性能・消費電力・信頼性の間のトレードオフを考慮出来るマルチ・クラスタ型コア・プロセッサ

佐藤, 寿倫
九州大学システムLSI研究センター

舟木, 敏正
九州工業大学大学院情報工学研究科

出版情報：電子情報通信学会技術研究報告, CPSY2007-31. 107 (276), pp.39-44, 2007-10-26. 電子情報通信学会CPSY研究会
バージョン：
権利関係：

性能・消費電力・信頼性の間のトレードオフを考慮出来るマルチ・クラスタ型コア・プロセッサ

佐藤 寿倫† 舟木 敏正‡

† 九州大学 システム LSI 研究センター
‡ 九州工業大学大学院 情報工学研究科
E-mail: † toshinori.sato@computer.org, ‡ t-funaki@klab.ai.kyutech.ac.jp

あらまし 半導体製造技術における微細化の進展により，ソフトエラーの増加が問題となっている．マルチコアプロセッサを利用しスレッドを冗長実行する方式では，消費電力が大幅に増大してしまう．一方，シングルプロセッサ内で命令を冗長実行する方式では，性能低下が深刻である．本稿では，マルチ・クラスタ型コア・プロセッサ（MCCP: Multiple Clustered Core Processor）と呼んでいるマルチコアプロセッサ上での，性能・消費電力・信頼性の間のトレードオフを考察する．高い電力効率と高性能とを両立するために，スレッド冗長実行と命令冗長実行の組み合わせを提案する．シミュレーションにより，MCCP 上で提案方式を利用すると，スレッド冗長実行方式と比較して，エネルギー遅延積を 13%改善出来ることを確認している．

キーワード 消費電力，ディペンダビリティ，マルチコアプロセッサ，トレードオフ，ソフトエラー

Performance, Power, and Dependability Trade-off on Multiple Clusterd Core Processors

Toshinori SATO† Toshimasa FUNAKI‡

† System LSI Research Center, Kyushu University
‡ Graduate School of Computer Science and System Engineering, Kyushu Institute of Technology
E-mail: † toshinori.sato@computer.org, ‡ t-funaki@klab.ai.kyutech.ac.jp

Abstract As deep submicron technologies are advanced, we face new challenges, such as power consumption and soft errors. A naïve technique, which utilizes emerging multicore processors and relies upon thread-level redundancy to detect soft errors, is power hungry. Another technique, which relies upon instruction-level redundancy, diminishes computing performance seriously. This paper investigates the trade-off between performance, power, and dependability on a multicore processor, which is named multiple clustered core processor (MCCP). It is proposed to hybrid thread- and instruction-level redundancy in order to achieve both high power efficiency and small performance loss. Detailed simulations show that the MCCP exploiting the hybrid technique improves power efficiency in energy-delay product by 13% when it compares with the one exploiting the naïve thread-level technique.

Keyword Power consumption, dependability, multicore processors, trade-off design, soft errors

1. はじめに

マイクロプロセッサにおける近年の消費電力増大の問題を解決するために，マルチコアプロセッサが登場している．プロセッサの性能はその面積の平方根に比例し消費電力は面積に比例するという経験則を考慮すると，マルチコアプロセッサは電力利用効率の面で非常に優れた選択肢であると言える．

一方で，半導体製造技術における微細化の進展により，ソフトエラー率（SER: soft error rate）が増大しつつある[1, 2]．トランジスタサイズの縮小に伴い，ビットあたりの面積が小さくなっている．また高電界による破壊を防ぐために，電源電圧も縮小している．これらの結果ノードの電荷が小さくなり，宇宙線やアルファ粒子などにより容易にビットセルが反転する．一方でビットセルの面積が小さくなれば，中性子などの粒子がそこに衝突する確率も小さくなる．したがって両者の影響を考慮すると，ビットあたりの SER は世代間でほぼ一定であると言われている．しかしチップあたりのトランジスタ数は指数的に増大しているのであるから，チップあたりの SER も爆発的に増大することがわかる．

