

大規模 SoC バス・アーキテクチャ性能評価手法

高橋 美和夏[†] 宮 嶋 浩 志[†] 福 井 正 博^{††}

本稿では、数多くのバス・アーキテクチャを短期間で評価できる性能解析手法を提案する。提案手法は、シミュレーション時にアーキテクチャ構成を自由に変更でき、かつ、各コンポーネントが正確な消費電力やコスト性能のモデルを持っているため、従来の評価手法に比べて、候補となるアーキテクチャの性能評価を高効率で行うことができる。本手法を実際の大規模 SoC 設計に適用した事例によれば、従来の手法に比べ、約 2 倍の開発効率向上が可能となる。また、消費電力などの評価モデルを簡単に実装でき、必要に応じて種々の指標を比較評価できるため、より設計者の要求に近い結果を得ることができるようになる。

Performance Evaluation Model for SoC Bus Architecture Exploration

MIWAKA TAKAHASHI,[†] HIROSHI MIYAJIMA[†] and MASAHIRO FUKUI^{††}

This paper proposes a new performance analysis technique that many bus architectures can be evaluated for a short period. The proposal technique provides a flexible architecture model for simulation with accurate power and cost models. That reduces the time for performance evaluation of candidate architectures. By applying this technique to a real LSI design, we have improved the efficiency of the simulation by double. Also, due to the function to append a new metrics like power easily, we have got a close design to designers' needs.

1. はじめに

近年、システムオンチップの開発がさかんになっており、いずれのシステムにおいても開発競争が激しく、高性能化、低コスト化、多機能化、設計期間の短縮が強く要望されている。

仕様検討からハードウェア/ソフトウェアの分割、フロアプランの工程、いわゆる上流設計の工数は設計全体の約 20% 程度の工数にすぎないが、製品性能の約 80% はこの段階で決定する¹⁾。そのため、設計の最適化においては、この上流設計の段階で高精度な性能見積りを行うことが重要である。

システムオンチップの設計として、すでにアーキテクチャが確定しているプラットフォームに対する HW/SW 分割の検討²⁾や、アーキテクチャの再利用のために、定められた構造の中での部品の検討手法³⁾は提案され、定着しつつある。

しかしながら、多くのアーキテクチャの中から最適なものを選択するための性能評価用モデルを短期間で

開発する手法は提案されていない。様々な機能を輻輳させ高速に処理するための部品間のデータ転送を制御するバス・アーキテクチャの最適化は、性能、コスト、消費電力などに大きな影響を与えるため重要である。そのため、実状はプラットフォーム設計前の段階で、種々の異なるバス・アーキテクチャの比較検討が実施される。

バス・アーキテクチャの性能を簡単に解析できる環境としては、市販ツール^{4),5)}などが提供されているが、これらのような設計環境を利用しても、アーキテクチャ性能評価のためのモデル開発には多くの工数が必要となる。大規模かつ、設計期間短縮の要求が厳しいシステムオンチップの設計においては、このモデル開発工数の短縮が課題であり、しかも様々なアーキテクチャを検討するためには、モデルの高抽象度化、性能評価の高速化が課題である。

本稿では、数多くのバス・アーキテクチャを短期間で評価できる性能解析手法を提案する。高速検証のポイントである性能評価モデルにおいては、バス・アーキテクチャ評価に必要な性能評価指標を定義する。そして、従来のバス・アーキテクチャに依存した性能解析用シミュレーションモデルに対して、本手法では、バス・アーキテクチャに依存しないシミュレーション

[†] 松下電器産業株式会社

Matsushita Electric Industrial Co., Ltd.

^{††} 立命館大学

Ritsumeikan University

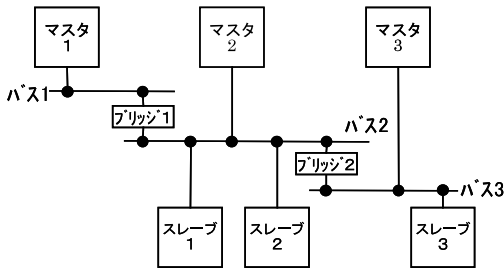


図1 バス・アーキテクチャ例
Fig.1 Example of bus architecture.

