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

2005－ARC－165（9）
2005／12／1

MiBenchの並列化およびオンチップマルチプロセッサの評価

住吉正人[†]　田辺靖貴[†]　天野英晴[†]

近年の組込み機器が行う処理は複雑かつ多様化しており，肥大化傾向にあるデータを扱える処理能力の高さも要求されている．さらに，省エネルギーへの要求も高く，これらを実現可能なアーキテクチャとしてオンチップマルチプロセッサが注目を浴びている．プロセッサの設計段階ではベンチマーク等による予備評価が有効であるが，特に組込み分野のマルチプロセッサを対象にしその並列処理能力を測定する目的で公開されているものがない．そこで，組込み分野のベンチマークである MiBench suite を並列化し，マルチプロセッサで動作可能にした．シミュレータを用いた評価例を示し，対象のマルチプロセッサシステムを組込み用途で使用した場合の性能評価が可能になったことを示す．

Parallel Benchmark Programs for the Evaluation of Embedded Chip Multi-Processors

Masato Sumiyoshi,[†] Yasuki Tanabe[†] and Hideharu Amano[†]

Today, even embedded systems are required to process complicated programs with large data with low energy consumption. Chip multi-processors (CMPs) have received an attention as one of the most promising candidates. Although benchmarking is useful for evaluating the performance of such CMPs, parallelized benchmarks targeting embedded CMPs have not been released yet. We parallelized several embedded programs from MiBench suite in order to build a new benchmark suite for embedded CMPs. Programs are parallelized in the manner that processor cores cooperate with each other to process a single job. By evaluating with a sample CMP architecture, we demonstrate that our parallel benchmark programs can be used for evaluation of embedded CMPs design.

1. はじめに

近年，オンチップマルチプロセッサ (Chip Multi-Processors: CMP) の組込み分野での利用が注目されており，半導体メーカー各社は組込み用途をターゲットとしたホモジーニアス CMP の開発に乗り出している．

ARM 社と NEC エレクトロニクスが連携して開発した MPCore[1] は ARMv6 のコア 1〜4 個をシングルチップに納めた CMP で，民生エレクトロニクス，車載機器，モバイル機器等の高性能・低消費電力製品をターゲットとしている．東芝はコンフィギュレーションや機能拡張が可能な複数のコアを 1 つのダイに集積したメディア処理用の CMP である MeP[2](Media embedded Processor) を開発し，DVD や携帯電話，車載機器等の製品に使用している．三菱電機とルネサステクノロジは，M32R 32bit RISC アーキテクチャのコア 2 つで構成される共有メモリ型の CMP を開発した[3]．これはデジタル情報家電，携帯情報機器をターゲットとしている．

ホモジーニアスな CMP は，複数のコアを並列動作させることにより高性能な処理を達成することができるため，個々のプロセッサコアの動作周波数を低く抑えることができる．この際，動作周波数と共に電源電圧を下げることにより，組込み分野における重要な要求である低消費電力化，省エネルギー化が可能である．また，専用ハードウェアやヘテロジーニアスなマルチプロセッサに比べて，プログラム開発環境が整備されており，並列化 OS や並列化コンパイラなどの利用も容易である．

現在，ホモジーニアスな CMP の利用は主として異なったジョブレベル，タスクレベルでの並列処理が中心であり，単一ジョブの並列処理における性能は良く調査されていない．しかし，ホモジーニアスな CMP では，単体ジョブを並列化して高速化を実現することもできるといった利点がある．そこで，組み込み用の単一ジョブを並列化した場合の性能について，きちんと評価する必要がある．

しかしながら，現在提供されている並列ベンチマークの多くは主に科学技術計算用のマルチプロセッサシステムやサーバ機などの計算能力の測定を目的としたものである．組込み分野におけるマイクロプロセッサの評価ベンチマークとしては，EEMBC[4] によるものや MiBench suite[5] 等がある．これらのベンチマーク集は組込みシステムを象徴するカテゴリ別に代表的なプログラムを提供しており，組込み分野の多様性を反映しているが，このままではマルチプロセッサによる協調動作の評価は行えない．

本研究は，組み込み用途における CMP の性能評価プログラムを提供することを目的とし，組込み向けベンチマークとして無料で提供されている MiBench suite に含まれるプログラムを並列化する．そして，実装したベンチマークを用いて組込み向け CMP を想定したシミュレーションモデルで動作させる．これにより，組込み分野で用いられるアプリケーションの単一ジョブを並列化して実行した際の挙動を把握できることを示す．