シングルイベントアップセット（SEU: single event

upset）による故障を検出するために，プログラムを冗長に実行することが提案されている[3, 4, 5, 6]. 近年登場しているマルチコアプロセッサは，この冗長実行に向いている．プログラムを複製し，同一チップ上の異なるコアで冗長に実行する．一つのプログラムから得られる二つの冗長な結果が一致しなければ，SEU が検出されたことになる[3,4]．本稿では，この方式をスレッド冗長方式（DTR: dependability with thread-level redundancy）と呼ぶ．残念なことに，DTR は大きな電力を消費する．二つのコアが動作すること，またそれらの結果を比較するための電力も考慮すると，少なく見積もっても一つのコアの消費する電力の二倍が必要である．

単一プロセッサ内での冗長実行を利用して SEU を検出することも可能である．プログラム自体ではなくプログラム中の命令を複製し，同一のプロセッサコア上で冗長に実行する．一つの命令から得られる二つの冗長な結果を比較し，もし一致しなければ SEU が検出されたことになる[5,6]．本稿では，この方式を命令冗長方式（DIR: dependability with instruction-level redundancy）と呼ぶ．DIR は DTR に比べると低電力である利点を持つが，深刻な性能低下を引き起こす[5,6].

DTR における消費電力増大の問題と DIR における性能低下の問題を解決するために，これらの方式を組み合わせることを提案する．現在，プログラムに適応可能なマルチコアプロセッサを検討中である[7]. マルチ・クラスタ型コア・プロセッサ（MCCP）と呼んでいるそれは，クラスタ型マイクロアーキテクチャ[8]を採用している．MCCP 上で DTR と DIR を組み合わせることで，電力効率と性能の改善を目指す．

2. マルチ・クラスタ型コア・プロセッサ

MCCP[7]の例を図 1 に示す．均質型マルチコアプロセッサである．従来の均質型マルチコアプロセッサとの違いは，各コアが単一的なコアではなくクラスタ型マイクロアーキテクチャ[8]を採用している点である．図 1 の例では，二つのクラスタから構成されるコアを二つ持っており，各コアには命令キュー（IQ），レジスタファイル（RF），そして機能ユニット（FU）が備わっている．命令キャッシュ（I$），データキャッシュ（D$），分岐予測器（BrPred），そしてデコーダ（Decode）は，同一コア内の全てのクラスタで共有されている．クラスタ型マイクロアーキテクチャの特徴を利用し，これまでに電力と性能との間のトレードオフに関して検討してきた[7].

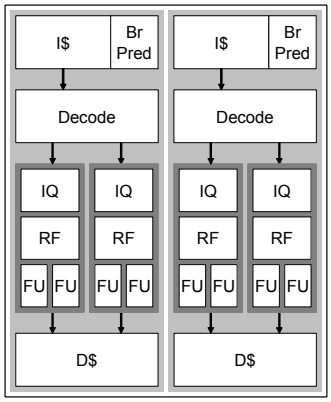

図 1. マルチ・クラスタ型コア・プロセッサ

2.1. 電力と性能のトレードオフ

マルチコアプロセッサは小電力で高性能を達成出来る魅力的な選択肢であると言われている．図 2 にマルチコアプロセッサの様々な構成を示す. 図 2a は単一プロセッサである．図 2b と図 2d は均質型マルチコアプロセッサであり，図 2c は不均質型マルチコアプロセッサである．容易にわかるように，不均質型マルチコアプロセッサには，大きさと性能の異なる様々なコアが備わっている．高性能が要求されるが並列性の低いスレッドを実行する時には，大きなコアを利用する．同様に高性能が要求されるが並列性の高いスレッドを実行する時には，小さなコアを複数利用する方が電力効率の点で好ましい．一方，高性能が要求されない時には，小さなコアを選択する．不均質なコアから適切なものを選択することで，必要最小限の電力で要求される性能を満足できる．様々な大きさと性能のコアを備える不均質型マルチコアプロセッサは，電力効率の面で優れた選択肢である[9, 10].

