LC-003

バスブリッジを含む SoC アーキテクチャの設計空間探索手法
A Design Space Exploration Method for SoC Architecture Containing Bus Bridges

小橋　一寛† Kazuhiro Kobashi　坂主　圭史† Keishi Sakanushi　武内　良典† Yoshinori Takeuchi　今井　正治† Masaharu Imai

1. はじめに

近年半導体微細加工技術の進歩により，従来は複数個の LSI で実装していた大規模なシステムが，1 個の LSI で実装可能になっている．その一方で，設計対象のシステムは大規模化かつ複雑化により，設計コストは増大している．設計コストを抑える手法として，IP(Intellectual Property)と呼ばれる既設計の機能モジュールを利用し，IP とバスアーキテクチャの構成を決定する IP ベース設計手法[1]が提案されている．従来の設計手法に比べて，新規設計する機能モジュールが少なくなることによる設計コストの抑制が期待できる一方で，アーキテクチャの性能見積もりの困難さや設計空間の膨大さから，設計者が望む性能を持つ構成を人手で探索するには，依然として大きな設計コストがかかる．

また，SoC アーキテクチャの設計ではバスブリッジの活用が重要である．バスブリッジは複数のバスを接続し，異なるバスと接続する IP 間のデータ転送を可能にするモジュールである．バスブリッジを用いたアーキテクチャは，複数の IP 間のデータ転送を同時に行えるため，バスブリッジを用いていないアーキテクチャに比べて，データ転送時間が減少し，制約によってはアーキテクチャ全体の性能が向上することが知られている．実際に[2,3]などの SoC でバスブリッジを用いることによる性能の向上が図られている．

バスブリッジを扱う探索手法として[4,5]などが提案されており，これらによってバスブリッジを含むバスアーキテクチャの探索が可能であるが，コンポーネントの探索は考慮されていない．一方，アーキテクチャの探索手法としてシステムレベル・プロファイリングを用いた探索手法[6,7]が提案されている．この手法により短時間で多数のアーキテクチャを探索する事が可能であるが，バスブリッジを含むアーキテクチャの探索は考慮されていない．

そこで本研究では，[6,7]の探索手法を拡張し，バスブリッジを含むアーキテクチャにも対応した探索手法を提案する．提案手法では，[6,7]のアーキテクチャモデルや探索手法，性能見積もり手法を拡張し，バスブリッジを含むアーキテクチャの探索を可能にする．

以下本稿の構成について述べる．第 2 節で [6,7]の手法を説明し，第 3 節で提案手法を説明する．第 4 節で評価実験について述べた後，第 5 節でまとめを行う．

2. システムレベル・プロファイリングを用いたアーキテクチャ探索手法

本節は[6,7]の手法について説明する．[6,7]では，対象システムのシステムレベルモデルを解析し，アーキテクチャに依存しないシステムの動作を表現するシステムレベル実行順序グラフを構築する．次に様々なアーキテクチャレベルモデルを生成し，アーキテクチャレベルモデルとシステムレベル実行順序グラフから，アーキテクチャ上でのシステムの動作を表現するアーキテクチャレベル実行依存グラフを構築する．この実行依存グラフを解析して，そのアーキテクチャで対象システムを実行したときの性能を見積もる．探索結果から，トレードオフ関係にあるアーキテクチャを提示する．

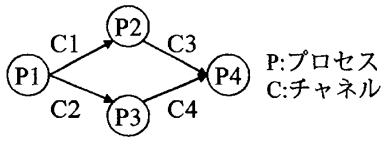

図 1 システムレベルモデルの例

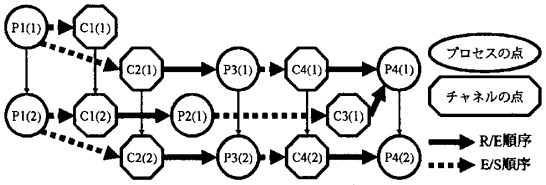

図 2 SL-EOG の例

2.1. システムレベルモデル

[6,7]では，対象システムのシステムレベルモデルを，データ処理を示すプロセスと，プロセス間のデータ転送を示すチャネルで表現する．プロセス及びチャネルは設計者が与える固有の優先度を持つ．プロセスを示す点 *Pi* とチャネルを示す有向辺 *Ci*(*i*=1,2,3,4)で構成するシステムレベルモデルの例を図 1 に示す．

