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

2009-SLDM-138 (25)
2009 / 1 / 30

組み込みシステム向けMPSoCのための マルチレイヤ構造をとるバスアーキテクチャ最適化手法

吉田 陽信† 戸川 望† 柳澤 政生† 大附 辰夫† 橘 昌良††

† 早稲田大学大学院基幹理工学研究科情報理工学専攻
〒169-8555 東京都新宿区大久保3-4-1
†† 高知工科大学工学部電子・光システム工学科
〒782-8502 高知県香美市土佐山田町宮ノ口185
E-mail: †yoshida@ohtsuki.comm.waseda.ac.jp

あらまし マルチレイヤ構造をとるバスアーキテクチャを対象とし，特定のアプリケーションに適した構成を選択するためのバスアーキテクチャ最適化手法を提案する．入力としてプロセッサシミュレータから取得したアプリケーションのトレースデータと時間制約を与え，まずメモリアクセス競合を考慮せずにトレースデータから求めたデータ転送時間によって制約を満たす可能性のある構成を限定する．その後，各構成についてメモリアクセス競合を考慮したスケジューリングをすることで，制約を満たすか否かを判定をする．この時，面積の小さい構成から大きい構成の順に探索することにより面積を最小とする構成を能率良く発見することができる．計算機実験を行った結果からマルチレイヤ構造のバスを面積が同等と考えられる共有バスと比較し，有効性を確認した．また提案する探索範囲削減手法は一般的な全探索手法と比較し，8.55倍高速に最適解を求められることを示した．

キーワード MPSoC，バスアーキテクチャ最適化，組み込みシステム

A Multi-layer Bus Architecture Optimization Algorithm for MPSoC in Embedded Systems

Harunobu YOSHIDA†, Nozomu TOGAWA†, Masao YANAGISAWA†, Tatsuo OHTSUKI†, and Masayoshi TACHIBANA††

† Dept, of Computer Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku, Tokyo, 169-8555 Japan
†† Dept, of Electronics and Photonic Systems, Kochi University of Technology
185 Miyanokuchi, Tosayamada, Kami, Kochi, 782-8502 Japan
E-mail: †yoshida@ohtsuki.comm.waseda.ac.jp

Abstract In this paper, we propose an on-chip bus optimization algorithm for a multi-layer bus architecture. Our algorithm efficiently searches for an optimal selection of the number and bit-size of buses, CPU-bus connection topology, and the priority of each CPU subject to the time constraint for given embedded applications. It is necessary to estimate the running time of applications with taking into consideration the effect of memory access conflict. Before taking into consideration the effect of memory access conflict, our approach removes configurations which violate the constraints. By reducing the design space in this way we can obtain an optimal configuration in shorter time. Our algorithm is 8.55 faster compared to the exhaustive approach.

Key words MPSoC, Bus Architecture Optimization, Embedded System

1. まえがき

高性能と低消費電力を両立するために1チップに複数のCPUを搭載し，複数のCPUで処理を分散させたり，並行させることで性能の向上を図るMPSoCが開発されている．MPSoCではCPUコア数の増加に伴いCPUコアから共有メモリへのア

クセスが増加するため，バス競合による処理の遅延が問題となってくる．バスの競合はバスの本数を増加させることで緩和させることができるが，バスの本数を増やすことは面積が増大することになる．面積を抑えて求められる性能を満たすようにバスアーキテクチャを最適なものにする必要がある．

MPSoC のバスアーキテクチャ最適化としてバスマトリクス，クロスバと呼ばれる構造を対象とした最適化手法 [7], [9], [10] や，バスブリッジで複数のバスを接続したアプリケーションに特化した構造を対象とした最適化手法 [1], [2], [3], [4], [5], [8] に関する研究が報告されている．しかし，ごく最近の MPSoC では NEC の MP211 [13] やルネサスの SH-Navi2 [11] のようにマルチレイヤバス構造をとるものが採用されており，文献 [13] では 64bit のバス 1 本,2 本，32bit のバス 4 本という構造において過負荷をかけたときにリアルタイム性能がどれだけ劣化するかを評価し，マルチレイヤ構造の 32bit のバス 4 本が最も性能の劣化を抑えられる構造であるという報告がされている．このようにリアルタイム性能を重視した構造として有効性が示されており，マルチレイヤバスに関して構造を最適化することは重要であると考えられる．しかし，現在のところマルチレイヤ構造をとるバスアーキテクチャの最適化に関する研究はほとんどされていない．

