Kaiser: 128-CPU SMP サーバの構成と評価

清水俊幸[†] 渡部徹[†]
小林健一[†] 石畑宏明[†]

4 CPU から 128 CPU までのスケーラビリティを持った Unix サーバ Kaiser を開発した．Unix サーバの主なアプリケーションである OLTP やデータベース処理の高速実行を目指し，高いメモリ性能を実現した．アプリケーションの処理速度の安定性，再現性を重視し，システム全体でのメモリアクセス時間を均質に提供できるスヌープベースの SMP アーキテクチャを採用した．大規模 SMP に求められる高いスヌープバンド幅やデータ転送バンド幅と，小規模構成からの拡張性を同時に満たすため，32 CPU 搭載可能なキャビネットをケーブルで接続するという構成を開発し，実装した．大規模システム実現で課題となる，資源競合の解消とアドレスアービトレーションのフェアネスを保証するために，バスプロトコルを新たに開発し，そのバスプロトコルの適正さと性能をシミュレーションにより検証した．実機により大規模データベースベンチマークを実施し，システムの高いスケーラビリティと小さいバスプロトコルオーバヘッドを確認した．

Kaiser: A 128-CPU SMP Server Design and Evaluation

Toshiyuki Shimizu,[†] Toru Watabe,[†] Kenichi Kobayashi[†]
and Hiroaki Ishihata[†]

We developed the largest SMP UNIX server called Kaiser, which scales up to 128 CPUs. Kaiser is designed to run database and OLTP applications that are the main applications of Unix servers. We selected a snoop-base SMP architecture, and optimized the memory access latency for fast execution of applications. Average memory access latency is around 300 ns. To realize two contradictory goals, high snoop rate and large data bandwidth, and expandability from a small configuration, we invented an expansion mechanism connecting 32-CPU cabinets by cables. To guarantee fairness of address arbitration, we developed a new retry based bus protocol and evaluated its characteristics using simulation. We evaluated the scalability of the system and protocol overhead of the retry based bus protocol by running the large-scale database benchmark.

1. はじめに

インターネットの普及にともなうネットワークビジネスの拡大により，トランザクション処理の大規模化が進んでいる．オペレーティングシステムを含む Unix サーバの高信頼化が進んだ結果，これまでメインフレームが担ってきたような大規模データベースアプリケーションへの適用もさかんである．この状況の下，Unix サーバは，さらなる性能向上を実現すること，およびメインフレームレベルの信頼性を備えることが急務となっている．主要アプリケーションの多くはマルチスレッドあるいはマルチプロセスで実行されるため，広大な共有メモリ空間を提供し CPU を多数接続することで高性能が実現される．

複数の CPU を接続する構成として，ディレクトリベースの ccNUMA や COMA などのアーキテクチャも一般的になっている[1),2)]．ccNUMA は小さなビルディングブロックから構成することができ，ワイヤ数やモジュールのサイズなどの物理的な制約が少ない．キャッシュコントローラやメモリコントローラを内蔵した CPU を使用すれば効率的にシステムを実現できる．

ccNUMA ではメモリアクセスレイテンシはローカルメモリの場合に良い．しかし，大規模なシステムでは相対的にリモートメモリの割合が高くなるため，平均的なアクセスレイテンシはリモートメモリの値に漸近する．ホームメモリへデータアクセスが集中する可能性やキャッシュ間データ転送を考慮すると，大規模 ccNUMA システムではスヌープバンドの飽和や three-party-transfer によるレイテンシの増加といっ

† 富士通株式会社
FUJITSU LIMITED

た性能バラツキが問題となる．ハードウェア的にこれらを解決するためには，ホームメモリのスヌープ能力の強化，キャッシュ間データ転送の高速化が必要となり，ビルディングブロックによる構成が可能というccNUMA の特長を生かせない☆1．

ハードウェア的な解決法に対し，ソフトウェア的にccNUMA の性能バラツキを吸収・回避する方法として，頻繁にアクセスするデータをローカルメモリに割り当てるなどの制御をするオペレーティングシステム[3]，およびアプリケーションに関する研究もさかんである．しかし，現時点でそれらの成果を一般的な利用環境で享受することはできない．アプリケーション開発者がメモリ利用を意識し，専用にプログラミングを行えば，高い性能を発揮することが可能である[4]が，汎用アプリケーションに対して，これらが適用されるのには長い時間が必要である．

我々は，ccNUMA の性能バラツキをハードウェア的に解決することや，OS やアプリケーションの作りにより制御することはまだ難しいと考えた．そして，1）流通ソフトウェアに対してバイナリコンパチビリティを持つこと，2）ベンチマークでの性能値の比較が他のアプリケーションを実行した場合の比較においても大きく異ならないこと，3）アプリケーションプログラマやシステムインテグレータが特別な配慮なしに性能チューニングできること，といった汎用性を大規模システムで満たすには，スヌープベースの SMP 構成が適当と考えた．大規模構成までスケーラブルな性能向上を実現するため，Kaiser ではスヌープベースの SMP アーキテクチャを採用した．