2.2. システムレベル実行順序グラフ

[6,7]では，プロセスは処理に必要な全てのデータをチャネルから受信した後にデータ処理を行い(R/E 順序:Receive/Execute 順序)，チャネルはプロセスの処理が完了した後にデータ転送を行う(E/S 順序:Execute/Send 順序)順序関係が存在する．そこで，SystemC[8]で記述された対象システムのシステムレベルモデルをコンパイルして実行することで解析し，取得した実行順序とデータ転送量を基に，アーキテクチャに依存しないシステムの動作を表現するシステムレベル実行順序グラフ(SL-EOG:System Level Execution Order Graph)を構築する．

SL-EOG は，*i* 番目のプロセスの *j* 回目の処理を表す点 *Pi(j)* と *k* 番目のチャネルの *l* 回目の処理を表す点 *Ck(l)*を点集合として持つ．また，R/E 順序関係及び E/S 順序関係を表す辺と，*Pi(j)*と *Pi(j+1)*間及び *Ck(l)*と *Ck(l+1)*間の辺を辺集合として持つ．図 1 のシステムレベルモデルを解析して構築した SL-EOG の例を図 2 に示す．

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

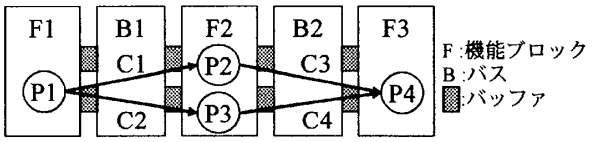

図 3 アーキテクチャレベルモデルの例

2.3. アーキテクチャレベルモデル

[6,7]では，対象システムを処理するアーキテクチャのアーキテクチャレベルモデルを，データを処理する機能ブロック，機能ブロック間のデータを転送するバス，データを一時的に格納するバッファで構成する．アーキテクチャレベルモデルの例を図 3 に示す．

機能ブロックには複数個のプロセスがマッピングされ，ある時間において処理可能なプロセスから優先度の高い 1 個のプロセスを処理する．ただし，1 個のプロセスの処理が完了するまでは他のプロセスを処理しないとする．また機能ブロックは，データベースに登録された IP から選択する．データベースには IP ごとにマッピング可能なプロセスと処理サイクル数，機能ブロックの動作周波数や動的消費電力及び静的消費電力が登録されている．

バスには複数個のチャネルがマッピングされ，ある時間において処理可能なチャネルから優先度の高い 1 個のチャネルを処理する．ただし，1 個のチャネルの処理が完了するまでは他のチャネルを処理しないとする．

バッファはチャネル毎に機能ブロックとバスの間に複数個存在し，処理前もしくは転送前のデータを一時的に格納する．1 個のバッファには 1 回のデータ転送で転送されるデータが格納されるとする．またバッファは，機能ブロックが受信したデータを格納する受信バッファと，機能ブロックの処理によって生成されたデータを格納する送信バッファに区別される．

2.4. アーキテクチャレベル実行依存グラフ

[6,7]では，アーキテクチャレベルモデルにシステムレベルモデルをマッピングすることで，プロセスとチャネルの間にシステムレベルモデルでは存在しなかった依存関係が生じる．そこでシステムレベルモデルにおける順序関係を表す SL-EOG に依存関係を表す有向辺を付加し，アーキテクチャレベル実行依存グラフ(AL-EDG:Architecture Level Execution Dependency Graph)を構築する．

バッファは，データの格納を始めた時に使用中となり，バッファ内の全てのデータが処理(転送)された時に空きとなる．したがって，チャネルは受信バッファが空くまで次のデータを転送できないため，アーキテクチャレベルモデルが決定することでプロセスとデータ転送の間に依存関係(E/R 依存:Execute/Receive 依存)が生じる．同様に，プロセスは送信バッファが空くまで次のデータを処理できないため，アーキテクチャレベルモデルが決定することでチャネルとデータ処理の間に依存関係(S/E 依存:Send/Execute 依存)が生じる．これらの依存関係はプロセスの処理データ数(チャネルの転送データ数)とチャネルの送信バッファ(受信バッファ)の個数から決定する．