本稿ではマルチレイヤ構造をとるバスアーキテクチャの最適化手法を提案する．組み込みシステムにおける MPSoC では各プロセッサコアに異なる処理を割り当てる機能分散型と，複数のプロセッサコアに同じ処理を割り当てる負荷分散型があるが，本手法では機能分散型の MPSoC を対象とし複数個の CPU それぞれで異なるアプリケーションを処理させることを想定し，各アプリケーションに時間制約を与えた時，CPU と共有メモリ間のバスを最適な構成にする．バスのビット幅と本数を増やすことによりバス内部のデータ転送を高速化できるため，面積を考慮しなければ制約を満たす構成を発見することは難しくないが，小面積である事が求められるため，性能を満たし最小の面積となる構成が最適な構成であるといえる．本手法では複数個のメモリアクセスのトレースデータと制約条件を入力することで，制約時間内にデータ転送が終了できる可能性がある最低限必要なビット幅と本数と，確実に制約を満たすビット幅と本数を求めることにより探索範囲を限定する．探索範囲内にはバスの構成要素の組み合わせの数だけ制約を満たす可能性のある構成が存在するが，それぞれの構成が制約を満たすかはメモリアクセス競合を考慮したスケジューリングによって判定する必要がある．探索時間はスケジューリングする構成数に依存するが，本稿で提案する探索範囲削減手法によりスケジューリングするバスの接続形態の数を限定し，面積の小さい構成から大きい構成の順に探索することで，面積を最小とする構成を発見するための時間を短縮することが期待できる．計算機実験の結果から全探索を行った場合と本稿で提案する探索範囲削減手法を適用させた場合の比較では約 8.55 倍実行時間が改善された．

2. 問題の定式化

本節では，対象とするマルチレイヤバスアーキテクチャ，入力となるメモリアクセスシーケンス，制約条件とアプリケーションの処理回数を説明し，その後，バス最適化問題を定式化する．

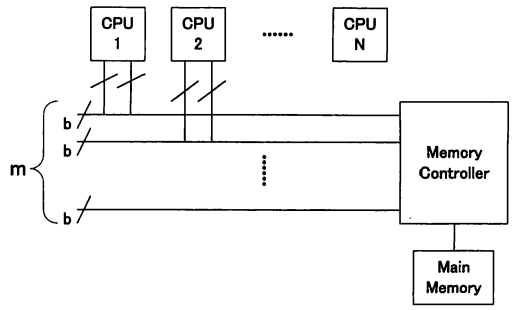

図 1　マルチレイヤバスアーキテクチャ

2.1 マルチレイヤバスアーキテクチャ

図 1 に本稿で対象とするマルチレイヤバスアーキテクチャを示す．CPU の個数を N [個]，バスの本数を m[本]，バス幅を b [bit]($b = 8, 16, 32, 64, 128$) とする．

各 CPU はインストラクションメモリを内蔵し，バスへのアクセスはメインメモリに対するデータアクセスのみとする．各 CPU はリード用とライト用のポートがあり，メモリと接続されるバスと 1 対 1 で接続される．また固定の優先順位を持ち，バスへのアクセスが同時に発生した場合は優先順位の高いものがバスの使用権を得る．バスとメモリはメモリコントローラを介して接続されると仮定する．

