

# UNIXサーバ用プロセッサ 「SPARC64 V」

Revision 1.0

August 2004

富士通株式会社

Copyright © 2004 FUJITSU LIMITED. All Rights Reserved.

## 商標について

Sun、Sun Microsystems、Sun ロゴ、Solaris およびすべての Solaris に関連する商標及びロゴは、米国およびその他の国における米国 Sun Microsystems, Inc.の商標または登録商標であり、同社のライセンスを受けて使用しています。

全ての SPARC 商標は、米国 SPARC International, Inc.のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。

SPARC64 は、米国 SPARC International, Inc.のライセンスを受けて使用している同社の商標です。

UNIX は、米国およびその他の国におけるオープン・グループの登録商標です。

その他各種製品名は、各社の製品名称、商標または登録商標です。

本書で説明されている情報は、富士通株式会社が特許を所有または申請している可能性があります。

本書で説明されている情報は、これらの特許に関する実施権を許諾するものではありません。

本書で説明されている情報は、予告無く変更される場合があります。

## 目次

まえがき	4
第1章 開発のねらい	4
第2章 SPARC64 V のマイクロアーキテクチャ	4
2.1 命令フェッチ部	5
2.2 命令実行部	6
2.3 キャッシュ	11
2.4 システムインタフェース	13
2.5 T S O (Total Store Order) の保証	13
2.6 MMU (Memory Management Unit)	13
2.7 プリフェッチ機構	14
第3章 SPARC64 V の RAS	15
3.1 内蔵 RAM の RAS	15
3.2 メモリの RAS	17
3.3 内蔵レジスタと演算機の RAS	18
3.4 同期一括更新方式と命令リトライ	18
3.5 エラー処理のソフトウェア・インタフェース	19
3.6 サービス性の向上	20
第4章 SPARC64 V の半導体テクノロジー	21
4.1 130nm プロセスの SPARC64 V	21
4.2 90nm プロセスの SPARC64 V	22
第5章 SPARC64 V の性能評価	22
むすび	23
参考文献	23

## まえがき

エンタープライズ・サーバ・システムは、社会経済機構の基盤として重責を担っている。この心臓部であるプロセッサは、高性能であるとともに、高信頼を達成していなければならない。SPARC64™ V は、これまで達成し得なかった性能と高度な信頼性を同時に実現するために、長年に渡るメインフレームの開発で蓄積した技術に基づいて、第五世代の SPARC64 プロセッサとして開発した。

富士通のメインフレーム GS8900 のプロセッサをベースとしたため、ハードの論理設計にあたって、はじめにメインフレーム・プロセッサと SPARC・プロセッサの違いを吸収する必要があった。命令セットアーキテクチャが全く違うために、命令のデコードが新規設計である事は当然であるが、プログラマブルなリソース、例えばアーキテクチャレジスタや、メモリモデルが違う事から、メインフレームには無かった多くの機構を新規設計した。さらに、RISC と CISC の違いや、ワークロードの特徴、OS コードの影響等によって、性能を支配する演算リソースの配置を新規設計する必要があった。全てが新規設計であるため、性能モデルを開発し、性能面での設計妥当性を検証しながら、並行してハードウェアの開発を行った。

## 第 1 章 開発のねらい

SPARC64 V は、富士通の UNIX®サーバである PRIMEPOWER に搭載することを前提として開発した。従って、PRIMEPOWER のエンタープライズ・サーバ・システムとしての魅力を最大にする事が開発のねらいであり、以下の方針で開発した。

### 1) SPARC V9 完全互換設計：

SPARC V9 に準拠し、アプリケーションのバイナリ互換性を維持する。オペレーティングシステムのレベルに至るさらに高度な互換性を追求するため、Sun と富士通で共同開発された共通仕様 JPS(Joint Programming Specification)に適合する。

### 2) システム搭載時性能に着目した高性能設計：

消費電力、チップサイズ、システム接続バス方式などのシステムの実装要件を満足しつつ、サーバ・システムとしてのポテンシャルを最大限に引き出す。実稼動時、特にインタラクティブな業務に着目して、メモリアクセスの負荷と演算パワーのバランスをとり、高いスループットを実現する。キャッシュ階層を単純化して、マルチプロセッサのスケラビリティを確保する。HPC 分野でも優位性を確保するために、演算ユニットを多重化し、それにバランスする演算資源をおく。

### 3) 高マシンサイクル設計：

マシンサイクルとサイクルあたりの処理量とのバランスの最適解を追求する。そして、メインフレームの開発で培った緻密で確実な設計手法を適用して、高機能なプロセッサとして最高水準のマシンサイクルを実現する。

### 4) 最先端の RAS(Reliability, Availability, and Serviceability)機能の実現：

ミッションクリティカルな使用環境に対応するため、最先端の RAS 機能を装備する。データ・インテグリティの保証を徹底し、訂正や再試行による回復処理を適用して、アプリケーションへの影響を抑止するとともにシステム動作の継続を図る。特に、エラーの発生頻度が他に比べて高くなる RAM の 1 ビットエラーは全て救済し、性能以外にはエラーの影響は一切発生しないことを目指す。エラー情報は専用ハードウェアで収集し、障害解析の時間の短縮と精度の改善を図る。

## 第 2 章 SPARC64 V のマイクロアーキテクチャ

SPARC64 V は、アウト・オブ・オーダー実行機構を備えたスーパースカラ・プロセッサ

である。SPARC64 V は、コア部、2次キャッシュ(L2\$)部、システムインタフェース部で構成される。コア部は更に、命令フェッチ部と命令実行部に分けられ、命令フェッチ部には命令専用の1次キャッシュ(L1I\$)を含み、実行部にはオペランド専用の1次キャッシュ(L1D\$)を含む。以下で、はじめに命令フェッチ部を、それから、命令実行部を説明し、最後にキャッシュについて、1次キャッシュと2次キャッシュをまとめて説明するとともに、システムインタフェース、その他について説明する。

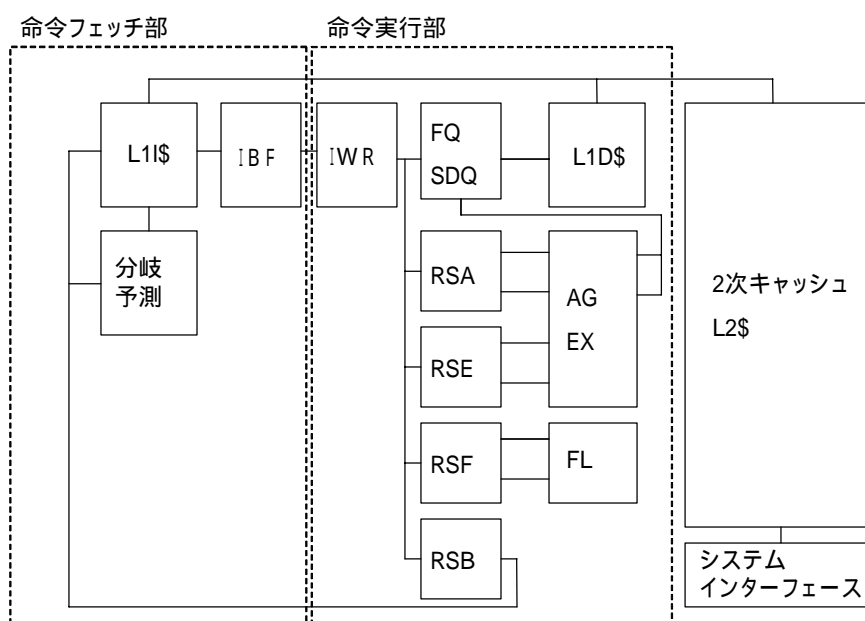


図1 SPARC64 Vの構成

## 2.1 命令フェッチ部

命令フェッチ部は、命令実行部とは独立して動作し、分岐予測に従って将来実行すると予測される命令列をなぞるように命令バッファ IBF(Instruction BuFfer)に取り込む。IBF は 192 バイトの容量があり、命令実行が滞った場合にも、IBF が一杯になるまで命令フェッチは停止しない。逆に、キャッシュ・ミス等で命令フェッチが停止した場合にも、IBF に既に命令が蓄えられていれば、そこから命令を取り出して実行を継続できる。命令フェッチは毎サイクル起動でき、一度に 32 バイトすなわち 8 命令を取り出す。命令実行のスループットはサイクルあたり最大 4 命令であるから、2 倍のスループットを確保して、IBF はデカップリング効果を発揮する。これにより、1 次キャッシュとしては大容量の命令キャッシュを比較的長いパイプラインでアクセスする損失を抑えて、大容量キャッシュの利益を享受する。

