right)^{\beta} \times N^{\gamma} \qquad \cdots (2.6)$$

IPC(N)は一定のトランジスタ数で N 個のプロセッサコアを作る場合の総合

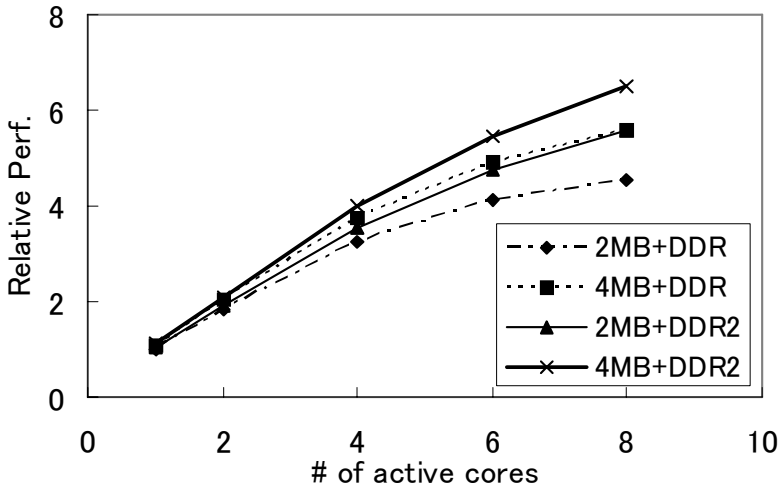


図 2.3 プロセッサコア数に対する TPC-C

IPC 性能であり，2.1 式から各プロセッサコアの性能は$(1/N)^{\beta}$となる。また，資源競合による性能飽和の度合いをN^{γ}で近似する。γ は理想的には 1.0 であるが，第 5 章で述べる小規模マルチコアプロセッサで TPC-C トランザクション処理を行う図 2.3 のケースでは 0.75～0.86 程度の値である。

小規模マルチコアプロセッサのスループット性能　T は IPC とクロック周波数 f に比例し 2.7 式で表される。

$$T = IPC(N) \times f$$

$$= IPC(1) \times \left(\frac{1}{N}\right)^{\beta} \times N^{\gamma} \times f \qquad \cdots (2.7)$$

2. 7 式を変形し，クロック周波数 f をスループット性能 T で表すと，2.8 式が得られる。

$$f = \frac{T}{IPC(1) \times \left(\frac{1}{N}\right)^{\beta} \times N^{\gamma}}$$

$$= \frac{T}{IPC(1)} \times N^{(\beta-\gamma)} \qquad \cdots (2.8)$$

2.5 式の f に 2.8 式を代入して整理すると，2.9 式が得られる。

$$P = s_f \times C \times K_f^{\frac{-2}{\alpha-1}} \times \left[\frac{T}{IPC(1)} \times N^{(\beta-\gamma)}\right]^{\left(1+\frac{2}{\alpha-1}\right)} \qquad \cdots (2.9)$$

ここでαは 1.5，βは 0.25～0.55，γは 0.8 とすると， N の指数は－1.25(β＝0.55)～－2.75(β＝0.25)となり，スイッチ確率 s_fが N に依存しないと仮定すると，N の増加に伴い一定のスループット T の達成に必要な電力は大幅に低下する。図 2.4 に N と消費電力，図 2.5 に消費電力を一定とした場合の N とスループット性能の関係を示す。この関係はトランジスタ数と IPC，小規模マルチコアの性能スケーラビリティーなどについて経験的関係を用い，また，一定の設計スタイルを想定するなど適用範囲の制約はあるが，γ=0.8 のスレッド並列度の高い処理に対しては，単一命令列の実行性能を最高にする設計よりも，同じ量のトランジスタで複数の小規模コアを作ることにより消費電力を低減できることを示している。また，図 2.5 は，複数の小規模コアを作ることにより，一定の消費電力であればより高いスループットを実現できることを示している。

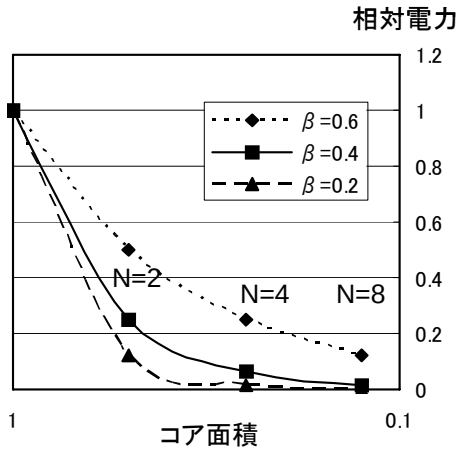


図 2.4 コア面積と相対電力
(スループット性能一定)

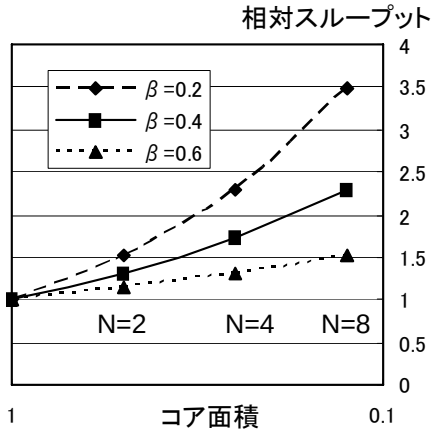


図 2.5 コア面積とスループット性能

図 2.4 では, β=0.4 で 4 コアの場合の消費電力は 1 コアの場合の 6.25%に減少している。しかし，2.2 式～2.5 式の関係に従うと α=1.5 の場合の電源電圧は消費電力の 0.4 乗に比例し, 消費電力を 6.25%とするためには電源電圧 Vdd を 0.33 倍に低減する必要があり，回路動作の点から現実的ではない。消費電力を 30%とする場合の電源電圧は 0.62 倍であり，この程度が現実的な下限である。従って, 4 コア以上の場合は, 電源電圧の低減により消費電力の低減をおこないつつ, スループットも向上させるという設計が現実的である。

2.2. まとめ

第 2 章では，プロセッサの歴史的なトランジスタ数や性能トレンドから，トランジスタ数と性能の関係を推定し，これを電源電圧と遅延時間の関係，マルチコアによる性能向上とをあわせてモデル化することにより，小規模コアを用いるマルチプロセッサの方が，従来の単一命令列の高速実行を最優先とする大型コアを用いるプロセッサより高い電力効率が得られることを示した。

- 性能がトランジスタ数の 0.4 乗に比例する場合（β=0.4)は，高 IPC を追求する大型コアのプロセッサと比較して, 1/4 面積の小規模コアを 4 個搭載するマルチプロセッサ化により，消費電力を一定とすると，大型コアを 1 個搭載するプロセッサに比べて 1. 7 倍程度のスループットが得られる。

- また, 同一スループットを実現するための消費電力は 1/16 に低減できる。但し，その時の電源電圧は消費電力の 0.4 乗に比例して低減する必要があり，0.33 倍の電源電圧とする必要がある。しかし，これは回路動作から見て過小であり，現実的な消費電力の下限は 30%～40%である。このため，消費電力の低減とあわせてスループットの向上も図るという設計が現実的である。

第 3 章 回路設計技術による低電力化

消費電力 $P = C*V^2*f$ から，電源電圧 V を下げることにより 2 乗で消費電力を減少させることが出来る。従って，電源電圧の低減は消費電力の低減にもっとも効果的な方法である。また，図 3.1 に示すようにトランジスタのリーク電流は電源電圧のおおよそ 1.5 乗に比例しており，リーク電流に起因する消費電力は電源電圧の約 2.5 乗に比例する。従って，電源電圧の低減は，動作電力だけでなくリーク電力の低減にも非常に有効である。

しかし，論理回路に用いられる相補型スタティック CMOS 回路は広い電圧範囲で動作するが，メモリやレジスタファイルなどのカスタム回路は正常動作する電源電圧範囲が狭いことが多い。このため，通常のマイクロプロセッサは正規の電源電圧の－10%程度までは動作が保証されているが，それ以下の電圧では正常な動作が保証されていないケースが一般的である。

従って，低電源電圧動作により低電力化を実現するためには，単にクロック周波数の低下を許容するだけでなく，低電圧で動作するカスタム回路について検討する必要がある。

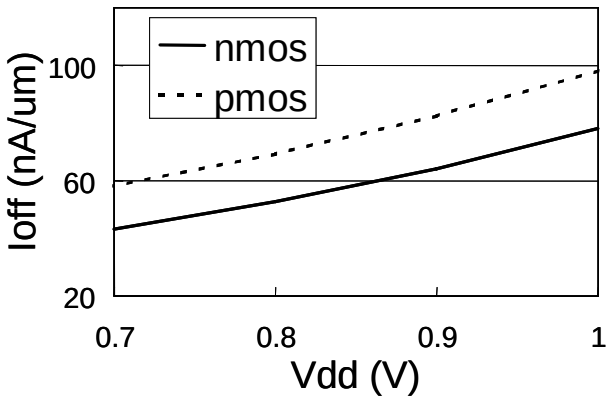


図 3.1 90nm トランジスタのリーク電流の電源電圧依存性

3.1. 低電力回路設計

回路の消費電力はスイッチングによる容量の充放電にともなうアクティブ電力が最大の要素であるが，半導体の微細化によりリーク電力が急増しており，130nm プロセスを用いた高性能プロセッサでは全体の約 30%の電力がリーク電力と報告[43]されており，また，別の 90nm プロセスを用いた高性能プロセッサでは約 40%がリーク電力と報告[24]されている。従って，プロセッサの低電力化には，アクティブ電力の低減と併せてリーク電力の削減が重要である。

以下の解析では富士通の高性能 90nm CMOS プロセス[44]の採用を想定し，このプロセスのトランジスタや配線のモデルを用い，回路設計において制御できるパラメタを最適化することによる低電力設計を研究した。

この方針に沿って，以下の項目の制御による低電力化を検討した。

(1) 最小寸法である 40nm より長いゲート長(Lg)を持つトランジスタの使用

(2) 通常の 1.0V より低い電源電圧の使用

この半導体プロセスでは 3 種類の Vth を持つトランジスタが使用できるが，低 Vth トランジスタはリーク電流が著しく大きいため原則として使用しないこととした。また，低リークの高 Vth トランジスタは電源電圧が低下した時の飽和電流の減少の程度が大きいため，チップ全体としてトランジスタ数が多くリーク電流を極力低減する必要のある SRAM セル以外には使用しないこととし，以下の回路設計では，中間の Vth を持つ標準トランジスタだけを使用することにした。但し， Vth-Lg のロールオフ特性から Lg が長いトランジスタは標準トランジスタより若干高い Vth を持つことになり，半導体プロセスを変更することなく複数 Vth のトランジスタを使う手法[45][46]のメリットを実現している。

3.1.1. ゲートライブラリの設計

トランジスタのゲート長の影響を見るにあたって，まず，最小チャネル長で

ある Lg=40nm，Vdd=1.0V で動作する標準的な論理ゲート，フリップフロップ(FF)，リピータ回路のライブラリを設計した。

インバータの N-チャネル Tr と P-チャネル Tr のチャネル幅の比率は遅延最小となるように決定し，Wp/Wn=1.5 とした。このトランジスタサイズでの遅延は Fan-out が 4.5 で最小値を取る。しかし，Fan-out=4～8 の間では遅延，エネルギーは 10%以下の違いであり，回路設計にあたってはこの範囲での使用を目安とすることにした。NAND や NOR などのインバータ以外のゲートの設計では Logical Effort の理論[47]に基づき Wp/Wn 比を決定し，インバータと同様に，若干の遅延増を許容して電力を減らす設計とした。

3.1.2. 長めのゲート長のトランジスタの使用

前に述べたように, 90nm プロセスではトランジスタのサブスレッショルドリーク電流の削減が重要課題である。図 3.2 に電源電圧=1.0V，温度=85℃の場合のゲート長をパラメタとしたトランジスタのリーク電流を示す。最小寸法である Lg=40nm では P-チャネル Tr，N-チャネル Tr ともにリーク電流は約 300nA/μm であるが，Lg=45nm とすることにより約 90nA/μm に減少する。トランジスタの Vth のロールオフ特性から，ゲート長を若干増加させることは間接的に Vth を上げる効果[48]があり， これによりリーク電流が減少している。

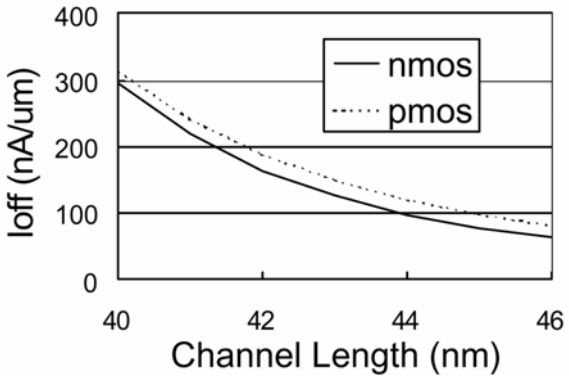


図 3.2 90nm トランジスタのリーク電流のゲート長依存性

通常，ゲート長を増加するとゲート容量が増えアクティブ電力が増加するが，この 90nm プロセスで作成したライブラリでは，単純ゲートの平均アクティブ電力は Lg を 45nm に増加させることにより 8%の減少が見られた。これはゲート長の増加に伴い Vth が増加し，充放電されるエネルギー $Cg \times (Vdd-Vth)^2$ が減少したことによると考えられる。従って，Lg=45nm のトランジスタを使用することにより，リーク電力だけでなくアクティブ電力も若干減少させることができる。

一方，Lg=45nm とすることによりトランジスタの飽和電流が低下するため，インバータ, NAND, NOR などの単純ゲートの平均の遅延時間は 22%遅くなり，リピータ込みの配線遅延時間は 11.3%遅くなる。

図 3.3 に示す 90nm プロセスで設計されたプロセッサのパス遅延の累積分布では，クロックサイクルから許容される最大遅延（図の 1.0 に対応）の 80%以上の遅延時間のパスは約 8%存在する。Lg=45nm のライブラリにより全てのパス遅延が一様に 25%増加すると，これらの 8%のパスは最大許容遅延を超えクロック周波数を低下させるので，この部分は Lg=40nm のトランジスタを使う必要がある。パス数の比率と Lg=40nm を使う必要があるゲート数の比率は等価ではないが，このパス数の比率から見て，大部分のゲートは Lg=45nm ライブラリを使用することができ，Lg=40nm のゲートを必要とする比率は小さいと考えられる。

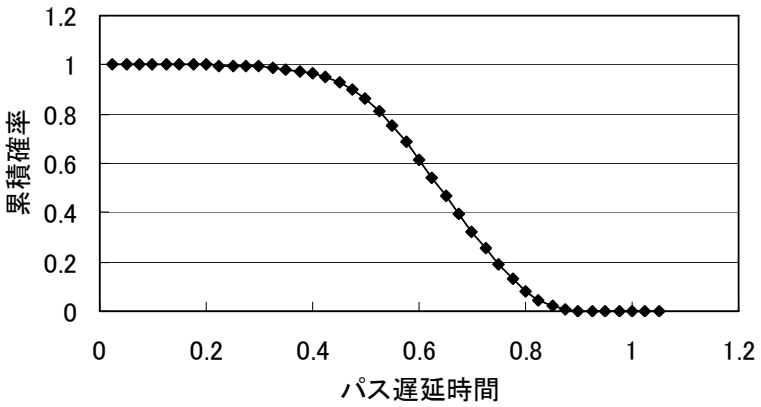


図 3.3 パスの遅延時間分布 （90nm プロセスで設計されたプロセサの例）
1.0＝クロック周波数に対応するサイクルタイム

3.1.3. 電源電圧の低減

アクティブ電力は電源電圧の 2 乗に比例するので，電源電圧を低下させることにより大きな電力低減が可能である。しかし，電源電圧を下げることにより遅延時間が増加し性能が低下するので，これらのバランスを取ることが必要である。図 3.4 に電源電圧に対する 1) リピータ込みの配線, 2)単純ゲートの平均, 3)インバータの遅延時間の変化を示す。

対象とする 90nm 半導体プロセスの通常電源電圧は 1.0V であるが，これを 0.8V に低下させると，単純ゲートの平均遅延は 26%増加し，インバータの遅延時間は 23%増加する。また，リピータ込みの配線の遅延は 15%増加する。リピータ込みの配線の配線遅延の増加が少ないのは，電源電圧によりリピータ回路の遅延時間は影響を受けるが，配線自体の遅延時間は変化しないからである。

以上のように，回路構成によって電源電圧を低下させた場合の遅延時間の変化率は異なるが，クロック周波数はインバータの遅延時間に逆比例すると見做した場合の相対的 MIPS/W と $MIPS^2/W$ を図 3.5 に示す。通常用いられるエネルギー遅延積の概念では $MIPS^2/W$ が性能指標であるが, チップマルチプロセッ

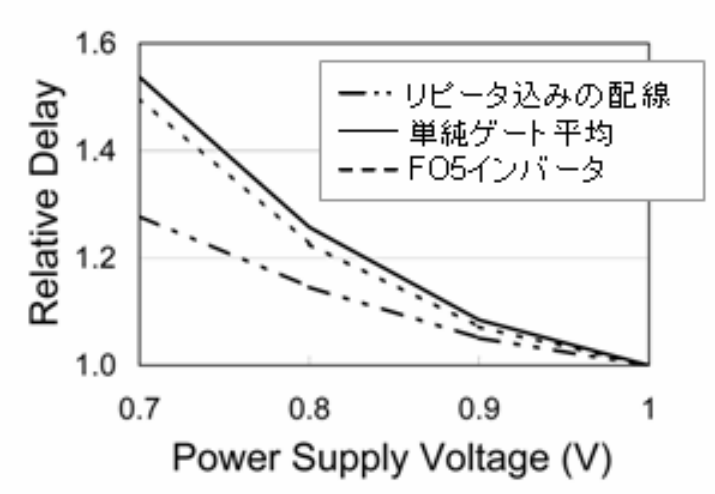


図 3.4 90nm プロセスのゲート遅延時間の電源電圧依存性

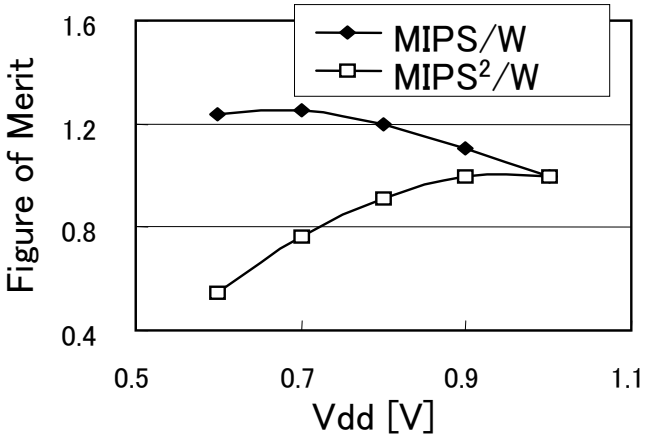


図 3.5 電源電圧に対する MIPS/W と MIPS²/W

サの場合は集積コア数を n 倍することにより,MIPS 性能も電力も n 倍となり，自動的に MIPS²/W が n 倍に改善されてしまう。従って，コア数に依存しない指標として MIPS/W も併せて考えることにした。図 3.5 に見られるように，MIPS²/W は 0.9V までは一定であるが 0.8V では 8.5%低下し，それ以下では急速に低下する。一方。MIPS/W は電源電圧の低下とともに増加し，0.7V で最大となる。これらの両者の指標を勘案して，低電力と性能をバランスさせる電源電圧として 0.8V を選択した。電源電圧を 0.8V とすることにより，1V と比較して MIPS/W は 1.2 倍に改善される。

一方，電源電圧を 1.0V から 0.8V に低下させることにより，ラッチのエラー率は論文[49]では約 1.5 倍に増加，[50]では 2 倍程度に増加し，[51]では SRAM のエラー率も約 1.5 倍に増加すると報告されており,電源電圧の低減を行う場合は，第 4 章で述べるチェックポイント-リカバリと組み合わせてシステムとして信頼度を確保することが必要である。

3.2. フルカスタムマクロの設計

SPARC64 V プロセッサでは，レジスタファイル，CAM などは，ラッチや FF から構成されたレジスタと一般ゲートを規則的に配置し，それらの間を規則的に配線する方法で作られているため占有面積が大きい。

このため，これらの中規模の配列をフルカスタム設計することによってチップ面積の縮小を図った。マイクロプロセッサではフルカスタム設計マクロの使用は一般的であるが，消費電力を低減するために電源電圧を低下させた場合に所望の動作速度を持つ回路設計が出来るかどうかが鍵である。

本研究では，FXRF レジスタファイル，高速 SRAM(1Kx80bit)，大容量 SRAM(4Kx72bit)， uTLB CAM の 4 種のマクロについて，90nm 半導体プロセスのトランジスタモデルを用いて 0.8V の電源電圧で動作するよう回路設計を行い，所望の動作速度が得られることを回路シミュレーションで確認し，このようなフルカスタムマクロが実現可能であることを示した。また，シミュレーション結果に基づき各マクロのアクティブ電力，および，リーク電力の計算式を作成し，プロセッサチップの消費電力計算に適用した。

表 3.1 に設計を行った各マクロのサイズ, 動作速度, 消費電力等の一覧を示す。

これらのマクロは小規模コア化を想定し，レジスタファイルの R/W ポート数や uTLB CAM のマッチポート数を削減するなど，機能が縮小されており，ベースラインプロセッサ用の回路と直接比較を行うことは出来ない。また，ベースラインプロセッサは 130nm テクノロジであり，90nm テクノロジへの移行だけで面積はほぼ半減する。このような違いを考慮する必要があるが，チップ面積でみると，FXRF レジスタファイルはベースラインでは 6.47mm^2 を占めていたが，R/W ポート数の減少，FXRF をキャッシュするレジスタファイルが不要になるなどのアーキテクチャ変更による回路量の削減を含めて， 90nm 化，フルカスタム化により 0.32mm^2 に減少した。また，uTLB CAM も FF と EOR 回路で構成していたため 1.78mm^2 を占めていたが，小規模コア化で CAM ポートが

表 3.1 マクロサイズと特性（Vdd=0.8V -5%，Tj=85℃）

マクロ	サイズ [μm x μm]	アクセス時間 [ps]	平均消費エネルギー [pJ]	リーク電流 [mA]	仕様
FXRF レジスタファイル	380x840	365	77.9	14.8	72bitx156 5R2W
高速 SRAM	670x314	380	23.1	5.1	80bitx1K
大容量 SRAM	684x910	565	47.4	13.0	72bitx4K
uTLB CAM	264x142	511	20.0	1.61	32Entry 64bit Match, 46bit 1R1W

2 から 1 に減少し回路量が減少したことと，フルカスタム設計，90nm 化を合わせて 0.04mm² に減少した。一方，RAM は元々カスタム設計であり，90nm 化の効果を除くと面積は殆ど同じである。

アクセス時間は 556ps(1.8GHz クロック)のサイクルタイムを想定し，FXRF と高速 SRAM はサイクルタイムの 60%程度，大容量 SRAM と uTLB CAM はフルサイクルで動作させる仕様であり，表 3.1 に見られるように，ほぼ，目標を達成しており，正式の開発においては更にチューニングを行う必要は残っているが，目標仕様の達成は可能との感触が得られた。

3.2.1. レジスタファイル

SPARC アーキテクチャでは，一時点でアクセスできる整数レジスタは 32 個であるが，レジスタウインドウをサポートしており，一般的な RISC プロセッサと比較すると必要なレジスタエントリ数が多く，ベースラインプロセッサの整数レジスタ（FXRF）は 156 エントリを必要とする。各レジスタのデータビットは 64bit であるが, データパスの誤り検出のためにバイト毎にパリティーチェ