残念なことに不均質型マルチコアプロセッサは，均質型に比較すると設計やプログラミングが困難である．この問題を解決するために，MCCP は均質型マルチコアプロセッサ上に不均質性を実現する[7]. 二つのクラスタを持つデュアルコア MCCP を例に用いて，クラスタゲーティングとコアゲーティングを利用して実現される不均質型マルチコアを図 3 に示す. 図 3a は大きなコアを二つ持つ均質型デュアルコアプロセッサである．性能が余っている時には，図 3b の様に一部のクラスタをオフにする．黒い四角はそのクラスタがオフの状態にあることを表している．クラスタゲーティングを利用することで，不要なクラスタをオフにし，アプリケーションが要求するに足るだけの性能を提供する．これは不均質型デュアルコアプロセッサである．図 3c では両コアでそれぞれひとつずつクラスタがオフになっており，小さなコアを二つ持つ均質型デュアルコアプロセッサとなっている．依然として性能に余

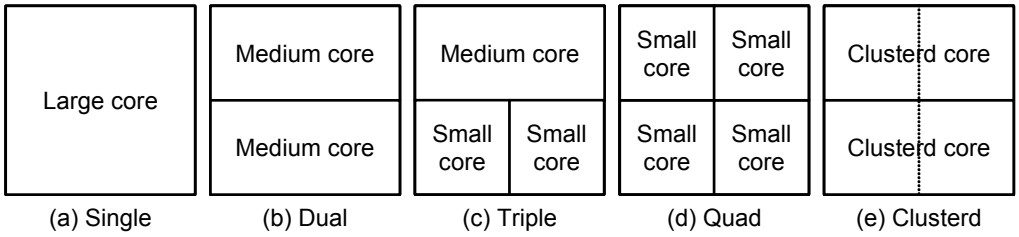

図 2. 様々なマルチコアプロセッサ

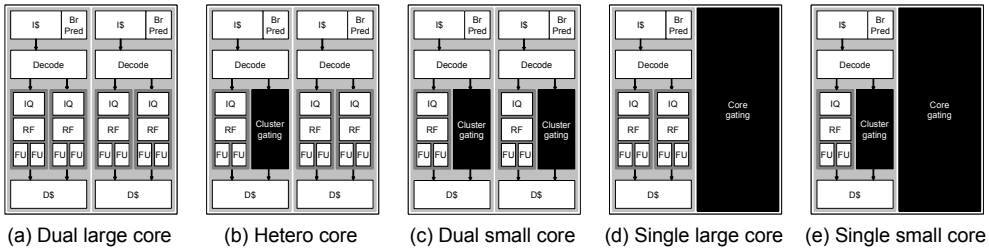

図 3. クラスタゲーティングとコアゲーティング

裕がある時には,図 3d のように一部のコアをオフにする．黒い四角はそのコアがオフの状態にあることを表している．コアゲーティングを利用することで，不要なコアをオフにし，アプリケーションが要求するに足るだけの性能を提供する．クラスタゲーティングとコアゲーティングを同時に利用すると,図 3e のように小さな単一プロセッサも構成出来る．

上述のように均質型マルチコアプロセッサ上に不均質性を実現出来るので，均質型を意識して作成したプログラムでも不均質性による恩恵を得ることが出来る．どのコアにどのスレッドを割り当てるべきかをプログラム時に考慮しなくて良くなるので，プログラミングが容易になると期待される．スレッドが要求する性能に足るコア構成を自在に用意できるので，不要な電力消費を削減出来る．

クラスタゲーティングを実装するには様々な選択肢が考えられる．一つはハードウエアによる方法である．コア内の専用ハードウエアがスレッドの振る舞いを観測し，必要なクラスタ数を決定する．ソフトウエアによる方法も考えられる．命令セットにクラスタをオン・オフするための命令を追加する．プログラマやコンパイラがその命令をスレッド中に挿入する．プログラミングの容易さに配慮すると，クラスタ数を明示するよりも必要な性能を明示出来る方が望ましい．この方式は，アノテーションや API の利用により実現出来ると考えられる．例えば，プログラマが挿入したアノテーションをコンパイラが適切な命令に変換し，クラスタ数を提示する．異なるマルチコアプロセッサ間での互換性や透過性は，ソースレベルで提供出来る．以上のような選択肢が存在するが，クラスタゲーティングの具体的な実現方法は将来の検討課題とする．本稿では，ハードウエアによる方法を扱う．