メインメモリは SDRAM を仮定する．メモリへのアクセス命令が発行されてから最初のデータの読み出し，書き込みの実行が完了するまでのクロックサイクル数をイニシャルリードレイテンシ，イニシャルライトレイテンシ，最後のデータの読み出し，書き込みの実行が完了するまでのクロックサイクル数をそれぞれリードレイテンシ，ライトレイテンシとよぶことにする．データ転送方式はシングル転送と 2,4,8 回連続のバースト転送があり，連続するメモリアクセスのデータ量に応じて，適切な転送方式をメモリコントローラが選択するものとする．シングル転送はイニシャルレイテンシで実行を完了し，バースト転送はイニシャルレイテンシ後に 1 クロック毎に連続してデータの読み出し，書き込みが実行される．シングル転送と 4 回連続のバースト転送のタイミングチャートを図 2，3 に示す．バースト転送中に，バスの使用権を持つ CPU よりも優先順位の高い CPU のアクセスが発生した場合は，バースト転送が終了するまで，バスの使用権は譲らない．

2.2 メモリアクセスシーケンス

メモリアクセスシーケンスは CPU 1，CPU 2,$\cdots$,CPU N が処理するアプリケーションのメモリアクセスの流れを表す．各 CPU は，それぞれ 1 つのメモリアクセスシーケンスを実行する．図 4 にメモリアクセスシーケンスを示す．ここで以下の記号を導入する．

MAS_i:　CPU i のメモリアクセスシーケンスを表す．MAS_i

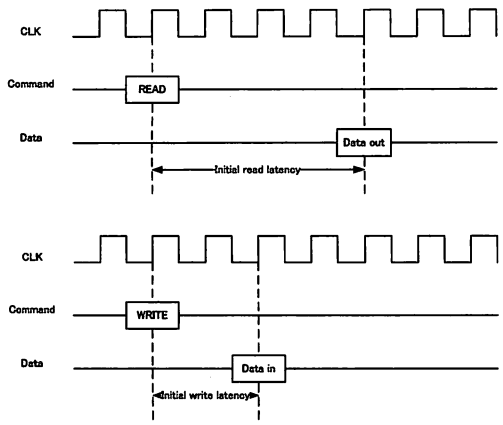

図 2　シングル転送のタイミングチャート

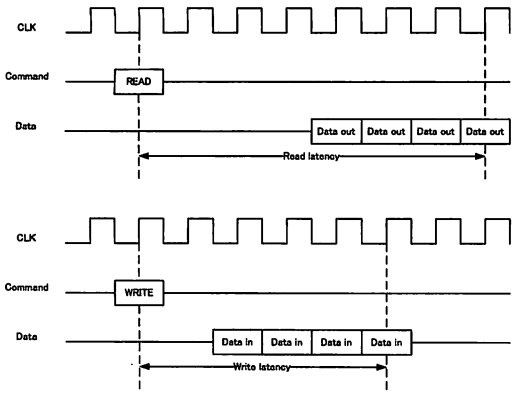

図 3　4 回連続のバースト転送のタイミングチャート

は式 (1) で表される.

$$MAS_i = (V_i, E_i), \tag{1}$$

V_i: ノードの集合を表す. 式 (2) で表される.

$$V_i = \{s, e\} \cup R_i \cup W_i (i = 1, \cdots, N). \tag{2}$$

E_i: 枝の集合を表す. 枝 $e \in E_i$ は CPU i のメモリアクセス以外の計算のサイクル数を持つ ($i = 1, \cdots, N$).

s: メモリアクセスシーケンスの開始を表す.

e: メモリアクセスシーケンスの終了を表す.

R_i: CPU i のリードの属性を持つノードの集合を表す. 式 (3) で表される.

$$R_i = \{r_1^i, r_2^i, \cdots\} \tag{3}$$

W_i: CPUi のライトの属性を持つノードの集合を表す. 式 (4) で表される.

$$W_i = \{w_1^i, w_2^i, \cdots\} \tag{4}$$

CPU i のメモリアクセスシーケンスにおける j 番目のノードが持つデータ量を $Data_j^i$ [byte], k 番目の枝が持つサイクル数を $Work_k^i$ [clock] と書くことにする.

