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

舟木, 敏正
九州工業大学情報工学部知能情報工学科

佐藤, 寿倫
九州大学システムLSI研究センター

出版情報：情報処理学会九州支部若手の会セミナー講演論文集, pp.15-20, 2007-09. 情報処理学会九州支部
バージョン：

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

舟木　敏正 †　　佐藤　寿倫 ‡

† 九州工業大学 情報工学部 知能情報工学科
‡ 九州大学 システムＬＳＩ研究センター

近年では性能と消費電力だけでなく，信頼性を考慮したプロセッサが要求されている．この３つの要求に対して、我々はマルチ・クラスタ型コア・プロセッサを提案している．提案方式では，性能，消費電力，信頼性のトレードオフを考慮したプロセッサ構成でプログラムを実行する．高性能が要求される時はプロセッサ全体で動作し，省電力が要求される時はプロセッサの一部で動作し，使用しない部分の電源電圧の供給を止める．また高信頼性を必要とするときは空間的冗長性を利用し，冗長実行を行う．本稿では，提案手法の性能と信頼性のトレードオフに着目し，２つの冗長実行モードの性能と信頼性のトレードオフの定式化を行う．

Formulating Dependability-Performance Trade-off for Multiple Clustered-Core Processors

Toshimasa Funaki†　Toshinori Sato‡

†Department of Artificial Intelligence, Kyushu Institute of Technology
‡System LSI Research Center, Kyushu University

Modern microprocessors requires dependability as well as high-performance and power-efficiency. We are investigating multiple clustered-core processors to achieve three requirements. The proposed processor executes the program with considering the trade-off between performance, power, and dependability. When high performance is requested, the whole processor is utilized. In contrast, when low power is expected, the portion of the processor is utilized. The unused portion is turned off and thus power consumption is reduced. Furthermore, when high dependability is required, the multiple clustered-core processor exploits space redundancy and redundantly executes the program. In this paper, we focus on the trade-off between performance and power, and formulate the trade-off for two dependability schemes we propose.

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，プロセッサの性能と機能は向上してきた．しかしプロセッサの高性能化・多機能化は消費電力の増大という問題を引き起こしている．消費電力と性能はトレードオフの関係にあり，高性能かつ低消費電力なマイクロプロセッサの設計は困難になっている．また最近では，LSI の一層の微細化により，二つの原因による信頼性の低下が深刻化している．

一つは，ソフトエラー率の増加による信頼性の低下である．ソフトエラーとはデバイスの動作中にランダムに発生する一時的な誤動作である．微細化技術の進展により，トランジスタはソフトエラーを引き起こす中性子などの外乱の影響を受けやすくなっている．従来は宇宙空間向けの LSI や医療機器のようなシステムでは考慮されていたが，民生機器の設計には考慮されていなかった．今後は，民生機器向けの LSI においても高信頼化設計が重要となる．

もう一つは，ばらつきの増加による信頼性の低下である．LSI におけるばらつきは，チップ間ばらつきとチップ内ばらつきに分類される．近年問題になっているのはチップ内ばらつきであり，特にランダムなばらつきが深刻となっている[1)]．1. 不純物密度の不均一な分布，2. リソグラフィーにおけるレイアウトパタンの再現性低下，3. 電力消費の不均一さに起因する電源のゆらぎ，4. 温度分布の不均一さの 4 つが，チップ内ばらつきの原因として考えられる．これらによる閾値電圧のゆらぎはトランジスタ遅延に影響し，回路動作の安定性を損なう．微細化の進展により，ばらつきの増加はチップの動作の安定性や，チップの品質に影響をあたえ，ばらつきを考慮した設計が重要となる．

