IP ベース設計における最適バスアーキテクチャ探索手法の提案

上田恭子[†] 坂主圭史[†] 米岡昇[†]
武内良典[†] 今井正治[†]

近年注目されている IP ベース設計では，設計対象システムの処理時間がバスアーキテクチャによって大きく異なるため，バスアーキテクチャの選択は非常に重要な問題である．本論文では，高速処理時間見積り手法を用いた最適バスアーキテクチャ探索手法を提案する．本研究では，バストポロジ，バス動作周波数，バスビット幅，バッファ数によってバスアーキテクチャをパラメータ化し，バスアーキテクチャ最適化問題を定式化する．そして，定式化に基づいてバスアーキテクチャを探索する．提案手法では，システムレベルのプロファイリング結果を用いた高速見積り手法によって処理時間を見積もり，様々なバスアーキテクチャの性能を評価する．評価実験により，提案手法が最適バスアーキテクチャの探索時間を大幅に削減できること，また，機能ブロック群構成も含めたアーキテクチャ最適化を容易とすることが分かった．

Optimal Bus Architecture Exploration for IP-based Design

Kyoko Ueda,[†] Keishi Sakanushi,[†] Noboru Yoneoka,[†]
Yoshinori Takeuchi[†] and Masaharu Imai[†]

In IP-based design, to find the optimal bus architecture is very important problem because bus architecture strongly affects the performance of the target system. This paper proposes a bus architecture optimization method using fast architecture-level performance estimation. The optimization problem of bus architecture that is parameterized by bus topology, bus data transfer rate, bus bit width, and the number of buffers, is formalized and its exploration method is proposed. Proposed method explores all possible bus parameter sets estimating the performance and hardware area. Experimental results show that proposed method can determine the optimal bus architecture and its time is drastically reduced compared to the conventional method. The results also show that the proposed method is helpful to find the optimal architecture.

1. はじめに

近年，多機能化が求められる組み込みシステムは複雑化の一途をたどり，その結果，設計コストが増大するという問題が生じている．そこで，既設計の機能ブロックを再利用することで設計時間を短縮する，IP ベース設計が注目されている[1)]．IP ベース設計では，設計者は IP データベースから機能ブロックを選択し，それらのブロックを接続するバスのアーキテクチャを決定する．面積や処理時間などの設計制約を満たす最適なアーキテクチャを見つけるためには，様々なアーキテクチャでの設計品質を評価する必要がある．したがって，様々なアーキテクチャを対象とした，高速な設計品質見積り手法，および，効率的なアーキテクチャ探索手法が求められる．

従来，システムの性能はアーキテクチャレベルのシミュレーション[2),3)] によって評価されていたが，アーキテクチャレベルのシミュレーションには時間がかかるという問題があった．そこでこれまでに，様々なバスアーキテクチャに適用可能で高速な性能評価手法が提案されている[4)~6)]．特に文献 6) では，実装する機能ブロックやバスアーキテクチャが異なるアーキテクチャの処理時間を，同一のシステムレベル・プロファイリング結果を用いて見積もる手法が提案されている．

バスアーキテクチャの探索手法としては，バスアーキテクチャの初期解を決定し，初期解を改良することで準最適なバスアーキテクチャを生成する手法が提案されている[7)]．この手法では，設計者の指定したバスアーキテクチャ・テンプレートへのデータ転送のマッピングのみを考えるため，探索空間が制限されており，様々なバストポロジを比較する場合はテンプレートを

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

変えて探索する必要がある．

本論文では，文献 6) で提案された高速処理時間見積り手法を用いた，最適バスアーキテクチャ探索手法を提案する．バスアーキテクチャをバストポロジ，バス動作周波数，バスビット幅，およびバッファ数でパラメータ化し，バスアーキテクチャ最適化問題を定式化する．提案手法はバス・パラメータ探索木を用いて，考えうるすべてのパラメータの組合せを探索する．また，探索時間を削減するために，バス・パラメータの探索順序について検討した．

本論文の構成は以下のとおりである．2 章でバスアーキテクチャ最適化問題の定式化および探索アルゴリズムについて述べる．3 章で実験結果について述べ，4 章でまとめる．

2. 最適バスアーキテクチャ探索

本章では，最適バスアーキテクチャ探索手法について述べる．本論文では，ハードウェア面積制約下での処理時間最短化を対象とする．

高速処理時間見積り手法[6] を用いた最適バスアーキテクチャ探索フローを**図 1** に示す．提案する最適バスアーキテクチャ探索手法は，対象システムのシステムレベルモデル（SLM），プロファイリング用テストデータ，機能ブロック群構成情報，ハードウェア面積制約，バス動作周波数候補，バスビット幅候補，および最大バッファ数を入力とし，バストポロジ，各バスの動作周波数およびビット幅，各チャネルの入力および出力バッファ数を最適化する．

