2003－ARC－153 (1)
2003／5／8

オンチップマルチプロセッサ向け内部接続網の検討

薬袋俊也† 緑川 隆† 田辺靖貴†
茂野真義† 天野英晴†

大きな配線遅延を引き起こす近年の半導体プロセス技術を持つオンチップマルチプロセッサでは、接続網がシステムのボトルネックとなる。そこで、バスとクロスバを組み合わせた接続網 CWB(Crossbar With Bus) を提案し、命令レベルシミュレータを用いて評価を行った。CWB では、スヌープ機能を持つバスを用いてアドレスの転送を行い、クロスバを用いてバーストデータの転送を行う。シミュレーションの結果、CWB がさまざまな条件下で、データ転送にバスを用いた場合よりも優れた性能を示し、実行時間で最大約 30%の性能向上を示すことを確認した。その一方で、L1 キャッシュで用いたライトスルー方式が CWB の効果に悪影響を与えていることが明らかになった。

Examination of the inside connection network for on-chip multiprocessors

Toshiya Minai,† Takashi Midorikawa,† Yasuki Tanabe,† Masayoshi Shigeno† and Hideharu Amano†

In on-chip multiprocessors with recent semiconductor process technology which causes a large wiring delay, interconnection networks will be a bottleneck of systems. Here, an interconnection method called CWB(Crossbar With Bus) which combines a bus and crossbar is proposed, and evaluated with an instruction level simulator. In CWB, a bus with snoop mechanism is used for address transactions, while the burst data is transferred through the crossbar. Simulation results shows that CWB improves the execution time of a system with a simple bus 30% at maximum. On the other hand, it appears that the write through policy used in L1 cache degrades the effect of CWB.

1. はじめに

命令レベルの並列性を利用した高速化は、引き出すことのできる並列性の限界や、並列性を引き上げるために生じたハードウェアの複雑化によって、これまでのような継続的な性能向上を続けることが困難になってきている。

そこで命令レベルよりも粗いレベルの並列性を利用するオンチップマルチプロセッサが注目されている。現在のオンチップマルチプロセッサは、強力なプロセッサを少数接続するもの[1]、比較的シンプルなプロセッサコアを 4 から数十個搭載し、粗流度の並列性を活かすことによって性能を向上させるもの[2][3] など様々な構成が試されている。しかし、全体としてオンチップマルチプロセッサ上に実装されるプロセッサコアに高性能・高周波数を実現したものが要求されるようになりつつあり、また、集積度の向上を活かし、現在よりも数多くのコアがチップ上で実装されるようになることが予想される。よって、将来、チップ上のプロセッサコア間の通信やプロセッサコアとオンチップキャッシュ間の通信への要求が増加してくると考えられる。

このため、最も単純な方法であるバスを用いたモジュール間の接続では、いかに高バンド幅なチップ内バスを用いても増加した通信量を円滑に扱うことは難しい。一方で、クロスバのようなスイッチによる接続が使われることも増えているが、他のプロセッサコアからのメモリアクセスをスヌープすることができないため、各プロセッサコアに付随し低レイテンシでアクセスされるキャッシュを用いて、プロセッサコア間で積極的にデータを共有することが困難になる。

そこで、本稿では、オンチップマルチプロセッサ用接続方式の一つとして、バスとクロスバの両者の接続方式の利点を活かした CWB(Crossbar With Bus) を提案し、評価する。まず、2 章で CWB についての説明と、CWB の予備評価を行うために設計したオンチップマルチプロセッサシステムについての説明を行い、3 章でこのシステムをシミュレーションによって検証するために開発した命令レベルシミュレータの設計と実装について取り上げ、4 章で評価、考察を行う。最後に 5 章で結論および今後の課題を述べる。

2. オンチップマルチプロセッサ向け接続網

2.1 CWB の提案

モジュール間の接続で最も一般的なバスとクロスバにはそれぞれ次のような特徴がある。

- バス
  - スヌープキャッシュの制御が容易
  - 一度に一つのモジュールのみアクセス可能
- クロスバ

† 慶應義塾大学 理工学研究科
Department of Computer Science, Graduate School of Keio University