ックを付加しており，物理的には 72bit となっている。

また，整数演算，アドレス計算，ストアの各ユニットへのデータ供給を行うため, 5 本の読み出しポートと演算とロードの結果の書込みのために 2 本の書込みポートを必要とする。通常の SRAM では差動のビットライン構造を用いるのが一般的であるが，R/W のポート数が多いため，ビット線の本数を削減し面積を縮小するため, 図 3.6 に示すようにシングルエンドのビット線構造のセルを採用した。この図では省略されているが，M1～M3 と wwl，wbl は書き込みポート数分設けられており，M4，M5 と rwl，lrbl は読み出しポート分設けられている。

低電圧動作のため各セルからの読み出し電流が減少し，通常のレジスタファイルのように 1 本のビット線に全セルを接続する方法では必要な速度が得られないので，図 3.7 に示すようにローカルビット線に接続するセル数を 10 個に制限し，このブロックの出力を上下に分割したグローバルビット線に接続するという 2 階層の回路構成として必要な読み出し速度を達成した。また，この構成により，読み出し動作に当たって，プレチャージされたビット線のうち放電されるのは，選択されたセルを含むローカルビット線とそれが接続されている上下どちらか一方のグローバルビット線だけとなるので，レジスタの高さ全体に

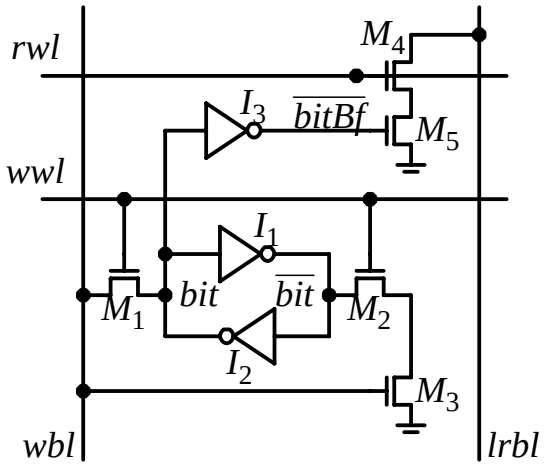


図 3.6 レジスタファイルのセル回路

渡る長いビット線を放電するアーキテクチャに比べてアクティブ電力を低減することが出来た。

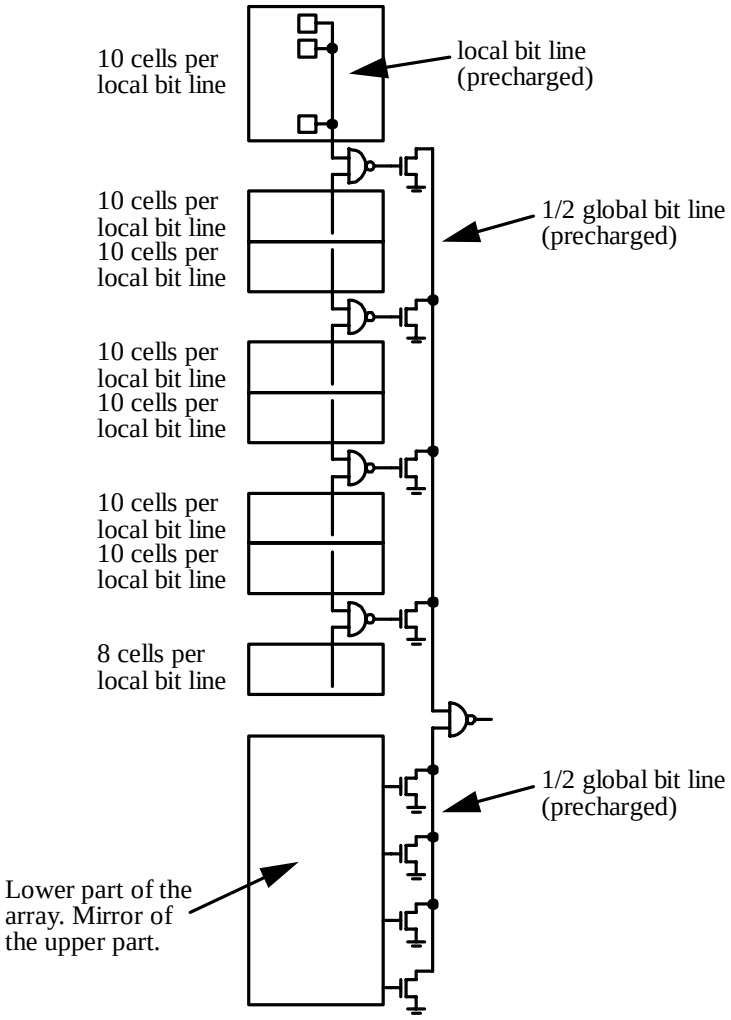


図 3.7 レジスタファイルのビット線構成

3.2.2. 低電圧動作 SRAM

まず，0.8V の低電圧で動作する L1$用の高速 SRAM について述べる。SRAM はセルのトランジスタサイズが小さく，かつ，リーク電流低減のために高 Vth トランジスタを用いているので，標準 Vth トランジスタを用いるレジスタファイルのセルと比べて低電圧で動作させた場合の読み出し電流の低下が大きい。このため，レジスタファイルと同様に，ローカル，グローバルの 2 階層のビットライン構成とし，ローカルビットラインに接続するセル数を高速 SRAM では 16 セルに制限し，必要な読み出し速度を得た。この SRAM のビット線構造を図 3.8 に示す。

セル構造は通常の 6Tr セルであり，書込みは差動で行うが，ローカルビットライン単位で必要となるセンスアンプの面積を縮小するため，読み出しはビッ

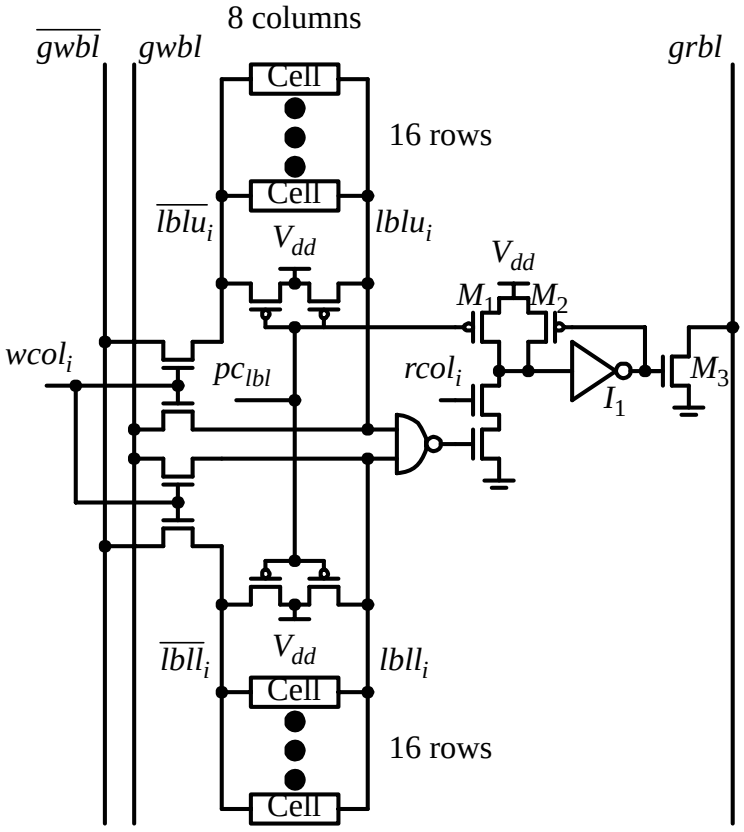


図 3.8 SRAM マクロのビット線アーキテクチャ

トラインをフルスイングさせ，片側の信号を NAND ゲートで読み出す方式を用いた。また，ビット線がフルスイングするため電源電圧マージンが大きく，低電源電圧動作にともなうノイズマージンの減少の問題を回避している。

また，L2$に用いる大容量 SRAM マクロは，72bitx4K と容量が大きく，密度を高めることが重要である。このため，速度と密度のバランスの点から，高密度 SRAM では 32 セルをローカルビットラインに接続し，グローバルビットラインにこれを 8 ユニット接続する構成とした。

L2$に用いられる大容量 SRAM マクロは全てのマクロが常に読み出されるのでは無く, アクセスされるアドレスに応じて一部の RAM マクロからの読み出しが行われるが, その他の RAM マクロは動作する必要がないという使い方が一般的である。このため，マクロの選択信号によりクロックをゲートする設計とした。選択信号によりクロックをゲートしない場合は, アイドル状態でも各 SRAM マクロは 1 サイクルあたり 10.6pJ の電力を消費しており，これをゼロにすることにより，1.8GHz 動作の場合において 32KB のマクロあたり 14mW の消費電力削減が可能となる。

3.2.3. マイクロ TLB CAM

マイクロ TLB はメイン TLB（Translation Lookaside Buffer）をキャッシュする機構であり，32 エントリの CAM 構造となっており，仮想アドレスを入力として CAM 検索を行い一致した仮想アドレスに対応する物理アドレスを読み出すことが出来る。このため，仮想アドレスとアドレス空間 ID の合計 64bit を比較する CAM と 40bit の物理アドレスを記憶する RAM が連結された構造となっている。

また，マイクロ TLB は，エラー検出のために CAM データと RAM データにはパリティービットが付加されており，読み出し時にパリティーチェックを行う。

CAM 部分は，他のフルカスタムマクロと同様に低電圧動作での電流の低下に対処するため，ローカルマッチラインに接続するセル数を 8 個に制限し，ローカルとグローバルの 2 段階のマッチライン構成を取ることにより，0.8V 動作で 511ps の物理アドレス出力を実現した。また，Lg=45nm トランジスタの多用により 1.61mA のリーク電流を達成した。

3.2.4. リーク電流の低減

各フルカスタムマクロの設計に当たっては可能な限り Lg=45nm のトランジスタを用い，リーク電流の削減を図った。また，Lg=45nm のトランジスタを用いることの効果を確認するために, 全てのトランジスタを Lg=40nm として各マクロのリーク電流を回路シミュレーションにより求めた。大部分を Lg=45nm トランジスタとした設計と, 全部のトランジスタを Lg=40nm とした場合のリーク電流の比較を表 3.2 に示す。

uTLB CAM は構造が複雑なため全部を 40nm トランジスタ設計とした場合の評価を省略したが, 構造から見て, アクセス時間とリーク電流の変化率は FXRF とほぼ同じと考えられる。

Lg=45nm トランジスタの多用により, FXRF レジスタファイルでは 23.6mA, 高速 SRAM では 3.5mA，大容量 SRAM では 11.3mA リーク電流が減少し，リーク電流は 40%～58%減少している。

一方，アクセス時間の増加は 30～45ps（7.4%～14.1%)であり，大幅なリーク電流の削減を考えると，Lg=45nm を多用する設計の方が優れていると考えられる。

表 3.2 45nm トランジスタ主体の低リーク設計と
全部 40nm トランジスタ設計の比較

	アクセス時間			リーク電流		
マクロ	低リーク設計 45nm Tr	全 40nm 設計	遅延増加	低リーク設計 45nm Tr	全 40nm 設計	電流減少
FXRF	365	320	14.1%	14.8	34.8	-57.7%
高速 SRAM	380	350	8.6%	5.1	8.6	-40.7%
大容量 SRAM	565	526	7.4%	13.0	24.3	-46.5%

3.3. まとめ

1.0～1.1V での使用に最適化された富士通の 90nm 世代の半導体プロセスを用いて，長めのゲート長(Lg=45nm)をもつトランジスタの多用と，0.8V の低電源電圧動作によりリーク電力の低減と MIPS/W を 1.2 倍に改善できることを示した。また，SRAM やレジスタファイルなどの回路を設計し，0.8V の低電源電圧で安定して動作し，要求される遅延時間を満足するカスタムマクロが実現できることを示し，低電源電圧動作が可能であることを示した。

- 最短 40nm のゲート長が可能な富士通の 90nm プロセスにおいて，ゲート長 45nm のトランジスタを用いることにより，リーク電流は 0.3 倍となることを示した。
- 45nm トランジスタの使用により遅延時間は 22%程度増加するが，これによりクロックサイクルを低下させるパスは全体の 8%程度である。これらのパスに含まれるゲートは 40nm トランジスタを用い，その他の大部分のゲートは 45nm トランジスタを用いることにより，リーク電力を削減できる可能性を示した。
- 電力と遅延時間の関係から電源電圧として 0.8V を用いることが

最適であることを示した。

- 2 種の SRAM マクロ，レジスタファイル，CAM の 4 種のマクロの回路設計を行い，0.8V の電源電圧において 1.8GHz クロックで動作する回路が実現可能であることを示した。
- また，これらのマクロにおいて 45nm トランジスタを多用することにより，全体を 40nm トランジスタで設計した場合と比較して，リーク電流を 60%以下に低減することが可能であることを示した。

第 4 章 投機実行用チェックポイントを用いたエラーリカバリ技術

4.1. 投機実行を行うプロセッサのチェックポイント-リカバリ

プロセッサの高速化の手法として，条件分岐命令の分岐の有無の履歴をもとに，条件コードが確定する前に分岐方向を予測して実行を行う投機実行がある。投機実行を行うプロセッサでは，予測外れの場合には元に戻って条件分岐命令からやり直す必要がある。最初の投機実行マシンである IBM 360 Model 91[64]においては，予測外れが検出されるとパイプラインをクリアし，投機的に実行した命令の影響を除去してから正しい分岐方向の命令列を実行する方法で回復を行っているが，最近のスーパスカラプロセッサでは複数の条件分岐命令を越えて投機的に命令を実行するために，各条件分岐命令の直前にチェックポイントを設けてリカバリを行う方法がとられる。

筆者らは，投機実行を行うプロセッサが既にチェックポイント-リカバリ機構を備えていることに着目し，これをエラー訂正にも共用する方式を米国特許として出願し，2003 年 2 月に特許 6,519,730[26]を取得した。

4.1.1. HAL R1 プロセサの投機実行チェックポイント-リカバリ機構

A. HAL R1 プロセサのリネーム機構

筆者らは，1994 年に投機実行を行うマイクロプロセッサ HAL R1[27][28]を開発した。このプロセッサでは，整数レジスタファイル，浮動小数点レジスタファイルなどへ結果を格納する命令に対して，書込みを行うレジスタ番号のリネーム(Rename)を行っている。

リネームは，レジスタの依存関係を維持しながら並列実行性能を改善するために導入された機構であり，命令の並列実行性を高めるようにプログラム上に記述された論理レジスタ番号(Rn)をハードウェアが持つ物理レジスタ番号(Pn)に対応付ける。

ADD R1,R2 → R2

ADD R2,R3 → R1

このような命令列の場合，2 番目の ADD 命令は 1 番目の ADD 命令の結果である論理レジスタ R2 を使用し，1 番目の ADD 命令のオペランドの R2 とは異なる値である。従って，この状態では二つの ADD 命令を同時に発行することは出来ない。このため，次に述べるように演算結果を格納する論理レジスタに対して異なる物理レジスタとの対応付けを行い，命令間の依存関係を解決している。

ADD P10,P15 → P16　；オペランド R1→P10，R2→P15；結果 R2→P16

ADD P16,P12 → P17　；オペランド R2→P16，R3→P12；結果 R1→P17

この例では，1 番目の ADD のオペランドである R2 は物理レジスタ P15 に割り当てられており，結果を格納する R2 は P16 に割り当てられる。そして，2 番目の ADD のオペランドの R2 は 1 番目の ADD の結果の R2 を必要とするので P16 から読まれる。また，2 番目の命令は R1 に結果を書き込むが，このレジスタも P17 にリネームされているので，1 番目の ADD のオペランドを格納している P10 には影響を与えない。

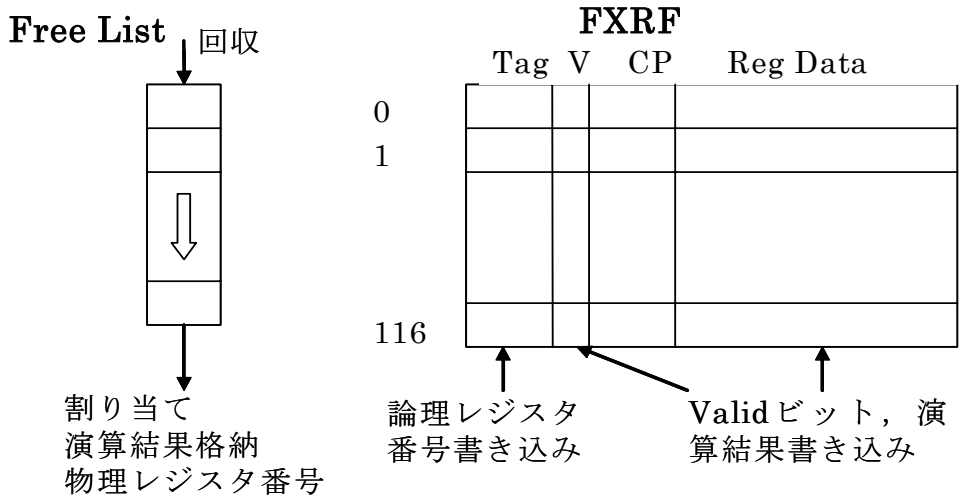


図 4.1 HAL R1 プロセッサの整数部物理レジスタファイル（FXRF）管理

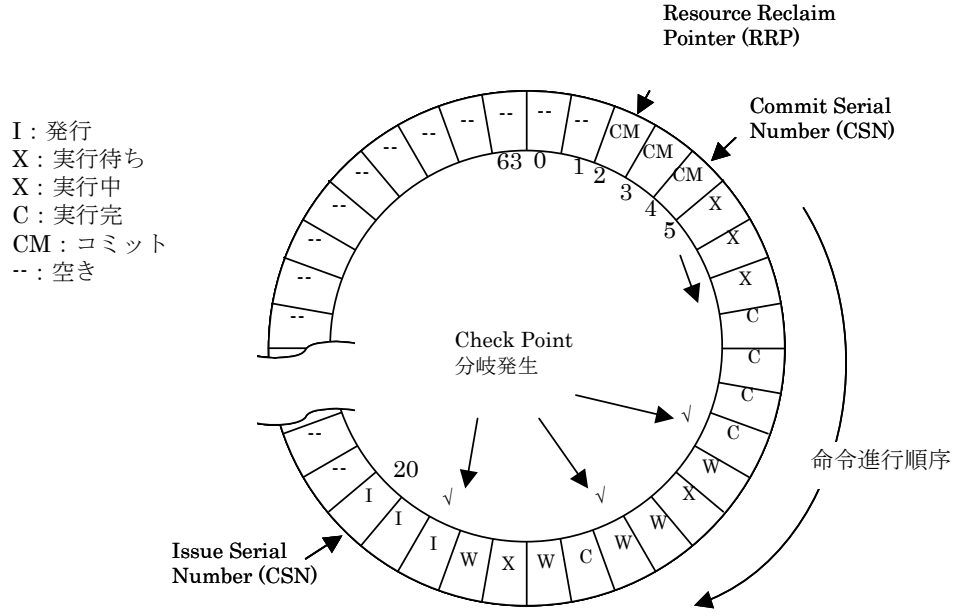


図 4.2 HAL R1 プロセッサのコミットリング

以上のようにリネームは並列実行を助ける機構であるが, R1 に対して P10 を保持すると同時に，より新しい状態 P17 を保持するというように投機実行中のアクティブ状態を保持する機構としても使用され，リカバリを行うために必須

の機構である。

HAL R1 プロセッサの整数部のレジスタ管理機構を図 4.1 に示す。物理レジスタファイル(FXRF)は論理レジスタの総数より多いエントリを持ち，データフィールドに加えて，バリッドビットや対応する論理レジスタ番号を示すタグフィールドを持つ。バリッドビットは、同一の論理レジスタ番号に対して複数の物理レジスタが存在する場合にどの物理レジスタエントリが最新であるかを示す。また, フリーリストと呼ぶ FIFO 構造で空き物理レジスタ番号を管理している。

論理レジスタに結果を格納する命令に対しては，命令のデコード時にフリーリストから空き物理レジスタ番号を取り出してリネームを行い，物理レジスタファイルの対応するタグエントリに論理レジスタ番号を格納する。また，その論理レジスタ番号を保持していたエントリのバリットビットをリセットし，フリーリストから取り出した新物理レジスタエントリのバリッドビットをセットする。

B. HAL　R1 プロセサのコミット機構

HAL R1 プロセッサは，演算が終了すると結果をリネームされたレジスタに書き込む。しかし，命令の実行は Out-of-Order であり，プログラム順で前にある命令が完了しておらず，分岐予測の外れやトラップが起こっている可能性もある。このため，プログラム順で前にあるすべての命令が正常に終了したことを確認してから，その命令を完了するコミット処理を行っている。

HAL R1 プロセッサは, 図 4.2 に示すコミットリングと呼ぶリング状の状態管理機構を持つ。コミットリングは，命令が発行されるごとに Issue Serial Number(ISN)を進めて 1 命令に対して一つのエントリを割り当て，状態ビットをセットする。条件分岐命令の場合は，予測外れから回復するためのチェックポイントを作成し，チェックポイント RAM と呼ぶ構造に分岐命令の Serial Number や PC 等を書き込む。レジスタファイルの各エントリのバリッドビッ

トは 16 段のスタックになっており，チェックポイントを作成する場合には，その時点でのバリッドビットをスタックにプッシュして格納することによりチェックポイントを記憶する。

そして，実行ユニットから命令処理の終了通知を受けると，当該命令の状態を実行終了（C）状態とする。

命令の実行が終了し，それ以前の命令が全て完了している場合は，その命令をコミット（CM）し，Commit Serial Number (CSN)を進める。

C. 予測外れと割り込みからのリカバリ

条件分岐命令のコミットを行う時点で，分岐予測外れの発生が検出された場合はその条件分岐命令はコミットせず，バリッドビットのスタックをポップすることにより条件分岐命令の直前のチェックポイントされたレジスタ状態を回復する。チェックポイント以前の全ての命令の実行は完了し結果が格納されており，バリッドビットの立ったエントリの集合はその時点で整合の取れたプロセッサのチェックポイントとなっている。そして，チェックポイント RAM から条件分岐命令のプログラムカウンタ値を取り出し，この命令から再実行を開始する。条件分岐命令が再実行される時点では条件コードレジスタをセットする命令は終了しており，再実行時には予測外れは発生せず，正しい分岐方向の命令が実行される。

一方，割り込みの場合は，投機実行用の一番古いチェックポイントまで戻り，そこからコミットリングを戻る方向に，最後にコミットした命令まで，1 命令づつリネームを元に戻すバックステップと呼ぶ動作を行い，割り込み発生時の正しい状態を回復する。

チェックポイントへの巻き戻しにあたって，チェックポイント以降の既発行の命令の取り消しが必要であり，これらの命令が使用する物理レジスタを開放しフリーリストに戻したり，命令を格納しているリザベーションステーション

のエントリを開放したりする操作が必要であるが，詳細は省略する。

なお，HAL R1 プロセッサでこのようなリネーム機構を設けているのは書込み頻度の高い整数レジスタファイル（FXRF）と浮動小数点レジスタファイル（FPRF)，それらの条件コードレジスタファイルだけである。その他のアーキテクチャレジスタについては、該当レジスタの更新命令発行時に以前の値を記憶することにより，直前の状態への復元を可能にしている。また，更新頻度が少ないアーキテクチャレジスタについては，それ以前の全ての命令がコミットしてから書き込みを実行することにより，取り消しの必要を無くしている。

4.1.2. 投機実行チェックポイント-リカバリ機構のエラー回復への共用化

前記の HAL R1 プロセッサの投機実行用チェックポイント-リカバリ機構を利用し，ハードウェアのエラーに対するリカバリを実現する方法について述べる。

A. エラー検出機構の追加

エラー検出の方法としては，データレジスタやバスなどのデータ伝送路に関しては 1.2.3 で述べたパリティーチェックなどを付加することにより，1bit エラーを検出することが出来る。また，ALU などのデータを加工するユニットに対しては回路の二重化比較や，予測パリティーと結果のパリティーの比較などの方法を用いることによりエラーを検出することが可能である。