本稿では，大規模な SMP 実現における課題を整理し Kaiser での実現方法を示す．特に大規模システムで問題となるアービトレーションを効率良く行うために開発したバスプロトコルを中心に議論する．以降の章では，2 章で Kaiser のデザインコンセプトについて述べた後，3 章で Kaiser のアーキテクチャについて紹介する．4 章で大規模システムを実現するために開発したバスプロトコルについて議論した後，5 章でその性質および性能についてシミュレーションで検証する．6 章で性能評価を行い，最後にまとめを行う．

2. Kaiser デザインコンセプト

メモリを含むハードウェアコスト，設置面積，OS やアプリケーションのスケーラビリティなどから，最大構成を 128 CPU と決定し，この規模まで安定した性能を実現できるスヌープベースの SMP の要件を検討した．我々は，Kaiser で解決すべき課題として以下の目標を掲げた．

- 低く均一なメモリアクセスレイテンシ
  高いキャッシュヒット率を期待できない OLTP などのアプリケーションの性能は，メモリアクセスレイテンシで決定される．大規模システム，特にスヌープベースの SMP は，その基本制御の方式からワイヤ数の増大やそれにともなうモジュール（たとえばバックプレーン）の巨大化[5]により，メモリアクセスレイテンシが悪化する傾向にある．この関係を断ち切り，メモリアクセスレイテンシをすべての構成で高性能ワークステーション並みの 300 ns 程度に抑える☆2．
- 高いスヌープバンド幅とデータバンド幅
  128 個の CPU，およびそれに似合う I/O バスからのアクセスを処理できるバンド幅を持たなければならない．OLTP などの代表的なアプリケーションのキャッシュミス率に対して十分なデータ供給能力を提供するために，1 CPU に対して最大構成時でも 400 MB スヌープ/秒☆3を確保する．データバンド幅も CPU 数に応じたバンド幅☆4を実現する．
- 柔軟な拡張性
  小規模システムからの拡張性を実現するため，モジュラリティを持った設計が必要である．ビルディングブロックの組合せによる拡張を可能とする．
- フェアネス，デッドロックフリーなバス
  多くのリクエストをフェアにサービスし，しかもデッドロックを起こさないことが重要である．128 CPU の規模でこれを実現するのは，制御を分散管理したり，1 CPU あたりに費やす物量を少なく抑えたりといった，小規模な構成にない工夫が必要になる．少ない物量で物理的に大きく離れたモジュールを適正に制御するプロトコルを実現する．

3. Kaiser アーキテクチャ

低レイテンシ，高バンド幅の実現

高い周波数での動作や，低いレイテンシを実現するためには，物理サイズを小さく抑えることが必要である．しかし，十分なメモリや I/O の搭載を可能としたり，高周波数動作による発熱の廃熱対策が必要になっ

☆1 高いスヌープバンド幅を外部に提供可能なキャッシュコントローラなどが必要．

☆2 StarFire では 600 ns，Origin2000 ではローカルアクセスは 300 ns だが，128 CPU 構成では 1 μs 程度である．

☆3 秒あたりのスヌープ可能回数 × キャッシュラインサイズ 64 B．

☆4 CPU 数 × 400 MB/s．

たりすることで，物理サイズの縮小には限界がある．

SMP では基本的にアドレストランザクションを同時刻に配信する必要から，概念的には，任意の CPU☆から CPU までの信号伝送時間を等しくする必要がある．いい換えれば，最も遠い CPU から CPU までの距離がシステム全体の性能を決定する．

これまで，SMP はこれらの要件を満たすために 1 つのキャビネット（バックプレーン）によって構成されてきた．この制約からも，64 CPU を超えるスヌープベースの SMP システムは実現されることがなかった．またクロック周波数を高くすることも困難であった．

我々は，CPU と CPU の間の中継機構（クロスバと呼ぶ）を物理的に CPU と CPU の中間に配置し，アドレスのアービトレーションや，キャッシュ状態のマージ，データのルーティングなどを行うことにした．

CPU とクロスバの距離は，信号伝送の品質やタイミングを考慮した現実的な距離とするために，32 CPU を 1 つのバックプレーンに実装することにした．高いクロック周波数（200 MHz）を適用するとともに，クロッキング回数を抑え，長距離配線部では wave-pipeline 伝送を採用した．

柔軟な拡張性の実現

32 CPU を超える構成では，同軸ケーブルによってクロスバを相互に接続することで，拡張によるレイテンシの増加を最小限に抑えることにした．一般的なプリント板の信号遅延は 7.3 ns/m であるのに対し，同軸ケーブルの信号遅延は 3.7 ns/m であり，拡張によるペナルティを小さく抑えることができる．