– 同一のタイミングで複数の通信が混在可能
– スヌープ機能を持たない

そこで、バスとクロスバを組み合わせることで、双方の長所を活かす接続網 CWB(Crossbar With Bus) を提案する。CWB では、スヌープキャッシュの制御が容易であるという長所を持つバスをアドレス転送用ネットワークとして用い、同一のタイミングで複数の通信が混在可能であるという長所を持つクロスバをデータ転送用ネットワークとして用いる。

2.2 CWB 検討のためのシステム

CWB を検討するために、図 1 に示した構成を取るシステムを設計した。このシステムを以後 CWB システムと呼ぶ。

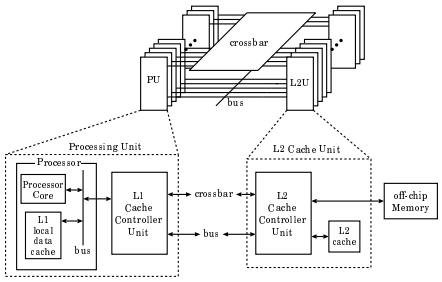


図 1 CWB 検討のためのシステムの構成

CWB システムは、PU (Processing Unit) と L2U (L2 Cache Unit) が、アドレス転送用ネットワークであるバスと、データ転送用ネットワークであるクロスバを介して接続されている。

バスおよびクロスバは最も基本的な接続法であるため、これらの組み合わせは珍しくはない。Hydra[2] では 64bit の Write through bus と 256bit の Read/replae bus を分離することで同様の効果を達成している。Piranha[3] では ICS(Intra Chip Switch) とキャッシュ制御用ハードウェアの組み合わせでキャッシュの一致と高い転送性能を実現する。また、我々が提案した SNAIL-2[4] でもキャッシュ制御用に特化したネットワークである MINC とデータ転送用の PBSF が分離されている。

しかし、バスとクロスバの直接的な組み合わせによる評価については今まで行われておらず、ここでは将来、より現実的なスイッチ構成の第一歩としての評価を行うことにする。

2.3 各ユニットの構成

2.2 節で示した、CWB システムの構成要素それぞれについて、その機能ブロックごとにその構成を示す。

2.3.1 PU(Processing Unit)

PU は、主に以下のブロックに分類される。

- プロセッサ … コア部分とライトバック方式でアクセスされる L1 ローカルデータキャッシュによって構成され、バスを介して L1 Cache Controller Unit と接続されている。
- L1 Cache Controller Unit … PU と内部バスを介して接続されており、他の PU、L2U ともバスとクロスバを介して接続されている。ライトスルー方式でアクセスされる L1 共有データキャッシュを持ち、共有データへのアクセス制御を行う。

2.3.2 L2U(L2 Cache Unit)

L2U は、L2 Cache Controller Unit と、L2 キャッシュから構成される。L2 Cache Controller Unit は、他の PU、L2U とバスとクロスバを介して接続されている。また、ライトバック方式でアクセスされる L2 キャッシュ、RDRAM のオフチップメモリにも接続されており、PU 側からの要求に応じて共有データへのアクセス制御を行う。

2.3.3 アドレス転送用ネットワーク (バス)

共有データへのアクセスする際のアドレス転送はすべてバスを介して行われるため、バスに接続された各 PU は他の PU のアクセスをスヌープすることが可能である。

2.3.4 データ転送用ネットワーク (クロスバ)

共有データへのアクセスする際のデータ転送はすべてクロスバを介して行われる。スイッチの切り替えによって複数の通信路が設定できるため、同一のタイミングで複数のデータ転送が混在可能である。

2.4 メモリアクセス要求に対する流れ

プロセッサから共有データへのメモリアクセス要求が発行されたときの流れを示す。

- Read アクセス
  1. プロセッサから発行される共有データへの Read 要求は、L1 Cache Controller Unit が受け取り、L1 共有データキャッシュを参照し、
     * 有効なラインが格納されている場合には、そのラインを読み出し、プロセッサに送信する。
     * 有効なラインが格納されていない場合には、バスを通して要求を L2U に転送する。
  2. アクセス対象のラインを保持する L2 Cache Controller Unit は、その要求を受け取り FIFO に追加する。バスに発行される要求を確実に受信するため FIFO が満杯のときには、バスをロックし新たな要求が発行されないようようにする。
  3. L2 Cache Controller Unit は、FIFO 内にある要求を順に処理する。L2 キャッシュを参照し、
     * 有効なラインが格納されている場合には、そのラインを読み出す。
     * 有効なラインが格納されていない場合には、オフチップメモリからラインを読み出し、そのラインを L2 キャッシュに格納する。
  4. L2 Cache Controller Unit は、読み出したラインをクロスバを通して PU に送信する。
  5. ラインを受け取った L1 Cache Controller Unit は、そのラインを L1 キャッシュに格納し、ラインをプロセッサに転送する。L2U からのラインの受信待ちの間に、他の PU が要求先のラインに Write 要