AL-EDG は，SL-EOG に対して E/R 依存関係及び S/E 依存関係にあるプロセスとチャネルの間に有向辺を張ることで得られる．図 2 の SL-EOG と図 3 のアーキテクチャレベルモデルを基に構築した AL-EDG を図 4 に示す．

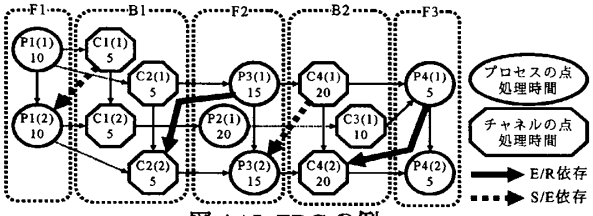

図 4 AL-EDG の例

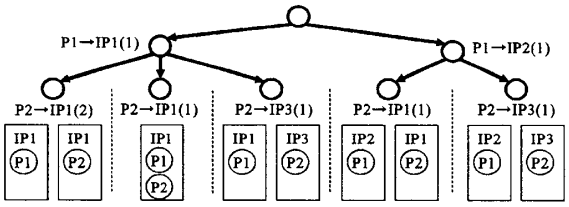

図 5 プロセスマッピングの探索木の例

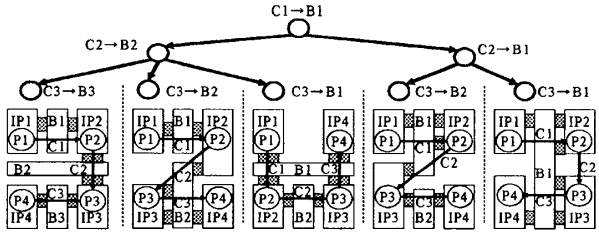

図 6 チャネルマッピングの探索木の例

2.5. アーキテクチャの実行時間見積もり手法

アーキテクチャの性能見積もりは AL-EDG を解析することで行う．AL-EDG の解析は，機能ブロック及びバス毎に他の点からの有向辺を持たない点を 1 つ選択し，その中から最短の実行時間を進め，実行時間が 0 になった点とその点からの有向辺を削除することを，AL-EDG から点がなくなるまで繰り返し行う．解析終了時の経過時間が実行時間の見積もり値である．

2.6. アーキテクチャ探索手法

[6,7]では探索木を走査して，アーキテクチャレベルモデルを構築する．分枝限定法で効率よく探索することを考慮して，探索木は根から順にプロセスマッピング，チャネルマッピング，バスビット幅，機能ブロックの動作周波数，バスの動作周波数，チャネル毎のバッファの個数に対応する点で構成し，深さ優先探索で走査する．

プロセスマッピングはプロセスを処理する機能ブロックを決定する．マッピングする機能ブロックはそのプロセスを処理可能な IP をデータベースから選択する．プロセス *P1* を *IP1* と *IP2* に，プロセス *P2* を *IP1* と *IP3* にマッピング可能なときのプロセスマッピングの例を図 5 に示す．

チャネルマッピングはチャネルを処理するバスを決定する．アーキテクチャに含まれるバスの最大本数は設計者が与える．プロセス *Pi* を *IPi*(*i*=1,2,3,4)にマッピングした時のチャネルマッピングの例を図 6 に示す．