モデルを提案し、短期開発を実現していることを示す。2章で性能評価指標を定め、3章で従来の性能評価手法について、4章で提案する性能評価手法、5章で実験結果、6章でまとめとする。

2. 定義

本稿で扱うバス・アーキテクチャの構造と、同バス・アーキテクチャ上を移動するひとまとまりのデータ、いわゆるトランザクションについて説明し、その性能評価指標について説明する。

2.1 バス・アーキテクチャ

対象とするバス・アーキテクチャの例を図1に示す。バス・アーキテクチャは複数のバスとそれらに接続する機能ブロック、2つのバスを接続するブリッジから構成される。機能ブロックは、CPU、DSP、DMAなどのデータを能動的に転送するマスタと、メモリなどのデータを受動的に処理するスレーブに分けられる。

トランザクションは、マスタで駆動され、一定時間バスを占有して、スレーブへ移動する一連のデータの流れることをいう。その表現方法はシミュレーションモデルに依存するため、3章で詳しく述べる。

2.2 性能評価指標

提案する性能評価手法における評価指標を定義する。複数のバス・アーキテクチャに対して、マスタから発生されるトランザクションの移動が完了するまでの応答時間、トランザクションがバスを占有する占有率、スループットを観測する。

■ 応答時間

マスタから発生されるトランザクションが、バスを経由し、スレーブへのアクセス処理を終えるまでの時間である。基準単位はクロック数など評価目的に応じて定める。異なるマスタから発生されたトランザクションが、同時に同一のバス、あるいはスレーブにアクセスした場合は、バス、スレーブに定義されている調停方式に従って処理順序が決められる。

この順番を待つ時間がバスのボトルネックとなり、

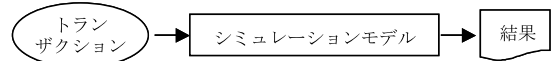


図2 性能評価フロー
Fig.2 Performance evaluation flow.

アプリケーション輻輳時の性能の劣化を示す。この改善のためには、異なる調停方式あるいは、異なるバス・アーキテクチャの選択が必要になる。

■ バス占有率

あるバスがトランザクションの経路のために占有されている割合を示す。システムを構成する複数のバスでバス占有率が極端に異なる場合は、異なる調停方式あるいは、異なるバス・アーキテクチャの検討が有効である。

■ バンド幅 (スループット)

単位時間に処理できるトランザクション量であり、バス容量を示す尺度である。

■ 消費電力

バス・アーキテクチャを構成する部品ごとに消費電力を定義する。マスタやスレーブでトランザクションを処理することによって消費される電力と、データ移動のためにバスやブリッジを充放電する電力、バスのプリチャージのための電力を加算したものである。

3. 性能評価手法

一般的な性能評価フローを図2に示す。対象とするシステムをモデル化したシミュレーションモデルに、トランザクションを入力することによって、結果を得ることができる。

本章では、このような評価フローを実現するために利用した環境、そして従来の一般的な性能評価のための性能評価モデルを説明する。

3.1 性能評価環境

性能評価シミュレーションは、事象 (event) の生起および終了の発生により系の状態が離散的に変化する離散型シミュレーションに属する⁵⁾。離散型シミュレーションのモデル化には、① 事象、② プロセス、③ アクティビティ、に着目する3種類がある。本稿では、② プロセス中心のモデル化を実現している環境を前提としている。

対象とするシステムは、複数のブロックダイアグラムを接続して実現する。ブロックダイアグラムは使用する言語によって異なるが、基本的にはトランザクションの移動を表現するために、トランザクションの発生、バスやメモリなどのリソース使用権利の獲得・開放、トランザクションの消滅を示す。図3に、一般的なブ

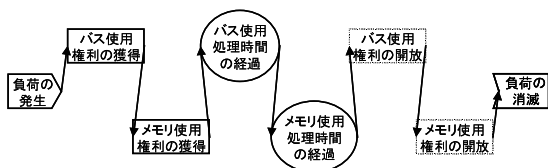


図 3 シミュレーションモデル例

Fig. 3 Example of simulation model.

表 1 構造型シミュレーションモデルの特徴

Table 1 Characteristics of structured simulation model.