2.2. ディペンダビリティモード

MCCP はディペンダビリティを提供する点でも好ましい特徴を備えている．MCCP はマルチコアプロセッサであるから，DTR を利用出来る．プログラムを複製し，図 4a の様に複数のコアで冗長実行する．残念ながら，DTR は二つのコアで冗長実行するために，消費電力が増大する．

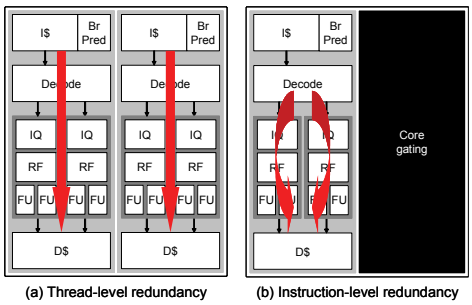

図 4. ディペンダビリティモード

MCCP は DIR を利用することも可能である．プログラム内の各命令を複製し,図 4b のように同一コア内の異なるクラスタで冗長実行する．しかし，実行命令数が二倍に増加するために，性能低下が深刻である．DTR における消費電力の問題と DIR における性能の

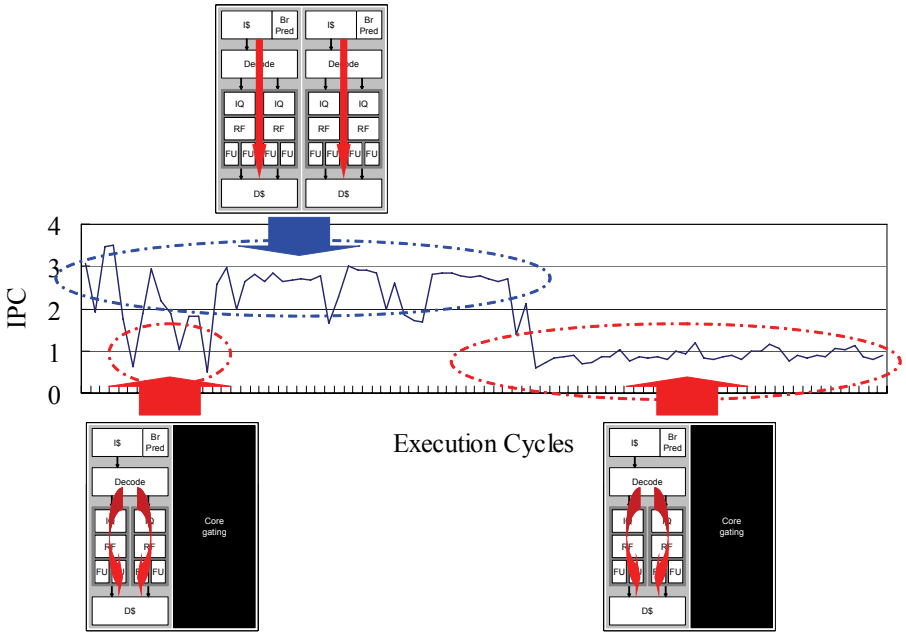

図 6 IPC を利用するモード選択

問題を解決するために，MCCP 上で両方式を組み合わせて利用することを提案する．この方式をハイブリッド方式（DHR: dependability with hybrid thread- and instruction-level redundancy）と呼んでいる．MCCP は要求される性能に基づいてディペンダビリティモードを選択する．ある場合には dtr モードになり，また別の場合には dir モードを選択する．これ以降ディペンダブル方式とディペンダビリティモードを明確に区別するために，方式には大文字（DTL，DIL，DHL）を使用し，モードには小文字（dtr，dir）を使用する．ディペンダビリティモードの選択に MCCP の持つ適用性を利用することで，DTR と DIR の組み合わせが可能になる．本稿の焦点はディペンダビリティモードの選択方法である．考えられる戦略のひとつは，プログラマに頼る方法である．プログラマが各スレッドの重要度を予め決定し，アノテーション等でプロセッサに伝える．他には OS による方式が考えられる．デッドラインなどの指標に基づいて，OS がスレッドの重要度を決定する．本稿では，完全に自動なハードウエアによる方式を提案する．