フェアネス，デッドロックフリーバスの実現

大規模システムでは，アドレスやバッファの競合が高い多重度で発生する危険がある．仮にすべての CPU が同じデータを連続してアクセスすると，そのアドレスに対し CPU 数分のアクセスが集中する．このリクエストをバッファして保留すると，バッファによるハードウェア量が増加するばかりでなく，別のバッファに保留されたリクエストとの制御依存関係を生む可能性がある．これらを回避する設計は容易ではなく，また回路規模の増大，ディレイの増加を招く．

我々は，リトライベースのバスプロトコルを新たに開発し，ハードウェア量を抑えるとともにフェアネス，デッドロックフリーの性質を満足させた．

3.1 全体構成

最大構成の 128 CPU システムは，**図 1** に示すように 32 CPU 搭載可能なキャビネット（ノードと呼ぶ）を 4 つケーブルによって接続することで実現される．全システムは 1 つのクロックソースによって同期的に動作する．ケーブル長は 2.1 m あるいは 2.5 m であり，wave-pipeline 伝送により信号が伝送される．

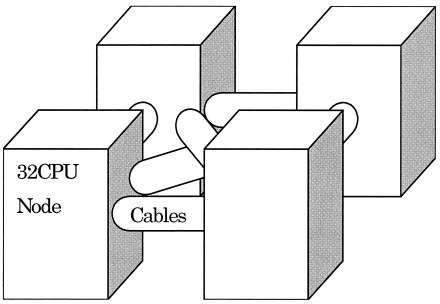

図 1 Kaiser 全体構成
Fig. 1 System configuration of Kaiser.

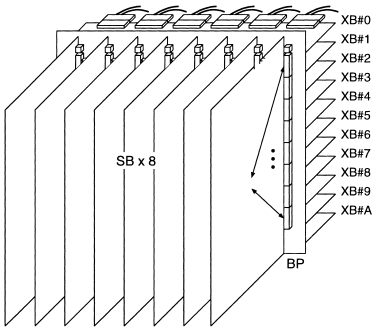

図 2 ノード内構成
Fig. 2 Node configuration.

3.2 ノード

図 2 に示すような縦横方向に実装方向をクロスさせた両面バックプレーン（BP）により，CPU，メモリ，I/O を搭載したシステムボード（SB）と，SB どうしおよびノード間を接続するクロスバボード（XB）を実装する．SB はノードあたり最大 8 枚搭載可能である．縦横方向に実装をクロスさせることによって，BP 配線は SB，XB の BP コネクタを跨ぐことなく配線できるため，大量の信号線を短い線長で配線できる．

3.3 Kaiser BUS（KBUS）

我々は SPARC64-GP[6] が採用するシステムバス（UPA バス[7]）を 128 CPU の構成に対応可能なレベルに拡張したインバリデート型のバス（KBUS）を定義した．KBUS の特徴は以下のとおりである．

- シンプルなフロー制御

  リトライを基本にしたシンプルなフロー制御プロトコルを開発し採用した．本フロー制御はバス利用効率が高く，ハードウェア量も少ない．

☆ 2 次キャッシュのタグのコピー（Dtag）をメモリの近くに置くことで距離の短縮も行われる．この場合は Dtag が CPU に相当する．

- 43 ビットアドレス空間の提供

物理アドレス空間を 2 ビット拡張することで物理アドレス空間を 8 TB に拡張した．物理メモリ空間は 4 TB である．

- 4 way アドレスバス

アドレススヌープバンド幅の確保のため，アドレスバスを 4 way 化し，システムの合計最大スヌープバンド幅を 51.2 GB スヌープ/秒とした．

- 分散アービトレーション

複数ノードを接続するという大規模化によっても性能劣化を引き起こさないように，ノード内でのアドレスアービトレーションを実現した．具体的には，way 単位に同時にアービトレーションに参加できるノードを分散させた．

- データ制御・データ転送バス

制御情報をデータに先行して，並行に送り出すことにより，データ受信側での処理遅延を 0 としている．

- インバリデート完了通知パス

物量の増加を抑えるため，インバリデート完了の回収にはデータ制御バスを利用した．

3.4 バス信号定義

KBUS の主な信号接続関係を**図 3** に示す．アドレス情報バス A，キャッシュ状態バス S，データ制御バス C，データバス D から構成される．それぞれ単方向の信号で XB を介して 1 対 1 接続される．

- アドレスバス A

SB から 40 ビット 2 クロック分（それぞれ AH，AL と呼ぶ）のアドレスパケットが出力されると（図 3 ①），XB ではそのアドレス way に応じて，アドレスを AO0，AO1，AO2，AO3 のいずれかにブロードキャストする（図 3 ②）．AOx は 80 ビットの幅を持つが，上位 40 ビットに AH が，その 1 クロック後に下位 40 ビットに AL が送り出される．XB には 6 パケット分のバッファがあり，アービトレーションで待ちが発生した場合にはバッファに格納される．各々のノードには 1 クロックに必ず 1 way（2 ノード以下の場合 2 way）にアドレスアービトレーションする権利が与えられており，ノード内の 8 SB からのアドレスパケットをアービトレーションする．アービトレーションに勝ったパケットは全 SB に対し同一時刻にブロードキャストされる．複数ノード構成の場合においては，自ノードへのブロードキャストは，この時刻合わせのために，ノード間伝送にかかるクロック分遅延させる（**図 4**）．