エラー検出に必要なハードウェア量は，パリティーチェックをつける単位に依存するが，8bit のデータに 1bit のパリティーを付加する場合はハードウェア量は 9/8 となり 12.5%のオーバヘッドとなる。一方，付加単位を 16bit とすると 6%強のオーバヘッドである。ALU や乗算器へのエラー検出回路の付加は 50%～100%のオーバヘッドとなるが，プロセッサ全体から見ると小さな回路であり，検出機構の追加ハードウェア量は殆どパリティーチェックの付加単位で決まり，第 5 章で述べる SPARC64 V プロセッサの場合は約 10%である。

これらの手法によりプロセッサ各部のエラーを検出し，エラーが検出されたという事象だけを OR ツリーでまとめてコミット機構へ伝達する。

エラーが発生した場合には，割り込みと同様に，エラー発生前のチェックポイントを回復し再実行を行うことにより，単発エラーからの回復が可能となる。この方法では既存の投機実行用のチェックポイント-リカバリ機構をハードウェアエラーからのリカバリに共用しており，1.2.5 で述べた従来の方法と比較して，実装に必要なハードウェア量を削減している。

B. エラー検出の有無のコミット条件への追加

4.1.1 に述べたように投機実行を行うプロセッサでは，命令の実行結果を確定するコミットという動作があり，分岐予測外れや割り込みが発生しておらず，その命令以前の命令が全てコミットしているかをチェックし当該命令のコミットを実行する。

本研究で提案する方法は，これらの条件に加えて，その命令の実行に関してエラーが発生していないことをコミット条件に追加するものである。これにより，コミットされた命令の実行段階では検出可能なエラーは発生していないことが保証される。

命令の実行が終わった時点から，Commit Serial Number ポインタがその命令に到達するのには最低 2 サイクルを必要とすることを利用し，エラー検出からコミット機構までのエラー報告の遅延を 2 サイクル以下にすることにより，実行中の命令のコミットよりもエラー報告が先に届くことを保証することが可能である。また，命令実行の終了からコミットまでのサイクル数を大きく設計することにより，エラー報告の遅延が大きいプロセッサにも対応可能である。

この方法では，エラーを発生した命令を特定することは出来ず，一般的には，それより前の正常に実行された命令まで取り消すことになるが，エラー発生の頻度は低いので，これによる性能オーバヘッドは問題とならない。

4.2. まとめ

本研究のプロセッサでは，エラーが検出された場合にはエラー発生前の投機実行用のチェックポイントに戻って実行をやり直すことにより単発エラーからの回復が可能となり，信頼度を向上することができる。エラー検出を行うハードウェアは追加の必要があるが，このチェックポイントは投機実行のリカバリを行うために既に存在する機構であり，チェックポイントの生成，格納に関してエラー訂正のために追加するハードウェアは不要であり，実行時間のオーバヘッドも生じない。

- チェックポイントとリカバリ機構は投機実行用のものを利用するので，エラー検出を除くと，リカバリのために必要な追加ハードウェアはほぼゼロである。
- 投機実行用のチェックポイント機構は，従来の研究で述べたキャッシュベースのチェックポイント-リカバリと比較してチェックポイント作成のオーバヘッドが小さく，実行性能を低下させることなくチェックポイント-リカバリを実現できる。
- エラー検出はパリティーチェックなどを用い，プロセッサ全体の 10% 程度のトランジスタを必要とする。キャッシュメモリ等の大規模アレイはパリティーチェックや ECC を持つのが一般化しており，本論文のエラーリカバリに必要となるのはプロセッサコアの論理回路部分の 10%程度のハードウェアの追加である。

このため，投機実行を行うプロセッサに対して少ない実装負担で適用することでき，信頼度を向上することが可能である。

第 5 章 SPARC64 V プロセッサの低電力化と高信頼化

5.1. SPARC64 V プロセッサの構造

5.1.1. 命令フェッチマシン

図 5.1 に SPARC64 V プロセッサのブロックダイアグラムを示す。また，図 5.1 の各ブロックにキャッシュ容量やバッファのエントリ数を記載し，図の上部に各処理とそのパイプラインの長さを記載する。

レベル 1 命令キャッシュ(L1I$)は 2way のセットアソシアティブ構成で, 総容量は 128KB である。この L1I$から 8 命令(32byte)を並列に読み出し命令バッファ(I-Buf)に格納する。命令フェッチマシンは，通常の命令の場合はアドレス加算器，分岐命令の場合は分岐予測テーブル（Branch Target Address Cache)，リターン命令の場合はリターンアドレススタックなどを使って次にフェッチす

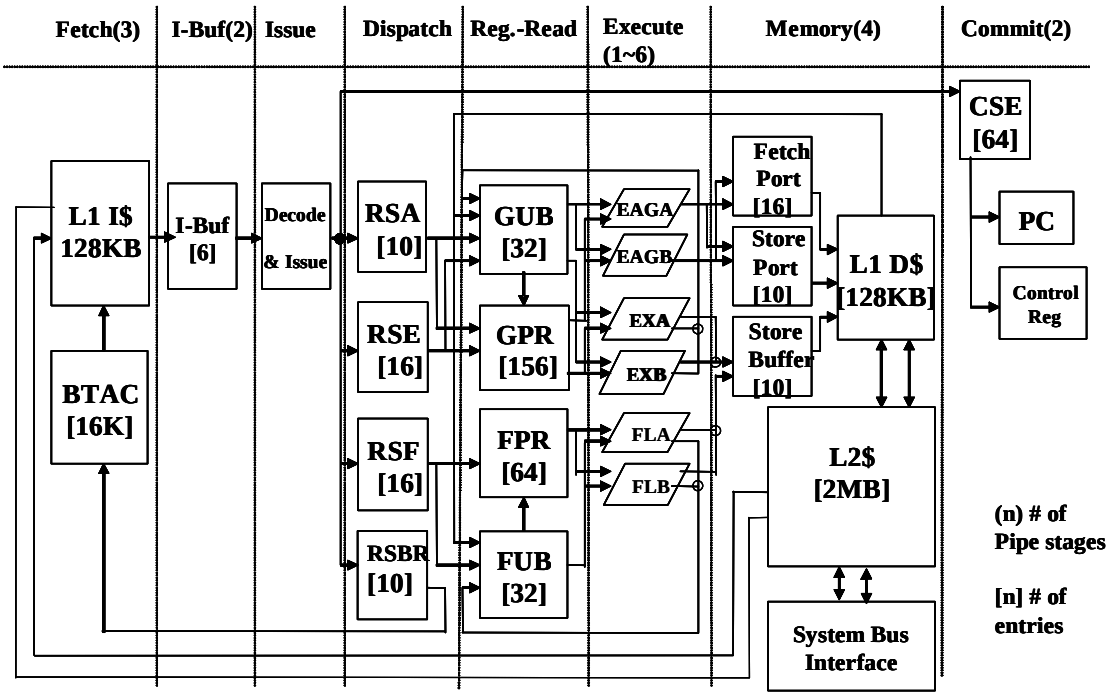


図 5.1 SPARC64 V プロセッサのブロックダイアグラム

る命令アドレスを予測し，8 命令 x6 エントリの I-Buf に空きがある限りフェッチを続行する。そして，I-Buf がフルになると L1I$の読み出しを停止し，I-Buf の空くのを待つ。

BTAC は 4way のセットアソシアティブ構成であり，過去に条件の成立した条件分岐命令の分岐先アドレスとその命令アドレスのタグ，予測の成功履歴などの情報を格納しており，BTAC をヒットした命令に対して分岐先の命令アドレスを出力する。一方，BTAC をミスした命令は，通常命令あるいは条件の成立しない分岐命令と見做し，連続するアドレスから命令フェッチを行う。

BTAC の内容は，条件分岐命令の実行が完了する時点で，分岐の有無や BTAC の予測の成功，不成功に応じて更新される。

5.1.2. 命令発行マシン

命令発行マシンは I-Buf から命令を取り出し，命令デコードを行い，命令の発行条件が整っていれば，命令の種別に対応するリザベーションステーションに格納する。命令を発行する順序はプログラムに記述された順序と同一(In-Order)である。

SPARC64 V プロセッサは整数演算命令を格納するリザベーションステーション RSE，浮動小数点演算命令を格納するリザベーションステーション RSF，アドレス計算を行うためロード/ストア命令を格納するリザベーションステーション RSA，分岐命令を格納するリザベーションステーション RSBR を備えている。図 5.1 に示すように RSE と RSF は 16 エントリ，RSA と RSBR は 10 エントリである。

命令の発行に先立ち，Out-of-Order 実行を行うため，演算結果を格納する論理レジスタはリネームし，整数命令の場合は GUB のエントリを割り当てる。また，浮動小数点演算の場合は FUB のエントリを割り当てる。これらのリオーダバッファのエントリ数はそれぞれ 32 エントリである。また，演算の入力となる

オペランドレジスタがリネームされた状態である場合は，アーキテクチャレジスタである GPR，あるいは FPR ではなくリネームされた GUB，FUB からオペランドを読み出すように命令をデコードしてリザベーションステーションに格納する。

命令発行マシンは，I-Buf に命令がない場合や GUB/FUB に空きが無くリネームが出来ない場合，デコードした命令を格納するリザベーションステーションに空きが無い場合は，命令のデコードや発行を中断し，I-Buf に命令が到着したり，必要な資源が開放されたりするのを待つ。

5.1.3. 命令実行パイプライン

SPARC64 V プロセッサは，整数演算を行う EXA，EXB の 2 本の実行パイプライン，浮動小数点演算を行う FLA，FLB の 2 本の実行パイプライン，アドレス計算を行う EAGA，EAGB の 2 本の実行パイプラインを備えている。

各命令実行パイプラインはリザベーションステーションにある命令の中から，オペランドが揃い，演算器が使用可能になるなどの条件を満たし，実行可能となった命令を発行の古い順に選択して実行を行う。入力オペランドが揃わない命令の実行は後回しにし，実行条件の整った命令を優先するため，命令の実行順序は順不同（Out-of-Order）になる。

実行可能となった命令は，整数演算とアドレス計算の場合は GPR/GUB，浮動小数点演算の場合は FPR/FUB から入力オペランドを読み出し，それぞれの演算器にデータを供給して演算を行う。

整数や浮動小数点演算の結果はそれぞれリネームされた GUB，FUB に格納される。また，EAGA，EAGB で計算されたロード/ストア命令のアドレスは，それぞれフェッチ(Fetch)ポート，ストア(Store)ポートと呼ぶバッファに格納され，ロード/ストア命令のアドレスとして使用される。また，ストア命令の場合は，演算器から 10 エントリのストアバッファと呼ぶ構造に格納すべきデータを

入れ，ストアポートのアドレスとともにストア命令の実行に使用する。ロード/ストア命令は，キャッシュをバイパスする I/O へのアクセス命令を除いて，L1D$をアクセスしてロード/ストア動作を行う。L1D$は 2way のセットアソシアティブ構造であり，各 Way 64KB で総容量は 128KB である。

L1I$，あるいは L1D$へのアクセスがミスすると，L2$がアクセスされる。L2$は容量 2MB で 4way セットアソシアティブである。そして L2$もミスするとシステムバスインタフェースを経由してメインメモリをアクセスする。

命令の実行状態を管理する CSE（Commit Stack Entry）は，リング構造であり最大 64 個の命令の実行状態を管理する。各命令は発行されると CSE にエントリが作られ，命令の実行が終了すると CSE は GUB，あるいは FUB の内容を GPR，あるいは FPR に書き戻すコミット動作を指令し，この書き戻しによりコミットした命令の実行結果がアーキテクチャレジスタである GPR，FPR に反映される。命令実行に用いるリソースの開放は，GUB，FUB はコミット時，リザベーションステーションのエントリは実行ステージの最初のサイクルに開放を行っている。

5.2. SPARC64 V プロセッサコアの小規模化

小規模プロセッサコアによるスループット性能向上，消費エネルギーの低減の検討にあたり，過去の設計から離れて理想的な小規模プロセッサコアを新たに検討するアプローチは，最適な小規模コアが得られる可能性があるが，探索範囲が広いため詳細な設計のレベルまで検討を深めることは困難であり，結果の信頼度が低くなるという問題がある。このため，既設計の SPARC64 V プロセッサをベースラインとし，これからハードウェアを削減する方法で小規模コアを導き出すアプローチを取った。この方法では探索範囲に制約があるのでグローバルな最適解とはならない恐れがあるが，既設計のプロセッサの論理設計や実装設計をベースにするため， 導出した小規模コアのトランジスタ数，面積，性能，消費電力等の精度が高いという利点がある。

図 2.4 に示したように小規模化を進めてコア数を多くする方がスループット性能は高くなるが，SPARC64 V プロセッサの誤りの検出，チェックポイント-リカバリによる訂正機構を含めて SPARC64 V プロセッサの基本構造を残すという方針から，現実的な回路量としてベースラインプロセッサの 1/2～1/3 程度のプロセッサコアを目標とした。ベースラインプロセッサは 130nm 半導体プロセスで製造されており，小規模化によりコアの面積が 1/2～1/3 になることと併せて，素子密度が 2 倍になる次世代の 90nm プロセスでは 4～6 コア，更に次の 65nm プロセスでは 8～12 コアを 1 チップに集積することが可能であり，この程度のコア面積でも，近い将来には十分なコア数のチップマルチプロセッサを構成することが出来る。

5.2.1. 性能評価の方法

リソース削減のトレードオフのためには，削減に対する性能の低下を把握する必要がある。このために SPARC64 V プロセッサの性能評価のために開発したアーキテクチャシミュレータ[53]をマルチコア用に改造して使用した。

また,プロセッサコアの性能を評価するベンチマークとしてはSPECint2000, SPECfp2000 を使用した。これらの SPEC ベンチマークはプロセッサコアの性能評価としては代表的なベンチマークである。アーキテクチャシミュレータによる性能評価にあたり， SPARC64 プロセッサを使った実機で SPECmark を実行し，Shadow ツール[54]を用いて実行トレースを採取した。また，トレース全体をシミュレートするのは時間が掛かるので， IPC，キャッシュミス率などがトレース全体と類似するよう各トレースから各 1M 命令の 20 ヶ所のサンプルを取り出してシミュレーションによる評価に用いた。

一方，マルチプロセッサの性能評価としては TPC-C ベンチマーク[55]を用いた。TPC-C は大規模な問屋のオーダ受付から出荷，請求書の発行などのオペレーションを模擬化したベンチマークであり，多くの発注や問い合わせを処理する代表的なトランザクションベンチマークである。また，インタネットで不特定多数のユーザがアクセスする形のオペレーションとも近い性格を持っている。TPC-C ベンチマークのトレースは 16CPU の SPARC64 サーバで TPC-C を実行し，各プロセッサの命令実行を自社開発のカーネルトレーサで採取した。このトレーサはアプリケーションの実行だけでなく，OS カーネルの命令実行も含めて命令トレースを採取することが出来る機能をもっている。SPECmark ではOS カーネルの実行時間は 2～3%であるが，TPC-C では全体の 20～30%の命令が OS カーネルで実行される[56][57]ので，OS 実行時を含めて命令トレースを採取することが重要である。

OS やデータベースでは複数プロセッサ間の同期や排他制御が行われている。トレースベースのアーキテクチャシミュレータではこれらの相互作用を（実際にロックを獲得するまでループを回るというように）適応的に実行することは出来ず，トレースを採取した環境で発現した命令列をそのまま再現して実行する。トレース採取環境は 16CPU であり本研究で評価しているプロセッサコア数より多く，同期待ち時間も長いと考えられるので，これらの相互作用は大きめに評価されていると考えられる。しかし，Ranganathan 等の評価[58]では，これらの相互作用の影響は全体の 5%以下であり，全体としては大きな影響を与え

ていないと考えられる。

5.2.2. ベースラインプロセッサからのリソース削減

ベースラインプロセッサの 1/2～1/3 の面積の小規模プロセッサコアを作るにあたり，以下に述べるように 3 段階の資源削減を行った。また，これらの資源削減に続いて，第 3 章で述べたカスタムマクロを適用し，面積削減をおこなった。各段階において削減したリソースのまとめを表 5.1 に示す。また，各段階の削減後のプロセッサコア面積を表 5.2，性能のまとめを図 5.6 に示す。

A. 第一段階の削減（命令発行幅と実行ユニットの削減）

130nm プロセスで作られた SPARC64 V プロセッサのプロセッサコア（一次キャッシュを含む）の面積は 104.6mm^2 であるが，デカップリングキャパシタや EC セルなどの能動領域以外の領域が含まれている。この領域を除くとプロセッサコアのサイズは 93.2mm^2 である。以下のリソース削減ではこの面積をベースラインプロセッサコアの面積とする。

ベースラインプロセッサの 1/2～1/3 の面積を目標としているので，第一段階では命令発行幅を 4 命令から 2 命令に半減する。命令発行を制限することにより，プロセッサコア全体の動きやキャッシュアクセスなどの動作が減少し，第二段階以降の削減のための余剰リソースを見つけやすくすることを狙っている。また，並列処理命令数を半減しているので，整数演算ユニット，浮動小数点演算ユニットなど二組ある演算パイプラインの数を半減した。

この資源削減により，表 5.1 に示すように面積は 86.06mm^2 に減少する。一方，図 5.6 に示すように 1 プロセッサコアの SPECint 性能は 89.1%，SPECfp 性能は 87.5%，TPC-C 性能は 98.1%に低下する。TPC-C はメモリアクセスに伴うキャッシュミスが多くメモリ待ちとなる比率が大きいので，SPECint/SPECfp に比べてプロセッサコアの性能低下の影響が全体性能に与え

る影響が小さくなっている。

B. 第二段階の削減（一次キャッシュの削減）

ベースラインプロセッサはそれぞれ128KBの大きなL1I$とL1D$をもっており，これらのキャッシュが大きな面積を占めている。第二段階では，このキャッシュ量の最適化を行った。

ベースラインプロセッサのキャッシュ量を削減して性能シミュレーションを行った結果を図 5.2 に示す。ここでは命令キャッシュとデータキャッシュの量は同じになるように削減を行った。図 5.2 に見られるようにキャッシュ量の削減による性能低下は緩やかである。しかし，図 5.3(a)に示すように面積削減はキャッシュ容量に比例するのに対して，SPECint2000 性能は容量半減ごとに低下量が増加する傾向にある。図 5.3(b)に示した性能の低下比率と面積の減少比率を見ると，16KB への減少までは性能低下よりも面積削減の比率が大きいが，16KB から 8KB に削減すると面積の削減率よりも性能の低下率が大きい。16KB と 8KB の場合の性能を比較すると SPECfp で 6.2%, SPECint で 4.3%, TPC-C で 3.0%の低下であるが，コア面積は 1.5%程度の縮小でしかない。このため，性能と面

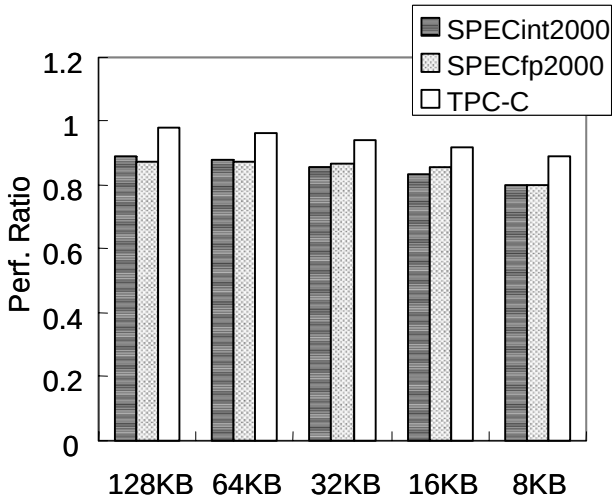
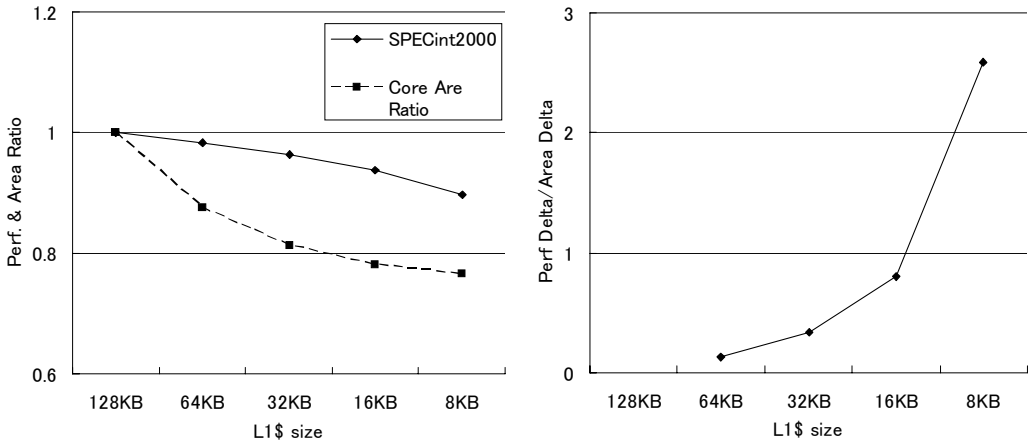


図 5.2 L1$サイズと相対性能
(ベースラインコアの性能＝1.0)



(a) SPECint2000 性能とコア面積　　(b) 性能減/面積減 比率

図 5.3　L1$サイズの面積削減と性能影響

積のトレードオフから小規模コアの一次キャッシュサイズを 16KB とすることにした。

この第二段階の削減によりプロセッサコア面積は 67.2mm² となった。一方，ベースラインと比較すると，SPECint 性能は 83.6%，SPECfp 性能は 85.5%，TPC-C 性能は 92.0%となった。

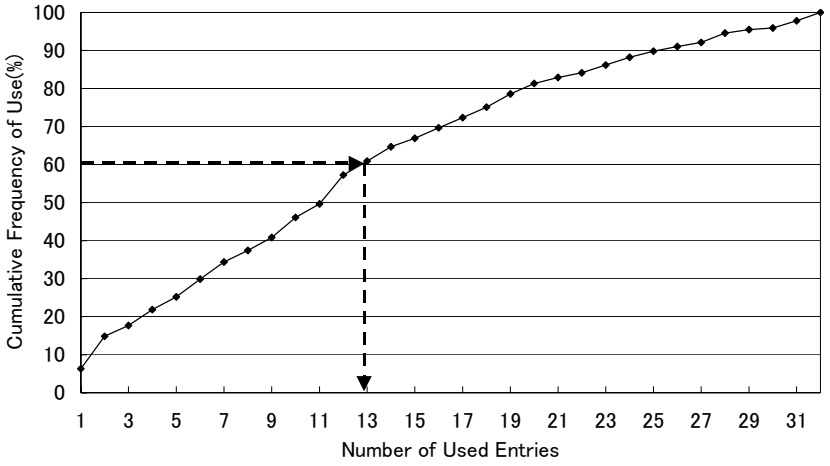


図 5.4　GUB の使用エントリ数の累積頻度分

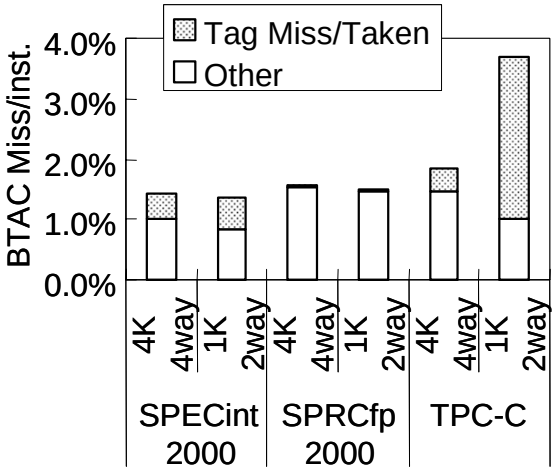


図 5.5　分岐予測テーブル（BTAC）の容量とミス率

C. 第三段階の削減（各種バッファ資源の削減）

前記の 2 段階の削減後，シミュレーションにより各種のリソースの使用状態を求めた。図 5.4 にベースラインプロセッサの整数演算側のリオーダバッファ（GUB）の累積使用頻度分布を示す。このリオーダバッファは 32 エントリであるが，13 個かそれ以下のエントリを使用している場合が全体の 60%であり，14 個以上のエントリを必要とするのは 40%の時間であることが分かる。