求を発行した場合には、受け取ったラインを L1 キャッシュに格納せずにプロセッサに転送する。

- Write アクセス
  1. プロセッサから発行される共有データへの Write 要求は、L1 Cache Controller Unit が受け取り、L1 共有データキャッシュを参照し、
     * 有効なラインが格納されている場合には、そのラインを無効化してから、Write 要求をバスを通して、Write データをクロスバを通して L2U に転送する。
     * 有効なラインが格納されていない場合には、Write 要求をバスを通して、Write データをクロスバを通して L2U に転送する。
  2. 書き込み対象のラインを保持する L2 Cache Controller Unit は、バスからアドレスを、クロスバよりデータを受け取り、FIFO と DATA BUFFER にそれぞれを追加する。FIFO が満杯のときには、バスをロックし新たな要求が発行されないようようにする。

     また各 PU の L1 Cache Controller Unit はバスをスヌープしており、自キャッシュ内に保持するラインへの書き込みが行われた場合にはそれを無効化する。
  3. L2 Cache Controller Unit は、FIFO 内にある要求を順に処理する。L2 キャッシュを参照し、
     * 有効なラインが格納されている場合には、FIFO 内のアドレスとセットにして格納されている DATA BUFFER 内のデータで、L2 上のエントリを更新する。
     * 有効なラインが格納されていない場合には、オフチップメモリに対して書き込みを行う。

3. CWB 命令レベルシミュレータの設計と実装

本研究では、CWB の詳細で且つさまざまな条件で評価が行える評価環境の構築を目的とする。さまざまな評価手法のうち、シミュレーションによる方法は、構成変更に柔軟に対応し、詳細な評価を行う環境の構築が実現可能である。特に命令レベルシミュレーションによる評価手法は、実アプリケーションが実行でき、正確な評価を行うことができる。また、本研究室で開発された汎用並列計算機シミュレータライブラリ ISIS を利用できることから、本研究では CWB システムの命令レベルシミュレータ (CWB 命令レベルシミュレータ) の設計および実装を行った。

3.1 CWB 命令レベルシミュレータ

CWB 命令レベルシミュレータは、図 1 で示した CWB システムとほぼ同じ構成であり、processing unit クラス、L2 cache unit クラス、address bus クラス、data transfer クラスから構成される。data transfer クラスでは、クロスバでの通信とバスでの通信の切り替えを可能にし、データ転送用ネットワークの違いによる評価を行えるようにした。

3.2 各クラスの構成とその機能

この節では、CWB 命令レベルシミュレータを構成する各クラスの構成とその機能を示す。

3.2.1 processing unit クラス

processing unit クラスは、以下のクラスを内包する。

- r3081 processing element クラス … R3000 ベースのプロセッサとライトバック方式でアクセスされる L1 local data cache クラスを内包する。ISIS のライブラリで提供されているものを用いた。
- L1 cache controller クラス … CWB システムの L1 Cache Controller Unit の機能を実現した。L1 cache controller クラスの構成を図 2 に示す。L1 cache controller クラスに内包された各クラスは、次の機能を持つ。
  - access controller クラス … プロセッサからの要求に応え、L1 shared data cache クラスへのアクセス、address bus クラス、data transfer クラスを通しての L2 cache unit クラスとの要求・データのやり取りを行う。
  - adrbus snoop クラス … address bus クラスに発行される要求を監視し、Write 要求が発行されたときには L1 shared data cache クラスを参照し、対応するラインを無効化する。
  - L1 shared data cache クラス … 32KByte の 4way セットアソシアティブキャッシュで、ライトスルー方式でアクセスされる。アクセスレイテンシは 1 system clock。

