社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ARC－155　(8)
2003／11／28

サーバ用CPUのハードウェア資源削減に基づくチップマルチプロセッサの設計

河場基行[†]　大河原英喜[†]　安島雄一郎[†]
安里彰[†]　安藤寿茂[††]

ビジネスアプリケーション分野で使用されるシステムは，単体スレッドの性能よりむしろスループット性能が要求される．この要件に答えるため我々は小型のプロセッサを複数搭載したCMPの検討を行っている．本論文は，既存のサーバプロセッサであるSPARC64 Vをベースした小型CPUコアの設計について述べたものである．

SPARC64 Vのシミュレータや実チップデータを利用しながら，4ステップにわたる段階的なハードウェア削減を行った結果，コア面積で54.5%，性能で70.9%程度を達成するCPUコアが実装できることがわかった．またこのコアを用いた2コアCMPによりほぼ同一チップ面積で22%のスループット向上が得られることを確認した．

Designing High Throughput Chip Multiprocessor by Reducing Hardware Resources of Server Processor

Motoyuki Kawaba,[†] Hideki Okawara,[†] Yuichiro Ajima,[†]
Akira Asato[†] and Hisashige Ando[††]

The commercial workloads requires total throughput rather than the performance for a single thread. To meet this situation, we have studied a CMP system with simple processors. This paper describes the design of the simple processor based on a server processor, SPARC64 V.

Utilizing the system simulator of SPARC64 V and actual chip information, we have applied 4-staged reduction of CPU hardware resources carefully. The performance of our eventual processor core is 70.9% of SPARC64 V, while the core occupies only 54.5% area. The CMP system with 2 CPU cores can deliver 22% higher through-put than SPARC64 V.

1. はじめに

これまでのハイエンドプロセッサ開発は，増加するトランジスタを単体スレッドのIPC向上に注ぎ込んできた．たとえば同時命令発行数を増やすためのスーパスカラ的なリソースやオンチップキャッシュの増加がこれにあたる．ところが近年，命令レベル並列(ILP)追求の限界が指摘されており，CMPに代表されるようなトランジスタをスレッドレベル並列(TLP)向上に投入する動きが活発である[1)2)3)]．

サーバ上で稼働するアプリケーションの側面からみてもTLP向上が要求されている．ハイエンドプロセッサが利用されることが多いビジネスアプリケーション分野では，単一の処理よりもトータルスループットの性能が要求される．このためビジネスアプリケーション分野ではシンプルなCPUコアを複数搭載するCMPが適していると考えられる．

これらの状況から，我々は小型のプロセッサを複数搭載し同一チップ面積で高スループットを達成するCMPの検討を行っている．検討にあたりハイエンドプロセッサであるSPARC64 V[5)6)]をベースにプロセサコアの小型・簡単化を行った．既存のサーバプロセッサからデザインを開始した理由は，設計期間の短縮だけではない．誤差検証が十分になされた高精度な性能シミュレータや実際のチップの実データの利用が可能であるため，より詳細な検討が可能であるといった利点もある．

第2節で，CPUコア小型化に関する目標を述べる．第3節でベースプロセッサであるSPARC64 Vの概要を述べる．ついで第4節でCPUコア小型化の方針を述べ，第5節で評価環境，第6節でCPUコア小型化の結果を示す．最後に考察しまとめを述べる．

† (株) 富士通研究所
FUJITSU LABORATORIES LTD.
†† 富士通株式会社
FUJITSU LIMITED

2. CPU コア小型化に関する目標

CPU コアを小型化するにあたり，SPARC64 V と同じテクノロジでかつ同じチップ面積上に 2 個の CPU コアを搭載した CMP の構築を目指すことにする．このため SPARC64 V から 2 次キャッシュを除いた部分のチップ面積を半分にすることを考える．

Pollack のルール[4] によれば，IPC はトランジスタ数の 0.5 乗に比例する傾向がある．このルールはもともとトランジスタの投入によるアーキテクチャ的な性能向上が小さくなりつつある傾向を指摘したルールであるが，ハードウェア物量と性能の目標値の設定に使用することができる．トランジスタ数とチップ面積が比例すると仮定すれば，同一チップ面積に CPU コアを 2 個搭載した場合には 1 コア当りのチップ面積が 1/2 程度となるため，1 コア当りの性能は 70% になることが期待される．