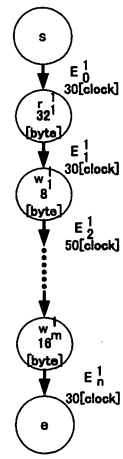

図 4　メモリアクセスシーケンス

2.3　制約条件とアプリケーションの処理回数

各 CPU に対し, 実行されるアプリケーションの処理の開始から終了までに許容される時間制約を与える. CPU i で実行されるメモリアクセスシーケンスの終了時間制約を $const_i$ [s] とする. 各アプリケーションには異なる終了時間制約を与えることができる. すべての終了時間制約に共通する最小の整数倍となる時間を全体制約 $T_{const_{all}}$ [s] とする. 各アプリケーションの処理は同時に実行開始され CPU i は $T_{const_{all}}$ [s] の間に以下の式 (5) で表わされる L_i 回の処理を終了させ, かつ 1 回の処理が終了時間制約以内であることが制約を満たす条件となる.

$$L_i = \frac{T_{const_{all}}}{const_i} \tag{5}$$

図 5 に各アプリケーションに与えられる終了時間制約をそれぞれ 0.12s, 0.06s, 0.08s, 0.08s とした時の例を示す. この時, 全体制約は 0.24s になる. その間に CPU1 では 0.12s 以内で 2 回, CPU2 では 0.06s 以内で 4 回, CPU3 では 0.08s 以内で 3 回, CPU4 では 0.08s 以内で 3 回の処理が終了していることが制約を満たす条件になる.

2.4　バス最適化問題

2.1 項で定義したマルチレイヤバスアーキテクチャを対象とするバス最適化問題を定式化する.

バス最適化問題とは, 入力として, CPU の個数 N, 各 CPU が処理するアプリケーションのメモリアクセスシーケンス, バスクロック C_{bus} [Hz], CPU のクロック C_{cpu} [Hz] を与えたとき, 制約条件を満たすバス構成要素であるバス幅 b [bit], バス本数 m [本], バスの接続形態, 各 CPU の優先順位と各 CPU の処理時間 [s] を出力することである. 制約条件を満たした上で式 (6) で表わされるバスのコスト関数を最小化し, その中で m を最小とするものを最適解とする.

$$Bus_{cost} = m \times b \tag{6}$$

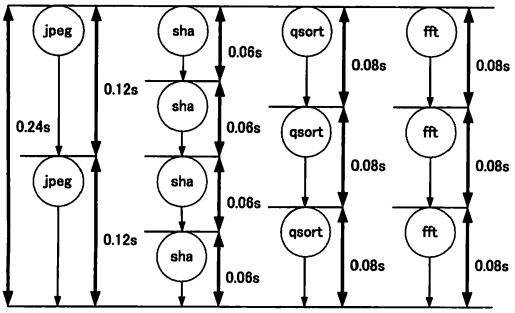

図 5　アプリケーションの処理回数の例

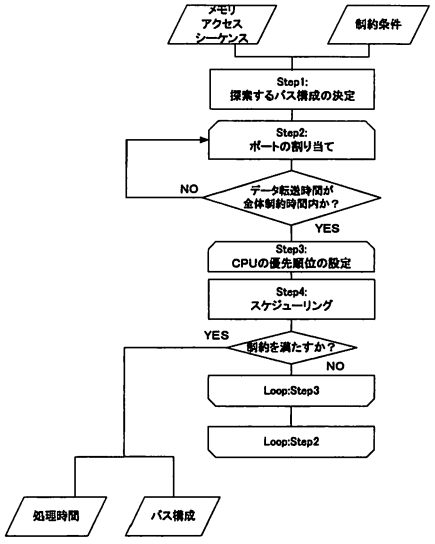

図 6　全体のフロー

3. バス最適化アルゴリズム

入力として，各アプリケーションのメモリアクセスシーケンスと終了時間制約条件が与えられた時，出力として制約を満たした上でコスト関数を最小かつバスの本数 m を最小とするバス構成要素と各 CPU の処理時間を得るアルゴリズムを提案する．図 6 に提案手法の概要を示す．

アプリケーションと制約が与えられ，各アプリケーションの処理の開始時刻，バス構成要素が決定している時，各 CPU がメモリにアクセスする時刻が決められる．同じバスに接続される CPU があり同時刻にメモリへのアクセスがある場合をバス競合とよぶことにする．あるバス構成が制約を満たすかはバス競合を考慮して各アプリケーションの CPU 内部の処理，リード転送，ライト転送がどの時刻に実行されるかをスケジューリングすることで判明する．