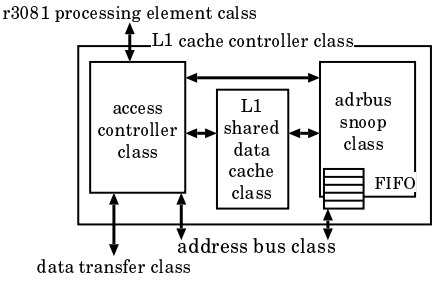


図 2 L1 cache controller クラスの構成

3.2.2 L2 cache unit クラス

L2 cache unit クラスは、以下のクラスを内包する。

- L2 cache controller クラス …L2 Cache Controller Unit の機能を実現した。
- L2 cache クラス …システム全体で 8MByte の 4way セットアソシアティブキャッシュで、ライトバック方式でアクセスされる。アクセスレイテンシは 2 system clock。
- mapped memory クラス … オフチップメモリを実現したクラスで、アクセスレイテンシは、1 回のアクセスで最初のデータのみ 48nsec で、それ以降はデータ

1 つにつき 8nsec。

3.2.3 address bus クラス

2.3.3 項で示したアドレス転送用ネットワークの機能を実現した。age technique 方式でアービトレーションを行う。

3.2.4 data transfer クラス

データ転送用ネットワークを構成するクラスで、複数の通信が混在可能なクロスバでの通信と混在不可能なバスでの通信の切り替えを可能にした。age technique 方式でアービトレーションを行う。

3.3 シミュレーションオプション

さまざまなパターンでの評価を簡易に実行できるように、シミュレータ起動時にコマンドラインオプションを指定することにより、以下の機能を制御できるようにした。

`-p[Num]` …プロセッサ数の指定
`-D[Num]` …データ転送ネットワークのバンド幅の指定
`-f[Num]` …FIFO サイズの指定
`-F[Num]` …プロセッサ周波数の指定
`--L1` …L1 shared data cache クラスを有効に
`--L2` …L2 cache クラスを有効に
`--xbar` …データ転送方式をクロスバに設定
　未設定時はバスに設定
`--half` …ネットワーク周波数を $\frac{1}{2}\times$ プロセッサ周波数に設定　未設定時にはプロセッサと同一の周波数を用いる

4. 評　　価

本章では、作成したシミュレータを用い CWB の評価をいくつか行い、考察する。評価には、 SPLASH2 ベンチマーク集から以下のアプリケーションを使用した。

表 1　評価に使用したアプリケーション

FFT	高速フーリエ変換	2^{12}
LU	行列の LU 分解	64×64
RADIX	整数の基数ソート	65536 keys

4.1 評 価 環 境

評価環境は各評価で断りのない限り、表 2 の通りである。

4.2 標準の評価環境における実行結果

表 2 に示した評価環境において、PU 数を 1、2、4、8、16 と変化させ、データ転送用ネットワークにバスを用いたときと、クロスバを用いたときの両方で、ネットワーク利用率と実行時間を測定した。

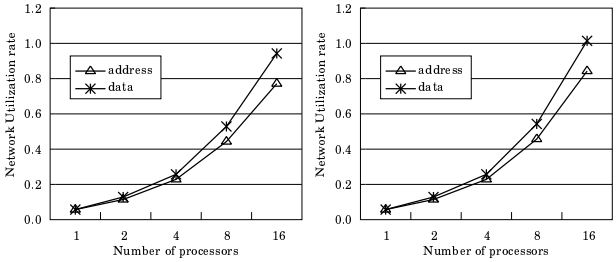
データ転送にバスを用いた場合と、クロスバを用いた場合のそれぞれで FFT を実行した際のネットワーク利用率を図 3 の (1)、(2) に示した。図 3 の (1) と (2) を比較すると、データ転送にバスを用いた場合とクロスバを用いた場合でネットワーク利用率の差はあまり見られないことが分かる。図 4 に、各 PU 数においてデータ転送にバスを用いたときに実行した際の実行時間を 1 として、データ転送にクロスバを用いたときの実行時間での性能向上

