社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ARC－177 (9)
2008－HPC－114 (9)
2008／3／5

信頼性と性能のトレードオフ評価指標の提案とそのマルチコアプロセッサへの適用

舟木 敏正[†] 佐藤 寿倫[‡,††]

[†] 九州工業大学大学院 情報工学研究科
[‡] 九州大学 システムＬＳＩ研究センター
[††] 独立行政法人 科学技術振興機構，CREST

近年，性能と消費電力だけでなく信頼性もプロセッサに要求されている．これらはお互いにトレードオフの関係にあるので，プロセッサの評価のために新たな指標が必要である．その最初の段階として，本稿では性能と信頼性のトレードオフ指標を提案する．この指標を我々が現在検討中のマルチ・クラスタ型コア・プロセッサ (MCCP:Multiple Clustered Core Processsor) に適用する．MCCP は性能・消費電力・信頼性のトレードオフを考慮してその構成を変更し，冗長実行モードを選択する．したがって，各実行モードの性能・電力・信頼性の評価指標が必要となる．提案指標を用いることで，命令冗長実行では性能低下に見合った信頼性が確保できないことが判明した．

A Metric for the Dependability-Performance Trade-off and its Application to a Multcore Processor

TOSHIMASA FUNAKI[†] TOSHINORI SATO[‡,††]

[†]Graduate School of Computer Science and System Engineering, Kyushu Institute of Technology
[‡]System LSI Research Center, Kyushu University
[††]Japan Science and Technology Agency, CREST

Modern microprocessors require dependability as well as high-performance and power-efficiency. Since there are trade-offs between them, a new metric considering the trade-offs to evaluate microprocessors. This paper proposes a metric for the dependability-performance trade-off as a first step toward considering the 3-dimensional (3D) trade-off. Using the metric, this paper evaluates Multiple Clustered Core Processor (MCCP) that is under investigation. We consider the 3D trade-off on the MCCP by selecting an appropriate processor configuration and a redundancy mode. We find that the MCCP diminishes both performance and dependability in the instruction-level redundancy mode.

1. はじめに

LSI の微細化が進展することで搭載可能なトランジスタ数は増加し，プロセッサの性能・機能は向上してきた．しかし，微細化の進展はソフトエラー増加による信頼性の低下という問題を引き起こしている．トランジスタの電荷容量が減少し，二次宇宙線やα線などソフトエラーの影響を受けやすくなっており，今後プロセッサの高信頼性設計が重要となる．また，高性能・省電力に対する要求は今後も続くことが予測される．今後 LSI の設計において，性能・電力・信頼性の三つを考慮した設計が重要となる．しかし，性能・電力・信頼性の間にはトレードオフがあり，これらを考慮してプロセッサを評価するための指標が必要である．本稿では，その評価指標策定の第一段階として，性能と信頼性の間のトレードオフ評価指標を検討し，MITF(Mean Instruction To Failure)[12] の利用を提案する．

性能・電力・信頼性の間のトレードオフを考慮するために，我々はマルチ・クラスタ型コア・プロセッサ (MCCP:Multiple Clustered Core Processor) を提案している[10]．MCCP は性能・電力・信頼性の三つを考慮し，適切なプロセッサ構成と冗長実行モードを選択する．そのために，冗長実行しない時，スレッドレベル冗長実行時，命令レベル冗長実行時の性能・電力・信頼性のトレードオフを評価する指標が必要となる．本稿では，前述の性能と信頼性のトレードオフ評価指標を MCCP に適用し，MITF の有用性を調べるとともに，MCCP の評価を行う．

2. マルチコアを利用する信頼性の提供

初めに定義をする．本稿で扱うフォールトとはSEU(Single Event Upset) によって起こる 1 ビットの反転であり，エラーとはフォールトによってプログラムの実行結果に以上が生じることである．