本手法では Step1 で入力するアプリケーションのメモリアクセスシーケンスと制約条件から各 CPU が制約を満たすために最低限必要とするバス幅とバスの本数の組み合わせを求め，探索する範囲を限定する．Step2 で各 CPU のリードポートとライトポートをバスに接続する．この時，バス競合を考慮せずにデータ転送時間が制約時間を超えてしまうことが分かる構成は Step3 へ進まない．Step3 で CPU の優先順位を設定し，Step4 で Step2,Step3 で設定したバスの構成でバス競合を考慮したスケジューリングを行い，条件を満たすものを出力する．

表 1　レイテンシ

転送回数	リードレイテンシ [clock]	ライトレイテンシ [clock]
1	4(シングル転送)	2(シングル転送)
2	5(2 連続バースト転送)	3(2 連続バースト転送)
3	7(4 連続バースト転送)	5(4 連続バースト転送)
4	7(4 連続バースト転送)	5(4 連続バースト転送)
5	11(8 連続バースト転送)	7(シングル転送 & 4 連続バースト転送)
6	11(8 連続バースト転送)	8(2 連続バースト転送 & 4 連続バースト転送)
7	11(8 連続バースト転送)	9(8 連続バースト転送)
8	11(8 連続バースト転送)	9(8 連続バースト転送)

3.1 Step1: 探索範囲と探索順序の決定

制約を満たすために必要なバス幅のバスが N [本] あり，各 CPU がバスと 1 対 1 で接続されたと仮定すると，バス競合が発生しないので確実に制約を満たす構成になる．制約を満たすために最低限必要なバス幅は以下のように求める．ここで MAS_i のメモリアクセス以外の処理に要する時間を T_{Ei} とする．

$$T_{Ei} = \frac{\sum_{k \in E_i} Work_k^i}{C_{cpu}} \tag{7}$$

メモリアクセスの転送時間はデータ転送回数から求められる．バス幅が b の場合の MAS_i の j 番目のノードのデータ転送回数 $n(b,i,j)$ は以下の式 (8) で表される．

$$n(b,i,j) = \left\lceil \frac{Data_j^i}{b} \right\rceil \tag{8}$$

転送回数 $n(b,i,j)$ の場合のリードレイテンシ，ライトレイテンシをそれぞれ $Rl(n(b,i,j))$，$Wl(n(b,i,j))$ とおく．イニシャルリードレイテンシを 4 clock，イニシャルライトレイテンシを 2 clock とした場合の転送回数とリードレイテンシ，ライトレイテンシの関係を表 1 に示す．リードレイテンシ，ライトレイテンシは転送時間が最小となるようにシングル転送と 2,4,8 回連続のバースト転送を組み合わせる．例えば転送回数が 3 回のリード転送では，シングル転送と 2 回連続バースト転送の組み合わせが考えられるが，転送完了までに 9 clock 要する．4 回連続バースト転送だと 7 clock で完了するので，この場合は 4 回連続バースト転送を選択する．

MAS_i の全てのリード転送，ライト転送に要する総クロックサイクル数 $C_{Ri}(b)$，$C_{Wi}(b)$ は式 (9)，(10)，全てのリード転送に要する時間とライト転送に要する時間 $T_{Ri}(b)$，$T_{Wi}(b)$ は式 (11)，(12) で表される．

$$C_{Ri}(b) = \sum_{j \in R_i} Rl(n(b,i,j)) \tag{9}$$

$$C_{Wi}(b) = \sum_{j \in W_i} Wl(n(b,i,j)) \tag{10}$$

$$T_{Ri}(b) = \frac{C_{Ri}(b)}{C_{bus}} \tag{11}$$

Step1.1 MAS_i の T_{Ei}, $T_{Ri}(b)$, $T_{Wi}(b)$ を求める.
Step1.2 以下の式を満たす最小のバス幅求め，これを b_i とする.
$const_i - (T_{Ei} + T_{Ri}(b) + T_{Wi}(b)) \geqq 0$
Step1.3 $b_i (i = 0, \cdots, N)$ の最大値 B_{max} を求める.
$B_{max} = \max b_i$