ただし Pollack ルールは適切にハードウェアリソースを配分したときに成立するルールであることは留意すべきである．ベースプロセッサ SPARC64 V のハードウェアリソース量は高い IPC を実現するために最適化されている．このため半分のチップ面積で 70% の性能を達成するには CPU コア内のさまざまなリソースについて検討しなくてはならない．我々は CPU コア面積を半分にするという目標設定から，先ず同時命令発行数と機能ユニット数を半分にした．この後，シミュレーションによって稼働率や使用率の低いハードウェアリソースを特定しハードウェア物量を削減するというアプローチをとった．

3. SPARC64 V の概要

ベースプロセサの SPARC64 V の概要を説明する．図 1 に SPARC64 V のブロック図を示す．

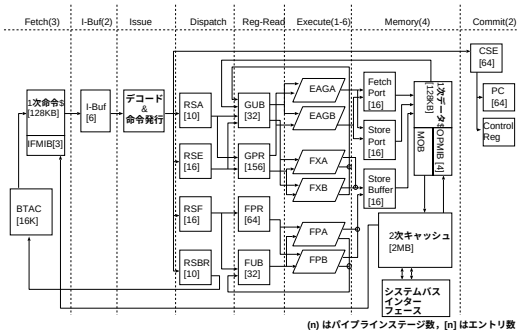

図 1 SPARC64 V のブロック図

SPARC64 V プロセサは 4 命令発行のスーパスカラアーキテクチャを採用している．リネーミングレジスタ (GUB/FUB)，コミットスタックエントリ (CSE)，リザベーションステーション (RSA,RSE,RSF,RSBR) を備え，アウトオブオーダ実行を可能にしている．機能ユニットとして整数演算パイプライン (FXA/B)，浮動小数点加算，乗算パイプライン (FPA/B)，ロードストアパイプライン (EAGA/B) を 2 組ずつ持っている．ロード・ストア命令は Fetch Port/Store Port にアクセスアドレスを格納する．また 1 次キャッシュと 2 次キャッシュ間には IFMIB/OPMIB や MOB と呼ばれるバッファがありノンブロッキングアクセスを可能にしている．

1 次キャッシュは命令・データ個別であり，それぞれ 128KB 2way 構成である．2 次キャッシュは 2MB 4way セットアソシアティブキャッシュである．分岐予測に使用される分岐先アドレスキャッシュ(BTAC) を 16K エントリ持つ．TLB は 2 レベル構成になっており，命令アクセス用とデータアクセス用それぞれにマイクロ TLB と呼ばれる小型の TLB(32 エントリ) とメイン TLB と呼ばれる大型の TLB(1024 エントリ 2way) がある．

図 2 に SPARC64 V プロセサのコア面積の内訳を示す (ただし 2 次キャッシュを除く)．この図より CPU コアにおいてメモリアクセス系ロジックおよび 1 次キャッシュが占める面積の割合が大きいことが分かる．

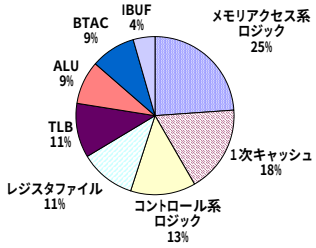

図 2 SPARC64 V の CPU コアおける面積割合

4. CPU コア小型化の方針

先に述べたように SPARC64 V と比較して 50% の CPU コア面積で，70% の単体 CPU 性能達成を目標に CPU コアアーキテクチャを検討する．削減の対象となるハードウェア資源を表 1 に示す．

ハードウェアを削減する手続きの便宜上，4 つの資源グループに分割している．1 つめは CPU の最大処理スループットを決定するデコード幅に関する資源 (機能リソース)，2 つめは 1 次キャッシュに関する資源 (キャッシュリソース)，3 つめはリネーミングレジスタ，リザベーションステーションなどのバッファ関連の資源 (バッファリソース)，4 つめは BAT とメイ

表 1　削減対象となるハードウェア資源

Type	Resource name
機能リソース	デコード幅，機能ユニット
キャッシュリソース	1 次命令/データキャッシュ
バッファリソース	CSE, GUB, FUB RSA, RSE, RSF, RSBR Fetch/Store port, IFMIB/OPMIB
各種テーブル	BTAC, メイン TLB

ン TLB(各種テーブル) である．

これらのハードウェア資源に対し，以下の 4 つのステップで削減を行った．

削減 1　機能リソースの削減
削減 2　キャッシュリソースの削減
削減 3　バッファリソースの削減
削減 4　各種テーブル類の削減

一部の削減を除いて，基本的には性能が 70% になるようにハードウェア資源を削減し，最終的に Pollack のルールに従う CPU コアがデザインできれば CPU 小型化に成功したと判断することにする．