文献 6) で提案された高速処理時間見積り手法は，様々なアーキテクチャの処理時間を，同一のプロファイリング情報を用いて見積もることができるため，提案手法はバスアーキテクチャ探索の前に一度だけシステムの実行順序関係をプロファイリングにより取得する．そして，バス・パラメータ探索木を構築，トレースして評価対象のアーキテクチャを決定し，対象アーキテクチャのハードウェア面積および処理時間を評価する．

2.1 対象システムモデル

本節では，バスアーキテクチャ最適化の入力となるシステムレベルモデル，および，出力であるアーキテクチャレベルモデルについて説明する．

2.1.1 システムレベルモデル

システムレベルモデル（SLM）はデータ処理を表すプロセスとプロセス間のデータ転送を表すチャネルで構成される．プロセスは，入力チャネルから受信したデータを処理し，出力チャネルから送信する．システムレベルモデルでは，データ処理時間およびデータ転送時間の情報は持たない．

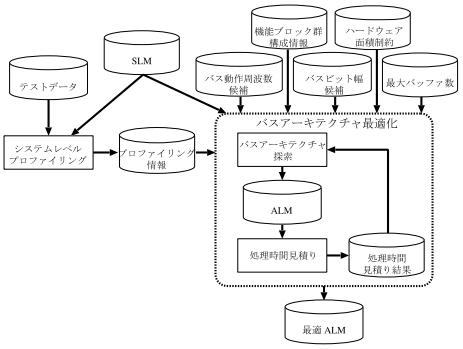


図 1 最適バスアーキテクチャ探索フロー
Fig. 1 Bus architecture optimization flow.

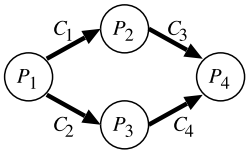


図 2 システムレベルモデルの例
Fig. 2 An example of SLM.

システムレベルモデルの例を**図 2** に示す．ノード P_1，P_2，P_3，P_4 がプロセス，有向枝 C_1，C_2，C_3，C_4 がチャネルとそのデータ転送方向を表す．

2.1.2 アーキテクチャレベルモデル

アーキテクチャレベルモデル（ALM）は機能ブロック群構成とバスアーキテクチャから構成される．機能ブロック群構成は，対象アーキテクチャに実装される各機能ブロックにマッピングされたプロセスのリストを含む．バスアーキテクチャは，チャネルのマッピングを表すバストポロジ，各バスの動作周波数およびビット幅，各チャネルの入力および出力バッファ数から構成される．機能ブロック，バス，バッファは次の特徴を持つ．

- 機能ブロックは，その機能ブロックにマッピングされたプロセスのデータ処理を行う．
- バスは，そのバスにマッピングされたチャネルのデータ転送を行う．
- すべての機能ブロックおよびバスは並列に動作する．
- 機能ブロックには，データ入力およびデータ出力ごとに，転送データを格納するためのバッファが存在する．各データ転送で転送される最大データ量を，各バッファのバッファサイズとする．入力バッファのデータは，機能ブロックが処理を完了するまで次のデータによって上書きされない．ま

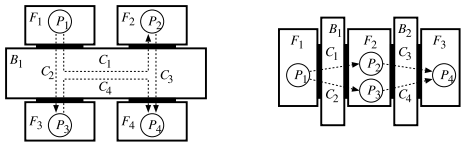


図 3 アーキテクチャレベルモデルの例
Fig. 3 Examples of ALM.

た，出力バッファにあるデータ転送前のデータは，機能ブロックによって上書きされない．

また，機能ブロックのハードウェア面積および各プロセスの処理時間は，あらかじめ IP データベースに登録されているものとする．

アーキテクチャレベルモデルの例を**図 3** に示す．F_i と B_i はそれぞれ機能ブロック，バスを表す．どちらも図 2 のシステムレベルモデルに対するアーキテクチャを表すが，$\boldsymbol{Arch_1}$ は機能ブロック 4 個，バス 1 本のアーキテクチャ，$\boldsymbol{Arch_2}$ は機能ブロック 3 個，バス 2 本のアーキテクチャを表す．

2.2 問題の定式化

バスアーキテクチャ最適化問題を次のとおり定式化する．

- 入力
  - システムレベルモデル：$\boldsymbol{M_{sl}} = (\boldsymbol{P}, \boldsymbol{C})$，ここで，$\boldsymbol{P}$ および $\boldsymbol{C}$ はそれぞれプロセスの集合，チャネルの集合を表す．S_i および N_{max_i} はそれぞれ，チャネル $C_i \in \boldsymbol{C}$ のデータサイズと最大転送データ数を表す．
  - 機能ブロック群構成情報：$\boldsymbol{Arch_{fb}}$．ここで，$\boldsymbol{P}_i$ は機能ブロック $F_i \in \boldsymbol{Arch_{fb}}$ にマッピングされたプロセスの集合を表す．また，A_i は機能ブロック $F_i \in \boldsymbol{Arch_{fb}}$ のハードウェア面積を表す．
  - バス動作周波数候補の集合：$\boldsymbol{R}$
  - バスビット幅候補の集合：$\boldsymbol{W}$
  - 最大バッファ数：N_{buf}
  - ハードウェア面積制約：A_{const}
- 出力
  - 最適バスアーキテクチャ：$\boldsymbol{Arch_{bus}} = (\boldsymbol{B}, \boldsymbol{N_{iobuf}})$，ここで，$\boldsymbol{B}$ および $\boldsymbol{N_{iobuf}}$ はそれぞれ，バスの集合，各チャネルの入力および出力バッファ数の集合を表す．$r_i \in \boldsymbol{R}$ および $w_i \in \boldsymbol{W}$ はそれぞれ，バス $b_i \in \boldsymbol{B}$ の動作周波数，ビット幅を表す．$n_{in_i} \in \boldsymbol{N_{iobuf}}$ および $n_{out_i} \in \boldsymbol{N_{iobuf}}$ はそれぞれ，チャネル $c_i \in \boldsymbol{C}$ の入力および出力バッファ数を表す．
- 目的関数：
  処理時間 $time(\boldsymbol{M_{sl}}, \boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{bus}})$ の最小化
- 制約：
  ハードウェア面積 $area(\boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{bus}}) \leq A_{const}$