- キャッシュ状態バス：S

キャッシュの状態は SB が持つ Dtag（2 次キャッシュタグのコピー）を AO でスヌープすることで取り出される．Dtag からの way ごとのステータスが SB から出力される（図 3 ③）．全 SB からのステータスはマージされ SB に戻される（図 3 ④）．ステータスには Dtag の状態のマージ結果と Share 状態にある CPU を含む SB の数が示されている．

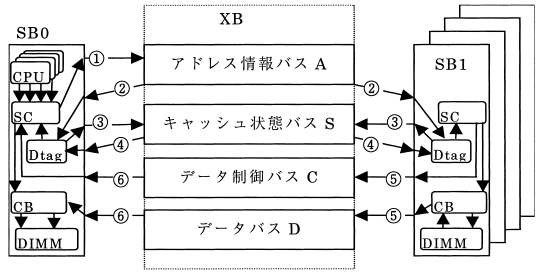

図 3 KBUS 信号接続概要
Fig. 3 Overview of KBUS operation.

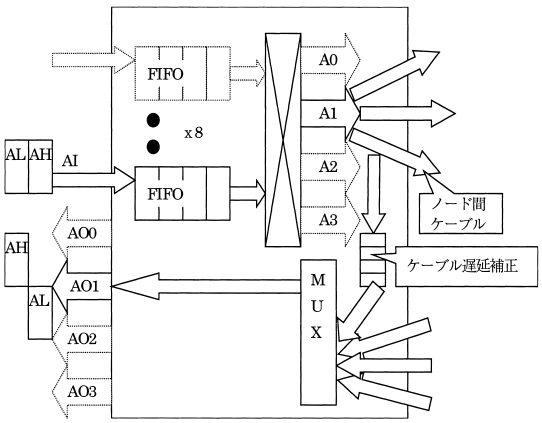

図 4 アドレスブロードキャスト
Fig. 4 Address broadcast control.

- データ制御バス：C

データ転送時に 4 クロック先行してデータ転送制御のための情報を 2 クロック長のパケットで転送する（図 3 ⑤，⑥）．これにより，データ転送に先立ち受信 SB でのデータスイッチの制御の準備を行い，クロッククロスのないデータ転送を可能とする．**図 5** に示すように，時刻 0（⓪）でシステムコントローラ（SC）はデータ制御パケットを送出すると同時にデータスイッチ（CB）へ制御情報を送る，CB での制御に 4 クロックを要するため（④），データパケットは 4 クロック遅延する．受信 SB でも SC から CB への制御は 4 クロック遅延するが，データの到着に一致する．

データ制御バスはデータバスと同様な構成で 32 対 32 のスイッチを実現している．ルーティングにはパケットの先頭のヘッダを用いる．データ転送をともなわないパケット（リプライと呼ぶ）を定義し，割込みや

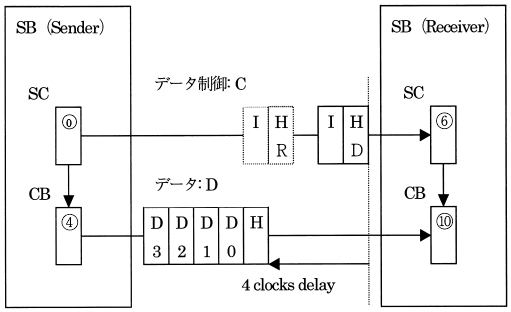

図 5 データ転送タイミング
Fig. 5 Data transfer control.

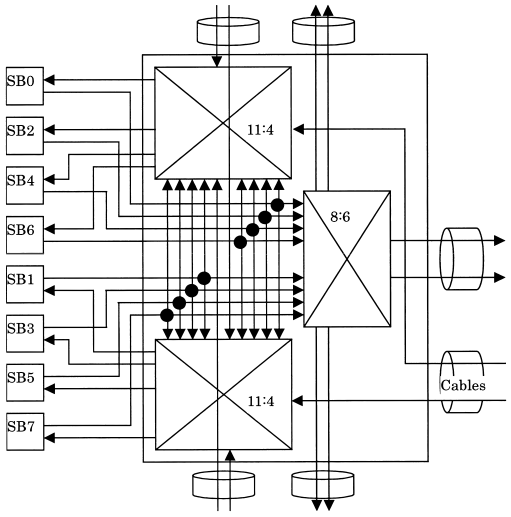

図 6 データクロスバの構成
Fig. 6 Blockdiagram of the data crossbar.