このように各種バッファの使用エントリ数の頻度分布を求め，累積使用頻度が 60%になる点を目安とし，最終的には 2 のべき乗のエントリ数に切り上げた数にエントリ数を削減した。また，分岐予測テーブルのエントリ数は 4Kx4way から 1Kx2way に縮小し，TLB のエントリ数も 1024 から 512 に削減した。

この第三段階の削減によりプロセッサコア面積は 51.3mm^2 となった。一方，ベースラインプロセッサと比較して，SPECint 性能は 74.0%，SPECfp 性能は 71.9%，TPC-C 性能は 67.1%となった。この削減で TPC-C 性能の低下に大きく影響しているのは分岐予測テーブルのエントリ数の削減である。図 5.5 に示すように SPECint/SPECfp ではエントリ数を削減してもミス率への影響は軽微であり，逆に小さい分岐予測テーブルの方がミス率は若干低くなる傾向にあるが，

TPC-C ではミス率が倍増している。しかし，このミス率はプログラムのワーキングセットの大きさとテーブルサイズの影響が大きく，一つのプログラムである TPC-C の結果だけを重視することは適当ではなく，合計 26 プログラムの集合である SPECmark の結果を判断基準とした。

D. カスタムマクロ化によるコア面積削減

ベースラインプロセッサコアは，レジスタファイルや TLB CAM などのアレイをレジスタ，セレクタ，EOR などのスタンダードセルを規則的に密に配置した構造で実装している。この設計手法は特別な回路設計が不要であり，比較的少ない工数，期間でこれらのマクロを作ることが出来，通常の相補型の CMOS 回路であるので動作も安定しているというメリットがあるが，トランジスタレベルで専用の回路設計，配置設計を行うカスタムマクロに比べて面積が大きいため，3.2 で述べたカスタムマクロを適用した。

その結果，これらのカスタムマクロの使用により元のレイアウトに比べ，プロセッサコアの面積を 14.0mm^2 縮小し，37.3mm^2 に出来るという見通しが得られた。このカスタムマクロ化による論理的な変更は無いので，性能は第三段階の削減の結果と同じである。

E. コア面積の削減のまとめ

プロセッサコアのリソース削減のまとめを表 5.1 に示す。そして，各段階でのユニットごとの面積を表 5.2 に示す。

表 5.1 リソース削減のまとめ

Step		Baseline core	Small core
1	Instruction Issue	4	2
	Execute Units	2 each	1 each
2	L1 I$/D$	128KB(2way)	16KB(2way)
3	CSE (Commit Stack)	64	24
	Fetch/Store port	16/10	8/4
	GUB/FUB (FX/FP Reorder buffers)	32/32	16/16
	RSA/RSBR	10/10	4/4
	RSE/RSF	8x2/8x2	8/8
	L1I$ Move In Buffer	3	1
	L1D$ Move In Buffer	4	2
	L1$ →L2$ data bus	16Byte	8Byte
	L2$ →L1$ data bus	32Byte	8Byte
	BTAC	4Kx4way	1Kx2way
	TLB	1024	512
4	Register File	Standard Cell	Custom
	TLB CAM	Standard Cell	Custom

表 5.2 リソース削減によるコア面積の変化

Chip Area (in mm²)			Area	Layout Improve	Reduction			
					step1	step2	step3	step4
IU	Array	BTAC	9.33	8.50	8.50	8.50	1.06	1.06
		IBUF	6.22	4.20	2.94	2.94	2.94	2.94
	Logic		15.55	12.62	10.56	10.56	6.54	6.54
	Subtotal		31.10	25.32	22.00	22.00	10.54	10.54
EU	RegFile		10.66	10.66	9.14	9.14	8.14	1.09
	Logic	ALU etc.	11.88	8.54	6.44	6.44	6.44	6.44
	Subtotal		22.54	19.20	15.58	15.58	14.58	7.53
SU	Array	L1I$, Tag	7.18	7.18	7.18	1.15	1.15	1.15
		L1D$, Tag	9.60	9.60	9.60	1.65	1.65	1.65
		TLB	3.10	3.10	3.10	3.10	2.10	2.10
		μTLB	7.22	7.22	7.22	7.22	7.22	0.32
	Logic		23.89	22.45	21.38	16.52	14.04	14.04
	Subtotal		50.99	49.55	48.48	29.64	26.16	19.26
	Total		104.63	93.21	86.06	67.22	51.28	37.33

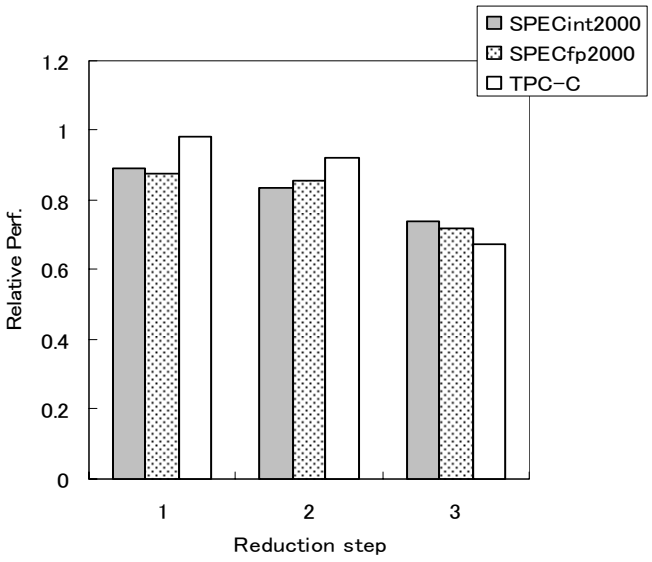


図 5.6 リソース削減による性能変化
(ベースライン コア性能=1.0)

F. 面積削減の性能に与える影響

図 5. 7 にコア面積と SPECint2000 性能を示す。図 5.7 に見られるように，性能はコア面積の 0.448 乗に比例しており，2.1.1 で述べた 0.547 乗よりは若干小さいが，ベースラインプロセッサからリソースを削減するというアプローチでも歴史的トレンドを逆転する形で小規模コアが作れることを確認した。

図 5.7 に見られるように, 第一段階の削減では面積の減少にくらべて性能の低下が大きい。これは，第一段階では命令発行を半減したため実行リソースが余剰になるが，演算パイプライン以外の実行リソースを削減していないためと考えられる。

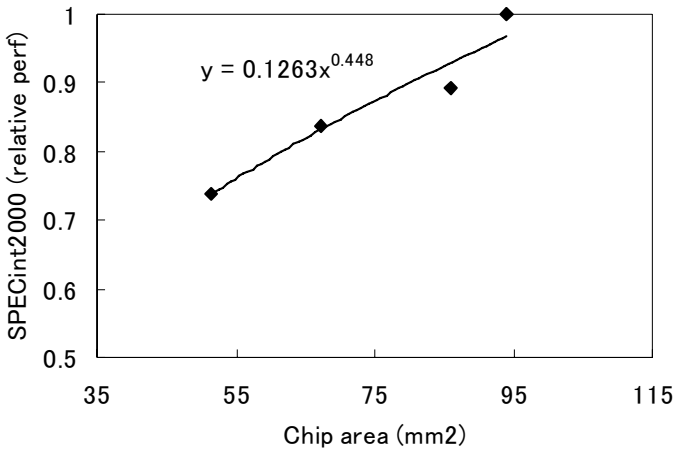


図 5.7 コア面積対相対 SPECint 性能

5.3. 小規模コア チップマルチプロセッサの構造

複数の小規模プロセッサコアが，二次キャッシュとメモリ，外部バスへのインタフェースを共有するマルチプロセッサチップの構成を図 5.8 に示す。各プロセッサコアは MIOP と呼ぶメモリアクセス要求を保持するキューを経由してメモリアクセス要求を出す。複数のプロセッサコアからの要求はラウンドロビン方式により次に処理する要求を選択し，L2$のタグをアクセスする。 L2$にヒットした場合は，Read の場合は L2$のデータアレイから 32B 単位で読み出されたデータを SP と表記された回路で 8B 幅 x4 回の転送に変換して要求元のプロセッサコアに送る。一方，Write の場合は，プロセッサコアから 8Bx4 回で転送されてきたデータを SP 回路により 32B にまとめ WBDQ バッファに格納し，L2$データアレイに書き込む。また，メモリや外部バスへの Write の場合は MODQ 経由でデータを送り出す。

メモリや外部バスから読み込まれたデータは MIDR に格納され，アドレスを含んだ書き込み要求は MIB に格納される。MIB に格納された要求は各プロセッサの MIOP に格納された要求と優先順位調停を行い，アクセス権を得ると L2$への書き込みが行われる。

図 5.8 において，DIMM と表記された DRAM メモリ以外は 1 個のマルチプロセッサチップに集積されている。この小規模マルチコアプロセッサを 90nm プロセスで製造する場合のチップイメージを図 5.9 に示す。

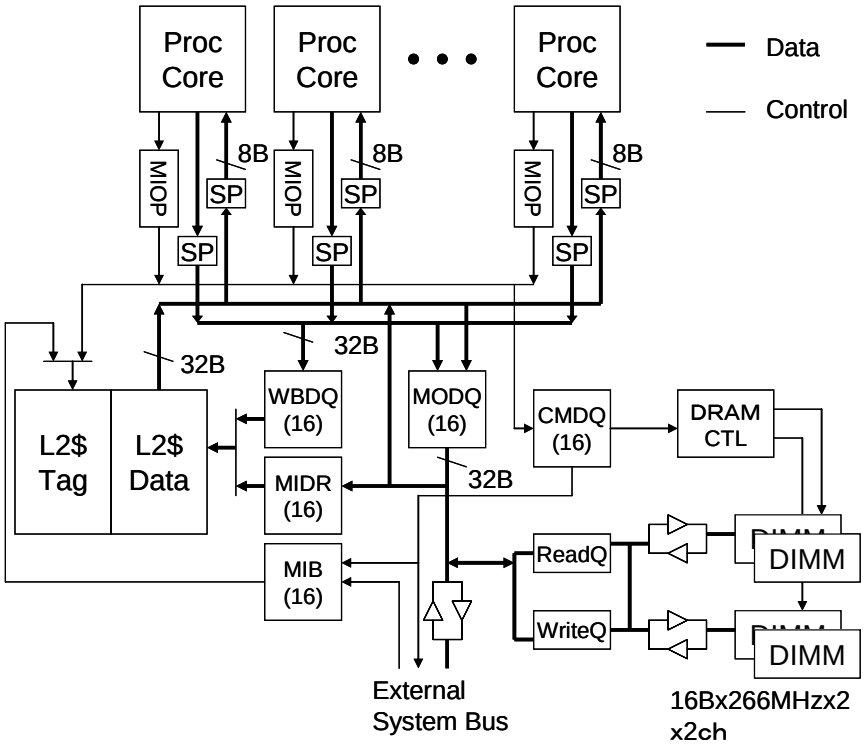


図 5.8 小規模コア チップマルチプロセッサの構成
()内の数字はバッファのエントリ数

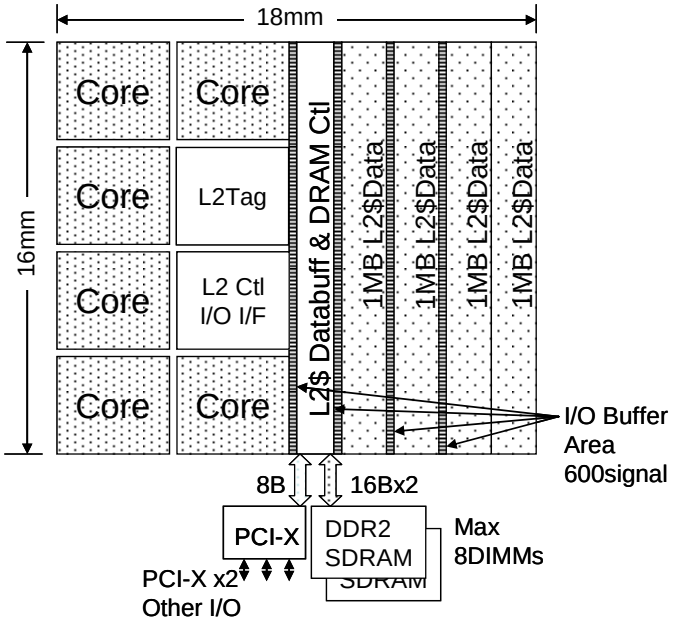


図 5.9 小規模コア チップマルチプロセッサのチップイメージ図

5.4. 小規模コアマルチプロセッサの性能

チップマルチプロセッサは，L2$とメモリを全コアが共有する構造であり，これらの資源の共用にともなう競合による性能低下が懸念されるので，TPC-C ベンチマークを用いて性能評価を行った。

なお, 図 5.9 のチップイメージは 6 コア, 4MB キャッシュのチップであるが, マルチコアの性能スケーラビリティーの評価は 8 コア，および 8MB L2$のケースまで拡大して評価を行った。

5.4.1. 二次キャッシュアクセス競合の影響

図 5.10 に，L2$量をパラメタとして，コア数を増加させた場合の相対 TPC-C 性能を示す。L2$が小さい場合は競合が多く，コア数を増加させても性能向上の程度が少なく，得られる最高性能も低い。一方，L2$容量を 4~8MB にするとコア数を増やした場合の性能改善の度合いが大きい。しかし，4MB から 8MB への L2$の増加では性能改善の程度が鈍化しており，最大 8 コア程度であれば, 面積を考慮すると TPC-C 処理では 4MB 程度が最適である。

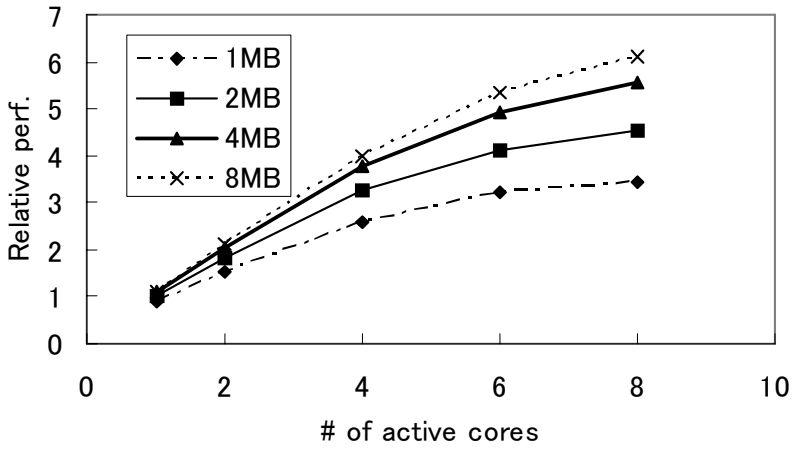


図 5.10　TPC-C 相対性能

図 5.11 は 4MB の L2$を使用した構成での L2$のアクセス頻度とアクセスレーテンシの増加の関係を示している。L1$のミスが発生してから L2$を読み出す最小レーテンシは 6 サイクルであるが, 1 コアの場合でも混雑のため平均 1 サイクルの追加レーテンシが存在し, 平均 7 サイクルのアクセス時間となっている。これが 8 コアになると 1 サイクルあたり 0.23 回のアクセス頻度(各コアは 0.029 回)であり，L2$は 2 サイクルに 1 回のアクセスが可能な構造であるので，46%ビジーの状態となる。この状態では L2$のアクセス権の競合により平均 9.8 サイクルの待ちが発生している。

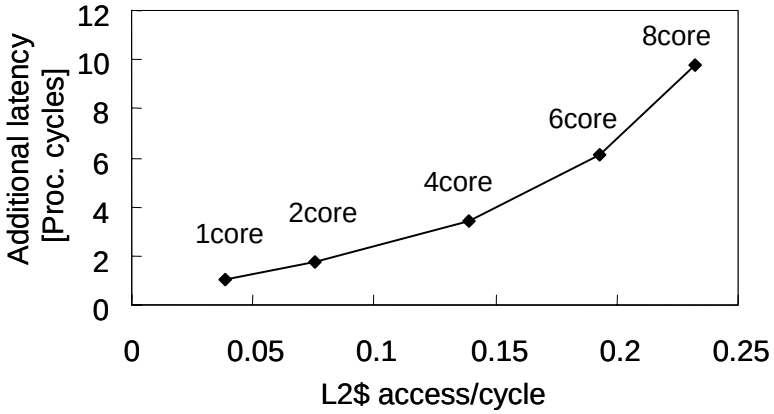


図 5.11　L2$のアクセス頻度とアクセスレーテンシの増加

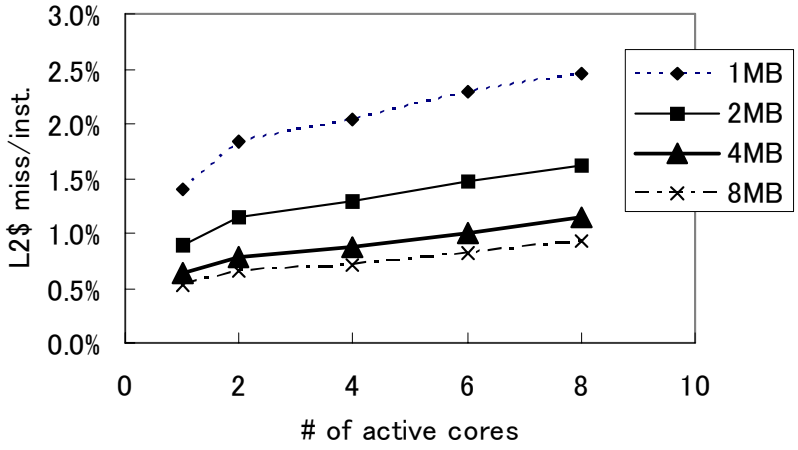


図 5.12 命令あたりの L2$ミス率

5.4.2. 二次キャッシュミス

図 5.12 に命令あたりの L2$ミス率を示す。コア数の増加に伴い L2$のミス率も増加しているが，その増加の程度は比較的緩やかである。例えば 4MB キャッシュの場合，1 コアでのミス率は 0.6%であるが，8 コアに増加してもミス率は 1.1%である。図 5.13 と図 5.14 に L2$アクセスのヒットとミスの内訳を示す。これらのアクセスの内，ミスと表記された部分は L2$をミスした部分である。L2$ヒットのうち，Con.(Constructive) Hit と表記された部分は，そのキャッシュラインへの直前のアクセスが他のプロセッサコアにより行われたヒットを示している。Self Hit と表記された部分は，そのキャッシュラインへの直前のアクセスを自コアが行ったヒットアクセスである。命令フェッチに起因する L2$アクセス (L1I$ミス率) は 5%/命令であり, L2$ミス率は 1 コアの場合は 0.058%,

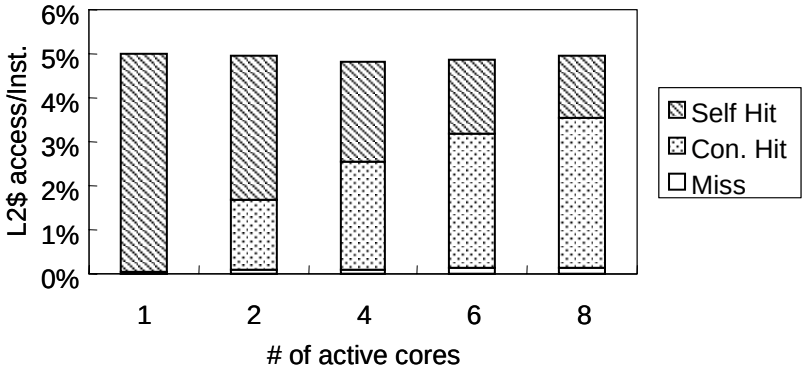


図 5.13 命令フェッチの L2$アクセスの内訳 (4MB L２$)

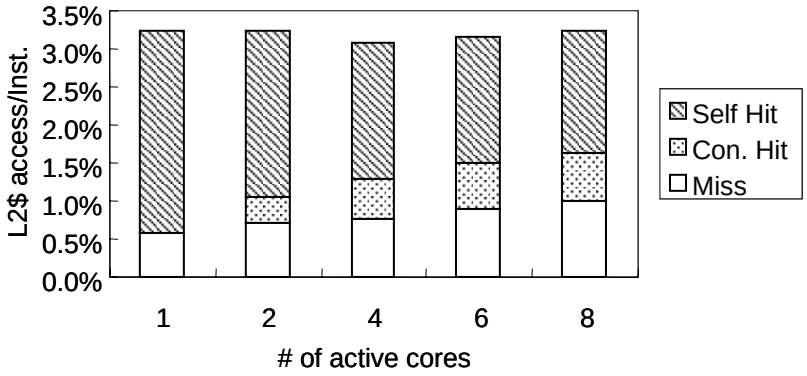


図 5.14 データの L2$アクセスの内訳(4MB L２$)

8 コアでも 0.128%と低く，コア数が増えるに従って Constructive Hit が増えている。TPC-C の命令フットプリントは 4MB キャッシュに殆ど収まり，すべてのコアがトランザクション毎に処理を分担しており基本的に同じプログラムを実行しているので，命令キャッシュミスが増えないと考えられる。

図 5.14 に示すように, L1D$のミスに起因する L2$のアクセスは約 3.2%/命令である。図 5.14 で特徴的なことは，2 コアの場合で 0.35%の Constructive Hit があり，8 コアでは 0.63%の Constructive Hit が見られることである。4 コア以上では Constructive Hit はミスの 60%程度の頻度であり，マルチコアの L2$共用によりミス率を低減している。

但し，このような傾向は全プロセサコアが同じプログラムを実行し，データもある程度の共用を行っているという TPC-C ベンチマークの処理特性に依存しており，全く独立のプログラムを実行するケースでは，コア数の増加に伴うミス率の増加は大きくなる可能性がある。

5.4.3. メモリアクセス競合の影響

図 5.8 では 2 チャンネルの DIMM を接続する構成を示しているが，ベースラインプロセッサではメモリサブシステムとして 1 チャンネルの DDR DRAM を

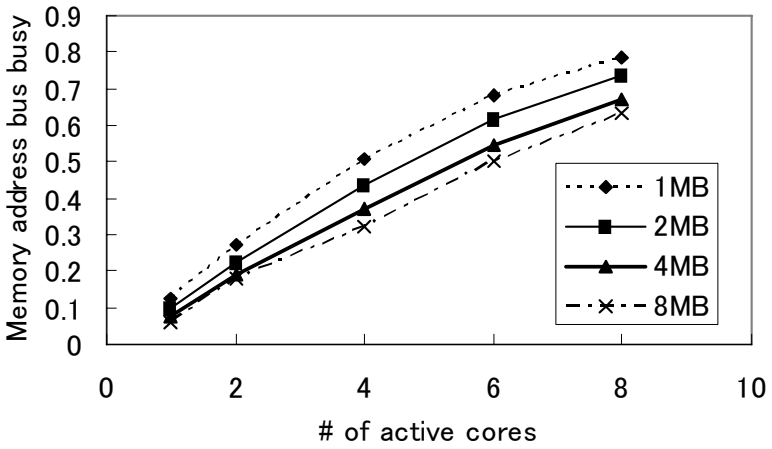


図 5.15　DDR メモリシステムのビジー率

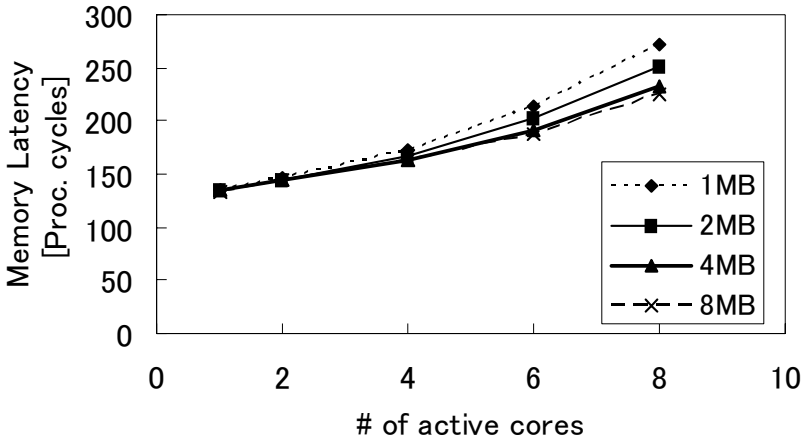


図 5.16　DDR メモリシステムのアクセスレーテンシ

接続する構成をとっている。図 5.15 は，このベースライン構成のメモリサブシステムに小規模マルチコアプロセッサを接続した場合のメモリバスのビジー率，図 5.16 はアクセスレーテンシを示している。