2.3 ハードウェア面積見積り

対象アーキテクチャのハードウェア面積は，機能ブロックの面積，バッファの面積，およびバスの面積の和である．アーキテクチャレベルでバスの配線面積を見積もることは困難であるため，本論文ではバスの面積については考慮しない．提案手法では，機能ブロック群構成 $\boldsymbol{Arch_{fb}}$，バスアーキテクチャ $\boldsymbol{Arch_{bus}}$ を持つアーキテクチャのハードウェア面積 $area(\boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{bus}})$ を，式 (1) で見積もる．

$$area(\boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{bus}}) = \sum_i A_i + \sum_i \{buf\_area(S_i \times N_{max_i}) \times n_{in_i} + buf\_area(S_i \times N_{max_i}) \times n_{out_i}\} \quad (1)$$

ここで，$buf\_area(s)$ は，サイズが s のバッファのハードウェア面積を表し，本論文では，サイズ s のレジスタの面積とした．レジスタの面積は，Synopsys 社[8] の Design Compiler の論理合成結果を用いた．

2.4 探索アルゴリズム

本節では，バストポロジ，バス動作周波数，バスビット幅，およびバッファ数の探索木を用いた，最適バスアーキテクチャ探索手法を提案する．提案手法では，バスアーキテクチャ最適化問題を，ハードウェア面積および処理時間の下限を用いた分枝限定法によって最適解を求める．

本手法では，面積や処理時間が等しいアーキテクチャが複数存在する場合は，以下のようにして最適解を選択する．バスの配線面積はバスを共有することによって削減することができる．そこで，面積や処理時間が等しい場合，バスの本数が少ないアーキテクチャを最適解とする．同様に，バスのビット幅は小さいほど配線面積は小さいと考えられるので，バスのビット幅が少ないものを最適解とする．また，バスの動作周波数が低いほど，消費電力は低いと考えられる．そこで，バスの動作周波数が低いものを最適解とする．

2.4.1 バス・パラメータ探索木の構造

図 4 に，バス・パラメータ探索木の構造を示す．以下で，各パラメータの探索木について説明する．

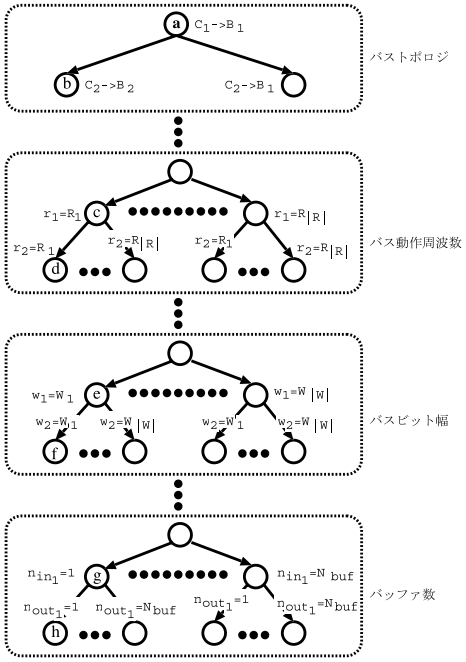


図 4 バス・パラメータ探索木の例
Fig. 4 An example of bus parameter set search tree.

2.4.1.1 バストポロジ

バストポロジは，チャネルのバスへのマッピングを表す．図 4 において，$C_i \rightarrow B_j$ はチャネル C_i がバス B_j にマッピングされることを表す．図 4 のノード a はチャネル C_1 がバス B_1 にマッピングされるが，チャネル C_2 のマッピング先が未定義であることを表し，ノード b はチャネル C_1，C_2 がそれぞれバス B_1，B_2 にマッピングされたバストポロジを表す．バストポロジ探索木の深さは $|\boldsymbol{C}|-1$ となる．バス本数が最大となるのは，各チャネルが専用のバスにマッピングされたバストポロジの場合であり，そのバス本数はチャネル数と同じとなる．