性能・電力・信頼性の三つの問題を解決するために，我々はマルチ・クラスタ型コア・プロセッサを提案し

ている[2]．マルチ・クラスタ型コア・プロセッサは複数のプロセッサコアから構成される．各プロセッサコアは複数のクラスタを持つ．高性能と省電力を提供するために，マルチ・クラスタ型コア・プロセッサはプログラムに適応したコア構成で実行する．動作していないプロセッサコアやクラスタは電源電圧の供給を止めることで電力を削減する．高性能を必要とするときは，プロセッサ全体で並列性を利用し高速に実行する．高性能が必要とされないときは，一つのコアで実行する．それでも能力に余裕があれば，一部のクラスタをオフにする．プログラムの求める性能に見合ったコアの構成で，無駄な電力消費をなくしプロセッサの省電力化を目指している．

またマルチ・クラスタ型コア・プロセッサは，高信頼性を提供できる．空間的冗長性を利用し，一つのプログラム (または命令) を二重に実行する．二つの実行結果を比較し，違う結果が出力されれば再実行し，高い信頼性を得ることができる．また，プログラムの重要度に応じて信頼性の粒度を変更することができる．重要度の高いプログラムなら，プロセッサコアを使用しスレッドレベルの冗長実行を行う．重要度の低いプログラムなら，クラスタを使用し命令レベルの冗長実行を行う．性能と信頼性の間にはトレードオフがある．例えば，一つのプログラムをプロセッサコア二台で冗長実行するよりも，並列実行するほうが性能はよくなる．しかし，冗長実行は並列実行より信頼性が高くなる．このように，性能と信頼性のトレードオフは重要で，特にマルチ・クラスタ型コア・プロセッサでは信頼性の粒度を変更して実行するので，性能と信頼性のトレードオフを数量化し評価することは重要である．本稿では，スレッドレベルの冗長実行と命令レベルの冗長実行の性能と信頼性のトレードオフについて定式化を行う．

本稿の構成は以下の通りである．2 節でクラスタ型コアを持つマルチコアプロセッサアーキテクチャを提案する．3 節で信頼性の定式化について説明する．4 節で性能と信頼性のトレードオフを見積もる．最後に 5 節でまとめと今後の課題を述べる．

2. マルチ・クラスタ型コア・プロセッサ

図 1 にマルチ・クラスタ型コア・プロセッサの構成を示す．マルチ・クラスタ型コア・プロセッサは複数のプロセッサコアから構成される．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，独立した命令ウィンドウ (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．

2.1 節で性能と消費電力のトレードオフについて，2.2 節で信頼性を保障する方法について，2.3 節でばらつきへの配慮を述べる．

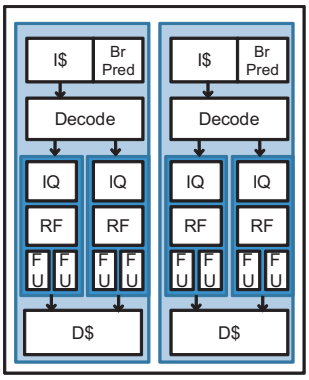

図 1 マルチ・クラスタ型コア・プロセッサ

2.1 性能と消費電力のトレードオフ

マルチ・クラスタ型コア・プロセッサではコアゲーティングとクラスタゲーティングによって不要な電力消費を抑えることができる[2]．コアゲーティングは動作しないプロセッサコアの電源電圧の供給を止めることである．クラスタゲーティングは，コアゲーティングと同様に動作しないクラスタの電源電圧の供給を止めることである．コアゲーティングとクラスタゲーティングの動作を図 2 に示す．図中 (a) はコアゲーティング，クラスタゲーティング行わない場合，(b) と (c) はクラスタゲーティングを行う場合，(d) はコアゲーティングを行った場合，(e) はクラスタゲーティングとコアゲーティングを併用した場合を表す．