4.1　削減 1: 機能リソースの削減

CPU において最大処理スループットを制限しているものは，命令デコード幅及び機能ユニット数である．多くのプロセッサはこれらを最大限に利用すべく様々なハードウェアリソースを投入している．コア面積を半分にするという目的から，先ずデコード幅を半分 (4→2) にし，全ての機能ユニット数を 1 にする．

4.2　削減 2: キャッシュリソースの削減

図 2 から分かるようにメモリアクセス系ロジックと 1 次キャッシュのチップ面積が CPU コア面積の 43% を占めている．メモリアクセス系ロジックには 1 次キャッシュ制御ロジックが含まれているため，1 次キャッシュサイズ削減は資源削減の効果が高い．そこでシミュレーションによって性能を検証しながら，1 次キャッシュサイズの削減を行う．

4.3　削減 3: バッファリソースの削減

表 1 に示すバッファリソースの削減を行う．最大処理スループットと 1 次キャッシュを削減すると，CPU コア内にあるリザベーションステーションやリネーミングレジスタといったバッファ資源の使用率が下がるため，より多くのバッファリソースが削減できることが期待できる．

数多く存在するバッファリソースに対し，一律の削減を施すことは妥当ではない．要求される性能によって必要な削減量が異なるはずである．そこで，バッファリソースの使用状況に基づいて適切な削減量を見積もった．具体的な方法を説明する．

(1)　シミュレーションによって各サイクル毎にリソースの使用数をカウントし，図 3 に示すような使用数累積ヒストグラムを作成する．

(2)　作成したヒストグラムに基づいて，バッファを 1 個以上使用した実行サイクル数のうち約 60% 程度をカバーするリソース数をリソース量と決定する．

(3)　ハードウェアにインプリメントの都合によって 2 の巾乗に合わせたり，物量と性能のバランスを考慮してわずかに増やすなどのリソース量の調整を行う．

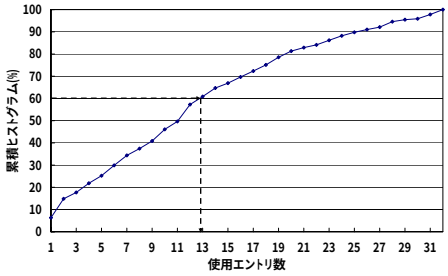

図 3　GUB の使用数と累積ヒストグラム

今回使用した 60% とういカバー率は，各ベンチマーク性能が 70% 強になるよう決定したものである．

4.4　削減 4: BTAC，メイン TLB の削減

シミュレーションで性能を検証しながら CPU コアの 9% の面積を占める分岐予測テーブルとメイン TLB の削減を行なった．今回はマイクロ TLB を削減対象から外した．これはもともと 32 エントリしかなく通常のワークロードのメモリ使用サイズから考えて削減する余地はないと判断したためである．

5.　評 価 環 境

5.1　性能・チップ面積見積について

SPARC64 V のトレースドリブンシミュレータ[7] を用いて性能を見積った．このシミュレータは，SPARC64 V CPU とメモリシステムの詳細モデルを持っておりシステムレベルのシミュレーションが可能となっている．さらに SPARC64 V の開発と並行して実機との誤差検証を行っており，CPU 内部の各モジュールの挙動について正確に評価することが可能である．また機能ユニット，各種バッファサイズ，キャッシュサイズといったハードウェアリソース量を変更することが可能で，様々なアーキテクチャの評価が可能である．

また SPARC64 V 上の実際の面積データに基づいて，ハードウェア物量に応じたチップ面積を算出した．

5.2　対象ワークロード

対象ワークロードとして SPEC CPU2000 と TPC-C を選択した．我々の CMP システムは，対象アプリケーションとしてスループットが要求されるサーバワークロードを想定しているが，より広範囲なアプリケーションの利用を考慮し，CPU ベンチマークである SPEC CPU2000 も評価対象とした．

5.3 トレースデータ生成について

2 種類の命令トレーサを用いてシミュレータの入力となる実行命令トレースを生成した．

SPEC CPU2000 ベンチマークについては，SUN Microsystems によって開発された Shade[8] を使用した．このトレーサはユーザ実行トレースしか採取できないが，非常に高速にトレースを採取する．このため CPU ベンチマークである SPEC CPU2000 の評価に適している．評価時間を短縮するため採取したトレースから代表的な命令トレースを抽出[9] し評価に用いた．