2.4.1.2 バス動作周波数

バス動作周波数探索木で定義された動作周波数で，各バスは駆動される．図 4 において，$r_i = R_j$ はバス B_i の動作周波数に $R_j \in \boldsymbol{R}$ が割り当てられたことを表す．図 4 のノード c はバス B_1 の動作周波数に R_1 を割り当て，バス B_2 の動作周波数が未定義であることを表し，ノード d はバス B_1，B_2 の動作周波数に R_1 が割り当てられたことを表す．バス動作周波数の探索木の深さは $|\boldsymbol{B}|$ となる．

2.4.1.3 バスビット幅

各バスは，バスビット幅探索木で定義されたビット幅を持つ．図 4 において，$w_i = W_j$ はバス B_i のビット幅に $W_j \in \boldsymbol{W}$ が割り当てられたことを表す．図 4 のノード e はバス B_1 のビット幅に W_1 を割り当て，バス B_2 のビット幅が未定義であることを表し，ノード f はバス B_1，B_2 のビット幅に W_1 が割り当てられたことを表す．バスビット幅探索木の深さは $|\boldsymbol{B}|$ となる．

2.4.1.4 バッファ数

各チャネルの入力および出力バッファ数は，バッファ数探索木で定義された数となる．図 4 において，$n_{in_i} = j$ はチャネル C_i の入力バッファ数に $j \leq N_{buf}$ が割り当てられたことを表し，$n_{out_x} = y$ はチャネル C_x の出力バッファ数に $y \leq N_{buf}$ が割り当てられたことを表す．図 4 のノード g はチャネル C_1 の入力バッファ数に 1 を割り当て，出力バッファ数が未定義であることを表し，ノード h はチャネル C_1 の入力および出力バッファ数に 1 が割り当てられたことを表す．バッファ数探索木の深さは $2 \times |\boldsymbol{C}|$ となる．

2.4.2 バス・パラメータ探索木の枝刈り

提案手法では，ハードウェア面積および処理時間の下限によりバス・パラメータ探索木の枝刈りを行う．枝刈りを行うことで，実際に探索するノード数を削減する．以下で，2 種類の枝刈りについて述べる．

2.4.2.1 ハードウェア面積の下限による枝刈り

バス動作周波数，バスビット幅，およびバッファ数が少ないほど，ハードウェア面積は小さくなるため，探索中のノードを頂点とする部分木の中で最小のハードウェア面積を持つアーキテクチャを表す葉は，探索中のノードで未定義のバス・パラメータに対して最小の値を割り当てた場合の葉となる．したがって，ハードウェア面積の下限は式 (2) で表される．

$$lower\_area(\boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{crnt}}) = area(\boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{small}}) \quad (2)$$

ここで，$\boldsymbol{Arch_{crnt}}$ は探索中のノードが表す，いくつかのパラメータが未定義のバスアーキテクチャを表し，$\boldsymbol{Arch_{small}}$ は探索中のノードで未定義のパラメータに対して最小値を割り当てたバスアーキテクチャを表す．

探索中のノードを頂点とする部分木におけるハードウェア面積が最小のアーキテクチャを表す葉が明らかであり，そのアーキテクチャのハードウェア面積がハードウェア面積制約よりも大きい場合，その部分木にはハードウェア面積制約を満たすバスアーキテクチャは含まれない．そこで，式 (3) を満たすノードの子孫を枝刈りする．

$$lower\_area(\boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{crnt}}) > A_{const} \quad (3)$$

2.4.2.2 処理時間の下限による枝刈り

バス動作周波数，バスビット幅，およびバッファ数が大きいほど，処理時間は短くなる傾向にあるため，探索中のノードを頂点とする部分木の中で最短の処理時間を持つアーキテクチャを表す葉は，探索中のノードで未定義のバス・パラメータに対して最大の値を割り当てた葉となる．したがって，処理時間の下限は式 (4) で表される．

$$lower\_time(\boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{crnt}}) = time(\boldsymbol{M_{sl}}, \boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{large}}) \quad (4)$$

ここで，$\boldsymbol{Arch_{crnt}}$ は探索中のノードが表す，いくつかのパラメータが未定義のバスアーキテクチャを表し，$\boldsymbol{Arch_{large}}$ は探索中のノードで未定義のパラメータに対して最大値を割り当てたバスアーキテクチャを表す．

探索中のノードを頂点とする部分木における処理時間が最短のアーキテクチャを表す葉が明らかであり，そのアーキテクチャの処理時間が探索済みの葉が表すアーキテクチャの最短処理時間よりも長い場合，その部分木には探索済みの葉が表すアーキテクチャの最小処理時間よりも短い処理時間を持つバスアーキテクチャは含まれない．そこで，式 (5) を満たすノードの子孫を枝刈りする．

$$lower\_time(\boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{crnt}}) > time(\boldsymbol{M_{sl}}, \boldsymbol{Arch_{fb}}, \boldsymbol{Arch_{fast}}) \quad (5)$$