高性能が要求されるプログラムでは，図 2(d) のようにプロセッサコアを使用して実行する．性能が必要な時には回路規模の大きいプロセッサコアで実行し性能を保障する．さらに高性能が要求されるプログラムでは，図 2(a) のようにプロセッサ全体でプログラムを実行する．一方で，性能が要求されないプログラムでは図 2(e) のように一部のクラスタで実行する．性能が要求されないときは回路規模の小さなクラスタで実行することで，電力を削減する．マルチ・クラスタ型コア・プロセッサは，性能と消費電力の要求度に応じてコアゲーティングとクラスタゲーティングを使用する．無駄な電力消費を抑え要求される性能を提供する．

2.2 信　頼　性

マルチ・クラスタ型コア・プロセッサは信頼性を確保することができる．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．各プロセッサコアまたは各クラスタから出力されたプログラムまたは命令の実行結果を比較することで，エラーを検出することができる．出力された結果の片方だけエラーが起これば，エラーを検出できる．出力された結果の両方にエラーが起こっても，出力された結果が同じでなければエラー検出される．つまり，同じ間違った結果が出力されたとき以外は，エラーを検出することができる．このように，エラーを検出できない場合は非常に少ないので，冗長実行は高い信頼性を得ることができる．

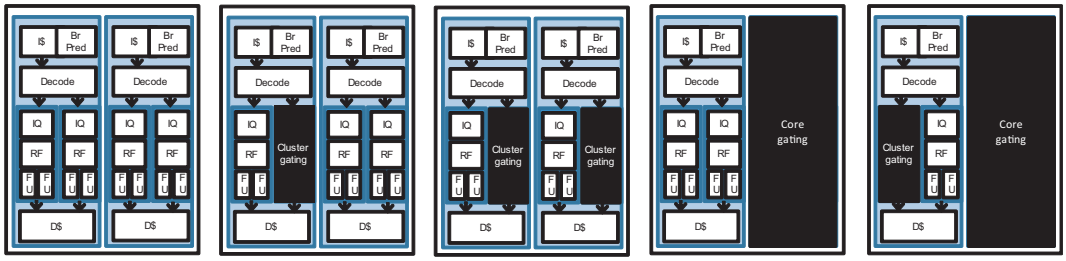

図 2 コアゲーティングとクラスタゲーティング

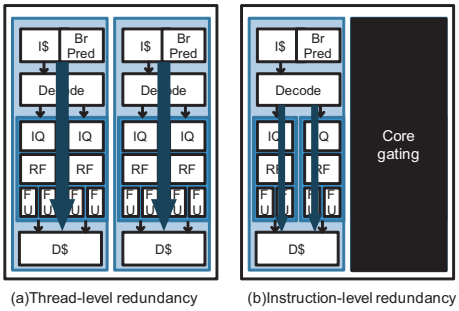

図 3 スレッド冗長実行と命令冗長実行

マルチ・クラスタ型コア・プロセッサの冗長実行には二つの実行方法がある．一つはスレッド冗長実行であり，複数のプロセッサコアで冗長実行する．図 3(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，ほとんどのエラーを検出することが可能である．このスレッド冗長実行は信頼性が特に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り，複数のクラスタで冗長実行する．図 3(b) にスレッド冗長実行の動作を示す．図のように，クラスタで冗長実行されるが，キャッシュ，デコードの部分は冗長実行されない．命令冗長実行はプロセッサコア一台のみ使用するので，スレッド冗長実行より消費電力が少ない．しかしクラスタで冗長実行するので，信頼性も保障される．命令冗長実行は，信頼性と省電力が要求されるときに選択される．

マルチ・クラスタ型コア・プロセッサでは，信頼性の要求度に応じてスレッド冗長実行と命令冗長実行を切替えて実行する．

2.3 ばらつき

遅延の統計的性質より，クリティカルパス数が増加すると回路遅延が増加し，ばらつきは小さくなる[1]．プロセッサの面積が等しいとき，マルチコアプロセッサはシングルコアプロセッサよりもクリティカルパス遅延は小さくなる．また，マルチコアプロセッサはシングルコアプロセッサよりも同じ遅延のパス数が増加する．