分岐予測の精度はプロセッサの性能を決めるひとつの鍵であるが、SPARC64 V は大規模トランザクション処理などの基幹業務に対応するために、大容量(16K エントリ、4 ウェイ・セットアソシアティブ)のブランチヒストリを装備する。ブランチヒストリには、強分岐(Strong-Taken)、弱分岐(Weak-Taken)、無効(Invalid) の状態を、分岐先アドレスとともに記録する。命令フェッチ時にブランチヒストリを参照し、有効なエントリが見つかった場合には、分岐成立と予測するようになっている。

高頻度で実行される複雑な挙動を示すルーチンの分岐予測精度向上のために、一種のグローバル履歴テーブルである WGHT (Write-cycle-driven Global History Table)を備える。WGHT はブランチヒストリの本体と連携して使用する。分岐の処理確定時に、ブランチヒ

ストリの更新に先だって WGHT を検索し、この結果に応じてブランチヒストリのエントリの状態を更新する。WGHT に履歴がないか、記録されている履歴が十分な深さではない場合には、実行完了した分岐命令の結果だけでブランチヒストリを更新する。このため、一般的なグローバルヒストリ予測機構に見られる履歴不足時の誤った予測動作は、WGHT では原理的に発生しない。

さらに SPARC64 V は、サブルーチンリタンの戻り先アドレスを予測するために、リターンアドレススタックを装備する。リターンアドレススタックは、LIFO(Last-in Fast-out)スタック構造をベースとした 4 エントリに加え、投機フェッチによる特別なもう 1 エントリの、合計 5 エントリで構成される。リターンアドレススタックへの登録は、分岐命令の完了時にブランチヒストリの更新と一緒に行われる。投機フェッチによる実行部との同期ずれをポインタで補正するなどの細かい制御を行って、精度の高いリターンアドレス予測を行う。

一般的なアプリケーションで消費される時間の多くを占める局所的なルーチンにも対処する。すなわち局所的なループ命令列であることを認識した場合、IBF を構成するレジスタ間でチェーンを張りあって、新たなフェッチを行わずに命令供給を行うことが出来る。

以上説明してきたように、命令フェッチは大容量キャッシュと大容量ブランチヒストリを備え、これを補助するさまざまな工夫によって、一般的なアプリケーションから大規模トランザクション処理に至るまで、どのような場合にも対応して、常に大きなスループットで命令実行部に命令を供給することが出来る。

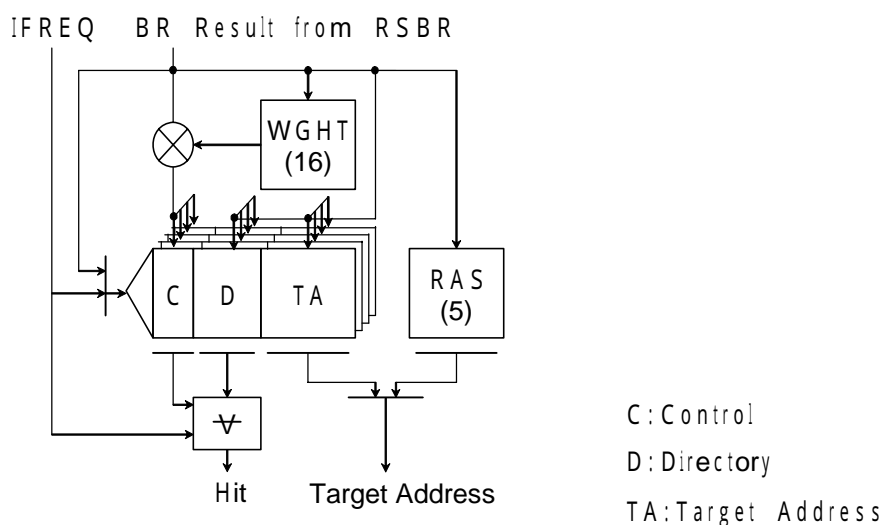


図2 SPARC64 Vの分岐予測

## 2.2 命令実行部

SPARC64 V は、それぞれ独立して同時に実行が出来る 6 つの演算器を備え、演算のオペランドが用意できたものからプライオリティを取って演算器に投入する、いわゆるデータドリブンのアウト・オブ・オーダー実行制御方式である。ここでは、命令フェッチが完了して IBF に蓄えられた命令が実行される手順を概観する。

### 1) 命令プリゼンテーションステージ：

命令フェッチ部が IBF にプリフェッチして蓄えた命令から、プログラムの実行順序で命

令を取り出すのが、命令プリゼンテーションステージである。命令プリゼンテーションステージでは、4命令分をまとめてIWR(Instruction Word Register)にセットする。連続した命令列の供給ばかりではなく、分岐命令の実行に関連した命令ストリームの切り替えやディレックスロットの命令供給は、命令プリゼンテーションステージの制御で行い、命令フェッチ部と命令実行部の独立性を高めて、それぞれ最適な動作を可能にする。

#### 2) 命令デコード / 発行ステージ :

命令デコード / 発行ステージでは、IWR 上の4つの命令を同時にデコードして、実行に必要な資源(各種リザーベーション・ステーション、フェッチ・キュー、ストアデータ・キュー、レジスタ更新バッファ)を決定する。そして、その資源に空きがあるかどうかをチェックして、空きがあれば、その資源を割り当てるとともに、0~63の範囲の命令識別子(IID、Instruction IDentification)を割り当てて命令を発行する。命令を発行するとそのIWRを開放する。

IWRのどのスロットにある命令であっても、リザーベーション・ステーション等の資源の割り当て制限はなく、また、命令種類の組み合わせの制限もない。このため、資源の空きさえあれば命令が発行できる。4命令分の空きが無い場合にも、プログラム順序の命令から可能な分だけを発行できる。このように命令発行の停滞を極力排除した方式により、どのようなバイナリコードであっても常に高い並列度を確保する。

#### 3) 演算投入ステージ :

SPARC64 Vは、整数演算用リザーベーション・ステーション RSE(Reservation Station for Execution)と浮動小数点演算用リザーベーション・ステーション RSF(Reservation Station for Floating point)を装備する。RSEとRSFは、演算器に対応して、それぞれ2つのキューに分かれている。すなわち演算用のリザーベーション・ステーションは、RSEA、RSEB、RSFA、RSFBの4種を備える。リザーベーション・ステーションでキューイングされたリクエストは、演算のオペランドが用意できたものから、個々のリザーベーション・ステーション毎にプライオリティを取って対応する演算器に投入する。したがって4つの演算を同時に投入できる。投入アルゴリズムは、基本的にはリザーベーション・ステーションでキューイングされたリクエストのうち、投入可能で1番古い(Oldest Ready)ものを選択する方法を採用している。しかしロード命令が更新するレジスタをオペランドとして演算するような場合には、ロード命令の結果が得られる以前に投機的に投入し、演算ステージで投機的な実行が良かったか否かを判断する。これを投機的投入(Speculative dispatch)と呼ぶ。投機的投入はキャッシュアクセスのパイプラインのレイテンシを隠蔽して演算器の使用効率を高める。RSEA、RSEB、RSFA、RSFBは、それぞれ8エン트리であるから、整数系、浮動小数点系の演算をそれぞれ最大16個ずつキューイングできる。