信頼性を提供するための最も単純な方式は，冗長実行である．時間的冗長性と空間的冗長性を利用する．マルチコアプロセッサは空間的冗長性の利用に適している[3),7)]．SEU を検出するために，プログラムを複製し同一プロセッサ上の異なるコアで冗長実行する．二つの結果が一致しなければ，SEU が検出されたことになる．CRT[7)] はマルチコアプロセッサであり，二つの冗長なスレッドを独立したコア上で実行する．CRTR[3)] は CRT を改良したもので，SEU の検出後にプロセッサ状態を回復可能である．マルチコアプロセッサで SEU に対処するためには，同期，複製，検出，そして回復のそれぞれで考慮しなければならない点がある[6)]．以下では CRTR で採用されている方法を説明する．

冗長に実行されるプログラムの結果を比較するためには，両者の同期を取る必要がある．二つのプロセッサコア間にキューを配置し，先行コアがキューに書き込んだ演算結果を後続コアが読み出し，二つの演算結果が比較される．このようにキューを利用して同期が取られる．複製された命令が別々のコアで独立に実行されるので，同じロード命令が読み出す値が一貫しないという問題が生じうる[6)]．冗長な二つのロード命令に挟まれたストア命令が値を更新する可能性がある．そのため，一貫性を保つための特別なキューが利用される．上記の同期の説明で述べたとおり，キューを利用して二つの演算結果が比較される．比較により SEU が検出されるとプロセッサ状態を回復しなければならない．後続コアの状態を先行コアにコピーすることで，SEU 検出前の状態に回復される．

3. 性能と信頼性のトレードオフ評価指標

信頼性を示す指標として MTTF(Mean Time To Failure) が広く使用されている．MTTF は時間とエラー率の関係を表している．同じ時間でプログラムを実行しても，プロセッサの性能が違えば，実行命令数は異なる．そのため，MTTF ではプロセッサの性能を反映することができない．プロセッサの性能と信頼性のトレードオフを表す指標として MITF[12)] が提案されている．MITF では，プログラムが処理する命令数とエラー率の関係を表しており，プロセッサの性能を考慮することが可能である．MITF の定義を以下に示す．AVF(Architectural Vulnerability Factor)[8)] は，プロセッサ内部で発生したソフトエラーがプログラム実行に影響をあたえる確率を示す．言い替えると，フォールトがエラーとなる確率である．

$$MITF = \frac{\#\_instructions}{\#\_errors} \tag{1}$$

$$= \frac{\#\_instructions}{\frac{total\_time}{MTTF}} \tag{2}$$

$$= \frac{\#\_instructions}{\frac{total\_cycle}{f \times MTTF}} \tag{3}$$

$$= IPC \times f \times MTTF \tag{4}$$

$$= \frac{IPC \times f}{error\_rate} \tag{5}$$

$$= \frac{IPC \times f}{fault\_rate \times AVF} \tag{6}$$

AVF を見積もる方法は文献[8)] で提案されている．式 (7) に AVF を求める式を示す．ACE とは，Architectural Correct Execution であり，プログラムの結果に影響する値や命令をさす．B_{ACE} はサイクルあたりのストラクチャに入る値や命令の平均ビット幅を，L_{ACE} はその値や命令のストラクチャにおける平均滞在時間を表している．ストラクチャとは，命令ウィンドウやレジスタファイルなどプロセッサを構成する機構のことである．$\#\_bits$ はストラクチャのビット数を，$\#\_cycles$ は実行時間を表している．

$$AVF = \frac{B_{ACE}}{\#\_bits} \times \frac{L_{ACE}}{\#\_cycles} \tag{7}$$

ACE を正確に求めることは困難であるため，全ての命令から ACE でないものを除いたものを ACE とする．ACE でないものは，nop や有効でないデータ，予測ミスに関連する値や命令がある．この文献の方法では，ストラクチャごとの AVF しか見積もることができない．プロセッサ全体の AVF は，各ストラクチャの面積と各ストラクチャの AVF の積の和で求める．