ベースラインプロセッサは小規模プロセッサ 1.5 コア相当の性能であるので，メモリバスのビジー率は 20%程度であり,このビジー率ではアクセス競合によるレーテンシの増加も殆ど見られないので妥当な設計である。しかし，6 コア～8 コアを搭載すると 60%を越えるビジー率となり，競合によりアクセスレーテンシが 60 サイクル以上も増加している。

この状況から，ビジー率を 20%程度に下げるためにはメモリバンド幅を拡大する必要があり，転送速度が 2 倍の 533MHz の DDR2 メモリを採用し，かつ，2 チャンネル化し図 5.8 の構成とした。この構成とすることにより，図 5.17 に示すように，8 コアでもビジー率は 20%以下となり，図 5.18 に示すようにアクセスレーテンシの増加も半減した。両者のメモリサブシステムの諸元を表 5.3 に示す。

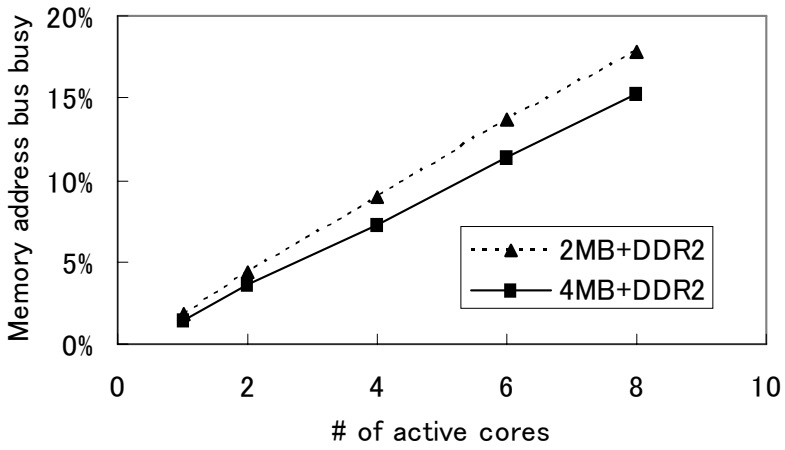


図 5.17 DDR2 メモリシステムのビジー率

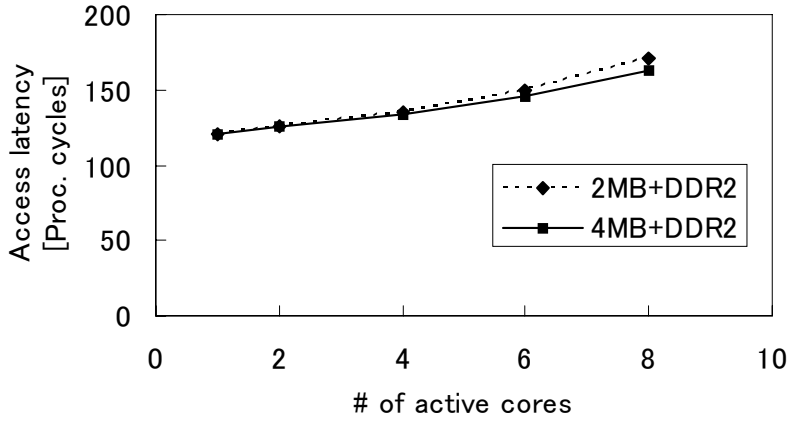


図 5.18　DDR2 メモリシステムのアクセスレーテンシ

表 5.3 ベースラインと DDR2 2 チャンネルメモリシステムの諸元

	DDR ベースのメモリ	DDR2 ベースのメモリ
クロック	133MHz	266MHz
データレート	266MHz	533MHz
DIMM 数	4	4
メモリバンク数	4	4
アドレスバス数	1	2
データバス数	1x16B	2x16B

5.4.4. メモリサブシステムの性能に与える影響

図 5.19 に 2 次キャッシュ容量とメモリサブシステムをパラメタとした相対 TPC-C 性能を示す。メモリを DDR2 2 チャネル化することによる性能向上は，オンチップの L2$の容量を倍増するのと同程度の効果が見られる。また，第 2 章で用いたマルチプロセッサの性能スケーラビリティーの近似式 2.6 式の γ は図 5.19 の 4 MB+DDR2 のケースで約 0.86, 2MB+DDR のケースで約 0.75 である。

図 5.20 に 4MB の 2 次キャッシュを搭載した場合の CPI（Cycle per Instruction, IPC の逆数）の内訳を示す。図 5.20 において L1$+TLBmiss と記された部分は，大部分が L1$ミスに伴う L2$のアクセス時間であり，L2$Miss は L2$のミスに伴うメモリアクセス時間である。図 5.20 に見られるように L2$ミスが CPI の増加の主因であり，この部分はメモリシステムを DDR2 2 チャンネル化することにより減少している。

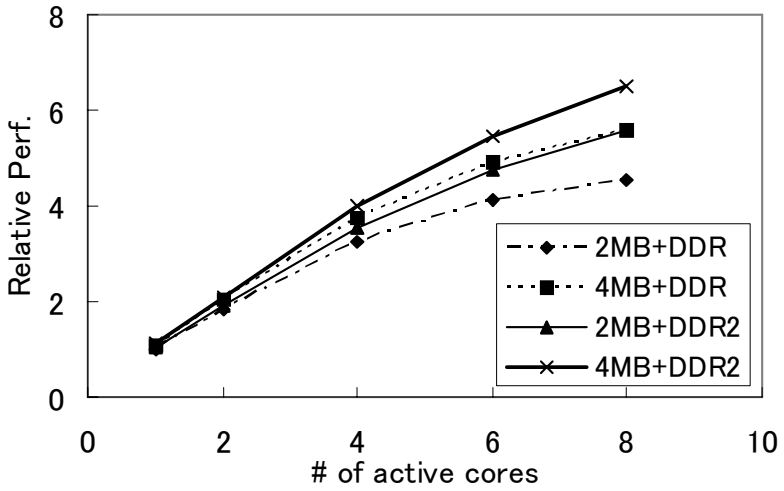


図 5.19 相対 TPC-C 性能

図 5.21 はマルチコアシステム全体の実効メモリアクセスレーテンシを示している。マルチコアでそれぞれ独立の命令列を実行することにより，一つのコアがメモリ待ちとなっても他のコアは命令実行を継続することができ，複数のメモリアクセスがパイプライン的に処理されることになる。図 5.16 や図 5.18 は個々のアクセス要求の発行から結果が戻ってくるまでの時間を計測しているが，図 5.21 はそれを並列に実行されているメモリアクセス要求数で割った実効メモリレーテンシを示している。コア数 1 の場合の実効メモリアクセスレーテンシは図 5.18 と比較すると約 1/1.3 となっている。コア数が 1 の場合はマルチスレッド効果はないが，命令フェッチとデータアクセスは並行して実行されており，また，投機的に実行されるメモリアクセスや，メモリシステムが行う次キャッシュラインのプリフェッチなどにより平均 1.3 個程度のメモリアクセスが並行処理されているためと考えられる。

また,図 5.21 に見られるように，マルチコア化により並列アクセス数が増加し 6 コア，DDR2 システムでは実効メモリレーテンシが 20 サイクル程度に減少している。

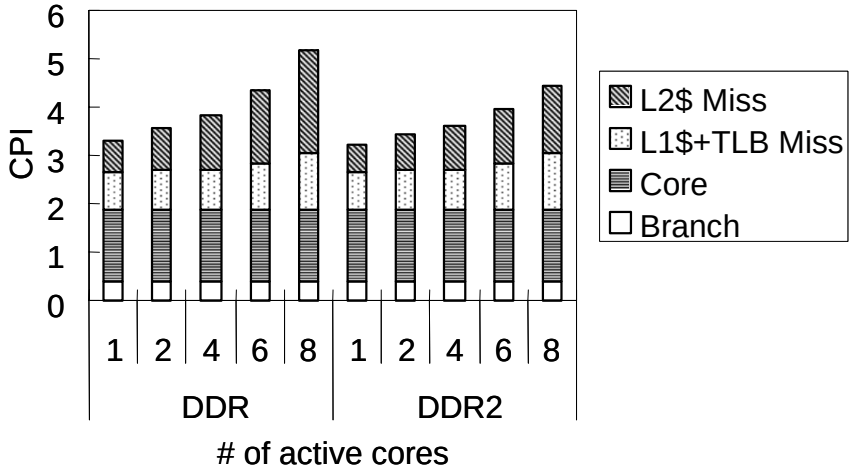


図 5.20　TPC-C 実行時間（Cycle Per Instruction）の内訳 (L2$=4MB)

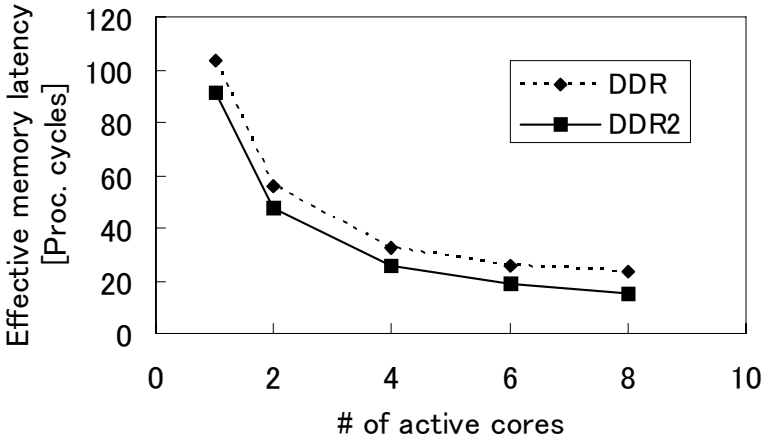
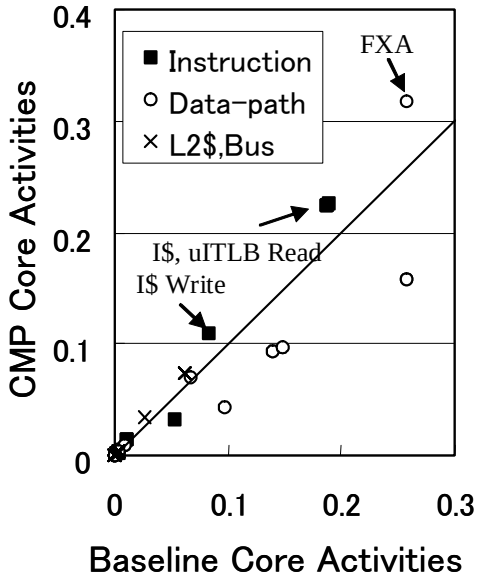


図 5.21 マルチコア全体の実効メモリレーテンシ

5.4.5. 小規模マルチコア化によるエネルギー効率の改善

A. 小規模コアのアクティブ電力密度

5.2.2 で述べた資源削減により作られた小規模コアとベースラインプロセッサの各機能ユニットの動作率の比較を図 5.22 に示す。コア単位でみると，命令系は，L1I$から 1 アクセスでフェッチされる命令数を 4 から 2 に削減したため約



縦軸：2 コア CMP
横軸：ベースラインコア

Instruction， Data-Path の動作率はコア単位
L2$，Bus の動作率は 2 コア分を含む。

図 5.22 プロセッサコア各部の動作率比較

30% 動作率が高くなっている。また，データパスでは，整数演算系（FXA）はパイプラインを 2 本から 1 本に削減したため 20%程度動作率が高くなっているが，この整数演算器を除いたその他のデータパス系のユニットの動作率はベースラインに比べて 30%程度減少している。

小規模コアでは性能当たりのトランジスタ数が少なくなっており，各ユニットの動作率はベースラインに比べて高く，電力密度は増加するとの推定があったが，実際の動作率を反映したアクティブ電力はベースラインコアが 6.51W であるのに対して小規模コアは 2.07W であり，電力密度ではベースラインコアが 140mW/mm^2 に対して小規模コアは 111mW/mm^2 と減少することが判明した。従って，2.1.3 で述べたスイッチ比率 s_f は，TPC-C 処理では，小規模コア化により増加することはなく，逆に若干減少している。

また，L2$，バスはチップ上の全てのコアに共通であり，その動作率はコア数に比例し，図 5.22 で示した 2 コアの場合の動作率はベースラインとほぼ同じである。

B. エネルギー効率

図 5.23 に小規模マルチコアチップの面積と消費電力を示す。ここでのチップ面積は 90nm 半導体プロセスを使用して製造する場合のチップ面積であり，Scaled Baseline は 130nm プロセスで製造されたベースラインプロセッサを 90nm プロセスに設計変更した場合の面積である。図 5.24 に TPC-C 性能と正規化した性能/消費電力を示す。ほぼ同じチップ面積である Scaled Baseline と 2

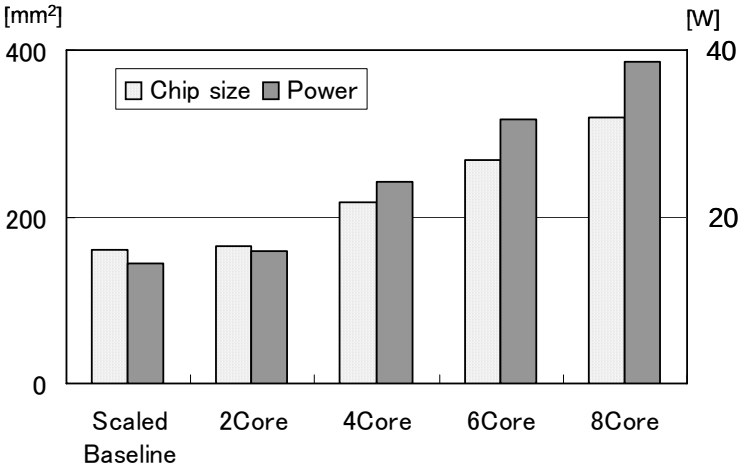


図 5.23 小規模マルチコアチップの面積と消費電力

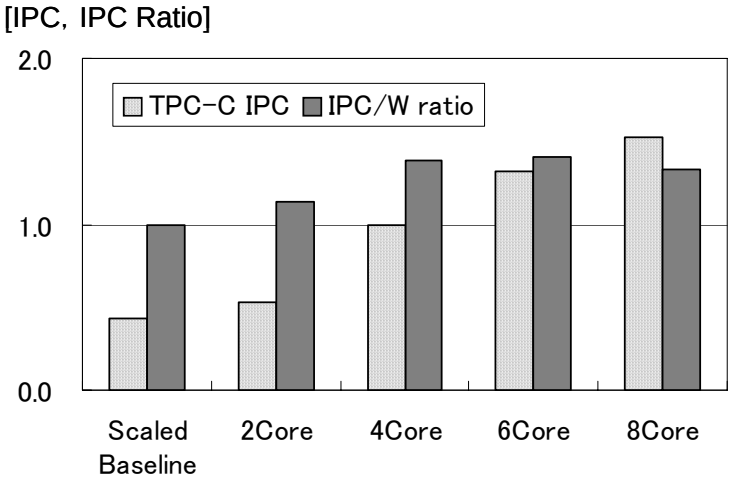


図 5.24 小規模マルチコアチップの性能

コアチップを比較すると，電力は 9%増であるが，スループット性能は 24%増であり，小規模 2 コア化により性能/電力の改善は 13%に留まっている。つまり，リソース削減によりコアサイズを 55%に縮小し，2 コアを搭載するだけでは大きな性能/電力の改善は得られない。また，コア数を更に多くすると TPC-C 性能は向上するが，一方，消費電力も増大するため消費電力あたりの性能は 6 コアで最大となり，8 コアでは若干低下する。電力効率最大の 6 コアの場合での IPC/W は，90nm にスケールしたベースラインプロセッサの 1.4 倍である。

5.5. 低電力回路技術の小規模コアへの適用

5.4.5 では小規模マルチコア化によるエネルギー効率の改善効果を分析したが，もう一つの改善要素は，低電力回路技術によるエネルギー効率の改善である。回路技術によるエネルギー効率の改善としては，1)低電力回路技術による消費電力の削減と 2)フルカスタムマクロ化によるチップ面積の縮小による一層の小規模コア化がある。

5.5.1. 回路技術による低消費電力化の効果

図 5.25 に各種の設計のプロセッサコアの消費電力，図 5.26 に 4MB の L2$と I./O 部の消費電力とその内訳を示す。ベースラインプロセッサを単純に 90nm プロセスに変換し，1V 電源で 2.4GHz で動作させた(a)のプロセッサコアの消費

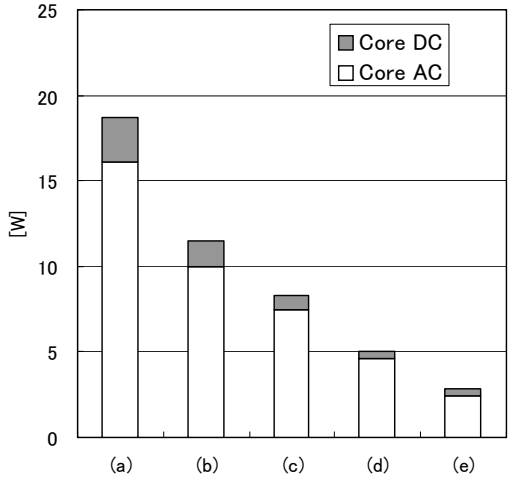


図 5.25　各種設計のプロセッサの消費電力コア消費電力

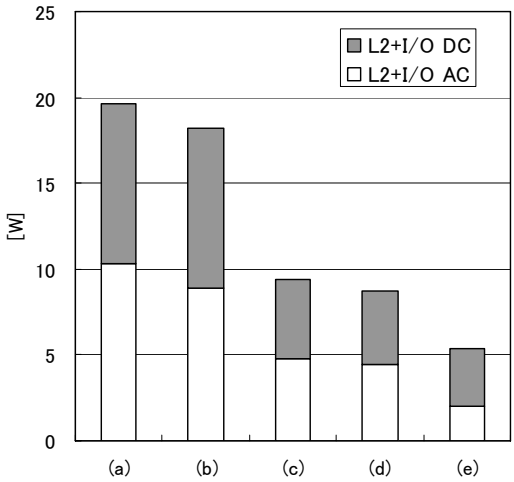


図 5.26　各種設計のプロセッサの L2$+I/O 部の消費電力

(a) 90nm スケールベースライン 1.0V，2.4GHz
(b) 小規模コア 1.0V，2.4GHz
(c) 90nm スケールベースライン 0.8V，1.8GHz
(d) 小規模コア 0.8V，1.8GHz
(e) 小規模＋カスタム低電力化 0.8V，1.8GHz

電力は 18.7W である。5.2.2 で述べた 3 段階のリソース削減を行った小規模コア(b)では，同じ 1V 電源，2.4GHz 動作でも，回路量の減少に伴い消費電力は 11.5W に減少する。これらの(a)と(b)のプロセッサコアを第 4 章で述べた低電源電圧動作可能な設計とし 0.8V 電源で動作させるものが(c)と(d)である。低電圧化により，クロック周波数は 1.8GHz に低下すると見積もられ，低電圧化とクロック低下が相俟って(c)の消費電力は 8.3W，(d)の消費電力は 5.3W と，1.0V 動作に比べて消費電力はほぼ半減する。更に 3.1.2 で述べた Lg=45nm トランジスタの多用によるリーク電流の低減や 3.2 で述べたフルカスタムマクロ化を適用した結果が(e)のプロセッサコアである。0.8V 電源，1.8GHz クロック動作での消費電力は 2.9W に減少している。

また，図 5.26 に示した L2$+I/O 部分の DC 電力には，DDR2 DRAM インタフェースの DC 電力 1.4W を含んでいる。(a)のベースラインプロセッサを 90nm にシュリンクした設計では消費電力は 19.6W である。一方，コアを小規模化した設計(b)の消費電力は 18.2W である。なお，(a)，(b)は同じ 4MB の L2$を搭載しているが，(b)ではコアの小規模化により性能が低下し L2$へのアクセスが減少し，動作電力が若干減少している。1.0V 電源で動作させる(a)，(b)の設計ではリーク電力が DC 電力の大半を占めており，アクティブ電力と同程度の電力を消費している。

(c)，(d)は (a)，(b)それぞれを 0.8V 動作させたものであり，それぞれ消費電力は 9.4W, 8.8W である。電源電圧の 0.8V への低減と，クロック周波数の 1.8GHz への低減により，プロセッサコアと同様に，電力が半分弱になっている。DC 電力もほぼ半減しており，電源電圧の 0.8V 化はリーク電力の削減にも有効であることが明らかである。

第 3 章で述べた低電力回路技術とフルカスタムマクロを適用した(e)では，消費電力は 5.4W に減少しており，(d)と比較しても約 61%の電力である。L=45nm トランジスタの適用により(d)では 2.9W であったリーク電力が 2.0W に減少し，L2$部での RAM マクロのクロックゲートによる電力削減を中心に，(d)では 4.4W であった動作電力が 2.0W に減少している。

電源電圧の低減による効果は，(b)と(d)を比較すると，電力はコア部で 11.5W から 5.1W（44.3%），L2$+I/O 部で 18.2W から 8.8W（48%)に減少している。一方，性能はクロック比例であり 0.75 倍である。電源電圧だけを変える場合はエネルギー遅延積で評価すべきであり，MIPS²/W で評価すると(d)は(b）に較べて 1.17～1.27 倍の改善であり，電源電圧を低減した効果として妥当である。

回路技術による低電力化の効果として(b)と(e)を比較すると，プロセッサコアの消費電力は 11.5W から 2.9W(25.2%)に低減し, L2$+I/O 部の電力は 18.2W から 5.4W(29.7%)に低減しており，約 1/3～1/4 の電力低減を実現している。これらを MIPS²/W で比較すると 1.89～2.23 倍の改善であり，フルカスタム化と Lg=45nm トランジスタの多用によるリーク電流の低減の効果が大きい。

5.5.2. リソース削減と回路技術を総合したエネルギー効率の改善

5.2.2 で述べたリソース削減と，3.2 で述べたカスタムマクロ化による面積削減を合わせると，プロセッサコアの面積は 93.2 mm² から 37.3mm² に縮小できる。更に，このコアを 90nm プロセスに移行することによりコア面積を約 1/2 の 19mm² に縮小することが可能であり，結果として 130nm プロセスで製造されたベースラインプロセッサと同程度の面積で 6 個の小規模コアを集積したチップが実現できると見積もられる。

図 5.27 に 5 種の設計ポイントのチップマルチプロセッサについて消費電力と TPC-C 性能の相対比較を示す。左から順に，(a)ベースラインコアを 90nm プロセスに移行し 2 コアを集積したチップ, (b)4 個の小規模コアを集積したチップ，（c)は（a)の構成で 0.8V 動作させたチップ，(d)は（b)の構成で 0.8V 動作させたチップ, (e)カスタム化等の回路技術を適用し 6 コアを搭載したチップである。これらのチップの諸元を表 5.4 に示す。こられの 5 種のチップ面積の比率は，0.99: 1.045: 1: 1.045: 1 であり，ほぼ同一の大きさである。