2.3. IPC を利用するモード選択

命令レベル並列性（ILP: instruction level parallelism）はプログラム毎に異なる．プロセッサが DTR だけに頼っていると，ILP が小さなプログラムでは電力を浪費してしまう．一方でプロセッサが DIR だけに頼っていると，ILP の大きなプログラムでは性能が大幅に低下してしまう．加えて，ILP はプログラム実行中でも大きく変化する．図 5 は SPEC2000 CINT ベンチマーク中の gcc をデュアルクラスタ・コアで実行した際の発行命令数の変化を示している．コアの構成の詳細は 3 節で述べる．X 軸は実行サイクル数を，Y 軸は 10,000 サイクル毎のサイクルあたりの平均発行命令数（発行 IPC）を現している．数百万命令に渡って発行 IPC が二倍以上の違いで変化していることがわかる．プロセッサが DTR だけに頼っていると，発行 IPC が小さな間には電力を浪費してしまう．一方でプロセッサが DIR だけに頼っていると，発行 IPC が大きな間には性能を著しく低下してしまう．この発行 IPC の変化を，ディペンダビリティモードの選択に利用することを検討する．

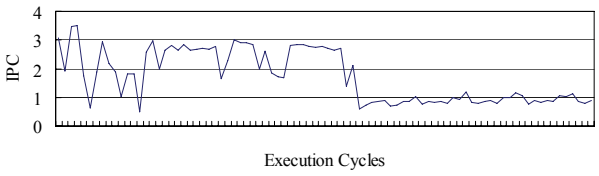

図 5. gcc における発行 IPC の変化

上述したように，DTR は発行 IPC が小さいと電力を浪費し，DIR は発行 IPC が大きいと性能を低下する．この観察結果を DTR と DIR の組み合わせに生かすことを考える．すでに述べたように，この組み合わせ方式を DHR と呼ぶ．図 6 に見られるように DHR は，発行 IPC が大きな時には dtr モードを，発行 IPC が小さな時には dir モードを利用する．発行 IPC が小さな時には演算器資源は余っているので，dir モードは深刻な性能低下を引き起こすことなくディペンダビリティを

提供出来る．一方で，dtr モードは状況に応じてオフになるので，無駄な電力の消費が抑えられる．

過去のプログラムの振る舞いで将来の振る舞いを予想できると仮定している．つまり，過去の発行 IPC を利用して将来の発行 IPC を予測する．浮動小数点数除算を避けるために，固定のサンプリング幅に発行された命令数を計数する．つまり，過去の発行 IPC ではなく過去の発行命令数で，将来の発行 IPC を予測するわけである．この予測された発行 IPC に基づいてディペンダビリティモードを選択する．dtr モード時に予測 IPC がある閾値（T_{T2I}）より小さくなると，dir モードに切り替える．同様に．dir モード時に予測 IPC がある閾値（T_{I2T}）より大きくなると，dtr モードに切り替える．

3. 評価環境

SimpleScalar/PISA ツールセット[11]を利用してシミュレーションを行う．各コアが二つのクラスタを持つデュアルコア MCCP で評価する．表 1 にプロセッサコアの構成をまとめる．フロントエンドと一次キャッシュは同一コア内の二つのクラスタで共有されている．同様に，二次キャッシュは同一チップ上の二つのコアで共有されている．dtr モード時に二つの結果を比較するためにはコア間の同期が必要であるが，それに要するオーバヘッドは無視出来るものとする．ベンチマークプログラムには，SPEC2000 CINT から選んだ 6 本の整数系プログラムを使用する．