#### 4) レジスタ読み出しステージ :

レジスタの読み出しポートは、全ての演算器が並列動作出来る様に設けている。即ち、整数系レジスタ GPR(General Purpose Register)は、整数演算器のために2リードポート×2とアドレス生成用に2リードポート×2の合計8リードポートを持っている。また浮動小数点系レジスタ FPR(Floating-Point Register)は、FMA演算に対応して演算器あたり3リード×2で合計6リードポートを持っている。

GPRは8ウィンドウレジスタを実装する。ウィンドウ数が大きいことは、サブルーチン呼び出しなどレジスタ入れ替えに要するソフトウェアのオーバーヘッドを小さくする効果がある。その反面、レジスタから高速にデータを読み出すことが困難である。この矛盾に対応するため、SPARC64 Vは、カレントウィンドウを含むGPRの一部のコピーをワークレジスタとして用意する。このGPRのコピーはJWR(Joint Work Register)と呼ばれ、カレントウィンドウにその前後のウィンドウを結合した、合計3ウィンドウ分(64x8バイト)を保持する。演算器へのレジスタ読み出しは、JWRから行き、命令コミット時のレジスタ書き込みは、JWRとGPR本体に同時に行う。ウィンドウ切り替え時には、ハードウェアがバックグラウンドでJWRに別のウィンドウを用意し、切り替え前後の命令がストールす

ることではない。これによりサブルーチン呼び出しの際にも最大の実行並列度を維持する。

5) 実行ステージ：

SPARC64 V は SPARC V9 命令に加え、FMADD(Floating-point Multiply & ADD)をサポートする。

整数系の演算器には EXA、EXB がある。加減論理演算、シフト演算は EXA、EXB いずれの演算器でも実行できる。乗除算は EXA のみで実行される。

浮動小数点系の演算器には FLA、FLB がある。加算、減算、乗算、除算、平方根演算は FLA、FLB いずれの演算器でも実行可能である。除算、平方根演算を除く各演算はパイプライン方式で実行する。さらに科学技術計算等で多用される行列積計算を高速処理するために、FLA と FLB の両方で FMADD 命令をサポートする。FMADD は 3 つのソースオペランドを同時に読み出して、乗算器でその中の 2 ソースオペランドの乗算を実行し、その結果を 3 目目のソースオペランドとともに加算器に転送して加算をする。この一連の処理もパイプライン実行なので、乗算、加算を毎サイクル同時に 4 演算処理する。

このように SPARC64 V は高い演算スループットを確保しているが、さらに依存関係のある命令間ではデータフォワーディングを行う。これによりオペランドの依存関係のある命令を連続して実行する事が可能である。

さらに、FLA にはグラフィック演算器も備えている。

演算結果は命令コミット時までレジスタ更新バッファに保持する。GPR に対して GUB(GPR Update Buffer)を、FPR に対して FUB(FPR Update Buffer)を、それぞれ 32 本ずつ用意している。これらのリソースは命令デコード・発行ステージで割り当てられ、命令コミット時に解放される。

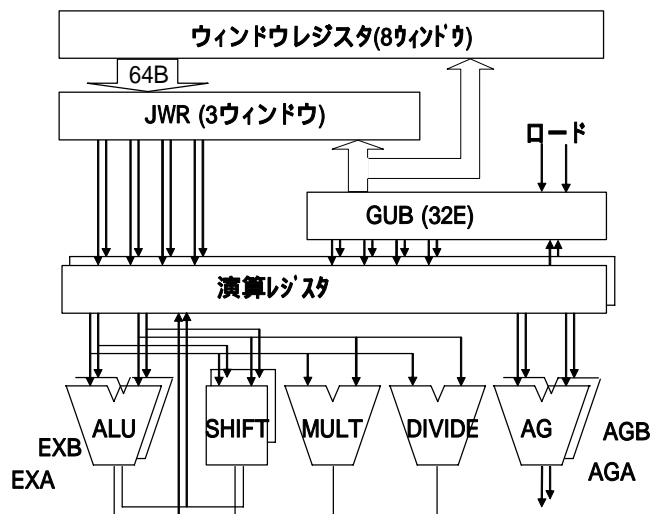


図3 SPARC64 V の整数演算とアドレス生成

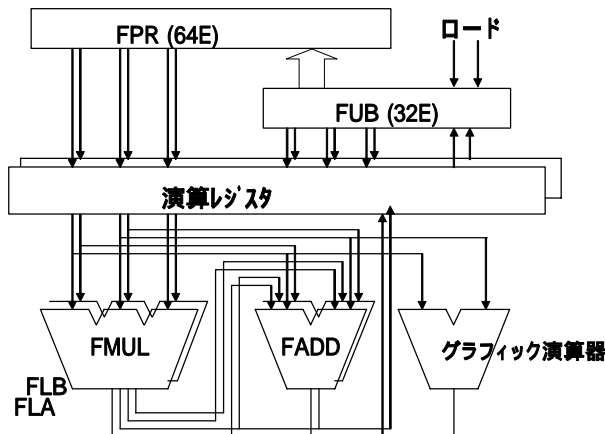


図4 SPARC64 V の浮動小数点演算

6) コミットステージ :

命令発行の際に割り当てられた IID に対応して 64 個のコミット・スタック・エントリ CSE(Commit Stack Entry)がある。実行結果をプログラマブルな資源に反映する制御は CSE が行う。命令発行から実行完了までの実行途中、すなわちインフライトの命令数は、この CSE の数によって制約され最大 64 命令である。

アウト・オブ・オーダーで実行した結果は、すべてソフトウェアから見えない作業用レジスタである GUB、FUB に一旦格納する。プログラムの順序性を保証するために、GPR や FPR などのレジスタやメモリは、コミットステージで命令実行順序に従って(イン・オーダー)更新する。さらに、PC をはじめとする制御レジスタも、コミットステージで一括して更新する。これにより、正確な割り込みを保証し、いつでも実行中の処理のキャンセルが可能である。これを同期一括更新方式と呼び、分岐の予測ミスに関わるやり直しを簡単にするばかりではなくて、後の章で説明するように、RAS の向上に寄与する。同時にコミットできる命令数は最大 4 である。

以上で説明できなかった分岐命令とロード・ストア命令の実行について以下で説明する。

7) 分岐命令実行 :

分岐命令には命令相対分岐とレジスタ間接分岐があるが、分岐命令の大多数を占める命令相対分岐は、命令デコードと同時に分岐アドレスを計算する。レジスタ間接分岐は命令発行後に EXA で分岐アドレスを計算する。いずれの場合にも、分岐命令では分岐用リザーベーション・ステーション RSBR(Reservation Station for BRanch)に命令を発行する。RSBR は 10 エントリで、分岐の確定を待つ役割と、分岐予測機構への情報のローディングと、予測失敗の場合の再命令フェッチの制御の役割がある。

RSBR のエントリには、分岐の種類を識別する情報とともに、条件分岐の場合には、その確定に関わる条件コードを生成する命令の IID を命令発行の際にセットする。この IID を手がかりにして、演算動作を RSBR 上で監視し、アウト・オブ・オーダーで分岐確定する。分岐の確定と分岐先アドレスの準備が出来ると、ブランチヒストリをはじめとする分岐予測機構の情報の更新を指示する。分岐予測の失敗が判明すると、命令フェッチ部に通知して再命令フェッチを起動し、さらに、投機実行中の命令のメモリアクセス要求をキャンセル

ルする。この再命令フェッチとキャンセルの起動もアウト・オブ・オーダーで処理する。そして、分岐命令がコミットする際に FLUSH 信号をあげて、投機実行中の後続の命令を捨てる。このように、分岐命令は他の命令と並列に実行され、また他の命令の実行処理を妨げる事が少なく、実効性能の向上に貢献する。

#### 8) ロード・ストア命令実行：

ロード命令やストア命令は、デコード/発行 ステージで、フェッチ・キュー FQ (Fetch Queue) を確保し、アドレス生成リザベーション・ステーション RSA (Reservation Station for Address generation) に命令を発行する。ストアの場合はさらに、ストアデータ・キュー SDQ (Store Data Queue) と演算リザベーション・ステーション (RSE もしくは RSF) も同時に確保する。

