Vol.2011-ARC-194 No.5
2011/3/11

CoreSymphonyの実現に向けたコアアーキテクチャの検討

坂口嘉一[†1]　松村貴之[†1]
永塚智之[†2]　吉瀬謙二[†1]

我々は複数のコアが協調して，1つの仮想コアとして機能可能な，CoreSymphonyアーキテクチャを提案している．これまでの研究では，協調動作のベースのコアとして，2命令発行程度のアウトオブオーダコアを想定してきた．本稿では，ベースとなるコアの構成について検討する．コアのバックエンドの発行幅を制限した構成について評価を行った．

Analysis of Core Configurations for CoreSymphony Architecture

Yoshito Sakaguchi,[†1] Takayuki Matsumura,[†1]
Tomoyuki Nagatsuka[†1] and Kenji Kise[†1]

We are developing CoreSymphony architecture which can compose a virtual core from some physical cores. The previous studies use 2-issue out-of-order core as physical core. In this paper, we examine configurations of physical core. We evaluate the core configuration of reduced issue width with SPEC benchmarks.

1. はじめに

CMP(Chip Multi-Processor) はプログラムに内在するスレッドレベル並列性を利用し，複数スレッドを複数コアで並列実行することで性能向上を得るアプローチである．しかし，Amdahl の法則[1] は，プログラム中に存在する並列不可能な処理（逐次処理）が CMP の性能を制限する可能性を示している．例えば，全処理の内の 90%が並列化可能なプログラムがあるとする．本法則によれば，このプログラムを 100 コアを使用して並列実行しても，1 コア時に対する性能向上は約 9.2 倍にとどまる．現状ではプログラム中の逐次部分を無くすことは難しく，CMP においても逐次処理の高速化は依然重要な課題であるといえる．

プロセッサの逐次実行能力を高める一般的な方法は発行幅を増加させることである．そこで我々は，複数個のプロセッサコアが協調することで，より発行幅の広い仮想コアを形成する CoreSymphony アーキテクチャ[2] を提案している．本アーキテクチャでは，2 命令発行の out-of-order コアが最大で 4 コア協調することにより，8 命令発行の仮想的なコアを構成できる．

CoreSymphony では 2 命令発行の out-of-order コアをベースとしている．これは，1 チップに集積可能なコア数を増やすために小規模なコアが望ましいこと，協調時は各コアが独立して命令を実行するため out-of-order 実行のための機構が必要であること，といった理由による．一方，RAW[3] では in-order 発行コアを集積している．また，階層的なクラスタ型実行ユニットをもつ WiDGET[4] では，個々の機能ユニットは in-order に命令を発行する．

CoreSymphony はフロントエンドでの通信を削減するため，自コアにステアリングされた命令のみを格納するローカル命令キャッシュを備えている．しかし，ローカル命令キャッシュから命令をフェッチする場合，ステアリング結果は毎回同じになる．そのため，コア間の負荷バランシングが出来ず，実行ユニットが有効に活用されていない可能性がある．

以上の点を踏まえ本稿では CoreSymphony の実行ユニットの構成について検討する．そのためにまず，実行ユニットにおける負荷の偏りを評価する．次に，CoreSymphony で使用されているステアリングアルゴリズムであるリーフノードステアリングの特性について解析する．その後，実行ユニットの構成について検討し，評価を行う．

2. CoreSymphony アーキテクチャ

以下では，複数のコアが協調して 1 つの強力なコアを構成できるアーキテクチャをコア融合アーキテクチャと呼ぶ．CoreSymphony もコア融合アーキテクチャの 1 つである．2 命令発行の out-of-order コアをベースとして，最大で 4 つのコアが協調して動作できる．以下の要因により，協調動作時の高い逐次性能が達成される．

発行幅の拡大　各コアの実行ユニットは，クラスタ型の実行ユニットにおける 1 つのクラスタとして機能する．そのため，発行幅が協調コア数に比例して増加する．

L1 データキャッシュのマルチバンク化　協調動作により，データキャッシュの容量が 1 コ

†1 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†2 東京工業大学 工学部情報工学科
Department of Computer Science, Tokyo Institute of Technology

ア分の (協調コア数) 倍に増加する．このとき，各コアのデータキャッシュは，アドレスによってインターリーブされたキャッシュのバンク 1 つとして機能する．

命令キャッシュの分散　CoreSymphony には従来型の命令キャッシュに加え，ローカル命令キャッシュを備えている．協調動作時，ローカル命令キャッシュに各コアの実行に必要な命令だけを格納することで，実質的な命令キャッシュ容量・帯域が増加する．