表 2　標準の評価環境

processing unit	
clock rate	200 MHz
cache line size	32 Byte
L1 cache size(local data)	16 KByte(per PU)
L1 cache size(instruction)	4 KByte(per PU)
L1 cache size(shared data)	32 KByte(per PU)
L1 cache set associativity	4 way
L1 cache access	1 system clock
band width(to processor)	32 bit
L2 cache unit	
fifo length	8
L2 cache size	8MByte(whole system)
L2 cache set associativity	4 way
L2 cache access	2 system clock
memory access	48 + 8×(n-1) nsec
address bus network	
clock rate	200 MHz
band width	32 bit
arbitration	1 network clock
data transfer network	
clock rate	200 MHz
band width	128 bit
arbitration	1 network clock

n: cache line size/data size

の度合を示したが、ネットワークの利用率にあまり差が出ていないため、実行時間にも大きな差は出ていない。



(1) バスを用いた場合　　(2) クロスバを用いた場合

図 3　標準の評価環境において FFT を実行した際のネットワーク利用率

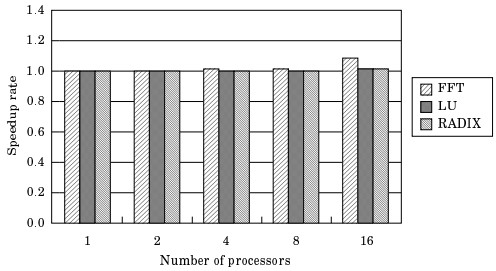
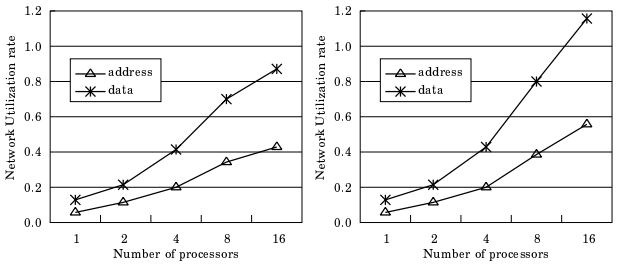


図 4　標準の評価環境におけるデータ転送用ネットワークの違いによる実行時間の変化

4.3 キャッシュラインサイズを大きくした際の実行結果

キャッシュラインのサイズを 32Byte から 128Byte に増やし、評価を行った。評価結果は、図 5 が 4.2 節の図 3

に、図 6 が 4.2 節の図 4 に対応している。



(1) バスを用いた場合　　(2) クロスバを用いた場合

図 5　キャッシュラインサイズを大きくして FFT を実行した際のネットワーク利用率

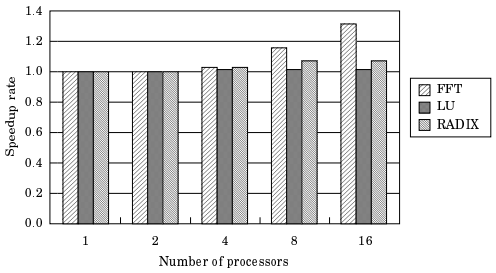


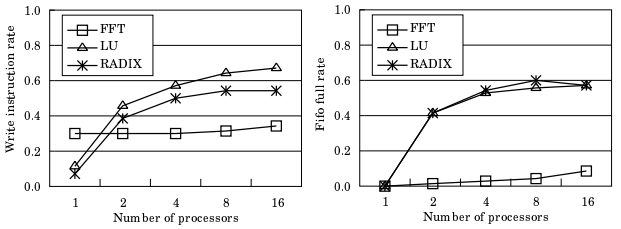
図 6　キャッシュラインサイズを大きくした際のデータ転送用ネットワークの違いによる実行時間の変化

まず、図 5 と図 3 を比較すると、図 5 の方が、データ転送用ネットワークの利用率が高いことが分かる。これは、キャッシュラインサイズが 32Byte のときよりもデータ転送用ネットワーク上でライン 1 つを転送するのに要する時間が増大したことにより、データ転送用ネットワークの混雑度が増したからであると考えられる。次に、図 5 の (1) と (2) を比較すると、PU 数が増えるにしたがい、データ転送にバスを用いた際のネットワークの利用率とデータ転送にクロスバを用いた際のネットワークの利用率の差が大きくなっていることが分かる。これは、キャッシュラインサイズを大きくしたことによって混雑したネットワークが、PU 数を増やすことによって更に混雑し、その結果クロスバの「同一のタイミングで複数の通信が混在できる」という利点が大きく活かされているためであると考えられる。特に 16PU で実行した場合に、クロスバ通信でのデータ転送用ネットワークの利用率が、バス通信での利用率の最大値である 1 を越えており、クロスバの特徴が活かされていると言える。