図 7 探索開始のバスの構成を求めるアルゴリズム

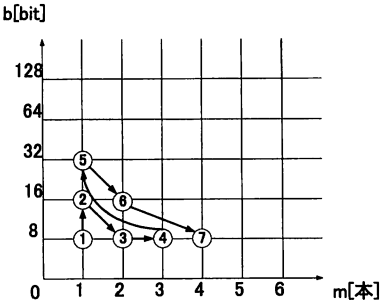

図 8 探索順序の例

$$T_{Wi}(b) = \frac{C_{Wi}(b)}{C_{bus}} \qquad (12)$$

T_{Ei}, $T_{Ri}(b)$, $T_{Wi}(b)$ の合計値で表わされる MAS_i 全体の処理時間が終了時間制約 $const_i$ 以下であれば制約を満たすことになる．MAS_i において式 (13) を満たす最小のバス幅 b を求め b_i とする.

$$const_i - (T_{Ei} + T_{Ri}(b) + T_{Wi}(b)) \geqq 0 \qquad (13)$$

b_i の最大値を B_{max} とおくと，すべてのメモリアクセスシーケンスの制約を満たすためには最低でも B_{max} [bit] 以上のバス幅が必要となる．B_{max} [bit] を探索開始のバス幅とし，確実に制約を満たす B_{max} [bit] のバスが N [本] の時が探索を終了するバス構成となる．コスト関数 $Bus_{cost} = B_{max}$ 以上で $Bus_{cost} = N \times B_{max}$ 以下にある構成が探索範囲となる．コスト関数の小さいものから順に探索し，制約を満たした時点で終了する．図 7 に制約を満たすために最低限必要なバス幅を求めるアルゴリズムを示す．図 8 に CPU が 4 個，最低限必要なバス幅が 8 bit の場合の探索の順序を示す．この例では 8 bit 幅のバス 1 本が探索開始の構成になり，すべての CPU がバスと 1 対 1 で接続される 8 bit 幅のバス 4 本の時が探索を終了する構成となる．探索開始の構成のコスト関数 $Bus_{cost} = 8$ と探索終了の構成のコスト関数 $Bus_{cost} = 32$ の間の 7 点が探索の範囲となる.

3.2 Step2: ポート割り当て

各 CPU のリードポートとライトポートをバスに接続する．この時，式 (11),(12) を用いて得られる各アプリケーションのデータ転送時間をもとに制約を満たす可能性を判定し，可能性があるポートの割り当て方であれば Step3 へ進む．制約を満たす可能性のある構成とはバスに接続される全ポートのデータ転送時間の合計値が制約時間以内になるものである.

図 9 にポートの割り当ての決定のアルゴリズムを示す．ここ

Step2.1 バス幅 b [bit], m [本] と設定.
Step2.2 CPU i のリードポート，ライトポートをバスに接続する.
Step2.3 制約時間とバス j に接続される全ポートの総データ転送時間の差をとる．$T_{bus_j} = T_{const_{all}} - \sum(T_{Ri(b)} + T_{Wi(b)})$
Step2.4 $T_{bus_j} (j = 1, \cdots, m)$ が全て正の値であれば，Step3 へ.