これまでシングルコアの MITF は検討されているが，マルチコアプロセッサで冗長実行する場合の MITF の研究は未だ無い．

4. マルチ・クラスタ型コア・プロセッサ

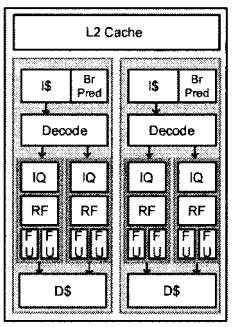

図 1 MCCP

図 1 に MCCP の例を示す．従来のマルチコアプロ

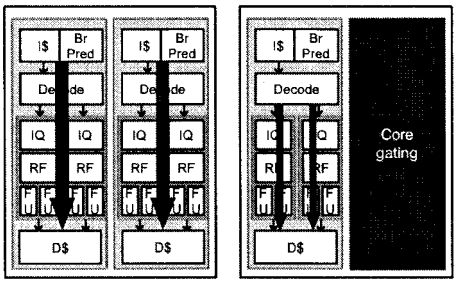

図 2 スレッド冗長実行と命令冗長実行

セッサとの違いは，各コアが単一なコアではなくクラスタ型マイクロアーキテクチャ[5] となっていることである．各プロセッサコアは複数のクラスタを持つ．各クラスタはそれぞれ，命令ウィンドウ (IQ)，演算器 (FU)，レジスタファイル (RF) を持っている．キャッシュ(I$，D$)，分岐予測器 (BrPred)，デコーダ (Decode) はクラスタ間で共有する．L2 キャッシュはプロセッサコア間で共有する．クラスタとコアのゲーティングを行い，要求に見合うだけの性能を提供することで，消費電力を削減する[10]．

4.1 二つの冗長実行モード

MCCP は空間的冗長性を利用し，一つのプログラムを二つのプロセッサコアまたはクラスタで冗長実行する．各プロセッサコアから出力されたプログラムの実行結果，または各クラスタから出力された命令の実行結果を比較することで，SEU を検出する．検出後にプロセッサ状態を回復するには，2 節で説明した機構が用意されているとする．

MCCP の冗長実行には二つの冗長実行モードがある[2),13)]．一つはスレッド冗長実行であり，実行するプログラムを複製し，異なるプロセッサコアで冗長実行する．図 2(a) にスレッド冗長実行の動作を示す．図のように，プロセッサコア全体で一つのプログラムを冗長実行する．プロセッサコア全体で冗長実行するので，ほとんどの SEU を検出することが可能である．このスレッド冗長実行は大きな信頼性が求められる時に選択される．

もう一方は命令冗長実行である．命令デコード時にプログラム中の命令を複製し，同一プロセッサコア上の複数のクラスタで冗長実行を行う．図 2(b) に命令冗長実行の動作を示す．キャッシュとデコーダの部分は冗長実行されないので，スレッド冗長実行に比べ信頼性は劣る．一方でスレッド冗長実行より消費電力が少ない．命令冗長実行は，信頼性と省電力が共に要求されるときに選択される．

MCCP は，要求される信頼性に応じてスレッド冗長実行と命令冗長実行を切替える．そのためには各モードの提供する性能と信頼性を知る必要がある．つづいて冗長実行を行わない時，スレッド冗長実行時，命令冗長実行時の MITF を定式化する．

4.2 冗長実行を行わない時の MITF

冗長実行を行わない時は，クラスタ 2 台用いたプロセッサコア 1 台で実行する．式 (6) より冗長実行を行わない時の MITF の式を導く．

$$MITF_{thread} \tag{8}$$

$$= \frac{IPC_L \times f}{fault\_rate \times AVF_L} \tag{9}$$

$$= \frac{IPC_L \times f}{fault\_rate\_per\_bit \times \#\_bit\_L \times AVF_L} \tag{10}$$

$$= \frac{IPC_L \times f}{E_b \times N_{b_L} \times AVF_L} \tag{11}$$

L はプロセッサコア 1 台のプロセッサ構成を表す．IPC_L はプロセッサ構成 L の IPC を，E_b はビットあたりのフォールト率を，N_{b_L} は L のビット数を，AVF_L は L の AVF を表している．$fault\_rate$ はチップ全体のフォールト率を表しており，ビット当たりのフォールト率 E_b と L のビット数 N_{b_L} の積で表せる．