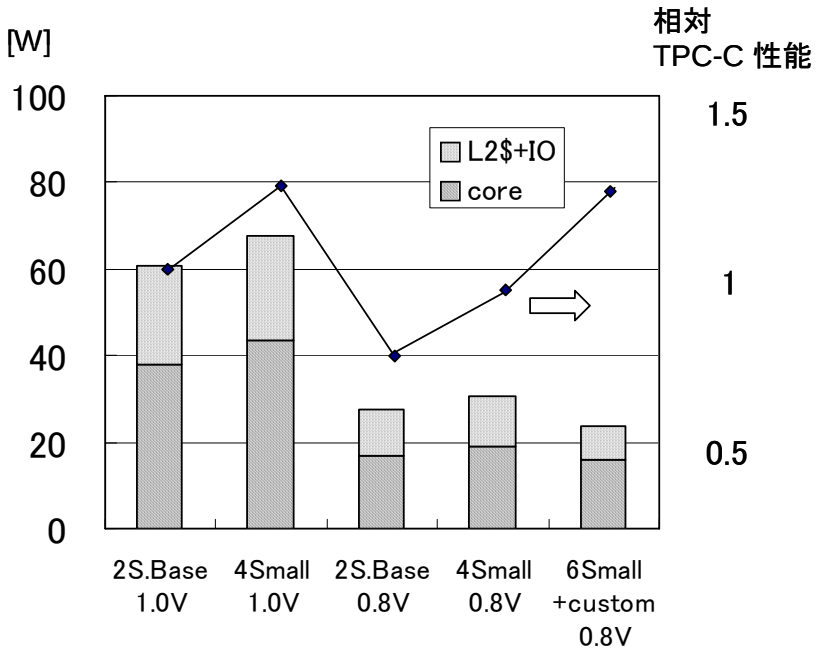


図 5.27　5 種の設計ポイントのプロセッサの電力と TPC-C 性能

表 5.4 5 種の設計ポイントのプロセッサチップの比較

Case		電源電圧 [V]	面積 [mm²]	消費電力 [W]	TPC-性能 相対値	TPC/W 相対値
a	2 Base Core	1.0	207.2	60.7	1	1
b	4 Small Core	1.0	216.6	67.5	1.24	1.12
c	2 Base Core	0.8	207.2	27.7	0.75	1.64
d	4 Small Core	0.8	216.6	30.5	0.93	1.85
e	6 Custom Core	0.8	205.0	23.9	1.24	3.15

(a)は 130nm 世代のプロセスで設計された高性能プロセッサコアを 90nm プロセスに移行することによりほぼ同面積で 2 コア化するアプローチであり, Sun の UltraSPARC IV+[59]，富士通の SPARC64 VI[60]，Intel の Montecito プロセッサ[61]などで実行されている。

(a)と(b)を比較すると，小規模コアを 4 個搭載した方が 90nm にスケールしたベースラインコアを 2 個搭載する場合と比べて 24%高い性能を達成している。

一方，消費電力の増加は 11.1%であり，小規模コア化により TPC/W は 12%改善している。

小規模コア 4 コアを 0.8V の低電圧動作させた(d)と(a)を比較すると，性能は 7%減であるが消費電力は半減しており，TPC/W は 1.85 倍に改善されている。(a)から(b)への小規模マルチコア化単独ではエネルギー効率の改善は小さいが，マルチコアによる性能向上を電圧，クロックスケーリングにつぎ込む(d)の設計により，(a)とほぼ同等の性能を維持し，エネルギー効率の改善が可能である。

また，(e)は(d)と比較して，フルカスタム化によるコア面積の縮小により搭載プロセッサコア数を 1.5 倍にし，TPC-C 性能を 33%改善している。加えて回路技術やフルカスタムマクロの使用により低電力化とリーク電流の低減などを実現し，6 コアにも拘わらず，電力を 21.7%削減しており，総合的にエネルギー効率を 1.7 倍に改善している。

以上のように，本研究で検討した小規模マルチコアと低電力回路設計を組み合わせた(e)の設計により，業界の標準的アプローチである(a)に比べて 1.24 倍の TPC-C スループットと電力当たり 3.15 倍の TPC-C 実行時のエネルギー効率を達成している。

また，第 2 章で構築したモデルである 2.9 式を変形すると 5.1 式となる。

$$P = k \times \left(T \times N^{(\beta-\gamma)} \right)^{\left(1+\frac{2}{\alpha-1}\right)} \qquad \cdots(5.1)$$

5.1 式において α=1.5，γ＝0.8 とし，5.3.2 で得られた β=0.448 を代入し，(a)と(e)のスループット性能比を 1.24 倍として両者の電力比を求めると 0.42 となる。表 5.4 に見られるように(a)と(e)の電力比は 0.394 であり，3.9 式による計算と良く一致している。

5.5.3. 他社プロセッサとの比較

表 5.5 に他社のデュアルコアプロッサの諸元と本論文のプロセサの比較を示す。但し，一般的には消費電力は最大消費電力しか示されておらず，平均消費電力は一律に最大消費電力の 70%であると推定して比較を行った。

Intel の Xeon はクロックを高めるためパイプライン段数を大幅に増やすなどの消費電力増大要因があり，IBM の POWER5 は 1 世代古い 130nm プロセスを用いているために消費電力が大きいが，ベースラインプロセッサを 2 コア集積した(a)のプロセッサは AMD の Opteron や Sun の UltraSPARC 4+といったデュアルコアプロセッサと同じ 90nm テクノロジを用い，平均消費電力も同程度である。これは，本研究のフィージビリティースタディーの消費電力の計算が妥当であることを裏付けている。

表 5.5　他社プロセサとの比較

	テクノロジ	クロック	最大消費電力	平均消費電力(他社は推定)
Intel Dual Core Xeon	90nm	2.8GHz	150W	105W
AMD Opteron 880	90nm SOI	2.4GHz	95W	66.5W
IBM Power5	130nm SOI	1.9GHz	160W	112W
Sun UltraSPARC 4+	90nm	1.8GHz	90W	63W
本論文　(a)	90nm	2.4GHz		60.7W
本論文　(e)	90nm	1.8GHz		23.9W

これらのデュアルコアプロセッサに対して，本論文の小規模マルチコア，低電力技術を適用した(e)のプロセッサは，40%程度に消費電力が減少し，かつ，TPC-C のようなトランザクション処理では(a)のプロセッサより高いスループットを実現しており,通常のデュアルコアプロセッサに比べて大幅な性能/電力比の改善を実現できる。

小規模マルチコアのプロセッサとして，Sun 社が Niagara というコードネー

ムで開発しているプロセッサ[69]がある。このプロセッサは 1 チップに小規模コアを 8 個集積し，かつ，それぞれのコアが 4 スレッドを並列実行するプロセッサであり，高いスループットを有すると発表されている。しかし，このプロセッサのクロック周波数，スループット性能，消費電力などは発表されておらず，現状では，本論文(e)のプロセッサと比較を行うことは出来ない。

5.6. SPARC64 V プロセッサのエラー検出とリカバリ

第 4 章に述べた筆者らのチェックポイント-リカバリを用いる SPARC プロセッサの研究[66]と並行して，富士通のメインフレームプロセッサの開発グループでもメインフレームプロセッサの投機実行，アウトオブオーダ実行化を行う開発[67][68]が行われた。メインフレームプロセッサは従来からチェックストップ-リカバリを行っており，投機実行化の結果としてチェックポイント-リカバリが実現され，両者とも独立にハードウェアエラーからの回復を行うチェックポイント-リカバリ方式を考案した。

SPARC64 V プロセッサは富士通のメインフレームプロセッサをベースとして開発された高信頼 SPARC プロセッサであるが，ハードウェア冗長回路によるメモリや論理回路のエラー検出と投機実行における予測外れから回復する機構を用いてリカバリを行う点で，第 4 章で述べた投機実行のためのチェックポイント機構を用いるチェックポイント-リカバリ方式の実装例となっている。以下において SPARC64 V プロセッサのエラー検出，チェックポイント-リカバリ方式の実装について説明し，ハードウェア量，実行性能の両面においてオーバヘッドの少ないハードウェアエラーからのリカバリが可能であり，信頼度が向上していることを示す。

SPARC64 V プロセッサは大規模サーバへの適用を主目的とし，高い信頼性の実現を目指した。しかし，一方ではチップサイズやコストの制約があり，エラー検出や訂正に必要なハードウェア量と検出，訂正能力のバランスに配慮した設計を行っている。表 5.5 に SPARC64 V プロセッサの各部のエラー検出・訂正方式のまとめを示す。

表 5.5 SPARC64 V プロセッサ各部のエラー検出と訂正方法のまとめ

	エラー検出	エラー訂正
L1I$ Data	パリティーチェック	無効化と再実行
TLB	パリティーチェック	無効化と再実行
BTAC	パリティーチェック	予測失敗と訂正
L1I$ Tag, L1D$ Tag	パリティーチェック ＋二重化	エラー無し出力を選択，エラー側への書き込み
L1D$ Data	SECDED コード	ECC 訂正
L2$ Data & Tag	SECDED コード	ECC 訂正
レジスタ	パリティーチェック	チェックポイント-リカバリ
ALU，シフタ	パリティー予測と比較	チェックポイント-リカバリ
乗算器(/除算)	レジジューチェック ＋パリティー予測と比較	チェックポイント-リカバリ

5.6.1. エラー検出，訂正符号の適用

プロセッサシステムの外部メモリからメモリバス，内部キャッシュ，内部バスを経由してレジスタに格納するまでの伝送でデータの形式や値は変化しない。従って，このデータの流れはパリティーチェックを付加することにより少ないオーバヘッドで 1bit エラーを検出する機能をもたせている。また，ハードウェア量が多く全体の fit 数の大きいメインメモリや 2 次キャッシュ，1 次キャッシュ, および外部ノイズが懸念されるシステムバスについては SECDED コードの適用によりエラー訂正機能を持たせている。

A. データパス

SPARC64 V プロセッサでは, プロセッサチップ内部のエラーを有効に検出するため，データを保持する FF やラッチには基本的に 8bit に 1bit を追加する奇数パリティーを付け，その信号伝送も 9bit 単位でパリティービットを含む伝送を行っている。また，チップ内部でのアドレスの伝送などもパリティービット

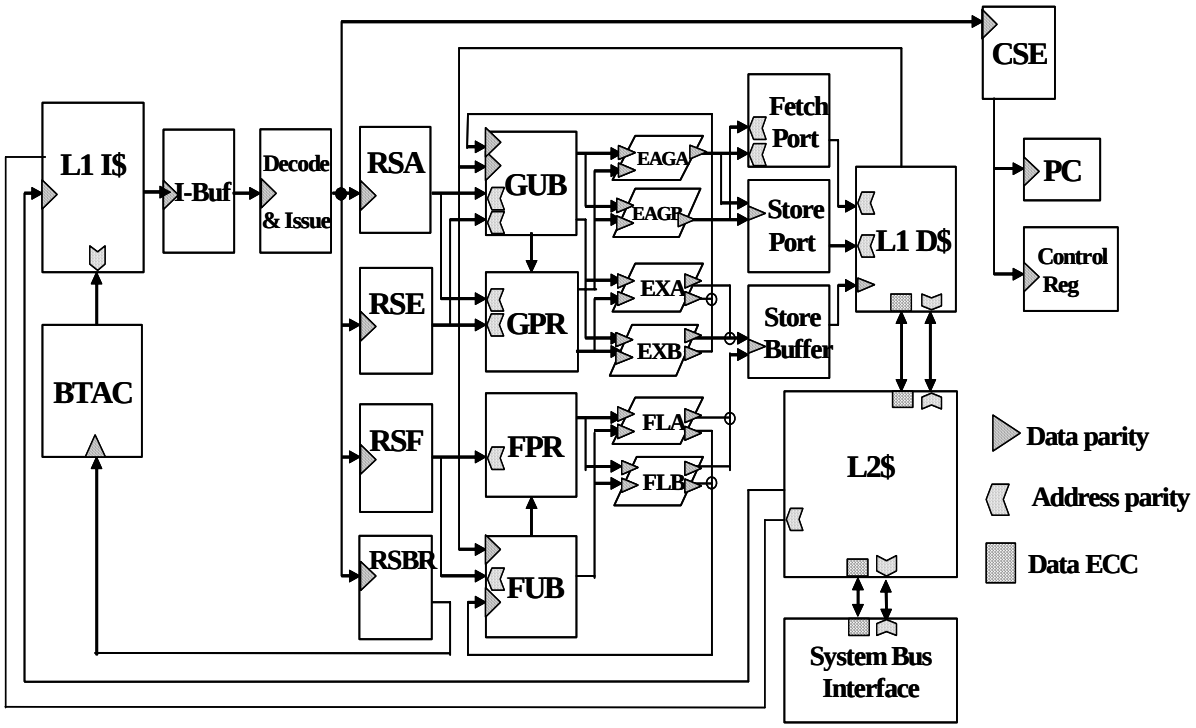


図 5.28 パリティーチェッカと ECC チェッカの配置

を付加している。そして，入力データのパリティーチェックを行う回路を各ブロックの入り口に設け，１ビット誤りを検出する構造を採っている。図 5.28 に SPARC64 V プロセッサのパリティー及び，ECC チェック回路の配置を示す。なお，データが形を変えずブロック内を伝達される場合は出力でパリティーを再生成せず, そのまま伝送して受け側でチェックを行う方式を採り, また, Adder では結果のパリティーを予測して伝播させており，冗長なパリティー生成回路とその回路でのエラー発生のリスクを排除している。

図 5.29 にプロセッサの各部の FF/ラッチ数とその中でのパリティーチェックの付加状況を示す。プロセッサのユニットによりパリティーの付加率にばらつきがあるが，全体としてみると SPARC64 V プロセッサチップ内の 77%のラッチがパリティー付きとなっている。パリティーチェックが付加されていないラッチは，制御回路のステートマシンなどの容易にパリティーチェックが出来ないものや, 図 5.29 で Other と示された部分の入出力端子に付属するバウンダリスキャンラッチのように通常の命令動作には影響が無く，単発故障の発生が信

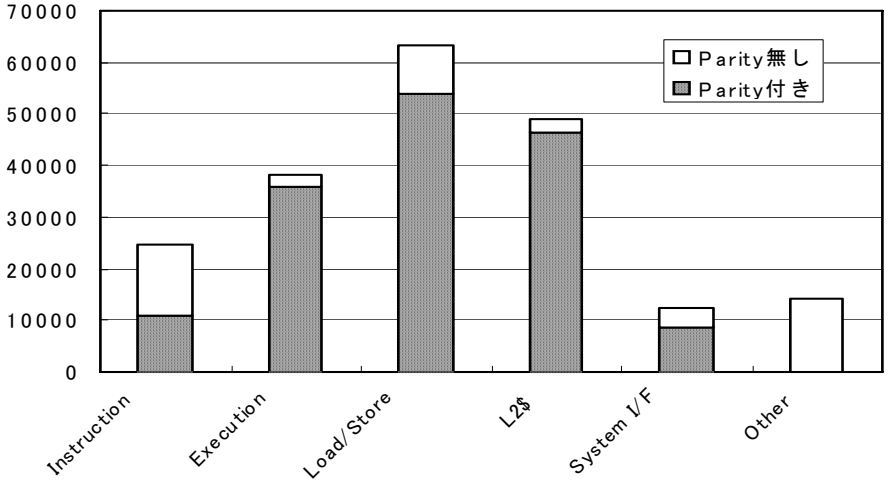


図 5.29 ラッチのパリティーチェックの付加状況

頼度上問題にならない部分である。この Other と表記した部分を除いた信頼度に影響があるラッチのパリティーチェックカバー率は 83%である。

B. メモリアレイ

プロセッサチップ内には，2MB の L2$のデータアレイとそのタグアレイ，各 128KB の L1$のデータアレイとそのタグアレイ，TLB などのアレイがある。これらのアレイの内，オリジナルの状態を保持するアレイについては ECC や二重化により 1bit エラーから回復できる機能を持たせた。

L1I$はプロセッサ内部では変更されず，命令のオリジナルデータがメモリに存在し，エラーが発生してもメモリから読み出すことにより回復が可能である。従って，パリティーチェックによるエラー検出だけを行う。そして，エラーが検出された場合は，当該エントリを無効化し，命令フェッチをやり直す。当該エントリが無効化されているので再実行の命令フェッチは L1I$ミスとなり，正しいデータが L2$あるいはメモリから L1I$に読み込まれ，結果としてエラー訂正が行われる。

L1D$はライトバック（Write Back）キャッシュであり，書き込まれたオリジナルデータを保持するので，エラーが発生するとプロセッサは正しい動作を継続できない。このため，訂正機能が必須であり，SECDED コードを使用している。1bit エラーが検出された場合には，訂正されたデータを要求元に送ると同時にデータアレイにも書き戻しを行い，単発故障を修復している。

L1D$のタグアレイはメモリアドレスの上位ビットや書き込みが行われたかどうかを示すビットを持っており，この部分にエラーが発生すると正常な動作が継続できない。しかし，タグアレイは高速に読み出す必要があることとビットフィールド単位での内容変更があることから SECDED コードの適用が難しい。このため，ビットフィールドごとのパリティーチェックを行うとともに，L1$部のタグアレイと L2$部にある L1$のタグアレイの写しの間で互いに通信することによって二重化した。いずれか一方でエラーが検出された場合は，そのキャッシュラインを互いに他方に通知し，エラーのない方のタグ情報に基づいて L1$を無効化，または L2$へ書き戻す。その後は通常のキャッシュミスと同じ動作で正しいデータが上書きされる。

L1I$のタグはフィールド単位での書き換えは必要ないが，設計の簡素化の点で L1D$のタグと同じ構造を流用した。

TLB は，メモリに格納された論理アドレスと物理アドレスの対応付けを行うページテーブル情報のキャッシュであり，L1I$と同様にプロセッサ内部では変更されない。従って，パリティーチェックによるエラー検出を行い，エラーが検出された場合には当該エントリを無効化し命令を再実行することにより，TLB ミスが生じて正しいページテーブルの内容がメモリから読み込まれることにより，訂正が行われる。

BTAC は，条件分岐処理の高速化のために予測された分岐先アドレスを与える機構であり，それが誤っていても条件分岐命令をコミットする時点で訂正できるので，正常動作の観点からはエラー検出も不要である。しかし，固定故障が発生すると実行性能が低下するので，パリティーチェックによるエラー検出

を行っている。

L2$のデータとタグアレイはSECDEDコードにより1bitエラーを訂正でき，かつ，2bitエラーを検出できる機能をもたせた。

以上のように，SPARC64 Vプロセッサは全てのアレイの1bitエラーから回復可能な設計となっている。

C. 演算回路のエラー検出

加算器などの整数演算器のエラー検出の方法として，図 5.30 に示すように，演算とは独立に結果のパリティービットを予測し，演算結果から作ったパリティービットと比較する方法を用いている。

S[0:63]=A[0:63]+B[0:63]を計算する場合の，和Sの各ビットはS[i]=A[i]⊕B[i]⊕C[i]である。従って，和のパリティーは次のように表わされる。

$$S[0] \oplus S[1] \oplus \cdots \oplus S[63] = A[0] \oplus A[1] \oplus \cdots \oplus A[63] \oplus B[0] \oplus B[1] \oplus \cdots \oplus B[63] \oplus C[0] \oplus C[1] \oplus \cdots \oplus C[63] \quad \cdots (5.2)$$

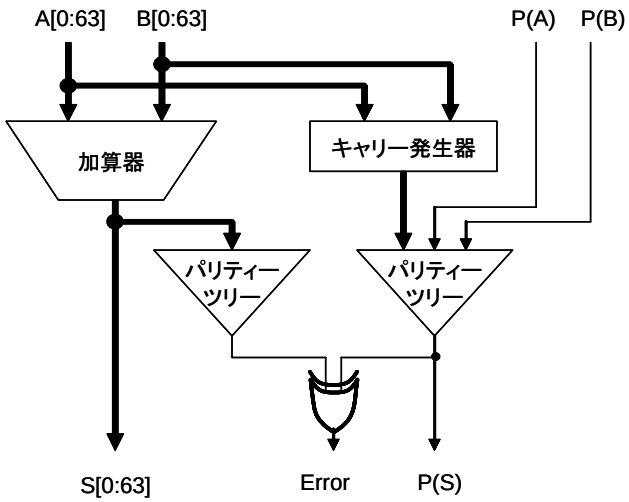


図 5.30 加算器のパリティー予測とチェック回路

5.2 式の右辺の A[0] ⊕A[1] ⊕…⊕A[63]は入力オペランド A のパリティーP(A), B[0] ⊕B[1] ⊕…⊕B[63]は入力オペランド B のパリティーP(B)であるので，C[0] ⊕C[1] ⊕…⊕C[63]を計算すれば和のパリティーP(S)を予測することができる。本方式に基づくチェック回路を図 5.30 に示す。この方法は演算器の出力ネットの 1bit エラーを検出するのと等価である。なお，SPARC64 V プロセッサでは 64bit 一括のパリティーではなく，8bit 単位で和のパリティーと予測パリティーを計算して比較を行っており，和の出力ネットの 8bit ごとに 1bit のエラーを検出できる。

乗算器は $Mod_3(A*B)=Mod_3(Mod_3(A)*Mod_3(B))$であることを利用したレジデューチェックを用いている。2 進数の各ビットを見ると 2^0 のビットは 10 進数の 1 であり 3 で割った剰余は 1，2^1 のビットは 10 進数の 2 であり 3 で割った剰余は 2，2^2 のビットは 10 進数の 4 であり 3 で割った剰余は 1，2^3 のビットは 10 進数の 8 であり 3 で割った剰余は 2 というように 2 進数の各桁で剰余は交互に 1 と 2 になる。これを利用して図 5.31 に示すように Carry Save Adder のツリーを用いて各桁の剰余である 1 と 2 を加算することにより，割り算器を用いることなく簡単に高速で 64bit の 2 進数の Mod_3 求めることができる。図 5.32 に示すように，A，B の入力の Mod_3 を図 5.31 の回路を用いて求めると，それらの積の Mod_3 は簡単な組み合わせ回路で求めることができる。そして，これと積

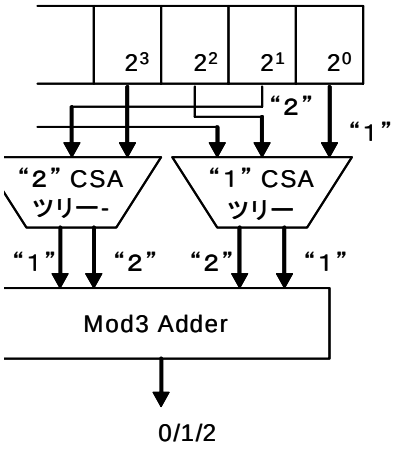


図 5.31 Mod_3 の計算法

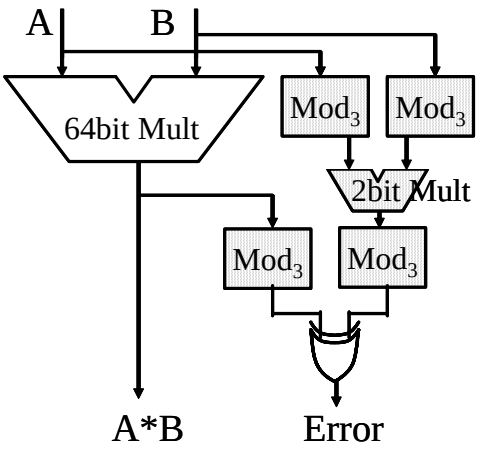


図 5.32　Mod_3 を用いた乗算器のエラー検出

P の Mod_3 とを比較し，不一致があればエラーと判断する。この構成も乗算器の出力の誤りを Mod_3 レジデューによりエラー検出するのと等価である。

以上で述べたデータパスのパリティーチェックと演算器のチェック回路に必要なトランジスタ数は，これらの回路全体の 10%であり，全体を二重化して比較する方式[19]に比べてエラー検出に必要なハードウェアが少ない。