一方，TPC-C ベンチマークは OS 実行の挙動が性能に与える影響が大きいという特長があるため，ユーザ実行とカーネル実行の双方トレースが採取可能な独自トレーサを用いた．このトレーサは Solaris の kdb をベースにしており，マルチプロセッサ環境下でもトレース採取が可能になっている．

6. 削減の詳細

6.1 CPU コア小型化

図 4 にハードウェア資源削減による SPARC64 V に対する相対性能を示す．ここでは便宜上，CINT2000, CFP2000, TPC-C の相対性能の幾何平均を総合性能とした．図 5 に各削減ステップでの CPU コア面積を示す．以降，各削減ステップの効果を説明する．

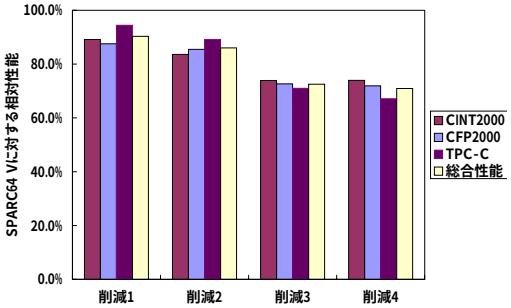

図 4 CPU コア資源削減による性能変化

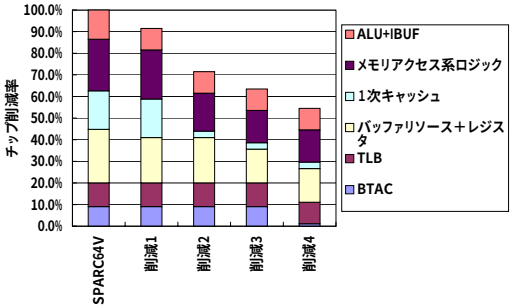

図 5 CPU コア資源削減によるコア面積の変化

6.1.1 削減 1 の効果

図 4 の最も左にあるグラフが命令デコード幅と機能ユニット削減後の性能を示している．総合性能では 90.3% と余裕があるため，これをベースさらに資源削減が可能であることが確認できる．

個別に見ると，CINT2000 が 89%，CFP2000 で 88%の性能であるのに対し，TPC-C では 94% と性能の劣化が少ない．これは全実行時間における CPU コア処理時間の差に起因している．坂本ら[7] が指摘しているように CPU2000 ベンチマークは CPU コア内の処理時間が多い．このため性能劣化が大きくなる．

ここでの削減は面積縮小が直接的な目的ではないが，91.5% までコア面積を削減する結果となった．

6.1.2 削減 2 の効果

1 次命令/データキャッシュサイズを 128K バイトから 8K バイトまで変化させたときの性能を図 6 に示す． この図より，1 次キャッシュサイズ削減に伴って

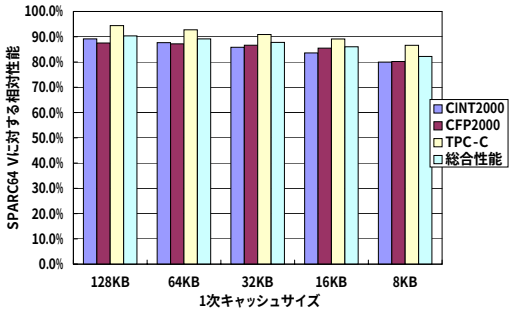

図 6 1 次キャッシュサイズと性能

性能が緩やかに低下していることがわかる．このため 8KB まで縮小しても 82.2% の総合性能が得られる．

しかしながら，図 5 の削減 2 のグラフが示すようにすでに 1 次キャッシュがコアに占める面積が 3.0% と大きくないこと，さらには CINT2000/CFP2000 については 16KB から 8KB にかけて性能劣化がやや加速し始めていることから，キャッシュサイズを 16KB に決定した．この削減によりコア面積は 71.5% になった．

6.1.3 削減 3 の効果

図 4 と図 5 が示すように，バッファリソース削減により総合性能が 72.5%，コア面積は 63.5% になった．図 7 は SPARC64 V に対する各バッファのエントリ数の削減率を示したものである．この図から，全てのバッファリソースが 50% 削減されているわけではないことがわかる．

削減量の多いハードウェア資源は，CSE, RSA/RSBR, StorePort, IFMIB である．CSE，RSBR 以外はメモリアクセスに関連したリソースである．高性能なプロセッサでは，CPU コア内の処理よりメモリアクセス処理が性能制約となる傾向があるため，メモリアクセ

