Vol.2012-ARC-201 No.27
2012/8/2

ヘテロジニアスマルチプロセッサ環境を対象とした AMBA バスフレームワークの設計と評価

瀬戸 勇介[†1]　佐々木 敬泰[†1]　大野 和彦[†1]　近藤 利夫[†1]

概要：近年，多様なプログラムをより効率的に実行するための手段として，構成の異なるスーパースカラコアを用いたヘテロジニアスマルチコアプロセッサが注目されている．しかし，各コアの構成及びキャッシュ仕様等の組み合わせの膨大さや設計・検証にかかる時間が実用の面で大きな障害となっている．この問題を解決するために，任意の構成のスーパースカラコアを自動生成するツールとして FabScalar が提案されているが，現状ではマルチコア構成に対応していない．そこで我々は，この FabScalar のマルチコア構成への適用を行うことで，多様なヘテロジニアスマルチコア環境を自動生成するツールセットの実現を目指す．本稿では，その上で必要となる柔軟なバス設計の実現のため，AMBA バスの自動生成機構の提案と実装を行った．評価結果により，現実的な回路面積でヘテロジニアス構成を対象としたバスフレームワークが実現されていることが確認出来た．

キーワード：ヘテロジニアスマルチコア，AMBA，スヌープバス，FutureBus

Improvement of AMBA Bus Frame-work for Heterogeneos Multi-processor

Seto Yusuke[†1]　Takahiro Sasaki[†1]　Kazuhiko Ohno[†1]　Toshio Kondo[†1]

Abstract: The demand of heterogeneous multi-core processors is increasing in an area of the computer architecture. FabScalar is automatic generation tool for superscalar cores. This tool-set contributes to the construction and verification of heterogeneous systems. But, FabScalar does not have mechanism to generate cache systems and bus systems, which are necessary for multi-core architecture. In this paper, we propose and design a tool to automatically generate bus frame-work for heterogeneous environment. Our verification shows that the bus system generated by the proposed frame-work works correctly.

Keywords: Heterogeneous Multi-core，AMBA，SnoopBus，FutureBus

1. はじめに

現在，コンピュータアーキテクチャの幅広い分野において低消費電力且つ高性能なプロセッサが求められており，この要求を実現する手段として，スーパースカラコアを用いたヘテロジニアスマルチコアプロセッサの需要が高まっている．ヘテロジニアス環境は異種類の機能のコアを組み合わせることで構成され，各プログラムの特徴に合わせて複数種類のコアを使い分けることによってプログラムの実行効率を高めることが出来る．しかし，設計の特性上，フェッチ幅，パイプライン段数等の各スーパースカラコアの仕様やコア及びキャッシュの構成の組み合わせが膨大となり，それに伴う設計・検証にかかる時間の増加が大きな問題となっている．

この問題を解決するために，N.K.Choudhary らによって任意の構成のスーパースカラコアを自動生成するツールセットとして FabScalar[1] が提案されているが，現状ではマルチコア構成に対応していない．そこで我々は，この FabScalar のマルチコア構成への適用を図り，ヘテロジニアス環境を対象としたマルチプロセッサシステムの自動生

[†1] 現在，三重大学大学院工学研究科情報工学専攻
Presently with Graduate School of Information Engineering, Mie University

成機構の開発を行う．

FabScalar は，フェッチ幅，パイプライン段数，各ユニットの構成等の様々なパラメータを与えることで，仕様の異なるスーパースカラプロセッサを VerilogHDL コードで自動生成するツールセットである．しかし現状では，採用している命令セットにおいてマルチスレッドのアプリケーションへの対応が行われていないことに加え，キャッシュシステム及び共有バス，キャッシュコヒーレンシ制御といったマルチコア環境において必要となる機能の実装が行われていない．先に述べたヘテロジニアス環境では複数のコアを同時並行で動作させるため，FabScalar のマルチコア対応は必須であると考えられる．本稿では，このマルチコア対応の内，チップ内の各ユニット間の接続に用いるシステムバス部分について扱う．

システムバスを設計する上で，プロセッサやキャッシュの構成等が可変となるシステムにおいては各モジュール間のバス接続に関しても同様に可変となるため，より柔軟なバス構成が必要となる．そこで我々は，その解決方法として，可変構成のマルチコア及びキャッシュシステムに対応したシステムバスフレームワークの自動生成機構を提案する．