本稿では，まず 2 章で MiBench suite について述べ，3 章において並列化手法を取り上げる．次に 4 章で評価環境とシミュレータの説明を行い，5 章で並列化した MiBench を用いて評価する．最後に 6 章で結論および今後の課題を述べる．

2. MiBench suite

MiBench suite は EDN Embedded Microprocessor Benchmark Consortium (EEMBC) をモデルとしたベンチマーク集である．組込み分野で用いられるアプリケーションの

† 慶應義塾大学 理工学研究科
Department of Science and Technology, Keio University

多様性を反映し，35 の組込みプログラムを 6 カテゴリに分類し無料で提供している (表 1)．各プログラムのソースコードは標準の C 言語で記述されており，コンパイラサポートのあるプラットフォームであれば容易に使用することが可能である．本研究では，組込み用途における CMP の性能評価を目的とする利用を前提に，MiBench suite が提供するプログラムを並列化する．

並列化するベンチマークは，表 1 の中から各カテゴリにおいて代表的なものをデータアクセスパターンの違いや抽出できる並列性に着目して選択する．

以下，本章では MiBench suite の分類するカテゴリの特徴及び本研究で並列化したプログラムについて述べる．

2.1 Automotive and Industrial Control

組込み分野の制御システムにおいて使用される組込みプロセッサをターゲットとしている．この分野のプロセッサには，基本的な算術演算，ビット操作，データの入出力における性能が要求される．典型的なアプリケーションとしてはエアバッグコントローラ，エンジンパフォーマンスモニタ，センサーシステム等が挙げられる．本研究では，センサーシステム等において物体の形状認識に応用される SUSAN のエッジ検出プログラムを並列化する．

2.2 Consumer Devices

近年市場規模の拡大が著しい，スキャナ，デジタルカメラ，PDA 等，コンシューマ向けの様々なデバイスをターゲットとしている．この分野はマルチメディアアプリケーションに焦点を絞り，代表的なアルゴリズムとして JPEG 圧縮/伸長，画像のフォーマット変換，ディザリング，カラーパレットの減数，MP3 圧縮/伸長，HTML 植字を提供している．本研究では，現在標準的に用いられている画像圧縮フォーマットである JPEG の圧縮アルゴリズムを並列化する．

2.3 Network

スイッチやルータ等，ネットワークデバイスに用いられる組込みプロセッサをターゲットとする．これらのプロセッサに要求される最短経路解析，木構造やテーブル参照，データ入出力の性能を見るために，グラフの最短経路検索と Patricia Trie データ構造の構築/探索を行うアルゴリズムを提供している．本研究においては，Patricia Trie の構築および探索の並列処理を行う．Patricia Trie はネットワークアプリケーションにおけるルーティングテーブルとして使用されるデータ構造であり，コードの複雑さと引き換えに検索時間を短縮しているという特徴がある．

2.4 Security

インターネットを用いた e-commerce の拡大に伴い，セキュリティがますます重要になって来ている．これを反映させて，MiBench には独自のカテゴリとして Security が設けられている．この分野にはデータの暗号化/復号化，およびハッシュ生成アルゴリズムが含まれる．本研究では共通鍵のブロック暗号である Blowfish を用いた暗号化及び復号化を並列化する．Blowfish は 32 から 448 ビットまでの可変長の鍵を用いる 64 ビットブロック暗号化アルゴリズムで，暗号化モードとして ECB，CBC，CFB，OFB が選択可能である．

2.5 Office Automation

この分野のアプリケーションには主にテキスト操作アルゴリズムが用いられ，プリンタ，ファックス，ワープロ等を代表とするオフィス機器で使用されている．Consumer のカテゴリに含まれる PDA においてもテキスト操作は重要である．

2.6 Telecommunication

Telecommunication は音声符号化，周波数解析，チェックサム等の通信機能に関連したベンチマークである．現在，多くのポータブルなデバイスが無線通信機能を持つようになり，Consumer のカテゴリと並んで重要な分野となっている．

表 1 MiBench suite

Automotive and Industrial Control	Consumer Devices	Office Automation
basicmath	jpeg *	ghostscript
bitcount	lame	ispel
qsort	mad	rsynth
susan edges *	tiff2bwrgba	sphinx
susan corners	tiff2rgba	stringsearch
susan smoothing	tiff2dither	
	tiffmedian	
Networking	**Security**	**Telecommunications**
dijkstra	blowfish enc. *	CRC32
patricia *	blowfish dec.	FFT
(CRC32)	PGP sign	IFFT
(SHA)	PGP verify	ADPCM enc.
(blowfish)	rijndeal enc.	ADPCM dec.
	rijndeal dec.	GSM enc.
	SHA	GSM dec.