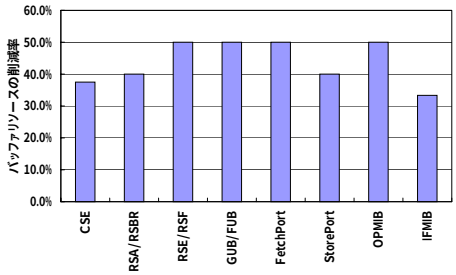

図 7 バッファリソースのエントリ数削減

ス処理を高性能化するリソースが必要となる．今回は高性能なサーバプロセッサをベースに性能の低いプロセッサを設計しているため，他のリソースに比べメモリアクセス処理に関するリソース削減率が大きくなったものと考えられる．

6.1.4 削減 4 の効果

BTAC を 4K エントリ 4way から 1K エントリ 2way に，メイン TLB を 1024 エントリから 512 エントリに減らした．これにより総合性能が 70.9% となりほぼ所望の結果が得られた．

個別にみると CINT2000 と CFP2000 は削減 3 とほとんど差がないが，TPC-C は 5.5% の性能低下が発生している．これは分岐予測ミス率の増加が原因である．図 8 に BTAC のコンフィギュレーションによる TPC-C の相対性能と分岐予測ミス率を示す．TPC-C では図に示すように BTAC サイズを削減することにより分岐予測ミス率が大きく増加する．これに従い性能も少しずつであるが低下する．

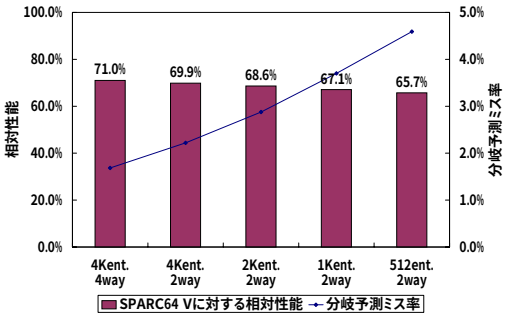

図 8 BTAC 削減による性能・分岐予測ミス率の変化 (TPC-C)

削減 2 と同様に BTAC がコア面積に占める割合が 1.1% と非常に小さいため 1K エントリ 2way とした．

図 5 に示すように総合性能が 70.9% に対し CPU コア面積が 54.5% となり．最終的な値は第 2 節で述べた目標をほぼ満たしていることがわかる．最終的なリソース値を表 2 に示す．

6.2 同一コア面積での SPARC64 V との比較

CPU コアは SPARC64 V の 54.5% でほぼ半分である．そこで CPU コア面積が等しいシステムを比較するため，1CPU の SPARC64 V と 2CPU コアによる CMP を比較した．CINT2000，CFP2000 は単体スレッドのベンチマークであるため，TPC-C のみを用いて性能比較を行った．結果を図 9 に示す．

表 2 CPU ハードウェア資源削減結果

	SPARC64 V	削減後コア
命令発行幅	4	2
機能ユニット	2 個ずつ	1 個ずつ
1 次キャッシュ	128KB	16KB
CSE	64	24
RSA/RSBR	10x1	4x1
RSE/RSF	8x2	8x1
GUB	32	16
FUB	32	16
Fetch port	16	8
Store port	10	4
OPMIB	4	2
IFMIB	3	1
BTAC	4K(4way)	1K(2way)
メイン TLB	1024	512

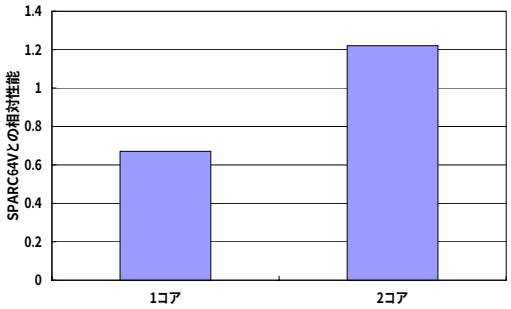

図 9 SPARC64 V との比較

この図より，1 コアの TPC-C の性能は SPARC64 V の 67% であるが，2 コアでは 22% 性能が向上している．ハードウェア資源削減後の CPU コアを用いることで，同一チップ面積で高スループットな CMP を構成できることがわかる．