ここで，$\boldsymbol{Arch_{fast}}$ は，探索済みの葉が表すアーキテクチャのうち，最短処理時間を持つアーキテクチャである．

2.4.3 バス・パラメータの探索順序

バストポロジが決定するまではバスの数が不明であるため，バス動作周波数およびバスビット幅は，バストポロジ探索後に探索する必要がある．

入力された面積制約が大きい場合，面積の下限による枝刈りをまったく行うことができない．一方，処理時間の下限による枝刈りは，ほとんどの場合に有効であると考えられる．以上の点より，処理時間の下限による枝刈りを最大限に利用できるように探索順序を決定する．

処理時間の下限による枝刈りを最大限に利用するためには，処理時間が短い傾向にあるパラメータから探索すべきである．そこで，提案手法では，処理時間に与える影響が少ない傾向にある，バッファ数の探索を最後に行う．

また，バス動作周波数とバスビット幅が処理時間に与える影響は，入力として与えられるそれぞれの候補に依存するため，どちらを先に探索すべきかは一意に決定できない．提案手法では，バス動作周波数を先に探索することとした．

以上より，提案手法では，(1) バストポロジ，(2) バス動作周波数，(3) バスビット幅，(4) バッファ数，の順序でバス・パラメータを探索する．

2.4.4 各パラメータの探索順序

各パラメータでの探索順序も探索時間に大きな影響を与える．

バスでのデータ転送量が多いほど，動作周波数などのパラメータ値による処理時間やハードウェア面積のばらつきが大きい傾向にあるため，広範囲の枝刈りが可能である．そこで，提案手法では，見積りのためのプロファイリング結果を用いて，データ転送量の多いチャネルおよびバスから順にパラメータの割当てを行う．

また，バスの数，バス動作周波数，バスビット幅，およびバッファ数が多いほど，システム全体の処理時間は短くなる傾向にあるため，提案手法では，それらの値が大きいものから順に割り当てる．

パラメータ値が大きいものから順に探索することで，バス動作周波数およびバスビット幅探索において式 (5) を満たす場合，探索前の兄弟は，探索中のノードよりも小さい動作周波数およびビット幅を持つため，最適解を含まない．したがって，探索前の兄弟も枝刈りすることで，実際に探索するノード数を削減する．

2.4.5 バッファ数最適化における枝刈り

バストポロジ，バスデータ転送速度，バスビット幅を固定し，バッファ数のみを変化させた場合の処理時間を比較したところ，ほとんどの場合，差がないことが分かった．処理時間が同じであれば，面積が小さい，すなわち，バッファ数の少ないアーキテクチャがより良い解となる．そこで，バッファ数探索木の各ノードにおいて，未定義のバッファ数に対して最大バッファ数を割り当てたバスアーキテクチャと，1 を割り当てたバスアーキテクチャの処理時間を比較し，差がない場合は 1 を割り当てたバスアーキテクチャのみを探索対象とする．

3. 評価実験

提案する最適バスアーキテクチャ探索手法の有効性を示すため，提案手法をデータ分割処理システムと音声映像圧縮システムに適用し，評価実験を行った．

データ分割処理システムのシステムレベルモデルを**図 5** に示す．データ分割処理システムは，6 個のプロセスと 8 本のチャネルを持つ．対象とした機能ブロック群構成を**表 1** に示す．$\boldsymbol{Arch_{fb1\_1}}$，$\boldsymbol{Arch_{fb1\_2}}$ は

表 2 音声映像圧縮システムの機能ブロック群構成
Table 2 Functional block sets of the experimental target system 2.

機能ブロック群構成	プロセスのマッピング						
	P_{AI}	P_{CCD}	P_{AV_MUX}	P_{RAM_A}	P_{RAM_V}	P_{MP3}	P_{JPEG}
$\boldsymbol{Arch_{fb2\_1}}$	F_{AI}	F_{CCD}	F_{AV_MUX}	F_{RAM_A}	F_{RAM_V}	F_{MP3}	F_{JPEG}
$\boldsymbol{Arch_{fb2\_2}}$	F_{AI}	F_{CCD}	F_{AV_MUX}	F_{RAM}		F_{MP3}	F_{JPEG}
$\boldsymbol{Arch_{fb2\_3}}$	$F_{AI/CCD/AV\_MUX}$			F_{RAM}		F_{MP3}	F_{JPEG}

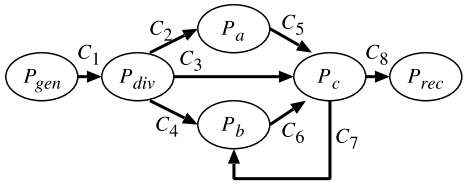


図 5 データ分割処理システムのシステムレベルモデル
Fig. 5 System-level model of the experimental target system 1.

表 1 データ分割処理システムの機能ブロック群構成
Table 1 Functional block sets of the experimental target system 1.