3. MiBench suite の並列化

MiBench suite の提供するベンチマークのうち表 1 に* で示す JPEG 圧縮，SUSAN Edges，Blowfish, Patricia のプログラムを並列化した．本章では，並列化の際のプログラミングモデルについて触れ，次に，並列化したプログラムの説明及びその手法を述べる．

3.1 並列化モデル

プログラムの並列化モデルは，MPI を利用するものと共有メモリを利用するものに大別できる．これらは，一般的に，システムのアーキテクチャや，その上で利用するアプリケーションの実装しやすさなどに応じて，適したものが選択される．今回は，ホモジーニアスな共有メモリ型アーキテクチャ上で単体ジョブを並列化させることを想定しており共有メモリを利用する方法で並列化した．

プロセッサ間の通信は共有メモリを介して LOAD 命令及び STORE 命令により行う．また，プロセッサ同士の待ち合わせは共有メモリ上に設けた同期変数を利用したバリア同期で行い，共有データに対する排他制御は TAS によるロックを用いる．

3.2 並列化手法

ここでは，各アプリケーションの動作について説明し，その並列化手法を述べる．

3.2.1 JPEG 圧縮

JPEG 圧縮は，フルカラー画像を対象とした非可逆の画像圧縮アルゴリズムである．入力画像に対して色変換，離散コサイン変換 (DCT)，量子化，エントロピー圧縮の処理を行うことで圧縮する．圧縮の最小単位は MCU (Minimum Coded Unit) と呼ばれ，一連の処理は MCU 毎に繰り返し行われる．

本研究では，JPEG 圧縮の処理を次の 3 つに分け，それぞれのステップに関して並列化を行った．

- STEP 1-3, 4 (初期化)
  圧縮パラメータの設定や量子化テーブルとハフマン符号化テーブルの生成を行う．生成されるパラメータやテーブルは以後の処理で書き換えられることはないが，圧縮の過程で頻繁に参照されるものである

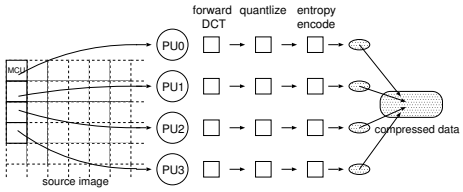

図 1 JPEG 圧縮の並列化

ので，各 PU 毎にローカルメモリ上に生成する事とした．また，初期化に必要なサイクル数はアプリケーション全体からみると微々たるものであるので，並列化による高速化を行わない事とした．

- STEP 5 (圧縮)
  DCT，量子化，ハフマン符号化を行う．本研究では図 1 のように各 PU に異なる MCU を割り当て，MCU 毎に同時に圧縮処理を進めることにより性能向上を図った．単純に並列化すると，DCT で得られたデータの DC 成分を量子化する際に，隣接する MCU の DC 成分との差分を取る必要から待ち合わせが必要となる．そこで，この依存関係による性能低下を避けるため，量子化の時点では DC 成分の差分を取らず，全ての MCU が圧縮された後でこれを解決することにより待ち合わせによるロスを少なくした．
- STEP 6 (データの収集)
  各 PU のローカルメモリ上で圧縮されたデータを，全 MCU の圧縮が完了した後に共有メモリ領域にコピーしてマージすることで，最終的な JPEG 圧縮画像データを出力する．

3.2.2 SUSAN Edges

SUSAN Edges[6] は画像の輪郭抽出を行う，エッジ検出アルゴリズムのプログラムである．このプログラムは，画像の各ピクセルに対して以下の処理を行うことでエッジを検出する．まず，注目するピクセルを囲むように円形のマスクを配置する．そのマスク内で中心の輝度と近いピクセル数を計算する．このピクセル数は USAN (Univalue Segment Assimilating Nucleus) と定義されている．閾値から USAN の大きさを減じるとエッジが強調された画像が得られ，さらにモーメントを算出しエッジ方向を検出し，非最大値抑制を行う．