7. 考　　察

7.1 Pollack のルールとの照合

各削減による CPU コア面積と Pollack ルールに基づき総合性能より期待されるコア面積 (=相対性能の自乗) をプロットしたものを図 10 に示す．

Pollack ルールと比較すると，チップ面積の傾向はほぼこのルールに従っているが，細かくみると削減 1 と削減 3 の後では Pollack ルールで期待されるチップ面積より 10%〜20% 程度大きくなっている．Pollack のルールは経験則なので厳密性はないが，もし Pollack

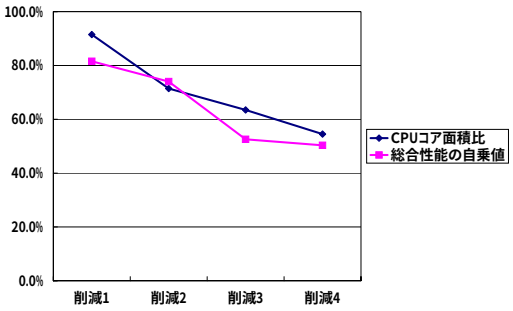

図 10 CPU コア資源削減によるコア面積変化

のルールが適正な性能とコア面積の関係を示していると仮定すれば，チップ面積が大きいという状況は性能に対してハードウェアリソースを過剰に投入していることを意味する．部分的なハードウェア資源削減により，一時的にアンバランスな CPU アーキテクチャになった結果を示しているのではないかと推測できる．

7.2 CMP 化について

本論文では主に CPU コアのハードウェア資源削減について述べてきた．今後，テクノロジの進歩に伴いより多くの CPU コアを 1 チップ上に搭載できるものと考えられる．このような場合に問題になるのは，CPU コア間で共有しているハードウェア資源で ある．共有しているハードウェア資源の代表的なものに 2 次キャッシュ，システムバス，メモリバンクがある．

2 次キャッシュについては，CPU コア数が増えることによるキャッシュミスの多発や，単位時間当りのアクセス頻度が増えることによるレイテンシの増大が予想される．またシステムバスのトラフィック増大，メモリバンクへのアクセス集中が発生する可能性も懸念される．これら問題点を定量的に検討し，現状のメモリシステムの改善点を明らかにする必要がある．

8. ま と め

同一チップ面積で高スループット性能を達成する CMP を構成するために，小型 CPU コアのアーキテクチャを検討した．この検討では既存のサーバ用プロセッサ SPARC64 V をベースにした．このアプローチは，設計期間短縮だけでなく高精度な性能評価シミュレータや実チップデータを利用できるといった利点がある．

SPARC64 V の半分の CPU コア面積で 70% の性能を達成するという目標のもとに，4 つのステップでハードウェアリソースを削減した結果，CPU コア面積を 54.5%に削減しながら，70.9% の性能を達成する CPU コアが実現できることがわかった．

さらにこれをベースに 2 コアの CMP を構成すると，SPARC64 V とほぼ同一チップ面積で TPC-C に対し +22% スループット向上が得られることを確認した．

今後は，テクノロジの進歩により多くの CPU コアが 1 チップに搭載されることが予想される．CMP コアを多数搭載した場合の問題点・解決策を明らかにしていく予定である．

参 考 文 献

1) L. Hammond et al, “The Stanford Hydra CMP”, IEEE MICRO, March-April 2000
2) L. Barrosso et al, “Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing”, 27th ISCA, June 1997.
3) “Session 1: PC & Server Processors” presentation notes of Microprocessor Forum 2003, Oct 2003.
4) F. Pollack, “New Microarchitecture Challenges in the Comming Generations of CMOS Process Technologies”, Micro32, 1999.
5) Inoue, ”Fujitsu’s New SPARC64 V for Mission-Critical Servers”, Microprocessor Forum 2002, Oct, 2002
6) H. Ando et. al, “A 1.3 GHz Fifth Generation SPARC64 Microprocessor”, in ISSCC Tech. Dig,. Feb. 2003,pp246-247
7) M. Sakamoto, A. Katsuno, A. Inoue, T. Aaskawa, H. Ueno, K. Morita, Y. Kimura, “Microarchitecture and Performance Analysis of a SPARC-V9 Microprocessor for Enterprise Server Systems”, Proc. of HPCA’03, Feb. 2003
8) B. Cmelik and D. Keppel, “Shade: A Fast Instruction-Set Simulator for Execution Profiling”, Proc. of the 1992 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, 1994.
9) 河場, 安里, “トレース長を考慮したクラスタリングによる評価用トレース選択”, コンピュータシステムシンポジウム論文集 p127-134,Nov.2002