機能ブロック群構成	プロセスのマッピング					
	P_{gen}	P_{div}	P_{rec}	P_a	P_b	P_c
$\boldsymbol{Arch_{fb1\_1}}$	F_{gen}	F_{div}	F_{rec}	F_a	F_b	F_c
$\boldsymbol{Arch_{fb1\_2}}$	F_{gen}	F_{div}	$F_{a/rec}$		F_b	F_c

それぞれ，6 個，5 個の機能ブロックを持つ．

音声映像圧縮システムのシステムレベルモデルを**図 6** に示す．音声映像圧縮システムは，7 個のプロセスと 6 本のチャネルを持つ．また，対象とした機能ブロック群構成を**表 2** に示す．$\boldsymbol{Arch_{fb2\_1}}$，$\boldsymbol{Arch_{fb2\_2}}$，$\boldsymbol{Arch_{fb2\_3}}$ はそれぞれ，7 個，6 個，4 個の機能ブロックを持つ．

各機能ブロックの面積と処理時間を**表 3** に示す．

バスアーキテクチャ探索の入力は次のとおりとした．

- バス動作周波数：1 MHz または 2 MHz
- バスビット幅：16 ビットまたは 32 ビット
- 最大バッファ数：2

また，面積制約を与えた場合と，面積制約なし，すなわち面積制約を無限大とした場合に対して実験を行った．

本実験は，Pentium 4（2.4 GHz）プロセッサ，768 MB メモリの PC 上で行い，プログラムのコンパイルには GNU コンパイラ gcc–2.96 を用いた．

3.1 提案手法と全探索の探索時間の比較

データ分割処理システムに対して全探索，提案手法を適用した場合の探索ノード数などを**表 4** に，音声映像圧縮システムに対して全探索，提案手法を適用した場合の探索ノード数などを**表 5** に示す．「全探索」はすべてのアーキテクチャ候補を探索した場合，「制約なし」，「$A_{const} = 3$」，「$A_{const} = 100$」はそれぞれ，面積制約を無限大，3 [mm^2]，100 [mm^2] として提案手法を用いた場合を表す．「探索バスアーキテクチャ数」，「探索ノード数」，「見積り回数」はそれぞれ，バス・パラメータ探索時にトレースした葉の数，トレースしたノードの総数，探索時に性能見積りを行った回数を表す．「探索時間」は探索に要した時間を表す．ここで，「全探索」の探索時間は非常に長いため，高速処理時間見積り手法における見積り時間の平均値である 0.01 秒と，見積り回数の積として見積もった．

実験結果より提案手法は，全探索と比較して，面積制約の値によらず探索時間を大幅に減少させることが分かった．面積の下限による枝刈りも多少行われているが，処理時間の下限による枝刈りがより大きな範囲

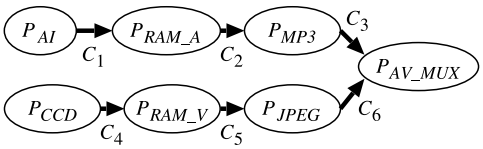


図 6 音声映像圧縮システムのシステムレベルモデル
Fig. 6 System-level model of the experimental target system 2.

表 3 各機能ブロックの面積と処理時間
Table 3 Hardware area and execution time of each functional block.

機能ブロック名	面積 [mm^2]	処理時間 [μs]
F_{gen}	0.039	3.0
F_{div}	0.032	2.5
F_a	0.019	3.0
F_b	0.045	4.0
F_c	0.065	7.0
F_{rec}	0.039	3.0
$F_{a/rec}$	0.065	4.5
F_{AI}	1.296	3.0
F_{MP3}	3.629	6.9
F_{CCD}	1.296	3.0
F_{JPEG}	5.864	6.9
F_{AV_MUX}	1.296	6.0
F_{RAM_A}	0.648	4.0
F_{RAM_V}, F_{RAM}	51.840	4.0
$F_{AI/CCD/AV\_MUX}$	1.944	6.0

表 4 探索時間と探索ノード数の比較（データ分割処理システム）
Table 4 Comparison of exploration time and the number of explored bus architectures (Target 1).

探索 方法	探索バスアーキ テクチャ数	探索 ノード数	見積り 回数	探索 時間
全探索	208,951,050,240	417,902,092,200	208,951,050,240	約 67 年
制約なし	3,327	93,462	127,826	230.37 秒
$A_{const} = 3$	3,246	91,186	124,713	224.76 秒

表 5 探索時間と探索ノード数の比較（音声映像圧縮システム）
Table 5 Comparison of exploration time and the number of explored bus architectures (Target 2).

探索 方法	探索バスアーキ テクチャ数	探索 ノード数	見積り 回数	探索 時間
全探索	173,490,176	346,979,946	173,490,176	約 20 日
制約なし	1,496	29,318	41,928	68.87 秒
$A_{const} = 100$	1,459	28,604	40,907	67.19 秒

表 6 バス・パラメータの探索順序による探索時間と探索ノード数の比較
Table 6 Comparison of exploration time and the number of explored bus architectures for different exploration order.