現在，チップ内バスの仕様は CoreConnect[2] や Wishbone[3]，AMBA[4] と様々なものが提案されているが，本稿では，その内で仕様が公開されており且つ組み込み分野でも広く用いられている AMBA を対象として実装を行った．

AMBA は ARM 社により提案されたマルチプロセッサ向けシステムバスの規格であり，いくつかのバージョンが存在するが，ハードウェア規模が小さく且つ実装が容易であるという点から，柔軟なシステムの開発を目的とする本稿では AMBA2.0 を採用した．AMBA2.0 は主に，キャッシュ等の内部メモリとプロセッサユニットとの間を接続する高性能な AHB バスと低周波数帯域で動作し周辺装置間の接続を行う APB バスとに分けられるが，今回はその内の AHB バス部分についてのみ扱う．

ここで，AMBA2.0 に関する注意点として，キャッシュコヒーレンシ制御が考慮されていないことが挙げられる．マルチコア対応においてキャッシュコヒーレンシ制御は必須のものであるのに加えて，ヘテロジニアス環境においてはキャッシュシステムと同様にコヒーレンシ制御についても柔軟な対応が必要となるため，本稿では基本的なシステムバスフレームワークに加えてキャッシュコヒーレンシ制御に関する実装も行った．また，後継のバージョンである AMBA4.0[5] ではすでにコヒーレンシ制御への対応が行われているが，コヒーレンシ処理部分を改良した場合に 4.0 の仕様との互換性が保てなくなることが考えられるため，本稿では 4.0 のサブセットではなく 2.0 のスーパーセットとしてバスの自動生成ツールを実現している．

検証の結果，一般的なバス通信処理を想定した環境において正常に動作し，目的とするバスフレームワークが実現されていることが確認出来た．

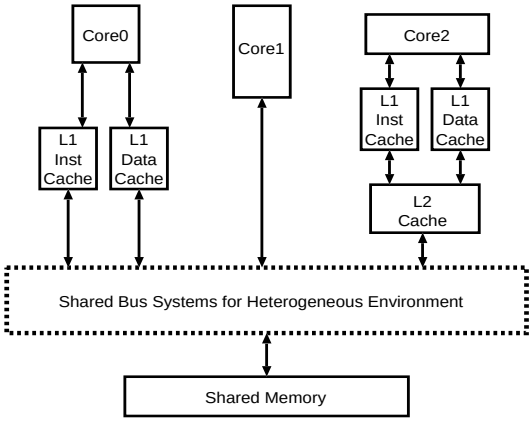

図 1　ヘテロジニアスマルチコア構成

2. ヘテロジニアス環境を対象としたバスフレームワーク

ヘテロジニアス環境は，一般的に用いられているホモジニアスなマルチコア構成と違い，各コアの内部仕様がそれぞれ異なった特性を持ったものとなっている．このコアアーキテクチャの違いから，マルチコア構成において必須となるキャッシュシステムや共有バスについても最適な構成を一意に決定することが難しくなると考えられる．

ここで，ヘテロジニアスマルチコアの構成例を図 1 に示す．図においても，各コアに付随するキャッシュ構成が L1，L2 の有無等の点で異なっており，また，それらの違いによって各コア側からの共有メモリへのバス接続も複雑なものとなっていることが分かる．この内，コア部分の設計の柔軟化については，前述の FabScalar の研究において多様なコアモジュールを自動生成するツールを実装することで解決されている．そこで我々は，マルチコア構成で必要となる他のシステムについても生成の自動化を行い，任意のヘテロジニアスマルチコア環境の容易な設計を可能とする総合的なツールセットの開発を目指す [6]．本稿では，図において点線で示されている共有バス部分を対象とするバスフレームワークの自動生成ツールの提案を行う．

3. 関連研究

AHB system generator[7] は，VHDL コードで記述された AMBA2.0 AHB の標準的なバスシステムを任意のユニット数で自動生成するツールセットである．ユニット数以外にバス認可アルゴリズム等の一部の機能も選択可能であり，また，バスフレームワークの生成に特化した本稿のツールとは違い，テスト用ユニットや信号生成機構，高速 (AHB)/低速 (APB) バス間のブリッジユニットも組み込まれている．しかし，AHB system generator は生成したバスモジュールに関してハードウェア規模の小面積化を目的