4.3 スレッド冗長実行時の MITF

スレッド冗長実行では，プロセッサコア 2 個を用いた冗長実行で実現する．スレッド冗長実行の MITF の以下に示す．

$$MITF_{thread} \tag{12}$$

$$= \frac{IPC_L \times f}{error\_rate\_PCs} \tag{13}$$

$$= \frac{IPC_L \times f}{fault\_rate\_per\_bit \times \#\_bit\_PCs \times AVF_{PCs}} \tag{14}$$

$$= \frac{IPC_L \times f}{E_b \times N_{b_{PCs}} \times AVF_{PCs}} \tag{15}$$

PCs は L2 キャッシュなどのプロセッサコア間で共有する部分を表している．複数のコアで冗長実行を行うため，プロセッサコア部分は SEU から守られる．コア間で共有する部分で SEU が発生した場合にエラーとなる．そのため，式 (15) の $error\_rate$ は PCs でのエラー率となる．PCs でのエラー率はビット当たりのフォールト率 E_b，PCs のビット数 $N_{b_{PCs}}$，PCs の AVF の積で表せる．

4.4 命令冗長実行時の MITF

命令冗長実行は，クラスタ 2 個を用いた冗長実行で実現する．命令冗長実行の MITF の式を以下に示す．

$$MITF_{instruction} \tag{16}$$

$$= \frac{IPC_S \times f}{error\_rate} \tag{17}$$

$$= \frac{IPC_S \times f}{error\_rate\_PCs + error\_rate\_CLs} \tag{18}$$

$$= \frac{1}{fault\_rate\_per\_bit} \times \frac{IPC_S \times f}{\#\_bit\_PCs \times AVF_{PCs} + \#\_bit\_CLs \times AVF_{CLs}} \tag{19}$$

$$= \frac{IPC_S \times f}{E_b \times N_{b_{PCs}} \times AVF_{PCs} + E_b \times N_{b_{CLs}} \times AVF_{CLs}} \tag{20}$$

S はクラスタ 1 台を用いたプロセッサコア 1 台のプロセッサ構成を，IPC_S はプロセッサ構成 S の IPC を，CLs はクラスタ間で共有する部分を表している．複数のクラスタで冗長実行を行うため，クラスタ部分は SEU から守られる．クラスタ間で共有する部分と，プロセッサコア間で共有する部分で SEU が発生した場合にエラーとなる．そのため，式 (17) の $error\_rate$ は PCs でのエラー率と CLs でのエラー率の和となる．PCs でのエラー率はビット当たりのフォールト率 E_b，PCs のビット数 $N_{b_{PCs}}$，PCs の AVF の積で表せる．同様に CLs でのエラー率はビット当たりのフォールト率 E_b，PCs のビット数 $N_{b_{PCs}}$，PCs の AVF の積で表せる．

5. 評価指標の MCCP への適用

本節では，提案した評価指標を MCCP に適用する．それぞれのの MITF を求め調査を行う．

MITF を見積もるためには，E_b，N_b，IPC，そして AVF を知る必要がある．まず E_b については，文献 9) より 0.01FIT を用いる．また，L1 D キャッシュ，L1 I キャッシュ，L2 キャッシュ，DTLB，ITLB が ECC(Error Correcting Code) によって守られているとする．そのときのエラー率は 10^4 分の 1 倍に減少する[11]．

表 1 プロセッサの回路規模 (mm^2)

L1 D cache	13.0
L1 I cache	10.4
ITLB	2.2
DTLB	2.2
Fecth, BrPred	4.5
Decode	1.7
ROB	16.6
IQ	2x 1.8
RF	2x 2.9
FU	2x 6.5
Misc	2.4
Routing	26.4
512K L2 cache	110
Misc	6.1
Cache coherence	6.3
I/O	13.7

総ビット数 N_b は面積で近似を行う．表 1 に回路規模を示す[10]．表は上部が MCCP のプロセッサコアの構成要素の面積を，下部がプロセッサコア間で共有するプロセッサの構成要素の面積を表している．