表 1.プロセッサコア構成

Fetch width	8 instructions
L1 instruction cache	16K, 2-way, 1 cycle
Branch predictor	1K-gshare + 512-BTB
Dispatch width	4 instructions
Instruction window size	16 entries / cluster
Issue width	2 instructions / cluster
Commit width	4 instructions / cluster
Integer ALUs	2 units / cluster
Integer multiplires	2 units / cluster
Floating ALUs	2 units / cluster
Floating multiplires	2 units / cluster
L1 data cache ports	1 port / cluster
L1 data cache	16K, 2-way, 1 cycle
Unified L2 cache	512K, 2-way, 10 cycles
Memory	Infinite, 100 cycles

dir モードから dtr モードへの切り替え時には 100 サイクルのオーバヘッドを被ると仮定する．このような大きなオーバヘッドを考慮すると小さなサンプリング幅は無意味であるので，10,000 サイクルのサンプリング幅を選択する．T_{T2I} と T_{I2T} としてそれぞれ 2.0 と 1.6 を採用する．先頭の 10 億命令をスキップし，その後の 20 億命令を詳細にシミュレーションする．

消費電力については単純な仮定を置く．動作中のコア数に比例して電力を消費すると仮定する．つまり，dtr モード時には dir モード時の二倍の電力を消費すると仮定する．

ディペンダビリティの品質については，ディペンダブル方式やディペンダビリティモードに関係なく，均一な品質を提供出来ると仮定する．実際には，プロセッサ上で保護されている領域の違いによりディペンダビリティの品質は異なる[12]が，本稿は消費電力と性能との間のトレードオフに焦点を当てているため，このような単純な仮定を採用する．

4. 結果

図 7 には，DHR と DIR をそれぞれ採用するプロセッサの発行 IPC と EDP がまとめられている．折れ線グラフは発行 IPC を，棒グラフは EDP を表している．折れ線グラフのうち，▲は DIR の結果を■は DHR の結果を表している．二本の棒グラフのうち，左は DIR の結果を右は DHR の結果を表している．いずれの値も，DTR の相当する値に対して正規化されている．DTR と比較すると，DIR はコミット IPC を 27%も低下させているにも関わらず，EDP は 5%しか改善できていない．DHR はこの性能低下を軽減し，EDP を改善できている．DTR と比較すると，DHR はコミット IPC を 13%しか低下させず，その上 EDP を 13%も改善出来ている．DHR が全てのプログラムで EDP を改善出来ていることは強調するに値する．しかし DIR は二つのプログラムで EDP を悪化させている．以上の結果から，DHR は DTR における電力増大の問題と DIR における性能低下の問題を同時に軽減出来ていると結論付けられる．

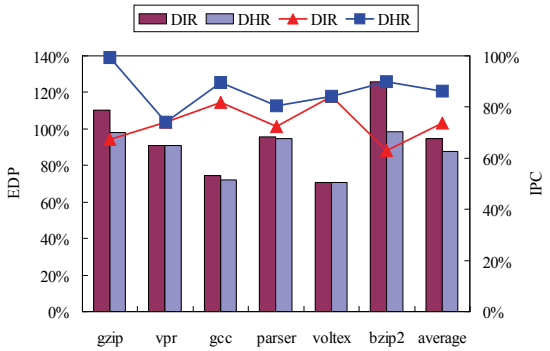

図 7. DHR と DIR の IPC と EDP
（サンプリング幅=10,000，オーバヘッド=100）

図 8 は DHR におけるディペンダビリティモードの内訳である．サイクル数で評価している．各棒グラフは三つの部分に分けられ，下から順に，dtr モードサイ

クルの割合，モード切替えサイクルの割合，そして dir モードサイクルの割合である．興味深いことに，ひとつのプログラムはほぼ全てを dtr モードで実行し，対照的に別の二つのプログラムはほぼ全てを dir モードで実行している．残りの三つのプログラムはモードを使い分けている．以上から，IPC に基づくモード選択はプログラムの特性を掌握出来ており，プログラムの特性に応じて演算資源を適応出来ていることがわかる．それゆえに，DHR は DTR の電力増大の問題と DIR の性能低下の問題を軽減出来たと考えられる．