図 9 ポートの割り当ての決定

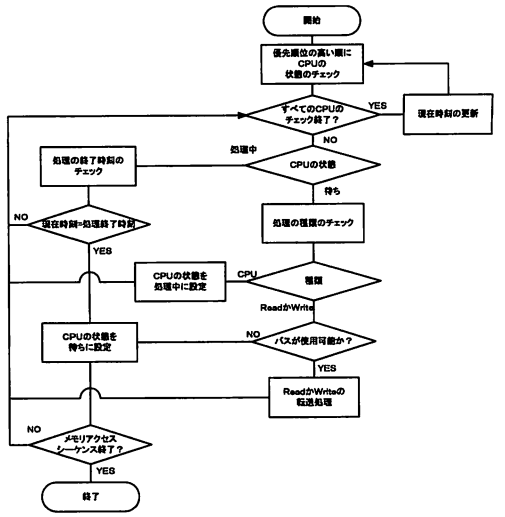

図 10 スケジューリングのフローチャート

表 2 入力アプリケーションのデータ

CPU	CPU 1	CPU 2	CPU 3	CPU 4
アプリケーション	jpeg	sha	qsort	fft
リード転送量 [byte]	1481005	1481168	1674377	2176004
ライト転送量 [byte]	495185	569547	1258547	1416674
CPU 内部処理の総サイクル数 [clock]	852712	944558	1056331	993869
終了時間制約 [s]	0.12	0.06	0.08	0.08

表 3 各アプリケーションの処理時間

	CPU1	CPU2	CPU3	CPU4
	jpeg	sha	qsort	fft
制約時間 [s]	0.12	0.06	0.08	0.08
バス構成	(8,((1,1),(2,2),(3,3),(1,1)),(2,3,4,1))			
1 回目の処理時間 [s]	0.08789784	0.04236585	0.05364814	0.07857446
2 回目の処理時間 [s]	0.07544798	0.04236585	0.05364814	0.06773732
3 回目の処理時間 [s]		0.04236585	0.05364814	0.06771989
4 回目の処理時間 [s]		0.04236585		

で CPU の個数を N [個]，バスの本数を m [本]，各アプリケーションに与えられる時間制約の最小の倍数である全体制約を T_{const_all}，バス $j(j = 1...m)$ の転送可能な時間を T_{bus_j} とする．Step1 で限定した探索範囲内にあるバス幅 b [bit]，バスの本数 m [本] の組み合わせについてこのアルゴリズムを適用させ，制約を満たす可能性のあるポートの割り当てを決定する.

3.3 Step3: CPU の優先順位の設定

CPU の優先順位を設定する．CPU が N[個] の時 $N!$ 通りの全ての組み合わせを設定する.

3.4 Step4: スケジューリング

バスの競合を考慮してスケジューリングすることで Step2, Step3 で設定したバスの構成が，制約を満たすかを判定する．図 10 にスケジューリングのフローチャートを示す.

表 4 全探索との比較

手法	スケジューリング構成数	実行時間 [s]
提案手法	14136	145940
提案手法 - Step2	29376	250666
全探索	81912	1247514

4. 計算機実験

提案する手法をC++言語によって計算機上に実装した．実験環境として，OSがDebian GNU/Linux，CPUはIntel Xeon 3.40GHz，メモリ容量4GB，対象とするアプリケーションはMiBench [6] を用いた．入力のメモリアクセスシーケンスはMiBenchのアプリケーションをSimpleScalar/ARM [12] で実行させた時のトレースデータを基に作成した．実験に用いたデータを表2に示す．各CPUのクロック C_{cpu} を100 MHz，バスクロック C_{bus} を100 MHz，イニシャルリードレイテンシを4 clock，イニシャルライトレイテンシを2 clockと設定した．

表3に解として出力されたバス構成と各CPUで実行されるメモリアクセスシーケンスの処理時間を示す．制約条件をjpegに0.12s，shaに0.06s，qortに0.08s，fftに0.08sと与えたので，それぞれの終了時間制約条件に共通する最小の倍数となる0.24sが全体制約になり，jpegは2回，shaは4回，qortは3回，fftは3回の処理が実行される．ここでバス幅を b [bit]，CPU i のリードポートが接続されるバスを B_{Ri}，ライトポートが接続されるバスを B_{Wi}，優先順位を P_i とした時，バスの構成を $(b,((B_{R1},B_{W1}),(B_{R2},B_{W2}),\cdots,(B_{RN},B_{WN}))$ $,(P_1,P_2,\cdots,P_N))$ と表す．