としておらず，各ユニット数についてはハードウェアレベルで可変となるが，その他の可変機能 (優先度アルゴリズム等) の中には if 文等の条件文で記述されているものがある．これは論理合成後にも回路構成として残るため，生成したバスモジュールに本来利用しない機能が含まれる可能性がある．HDL コードレベルで動作の評価を行う場合はこれで十分であるが，実際にハードウェア化することを考えた場合にこの冗長部分がネックとなる危険性がある．

これに対して，本稿で実装したツールでは，指定された機能以外の余分なコード生成は極力抑えられており，一部冗長となる部分についてもハードウェア化する段階で考慮が不要となるように設計されている．これにより，ハードウェア化を前提としたシステムに搭載する際にも即時の論理合成が可能となっている．

加えて，AHB system generator は AMBA2.0 のバス仕様に準拠することを保証するものであるのでこの仕様において未対応の機能については考慮されていない．これは当然のことであり，単純に AMBA2.0 準拠のバスのみが必要な場合には問題とならない．しかし，一般的なマルチプロセッサ構成に適応するバスの要求を考えた場合に，この未対応の機能の内，キャッシュコヒーレンシ処理が必須のものとなってくる．

本ツールではこのキャッシュコヒーレンシ制御に関するユニットの付加及びその生成の自動化も行っており，より総合的なマルチプロセッサ構成用バスの需要に応えた自動生成機構を実現している．

4. FabScalar

FabScalar は，現在ハイエンドなコンピュータシステムにおいて広く用いられているスーパースカラ構成のプロセッサを対象とした自動生成ツールである．スーパースカラプロセッサの複雑な構成や各ユニットの多様な仕様に対応していることに加えて，いくつかの動作シミュレータも備えており，ヘテロジニアス環境でのプロセッサの設計・検証における時間的なネックの解消を可能とする大規模なツールセットとなっている．

自動生成ツールの実体としては，様々なプロセッサ構成に対応した SystemVerilog(HDL) コードのスーパーセットとなっており，切り分けられた各機能コードをマクロ変数定義によって有効/無効化することで可変処理を実現している．これにより，ユーザはマクロ宣言ファイルの内容を変更するだけで容易に任意のプロセッサコードを生成することが可能となっている．

しかし，ヘテロジニアス環境の設計を考えた場合，現状の FabScalar には以下の 2 つの問題がある．

- 命令セットがマルチスレッドアプリケーション等に未対応
- マルチプロセッサ構成を可能とするための各システムが実装されていない

この内，本稿では，後者に関連した FabScalar のマルチプロセッサ構成への対応について扱っていく．

FabScalar を用いてマルチコア構成を実現することを考えた場合に，現状の FabScalar に加えてキャッシュシステムの追加が必要となる．さらにヘテロジニアスな環境ではキャッシュにおいてもプロセッサと同様に L2，L3 キャッシュの有無や共有/分散等の様々なキャッシュの構成パターンが存在するため，それらへの柔軟な対応も必要となる．また当然ながら，キャッシュシステムが可変となった場合には，それに伴うキャッシュコヒーレンシ制御の複雑化についても考慮しなければならない．

以上の改良点に関しても現状の我々の課題となってはいるが，本稿では，これらに加えて必要となる共有バスを対象とした改良について扱う．今後，コアやキャッシュの自動生成化が実現出来た場合に，それぞれのユニット毎に様々な仕様を取るため各モジュール間のバス接続は極めて複雑且つ多岐に渡るものとなる．そのため，このバス構成についても柔軟性を持ったシステムが必要であることが考えられる．本稿ではこの柔軟な共有バスという要求を満たすためのバス生成ツールを提案する．以降は，その詳細と実装方法について記述する．

5. AMBA バス

AMBA バスは ARM 社によって開発されたマルチプロセッサ用システムバスの規格である．ARM プロセッサに限らず MIPS コア等の様々なシステムに搭載されており，実際のハードウェアシステムにおいて現在極めて広く採用されているバス仕様である．本稿においても，生成したバスの汎用性の観点からこの AMBA 仕様に則ったバス設計を行った．

5.1 選定理由

AMBA にはいくつかのバージョンが存在し，その中でも Rev2.0 と 4.0 が一般的なシステムで用いられている．後継の 4.0 の方が 2.0 に比べて多くのバス通信機能に対応しているが，以下の利点から今回は Rev 2.0 を採用した．