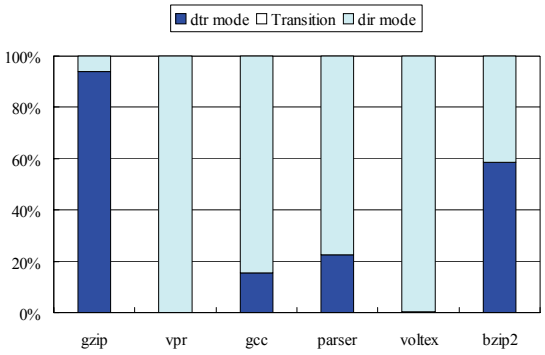

図 8.ディペンダビリティモードの内訳
（サンプリング幅=10,000，オーバヘッド=100）

5. おわりに

本稿では MCCP における電力と性能との間のトレードオフを検討した．マルチコアプロセッサは，スレッド冗長性を利用して SEU を検出可能なので，ソフトエラーの問題解決に向いている．しかし，残念なことに，冗長実行は多大な電力を消費する．本稿でスレッド冗長性と命令冗長性を組み合わせて利用することを提案し，スレッドの要求する性能に応じてディペンダビリティのモードを選択する方法について検討した．ILP が大きい時にはスレッド冗長性を利用し，逆に ILP が小さな時には命令冗長性を利用する．スレッド冗長性のみを利用する場合と比較して，提案方式はプロセッサ性能を 13%低下させるものの，EDP を 13%改善出来ることがわかった．一方で命令冗長性だけに頼ると，性能が 27%悪化する上に EDP は 5%しか改善出来ないこともわかった．これらのことから，MCCP 上で提案方式を利用する有効性が確認出来たと言える．

将来の課題のひとつは，ディペンダビリティの品質を評価できる指標を改善することが挙げられる．本稿では，いずれの方式も均一の品質を提供出来ると仮定した．実際には，方式が異なれば提供出来る品質も異なるはずである．現在，アーキテクチャレベル脆弱性（AVF: architectural vulnerability factor）[13]を考慮したディペンダビリティ品質の指標を検討中である．

謝辞

本研究の一部は，文部科学省科学研究費補助金・基盤研究 A（No. 19200004），および科学技術振興機構・CREST プロジェクトの支援によるものである．

文　献

[1] S. Borker, “Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation,” IEEE Micro, Vol. 25, No. 6, 2005.

[2] T. Karnik, P. Hazucha, and J. Patel, “Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes,” IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 2, 2004.

[3] S. S. Mukherjee, M. Kontz, and S. K. Reinhardt, “Detailed Design and Evaluation of Redundant Multithreading Alternatives,” 29th International Symposium on Computer Architecture, 2002.

[4] K. Sundaramoorthy, Z. Purser, and E. Rotenberg, “Slipstream Processors: Improving Both Performance and Fault Tolerance,” 9th International Conference on Architectural Support for Programming Languages and Operating Systems, 2000.

[5] J. Ray, J. C. Hoe, and B. Falsafi, “Dual use of Superscalar Datapath for Transient-Fault Detection and Recovery,” 34th International Symposium on Microarchitecture (2001)

[6] T. Sato and I. Arita, “In Search of Efficient Reliable Processor Design,” 30th International Conference on Parallel Processing, 2001.

[7] T. Sato and A. Chiyonobu, “Multiple Clustered Core Processors,” 13th Workshop on Synthesis and System Integration of Mixed Information Technologies, 2006.

[8] R. E. Kessler, “The Alpha 21264 Microprocessor,” IEEE Micro, Vol. 19, No. 2, 1999.

[9] M. Annavaram, E. Grochowski, and J. Shen, “Mitigating Amdahl’s Law through EPI Throttling,” 32nd International Symposium on Computer Architecture, 2005.

[10] R. Kumar, K. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, “Single-ISA Heterogeneous Multi-core Architectures: the Potential for Processor Power Reduction,” 36th International Symposium on Microarchitecture, 2003.

[11] T. Austin, E. Larson, and D. Ernst, “SimpleScalar: an Infrastructure for Computer System Modeling,” IEEE Computer, Vol. 35, No. 2, 2002.

[12] T. Funaki and T. Sato, “Dependability-Performance Trade-off on Multiple Clustered Core Processors,” 4th International Workshop on Dependable Embedded Systems, 2007.

[13] C. T. Weaver, J. Emer, S. S. Mukherjee, and S. K. Reinhardt, “Reducing the Soft-Error Rate of a High-Performance Microprocessor,” IEEE Micro, Vol. 24, No. 6, 2004.