探索木の走査によって構成したアーキテクチャレベルモデルは AL-EDG を解析して実行時間を，さらに面積や消費電力量も見積もって([6,7])評価する．

3. バスブリッジに対応したアーキテクチャ探索手法の概要

[6,7]の手法は，1 本のバスでデータ転送をおこなうため，バスブリッジを経由するデータ転送は考慮されていない．

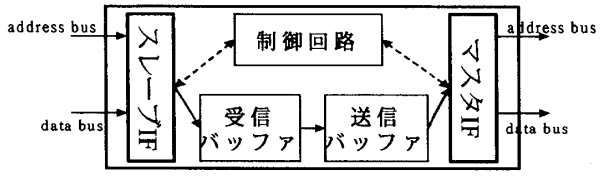

図 7 バスブリッジモデル

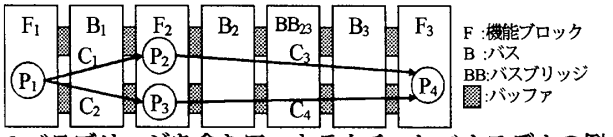

図 8 バスブリッジを含むアーキテクチャレベルモデルの例

そこで提案手法では，新たにバスブリッジモデルを[6,7]のアーキテクチャモデルに導入し，チャネルマッピングを拡張して，探索木の走査によるバスブリッジを含むアーキテクチャレベルモデルの構成を可能にする．また，[6,7]の実行時間見積もりに用いる AL-EDG の構築手順及び解析手順を拡張し，バスブリッジを含むアーキテクチャレベルモデルの性能見積もりを可能にする．

3.1. バスブリッジに対応したアーキテクチャレベルモデル

本研究で導入するバスブリッジは，インタフェース(マスタ，スレーブ)，バスブリッジの動作を制御する回路，転送するデータを一時的に格納するバッファで構成される．バスブリッジの動作はスレーブインタフェースで受信したデータを受信バッファに格納し，送信バッファに転送する．次に送信バッファのデータをマスタインタフェースで送信する．バスブリッジモデルを図 7 に示す．

バスブリッジのバッファは送信バッファと受信バッファに区別され，それぞれのバッファは複数個のバッファで構成された FIFO(First In First Out)バッファとして動作する．各バッファの個数は探索木の走査で決定し，1 個のバッファには 1 つのデータが格納されるとする．バスブリッジのバッファの個数の候補は設計者が与える．

提案手法ではアーキテクチャレベルモデルを機能ブロック，バス，バッファ，バスブリッジで構成する．バスブリッジを含むアーキテクチャレベルモデルを図 8 に示す．

機能ブロックからバスブリッジへのデータ転送は，バスブリッジの受信バッファが空いているときに行う．また，バスブリッジの受信バッファから送信バッファへのデータ転送は，送信バッファが空いているときに行う．

3.2. バスブリッジに対応したアーキテクチャレベル実行依存グラフ

バスブリッジから機能ブロックへのデータ転送は転送するデータが送信バッファに存在し，かつ転送先の受信バッファが空いているときに行う．したがって，プロセスとバスブリッジからのデータ転送の間には，プロセスとデータ転送の間と同様に E/R 依存関係が存在する．同様にバスブリッジからのデータ転送とプロセスとの間には S/E 依存関係が存在する．

本研究では，バスブリッジを含むアーキテクチャレベルモデルの順序関係及び依存関係を表現するために，[6,7]の AL-EDG を拡張した．提案手法では SL-EOG に対して，送信元と受信先のバスが異なるチャネルの点を分割し，機能ブロックからバスブリッジへのデータ転送を示す点と，バスブリッジから機能ブロックへのデータ転送を示す点を生成する．次にそれらと有向辺で接続される，バスブリッジの受信バッファから送信バッファへのデータ転送を示す点を生成する．さらに E/R 依存関係にあるプロセスとチャネルの間と，S/E 依存関係にあるチャネルとプロセスの間に辺を張って AL-EDG を構築する．図 2 の SL-EOG と図 8 のアーキテクチャレベルモデルを基に構築した AL-EDG の例を図 9 に示す．

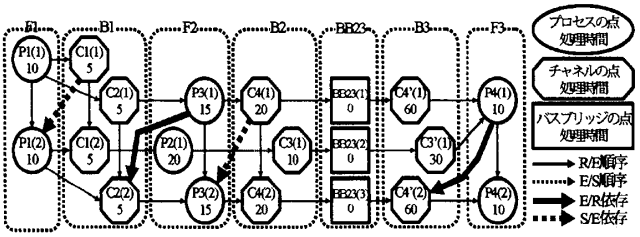

図 9 バスブリッジに対応した AL-EDG の例

3.3. バスブリッジに対応したアーキテクチャ実行時間見積もり手法

バスブリッジを含むアーキテクチャの実行時間を見積もるために，[6,7]の AL-EDG の解析手法を拡張した．提案手法では，バスブリッジの受信バッファから送信バッファに転送する点と，データ処理とバスブリッジのデータ転送間の依存関係の扱いを新たに定めた．バスブリッジに対応した AL-EDG の解析手順は以下の通りである．