- ハードウェア規模が小さい
- 仕様がシンプルであり，改良が容易
- キャッシュコヒーレンシ制御に未対応

上の 2 項目については，生成したバスの小規模化や柔軟なバス構成の実現という本稿の実装要求と合致することが分かる．また，コヒーレンシ制御についても柔軟な処理ユニットの実現が必要であることから，キャッシュコヒーレンシを考慮していないことも決定の一因となっている．コヒーレンシ処理部分の改良を行うことを前提とした場合に，すでにコヒーレンシ制御がバス仕様の中に組み込まれている 4.0 をベースにすると追加の処理ユニットをサブセット

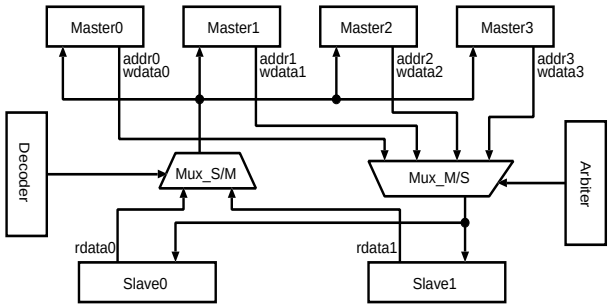

図 2　AMBA バス

として設計することになるため，4.0 仕様との互換性を維持することが困難になる．このことから，本稿では，バス通信の仕様と別にコヒーレンシ制御ユニットを付加することが可能な 2.0 をベースに実装を行っている．

5.2　AMBA バスの基本構成

AMBA バスは基本的に図 2 に示すようなマスタとスレーブユニットをマルチプレクサで相互接続する形態をとる．アドレス送信と書き込みデータはマスタユニットから，読み出しデータはスレーブユニットから送信されることが仕様で決定されているため，図 1 で示したようなマルチコア環境の共有バスに適用する場合には，コアモジュール側に位置するユニット (キャッシュ等) がマスタとなり，共有メモリ等のより下位のストレージユニットがスレーブとなることが規定されている．各マスタからの通信許可のリクエストをアービタで優先度付けし，その情報を用いてマルチプレクサでバスの選定を行うことで基本的なマルチコア構成でのデータ通信を可能としている．読み出しデータの送信を行うスレーブの選択に関しては，通信マスタから送られるアドレス値を用いてデコーダユニット内で簡単な組み合わせデコーディングを行うことで実現している．バスマスタ認可のための優先度アルゴリズムは AMBA の仕様では規定されていないため，本稿では実装の容易さからラウンドロビンアルゴリズムを用いた．

また，AMBA2.0 の仕様より，本ツールのバスフレームワークでは基本的な通信方法として，ロック通信，バースト転送，スプリット転送に対応している．

6.　キャッシュコヒーレンシ制御

前章で述べた AMBA バス仕様への準拠によって，多対多のマスタ/スレーブ通信は動作上実現出来たが，一般的なシステムにおけるバス通信では，図 1 の例からも分かるようにマスタユニット側にキャッシュシステムを含んでいることが多い．このことから，マルチコア環境に適用可能なバスを考えた場合に，各キャッシュ内のデータの一貫性を保証するキャッシュコヒーレンシ制御への対応が必須のものであると考えられる．以上より，マルチコア構成を対象とした柔軟なバスフレームワークを実現するためにコヒーレンシ処理も考慮に入れたバスの実装を行った．前述の通り，AMBA2.0 においてはキャッシュコヒーレンシ制御への対応が行われていないため，AMBA バスとは別に専用のユニットを付加することで実現した．この付加ユニットの詳細な実装内容について以下で説明する．