Variable	
	部品性能
	ワークロード(トランザクションの発生時刻・量)
Fixed	
	ワークフロー

ロックダイヤグラムに基づき、バスを経由しメモリにアクセスするシステムのモデル化例を示す。

このモデル化されたブロックダイヤグラムに沿って移動する「もの」がトランザクションであり、待ち行列を構成する (a) future event chain あるいは (b) current event chain のいずれかにつながれている。(a) では、現在入っているブロックからの移動予定時刻が現在の event 時刻よりも後であるトランザクションを、移動予定時刻の順に並べる。(b) では、移動予定時刻が現在の event 時刻に等しいトランザクション、あるいはそれ以前に移動すべきであったが、条件が満たされなくて移動できないでいるトランザクションを優先度の高い順に並べ、同一の優先度を持つトランザクション間では移動予定時刻順に並べる。

3.2 従来の性能評価モデル

文献 4)、5) を用いた性能評価モデルは、対象とするシステムのトランザクションの移動経路を示すワークフローを、図 3 で示すように、ブロックダイヤグラムの接続によってモデル化している。そのワークフローへの入力、ワークフローを通過するトランザクションの発生時刻と発生量である。

以降、本稿では、従来のシミュレーションモデルを構造型シミュレーションモデルと定義する。構造型シミュレーションモデルの特徴を表 1 にまとめる。

構造型シミュレーションモデルの特徴は、構成する部品の性能と、トランザクションの発生時刻と発生量を示すワークロードを、シミュレーション実行時の変数として入力し、トランザクションの移動経路を示すワークフローを、シミュレーションモデルとして固定情報として扱うことである。ワークフローは、実現するバス・アーキテクチャに依存して異なるため、複数

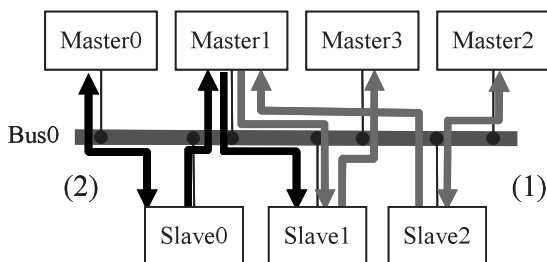


図 4 対象システム構成 (A) とワークフロー

Fig. 4 Workflow on bus architecture (A).

表 2 システム構成 (A) に対するワークフロー

Table 2 Workflow on bus architecture (A).

処理	手順	ワークフロー
(1)	エンコード	Slave2→Bus0→Master2→Bus0→Slave2
画像再生	補正	Slave2→Bus0→Master1→Bus0→Slave1
	表示	Slave1→Bus0→Master3
(2)	デコード	Slave0→Bus0→Master0→Bus0→Slave0
画像録画	補正・記録	Slave0→Bus0→Master1→Bus0→Slave1

のバス・アーキテクチャ評価には複数のシミュレーションモデルを用意する必要がある。

次に、構造型シミュレーションモデルを使用した性能評価手法の具体例を示す。図 4 に評価対象システムの構成 (A) および、灰色矢印でワークフロー (1) を、黒色矢印でワークフロー (2) を示す。

Bus0 に Master0, 1, 2, 3 と Slave0, 1, 2 が接続され、灰色矢印のワークフロー (1) は、たとえば画像再生のようなエンコード、補正、表示処理を示し、黒色矢印のワークフロー (2) は、画像録画のようなデコード、補正、記録処理を示す。具体的な処理とワークフローの対応を表 2 に示す。ここでのワークフローは、図 3 で示したような実際の具体的な権利の獲得・開放などのブロックダイヤグラムでの接続ではなく、トランザクションの順序を説明するために、マスタ、バス、スレーブなどの構成部品としている。

マスタ部品である Master0, 1, 2, 3 が、画像再生や、録画のためのデコード、画像処理、記録処理を行う。スレーブ部品である Slave0, 1, 2 は、これらの処理のためのデータを格納するメモリである。

画像再生処理を示すワークフロー (1) は、Master2 が Slave2 からデータを読み出し、エンコード処理を行い、エンコード処理結果を Slave2 に書き込む。Master2 によって書き込まれたデータを、次に、Master1 が画像補正を行い別のメモリである Slave1 に書き込む。そして Master3 がその書き込まれたデータを表示する。画像録画を示すワークフロー (2) は、デー

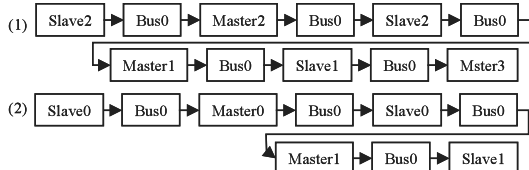


図5 従来のシミュレーションモデル (A)
Fig. 5 Conventional simulation model (A).