ネットワークの利用率の差は、結果として実行時間にはっきりと出ている。図 6 と図 4 を比べると、図 6 の方がデータ通信にバスを用いた場合とクロスバを用いた場合の実行時間での性能差が大きくなっていることが分かる。特に、PU 数が増えネットワークが混雑したときに実行時間においても差が大きくなっている。

しかし、LU を実行した場合には、データ転送にバスを用いた場合とクロスバを用いた場合で実行時間の差がほとんど出ておらず、RADIX を実行した場合にも、FFT を実行したときほどの差は出ていない。

図 7(1) に 3 つのアプリケーションそれぞれを実行したときにプロセッサから発行される全要求に占める Write 要求の割合を示し、図 7(2) に 3 つのアプリケーションを実行したときの実行時間に占める L2 Cache Unit の FIFO が満杯になっている時間の割合を示した。図 7(1) を見ると、RADIX や LU は FFT に比べ Write 要求の割合が大きいことが分かる。今回のシステムの場合、プロセッサは Read 要求を発行するとデータを受け取るまでストールするが、Write 要求を発行した場合にはメモリ上に実際に書き込みが反映されるのを待たずに次の要求を実行できる。よって、Write 要求の占める割合が大きい LU と RADIX では、FIFO に格納される Write 要求のキューが多く、図 7(2) のように FIFO が満杯になり、バスがロックされている時間の割合が多い。このためプロセッサがバスに要求を発行できず、ストールしている時間が増えたと考えられる。よって、単位時間にデータ転送用ネットワークに発行されるデータ数も減少し、データ転送にバスを用いた場合でも十分に処理できたため、データ転送用ネットワークの違いによる性能差が現れなかったと考えられる。



(1)Write 要求の占める割合　　(2)FIFO が満杯になっている時間の割合

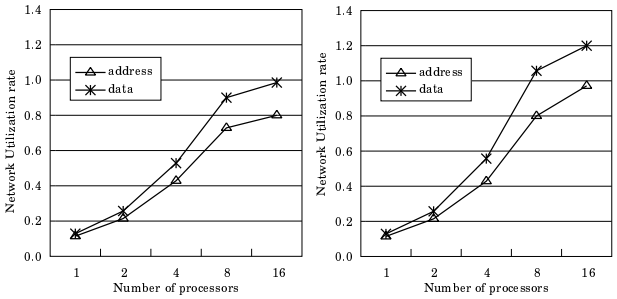
図 7　各アプリケーションを実行した際の Write 要求の占める割合と FIFO が満杯になっている時間の割合

しかし、FFT のように Read 要求の占める割合が大きいアプリケーションを実行した際には、クロスバの利点を大きく活かすことができ、データ転送にバスを用いた場合よりも実行時間を大きく短縮することができる。将来、集積度が向上し、現在より数多くの PU によってアプリケーションを実行する際には、データ転送用ネットワークにかかる負荷が現在よりも大きくなると考えられる。その場合、FFT のようなアプリケーションを実行する際には、CWB が有効な接続網であり、バスを用いた場合よりも高い性能をあげることができると言える。また、CWB とバスのみの接続網との性能差は PU 数が増えれば増えるほど、大きくなると考えられる。

4.4　プロセッサ周波数を上げた際の実行結果

ネットワークへの負荷が高い状態での評価を行うため、プロセッサ周波数を 200MHz から 400MHz に上げて、評価を行った。ネットワーク周波数は 200MHz のままであ