IPC と AVF は sim-SODA[1] を使用してアーキテクチャレベル・シミュレーションによって求める．命令セットは Alpha である．small_core，large_core の構成を表 2 に示す．表中の small_core はクラスター台を用いたプロセッサコア一台の構成を，large_core はクラスタ二台を用いたプロセッサコア一台の構成を表している．冗長実行を行わない時とスレッド冗長実行は large_core での IPC を，命令冗長実行は small_core での IPC を適用する．

IQ，ROB，Func Units，RFs，L1 Dcache，DTLB の AVF を測定した．それ以外の部分の AVF は sim-SODA では測定できないので 1 と仮定した．

表 2 プロセッサ構成

	small_core	large_core
fetch/slot/map/	4/4/2/	4/4/4/
issue/commit width	2/4	4/4
IQ	16	32
Int ALU/MULT	2/2	4/4
RFs	40	80
ROB	128	128

5.1 IPC と AVF の見積り

5.1.1 IPC の見積り結果

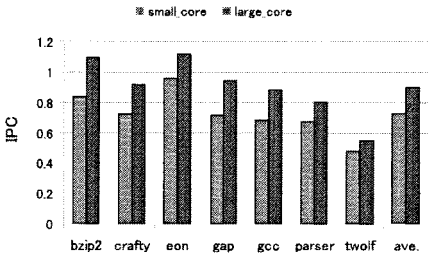

図 3 IPC の評価

まず，IPC のシミュレーション結果を図 3 に示す．縦軸は IPC を，横軸は各ベンチマークを表している．グラフの左側は small_core の IPC を，右側は large_core の IPC を表している．small_core は large_core に比べ，命令発行幅，演算器個数，IQ の計算資源が少ない．このため，ILP を抽出できず IPC が減少している．

5.1.2 AVF の見積り結果

次に，AVF のシミュレーション結果を図 4 に示す．横軸は AVF を，縦軸は各ベンチマークと各ストラクチャを表している．グラフの左側は small_core の AVF を，右側は large_core の AVF を表している．large_core は small_core と比較して，IQ と FU で AVF が低下した．これは式 (7) の分母の $\#\_cycles$ が低下したためである．また，ROB は small_core のほうが AVF は高くなっている．ROB は small_core と large_core で同じエントリ数である．しかし，small_core の方が，演算処理を行う機構の数が少ないため，式 (7) の L_{ACE} が大きくなる．そのため，ROB では small_core の方が AVF は高くなっている．

5.2 MITF の見積り

上記で仮定したフォールト率とビット数と，シミュレーションにより求めた IPC と AVF の評価結果を，3 節で求めた各実行モードの MITF の式に代入し，それぞれの MITF を求める．