キャッシュの無効化完了の通知などに利用する．リプライ（HR）は，データ転送との同期をとるために，データ転送のヘッダ（HD）が入らないタイムスロット，あるいは，データ転送がないスロットを使用して転送される．

- データバス：D

144 ビットからなるデータ転送バスで，ヘッダを含め 5 クロック分のパケットからなるデータパケットを伝送する．**図 6** に示すように 8 対 6 と 11 対 4 のクロスバを組み合わせ，32 対 32 のスイッチを実現している．クロスバスイッチは受信部にのみバッファを持つ．フロー制御はクレジットベースで最小通過レイテンシは 1 クロックである．

4. バスプロトコル

大規模システムのバスプロトコルとして最も重要なプロパティは，フェアネス，デッドロックフリー，ライブロックフリーである．KBUS では，以下のようなプロトコルを実装することで，これらの要件を少ない物量で満足させるとともに，最悪レイテンシを保証している．

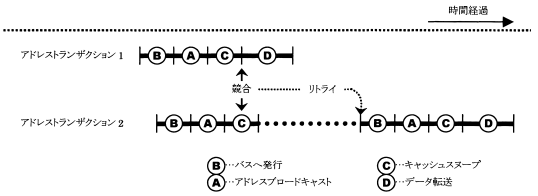

図 7 リトライプロトコル
Fig. 7 Retry protocol.

4.1 デッドロック防止のリトライ

アドレストランザクション処理においては，アドレスコンフリクトやスヌープアドレスに対応するタグ（Dtag）のビジーなどの資源競合や制御コントローラやバッファの資源不足により，スヌープが一時的に行えない事態が発生しうる．このような場合には，当該トランザクションをそこでいったん失敗させ，後続のトランザクションが滞留しないようにする．

トランザクションを滞留させず，順序依存関係を発生させないようにしているため，デッドロックの原因が発生しない．また，バスパイプラインが乱されない．多量の滞留バッファが不要になるといったメリットもある．失敗したトランザクションは，しばらく待機した後（自粛 A）にアドレスアービトレーションの段階から再実行（リトライ）する（**図 7**）．

4.2 フェアネス保証

リトライを持つプロトコルでは，ある不運なアドレストランザクションが連続してリトライされることにより，システム許容時間（タイムアウト時間）を超えてもトランザクションが完了しないことが起こりうる．リトライの頻度の制御によってタイムアウトの確率を下げることはできるが，回避することはできない．

KBUS ではリトライレベルという優先度制御を取り入れ，トランザクション処理がタイムアウト時間内に収まるように制御することによって，トランザクション間のフェアネスを保証しライブロックを回避している．

通常のアドレストランザクションは，リトライレベル（RL）が 0 である．リトライ回数が既定の閾値を超えたトランザクションは，処理優先度を上げるために RL を 1 にする．RL = 1 のトランザクションは，競合する可能性のあるすべての RL = 0 のトランザクションに優先して処理される．

具体的には，リトライによって RL = 1 のトランザクション（**図 8** の左端の●）がシステム上に現れた

ら，そのリトライ原因のキャッシュラインは「強制リトライモード」と呼ぶモードになる．このモードではRL = 1 のトランザクションのみを成功させ，RL = 0 のトランザクションは失敗させる☆．この失敗させられたトランザクションは強制リトライモードが終了するまで待つべく，通常のリトライよりも長い期間リトライを保留する（自粛 B）．RL = 1 のトランザクションは自粛 A の間隔でリトライさせる．強制リトライモードの間は新たな RL = 1 のトランザクションを発生させないことで，有限時間内ですべての RL = 1 のトランザクションが完了することを保証する．

システムから当該キャッシュラインの RL = 1 のトランザクションがすべて消滅したとき☆☆に，このキャッシュラインに対する強制リトライモードは終了し，自粛 B のタイマがクリアされ，ペンディングになっている RL = 0 のトランザクションはリトライされる．自粛 B の期間は長いほど，RL = 1 のトランザクションの成功率は高くなり，強制リトライモードの完了が早まる．自粛 A の期間の 6 倍程度にすると完了時間は十分短くなる．RL = 0 のトランザクションを含めた平均アクセスレイテンシを低く抑えるために 6 倍としている．

リトライレベルの管理はキャッシュライン単位で行うため，他のキャッシュラインのトランザクションは影響されず，強制リトライモードによる影響は極小化されている．

4.3 プロトコルの実装

デッドロック防止，フェアネス保証を「リトライ」をベースに設計し，リソースの共通化を実現した．アクセスのペンディング状態を分散管理し，全体のマージされた状態を通知することで，大規模システムでの物量の爆発的増加を抑える．具体的には，競合・枯渇状態を検出したターゲット（タグ，制御コントローラ）の状態と，RL = 1 のトランザクションを持つ CPU の状態を，タグのスヌープ結果に代えて，キャッシュ状態バス（S）に応答することにより，システム全体にアクセスのペンディング状態（失敗や強制リトライモード）を通知する．プロトコルを実現する管理資源は分散配置され，物量のオーダは CPU 数程度に抑えられる．

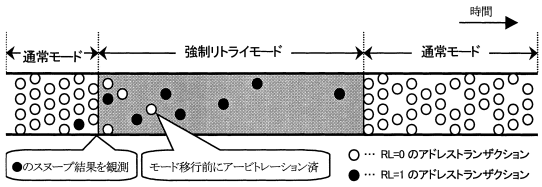

図 8 強制リトライモード（同一キャッシュライン）
Fig. 8 High-priority retry mode operation at the same cache line.