本研究では，処理を次の 4 ステップ (STEP 1〜STEP 4) に分け，それぞれに関し並列化の検討を行った．

- STEP 1 (初期化)
  エッジ検出の準備段階で，共有領域の確保や USAN 値の算出に用いる重み付けの LUT 生成を行う．LUT の生成は乗算 6 回，除算及び指数関数の演算を 512 回のループで行う．このループ間にはデータ依存がなく容易に並列化可能である．しかし，このステップの実行に要するサイクル数はプログラム全体からみると少ないので並列化による高速化を行わない事とした．
- STEP 2 (エッジの強調)
  このステップで行う計算はマスクごとに独立して算出できることに着目し，入力画像データを PU 数で分割して並列処理を行うことによる性能向上を図った．共有メモリ上の原画像から各画素の輝度データを読み出し，USAN 値の算出やエッジ方向検出及びエッジ強調を行い，共有領域に書き込む．PU 間でのデータ依存関係は無いが，共有メモリからの読み出しと書き込みが頻繁に発生する．
- STEP 3 (エッジの補正)
  検出されたエッジに対して非最大値抑制を行い，エッジを補正する．このステップは隣接画素間のデータ依存関係により左上端画素から右下端画素まで逐次的に処理する必要があり，条件によっては既に処理済みのピクセルまで再帰的に戻る場合があるので，並列化を行わず単一 PU が処理する事とした．
- STEP 4 (原画像へ重ね合わせ)
  共有メモリからエッジ画像データを読み出し，それを共有メモリ上にある原画像データ上に書き込むことでエッジ画像を原画像に重ね合わせる．このステップは STEP 2 と同様の方法で並列した．

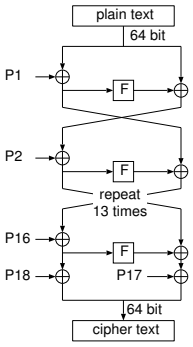

図 2 Blowfish 暗号化

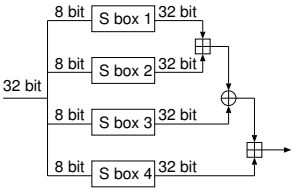

図 3 関数 F

3.2.3 Blowfish encrypt

Blowfish[7] は鍵拡張とデータ暗号化という 2 つの部分からなる．はじめに暗号化に用いるサブ鍵を鍵拡張により生成し，次にブロック単位の平文を繰り返し暗号化する．

本研究では，ECB モードを用いた Blowfish の暗号化を並列化する．暗号化モードとして ECB を用いた場合，平文の各ブロックが独立して暗号化されるので比較的容易に並列処理が行える．そこで，本研究では ECB モードを用いた Blowfish の暗号化処理を次のようにステップに分け，並列化を検討した．

- STEP 1 (鍵拡張)
  鍵拡張は Blowfish のアルゴリズムを利用して行われる．必要なサブ鍵を全て生成するためには合計 521 回転するが，サブ鍵の生成は直前に生成したサブ鍵を用いて行う必要があり，並列化による実行時間の短縮が見込めないので並列化しない事とした．また，生成した鍵は暗号化時に頻繁に読み出されるので，高速にアクセス可能なローカルデータとして各 PU に持たせる事とした．
- STEP 2 (暗号化)
  データ暗号化は図 2 のように 16 回転する関数からなる．関数 F は図 3 のように入力データを 4 分割し，S ボックスを参照しその出力に対して OR や XOR をとる．暗号化の各ラウンドは鍵依存の転置 1 つと鍵とデータの両方に依存する置換で構成される．共有メモリに置かれた平文に対して，前もって鍵拡張で生成しておいたサブ鍵の配列を用いる．本研究では，ECB モードを用いた暗号化を同時に複数のブロックに対して動作させ並列化した．
- STEP 3 (データの収集)
  各 PU がローカルメモリに生成した暗号データを共有メモリ上に集めて暗号文を完成させる．

3.2.4 Patricia

Patricia Trie[8] は 4.3 Reno リリース以降の BSD カーネルでルーティングテーブル検索用に用いられているデータ構造で，Radix Tree の内部の不要な中間ノードを削除