MCCP の MITF を図 5 に示す．全ての値は，冗長実行を行わない時の MITF の値により正規化されて

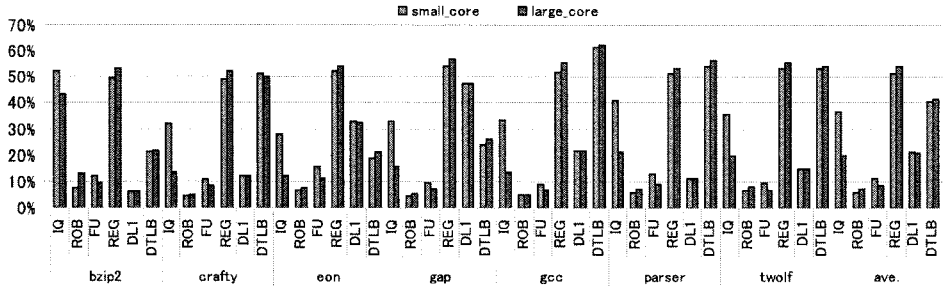

図 4　AVF の評価

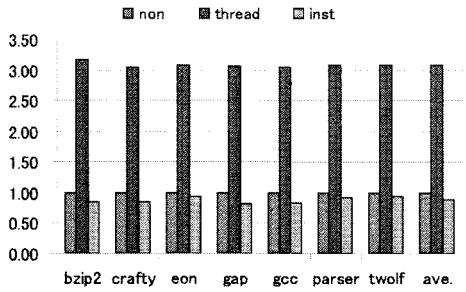

図 5　MITF

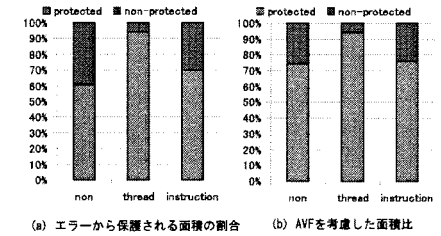

図 6　エラーから保護される面積の割合

いる．MITF は冗長実行を行わない時に比べ，スレッド冗長実行で平均約 3 倍向上し，命令冗長実行で約 12%が低下した．

式 (11)，(15)，(20) にあるように，MITF は IPC，ビット数 (面積)，AVF の三つの値で決定される．図 6 にエラーから保護される面積とエラーに脆弱な面積の比を表す．横軸は各実行モードを，縦軸は面積の割合を表している．グラフの下側はエラーから保護される面積の割合を，上側はエラーに脆弱な面積の割合を表している．

図 6(a) はエラーから保護される面積とエラーに脆弱な面積の比を，図 6(b) は冗長実行，ECC，AVF によりエラーからマスクされる面積とエラーに脆弱な面積の比を表している．冗長実行，ECC，AVF を考慮したエラーに脆弱な面積は各プロセッサストラクチャの面積と AVF の積の和により求める．ビット当たりのフォールト率は冗長実行を行わない場合，スレッド冗長実行，命令冗長実行で同じなので，冗長実行，ECC，AVF を考慮した場合のエラーに脆弱な面積は式 (11)，(15)，(20) の分母となる．エラーからマスクされる面積は，冗長実行と ECC によりエラーより保護される面積と，エラーに脆弱な面積のうち AVF によりマスクされる面積の和である．つまり，全面積と冗長実行と AVF を考慮したエラーに脆弱な面積の差である．図 6(a) に示すように，冗長実行を行わない時は 61%，スレッド冗長実行で 94%，命令冗長実行で 70% の面積が冗長実行と ECC でエラーから保護される．エラーに脆弱な面積の割合に AVF を考慮する．AVF を考慮した面積の割合を図 6(b) に示す．冗長実行を行わない時は 74%，スレッド冗長実行は 94%，命令冗長実行は 77%エラーから保護される．冗長実行を行わない時と命令冗長実行の保護される面積の割合の差は 3%しかない．

図 3 に示すように，small_core における IPC は演算資源の欠乏により large_core に比べ平均で 20%低下している．IPC，ビット数，AVF の三つの値を考慮すると，IPC の低下の割合がエラーに脆弱な面積の割合と AVF の低下の割合より大きかったため，命令冗長実行の MITF が低下した．IPC の低下が MITF の低下の大きな要因であることがわかる．ILP の抽出が困難な逐次処理のようなプログラムを処理すると，small_core と large_core との IPC の差が小さくなり，命令冗長実行は，冗長実行を行わない時より MITF が高くなると予想できる．ILP が容易に抽出できるプログラムは，冗長実行を行わない，またはスレッド冗長実行を行うほうが良い．

測定不可能部分の AVF によっては結果が変動する可能性がある．Routing などの測定不可能部分の AVF によっては，冗長実行を行わない時より命令冗長実行の方が MITF が良くなる可能性がある．図 7 に測定不可能部分の AVF を 0%から 100%の間で変化させたときの各冗長実行モード MITF を示す．コア構成とプロセッサストラクチャ毎に AVF の値は異なるが，測定不可能部分の AVF は全て同じ場合と仮定している．縦軸は MITF の値を表しており，全ての値は冗長実行を行わない時の MITF の値により正規化されている．横軸は AVF を表している．今回の評価は測定不可能部分の AVF を 100%としている．測定不可能部分の

AVF が 39%以下になると，命令冗長実行は冗長実行を行わない時よりも MITF が良くなる．測定不可能部分の AVF の値次第では，命令冗長実行は冗長実行を行わない時よりも MITF が良くなる可能性がある．