☆ RL = 1 のトランザクションをリトライ中の CPU を持つ SB は，RL = 0 のトランザクションのスヌープに対してキャッシュ状態の応答で「強制リトライモード」を通知する．

☆☆ 自粛 B 満了によるリトライ，あるいは新規アクセスにより，当該キャッシュラインの RL = 0 のトランザクションが成功した場合．

5. KBUS 性能検証

KBUS のバスとしての仕様の適正さを検証するために，机上での検証に加え，シミュレータによる性能検証および形式的検証ツール（モデルチェッキングツール）による仕様検証を実施した．性能検証については，アドレスバスのフェアネス，効率，および負荷の偏りによる性能変化を評価した．形式的検証に関しては作成したアドレスバスモデルによる不具合は検出されなかった．

5.1 BUS モデル

合成可能な HDL でモデルを構築することにより，モデルのみによる検証に加え，HDL で設計された LSI を組み合わせた検証を可能とした．さらに，モデル全体をハードウェアエミュレータによって実行することで，高速な性能評価を実現した．

モデル化は，前記目標を実現するために，アドレスバスインタフェースを実機と同じ構成とし，メモリ（SLAVE）数を抑え，記述の詳細度を低くすることで，**表 1** に示すように，最大構成までのモデルを現実的な規模で実現した．SLAVE あたりのアドレス範囲は 10 ビット（1024 キャッシュブロック）に制限している．メモリ数を制限することでアドレスコンフリクトが現実よりも多く発生することになり，検証は加速される．SB には CPU 用に 4 本，I/O 用に 2 本の UPA バスがある．評価ではこれらを区別せずに合わせて扱う．

パケットの送出から処理完了までの時間が最速ケースで 10〜26 クロック（平均 18 クロック）となるようにモデルを作成した．このうち SLAVE がビジーとなる期間は 15 クロックである．

表 1 バスモデル概要
Table 1 Bus model parameter.

モデル	K32-1	K32-2	K32-4	K64	K128
SB 数	8	8	8	16	32
CPU 数	32	32	32	64	128
I/O 数	16	16	16	32	64
SLAVE 数	1	2	4	2	1

表 2 発行パケット数
Table 2 Issued address packets.

モデル	K32-1	K32-2	K32-4	K64	K128
平均発行パケット数（k）	580	626	643	328	147
標準偏差	496	420	271	167	232
平均リトライ回数（k）	79.4	43.5	29.5	16.4	20.6
標準偏差	259	224	148	112	121
リトライ率（%）	12.0	6.5	4.4	4.8	12.3
バス使用率（%）	79.2	80.3	80.7	82.7	80.5
有効スヌープ数（%）	69.6	75.1	77.1	78.8	70.6
平均自粛回数	2320	628	255	1060	3910

5.2 均一分散

すべての UPA バスマスタ（CPU および I/O）が任意のアドレスのトランザクションを最大スループットで生成する場合について評価した．モデルごとの発行パケット数およびリトライ率の評価結果を**表 2** に示す．以降表中で（k）は 1000 回を示す．シミュレーションは 10 M クロック実施している．4 way にアドレスが分散するため，最大スヌープ回数は 40 M 回である．

平均発行パケット数は，各 UPA バスマスタから発行されたアドレスパケットのうちで成功したパケット数の平均であり，リトライパケットを含まない．アドレス範囲を 10 ビットに制限していることで，アドレスコンフリクトによるリトライが発生している．SLAVE 数が増えアドレス範囲が広がるとリトライ率，自粛回数ともに低下する（K32-1，2，3）．

アドレスバス使用率は約 80%と，自粛から復帰するまでの間パケット発行が行われないことを考慮すると，非常に高い効率を示している．有効スヌープ率は，データアクセスに利用されたアドレススヌープの率（リトライを含まない総発行パケット数/最大スヌープ回数）である．同一 SLAVE 数で比較すれば UPA バスマスタ数が増加しても有効スヌープ率が低下しないことから，資源が有効に利用されているといえる（K32-1 と K128，K32-2 と K64）．

5.3 アドレス集中

特定アドレスにアクセスが集中した場合の評価結果を**表 3** に示す．1 つのアドレス（way）にアクセスが集中するため，最大スヌープ回数は 10 M 回である．データアクセス中のスヌープはリトライされることから，SLAVE ビジー時間（モデルでは平均 15 クロック）を考慮した実効スヌープ率（リトライを含まない総発行パケット数 × SLAVE ビジー期間/10 M）は，70%程度と非常に高いことが分かる．

小さな規模のモデルではバス使用率が低く，リトライ率も低いのは，自粛期間が長すぎると判断できる．

表 3 アドレス集中時の効率
Table 3 Utilization of address bus at hot spot.