3. コアアーキテクチャの検討

本章では，CoreSymphony のベースとなるコアのアーキテクチャについて検討する．はじめに現在の CoreSymphony における各コアの実行ユニットの動作について調査する．

3.1 リーフノードステアリング

CoreSymphony では，(協調コア数)x4 命令長のフェッチブロック（FB）単位で命令をフェッチ・ステアリングする．リーフノードステアリングは FB に含まれる命令のデータフローグラフのリーフノード（子ノードが存在しないノード．FB 内にその命令に依存する命令が存在しない）に着目するステアリングアルゴリズムである．通信を抑制することで性能向上を図っている．

このアルゴリズムではまず，リーフノードをラウンドロビンでコアに割り振る．次に，FB のリーフノードが依存する命令を，そのリーフノードが割り当てられたコアにステアリングする．このとき必要であれば命令を複製する．結果として，同一の命令を複数のコアで実行することがあるが，1 つの FB 内に含まれる命令間で，実行結果を通信する必要がなくなる．

3.2 負荷の偏り

CoreSymphony は，ステアリング済みの命令をローカル命令キャッシュからフェッチすることで，複数コアの協調実行を実現している．ローカル命令キャッシュから命令をフェッチした場合，FB 中の命令は毎回同じコアにステアリングされる．そのため，コア間の負荷に偏りが生じることが考えられる．

そこで，各コアにステアリングされる命令数を計測し，コア間の負荷の偏りを調べる．CoreSymphony のサイクルレベルシミュレータを使用して，SPEC2006 の 5 つのベンチマークを実行する．1G 命令をスキップした後の 100M を対象とする．シミュレーションに使用したコアのパラメータは表 1 のとおりである．

結果を図 1 に示す．グラフの値は，協調しているコアの内，実行命令数が最も多いコアの値を最も実行命令数が少ないコアの値で割ったものである．この値が小さいほどコア間の負荷の偏りは少ないと考えられる．2 コアおよび 4 コア実行時について計測した．グラフから分かるように，2 コア協調時における負荷の偏りは少なく，最大でも 14%程度である．一方，4 コア協調時は最も偏りが少ない h264ref で約 19%，最大では，bzip2 のように 48%程度の偏りが存在している．

表 1　シミュレーションに使用したコアのパラメータ

fetch・decode	2 inst / cycle
conv. instruction cache	8kB, 2way
local instruction cache	8kB, 4way
data cache	16kB, 2way
L2 cache	2MB, 4way , 10 cycle latency
memory	100 cycle latency
execution unit	int:2, fp:2
physical register file	int:32, fp:32
instruction window	24 entry

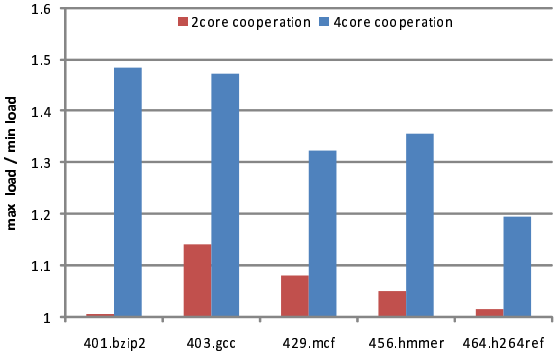

図 1　協調時における各コアの負荷の偏り

3.3 実行ユニットの使用率

図 2 に，4 コア協調時における各コアの実行ユニットの使用率を示す．縦軸は，各コアが 1 サイクルに実行する平均命令数である．リーフノードステアリングによって命令が複製されるため，グラフの値を合計しても実際の IPC にはならない．また，投機ミスによりコミットされない命令の実行も統計に含まれている．そのため，実際の使用率に比べ高い値が出ていると考えられる．