5.6.2. SPARC64 V プロセッサの投機実行，リカバリ機構

5.1.3 で述べたように，各命令の実行結果は，一旦，GUB，FUB と呼ぶリオーダバッファに格納され, コミット時点でアーキテクチャレジスタである GPR, FPR レジスタに書き戻される。アーキテクチャレジスタはコミットされた命令の実行結果だけを格納し，コミットされていない仕掛りの命令の実行結果は GUB，FUB に格納されている。つまり，アーキテクチャレジスタはコミット済みの命令の実行によるチェックポイント状態を保持し，GUB，FUB はコミット前の命令の実行による Active 状態を保持している。

このため，条件分岐命令の予測が外れた場合には，条件分岐命令をコミット時にリザベーションステーションや GUB/FUB，実行パイプラインの各種レジスタなどのアクティブ状態を保持するリソースを初期化することにより誤った投機実行の影響はクリアされ，分岐命令の直前の命令までの実行が完了した状態となる。そして，予測の外れた条件分岐命令を再実行すると，既に条件コードを計算する命令はコミットしているので，正しい方向へ分岐が行われる。

正しい分岐先の命令フェッチは分岐命令のコミットを待たずに，分岐予測の誤りと分岐先アドレスが判明した時点で開始され，新たに読み出された命令は I-BUF に格納される。これにより，分岐予測外れの回復に必要な時間を短縮している。

なお，L1D$への書き込みが実行されると取り消すことは出来ないので，他の

命令の実行結果の GPR への書き込みと同様に，ストア命令のコミット時に書き込みが実行される。一方, 投機的に実行された命令により引き起こされた L1I$や L1D$へのキャッシュラインのフェッチは取り消すことは出来ないが，実行結果の正しさには影響を与えない。

5.6.3. ハードウェアエラーからのリカバリ

スーパースカラ実行であるので，図 5.33 に示すように命令は並列に実行されている。ここで命令 n の実行中にエラーが発生したとすると，エラー検出信号がコミット機構に送られる。コミット機構はエラー検出信号を受け取ると，命令フェッチ，命令発行，命令コミットなどを停止し，アクティブ状態のバッファをクリアし，投機実行の場合と同様に，直前のチェックポイントに戻る。キャッシュをミスしたロード命令がメモリをアクセスしているようなケースもあり，これらの動作が全て終了するのに十分な時間待ち，プロセッサ内部を静粛化し，次に，コミットした最後の命令の次の命令アドレスから命令をフェッチ

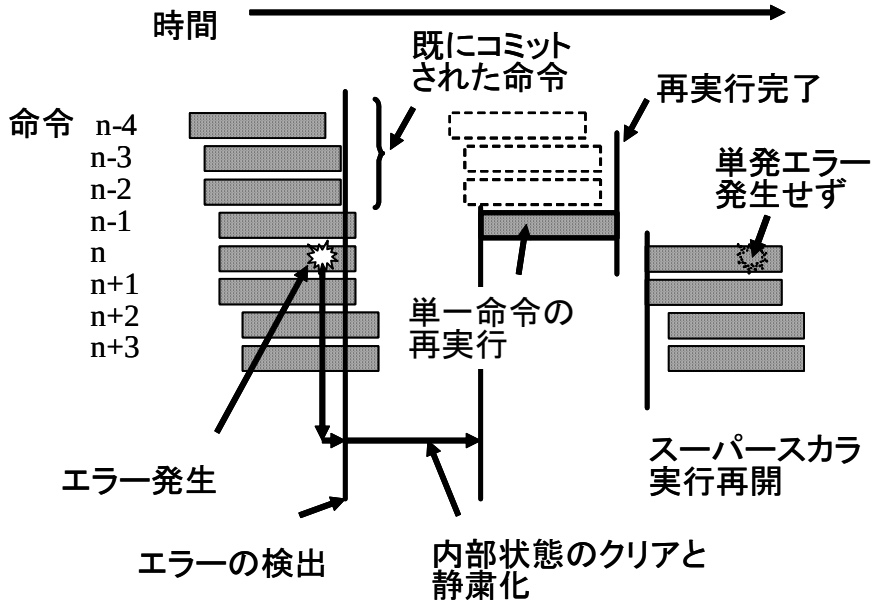


図 5.33 チェックポイント-リカバリによるエラー訂正

して実行の再開を指示する。

命令実行の再開に当たって，成功の確率を高めるために最初は 1 命令だけを発行し，その命令がエラー無く完了すると通常の最大 4 命令同時発行状態に移行する。エラーの原因が中性子ヒットのような単発的なものであると，再実行時に再度エラーが発生する確率は無視できる程度であるので，結果的にエラーを訂正することが出来る。

一方，固定故障である場合は，再実行を行っても同じエラーが検出される。このため，同一命令での再実行が 4 回繰り返されると訂正不能な固定故障であると判断し，マシンチェック状態に移行する。マシンチェックになるとプロセッサの実行は停止し，エラーの発生をソフトウェア割り込みによって通知し，オペレーティングシステムにエラー処理をゆだねる。

再実行はプロセッサ内部の動作が静まった状態で単一命令を実行するので，プロセッサ内部の信号間のクロストークなどの内部雑音がエラーの原因であっても再実行すると成功する場合がある。

また，スーパスカラプロセッサでは各種のバッファを使って効率的に処理を行っており，バッファが一杯になった状態で何かのイベントが起こると顕在化するようなエラーが存在しうる。このような場合でも，再実行を行う時点ではアクティブ状態の資源をリセットし，これらのバッファもクリアされてしまうので，再実行が成功し実行を継続することができる場合がある。

5.6.4. ハードウェアエラー検出と訂正の効果

SPARC64 V プロセッサチップは約 30Mbit のメモリを持っており，典型的な値として 0.001fit/bit のソフトエラー率を想定すると，メモリ部の総 fit 数は 30000fit となる。これは 3.3 万時間の平均故障間隔であり，3.8 年に 1 回のエラー頻度である。先に述べたように SPARC64 V プロセッサはこれらのメモリアレイの 1bit 単発故障を全て訂正することが出来るので，30000fit のエラー要因

を除去している。

また，チップ内の FF の数は約 200K 個であり，これらのソフトエラー率を 0.001fit とすると FF の総 fit 数は 200 となる。一方，組み合わせ論理回路への中性子ヒットによりラッチが反転する程度のノイズが発生しても，ノイズが比較的小さい場合はゲートを伝播している間にパルスが小さくなり消えてしまう場合が多い。更に，後続のゲートで論理的に信号が伝播しない確率があり，また，ノイズパルスが次の FF の入力に到達しても，それがラッチされるタイミングで到着しないとエラーにはならないので，ラッチと比較して組み合わせ論理回路がエラーを起こす確率は小さい。クロック周波数などの条件が異なるためそのままの値を使用することはできないが，[5]では組合せ回路の中性子ヒットに起因するノイズがラッチされる確率は 1/1000 程度と報告されている。

SPARC64 V プロセッサのトランジスタ数を見ると FF を構成するトランジスタ数と論理を構成する組み合わせ回路のトランジスタ数は同程度であるが，前記のフォールトの顕在化比率を考えると組み合わせ論理回路のエラーは FF のエラーに比べて少なく，プロセッサチップ全体の fit 数はほぼラッチで決まり，200fit 程度と考えられる。

プロセッサチップ内の 83%の論理回路がパリティーチェックでカバーされているので，このプロセッサにおいて検出されないエラーが発生する確率は 3000 年に 1 回のオーダである。

また，エラー検出回路がメモリアレイ以外のエラーを検出する確率はプロセッサチップあたり 600 年に 1 回程度であり，128 プロセッサチップを使う大型サーバの場合では 5 年に 1 回程度の確率である。

SPARC64 V プロセッサでは，レジスタについてはパリティーチェックによるエラー検出だけであり，アーキテクチャレジスタなどのリカバリ時にクリアされないレジスタの内容がエラーにより変更されてしまった場合には回復することが出来ない。全体で 200K ビットのラッチのうち，これらのエラーが許容できないビットは約 15%である。

従って，論理回路のエラーの 83%を検出し，85%のエラーに対してはリカバリが有効に働くので，全体としては 70.6%のエラーについて回復が行われる。つまり，SPARC64 V プロセッサでは，エラー検出と投機実行用チェックポイントを用いるリカバリにより，平均故障間隔を 3.4 倍に改善している。

128 プロセッサのサーバの場合は 5 年に 1 回，論理回路のエラーが検出されリカバリが行われるが，33 年に 1 回程度は回復できないエラーが発生すると見積もられる。また，検出されないエラーも 25 年に 1 回程度発生するので，両者を合わせた 128 プロセッサシステムの回復不能な平均エラー間隔は 15 年程度と見積もられる。

このようにエラーの発生，検出頻度が低いことから統計的に有意な実測データは得られていないが，検出された論理回路エラーの多くが再実行により訂正され処理を続行できていると考えられるデータが得られている。

5.7. まとめ

消費電力の増大の問題については, SPARC64 V プロセッサをベースにプロセッサコアの小規模化と低電力回路技術の適用による対策を研究した。

- 第 2 章に述べたプロセッサコアの小規模マルチコア化を SPARC64 V プロセッサに適用し，プロセッサコアの面積を 55%に削減できることを示した。
- また, フルカスタムマクロの使用によりベースラインプロセッサコアの 40%の大きさである 37.3mm^2 に出来るという見通しが得られた。
- この小規模プロセッサコアは，ベースラインプロセッサと比較して SPECint 性能は 74.0%，SPECfp 性能は 71.9%，TPC-C 性能は 67.1%であり，シリコン面積あたりの性能は約 1.7 倍に改善される。
- 90nm 半導体プロセスを使用する場合，0.8V の電源電圧が性能/電力の点で最適であることを示した。
- 0.8V の電源電圧で動作するレジスタファイルや SRAM などのカスタムマクロを設計し，1.8GHz クロックで動作する回路が実現可能であることを示した。

この小規模コアを使用することにより,

- 130nm プロセスで製造されたベースラインプロセッサと同程度の面積で 6 個のコアを集積した小規模マルチコアチップが実現できると見積もられる。
- 小規模コア化と低電力化技術を組みあわせることにより, ベースラインプロセッサコアを 2 コア化した標準的なアプローチのチップと比較して，24%高い TPC-C 性能を 39.3%の消費電力で実現でき，TPC/W を 3.15 倍改善できることを示した。

電源電圧の低減は低電力化には有効な手段であるが，中性子などによるラッチや SRAM のエラー率は，0.8V 動作の場合は，1.0V 動作に較べて 1.5～2 倍に増加すると報告されている。このため，単純に低電源電圧化することは信頼度上問題であるが，本研究の小規模マルチコアプロセッサでは投機実行用のチェックポイント-リカバリ機構を利用するハードウェアエラーからのリカバリを組み合わせることにより，信頼度を確保することができる。

SPARC64 V プロセッサはエラー検出回路と投機的実行用のチェックポイント-リカバリを用いた誤動作からの回復機能を備え，少ない追加ハードウェアで高信頼度を実現している。

- SRAM のエラーは全ての 1bit エラーを検出，訂正を可能とし，高信頼度を確保した。
- チップ内の 83%の論理回路をパリティーチェックでカバーしており，ほぼ 83%のハードウェアエラーを検出可能である。
- このエラー検出に必要なハードウェア量は，全体のハードウェア量の約 10%である。
- チェックポイント-リカバリ機構は投機実行用のものを利用しており，追加ハードウェアはほぼゼロである。
- 一般的なプロセッサも大規模 SRAM には ECC を適用おり，これらと比較すると SPARC64 V プロセッサはコア論理回路部の 10%のハードウェアを追加しただけであり，チップ全体の面積の 3～4%という少ないオーバヘッドである。
- チェックポイント-リカバリにより，検出されたエラーの 85%程度について回復が可能であると見積もられる。

中性子ヒット等に起因する SRAM セルやラッチのソフトエラー率を 0.001fit とすると，エラー検出，訂正を行わない場合のこのプロセッサチップの論理回路部のエラーによる fit 数は 200fit（平均故障間隔 500 万時間=570 年）と見積

もられる。単一のプロセッサチップとしては長い時間であるが，128 プロセッサを使う大規模サーバでは約 5 年に 1 回の頻度となり十分ではない。これに対して，チェックポイント-リカバリにより，以下のように信頼度を 3.4 倍に向上した。

- 83%の検出率と 85%のリカバリ成功率から，チップ全体のラッチのエラーの内，70.6%については回復が可能であり，エラーを見逃したり，リカバリが失敗したりする確率は 29.4%に減少する。従って，平均故障間隔は 3.4 倍に改善する。
- エラー検出回路がラッチのエラーを検出する頻度はプロセッサチップあたり 600 年に 1 回程度であり，128 プロセッサチップを使う大型サーバの場合，エラー検出，訂正を行わないと 5 年に 1 回程度の頻度でエラーが発生すると見積もられるが，本研究の技術を適用した，SPARC64 V プロセサを使うことにより，検出漏れ，およびリカバリの失敗によるシステム停止は 15 年に 1 回程度となると見積もられる。

このようにエラーの発生頻度が低いことから，上記の見積もりを裏付ける統計的に有意なデータは得られていないが，検出された論理回路エラーがチェックポイント-リカバリによる再実行で訂正され，処理を続行できているというデータが得られている。

第 6 章 結論

6.1. 本研究の成果

6.1.1. 研究のアプローチ

半導体の微細化とそれに伴うトランジスタの性能向上とトランジスタ数の増加がマイクロプロセッサの性能向上の主要なドライビングフォースであるが，消費電力の急増，中性子ヒットやその他のノイズによる誤動作の危険性の増加が今後の性能向上の障害となってきている。

マイクロプロセッサの消費電力は，既に 100W を超え，従来の設計手法では消費電力が制約となりこれ以上の性能向上が難しい状況になっている。これに対して，マイクロプロセッサの性能とトランジスタ数の年次推移を分析し，小規模なプロセッサコアの方が最近の大規模プロセッサコアに比較してチップ面積あたりの性能，消費電力あたりの性能が高いことに着目し，プロセッサコアのリソース削減による小規模化と，小規模プロセッサコアを複数個集積するチップマルチプロセッサを研究した。

高い精度で検討を行うため,既設計の SPARC64 V プロセッサをベースとし，回路を削減して小型のプロセッサコアを作り，これを複数個集積するチップマルチプロセッサの手法と，低電圧動作による低電力化などの回路的手法を組み合わせて，小規模マルチコアプロセッサの性能と消費電力について詳細に検討を行い，性能/電力比の高い方式を提案した。

また，中性子ヒットなどによる誤動作に関して，本研究ではエラーを検出し訂正を行うマイクロアーキテクチャによる信頼度向上について検討を行った。エラー訂正の手法として，投機実行を行うプロセッサでは分岐予測外れから回復するためのチェックポイント-リカバリ機構が存在することに着目し，これを

エラー訂正用のチェックポイント-リカバリにも使用することにより，追加のハードウェアや実行時間の損失の殆ど無いエラーリカバリ方式を提案した。

6.1.2. 小規模マルチコア化による低電力，高性能化技術の確立

プロセッサの性能とトランジスタ数の年次推移を逆に辿る形で小規模コア化を行うと，面積削減に比較して性能の減少は少なく，同一面積のチップにおいてスループット性能の改善が可能であり，また，電源電圧とクロック周波数を低減し，並列化によるスループット性能向上の一部を低電力化に振り向けることにより，エネルギー効率を大きく改善したチップマルチプロセッサが実現できることを示した。

また, 90nm 半導体プロセスのパラメタと設計ルールを用いて回路設計を行い, 高速動作を追求する 1.0～1.1V の電源電圧よりも 0.8V が性能/電力の点で最適であるとの結論を得た。しかし，低電圧動作を行うと，動作マージンが低下し，また，動作速度も低下する。このため， 0.8V の電源電圧で動作可能な SRAM やレジスタファイルの設計を行い，回路シミュレーションにより低電圧においても 1.8GHz クロックが実現できるマクロが設計可能であることを示した。

そして，小規模マルチコアプロセッサと低電力回路技術の組合せにより，低電力，高性能のプロセッサを設計する技術を確立した。

0.8V の低電圧動作を行うと FF などの回路に蓄積される電荷が減少し，1.0V 動作に較べてアルファ線や中性子ヒットによる誤動作が 1.5～2 倍に増加するという問題があるが，チェックポイント-リカバリによる高信頼化と組み合わせてエラー回復を行うことにより，安定な動作を確保することが可能である。

6.1.3. 投機実行用チェックポイントを利用するリカバリ技術の確立

マイクロプロセッサの論理回路にエラー検出回路を設け，エラーが検出され

た場合には直前のチェックポイントに戻り，実行を再開することにより単発エラーからの回復を行う方式の研究を行った。

従来の方式では，チェックポイントを作成するために大量のメモリへの書き込みが必要であったり，チェックポイント専用のレジスタファイルを追加し多くのレジスタ状態を書き込むなどの必要があったりして負担が重かったが，本研究の方式は投機実行に必須のチェックポイント-リカバリ機構を利用し，非常に少ない追加回路でエラーからの回復を可能にしている。また，メモリへの書き込みも必要とせず，実行時間の点でも全く負担がない方式である。

投機実行用チェックポイント-リカバリの実装例である SPARC64 V プロセッサでは, チップ全体のトランジスタの約 10%がパリティーや ECC ビットの追加, パリティーチェッカなどに用いられている。しかし，リカバリ機能を持たないプロセッサでもキャッシュなどの大容量メモリアレイにはパリティーチェックや ECC を付加することが一般化しており, このようなプロセッサと比較すると, SPARC64 V での追加ハードウェアは論理回路や小規模アレイのパリティーチェックに必要となる論理回路部の 10%程度の面積（SPARC64 V では約 10mm^2）であり，チップ全体から見ると 3～4%の面積増加という少ないオーバヘッドで論理回路のエラー検出を実現している。

また，ハードウェアエラーからのチェックポイント-リカバリの適用により 83%のエラーを検出し，それらの 85%についてリカバリが可能と見積もられ，信頼度を 3 倍以上に改善することが可能となった。これは，ラッチのソフトエラー率を 0.001fit/bit とすると, 128 プロセッサを使用する大規模サーバの場合, 5 年に 1 回のエラー頻度を 15 年に 1 回に改善することに相当する。

以上の様に，半導体の微細化により，エラーの危険性が増加する将来のマイクロプロセッサについても，動作信頼度を大幅に改善することを可能とする技術を確立した。

6.1.4. スーパスカラプロセッサへのチェックポイント-リカバリ技術と小規模マルチコア技術の適用

低電力化に関しては，SPARC64 V プロセッサをベースラインとして，小規模マルチコア，低電力回路設計のフィージビリティーを研究した。

5.2.2 に述べたように，3 段階のリソース削減とフルカスタムマクロの適用により，プロセッサコアの面積を 40%に縮小した。一方，性能は，ベースラインプロセッサと比較して SPECint 性能は 74.0%，SPECfp 性能は 71.9%，TPC-C 性能は 67.1%となり，第 2 章で述べた小規模コア化により面積あたりの性能改善が可能であるとの見通しを，商用のハイエンドプロセッサの設計をベースとして性能，チップ面積を詳細に検討し，実現可能であることを示した。単に小規模コアを複数集積するだけでは性能/電力比は 12%の改善であり省電力効果は大きくないが，複数コアによるスループット性能の向上分をクロックの低減，電源電圧の低減に廻すことにより大きな省電力化が可能であり，小規模マルチコア化と電源スケーリングにより，性能/電力比を 1.85 倍に改善できることを示した。

また，低電源電圧で動作する RAM やレジスタファイルなどのフルカスタムマクロを適用することにより，コア面積の縮小効果と低電力回路設計技術の相乗効果により，ベースラインプロセッサのコアを 2 個集積する標準的な設計に比べ，小規模コアを 6 個集積する本研究のプロセッサチップは，24%高いスループット性能を 40%弱の電力で達成し，性能/性能比を 3.15 倍改善できる見通しを得た。

0.8V の低電源電圧動作では中性子起因の平均故障間隔が 1/1.5～1/2 になるが，投機実行用チェックポイントを利用するリカバリにより平均故障間隔を改善することが可能である。

全体として，本研究の小規模，低電力マルチコア技術と投機実行用チェック

ポイント-リカバリ機構をもちいるハードウェアエラーからのリカバリ方式との組み合わせにより，高性能，低消費電力と，高信頼性をバランスよく両立させる設計が可能となることを示した。

6.2. 今後の課題

6.2.1. 低電圧動作回路設計

低電力化に関しては，余剰性能を転用してクロック周波数を下げ，低電圧動作を行う方式がもっとも有効であるが，回路の動作マージンは電源電圧に比例かそれ以下に減少する。微細化に伴い半導体プロセスのバラツキが増加する環境で，低電圧で安定，かつ，高速に動作する回路の設計は今後ますます困難になると考えられる。回路的な工夫とアーキテクチャの工夫により，低電圧で動作する設計手法の研究は，今後の大きな課題である。電源電圧や MOS トランジスタのボディー電圧を制御してバラツキを補正したり，特性バラツキに敏感な回路に対しては調整回路を内蔵し，チップの製造後に必要に応じて調整を行ったり，あるいは，冗長回路を内蔵し，不良部分を良品のスペアに交換することなどにより，歩留りを改善する技術の研究が課題である。

6.2.2. 小規模マルチコアプロセッサの開発と実証

この小規模マルチコアプロセッサの研究は，富士通株式会社の社内で 2001 年から 2002 年にかけて行ったものであり，その当時は小規模マルチコアの考え方は発表されていなかったが，2005 年現在では，Sun Microsystems 社が Niagara という開発コードで 8 個の小規模コアを搭載するプロセサを開発しており，同社のサーバ製品に搭載して販売すると言われている。また，Intel 社などの汎用マイクロプロセッサ会社も，消費電力の問題から，2004 年頃から従来のクロック向上一辺倒の方針を変更し，マルチコア化を強力に進めており，2007 年ころ

には 4 コアのプロセッサを商品化する予定であると発表されている。

このように本研究の小規模マルチコアによるスループット性能向上，消費電力低減の考え方は，既に他社で実用化に向かっており，考え方が正しかったことは確認されているが，我々としても小規模コアを複数搭載したプロセッサを開発し，スループット性能や電力削減を実証するのが次の課題である。

6.2.3. チェックポイント-リカバリによる信頼度改善効果の検証

チェックポイント-リカバリ方式により，SPARC64 V プロセッサではロジック部の約 80%のエラーを検出し，そのうちの 85%程度は回復が可能と見積もられるが，エラーの発生頻度が低く，統計的に有意な実測データは得られていない。実測データを集め，効果を証明することが今後の課題の一つである。5.6.4 で述べたように，エラー検出機構がメモリ以外の組み合わせ論理回路のエラーを検出する頻度は 600 年に 1 回程度と考えられるので，多くのプロセッサの動作を長期に渡って監視する必要がある。オンラインのデータ収集が理想的であるが，秘密保持の点からオンラインのモニタを望まない顧客も多く，現状では，顧客の計算機センターで動作しているサーバの動作状況に関する詳細な情報を入手する手段がなく，エラー検出やリカバリデータを精度良く集めることができない。将来方向として，オンラインのモニタリングの充実や，飛行機のフライトレコーダのように重要な機器状態の情報は全て記録し，回収を可能とするなどの手段によるデータ収集が望まれる。

6.2.4. エラー検出，リカバリの改善

5.2 で述べたように制御回路に関してはチェック回路の付加が難しく，SPARC64 V プロセッサではエラー検出を行っていない部分がある。これらの回路がチップ全体に占める割合は約 20%と比較的小さいので，コストを無視して二重化比較のような完全なチェックを行うことは必須ではないと考えているが，