モデル	K32	K64	K128
平均発行パケット数（k）	9.75	5.06	2.55
標準偏差	128	70.5	6.5
平均リトライ回数（k）	106	69.4	41.6
標準偏差	12300	4110	942
リトライ率（%）	91.6	93.2	94.2
バス使用率（%）	55.6	71.5	84.7
有効スヌープ率（%）	4.7	4.9	4.9
実行スヌープ率（%）	70.2	72.9	73.4
平均自粛回数（k）	27.4	14.7	7.5
最大待ちクロック（k）	8.1	11.9	23.4
標準偏差	314	98.5	108
平均待ちクロック（k）	1.03	1.98	3.92
理論平均待ちクロック（k）	0.7	1.4	2.9

実効スヌープ率でも規模が大きなモデルの方が高い効率を示しており，システム規模に応じてパラメータを可変にすることが重要であることが分かる．

UPA バスマスタがトランザクションを発行してからデータを受け取るまでの待ち時間の最大値は UPA バスマスタ数に応じて増加する．UPA バスマスタ数による偏差は小さく，アービトレーションのフェアネスが実現されているといえる．また，平均待ち時間は，アービトレーションが隙間なく順番に行われた場合の理論平均待ち時間（= 10 M/UPA バスマスタ数/SLAVE ビジー期間）の 1.4 倍程度となっている．

6. 性能評価

6.1 メモリアクセスレイテンシ

図 9 に Kaiser の UPA バスからのアドレススヌープ開始から，完了までのタイミングを示す．1 マスは 1 システムクロック（200 MHz）に相当する．ラベルは，それぞれの信号を制御する LSI の略称（SC は SB のシステムコントローラ，A，S はそれぞれアドレス制御，キャッシュ状態マージを行うクロスバ LSI）や伝送媒体（BP，Cable はそれぞれ SB–XB 間，ケーブルの伝送部分）である．帯の下に示された記号，n，e，

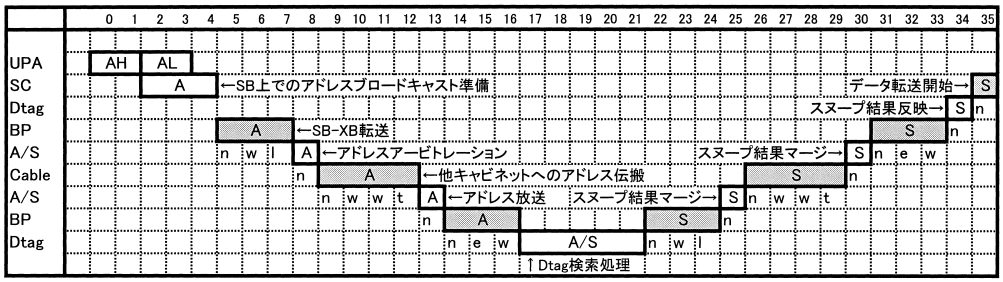

図 9 最大構成時のスヌープ開始から完了までのタイミング（最速ケース）
Fig. 9 Timing diagram of the address snoop operation (best case timing of the maximum configuration).

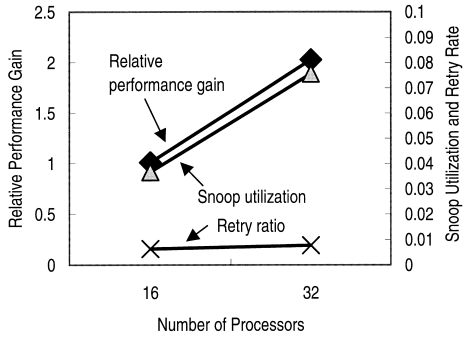

図 10 大規模データベースベンチマークによる CPU 数に対する相対性能，アドレスバス使用率，リトライ率
Fig. 10 Relative performance gain and address snoop utilization using a large-scale database benchmark.

l，w は，それぞれ，normal，early，late クロックおよび wave-pipeline 伝送である．t はケーブルディレイ調整のためのクロックである．クロック 0 で UPA にアドレス AH，AL が現れ，SB から BP，Cable 経由で全 SB の Dtag をアクセスし，そのスヌープ結果が，マージされ，全 SB に返される一連の動作が，最小のクロッキングで実現さていることが分かる．

6.2 性能スケーラビリティとリトライ率

図 10 に大規模データベースベンチマークプログラムを 16 CPU および 32 CPU 構成の実機を用いて走行させた場合の性能比，アドレススヌープバンドの利用率，およびリトライ率を示す．スヌープバンド利用率，リトライ率は Kaiser のクロスバが備えるパフォーマンスカウンタを用いて実測した．CPU 数の増加に比例して性能が向上していること，アドレスバスの使用率も増加していることが分かる．32 CPU の場合でもアドレスバス使用率は 7.6%であり，アドレススヌープのうち 0.8%がリトライされるにとどまっている．リトライ率の増加はわずかである．また強制リトライ（自粛 B）は発生していない．実アプリケーションにおいては導入したリトライ方式による性能ペナルティは無視できるほど小さく，単純で小さな回路規模で実現可能な本方式の有効性は高いといえる．