RSA のエントリ数は 10 個であり、投入準備の完了したリクエストの中から、もっとも古いものと次に古いもの (Oldest-Ready、2nd Oldest Ready) の 2 つを同時に選択してアドレス生成演算器 EAGA と EAGB (Effective Address Generation Units A and B) に投入する。アドレス生成の場合にも、他の演算同様、投機的投入とデータフォワーディングを適用する。

EAGA、EAGB でアドレスを生成すると、FQ にセットする。この時、実行待ちの他のアクセス要求がなければ、すぐさまキャッシュアクセスを開始する。アドレスを FQ にセットすると RSA を解放し、ストアデータを SDQ にセットすると RSE もしくは RSF を解放する。

FQ は、キャッシュ・アクセス・パイプラインを制御し、SDQ は、ストアデータのキャッシュ書き込みを制御する。FQ は 16 エントリ、SDQ は 10 エントリである。

キャッシュ・アクセス・パイプラインは、独立して動作できる OF (Operand pipeline F)、OG (Operand pipeline G) と呼ばれる 2 本のパイプラインで構成され、同時に 2 つのアクセス要求を受け付ける。

キャッシュ・ミスをはじめとして処理が継続できない場合には、キャッシュ・アクセス・パイプラインはアボートする。しかし、OF、OG のアボートは独立であって、互いに他方がアボートしても処理は継続できる。また、アボートしたパイプラインで、実行中の後続のリクエストは、それ自身にアボートする要因がない限りは、アボートしたリクエストを追い越して処理を継続することが出来る。アボートしたリクエストは FQ から再投入する。

ストアはキャッシュ上に対象ラインが存在しない場合にも、これを SDQ 上に保留したまま命令をコミットして後続の処理を続行する、いわゆるストアの突き放し処理が可能である。SDQ はストアの書き込み処理を行って開放する。また、ロード命令が先行するストア命令の結果を参照しようとする場合には、SDQ からデータをバイパスする事が可能である。

このように、SPARC64 V では 2 つのオペランドアクセスを同時に行うことができ、ノン・ブロッキング、高スループットを実現している。

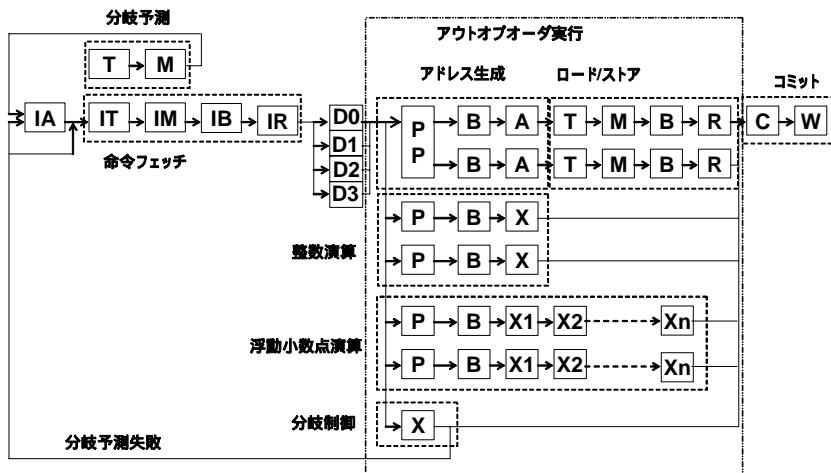


図5 SPARC64 V のパイプライン

### 2.3 キャッシュ

SPARC64 V のキャッシュは、中容量の 1 次キャッシュ(L1\$)と大容量の 2 次キャッシュ (L2\$)の二階層で構成される。

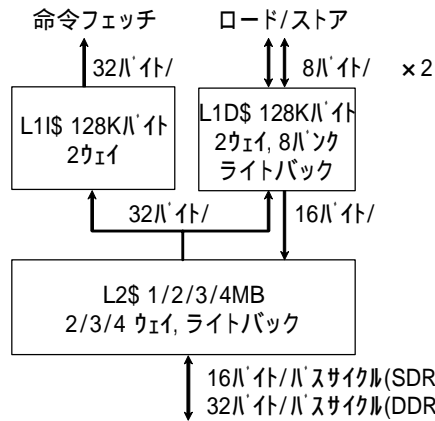


図6 SPARC64 V のキャッシュ構成

L1\$は、命令専用のキャッシュ(L1I\$)とオペランド(データ)専用のキャッシュ(L1D\$)で構成され、いずれも容量 128K バイト、2 ウェイ・セットアソシアティブ、ブロックサイズ

は 64 バイトである。L1D\$は、4 バイトアドレス境界によって、8 つのバンクに分割されており、バンクごとに別のアドレスでアクセスできる。前述の 2 つのオペランドの同時アクセスは、この仕組みによって実現している。L1\$は、キャッシュインデックスに仮想アドレスを、キャッシュタグには物理アドレスを用いる(VIPT : Virtually Indexed Physically Tagged)。VIPT では、メモリの同じ領域が異なる仮想アドレスでアクセスされると、キャッシュの別のインデックスに登録されて、一貫性を損なうおそれがある。これをシノニム問題と呼ぶが、SPARC64 V では L2\$との連携によってハードウェアで解決している。L1I\$は、V(Valid)・I(Invalid)の 2 ステートで制御され、L1D\$は、M(Modified)・C(Clean)・I(Invalid)の 3 ステートで制御される。置換方式は L1I\$、L1D\$ともに LRU(Least Recently Used)である。

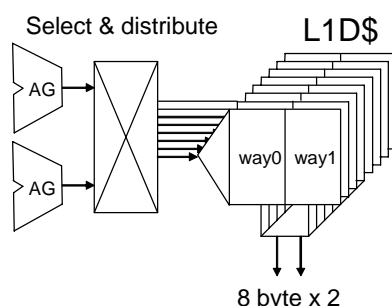


図7 SPARC64 VのL1D\$

L2\$は、容量 1/2/3/4MB、2/3/4 ウェイ・セットアソシアティブ、ブロックサイズは 64 バイトである。キャッシュインデックス・キャッシュタグ共に物理アドレスを用い(PIPT : Physically Indexed Physically Tagged)、M(Exclusive Modified)・O(Shared Modified)・E(Exclusive Clean)・S(Shared Clean)・I(Invalid)の 5 ステートで制御される。L2\$の置換方式は LRU である。L2\$のデータ部はコアの半分の周波数で動作し、アクセス幅は 64 バイトである。データ部の動作周波数が半分である事は省電力に貢献するが、この他にもデータ部を構成する RAM のうち、データを使用しない部分にクロックを印加しないことで、更に省電化している。

L1\$、L2\$とも、ライトバックと呼ばれるストア制御方式であり、書き込みはその階層のキャッシュに対して行う。ライトバック方式では、キャッシュ・ミスした場合には、ストアであってもメモリ上の旧データを一旦キャッシュに持ち込む必要がある反面、キャッシュ・ヒットしたストアは、キャッシュ上だけで完了する。一般的にストアの頻度はかなり高いために、ライトバック方式は、キャッシュ間のトラフィックやメモリアクセスのトラフィックが削減できる点が有利である。

ライトバック方式で、キャッシュ上のデータに対して自プロセッサ以外からアクセス要求があった場合、または、キャッシュ上のデータをリプレースする場合には、下位(外側)

階層のキャッシュまたはメモリに書き込みを行う。マルチプロセッサ構成で高性能を目指す場合には、前者、すなわち自プロセッサ以外からのアクセス要求に対する応答の速さが問題となる。しかし、SPARC64 V の 2 階層キャッシュは、L1\$ と L2\$ の間が緊密であり、L1\$ からのデータのはきだしを高速に処理する。

また、ライトバック方式では最新データをキャッシュ上に握っているために、そのプロセッサでエラーが発生すると、被害がプロセッサ内に止まらずにシステムに及ぶ危険性が高い。しかし SPARC64 V は強力な RAS 機能で対処する。