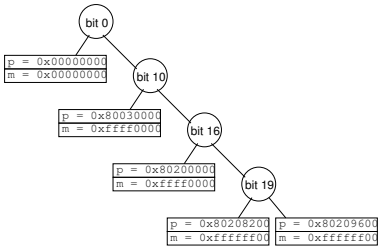

図 4 Patricia Trie のデータ構造

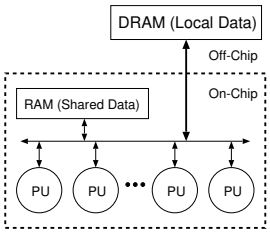

図 5 バス結合型 CMP

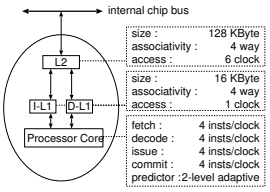

図 6 PU の構成

し検索回数及びメモリ使用量を改善したものである．図 4 に Patricia Trie の内部データ構造の例を示す．IP パケットの宛先を検索キーとしルートから中間ノードを辿ることで検索を行うと，下流にマッチするノードを持たないノードに到着する．検索したノードのうち，マッチするネットワーク部の長さが最長の中間ノードが保持する経路情報を，該当アドレスへの経路情報として用いる．

ベンチマークでは，キーとする IP アドレスを次々に与えて Patricia Trie のデータ構造を検索する．もし，検索の結果該当する宛先に関するルーティング情報の登録が無いことがわかれば，新たにノードを追加する．

本研究では 1 つの検索を 1 つの PU に割り当て，同時に複数の検索処理を行うことで高速化した．検索を行う木構造は共有メモリ上に構築し，ノードの追加の際に木構造へのアクセスを一時的にロックする時以外は全 PU が同時に木構造へアクセスすることができる．

4. 評価環境

並列化したベンチマークを動作させる評価環境として，ISIS-SimpleScalar[9] を利用しバス結合型マルチプロセッサシステムのシミュレータを構築した．本章では ISIS-SimpleScalar およびこれを用いて構築したシミュレーション環境について述べる．

4.1 ISIS-SimpleScalar

ISIS-SimpleScalar は，主として並列計算機をターゲットとした計算機シミュレータのための C++言語によるクラスライブラリである ISIS に，SimpleScalar が提供する out-of-order 実行，分岐予測などをサポートした sim-outorder モデルを組み込み，マルチプロセッサシステムに対応させたシミュレーション環境である．C++を採用したことによりシステム全体をクラス単位で効率良く管理でき，しかも C 言語と同様に高速に動くプログラムが容易に作成可能である．ISIS では計算機内部のプロセッサやメモリなどの機能ブロックがユニットと呼ばれるクラスライブラリとして実装されている．

4.2 ターゲットアーキテクチャ

バス結合型は最も基本的なマルチプロセッサアーキテクチャであり，現在開発されている組込み CMP の多くもバス型を採用している．そこで，今回の評価でも図 5 のようなバス結合型の CMP をモデルとして，ISIS-SimpleScalar を用いてシミュレータを構築し，その上で，並列化した各ベンチマークを動作させた．

各 PU には SimpleScalar の sim-outorder モデルを組み込みマルチプロセッサシステムのシミュレーションに対応させたモデルを用い，図 6 のような構成となっている．ローカルデータは各コア毎の L1，L2 キャッシュでキャッシュされ，L2 でミスした場合には，オフチップのメモリにアクセスされる．ローカルデータへのアクセスレイテンシは，L1 に Hit した場合は 1 cycle, L2 で Hit すると 6 cycle, L2 でミスしオフチップのメモリにアクセスする場合には 18 cycle とした．L1，L2 キャッシュのサイズはそれぞれ，128 Byte，1024 Byte である．

表 2 入力データサイズ

Application	Format	Size
JPEG encode	8bit Color	100 x 100 pixel
SUSAN Edges	8bit Gray	76 x 95 pixel
Blowfish encrypt	ascii text	304 KB
Patricia Trie	IP address	10890 keys