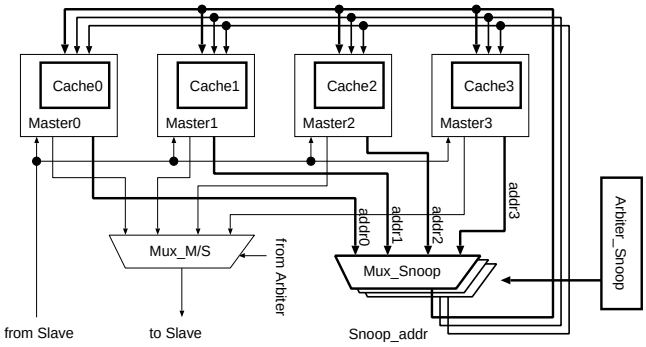

図 3　コヒーレンシ制御部分の構造

6.1　スヌープバス

キャッシュコヒーレンシ制御を実現する一般的な手段の 1 つとしてスヌープバスの導入がある．これは他のキャッシュのデータ書き込み情報を監視するためのバスであり，あるマスタでキャッシュへの書き込みが行われたときにこのバスを通して書き込み情報の受け渡しを行うことによってキャッシュの一貫性が保持される仕組みとなっている．

具体的には，AMBA バスのフレームワークに対して図 3 に示すようなユニットの追加を行った．マスタからのスヌープ処理のリクエストを元に専用のアービタで優先度決定を行い，通信を許可されたアドレスデータをマルチプレクサで選択後，各マスタにブロードキャストすることでスヌープ処理が実現される．スヌープ用アービタの基本的な処理内容は各マスタのリクエストに対するアクセス認可のみで，優先度アルゴリズムについては AMBA バスと同じものを使用している．AMBA バス用のアービタとの間でバスマスタの情報を共有することでより正確なアービトレーション処理を行っている．また，スレーブへのアクセスの有無を判断するための信号線も新たに付加しており，より効率的なバス認可アルゴリズムによる優先度処理を実現した．

6.2　FutureBus

ヘテロジニアス環境を対象としたバスを考えた場合に，前節で述べた単純なアドレススヌーピングへの対応だけでは不十分であると考えられる．一般的なプロセッサ環境ではキャッシュコヒーレンシ処理をより効率化するためにコヒーレンシ制御のためのプロトコルが規定されており，現在 MOESI や MESI 等の様々な手法が提案されている．このことから，キャッシュシステムによってその仕様が変化するため，本ツールではスヌープバスの実装に加えてこの

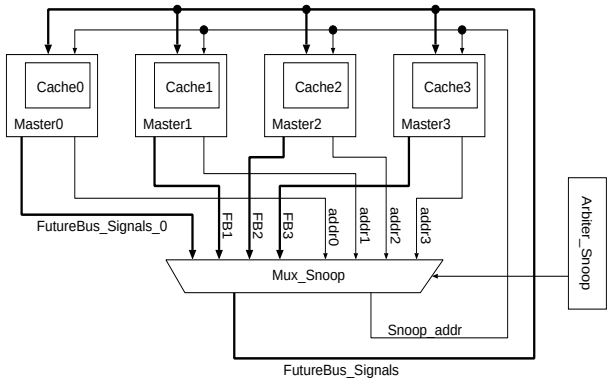

図 4 本稿の実装範囲

コヒーレンシプロトコルに関する改良も行った．

スヌープバスをベースとしたさらなる改良として，FutureBus[8] に準拠したコヒーレンシ制御信号を導入した．FutureBus は IEEE に承認されたシステムバス規格の 1 つであり，多様なコヒーレンシプロトコルへの対応を可能としている．実装方法としては，図 4 に示すように，スヌープバスによる各マスタへの書き込みアドレスの通知と同時に，キャッシュ制御ユニットで必要となる詳細な通信情報 (FutureBus 信号線) も合わせてブロードキャストする仕組みとなっている．これにより，一般的に広く用いられているコヒーレンシプロトコルである MOESI に準拠した制御処理が AMBA バス上で可能となり，MESI や Berkeley プロトコルを始めとする，信号線仕様において MOESI に準ずる様々なコヒーレンシプロトコルについても同バスフレームワーク上で対応されている．

7. バス生成の自動化

本稿では，これまでに AMBA バスとコヒーレンシ制御機構を組み合わせた一般的なマルチコア環境のためのバスフレームワークを示した．ここで，さらにヘテロジニアスな環境を対象としたバス構成を可能にするため，上記のバスフレームワークを任意の構成で生成出来るツールの実装を行った．実装した自動生成ツールは SystemVerilog(HDL) で記述されたバス構成のスーパーセットとなっている．具体的には図 5 のように，各パラメータの定義が列挙されたマクロ宣言ファイル内の値をユーザが変更することで容易に多様なバス構成を生成出来るようになっている．図の例では，マスタ台数/スレーブ台数/スヌープバス本数の各パラメータがそれぞれ 4/2/3 と定義されているため，図 3 のバス構成が生成出来る．