る。評価結果は、図 8 が 4.2 節の図 3 に、図 9 が 4.2 節の図 4 に対応している。



(1) バスを用いた場合　　(2) クロスバを用いた場合

図 8　プロセッサ周波数を上げて FFT を実行した際のネットワーク利用率

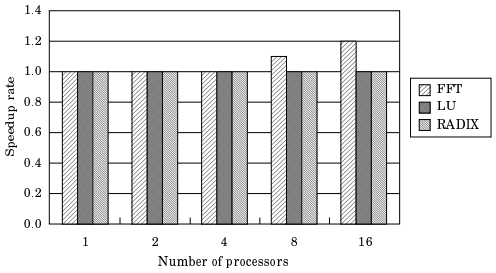


図 9　プロセッサ周波数を上げた際のデータ転送用ネットワークの違いによる実行時間の変化

図 8 と図 3 を比較すると、データ転送にバスを用いた場合もクロスバを用いた場合も、どの PU 数においてもネットワーク利用率がアドレス転送、データ転送ともに上昇している。これは、プロセッサ周波数が高くなることによってプロセッサの処理速度上がり、単位時間にネットワークに発行される要求数やデータ数が増えたからである。まず、図 8(1) に注目すると、16PU のときにデータ転送用ネットワークが飽和し、アドレス転送用ネットワークの利用率も頭打ちになっていることが分かる。図 8(2) を見ると、クロスバの「同一のタイミングで複数の通信が混在できる」という特徴が活かされ、データ転送用ネットワークの利用率が、バス通信の利用率の最大値である 1 を越え、上昇していることが分かる。また、そのことが、データ転送にバスを用いた場合に頭打ちになっていたアドレス転送用ネットワークの利用率を上昇させ、アドレス転送用ネットワークの有効利用を可能にさせる結果となった。しかし、16PU のときにはアドレス転送用ネットワークが飽和し、データ転送用ネットワークの利用率の上昇率が下がってしまっていることが分かる。

図 9 と図 4 を比較すると、FFT の場合のみデータ転送にバスを用いた場合に対する実行時間での性能向上が見られ、LU や RADIX では性能向上が見られないが、これは 4.3 で考察したように、LU や RADIX が FFT に比べ、プロセッサが発行する全要求に占める Write 要求の割合が大きいためである。

プロセッサ周波数を上げた際にも、FFT のように Read 要求の占める割合が大きいアプリケーションを実行した際には、クロスバの利点が大きく活かされ、データ転送にバスを用いた場合よりも実行時間を大きく短縮できる。将来、プロセッサが現在よりも高性能・高周波数化され、単位時間に発行される要求が増加した場合にも、FFT のようなアプリケーションを実行する際には、CWB が有効な接続網であり、バスのみの接続網を用いた場合よりも高い性能を示すことができると考えられる。

5. 結論および今後の課題

本稿では、オンチップマルチプロセッサ用接続方式の一つとして、バスとクロスバの両者の接続方式の利点を活かした CWB を提案し、CWB をさまざまな構成のもとで詳細に評価できる環境の構築を目的に、CWB 命令レベルシミュレータの設計と実装を行った。

実装したシミュレータを用いた評価の結果、CWB がさまざまな条件下で、データ転送用ネットワークにバスを用いた場合よりも優れた性能を示すことを確認した。その一方で、L1 共有データキャッシュのアクセスにライトスルー方式を用いているため、書き込みが連続して発生する場合には、L2U へのアクセスが集中し、性能低下を招いていることが明らかになった。今後の課題としては、以下の項目が挙げられる。

- Write 要求によって性能低下が起こることを避けるために、L1 共有データキャッシュへのアクセスをライトバック方式に変更し、キャッシュ間のデータ転送を可能にする
- 実行可能なアプリケーションを増やし、多様なアプリケーション実行時の性能評価を行っていく
- オンチップマルチプロセッサを用いたシステムに、より高い規模拡張性を持たせるため、チップ間の接続方式やチップ間でのデータ共有の方法について検討していく

参 考 文 献

1) Jim Kahle, “Power4 Focuses on Memory Bandwidth”, MICROPROCESSOR REPORT, Vol. 13, No. 13, Oct. 1999.
2) Kunle Olukotun, Basem A. Nayfeh, Lance Hammond, Kenneth G. Wilson, and KunyougChang, “The Case for a Single-Chip Multiprocessor”, Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp.2-11, Oct. 1996.
3) Luiz Andre Barroso, et al, “Piranha: A scalable architecture based on Single-Chip Multiprocessing”, International Symposium on Computer Architecture, June. 2000.
4) 茂野 真義, 緑川 隆, 白石 大介, 田辺 靖貴, 天野 英晴, “キャッシュ制御機構を持つスイッチ結合型並列計算機 SNAIL-2 の評価”, 情報処理学会研究報告 ARC03-152, pp.103-108, 2003.