i. 実行時間を示す変数 T を 0 に設定する．
ii. 他の点からの有向辺が存在しない点を実行可能点とする．ただし，バスブリッジへのデータ転送は受信バッファに空きがある点を実行可能点とする．
iii. 機能ブロック及びバス毎にマッピングされた実行可能点から，優先度の高い実行可能点を実行点とする．また送信バッファに空きがあるバスブリッジ毎に，受信バッファの先頭に格納されているデータを送信バッファに転送する点を実行点とする．実行点が存在しない時はデッドロックが生じたとして解析を中止する．
iv. 全ての実行点の中で最短の実行時間を経過時間 T に加え，各実行点の残りの実行時間から最短の実行時間を引く．
v. 残り実行時間が 0 になったノードとそのノードから引いている有向辺を AL-EDG から削除し，ii.に戻る．

3.4. バスブリッジに対応したアーキテクチャ探索手法

バスブリッジを含むアーキテクチャを構成するために，[6,7]の探索木のチャネルマッピングを拡張し，バスブリッジの受信バッファ(送信バッファ)の個数のマッピングを探索木に導入した．

探索木の走査によってバスブリッジを含むアーキテクチャを構成するために，チャネルマッピングをチャネルの送信元と受信先のそれぞれに対してバスをマッピングし，チャネルの送信元と受信先のバスが異なるときにそのバス間を接続するバスブリッジを配置する．拡張したチャネルマッピングの例を図 10 に示す．

バスブリッジのバッファの個数はバスブリッジへのデータ転送やバスブリッジからのデータ転送の待ち時間に大きな影響を及ぼすため，バスブリッジ毎に受信バッファ(送信バッファ)の個数も探索木を用いて探索する．分枝

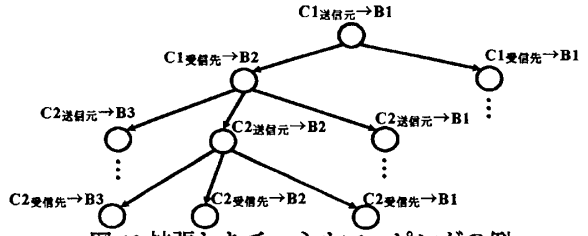

図 10 拡張したチャネルマッピングの例

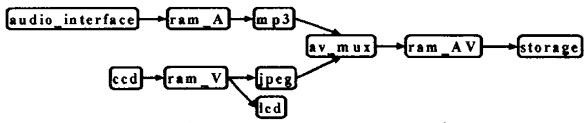

図 11 対象のシステムレベルモデル

限定法を用いて効率良く探索するために，バスの動作周波数の次にバスブリッジのバッファ数を決定するように探索木を構成する．

4. 評価実験

本節では，提案手法に基づくプログラムを用いて，バスブリッジを含むアーキテクチャを考慮した設計空間を探索した結果を示す．実験環境は CPU が Intel Pentium4 2.80GHz，メモリが 4GB，OS が Fedora core 6 であり，図 11 に示す音声/動画エンコードシステムを対象システムとした．また，データベースに登録されている IP を表 1 に示し，探索に必要なパラメタの候補を以下と与えた．

- アーキテクチャに含まれる最大バス本数：2
- アーキテクチャに含まれる最大バスブリッジ数：1
- バスのビット幅候補：16,32(bit)
- バスの動作周波数候補：100,120(MHz)
- バスブリッジ及びチャネルのバッファ数の候補：1,2