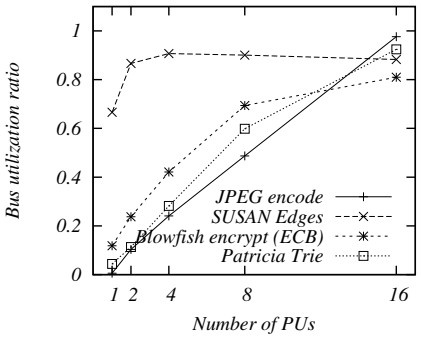

図 7 バス利用率の変化

共有データ用の領域としては，各コアからバスを介してアクセスされる共通のオンチップ RAM 領域を利用する事とし，本稿の評価では，評価条件を簡略化するため，共有データはすべて，この RAM 領域に格納できる事とした．この共有データは各コアから，最短 6 cycle でアクセスされる事とし，バスにて衝突が発生した場合等は，それを考慮して，より多いレイテンシとなる．

5. 評価

本章では，各アプリケーションをバス結合型マルチプロセッサで動作させた際の性能評価を行い，ターゲットアーキテクチャの問題点を明らかにする．入力データのサイズを表 2 に示す．評価は入力データが共有メモリにセットされた状態から開始した．

5.1 バス利用率と台数効果

図 5 に示したシステムにおいて，PU 数を 1，2，4，8，16 と変化させ，そのそれぞれにおいて表 1 の中から今回並列化を行った 4 つのアプリケーションを実行した．PU 数を増やすことによって共有バス利用率がどの程度変化したかを図 7 に，実行クロック数での性能向上がどの程度得られたかを図 8 に示す．図 7 縦軸のバス利用率は，バスの場合複数のアクセスを同時に処理することができないため，その値が 1 を越えることはない．

JPEG 圧縮と，SUSAN Edges では，コア数を増して実行しても性能向上が得難く，特に，SUSAN Edges では，少ない PU 数でもバスへの負荷が高くなっており，これ

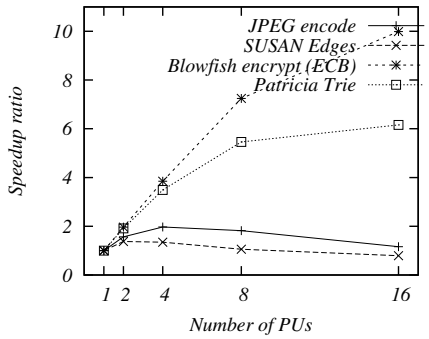

図 8 PU 数の違いによる実行クロック数の変化

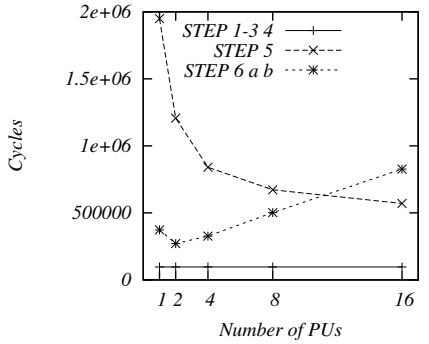

図 9 PU 数の違いによる実行クロック数の変化 (JPEG 圧縮)

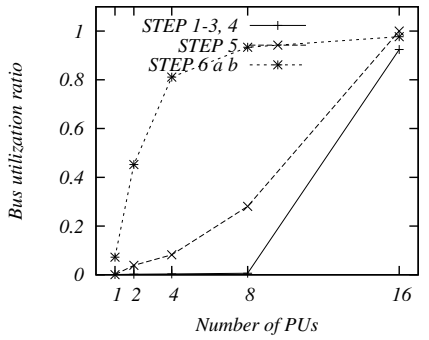

図 10 PU 数の違いによる共有メモリバス利用率の変化 (JPEG 圧縮)

が性能向上を妨げる要因となっている事がわかる．また，Blowfish encrypt と，Patricia Trie は，コア数に応じた性能向上が比較的得やすく，バスへの負荷もコア数に応じて増えている事がわかる．

5.2 JPEG encode

JPEG 圧縮は PU 数 4 までは台数を増す毎に高速化することができたが，それ以上の台数ではかえって遅くなってしまう．この原因を探るため，図 9 と図 10 とを用意した．図 9 では，JPEG 圧縮の処理を 3.2.1 章に示した 3 つのステップに分けて測定した実行クロック数を示している．以下では，並列化による影響の大きかった STEP 5 及び STEP 6 について調べる．

- STEP 5 (圧縮)
  図 9 に示す通り，並列化により PU 数の増加に伴う実行時間の削減がみられるが，その削減率は台数が増すほど鈍ってしまっている．バス利用率は図 10 に示す通り PU 数 16 ですでに飽和してしまっている．