L2\$ から読み出したデータを L1\$ に送るバスは 32 バイト幅でコアの周波数で動作する。従って L2\$ の読み出しと L1\$ への転送のスループットが一致する。一方、L1\$ から L2\$ にデータを送るバスは 16 バイト幅でコアの周波数で動作する。いずれもコアの命令実行の最大スループットに匹敵するスループットを確保しており、L1\$ のヒット率が極度に低下するような過酷な状況においても、処理の停滞を招きにくい堅牢さの一端を担う。

以上で説明してきたように、SPARC64 V のキャッシュは、大規模トランザクション処理にも HPC 処理にも適合し、常にプロセッサが高性能を発揮することを支えるものである。

## 2.4 システムインタフェース

SPARC64 V のシステムインタフェースでは、従来の SPARC プロセッサに比べて、発行した要求が完了する前に次の要求を発行できる数、すなわちアウトスタンディング数を大幅に強化している。リード要求、メモリへの書き戻し(ライトバック)のアウトスタンディング数は、それぞれ 16 個、8 個であり、また、システムからプロセッサへのコヒーレント要求も 4 個アウトスタンディング出来る。さらに識別子によって要求と完了の対応をとることにより、リード要求とその完了の順序をはじめとして、順序制約を緩和している。

これらの機能強化により、SPARC64 V のシステムインタフェースでは、システム動作の並列度を向上させ、高性能の達成に貢献している。なお、従来機種との互換動作を含め、旧来のシステムの制約に合わせた設定も可能である。

## 2.5 TSO(Total Store Order)の保証

OS を含む既存のソフトウェアが動作するためには、ロードの実行結果がストアの実行順序を正しく反映しているように見える必要がある。これは、TSO あるいは、実行順序の一貫性と呼ばれる。

SPARC64 V は、キャッシュ・ミスしてもアウト・オブ・オーダーで処理を継続して高い並列度を維持する。しかしこのため、キャッシュ・ミスなどの理由でロードが滞っている間に、後続のロードをアウト・オブ・オーダーで実行して、その後で滞っていたロードを実行すると、先行するロードが最新データを取り出し、先に行なわれた後続のロードが古いデータを取り出して、TSO に違反する危険がある。そこで SPARC64 V は、FQ のオペランドアドレスを外部からの無効化やムーブアウトのアドレスと比較するとともに、外部からのムーブインを監視する。これにより、キャッシュ上にあった領域を他のプロセッサがストアした可能性とともに、遅れて実行されたロードが新しいデータを受け取った可能性を監視して、TSO に違反する可能性がある場合には、そのロードの次の命令からの再実行を要求する。

この機能の装備によって、SPARC64 V は、L2\$ の動作も、外部へのメモリアクセス要求も、アウト・オブ・オーダーで実行し、フルレンジのアウト・オブ・オーダー実行を実現している。

## 2.6 MMU(Memory Management Unit)

SPARC64 V の TLB(Translation- Lookaside Buffer)は、mTLB(main TLB)と  $\mu$  TLB(micro TLB)

の二階層から構成される。ページサイズは 8K バイト、64K バイト、512K バイト、4M バイトの 4 種類をサポートする。TLB は、命令フェッチ用とオペランドアクセス用にそれぞれ独立して同じ構成の一組を装備する。以下ではこの一組について説明する。

mTLB は、RAM 部と CAM(Contents Access Memory)部から構成される。RAM 部は、1024 エントリ、2 ウェイ・セットアソシアティブ構成である。RAM 部は 8K バイトページ専用である。RAM 部のリプレース・アルゴリズムは、LRU である。CAM 部は、32 エントリのフルアソシアティブ構成である。8K バイト以外のページサイズのエンタリは CAM 部に登録する。また、8K バイトページであっても、ロックエンタリは CAM 部に登録する。CAM 部のリプレース・アルゴリズムは擬似 LRU である。

$\mu$ TLB は L1\$ をアクセスするパイプラインから高速に索引するための mTLB の部分コピーである。32 エントリのフルアソシアティブ構成で、リプレース・アルゴリズムは、FIFO(First In First Out)である。 $\mu$ TLB は、全てのページサイズに対応し、ロックエンタリもリプレース対象である。 $\mu$ TLB ミスではパイプラインはアボートするが、後続のロード、ストアを処理する間に、ハードウェアがバックグラウンドで mTLB からコピーして再起動する。

このように SPARC64 V の TLB は大容量と高速アクセスを両立させている。

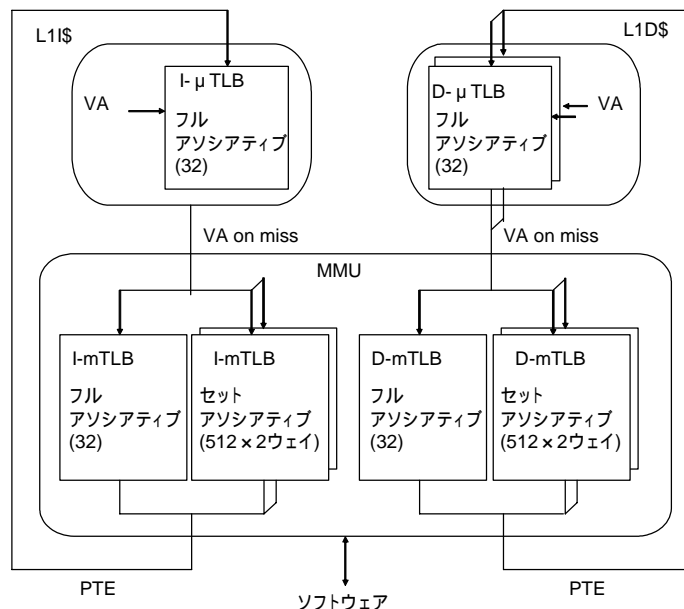


図8 SPARC64 VのMMU

## 2.7 プリフェッチ機構

SPARC64 V の備えるプリフェッチ機構を説明する。

### 1) 連鎖プリフェッチ (Chained pre-fetch) 機構

連鎖プリフェッチ機構は、連続したアドレスに対するリード要求があった場合には、さらに先の連続アドレスに対するアクセスが将来行われる可能性が高いという予測に基づいて、ハードウェアが投機的にリード要求を発行するものである。

連続アクセスを検出するため、SPARC64 V は専用アドレステーブルを備える。L1\$ミス時には、次のアドレス、すなわち +64 を、このテーブルに登録し、後続のキャッシュミスアドレスが、すでにテーブルに登録済みであった場合には、連続アクセスを検出し、さらに次のアドレスに対してプリフェッチを要求する。

この連鎖プリフェッチ制御のためのアドレステーブルは、命令フェッチ用とオペランドアクセス用それぞれ 16 エントリずつあり、それぞれ 16 系列のストリームに対応可能である。なお、連鎖プリフェッチでは L2\$までロードする。

## 2) ソフトウェア・プリフェッチ機構

プログラムによってデータを使用するより前に、あらかじめキャッシュ上にデータをロードするようなスケジューリングがコンパイラによる高速化手法の一つとして一般的に行われる。SPARC64 V では、キャッシュのどの階層にロードするか、キャッシュ状態を Modified/Shared どちらにするか、TLB ミス時に抑止するか否かなど、プリフェッチ命令のきめ細かい制御オプションをサポートすることで、コンパイラによる効果的なスケジューリングを支援する。

SPARC64 V は上記以外にも、リード要求用のポートが不足している場合に、単に要求を保留やアボートする代わりに、プリフェッチ要求を発行するなど、積極的にプリフェッチを行う。これによって、メモリリード処理の並列度を向上し、キャッシュミスペナルティを可能な限り隠蔽する。

## 第 3 章 SPARC64 V の RAS

SPARC64 V は、メインフレームに匹敵する RAS の実現を狙う。確実にエラーを検出すること、エラーの影響範囲を限定すること、回復処理を試みること、エラーの記録を残すこと、ソフトウェアに通知することなど、RAS の基本的な事項を徹底した。これにより SPARC64 V は、ミッションクリティカルな UNIX サーバのプロセッサとして、高度な信頼性、可用性、サービス性、データ安全性を提供する。