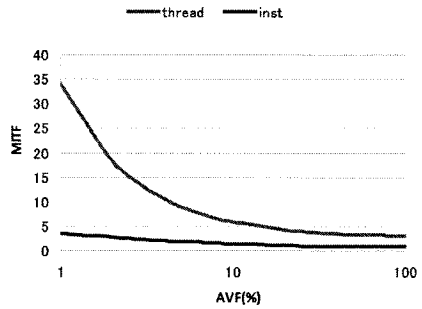

図 7 測定不可能部分の AVF の値を変化させたときの MITF

以上の評価より，スレッド冗長実行は信頼性を確保することが可能であるが，命令冗長実行では冗長実行を行わない時よりも信頼性が損なわれることがわかった．つまり，冗長実行を行わない時と命令冗長実行時では，MCCP は信頼性と性能のトレードオフを考慮することは困難であるということを確認できた．しかし，この結果は MITF が役に立たないことを意味しない．性能と信頼性のトレードオフを検討する際に，性能が重要な要素となることを表しており，MITF が評価指標として妥当であることを示している．

6. ま と め

微細化によりソフトエラーが大きな問題となりつつある．信頼性だけでなく，高性能・省電力の要求は今後も続くことが予測される．

性能・電力・信頼性の間トレードオフを考慮するために，MCCP を提案している．MCCP では性能・電力・信頼性を考慮し，適切な実行モードを選択することが目標である．そのために，各実行モードの性能・電力・信頼性の評価指標が必要となる．本論文では，その評価指標を決める最初の段階として，まず性能と信頼性のトレードオフの評価指標を提案した．

提案する評価指標を MCCP に適用し，評価を行なった結果，命令冗長実行時では，MCCP は，信頼性と性能のトレードオフを考慮することは困難であるということが判明した．性能と信頼性のトレードオフを考えるためには，MCCP の構成を再検討する必要があると考えられる．

謝　　辞

本研究の一部は，文部科学省科学研究費補助金 (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．

参 考 文 献

1) X. Fu et al., Sim-SODA: Unified framework for architectural level software reliability analysis, Workshop on Modeling, Benchmarking and Simulation, 2006.
2) T. Funaki et al., Dependability-performance trade-off on multiple clustered core processors, 4th Int. Workshop on Dependable Embedded System, 2007.
3) M. Gomaa et al., Transient-fault recovery for chip multiprocessors, 30th Int. Symp. on Computer Architecture, 2003.
4) G. Hamerly et al., SimPoint 3.0: Faster and more flexible program analysis, Workshop on Modeling, Benchmarking and Simulation, 2005.
5) R. E. Kessler, The Alpha 21264 microprocessor, IEEE Micro, 19(2), 1999.
6) C. LaFrieda et al., Utilizing dynamically coupled cores to form a resilient chip multiprocessor, 37th Int. Conf. on Dependable Systems and Networks, 2007.
7) S. S. Mukherjee et al., Detailed design and evaluation of redundant multithreading alternatives, 29th Int. Symp. on Computer Architecture, 2002.
8) S. S. Mukherjee et al., Measuring architectural vulnerability factors, IEEE Micro, 23(6), 2003.
9) E. Normand, Single event upset at ground level, IEEE Trans. Nuclear Science, 24(6), 1996.
10) T. Sato et al., Multiple clustered core processors, 13th Workshop on Synthesis and System Integration of Mixed Information Technologies, 2006.
11) M. Sugihara et al., A Simulation-based soft error estimation methodology for computer system, 7th Int. Symp. on Quality Electronic Design, 2006.
12) C. T. Weaver et al., Reducing the soft-error rate of a high-performance microprocessor, IEEE Micro, 24(6), 2004.
13) 舟木 他, マルチ・クラスタ型コア・プロセッサにおける信頼性と性能のトレードオフ, 情報処理学会九州支部若手の会セミナー, 2007.
14) 佐藤 他, 性能・消費電力・信頼性の間のトレードオフを考慮できるマルチ・クラスタ型コア・プロセッサ, 信学技報 CPSY2007-31, 2007