表3 性能評価結果 (A)
Table 3 Performance evaluation results.

		処理	待ち	合計
応答 時間	(1)画像再生	680	40	720
	(2)画像録画	880	320	1,200
占有率(%)	Bus0	33.3		

タが格納されている Slave0 から Master0 がデータを読み出し、デコード処理を行い、デコード処理結果を Slave0 に書き込む。書き込まれたデータを Master1 が Slave0 から読み出し、補正処理を行い、処理結果を Slave1 に書き込み保存する。マスタ、スレーブ間のデータ転送はすべて Bus0 を経由する。

図4のシステム構成(A)のワークフローに対応する従来のシミュレーションモデルを図5に示す。各々のワークフローを、マスタ、スレーブ、バスをブロックダイアグラムとして接続することによって実現する。シミュレーションモデル(A)での性能評価結果を表3に示す。

処理時間は、実際にトランザクションが移動している時間であり、待ち時間は、他のマスタが共通資源であるバスあるいはスレーブを占有しているために、トランザクションが資源の空きを待っている時間である。表3に示す性能評価結果から、応答時間として、(1)画像再生処理に720、(2)画像録画処理に1,200、そのうち、システム構成(A)に依存して消費される各々の待ち時間は、40と320であることが分かる。この待ち時間は、図4に示したシステム構成(A)から、Bus0の使用調停によることが予測できる。

次に、システム構成(A)の性能改善のために、新たなシステム構成(B)を検討する場合を示す。システム構成(B)を図6に示す。処理手順は改善前のシステム構成(A)と同様であるが、構成(A)と(B)では、バス・アーキテクチャを変更しているためにワークフローが異なる。そのために、シミュレーションモデルを新たに用意する必要がある。システム構成(B)での性能を評価するためのシミュレーションモデル(B)を図7に示す。

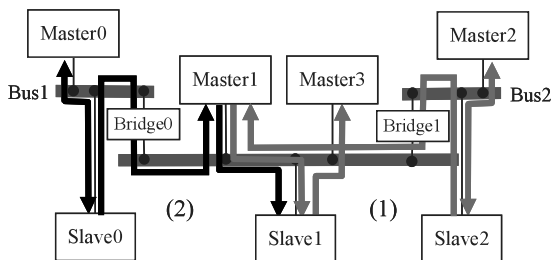


図6 対象システム構成 (B) とワークフロー
Fig. 6 Workflow on bus architecture (B).

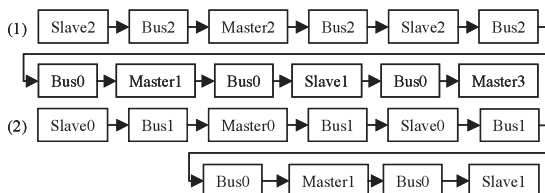


図7 従来のシミュレーションモデル (B)
Fig. 7 Conventional simulation model (B).

表4 性能評価結果 (B)
Table 4 Performance evaluation results.

		処理	待ち	合計
応答 時間	(1)画像再生	840	0	840
	(2)画像録画	960	0	960
占有率(%)	Bus0	25.0		
	Bus1	21.4		
	Bus2	21.4		

対象システム構成(B)は、システム構成(A)で Bus0 に集中していたトランザクションを、Bus1 と Bus2 に分散する。Bus2 は、Master2 と Slave2 を移動するトランザクション用であり、Bus1 は Master0 と Slave0 を移動するトランザクション用である。