現在の実装時点で可変対応されているパラメータを表 1 に示す．ユニット数の変更が可能となっている部分はマスタ/スレーブユニット台数とスヌープバス本数であり，それぞれ上限は 16 である (スヌープバスはマスタ/スレーブが 1/1 のとき，0 本になる)．FutureBus 信号線の本数はスヌープバス本数と同数である．コヒーレンシプロトコルに

Parameter.vh

```
\*** 可変パラメータの宣言 ***\
\* マスタユニット台数の指定 *\
'define MASTER_NUM  4  \\ ← [ここの値を変更]

\* スレーブユニット台数の指定 *\
'define SLAVE_NUM  2

\* スヌープバス本数の指定 *\
'define SNOOP_NUM  3
        :
        :
```

図 5 パラメータの指定

表 1 実装ツールの可変仕様

ユニット数 [最大]	
マスタユニット	16
スレーブユニット	16
スヌープバス	16
コヒーレンシプロトコル	
MOESI	動作検証済
MESI	動作検証済
Berkeley	動作未検証
Dragon	動作未検証
Firefly	動作未検証
...etc	

関しては前述の通り MOESI とそれに準じた信号線を扱うプロトコルについて対応している．また，表に示した部分以外にもバス認可の優先度アルゴリズム (固定/ラウンドロビン) やバス幅等のいくつかの機能が変更可能となっているが，現状，機能としては不十分なものであるので詳しい説明は省く．

8. 評価

本稿で実装したツールにより自動生成したバスフレームワークに関して基本的な通信が達成出来ていることを確認するため，テスト用に実装した環境を用いて評価を行った．一般的なバス通信を模した信号を用いた通信動作の確認と，2 章で述べた既存ツールである AHB system generator との性能比較を行うことで本ツールの有用性を検証した．

8.1 評価環境

評価環境として標準的なマスタ/スレーブユニットを実装した．各ユニットの仕様を以下に示す．

マスタユニットは，L2 キャッシュを想定したモジュールとなっており，キャッシュメモリ以外に簡単な通信制御ユニットとキャッシュ制御ユニットを備えている．通信制御ユニットは信号のジェネレータと通信処理用のステートマシンにより構成され，一般的なタスク通信の信号例に加えて，ロック通信やバースト転送用のテスト信号が生成可能となっている．キャッシュ制御ユニットはコヒーレンシ

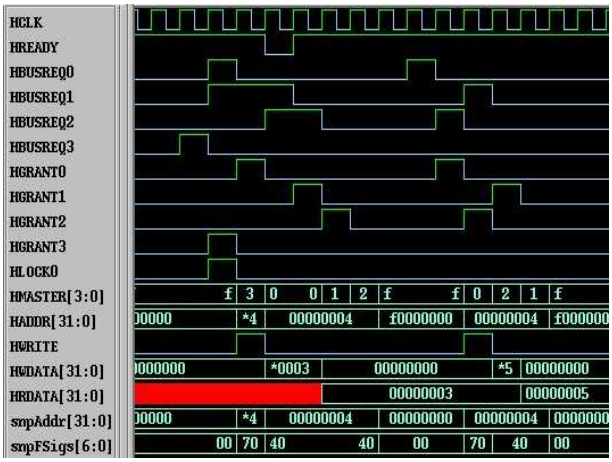

図 6 波形図の一例

処理の正当性を検証するためのテスト環境となっており，MOESI 及び MESI プロトコル通信を模したテスト用のコヒーレンシ制御信号の生成が可能となっている．

スレーブユニットは共有メモリを想定したモジュールであり，AMBA における基本的なエラー処理，ウェイト処理のテストが可能となっている．また，スプリット転送も可能ではあるが，スプリット転送を実行中に他のスプリット転送を重ねて開始することは許していない (AMBA の仕様により)．

8.2 評価結果

図 6 にバス通信テストにより得られた波形図の一例を示す．この検証により，本ツールで生成されたバスが正常に動作していることが確認出来た．

次に，生成したバスモジュールを論理合成し，得られた総ゲート数と動作周波数について AHB system generator との機能比較を行った．合成にはシノプシス社の Design compiler(ver. 2010.03-SP5) を使用し，テクノロジにはローム社の ROHM 0.18μm CMOS プロセスを用いた．評価時のマスタ/スレーブ台数設定はそれぞれ 6/1 台で，本ツールで生成したモジュールにはスヌープバスを 1 本付加している．