### 3.1 内蔵 RAM の RAS

RAM はプロセッサ内で最もエラーの発生頻度が高い部分である。SPARC64 V は、RAM の 1 ビット・エラーは、ソフトウェアの介入なしに、全てハードウェアで自動的に訂正し、ソフトウェアへの影響は皆無である。

表 1 SPARC64 V の RAM のエラー保護

RAM 種別	エラー検出/保護方法	エラー訂正方法
L1I\$データ	パリティ	無効化および再読み込み
L1I\$タグ	パリティ + 二重化	二重化データの再書き込み
L1D\$データ	SECDED ECC	ECC による 1 ビットエラー訂正
L1D\$タグ	パリティ + 二重化	二重化データの再書き込み
L2\$データ	SECDED ECC	ECC による 1 ビットエラー訂正
L2\$タグ	SECDED ECC	ECC による 1 ビットエラー訂正
命令 mTLB	パリティ	無効化
データ mTLB	パリティ	無効化
ブランチヒストリ	パリティ	分岐予測失敗からの回復

#### 1) ECC によるエラーの検出と訂正動作：

L1D\$のデータ部、L2\$のデータ部、L2\$のタグ部は、1 ビットエラー訂正 2 ビットエラー検出(SECDED : Single Error Correction Double Error Detection)の ECC(Error Correcting Code)を備えている。L1D\$、L2\$ともに、読み出したデータがエラーしている場合は、ECC 回路で訂正し、訂正したデータを演算など以降の処理に使用する。従って、RAM の固定障害であっても、1 ビットエラーであれば救済する。また、エラーを保持していた RAM のエ

ントリには、訂正したデータを上書きして間欠エラーを消去し、修正不能な多ビットエラーが発生することを防止する。L2\$のタグ部は、1ビットエラーを検出した場合に、内容を全ビット反転して再書き込みする、リバースライト機能を持ち、1ビット固定故障を救済する。

## 2) パリティによるエラーの検出と訂正動作：

L1I\$のデータ部、L1I\$とL1D\$のタグ部、ブランチヒストリのタグおよびデータ部、mTLBはパリティによってエラー検出する。

L1I\$は、読み出したデータがエラーしている場合、データを捨てて、該当するエントリを無効化する。L1I\$ミスとなった状態で再起動し、このときL2\$から読み出したエラーのないデータをバイパスして以降の処理に使用する。これによって1ビットエラーは固定障害も救済する。RAMに訂正したデータを上書きして間欠エラーを消去するのは、L1D\$、L2\$と同じである。

L1I\$とL1D\$のタグ部はコピーがあって二重化されている。いずれかでエラーが発生した場合には、そのキャッシュラインを互いに他方に通知して、エラーしてい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。

ブランチヒストリは、RSBRで分岐命令の処理を行う過程で予測内容を検査する。エラーがあれば必然的に予測失敗となり、予測失敗の回復動作の過程で、命令実行に対するエラーの影響は取り除かれる。したがって、エラーのビット数にも、間欠か固定の別にもよらず救済する。パリティはエラーログの生成に関与するだけである。

mTLBで1ビットエラーが検出されると無効化される。これによりTLBミスとなって、その回復の過程で再度エントリが登録されて、エラーが消去される。

このように、パリティによるエラー保護の場合にも、RAMの1ビットエラーは全てハードウェアによる回復処理を行い、ソフトウェアに対するダメージは皆無である。

## 3) 縮退：

L1\$, L2\$, mTLBはウェイ単位での縮退を行う。エラーの発生回数を機能単位ごとにカウントし、単位時間あたりのエラー回数が上限値を越えると縮退して、それ以降はそのウェイを使用しないようにする。縮退はハードウェアで自動的に行うが、これに際して、コヒーレンスを維持するための保証動作もハードウェアで自動的に行う。すなわち、L1D\$の縮退されるウェイのダーティラインを全てL2\$に書き戻す動作、L2\$の縮退されるウェイのダーティラインをメモリに書き戻す動作を伴ってウェイ縮退を行う。

ウェイ縮退は、ソフトウェアにダメージを与えることなく実行され、処理速度の低下を除いて、ウェイ縮退中およびウェイ縮退後に、ソフトウェアは動作に影響を受けない。

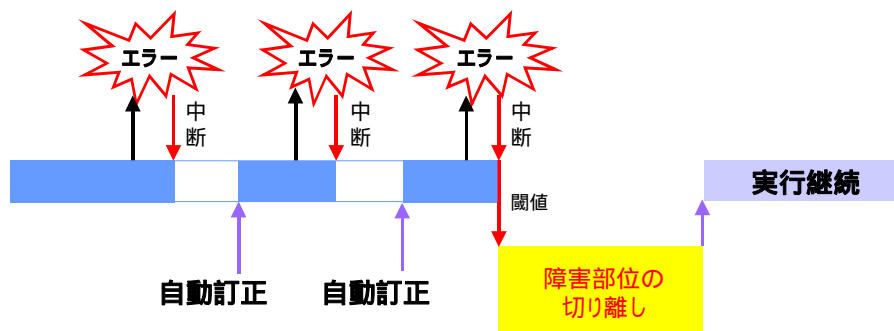


図9 内蔵RAMの自動エラー訂正と縮退

### 3.2 メモリのRAS

SPARC64 V は、メモリのエラー保護とエラー分離の機構を装備する。

#### 1) ECC によるエラー保護と訂正

PRIMEPOWER のメモリは、ECC で保護されている。SPARC64 V がメモリデータをムーブインして1ビットエラーを検出すると、ハードウェアで訂正して内部で使用する。修正不能な多ビットエラーは、割り込みでソフトウェアに通知する。

#### 2) ソフトウェア介入によるメモリエラーの訂正

SPARC64 V が検出したメモリの1ビットエラーおよび多ビットエラーは、エラー表示レジスタにアドレスとエラー種別が示されて、割り込みまたはポーリングによりソフトウェアが認識する。メモリの1ビットエラーを認識すると、ソフトウェアはエラーしたメモリを上書きして、該当するキャッシュラインをダーティに変更する。これにより、メモリに書き戻して1ビット間欠エラーを消去することができる。

また、ソフトウェアが一定の時間間隔で広範囲のメモリを読み込むことにより、メモリの1ビットエラーを検出し、上記の手順でそれを消去することができ、1ビット間欠エラーを保持し続けることによる多ビットエラーの発生を防止する効果がある。

#### 3) メモリのエラーマーキング

読み込んだメモリに多ビットエラーがある場合に、そのデータにエラーの発生元を示す識別子を埋め込み、シンドロームが特別な値となる様な ECC を付加する動作を行う。これをエラーマーキングという。メモリの多ビットエラーを検出した装置は、エラーがマークされているか否かにより、自身が発生元か否かを判定する。エラーマーキングは、エラーの発生元を特定するために有用な情報を提供し、余分なエラーログを防止して必要な情報収集を助ける。

### 3.3 内蔵レジスタと演算機のRAS

SPARC64 V は、レジスタや演算器もエラー保護し、データ保全性は万全を期する。

表 2 SPARC64 V のレジスタと演算器のエラー保護

演算器種別 / レジスタ種別	エラー保護機構
GPR, FPR, JWR, GUB, FUB	パリティ保護
PC, PSTATE など	パリティ保護
演算入出力レジスタ	演算前後データについてのパリティ保護
加減算器	パリティ予測回路による出力へのパリティ付加
乗除算器	剰余チェック回路による検算と パリティ予測回路による出力へのパリティ付加
シフタ	パリティ予測回路による出力へのパリティ付加
グラフィック演算器	パリティ予測回路による出力へのパリティ付加

#### 1) 内蔵レジスタの保護

GPR や FPR にパリティを備える。GPR に関してはウィンドウレジスタ本体だけでなく、その写しである JWR にもパリティを備え、JWR のエラーはウィンドウレジスタ本体からハードウェアが再ロードして自動的に回復する。

また、PC(Program Counter)や PSTATE (Processor STATE)レジスタ、割込み表示レジスタなどの制御レジスタにもパリティを備え、エラーによるプログラムの暴走など被害拡大を防止する。