表4に、シミュレーションモデル(B)での性能評価結果を示す。

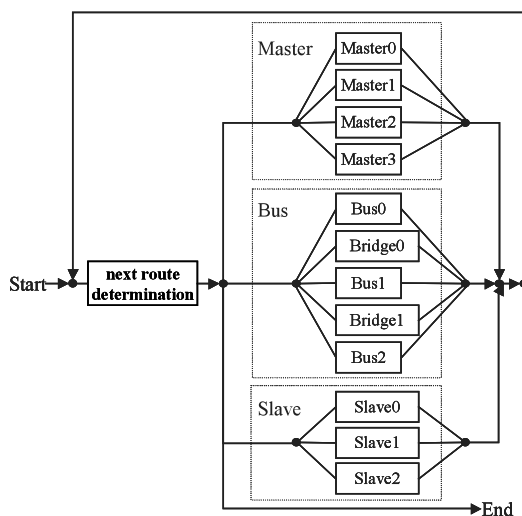
バス・アーキテクチャ(A)から(B)への変更によって、表3で示されていたシステム構成(A)として応答時間を劣化させている画像録画の応答時間が、表4が示すシステム構成(B)では改善されていることが分かる。それは、資源を利用するための待ち時間の削減による。また、各バスの占有率も均等であることから、現状の構造のさらなる改善は困難であることも分かる。

従来のシミュレーションモデルでは、図5と図7が示すように、処理順序は同じであっても、システム構成が(A)から(B)のように変わればワークフローが異なり、シミュレーションモデルを変更する必要がある。

表 5 非構造型シミュレーションモデルの特徴

Table 5 Characteristics of non-structured simulation model.

Variable
部品性能
ワークロード(トランザクションの発生時刻・量)
ワークフロー

図 8 非構造型シミュレーションモデル
Fig. 8 Non-structured simulation model.

あった．このシミュレーションモデルの開発に多くの工数を要していた．

4. 提案する性能評価モデル

従来の課題を解決するために，新たな性能評価手法を実現する性能評価モデルを提案する．

以降，本稿では，提案するシミュレーションモデルを非構造型シミュレーションモデルと定義する．非構造型シミュレーションモデルの特徴を表 5 にまとめる．

非構造型シミュレーションモデルの特徴は，構造型シミュレーションモデルでは固定であったワークフローを，シミュレーション時の変数として入力することである．これはすなわち，バス・アーキテクチャなど異なるシステム構成であっても複数のシミュレーションモデル開発が不要であることを示す．システム構成ごとにシステム構成に依存したシミュレーションモデルを開発する必要があった従来手法に比較すると，大幅な工数削減を図ることができる．

図 8 に，図 4，図 6 で説明したシステム構成 (A)，(B) を実現する非構造型シミュレーションモデルを示す．

このモデルは，システム構成に依存したワーク

- (1) Slave2, Bus0, Master2, Bus0, Slave2, Bus0, Master1, Bus0, Slave1, Bus0, Master3
- (2) Slave0, Bus0, Master0, Bus0, Slave0, Bus0, Master1, Bus0, Slave1

図 9 入力データ (A)

Fig. 9 Input data (A).

- (1) Slave2, Bus0, Master2, Bus0, Slave2, Bus0, Master1, Bus0, Slave1, Bus0, Master3
- (2) Slave0, Bus0, Master0, Bus0, Slave0, Bus0, Master1, Bus0, Slave1

図 10 入力データ (B)

Fig. 10 Input data (B).

ローは表現せずに，ワークフローを入力変数として扱う．ワークフローは発生するトランザクションに与えられ，そのワークフロー情報に従って，トランザクションが部品間を移動する．部品間の移動は next route determination 部品が制御する．従来の非構造型シミュレーションモデルでトランザクションが保持する情報は，データ量と発生から経過している時間であったが，非構造型シミュレーションモデルでは，これらに加えて，ワークフロー情報を保持する．このようにシミュレーションモデルの中を移動するトランザクションにワークフロー情報を与えることによって，システム構成に依存したシミュレーションモデルを開発する必要がなく，複数のシステム構成の性能評価を短時間で実現する．

図 9 に，対象システム構成 (A) の非構造型シミュレーションモデルに入力するワークフローデータを示す．これは，図 5 に示す従来の構造型シミュレーションモデルの接続情報と同等である．対象システム構成 (B) においても同様である．図 10 に非構造型シミュレーションモデルの入力を示す．これは，図 7 に示す従来の構造型シミュレーションモデルの接続情報と同等である．