7. ま と め

128 CPU までの拡張性を持った Unix サーバ，Kaiser を開発した．大規模構成での汎用性を実現するためスヌープベースの SMP により実現した．

32 CPU までのバックプレーン接続の SMP をケーブルによって接続するというパッケージングの工夫により，物理サイズの制約を緩和した．性能のスケールアップとモジュラリティを同時に満たした．

大規模システムを考慮したアドレススヌープのプロトコルを開発した．巨大システムで懸念されるスヌープ効率の低下，フェアネスの低下，デッドロックの発生を，少ない物量で回避できるプロトコルであることを，シミュレーションによる評価と，実システムでの実アプリケーションを使った評価を通して示した．

平均メモリアクセスレイテンシは最大構成において 300 ns を実現しており，最速のワークステーションのメモリアクセス速度に迫る．高いメモリバンド幅の提供と大容量 2 次キャッシュメモリの採用により多くのアプリケーションの高速実行を可能とした．

大規模実アプリケーションを実機によって評価した結果，アドレスバス使用率やリトライ率が低く抑えられており，Kaiser が十分なバス性能を提供していることを示した．今後，さらに多くの大規模実アプリケーションについて評価を進めるとともに，より高い性能を実現するためのアーキテクチャの検討を進める．

謝辞 Kaiser システム成立のためにご尽力いただいた，西村幸介，河部本章両氏に深謝いたします．システム性能に対する諸検討の協力をしていただいた仲川明和，土手口正裕両氏，システムボード搭載の LSI の設計，およびシステム立ち上げ，評価の指揮をとっていただいた後藤裕一氏，伝送路解析およびプリント板デザインで，高精度なスキュー制御の実現手法の開発

と，その実装を行っていただいた船木淳氏，LSI の開発評価をすすめていただいた諸氏に感謝いたします．伝送路解析やテクノロジ選択に関してサポートいただいた森豊氏，古谷和弘氏，評価プログラムの開発指揮をとっていただいた指宿剛氏に感謝いたします．また，貴重なコメントをいただいた，査読者の方々ならびにプログラム委員の方々に深謝いたします．

参 考 文 献

1) Laudon, J. and Lenoski, D.: The SGI Origin: A ccNUMA highly scalable server, *Proc. 24th International Symposium on Computer Architecture*, pp.241–241 (1997).
2) Lovett, T. and Clapp, R.: STiNG: A CC-NUMA Computer System for the Commercial Marketplace, *Proc.23rd International Symposium on Computer Architecture*, pp.308–317 (1996).
3) Noordergraaf, L. and van der Pas, R.: Performance experiences on Sun's WildFire prototype, *Proc. Supercomputing '99* (1999).
4) Jiang, D. and Singh, J.P.: Scaling Application Performance on Cache-coherent Multiprocessors, *Proc. 26th International Symposium on Computer Architecture*, pp.308–317 (1999).
5) Charlesworth, A.: Starfire: Extending the SMP Envelope, *IEEE MICRO*, Vol.18, No.1, pp.39–49 (1998).
6) Song, P.: Hal Packs SPARC64 onto Single Chip, *Microprocessor Report*, Vol.11, No.16, pp.1–5 (1997).
7) Normoyle, K., Ebrahim, Z., VanLoo, B. and Nishtala, S.: The UltraSPARC Port Architecture, *Proc. Hot Interconnects Symposium III* (1995).

(平成 12 年 9 月 2 日受付)
(平成 13 年 2 月 1 日採録)

清水 俊幸（正会員）

1986 年東京工業大学工学部卒業，1988 年同大学大学院理工学研究科修士課程修了，同年（株）富士通研究所入社．並列計算機のアーキテクチャの研究に従事．現在，富士通（株）コンピュータ事業本部に勤務．コンピュータアーキテクチャ開発に従事．信学会会員．

渡部 徹

1985 年青山学院大学理工学部卒業，同年富士通（株）入社．UNIX サーバーのハードウェア開発に従事．現在，コンピュータ事業本部に勤務．UNIX サーバのハードウェア開発に従事．

小林 健一（正会員）

1992 年東京大学工学部計数工学科卒業，1994 年東京大学情報工学修士課程修了．同年（株）富士通研究所入社．並列処理系の研究に従事．現在，富士通（株）コンピュータ事業本部に勤務．計算機アーキテクチャ開発に従事．

石畑 宏明

1980 年早稲田大学理工学部卒業．同年（株）富士通研究所入社．画像処理システムの研究，並列コンピュータアーキテクチャの研究に従事．現在，富士通（株）コンピュータ事業本部に勤務．コンピュータアーキテクチャ開発に従事．1992 年元岡賞受賞，工学博士．信学会，IEEE 各会員．