探索 順序	探索バスアーキ テクチャ数	探索 ノード数	見積り 回数	探索 時間
R→W→B	1,496	29,318	41,928	68.87 秒
B→R→W	5,984	117,272	167,712	27.95 分

を枝刈りできていることが分かる．以上より，高速見積り手法を用いた最適バスアーキテクチャ探索手法は，最適なバスアーキテクチャを短時間で発見できるといえる．

3.2 バス・パラメータの探索順序による探索時間の変化

音声映像圧縮システム，機能ブロック群構成 $\boldsymbol{Arch_{fb2\_3}}$ を対象に，面積制約なしの場合の，バス・パラメータの探索順序が異なる場合の探索ノード数などを**表 6** に示す．「R→W→B」が提案手法である，データ転送速度，データビット幅，バッファ数の順に探索した場合を表し，「B→R→W」がバッファ数，データ転送速度，データビット幅の順に探索した場合を表す．

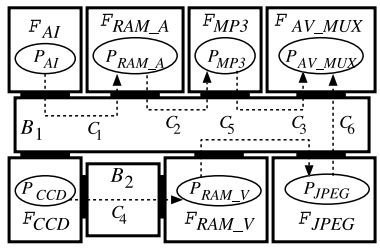
表 6 に示すとおり，提案手法はバッファ数を初めに探索する場合に比べて，探索ノード数が少なく，探索時間が短いことが分かる．バッファ数が処理時間に与える影響は，バスデータ転送速度やバスビット幅に比べて小さいため，バッファ数を初めに探索すると探索木の浅い部分で枝刈りがあまりできず，結果的に探索ノード数が増加してしまう．以上の点より，提案手法での探索順序を採用することで，効率的な枝刈りが行えるといえる．

3.3 異なる機能ブロック群構成での見積り結果の比較

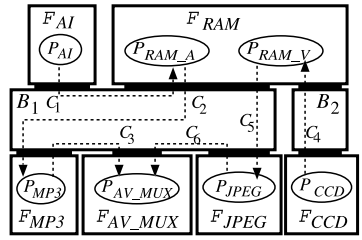
音声映像圧縮システムにおける，$\boldsymbol{Arch_{fb2\_1}}$，$\boldsymbol{Arch_{fb2\_2}}$，$\boldsymbol{Arch_{fb2\_3}}$ の 3 種類の機能ブロック構成に対する，面積制約なしの場合の探索結果のアーキテクチャを**図 7** に，最適アーキテクチャの処理時間と面積の見積り結果を**表 7** に示す．

最適アーキテクチャのバスはすべて 32 ビット，2 MHz 駆動という結果であった．$\boldsymbol{Arch_{fb2\_1}}$ および $\boldsymbol{Arch_{fb2\_2}}$ は 2 本のバスを持つ，データ転送の競合が起きないアーキテクチャが選択されている一方，$\boldsymbol{Arch_{fb2\_3}}$ は全チャネルでの 1 本のバスを共有するバスアーキテクチャが最適となった．これは，機能ブロックの処理時間がネックとなるため，バスを共有してもシステム全体の処理時間には影響しないからであった．

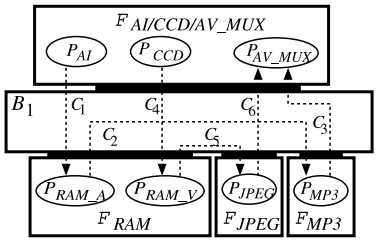
3 種類の機能ブロック群構成の最適アーキテクチャを比較すると，$\boldsymbol{Arch_{fb2\_2}}$ が最適なアーキテクチャであることが分かる．$\boldsymbol{Arch_{fb2\_1}}$ に比べて $\boldsymbol{Arch_{fb2\_2}}$ は，RAM 1 個分ハードウェア面積が少なく，処理時間は同じであるため，より良い構成であるといえる．$\boldsymbol{Arch_{fb2\_2}}$ では RAM を共有しているが，RAM へのアクセスが競合しないためにシステム全体の処理時間には影響していない．また，$\boldsymbol{Arch_{fb2\_3}}$ は機能ブ



(a) **Arch** $_{fb2\_1}$



(b) **Arch** $_{fb2\_2}$



(c) **Arch** $_{fb2\_3}$

図 7 探索結果の最適アーキテクチャ
Fig. 7 Architectures of exploration results.

表 7 最適アーキテクチャの面積と処理時間
Table 7 Estimation results of hardware area and execution time of optimal architectures.

機能ブロック群構成	面積 [mm^2]	処理時間 [sec]
$\boldsymbol{Arch_{fb2\_1}}$	65.89	29.08
$\boldsymbol{Arch_{fb2\_2}}$	65.24	29.08
$\boldsymbol{Arch_{fb2\_3}}$	63.30	38.08

ロック $F_{AI/CCD/AV\_MUX}$ が 3 個のプロセス P_{AI}, P_{CCD}, P_{AV_MUX} を実行するため，実行待ちとなるプロセスが存在し，その結果，システム全体の処理時間が長くなってしまう．

以上の結果から，提案する最適バスアーキテクチャ探索手法を用いて異なる機能ブロック群構成に対するバスアーキテクチャを最適化，評価することができるため，提案手法は最適なアーキテクチャの選択に有用であることが分かる．