＜論理ゲート数＞

実装したツール：	1,189 [ゲート]
比較対象：	1,130 [ゲート]

＜動作周波数＞

実装したツール：	454.5 [MHz]
比較対象：	409.8 [MHz]

以上の結果より，本稿で実装したツールで生成したバスフレームワークについて，キャッシュコヒーレンシ処理や生成の自動化等の機能付加によるハードウェア規模の増加や動作速度の低下は十分に抑えられていると判断出来る．

9. おわりに

本稿では，ヘテロジニアスマルチプロセッサ環境を対象とした柔軟なシステムバスフレームワークの提案とその設計を行った．前章の評価より，AMBA バス仕様に準拠したバスフレームワークが正常に動作としていることに加えて，一般的なコヒーレンシプロトコルにおいて正しいキャッシュコヒーレンシ制御が行われていることが確認出来た．また，上記の機能追加による面積や機能の低下も抑えられていることから，柔軟且つ小面積という要求も満たしており，目的としていたバスの自動生成機構が実現出来たと考えられる．

今後の課題としては以下のことが挙げられる．

- プロセッサモジュールを用いたバス通信テスト
  今回の評価結果は評価用に実装したマスタ/スレーブユニットによる動作結果であるので，より信頼性の示しやすい一般的なプロセッサモジュールを備えたシステム上での動作検証を行う必要がある．
- 大規模なマルチコア構成への対応
  処理の高速化の観点から，マルチコア化の技術は今後さらに進歩することが予測される．現時点での各ユニット台数の可変の上限は 16 台でありこの要求を満たすには不十分であるため，今後は，より大規模なマルチコア構成を視野に入れた改良が課題となる．
- 切り替え可能な機能の追加
  ユニット数の可変上限の拡張に加えて，より多くのプロセッサ仕様への対応を可能とするためには，アドレス幅等の各バス幅やライトスルー，ライトバック等のキャッシュ仕様を始めとするバスの構成を決定する他の要素についても可変対応が必要となる．
- ハイエンドコンピュータへの対応
  4 章でも述べたように，本稿ではよりハイエンドなコンピュータシステムを対象としたバスフレームワークの実現を考えている．しかし，現時点のバス仕様ではルックアップフリーキャッシュ等の一部の高速化手法に対応出来ていないため，機能の拡張可能性を考慮した仕様の見直しが必要であると考えられる．
- 通信許可アルゴリズムの改良
  現状では，マスタへの通信許可を単純な優先度処理アルゴリズムにより行っているが，さらに多様なコア仕様を持つヘテロジニアス環境を実現する場合には，この優先度決定の効率化も課題となる．そのため，アービタユニット内の動作アルゴリズムについても考慮する必要がある．

謝辞

本研究の LSI 設計は東京大学大規模集積システム設計教育研究センターを通し，ローム株式会社，シノプシス株式会社の協力で行われたものである．

参考文献

[1] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi, E. Rotenberg. FabScalar: Composing Synthesizable RTL Designs of Arbitrary Cores within a Canonical Superscalar Template. Proceedings of the 38th IEEE/ACM International Symposium on Computer Architecture (ISCA-38), pp. 11-22, June 2011.

[2] IBM: CoreConnect Bus Archirecture, http://ibm.com/chips/products/coreconnect/

[3] OpenCores: WISHBONE System-on-chip (SoC) Interconnection Architecture for Portable IP Cores, http://www.opencores.org/

[4] ARM: AMBA 仕様 (Rev2.0) http://www.arm.com/

[5] ARM: AMBA AXI and ACE Protocol Specification, http://www.arm.com/

[6] 中林智之，佐々木敬泰，Eric Rotenberg，大野和彦，近藤利夫：FabScalar の Alpha 21264 命令セット対応とマルチプロセッサ環境フレームワークの構築，SACSIS 2012，2012 年 5 月

[7] OpenCores: AHB system generator http://opencores.org/

[8] P. Sweazey, A. J. Smith. A class of compatible cache consistency protocols and their support by the IEEE Futurebus, Proceedings of the 13th annual international symposium on Computer architecture Vol. 14 No. 2 pp. 414-423, 1986.