#### 2) 演算器の保護

演算器の入力にパリティを備え、演算器の中間レジスタや出力レジスタのパリティをチェックする。また、乗除算では、入力のパリティをチェックした上で、CSA(Carry Save Adder)ツリーでは剰余チェックによる検算を行い、そのあと加算器を通過する段階ではパリティ予測を行って、出力レジスタのパリティをチェックする。

演算器の出力のパリティはデータと共に GUB、FUB にセットされ、GPR、FPR に書き込まれるまで伝搬する。

この他に、レジスタアドレス、メモリアドレス、キャッシュインデクス、命令コード、システムインタフェースのアドレスバスやデータバス等あらゆる情報は監視の対象として、出来る限りのエラー保護を適用している。さらに命令実行が長時間滞った場合、いわゆるハングアップも、専用ハードウェアで検出する。

### 3.4 同期一括更新方式と命令リトライ

命令実行部で説明したように、SPARC64 V は同期一括更新方式を採用している。エラーが検出されると、その時点で実行途中の処理を全てキャンセルする。コミットまでの間の途中結果は廃棄可能であるから、エラーに遭遇することなく完了した命令が更新した結果だけがプログラマブルな資源に残る。

このことは、エラーによるプログラマブルな資源の破壊を防止するばかりではなく、エラー検出後にハードウェアで命令リトライを行う機会を与える。ハングアップの場合にも、滞っている処理をいったん捨てて、はじめからやり直すことが出来るので、回復の可能性はある。

命令リトライはエラーを契機に自動的に起動する。リトライ中は、1 命令ずつ実行して正常実行の可能性を上げ、正常に実行完了すると、通常の実行状態に自動的に復帰する。この間ソフトウェアに介入は不要であり、命令リトライが成功すれば、エラーのソフトウェアへの影響は皆無である。

命令リトライは閾値に達するまで繰り返し、これを越えた場合には、エラーの発生を割

り込みによって、ソフトウェアに通知する。

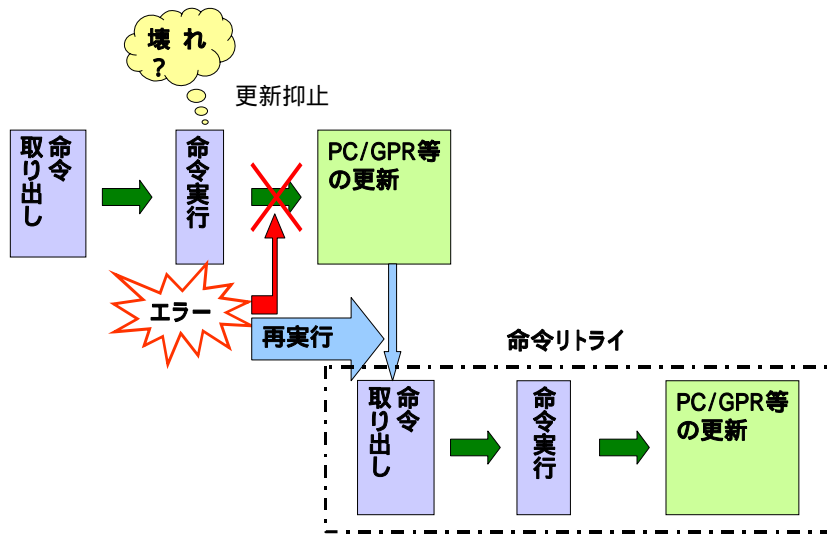


図10 ハードウェア命令リトライ

### 3.5 エラー処理のソフトウェア・インタフェース

SPARC64 V は、ソフトウェアに割り込む際に、ソフトウェアに与える影響の大きさによりエラーを分類し、エラーの発生個所を資源毎に表示する。これによって、ソフトウェアがエラーの波及範囲を限定し、また、故障個所に応じた処理ができる様にする。さらに、エラーに関する情報を収集するダンプ処理を支援するためにエラーを可能な限り無視して動作するモードを装備する。

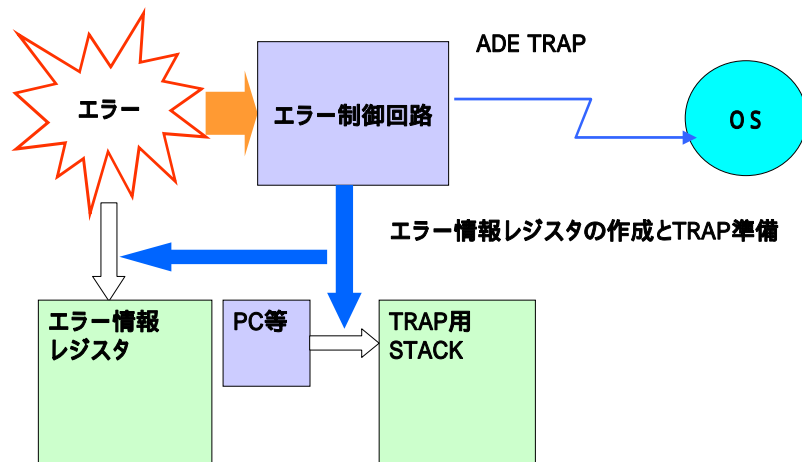


図11 ソフトウェア割り込みとエラー情報レジスタ

### 3.6 サービス性の向上

これまで説明したように、SPARC64 V は、ありとあらゆる個所にエラーのチェックを装備する。この結果として、エラーがどこで見つかったかを表示するリージョンコード (RC) は 679 個、またエラーに関する補助情報(シンドロームやエラーアドレスなど)を表示するリテイン(RT)は 821 個を備える。さらに、エラー発生時点の時系列の記録を 80 ビット幅×1,024 インデックスの RAM8 個から成るイベントヒストリに保存する(90nm 版 SPARC64 V では RC が 803 個、RT が 3,077 個、ヒストリが 80 ビット幅×1,024 インデックスの RAM7 個)。

エラーの発生時には、専用のインタフェースによってシステムに通知する。この通知により、SCF(System Control Facility)ファームウェアは、専用のインタフェースを使用して、エラーログを収集し、その解析を行う。これらの動作は、ソフトウェアに影響を与えずにバックグラウンドで行う。

これらの機構によって、SPARC64 V を搭載するシステムは、運用を継続しながら故障個所と故障種別を迅速かつ正確に特定し、予防保守に有用な情報を得て、サービス性の向上を図ることができる。

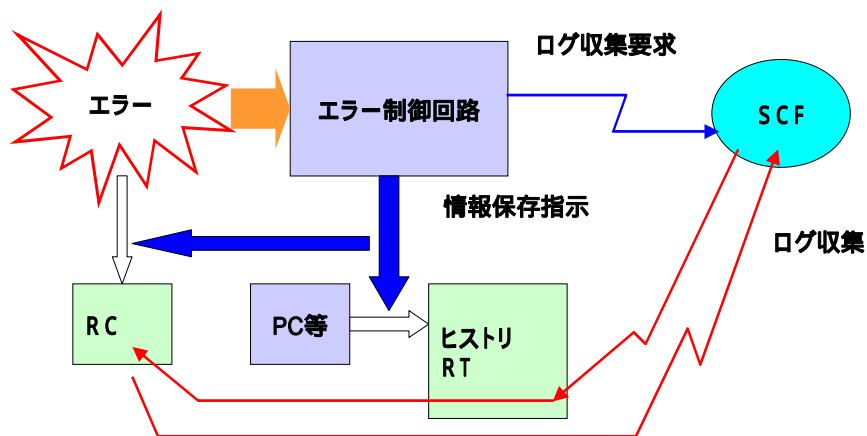


図12 RC/RTとヒストリによるエラーログ収集

## 第4章 SPARC64 V の半導体テクノロジー

SPARC64 V は 130nm または 90nm の CMOS プロセスで製造される。

### 4.1 130nm プロセスの SPARC64 V

130nm CMOS プロセスの SPARC64 V は Cu 8 層、トランジスタ数は約 191 百万、信号 269 ピン、サイズ 18.11mm × 15.99mm である。動作周波数は 1.35GHz を達成し、消費電力は約 50W である。