4. おわりに

本論文では，システムレベルのプロファイリング結果を用いた最適バスアーキテクチャ探索手法を提案した．バスアーキテクチャ最適化問題を定式化し，最適なバス・パラメータの探索手法を提案した．実験結果より，提案手法は最適バスアーキテクチャを短時間で探索可能であることが分かった．

今後の課題は，バスのハードウェア面積の見積り，消費電力見積り，および，各種設計品質見積りを用いたアーキテクチャ最適化手法の開発である．

謝辞 本研究を進めるにあたり貴重なコメントをいただいた，株式会社半導体理工学研究センターの宮本俊介氏，株式会社富士通研究所の藤田隆司氏，ソニー株式会社の柿本勝氏，シャープ株式会社の大西充久氏，三洋電機株式会社の大山達史氏，鶴岡工業高等専門学校の佐藤淳助教授，大阪大学今井研究室の諸氏に深謝する．本研究の一部は，株式会社半導体理工学研究センターとの共同研究（研究番号 202）による．

参 考 文 献

1) Gajski, D.D.: IP-based methodology, *Proc. 36th Design Automation Conference* (*DAC 1999*), p.43 (1999).
2) Balarin, F., Lavagno, L., Passerone, C., Sangiovanni-Vincentelli, A., Watanabe, Y. and Yang, G.: Concurrent execution semantics and sequential simulation algorithms for the Metropolis meta-model, *Proc. 10th International Symposium on Hardware/Software Codesign* (*CODES2002*), pp.13–18 (2002).
3) Buch, J., Ha, S., Lee, E.A. and Messerschmitt, D.G.: Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems, *International Journal of Computer Simulation*, Vol.4, pp.155–182 (1994).
4) Takahashi, M., Miyajima, H. and Fukui, M.: An Efficient Power and Performance Evaluation Method with Reconfigurable Bus Architecture Model, *Proc. 11th Workshop on Synthesis and System Integration of Mixed Information Technologies* (*SASIMI2003*), pp.345–350 (2003).
5) Yoo, S., Nicolescu, G., Gauthier, L. and Jerraya, A.A.: Automation Generation of Fast Timed Simulation Models for Operating Systems in SoC Design, *Proc. Design Automation and Test in Europe 2002* (*DATE 2002*), pp.620–627 (2002).
6) Ueda, K., Sakanushi, K., Takeuchi, Y. and Imai, M.: Architecture-level performance estimation for IP-based embedded systems, *Proc. Design Automation and Test in Europe 2004* (*DATE 2004*), pp.1002–1007 (2004).
7) Lahiri, K., Raghunathan, A. and Dey, S.: De-

sign space exploration for optimizing on-Chip communication architectures, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.23, No.6, pp.952–961 (2004).
8) Synopsys, Inc.: Synopsys. http://www.synopsys.com/

(平成 16 年 11 月 18 日受付)
(平成 17 年 4 月 1 日採録)

上田　恭子

平成 13 年大阪大学基礎工学部情報科学科卒業．平成 15 年大阪大学大学院情報科学研究科博士前期課程修了．現在，同大学院博士後期課程に在籍．主としてアーキテクチャの最適化に関する研究に従事．IEEE 会員．

坂主　圭史（正会員）

平成 9 年東京工業大学理工学部電気電子工学科卒業．平成 11 年東京工業大学大学院修士課程修了．平成 14 年同大学院博士後期課程修了．平成 14 年 4 月より大阪大学大学院情報科学研究科助手．博士（工学）．VLSI 設計アルゴリズムに関する研究に従事．IEEE，電子情報通信学会各会員．

米岡　昇

平成 15 年大阪大学基礎工学部情報科学科卒業．平成 17 年大阪大学大学院情報科学研究科博士前期課程修了．現在は株式会社富士通研究所に所属．主としてアーキテクチャレベルの設計品質見積り手法の研究に従事．

武内　良典（正会員）

昭和 62 年東京工業大学工学部電気・電子工学科卒業．平成 4 年東京工業大学大学院博士後期課程修了．博士（工学）．平成 8 年大阪大学大学院基礎工学研究科情報数理系専攻講師．現在，同大学院情報科学研究科助教授．ディジタル信号処理，VLSI 設計および VLSI CAD の研究に従事．IEEE，電子情報通信学会各会員．

今井　正治（正会員）

昭和 49 年名古屋大学工学部電気工学科卒業．昭和 54 年名古屋大学大学院博士後期課程修了（工学博士）．同年豊橋技術科学大学奉職．平成 6 年同教授．平成 8 年大阪大学大学院基礎工学研究科情報数理系専攻教授．現在，同大学院情報科学研究科教授．その間，昭和 59〜60 年にかけて米国サウスカロライナ大学工学部電気計算機工学科客員助教授（文部省在外研究員）．これまで組合せ最適化アルゴリズム，ハードウェア/ソフトウェア協調設計等の研究に従事．平成 3 年より日本電子機械工業会および IEEE/DASC において EDA 標準化作業に従事．現在，情報処理学会設計自動化研究会主査．IEEE，ACM，電子情報通信学会，人工知能学会各会員．