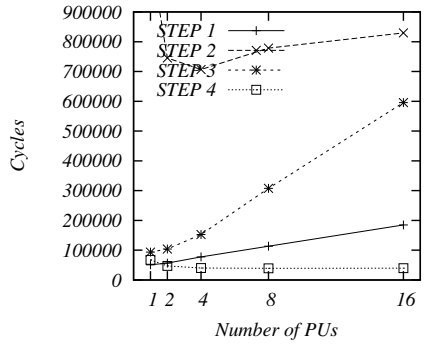

図 11 PU 数の違いによる実行クロック数の変化 (SUSAN Edges)

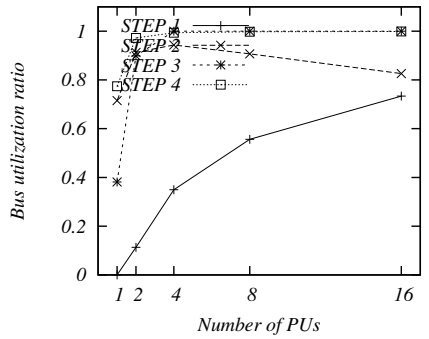

図 12 PU 数の違いによる共有メモリバス利用率の変化 (SUSAN Edges)

単純なバス結合のマルチプロセッサでは限界があり，さらなる台数効果を得るためには共有データのキャッシュを設けネットワーク利用率を低下させるか，クロスバスイッチ等の転送能力が高いネットワークを使用する等の改良が必要であることが，これよりわかる．

- STEP 6 (データの収集)
  このステップでは主に共有メモリへの書き込みを行うので，図 10 のように PU 数を増加させるとバスの飽和が急激に起こってしまう．転送能力の高いネットワークを使用することにより，Write アクセスによるネットワークの飽和は緩和されると考えられる．

5.3 SUSAN Edges

SUSAN のエッジ検出は，今回評価した中では最も並列化の効果を得難いベンチマークであった．

PU 数を 1 から 16 まで増加させたところ，PU 数が 2 のとき 1.37 倍の実行速度で最高となるがそれ以降は台数を増やすと速度が低下してしまう．エッジ検出の処理を STEP 1 から STEP 4 に分けて，それぞれのステップにおける実行時間を図 11 に，バス利用率を図 12 に示す．

- STEP 2, 4 (エッジの強調，原画像へ重ね合わせ)
  この 2 つのステップは並列化されているが，2 台以上では PU 数の増加による台数効果が得られていない．両ステップともに共有メモリの読み出しと書き込みが頻繁に発生するという特徴があり，図 12 からもわかる通りバスがすぐに飽和してしまっている．転送能力の低いバスによる結合網がボトルネックとなり，実行時間の短縮を妨げてしまっている．
- STEP 3 (エッジの補正)
  単一 PU が共有メモリ上のエッジデータの補正を行う．しかし，STEP 3 を実行しない PU は同期待ちのため共有メモリ上の同期変数をアクセスし続けるビ

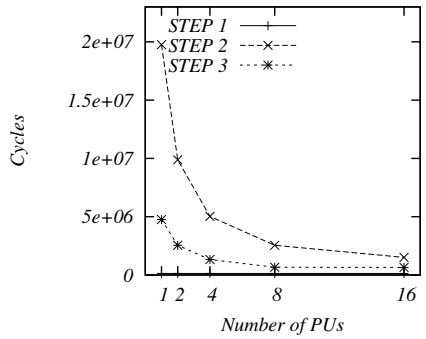

図 13 PU 数の違いによる実行クロック数の変化 (Blowfish)

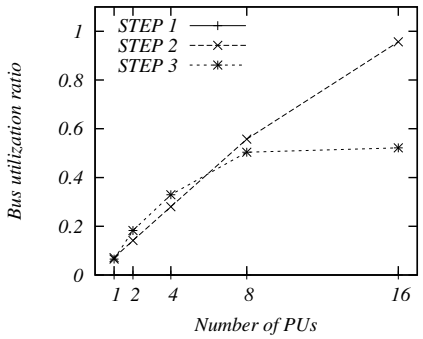

図 14 PU 数の違いによる共有メモリバス利用率の変化 (Blowfish)