また，非構造型シミュレーションモデルを構成するバスおよびスレーブ部品は，従来モデルでの部品同様，マスタから発生されるトランザクションに対して，待ち行列ルールを適用し使用権を与える．消費電力はトランザクションの処理を順次行うごとにマスタ，スレーブ，バスそれぞれの部品で消費される電力に加えて，使用権獲得のための待ち時間における電力も加算する．消費電力などの性能評価の情報は，バスやマスタやスレーブという機能ブロックのそれぞれに性能評価モデルの計算式として持たせ，それぞれの部品がトランザクションによって起動されたときに評価が行われる．

表 6 シミュレーション速度比較
Table 6 Comparison of simulation speeds.

	(A)	(B)
構造型 (秒)	41	44
非構造型 (秒)	88	95
Ratio	2.1	2.2

さらに、従来のシミュレーションモデルは、部品をワークフローに展開する形であるため、シミュレーションモデルのデータ量は、部品の総数よりかなり多くなる。データ量のオーダとしては、部品数を c 、ワークフロー数を w とした場合に、 $o(cw)$ となる。それに比べ、本提案では、部品数のみに依存し、 $o(c)$ である。対象システムが大規模になるほどこの差は顕著になり、シミュレーションを実行するハードウェアのメモリ空間に影響を及ぼすことになる。

5. 実験結果

非構造型シミュレーションモデルの効果を証明するために、従来の構造型シミュレーションモデルとの比較実験を行った。比較項目は、性能評価結果である応答時間と、シミュレーション速度、シミュレーションモデル開発工数である。

5.1 応答時間

非構造型シミュレーションモデルでの応答時間に関しては、構造型シミュレーションモデルでの結果である表 3、表 4 と同等の結果を得られた。

5.2 シミュレーション速度

表 6 に、システム構成 (A) と (B) に対して、構造型シミュレーションモデルで性能評価のためにシミュレーションに要した時間と、非構造型シミュレーションモデルとのシミュレーションに要した時間をまとめる。

構造型シミュレーションモデルでは、システム構成 (A) の性能を評価するためのシミュレーション時間は、41 秒であり、非構造型シミュレーションモデルでは、88 秒であった。また、システム構成 (B) の性能を評価するためのシミュレーション時間は、44 秒と 95 秒であった。非構造型シミュレーションモデルでのシミュレーション速度は、構造型シミュレーションモデルに比較すると、約 $1/2$ 倍である。

非構造型シミュレーションモデルが、構造型シミュレーションモデルに比較して、シミュレーション時間がかかる理由は、ワークフローの制御のために、next route determination 部品など、システム構成に存在しない部品を、トランザクションが経由するからである。

表 7 開発工数比較
Table 7 Comparison of development terms.

	BS1		BS2	
	(A)	(B)	Type-1	Type-2
構造型 (時間)	6	4	32	16
非構造型 (時間)	8	-	40	-
Ratio	0.5(N+0.5)		0.4(N+1)	

N: 評価対象構造数

5.3 開発工数

表 7 に、非構造型シミュレーションモデルと構造型シミュレーションモデルについて、BS1 と BS2 の 2 セットの開発工数をまとめる。

BS1 は図 5、図 7 で示すシステム構成 (A) と (B) のシミュレーションモデル開発工数である。BS2 は、マスタ部品が約 10、スレーブ部品が約 15 の大規模構成のシステムにおける Type-1 構造と Type-2 構造のシミュレーションモデル開発工数である。BS1、BS2 とも構造型シミュレーションモデルの場合は、システム構成ごとにシミュレーションモデルを開発する必要があるため、システム構成ごとに開発工数が必要になる。

BS1 の場合、システム構成 (A) の開発に約 6 時間、システム構成 (B) の開発に約 4 時間かかった。システム構成 (A) の開発工数内訳は、仕様開発に約 2 時間、コーディングに約 2 時間、テストに約 2 時間の合計約 6 時間である。システム構成 (B) の場合は、すでに (A) で構成部品の開発は行っているため、仕様開発、コーディング工数が短縮化され、約 4 時間であった。非構造型の場合のシミュレーションモデル開発工数は、評価対象となる複数のシステム構成で共通のモデルを 1 つ作る工数であり、共通部品の検討などの仕様設計に約 3 時間、コーディングに約 2 時間、テストに約 3 時間の合計約 8 時間かかった。BS2 においても同様である。