提案手法による探索，提案手法からStep2を除いた探索，全探索の比較を表4に示す．全探索は $Bus_{cost}=8$ から $Bus_{cost}=128$ の範囲にある構成全てに対しての探索とした．表5に提案手法が探索した構成を示す．32 bit幅のバス1本の時に制約を満たす構成は存在しないが，コスト関数が同等以下の8 bit幅のバス3本，8 bit幅のバス4本，16 bit幅のバス2本の時に制約を満たす構成が存在することが分かる．提案手法と提案手法からStep2を除いた探索を比較するとスケジューリング構成数を約52%削減し，約1.72倍高速にバス構成を最適化し，提案手法と全探索を比較するとスケジューリング構成数を約83%削減，約8.55倍高速にバス構成を最適化した．

5. むすび

本稿ではマルチレイヤバスアーキテクチャの最適化手法を提案した．MiBenchのアプリケーションを用いて評価実験を行った．実験結果から提案手法を全探索と比較し，約8.55倍高速にバス構成を最適化することを示した．

謝辞

本研究に関し，有用な議論および討論をいただきました，株式会社東芝西尾誠一氏，相原雅己氏および影島淳氏に感謝致します．

表 5 探索した構成

バス幅 [bit]	本数 [本]	制約を満たす構成数
8	1	0
16	1	0
8	2	0
8	3	1273
32	1	0
16	2	377
8	4	24

文献

[1] M. Drinic, D. Kirovski, S. Meguerdichian and M. Potkonjak, "Latency-guided on-chip bus network design," in *Proc. ICCAD 2000*, pp. 420–423, 2000.

[2] K. Lahiri, A. Raghunathan and S. Dey, "Efficient exploration of the SoC communication architecture design space," in *Proc. ICCAD 2000*, pp. 424–430, 2000.

[3] K. Lahiri, A. Raghunathan, and S. Dey, "System-level performance analysis for designing on-chipcommunication architectures," *IEEE Transactions on Computer Aided Design of Integrated Circuit and Systems*, vol. 20, no. 6, pp. 768–783, 2001.

[4] K. Lahiri, A. Raghunathan, G. Lakshminarayana and S. Dey, "Design of high-performance system-on-chips using communication architecture tuners," *IEEE Transactions on Computer Aided Design of Integrated Circuit and Systems*, vol. 23, no. 5, pp. 620–636, 2004.

[5] K. Lahiri, A. Raghunathan and S. Dey, "Design space exploration for optimizing on-chip communication architectures," *IEEE Transactions on Computer Aided Design of Integrated Circuit and Systems*, vol. 23, no. 6, pp. 952–961, 2004.

[6] MiBench
http://cares.icsl.ucla.edu/MediaBench/.

[7] S. Murali, L. Benini and G. D. Micheli, "An application-specific design methodology for on-chip crossbar generation," *IEEE Transactions on Computer Aided Design of Integrated Circuit and Systems*, vol. 26, no. 7, pp. 1283–1296, 2007.

[8] S. Pasricha, N. Dutt, E. Bozorgzadeh and M. Ben-Romdhane, "Floorplan-aware automated synthesis of bus-based communication architectures," in *Proc. DAC 2005*, pp. 565–570, 2005.

[9] S. Pasricha and N. D. Dutt, "A framework for cosynthesis of memory and communication architectures for MP-SoC," *IEEE Transactions on Computer Aided Design of Integrated Circuit and Systems*, vol. 26, no. 3, pp. 408–420, 2007.

[10] S. Pasricha, N. D. Dutt and M. Ben-Romdhane, "BMSYN: Bus matrix communication architecture synthesis for MP-SoC," *IEEE Transactions on Computer Aided Design of Integrated Circuit and Systems*, vol. 26, no. 8, pp. 1454–1464, 2007.

[11] PC Watch, MicroProcessor Forum 2007 レポート,
http://pc.watch.impress.co.jp/docs/2007/0525/mpf04.htm.

[12] SimpleScalar LLC
http://www.simplescalar.com/.

[13] S. Torii, S. Suzuki, H. Tomonaga, T. Tokue, J. Sakai, N. Suzuki, K. Murakami, T. Hiraga, K. Shigemoto, Y. Tatebe, E. Ohbuchi, N. Kayama, M. Edahiro, T. Kusano, and N. Nishi, "A 600MIPS 120mW 70 μA leakage triple-CPU mobile application processor chip, " in *Proc. 2005 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 136–137, 2005.