各コアは 2 命令発行の実行ユニットをもつので，縦軸の最大値は 2 である．しかし，実

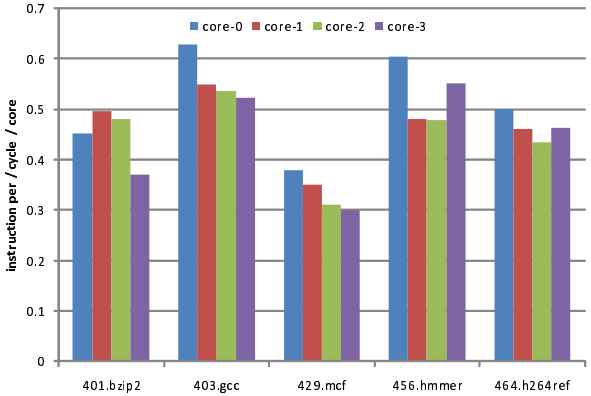

図 2 4 コア協調時における各コアの使用率

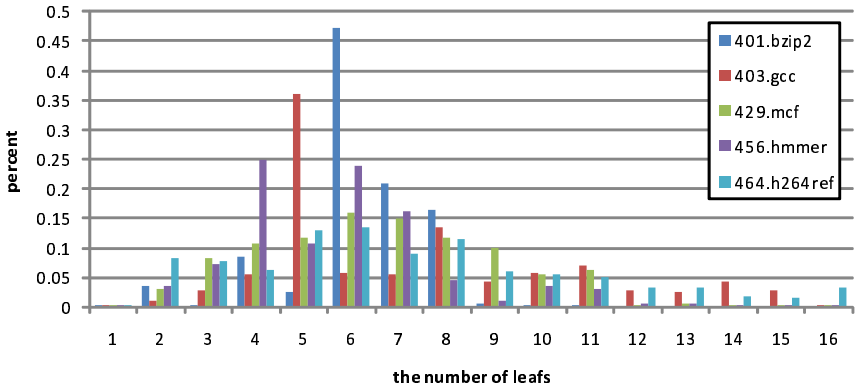

図 3 4 コア協調時における FB に含まれるリーフノードの分布

際の使用率は高々0.6 程度であり，平均した場合，実行ユニットの使用率は半分に届いていないことが分かる．

3.4 FB に含まれるリーフノードの数

リーフノードステアリングにおけるリーフノードの数を計測する．リーフノードの数が協調動作しているコア数（この評価では 4 コア）で割り切れない場合，負荷のばらつきは避けられない．特に，ある FB に含まれるリーフノードが 4 つ未満の場合，(i) 命令を割り当てられないコアが存在すること，(ii) リーフノード数が少ないため 1 つ当りの命令数が長い可能性があること，といった理由により負荷のバランスに大きな悪影響を与えると考えられる．一方，リーフノードの数が 5 以上であれば，最低 1 つのコアに複数のリーフノードが割り当てられる．命令が複製される場合を除けば，それぞれのリーフノード間に命令の依存関係は存在しない．そのため，1 つのコアに多くのリーフノードが割り当てられるほど，そのコアでより多くの命令レベル並列性が期待できる．

計測結果を図 3 に示す．横軸は，FB に含まれるリーフノードの数，縦軸はフェッチした全 FB に占める割合である．5 種のベンチマークにおいて，ピークは 4 つから 6 つ辺りにある．表 2 に，1 つから 3 つ，4 つから 7 つ，8 つから 11 つ，12 つ以上の 4 区間についてリーフノードの数の割合を示す．5 種のベンチマークで，リーフノードが 4 つから 7 つの間であるケースが最も多い．次に多いのは 8 つから 11 つの区間である．

表 2 リーフノードの割合

リーフノードの数	bzip2	gcc	mcf	hmmer	h264ref
1 - 3	3.8	4.0	11.6	10.0	16.0
4 - 7	79.2	52.8	53.7	75.8	42.0
8 - 11	16.9	30.6	33.5	12.1	28.2
12 -	0.0	12.6	1.2	1.2	13.5

3.5 コアアーキテクチャの検討

以上の結果をもとに，コアの構成について検討する．先に示したように，実行ユニットの使用率は多くとも 30%を超える程度である．そのため，実行ユニットを制限したものに置き換えても，性能低下が抑えられる可能性がある．そこで，(i) 命令の発行幅を狭くする，(ii) 命令の発行を in-order に制限する，の 2 通りについて性能を評価する．

ベースライン (base) 従来どおりの 2 命令発行の out-of-order コア．

1 命令アウトオブオーダ発行 (1issue) ベースラインにおいて，バックエンドの発行幅を 1 命令に制限したもの．フロントエンドは，ベースラインと同じく 1 サイクル当り最大 2 命令をフェッチ・デコードする．

2 命令インオーダ発行 (inorder) ベースラインにおいて，バックエンドでの命令発行を in-order に制限したもの．すなわち，命令ウィンドウ内で最も古い 2 命令の 1 つにならない限り，命令は発行されない．

Vol.2011-ARC-194 No.5
2011/3/11

4. 評　　価

前章の評価と同じシミュレータ，ベンチマークを使用して評価する．シミュレータのパラメータは base, 1issue, inorder の 3 つの構成全て同一で，表 1 のとおりである．ただし，コア数によらず L2 キャッシュの容量は，2MB とした．