Ratio は、評価対象構造数 N を変数として、非構造型の開発工数と、構造型モデル開発工数比率を示す。開発工数の増加を BS1 の (A) と (B) から値 (4) を求め、最初のモデル (A) の開発工数 (6) を加え、非構造型モデル開発工数 (8) で比率を求めている。BS2 も同様である。この Ratio から、3 タイプのシステム構成にかかるシミュレーションモデル開発工数は、BS1 のような小規模モデルでは 1.75 倍、BS2 の大規模モデルでは 1.6 倍の効率化の効率化が可能である。

6. ま と め

本手法の特徴は、従来のように構造に依存したワー

クフローをモデル化するのではなく、構造とワークフローを分離して扱うことである。SoC 設計に重要なパス・アーキテクチャを検討するための拡大する探索空間への対応、ならびに、定量的な性能評価の実施の短期間化が可能になる。

本実験では、非構造型シミュレーションモデルと構造型シミュレーションモデルの開発工数の比較により、非構造型シミュレーションモデルが、短期間で開発できることを確認した。そして、非構造型シミュレーションモデルでも、従来の構造型シミュレーションモデルとまったく同等に、応答時間、パスの使用率（競合）結果を得ることができた。また、シミュレーション速度の点からは、非構造型シミュレーションモデルを用いた場合に、従来型に比較して 2 倍の時間がかかる。しかしながら、モデル開発工数が日・週の単位を要することに比べれば、シミュレーション時間は秒・分の単位であるので影響は微小である。

シミュレーションの評価値は、トランザクションによって異なるため、本シミュレータの使用者は、対象システムに想定されるトランザクションを与え、複数のパス・アーキテクチャに対して同トランザクション条件のもとでの、パスの競合、応答時間、消費電力などの評価結果を比較検討することによって、最適なパス・アーキテクチャを短期間で選択することができる。

参 考 文 献

- 1) Analyzing Packaging Trade-Offs During System Design, *IEEE Design & Test of Computers*, July-September (1998).
- 2) Solden, S.: Architecture Services Modeling for Performance in HW-SW Co-design, *Proc. SASIMI2001*, pp.72-77 (2001).
- 3) August, D.I., Keutzer, K., Malik, S. and Newton, A.R.: A Disciplined Approach to the Development of Platform Architectures, *Proc. SASIMI2001*, pp.67-71 (2001).
- 4) SES/Workbench User's Guide, HyPerformix, Inc., Austin, Texas (April 1989).
- 5) NitroVP, <http://www.cardtools.com>
- 6) 秋丸春夫, 川島幸之助: 情報通信トラヒック, オーム社 (1997).

(平成 14 年 10 月 24 日受付)

(平成 15 年 3 月 4 日採録)



高橋美和夏

昭和 61 年京都工芸繊維大学卒業。同年、松下電器産業（株）入社。論理合成用仮想配線モデル自動生成、配線パラツキ予測、RTL 消費電力推定、パス・アーキテクチャ性能解析システム・手法の研究開発に従事。



宮嶋 浩志

1994 年九州大学工学部情報工学科卒業。1996 年同大学大学院総合理工学研究科情報システム学専攻修士課程修了。1996 年～1997 年日本學術振興会特別研究員。1998 年アプリオリ・マイクロシステムズ社。1999 年九州大学大学院システム情報科学研究科博士課程単位取得退学。在学中、計算機アーキテクチャ、コンパイラ、並列処理システムの研究に従事。2000 年松下電器産業（株）入社。現在、同社半導体社開発本部プロセッサ開発センター勤務。組み込み向けシステム LSI 開発に従事。IEEE、電子情報通信学会各会員。



福井 正博（正会員）

昭和 58 年大阪大学大学院修士課程修了。同年、松下電器産業（株）入社。平成元年～平成 3 年カリフォルニア大学バークレー校にて客員研究員。平成 15 年立命館大学理工学部教授。自動配置配線、モジュール合成、セル合成等半導体 CAD およびシステム LSI 設計手法の研究開発に従事。工学博士。電子情報通信学会、IEEE 各会員。