妥当なコストで検出率を上げる努力を継続することが必要である。

SPARC64 V プロセッサでは浮動小数点演算器のエラー検出を行っているが，レジデューチェックのためにかなりの追加ハードウェアを必要としている。多数の浮動小数点演算器を使用するスーパーコンピュータではこのような検出回路の付加は大きな負担であり，少ない追加ハードウェアで高い検出力をもつ浮動小数点演算器のエラー検出方式の研究が課題である。

更に，リカバリに関しては，アーキテクチャレジスタは再実行に当たってクリアされず，これらのレジスタを構成する FF にエラーが発生すると回復できないと言う問題がある。リカバリに当たってクリアされないレジスタは FF 全体の 15%程度であるが，これらのレジスタで発生するエラーに対する対処も今後の課題である。

現状では，エラーの検出率とリカバリの成功率の制約で，信頼度の向上は 3.4 倍に留まっているが，これらの要素の改善により，信頼度の改善率を高めることが望まれる。

謝辞

今回，これらの研究を纏める機会を与えて下さり，また，論文をまとめるにあたり，終始ご懇切なるご指導，ご鞭撻を賜りました東京工業大学 藤原英二教授に心からお礼を申し上げます。

また，本論文を纏めるのにあたり，有益なご教示，ご助言を賜りました東京工業大学 森欣司教授，前島英雄教授，横田治夫教授，松岡聡教授に深く感謝申し上げます。

本研究は，1992 年から HAL Computer Systems 社で HAL R1 プロセッサの開発を行って以来の研究を纏めたものであり，多くの上司に助けられ，また，同僚，部下との共同作業の結晶である。

また，この論文は，富士通研究所のユビクタスセンター長である山澤昌夫博士の強い勧めと後押しが無ければ完成しておらず，深く感謝申し上げます。

また，本研究の始まりとなった HAL 社でのマイクロプロセッサ開発に従事させて下さった，富士通株式会社での当時の上司であった槌本隆光常務（現 富士通テン社常任顧問），小規模マルチコアプロセッサの研究をやらせて下さった当時の上司である富士通株式会社の青木隆執行役には特に感謝申し上げます。

また，HAL 社の同僚，部下では，R1 開発においてアーキテクトと設計マネージャを勤めてくれた Michael Shebanow 博士，アーキテクトの Joel Bony 氏，プロセサ設計者の Niteen Patkar 氏，Gene Shen 氏，丸山拓巳氏等との技術的な議論と検討に負うところが大であり，感謝申し上げます。本研究のチェックポイント-リカバリ方式の特許は，HAL 社におけるオリジナル SPARC64 V プロセッサの開発の過程での Michael Shebanow 博士，Michael Butler 博士，北村俊明博士（現 広島市大教授）等との技術的な検討の成果であり，詳細な実装検討を行って戴き，感謝申し上げます。

富士通での現 SPARC64 V プロセッサの開発は，開発部長である井上愛一郎氏（現 E サーバ開発統括部長代理）とチームの方々によって成し遂げられたものであります。これらの方々の努力と協力に深く感謝申し上げます。特に，論文を纏めるにあたり多くの詳細なデータを収集，提供戴いた山下英男氏に感謝申し上げます。

小規模コアの研究は，富士通研究所の主管研究員 安里彰氏，河場基行氏，大河原英喜氏，浅田善己氏，米国富士通研究所の William Walker 氏，Nestoras Tzartzanis 博士の協力を得て実施したものであり，なかでも，シミュレータの構築やシミュレーションによる性能検討を行って下さった河場氏，大河原氏，低電力回路，カスタムマクロの検討を行って下さった Walker，Tzartzanis 両氏に深く感謝申し上げます。

最後に，この論文の執筆を終始支えてくれた妻啓子に感謝いたします。

参考文献

[1] T.C. May and M.H. Woods, “A new physical mechanism for soft errors in dynamic memories” Proceesings, IEEE International Reliability Physics Symposium (IRPS), pp. 33-40, 1978.

[2] J.F.Ziegler et.al., “IBM experiments in soft fails in computer electronics (1978-1994)”, IBM Journal of Research and Development, Vol 40, No.1, pp. 3-17, Jan. 1996.

[3] R. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," Tech. Dig. International Electron Device Meeitng (IEDM), pp. 329 - 332, Dec. 2002.

[4] Y. Tosaka, H. Ehara, M. Igeta, T. Uemura, H. Oka, N. Matsuoka, and K. Hatanaka, "Comprehensive study of soft errors in advanced CMOS circuits with 90/130 nm technology," Tech. Dig. International Electron Device Meeitng (IEDM), pp. 941 - 944, Dec. 2004.

[5] P.Liden, et.al. "On Latching Probability of Particle Induced Transient in Combinational Networks", International Symposuim on Fault Tolerant Computing FTCS24, pp340-349 , June 1994.

[6] http://www.spec.org/benchmarks.html

[7] http://www.itrs.net/Common/2004Update/2004Update.htm

[8] E.J.Nowak,”Maintaining the benefits of CMOS scaling when scaling bogs down”, IBM Journal of Research and Development, VOL. 46, NO. 2/3, pp.169-180, Mar./May 2002.

[9] 電子計算機概論 池田敏雄編 オーム社, 1968.

[10] R.W.Hamming , “Error Detecting and Error Correcting Codes”, Bell System Technology Journal 29, pp. 147-160, 1950.

[11] I.S.Reed and G.Solomon, “Polynomial Codes over Certain Finite Fields”, Journal of Society for Industrial Applied Mathematics 8, pp.300-304, 1960.

[12] S.Kaneda and E.Fujiwara, "Single Byte Error Correcting-Double Byte Error Detecting Codes for Memory Systems", IEEE Transaction on Computers, Vol. C-31, No.7, pp. 737-739, July 1982.

[13] T. Karnik, S. Vangal, V. Veeramachaneni, P. Hazucha, V. Erraguntla, and S. Borkar, “Selective node engineering for chip-level soft error rate improvement,” IEEE Symposium on VLSI Circuits, pp. 204-205, June 2002.

[14] P.Hazucha et.al., “Measurements and analysis of SER tolerant latch in a 90 nm dual-Vt CMOS process”, IEEE Journal of Solid-State Circuits, Volume 39, Issue 9, pp.1536-1543, Sept. 2004.

[15] Y. Arima, T. Yamashita, Y. Komatsu, T. Fujimoto, K. Ishibashi,; “Cosmic-ray immune latch circuit for 90nm technology and beyond”, Digest of Technical Papers, IEEE International Solid-State Circuits Conference (ISSCC 2004), pp. 492-493, Feb. 2004.

[16] D.B.Hunt and P.N.Marinos, “A General Purpose Cache-Aided Rollback Error Recovery (CARER) Technique”, Digest of Papers,17th Symposium on Fault Tolerant Computing (FTCS-17), pp.170-175, 1987.

[17] R.E.Ahmed, R.C.Frazier and P.N.Marinos, “Cache-Aided Rollback Recovery (CARER) Algorithms for Shared-Memory Multiprocessor Systems”, Digest of Papers, 20th International Symposium Fault-Tolerant Computing (FTCS-20), pp. 82-88, June 1990.

[18] N.S.Bowen, D.K.Pradhan, "Processor- and Memory-Based Checkpoint and Rollback Recovery", IEEE Computer, Vol.26 No.2, pp. 22-31, Feb. 1993.

[19] C. F. Webb and J. S. Liptay, “A high-frequency custom CMOS S/390 microprocessor”, IBM Journal of Research and Developemnt, Vol. 41, No. 4/5, pp463-474 , July/Sept. 1997.

[20] T. J. Slegel,E. Pfeffer and A. Magee, “The IBM eServer z990 microprocessor”, IBM Journal of Research and Developemnt, Vol. 48, No. 3/4, pp. 295-310, May/July 2004.

[21] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, “Low-Power CMOS Digital Design”, IEEE Journal of Solid State Circuits, Vol. 27, No.4, pp. 473-484, Apr. 1992

[22] M.Takahashi et. Al., “A 60 mW MPEG4 video codec using clustered voltage scaling with variable supply-voltage scheme”, Digest of Technical Papers, 45th IEEE International Solid-State Circuits Conference (ISSCC 1998), pp. 36-37, Feb. 1998.

[23] T.Kuroda et al., “A 0.9V 150MHz 10mW 4mm2 2-D discrete cosine transform core processor with variable-threshold-voltage scheme”, IEEE Journal of Solid State Circuits, Vol. 31, No. 11, pp. 1770-1779, Nov. 1996.

[24] S.Naffziger, B.Blaine, T.Grutkowski, “The Implementation of a 2-core Multi-Threaded Itanium-Family Processor”, Digest of Technical Papers, IEEE International Solid-State Circuits Conference (ISSCC 2005), pp.182-183, Feb. 2005.

[25] Q.Wu,; M.Pedram,; X.Wu,; “Clock-gating and its application to low power design of sequential circuits”, Proceedings of the IEEE Custom Integrated Circuits Conference, 1997, pp.479-482, May 1997.

[26] H. Ando, T.Kitamura, M.Shebanow, M.Butler, US Patent 6,519,730 Computer and error recovery method for the same; 出願 March 16, 2000, 登録 February 11, 2003

[27] G. Shen, N. Patkar, H. Ando, D. Chang, C. Chen, C. Chen, F. Chen, P. Forssell, J. Gmuender, T. Kitahara, H. Li, D. Lyon, R. Montoye, L. Peng, S. Savkar, J. Sherred, M. Simone, R. Swami, D. Tovey, and T. Williams, "A 64b 4-issue out-of-order execution RISC processor," Digest of Technical Papers, IEEE International Solid-State Circuits Conference (ISSCC 1995), pp. 170-171, Feb. 1995.

[28] T. Williams, N. Patkar, and G. Shen, "SPARC64: A 64-b 64-active-institution out-of-order-execution MCM processor," IEEE Journal of Solid State Circuits, Vol. 30, Issue 11, pp. 1215-1226, Nov. 1995.

[29] B.Nayfeh, L.Hammond, K.Olukotun, “Evaluation of Design Alternatives for a Multiprocessor Microrocessor”, Proceedings, The 23rd Annual International Symposium on Computer Architecture, pp. 67-77, May, 1996.

[30] L.Hammond et.al., “A Single-Chip Multiprocessor”, IEEE Computer , Vol. 30 , Issue 9 , pp. 79 - 85, Sept. 1997.

[31] L.Hammond et. al., “The Stanford Hydra CMP”, IEEE Micro,Vol. 20 , Issue 2 , pp. 71-84, March-April 2000.

[32] L.Codrescu, D.Wills, “Architecture of the Atlas Chip-Multiprocessor: Dynamically Parallelizing Irregular Applications”, IEEE Transactions on Computers, Vol. 50 , Issue 1 , pp. 67-82, Jan. 2001.

[33] J.Tendler et. al., “POWER4 system microarchitecture”, IBM Journal of Research and Development, Vol.46, No.1, pp. 5-25, Jan. 2002.

[34] A.Alameldeen et.al., “Evaluating Non-deterministic Multi-threaded Commercial Workloads”, Proceedings, Fifth Workshop on Computer Architecture Evaluation using Commercial Workloads, 2002. HPCA-8, Feb. 2002.

[35] L. Barrosso et. al., “Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing”, Proceedings of the 27th International Symposium on Computer Architecture, 2000, pp. 282-293, June 2000.

[36] L.Cordrescu et.al., “Exploring Microprocessor Architectures for Gigascasle Integration”, Proceedings, 20th Anniversary Conference on Advanced research in VLSI, 1999, pp. 242-245, Mar 1999.

[37] A.Inoue, “SPARC64 V Processor for UNIX Servers,” FUJITSU, Vol.53, No.6, pp.450-455, Nov. 2002.

[38] A.Inoue, “Fujitsu’s New SPARC64 V for Mission-Critical Servers”, Microprocessor Forum 2002, Oct, 2002.

[39] H.Ando et.al “A 1.3GHz Fifth Generation SPARC64 Microprocessor”, Digest of Technical Papers, IEEE International Solid-State Circuits Conference (ISSCC 2003), pp. 246-247, Feb. 2003.

[40] H.Ando et.al., “A 1.3GHz Fifth Generation SPARC64 Microprocessor”, IEEE Journal of Solid State Circuits,Vol. 38, Issue 11, pp. 1896-1905, Nov. 2003.

[41] F.Pollack, “New Microarchitecture Challenges in the Coming Generations of CMOS”, Keynote, 32nd Annual International Symposium on Microarchitecture (MICRO-32), Nov. 1999.

[42] M.Horowitz, W.Dally, “How Scaling Will Change Processor Architecture”, Digest of Technical Papers, IEEE International Solid-State Circuits Conference (ISSCC 2004), pp. 132-133, Feb. 2004.

[43] J. Clabes et.al., "Design and implementation of the POWER5™ microprocessor," Digest of Technical Papers, IEEE International Solid-State Circuits Conference (ISSCC 2004), pp. 56-57, Feb. 2004.

[44] S.Nakai et.al, “A 100nm CMOS Technology with “Sidewall-Notched” 40nm Transistors and SiC-Capped Cu/VLK Interconnects for High Performance Microporcessor Applications”, Digest of Technical Papers, Symposium on VLSI Technology, pp. 66-67, June 2002

[45] L.Wei et.al., “Design and Optimization of Dual-Threshold Circuits for Low-Voltage Low-Power Applications”, IEEE Transaction on VLSI Systems, Vol.7, Issue 1, pp. 16-24, Mar. 1999.

[46] J. Cao and A. Chandrakasan, "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits", IEEE Journal of Solid State Circuits, Vol. 35, Issue 7, pp. 1009-1018, July 2000.

[47] I.Sutherland, R.Sproull, D.Harris, “Logical Effort”, Morgan Kaufmann, 1999.

[48] D.Dobberpuhl, "The Design of a High-Performance Low Power Microprocessor", Processdings, International Symposium on Low Power Electronics and Design, pp. 11-16, Aug. 1996.

[49] P.Hazucha et.al., “Neutron soft error rate measurements in a 90-nm CMOS process and scaling trends in SRAM from 0.25-/spl mu/m to 90-nm generation”, I Technical Digest, EEE International Electron Devices Meeting, 2003. (IEDM '03), 21.5, Dec. 2003.

[50] H.Fukui et.al., “Comprihensive Study on Layout Dependence of Soft Error in CMOS Latch Circuits and Its Scaling Trend for 65nm Technology node and Beyond”, Digest of Technical Papers, Symposium on VLSI Technology , pp.222-223, June 2005,

[51] Y.Kawakami et.al., “Investigation of soft error rate including multi-bit upsets in advanced SRAM using neutron irradiation test and 3D mixed-mode device simulation”, Technical Digest, IEEE International Electron Devices Meeting, 2004. (IEDM’04), pp. 945-948, Dec. 2004.

[52] C. Asato, "A 14-port 3.8-ns 116-word 64-b read-renaming register file," IEEE Journal of Solid State Circuits, Vol. 30, Issue 11, pp. 1254-1258, Nov. 1995.

[53] M.Sakamoto et.al., “Microarchitecture and Performance Analysis of a SPARC-V9 Microprocessor for Enterprise Server Systems”, Proceedings, The Ninth International Symposium on High-Performance Computer Architecture, 2003. (HPCA-9), pp. 141- 152, Feb. 2003.

[54] R.Cmlick and D.Keppel, “Shade: A Fast Instruction-set Simulator for Execution Profiling”, Sun Labs technical reports, TR-93-12, May 1993.

[55] TPC-C Benchmark; http://www.tpc.org/tpcc/

[56] L.Barroso, et.al., “Memory System Characterization of Commercial Workloads”, Proceedings, The 25th Annual International Symposium on Computer Architecture, pp. 3-14, June 1998.

[57] A.K.Nanda, “Multiprocessor Architecture Evaluation using Commercial Applications”, Proceedings, First Workshop on Computer Architecture Evaluation using Commercial Workloads, 1998. HPCA-4 , Feb. 1998.

[58] P.Ranganathan et.al., “Performance of Database Workloads on Shared-Memory Systems with Out-of-Order Processors”, Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII), pp. 307-318, Oct. 1998.

[59] H.Jason et.al., “Implementation of a 4th-Generation 1.8GHz Dual-Core SPARC V9 Microprocessor”, Digest of Technical Papers, IEEE International Solid-State Circuits Conference (ISSCC 2005), pp. 186-187, Feb. 2005.

[60] A.Inoue, “SPARC64™ V/VI for Mission-Critical Servers”, Microprocessor Forum 2004, Oct, 2004.

[61] S.Naffziger, B.Stackhouse, T.Grutkowski, “The Implementation of a 2-core Multi-Threaded Itanium®-Family Processor”, Digest of Technical Papers, IEEE International Solid-State Circuits Conference (ISSCC 2005), pp. 182-183, Feb. 2005.

[62] H.Ando et al., “A Case Study: Energy Efficient High Throughput Chip Multi-Processor using Reduced-complexity Cores for Transaction Processing Workload”, IPSJ Transaction on Advanced Computer Systems, pp. 103-114 , May 2005,

[63] H.Ando, N.Tzartzanis and W.Walker, “A Case Study: Power and Performance Improvement of a Chip Multi-Processor for Transaction Processing”, IEEE Transaction on VLSI Systems, pp865-868, July 2005.

[64] D.Anderson, F.Sparacio, M.Tomasulo,;”The IBM System/360 Model 91: Machine Philosophy and Instruction Handling”, IBM Journal of Research and Development, Vol.11 No.1, pp. 8-24, Jan. 1967.

[65] C.Chen, et.al., “Fault Tolerance design of the IBM enterprise System/9000 Type 9021 Processors”, IBM Journal of Research and Development , Vol.36 No.4, pp. 765-779, July 1992.

[66] M.Shebanow, “SPARC64™-V, A High Performance System Processor”, Microprocessor Froum, Oct. 1999.

[67] Aiichiro Inoue, US patent 6,502,186 Instruction processing apparatus; 出願Dec.20,2000, 優先権10-191900 Jul.7,1998 (JP)Japan, 登録 Dec.31,2002

[68] Aiichiro Inoue, US patent 6,851,043 Branch instruction execution control apparatus; 出願 Dec.15,1999, 優先権 10-358771 Dec.17,1998 (JP)Japan, 登録 Feb.1,2005

[69] P.Congetira, "A 32-way Multithreaded SPARC Processor", Proceedings, Symposium on High Performance Chips (HOT CHIPS 16), Aug. 2004.

著者による発表論文リスト

査読付ジャーナル論文

1. Ando, H.; Yoshida, Y.; Inoue, A.; Sugiyama, I.; Asakawa, T.; Morita, K.; Muta, T.; Motokurumada, T.; Okada, S.; Yamashita, H.; Satsukawa, Y.; Konmoto, A.; Yamashita, R.; Sugiyama, H.; "A 1.3-GHz fifth-generation SPARC64 microprocessor", IEEE Journal of Solid-State Circuits, ,Volume: 38 , Issue: 11 , pp. 1896-1905, Nov. 2003.

2. Ando, H.; Asato, A.; Kawaba. M.; Okawara. M.; Walker. W.; "A Case Study: Energy Efficient High Throughput Chip Multi-Processor using Reduced-complexity Cores for Transaction Processing Workload", IPSJ Transcation on Advanced Computing Systems, Vol.46, No. SIG7 ACS10, pp. 103-114, May 2005.

3. Ando. H.; Tzartzanis. N.; Walker. W.; "A Case Study: Power and Performance Improvement of a Chip Multi-Processor for Transaction Processing", IEEE Transaction on VLSI Systems, Vol. 13, Isuue 7, pp. 865-868, July 2005.

国際会議論文

4. Ando, H., “Testing VLSI with Random Access Scan”, Digest of Papers, COMPCON spring 80 20th IEEE Computer Society International Conference, pp 50-52, Feb. 1980,

   F. Tsui 著：LSI/VLSI Testability Design (McGraw-Hill Feb. 1987) 第 5 章で引用

   N.Weste, G.Eshraghian 共著：Principles of CMOS VLSI Design (Addison Wesley Oct.1994) 第 7 章で引用

5. G. Shen, N. Patkar, H. Ando, D. Chang, C. Chen, C. Chen, F. Chen, P. Forssell, J. Gmuender, T. Kitahara, H. Li, D. Lyon, R. Montoye, L. Peng, S. Savkar, J. Sherred, M. Simone, R. Swami, D. Tovey, and T. Williams, "A 64b 4-issue out-of-order execution RISC processor," Digest of Technical Papers, IEEE International Solid-State Circuits Conference, pp. 170 - 171, February 1995

6. Ando, H.; Yoshida, Y.; Inoue, A.; Sugiyama, I.; Asakawa, T.; Morita, K.; Muta, T.; Motokurumada, T.; Okada, S.; Yamashita, H.; Satsukawa, Y.; Konmoto, A.; Yamashita, R.; Sugiyama, H.; “A 1.3-GHz fifth-generation SPARC64 microprocessor”, Digest of Technical Papers, 50th IEEE International Symposium on Solid-State Circuits (ISSCC 2003) , pp. 246 - 247, 491, Feb. 2003.

7. Ando, H.; Yoshida, Y.; Inoue, A.; Sugiyama, I.; Asakawa, T.; Morita, K.; Muta, T.; Motokurumada, T.; Okada, S.; Yamashita, H.; Satsukawa, Y.; Konmoto, A.; Yamashita, R.; Sugiyama, H.; “A 1.3-GHz fifth-generation SPARC64 microprocessor”, 40th Design Automation Conference

2003, Paper 41.2, Highlight of ISSCC Papers, June. 2003.

国際会議パネルディスカッション（パネラー）

8. VLSI Symposia 2003, 11-14 June 2003, Technology and Circuits Joint Rump Session：Judgment Day for Power Management
9. VLSI Symposia 2004, 15-19 June 2004, Circuits Rump Session: Limitations of Low F04 Design
10. VLSI Symposia 2005, 15-19 June 2005, Technology and Circuits Rump Session: Variability has stopped scaling
11. 51st ISSCC 2004, 16-18 Feb. 2004, Evening Discussion Session, E3 Processors and Performance: When do GHz Hurt?

研究会等（口頭発表）

12. Ando, H., “A SPARC Microprocessor for High-End Servers”, Microprocessor Forum 1997, 14-15 Oct. 1997
13. 安藤 壽茂, 吉田 裕司, 井上 愛一郎 他, ”高信頼設計 SPARC64 V マイクロプロセサ”, 電子情報通信学会技術研究報告 DSP2003, 2003 年 10 月, Pages: 35－40
14. 52nd ISSCC 2005, 7-10 Feb. 2005, Circuit design workshop presenter, Robust Design Solutions for Nano-Scale Circuits, “Tolerant Architecture”

米国特許

15. T.Takezono H.Ando, US Patent 4,064,483, “Error correcting circuit arrangement using cube circuits”, December 20, 1977

16. H.Ando, US Patent 4,091,293, “Majority decision logic circuit”, May 23, 1978

17. H.Ando, US Patent 4,150,441, “Clocked static memory”, April 17, 1979

18. H.Ando, US Patent 4,162,540, “Clocked memory with delay establisher by drive transistor design”, July 24, 1979

19. H.Ando et.al., US Patent 4,888,584, “Vector pattern processing circuit for bit map display system”, December 19, 1989

20. H.Ando et.al., US Patent 4,933,879, “Multi-plane video RAM”, June 12, 1990

21. H.Ando et.al., US Patent 4,969,029, “Cellular integrated circuit and hierarchial method”, November 6, 1990

22. H.Ando et.al., US Patent 5,095,356, “Cellular integrated circuit and hierarchical method”, March 10, 1992

23. H. Ando, T.Kitamura, M.Shebanow, M.Butler, US Patent 6,519,730, “Computer and error recovery method for the same”, February 11, 2003

国内特許

24. 安藤 寿茂　他2名,【特許番号】第２７３７８９８号, ”ベクトル描画装置”, 平成10年1月16日