なお，制約としてプロセス ram_A，ram_V，ram_AV は機能ブロック(IP9)にマッピングし，固定した．

本実験では，探索木を全て走査するとアーキテクチャレベルモデルの数が爆発するため，2 段階で探索を行った．まず全ての IP とバスアーキテクチャの組み合わせに対して実行時間と面積の上限と下限を見積もり，探索結果から図 12 に示すバスブリッジを含むアーキテクチャの構成と，バスブリッジを含まないアーキテクチャの構成を選択した．次に選択したアーキテクチャの構成それぞれに対してバスビット幅などのパラメタを探索した．実験結果から作成したトレードオフ曲線を図 13 に示す．

本実験ではバスブリッジのバッファによる面積の影響が大きいため，図 13 から面積の制約が厳しいときはバスブリッジを含まないアーキテクチャの方が性能は良いが，面積の制約を緩くすると，バスブリッジを含むアーキテクチャの方が性能は良くなることが確認できた．以上のことから，提案手法は従来手法より良いトレードオフ曲線を取得できる事を確認した．

5. おわりに

本研究は[6,7]のアーキテクチャ探索手法を拡張してバスブリッジを含む SoC アーキテクチャに対応した探索手法を提案し，評価実験によって提案手法の有効性を確認した．今後の課題としては探索時間の短縮や探索したアーキテクチャの自動合成手法の確立が挙げられる．

表 1 データベース

IP name	Hardware area [gate]	Exec. Freq. candidates [MHz]	Executable Process ($\times 10^3$cycle)
IP_1	15000	300, 400	av_mux(90)
IP_2	40000	250	mp3(150), jpeg(200)
IP_3	20000	300	mp3(100)
IP_4	20000	300	jpeg(150)
IP_5	5000	100	audio_interface(15)
IP_6	5000	100	ccd(25)
IP_7	5000	100	lcd(25)
IP_8	5000	100	storage(15)
IP_9	150000	80	ram_A(4), ram_V(4), ram_AV(4)

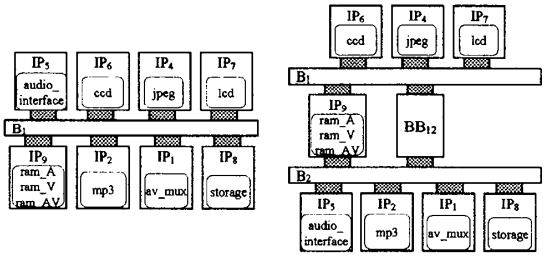

図 12 選択した IP とバスアーキテクチャ

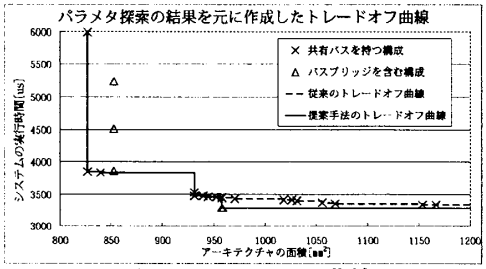

図 13 トレードオフ曲線

参考文献

[1] Daniel Gajski, “IP-based Design Methodology”, In Proceeding of 36th Design Automation Conference (36th DAC), pp.43, June 1999.

[2] デジタル AV 向け周辺機能搭載 SoC(MB93461), http://img.jp.fujitsu.com/downloads/jp/jed/brochures/find/23-2j/12-17.pdf

[3] シングルチップマイクロコントローラ(ML674000), http://www.okisemi.com/jp/dbps_data/_material_/datasheet/images_datasheet/FJDL674000-01.pdf

[4] Sudeep Pasricha, Nikil Dutt, Mohamed Ben-Romdhane, “Fast exploration of bus-based on-chip communication architectures”, Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis(CODES+ISSS2004), pp.242-247, September, 2004.

[5] Kanishka Lahiri, Anand Raghunathan, Sujit Dey, “Design Space Exploration for Optimizing On-Chip Communication Architectures”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.23, no.6, June 2004.

[6] Kyoko Ueda, “An Embedded System Design Methodology based on System-level Profiling”, Doctoral Dissertation, Graduate School of Information Science and Technology, Osaka University, 2006.

[7] 上田恭子, 坂主圭史, 米岡昇, 武内良典, 今井正治, 「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」, 情報処理学会論文誌, Vol.46, No.6, pp.1374-1382, 2005 年 6 月.

[8] “SystemC Language Reference Manual”, http://www.systemc.org/.