ジーウェイトを行う．
評価に用いたアーキテクチャでは各 PU が共有メモリに対するキャッシュを持たないので，このビジーウェイトが原因で図 12 の通り PU 数を増加させるとバス利用率が上がってしまい，STEP 3 の実行サイクル数が増加してしまっていた．

5.4 Blowfish encrypt

このアプリケーションは，並列化することによりほぼ理想的な性能向上が達成できた．3.2.3 章に記した各ステップ毎における並列化の効果を見るため，実行サイクル数とバス利用率を図 13 と図 14 に示す．

- STEP 1 (鍵拡張)
  このステップは並列化していないが，図 13 の通り，プログラム全体からみるとこのステップにかかる実行サイクルは微々たるものである．このステップを高速化することによる恩恵はほとんど得られない．
- STEP 2 (暗号化)
  図 13 によると，このステップは PU 数 1〜8 までは理想的な台数効果が得られている．この要因としては，頻繁に読み出されるサブ鍵を各 PU のローカルデータとして持たせたことが挙げられる．本研究で用いたアーキテクチャでは，ローカルデータがキャッシュ可能となっており，これがサブ鍵のアクセス時に有効に働いた．
  また，16 台まで増やすと実行速度は 13.14 倍と若干伸びが鈍っているが，この原因のひとつに図 14 に示すバス利用率の上昇が挙げられる．さらなる高速化がが必要であれば，転送能力の高いネットワークを用いることで可能となる．
- STEP 3 (データの収集)
  各 PU によるローカルメモリから共有メモリへの書き込みが一斉に発生するが，バス利用率の急激な増加は見られず実行時間の短縮が達成された．しかし，このステップは全体の実行時間からみると占める割合が低いので高速化による恩恵は少ない．

5.5 Patricia Trie

図 8 に示す通り，PU 数の増加に伴う高速化が達成できており，特に PU 数が 8 まではほぼ線形に性能が伸びている．共有メモリアクセスの大部分は木構造の検索時のロード命令であること，そして本ベンチマークがネットワークアドレスに対する問い合わせを扱う性質から，共有データをキャッシュすることでさらなる高速化が望める．

6. 結論および今後の課題

本稿では，組込み分野における利用を目的とする CMP で単一ジョブを並列化した場合の性能評価を行うために，MiBench suite が提供するベンチマークプログラムを並列化した．バス結合のマルチプロセッサシステムシミュレータ上で動作させ評価を行い，並列化の効果および性能低下となる要因を検証できることを示した．

これにより，本研究で作成した並列プログラムは CMP のアーキテクチャを検討するために利用でき，ボトルネックとなる要因等を明らかにする上で有益であることが示せた．

また，今後の課題として，MiBench suite からさらに多くのプログラムを並列化すること，今回想定したアーキテクチャも含め，多様な構成の CMP を想定し，それぞれにおいて並列化した場合に得られる性能を評価する事，他のプラットフォームにも移植が容易な形で提供する方法を検討し，公開すること等が挙げられる．

参 考 文 献

1) ARM. Arm developers' guide (arm11 mpcore multiprocessor semiconductor).
2) 東芝半導体. Mep(media embedded processor) 概説. 製品カタログ http://mepcore.com/, April 2004.
3) Satoshi Kaneko et al. A 600 mhz single-chip multiprocessor with 4.8 gb/s internal shared pipelined bus and 512 kb internal memory. In *ISSCC Digest of Technical Papers*, pp. 254–255, 2003.
4) Edn embedded microprocessor benchmark consortium. http://www.eembc.org.
5) Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, and Richard B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *Proc. of IEEE 4th Annual Workshop on Workload Characterization, Austin, TX*, December 2001.
6) Stephen M. Smith and J. Michael Brady. Susan - new approach to low level image processing. *Int. J. Comput. Vision*, Vol. 23, No. 1, pp. 45–78, 1997.
7) Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *Fast Software Encryption, Cambridge Security Workshop*, pp. 191–204, London, UK, 1994. Springer-Verlag.
8) Donald R. Morrison. Patricia – practical algorithm to retrieve information coded in alphanumeric. *ACM*, Vol. 15, No. 4, pp. 514–534, October 1968.
9) 薬袋俊也, 塙敏博, 田辺靖貴, 天野英晴. Isis-simplescalar の実装. 情報処理学会研究報告, 2004-ARC-160, pp. 29–34, December 2004.