図 13 130nm プロセスの SPARC64 V チップ写真

#### 4.2 90nm プロセスの SPARC64 V

90nm CMOS プロセスの SPARC64 V は Cu 10 層、トランジスタ数は約 400 百万、信号 279 ピン、サイズ 18.46mm × 15.94mm である。動作周波数は 2GHz 以上を達成した。

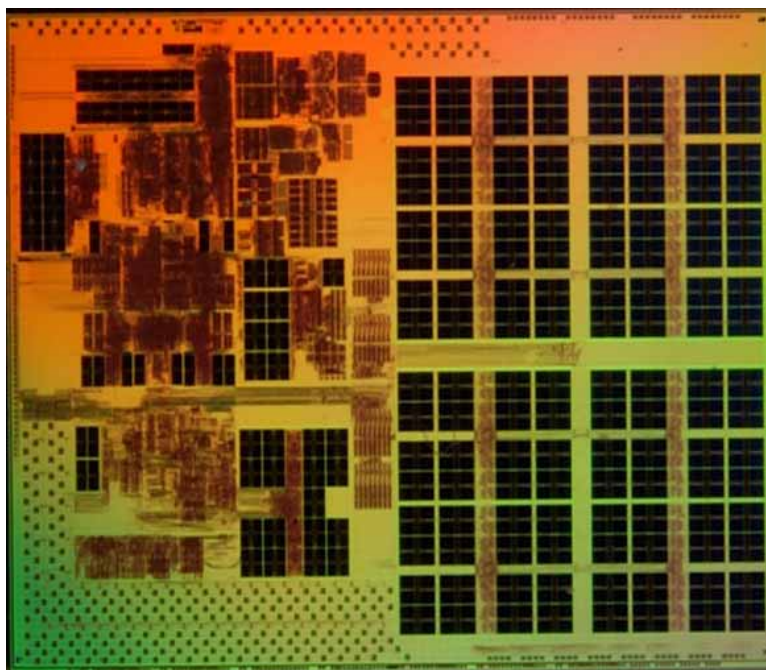


図 14 90nm プロセスの SPARC64 V チップ写真

### 第 5 章 SPARC64 V の性能評価

近年のプロセッサの開発の流れにおいて、処理速度の向上と開発期間の短縮の両立が、課題となっている。プロセッサ開発手順は、最初に目標性能を設定し、それを実現するマイクロアーキテクチャの策定を行う。処理速度の向上の側面から、採用する LSI テクノロジ、目標クロック速度、目標チップ面積などの既決の基盤が許す最大限の性能(IPC)を引き出すことが求められ、戦略的な新規機能が策定される。開発期間の短縮の側面では、開発の効率化への努力が一層求められる。ハードの作り直しの低減は、効率化の重要な戦略である。戦略的な機能を策定し、かつハードの作り直しの低減を実現するために、策定したマイクロアーキテクチャが目標性能を満たすことを、開発初期の段階に確認する手段が必須である。我々は、この性能解析を、ソフトウェア性能シミュレータ(以下、性能モデル)で行った。

富士通研究所では、過去メインフレームの時代から、マイクロアーキテクチャ策定に役立つ性能評価技術を開発してきた。特に、性能モデルの開発に力をいれてきた。SPARC64 V の開発にあたって、メインフレーム向けの性能モデル開発の経験と実績をもとに、SPARC V9 用性能モデルを開発した。SPARC64 V の開発に企画段階から参加し、基本仕様決定、詳細論理決定、仕様変更まで、本性能モデルを用いた性能評価を行ってきた。多くのマイクロアーキテクチャを本性能モデルで評価した。その中には、基本パイプライン構造、On-Chip 2 次キャッシュ、分岐予測方式、スーパースカラ・アウト・オブ・オーダーのための様々なバッファ量、およびハードウェアプリフェッチ機能などがある。本性能

モデルは、実機で採集されたトレースを入力としたトレースドリブン型であり、その特徴は、

- ・評価ターゲットを忠実にシミュレートするモデルの精度の高さ
- ・プロセッサとメモリシステムからなる、モデルの構成の一貫性
- ・サーバ・システムの性能評価までを可能とする、評価範囲の広さ

である。

"精度の高さ" は重要なポイントである。初期評価のために、概略の仕様に基づくモデルでシミュレータの開発を開始し、論理設計の進捗にあわせて、その論理をモデル化し、シミュレータに組み込みながら、全体の性能モデルの精度を上げていった。"一貫性"は、従来は大事にされておらず、プロセッサのみをモデル化しメモリシステムにはラフなモデルが使われることが多かった。しかし、近年のプロセッサ速度とメモリ速度のギャップによるメモリまわりの遅延が大きく性能を低下させる原因になっている。そのため、本性能モデルではプロセッサとメモリシステムからなる一貫したシミュレータを採用している。

"評価範囲"については、従来は小さなUPのベンチマークを評価対象としていた。しかし、今後のUNIXサーバの主適用市場である、トランザクションプロセッシングやHPC分野のベンチマークの評価に対応する必要がある。そのため、本性能モデルはマルチプロセッサをモデル化している。

本性能モデルはC言語で記述され合計約9万ステップである。入力パラメータは、およそ500個が定義されている。分岐予測などの様々な方式、モデル各部のリソース量、遅延などを指定することができ、非常に詳細で柔軟なモデリングを可能にしている。

SPARC64 Vの開発にあたり、各種設計パラメータの決定の際には、このような性能評価技術を適用した。

## むすび

SPARC64 Vは多くの高性能技術を投入し、また、これまでのUNIXサーバでは達成し得なかった高度なRASを実現した。消費電力も既存のサーバ・プロセッサに比べて小さく、高性能でコンパクトなシステム構築を約束する。そして、UNIXサーバにおけるデファクトスタンダードなOSであるSolaris™ Operating Environment(以降、Solaris OE)が走行し、最も豊富な品揃えを誇るSolaris OE上のサーバ用ミドルウェアやアプリケーション群が使用できる。

SPARC64 Vの最初のチップは2001年の12月末に製造されて、富士通川崎工場の片隅で関係者に見守られる中で産声をあげた。このプロセッサが世に出るまでには、数多くの人々の大変な努力を要した。しかし、このプロセッサは今、ユーザの期待に最も良く応えるものであると確信している。

## 参考文献

1. SPARC International Inc.: The SPARC Architecture Manual-Version 9, 1994.
2. Sun Microsystems, Inc. and Fujitsu Limited: SPARC Joint Programming Specification (JPS1): Commonality.  
<http://www.fujitsu.com/downloads/PRMPWR/JPS1-R1.0.4-Common-pub.pdf>
3. Fujitsu Limited: SPARC JPS1: Fujitsu SPARC64 V Implementation Supplement.  
<http://www.fujitsu.com/downloads/PRMPWR/JPS1-R1.0-SPARC64V-pub.pdf>
4. 利根 廣貞、杉浦 聡、豊木 則行、千葉 隆: GS8600のハードウェア. FUJITSU, Vol.47, No.2, p.96-108(1996).
5. 引地 徹、加藤 哲、大田 秀信、嘉喜村 靖: 64ビットRISCプロセッサ:SPARC64 . FUJITSU, Vol.51, No.4, p.226-231(2000).  
<http://magazine.fujitsu.com/vol51-4/paper06.pdf>
6. D.H. Brown Associates, Inc.: SPARC64 V Microprocessor Provides Foundation for

PRIMEPOWER Performance and Reliability Leadership.

[http://primeserver.fujitsu.com/primepower/catalog/data/pdf\\_db/sparc64\\_v.pdf](http://primeserver.fujitsu.com/primepower/catalog/data/pdf_db/sparc64_v.pdf)

7. 井上 愛一郎:UNIX サーバ用プロセッサ: SPARC64 V. FUJITSU, Vol.53, No.6, p.450-455(2002).

<http://magazine.fujitsu.com/vol53-6/paper05.pdf>

8. Kevin Krewell: Fujitsu's SPARC64 V Is Real Deal. Microprocessor Report, October 21, 2002.