4.1 評 価 結 果

評価結果を図 4 に示す．横軸はベンチマークおよび協調コア数を，縦軸は対応する IPC を表している．コア数が同じであれば，base がもっとも IPC が高いことがわかる．また，1 コアでは inorder の IPC が 1issue に比べて高い．しかし，協調コア数が増加したときの伸びは 1issue が大きく，2 コア，4 コアにおいて inorder に並ぶ，もしくは逆転している．

1issue と inorder　1 コアの 1 命令発行では，out-of-order 実行が可能であっても，その機会はキャッシュミス時などに限られ，少ないと予想される．そのため，1 コアにおいて inorder が 1issue に比べ高い IPC を示していると考えられる．

base と inorder　同じコア数同士で比較した場合，1 コア協調から 4 コア協調まで base の IPC が高い．しかし，性能比でみた場合，その差は協調するコア数が増えるごとに広がる．

base と 1issue　同じコア数同士で比較した場合，1 コア協調から 4 コア協調まで base の IPC が高い．しかし，base と 1issue の性能比でみた場合，コア数が多くなるにつれ，その差は小さくなる．1 コアの場合では，平均で 24%程度の性能低下があるが，4 コアの場合では，10%以内になる．

コア数ではなく実行ユニットの合計を基準に比較した場合平均で，base の 1 コアと 1issue の 2 コアでは 1issue が約 3%高く，base の 2 コアと 1issue の 4 コアでは 1issue が約 6%低い．発行幅の合計は同一であるのに，このような結果が出る理由としては，協調動作するコア数が多くなるほど実質的な命令キャッシュ，データキャッシュの容量が増加する CoreSymphony の特徴が考えられる．

発行幅の削減によるハードウェア量の縮小量によっては，1 命令発行に変更することで，コア面積当たりの性能が高くなることが期待できる．

5. ま と め

CoreSymphony のステアリングアルゴリズムと実行ユニットについて調査した．これらをもとに，実行ユニットの構成について検討し，評価した．その結果，コア 1 つ当りの発行幅を削減した場合，協調動作時の発行幅を同一にして比較すると，発行幅の合計が 2 の場合は性能向上，合計が 4 の場合は，約 6%の性能低下であった．

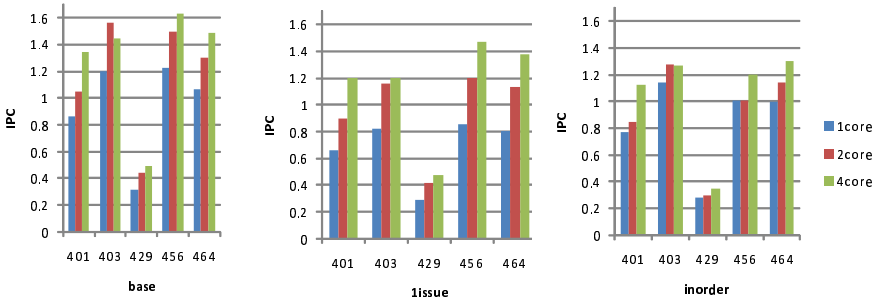

図 4　base, 1issue, inorder の各構成における，協調コア数別の IPC

今後課題としては，構成を変更した場合のハードウェア規模について見積もり，実行ユニットの制限の有用性を評価することがある．

謝辞　本研究の一部は科学研究費補助金（課題番号:22700046 若手研究 (B)）による．

参 考 文 献

1) M. D. Hill and M. R. Marty: Amdahl's law in the multicore era, *IEEE Computer*, Vol.41, No.7, pp.33–38 (2008).
2) 若杉祐太，坂口嘉一，吉瀬謙二：協調可能スーパースカラ CoreSymphony，情報処理学会論文誌コンピューティングシステム， Vol.3, No.3, pp.67–87 (2010).
3) Waingold, E., Taylor, M., Sarkar, V., Lee, V., Lee, W., Kim, J., Frank, M., Finch, P., Devabhaktumi, S., Barua, R., Babb, J., Amarsinghe, S. and Agarwal, A.: Baring it all to Software: The Raw Machine, Technical report, Cambridge, MA, USA (1997).
4) Watanabe, Y., Davis, J.D. and Wood, D.A.: WiDGET: Wisconsin decoupled grid execution tiles, *Proceedings of the 37th annual international symposium on Computer architecture*, ISCA '10, pp.2–13 (2010).