一つの大きなプロセッサを分割し，クラスタ化することで，最長パス遅延は小さくなり，その結果同じ遅延のパス数が増加する．マルチ・クラスタ型コア・プロセッサでは，回路遅延の増加を抑え，ばらつきを小さくすることが可能である．

3. 信頼性の定式化

初めに語句の定義をする．フォールトとはソフトエラーによって起こるビット反転であり，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間違いがあることである．

従来は，信頼性を示す指標として平均故障時間 MTTF(Mean Time To Failure) が広く使用されている．しかし，MTTF ではプロセッサの性能とプロセッサで発生するエラー率とのトレードオフを見ることができない．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(Mean Instruction To Failure)[3] がある．この MITF は平均故障命令数を表す．MITF の式を図 4 に示す．

$$
\begin{aligned}
MITF &= \frac{\text{実行命令数}}{Error\text{数}} \\
&= \frac{\text{実行命令数}}{\text{動作時間} \times Error\text{する確率 (件/時間)}} \\
&= \frac{\text{実行命令数}}{\frac{\text{実行サイクル数}}{\text{周波数}} \times Error\text{する確率}} \\
&= \frac{IPC \times \text{周波数}}{Error\text{する確率}} \\
&= \frac{IPC \times \text{周波数}}{SoftError\ \text{確率(件/時間)} \times AVF}
\end{aligned}
$$

図 4 MITF

ここで，AVF は Architectural Vulnerability Factor[3] であり，プロセッサ内部で発生したフォールトがプログラム実行に影響を与え，エラーとなる確率を示す．全てのフォールトがプログラムの実行結果に影響を及ぼすとは限らない．例えば，分岐予測ミスをする命令でフォールトが起こっても，プログラムの実行結果に影響を及ぼさない．

現在想定しているマルチ・クラスタ型コア・プロセッサの冗長実行モードの性能と信頼性のトレードオフを数量化し評価するために，この式を利用し冗長実行し

ない時，スレッド冗長実行時，命令冗長実行時の MITF を求める．

3.1 冗長実行なし時の MITF

冗長実行なしではプロセッサコア 1 台で従来通りに実行する．図 4 の MITF の式より冗長実行しない時の MITF の式を求める．その式を図 5 に示す．PC はプロセッサコア，N_b は総ビット数，E_b はビットあたりのソフトエラー率を表している．AVF_{PC} はプロセッサコアの AVF である．

$$MITF_{non} = \frac{\text{実行命令数}}{\text{Error数}} = \frac{\text{実行命令数}}{\text{実行時間}\times\text{Errorする確率}}$$

$$= \frac{IPC\times\text{周波数}}{PC\text{でError}\text{が起こる確率}}$$

$$= \frac{IPC\times\text{周波数}}{PC\text{でFault}\text{が起こる確率}\times AVF_{PC}}$$

$$= \frac{IPC\times\text{周波数}}{\text{ビットあたりFault}\text{する確率}\times N_{b_{PC}}\times AVF_{PC}}$$

$$= \frac{IPC\times\text{周波数}}{E_b\times N_{b_{PC}}\times AVF_{PC}}$$

図 5 冗長実行なし時の MITF

3.2 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．そのため，スレッド冗長実行時にエラーが起こる場合は，プロセッサコア 2 個で同じ間違った結果が出力されたときである．プロセッサコア 2 個でエラーが起こっても出力される結果が同じでなければフォールトは検出され，エラーは起こらない．エラーは，プロセッサコア 2 個で，同じ時間に同じビットでフォールトが起こり，エラーとなる時のみである．スレッド冗長実行時の MITF の式を図 6 示す．

$$MITF_{thread} = \frac{\text{実行命令数}}{\text{Error数}} = \frac{\text{実行命令数}}{\text{実行時間}\times\text{Errorする確率}}$$

$$= \frac{\text{実行命令数}}{\text{実行時間}\times PC2\text{個で同時に}\text{Error}\text{が起こる確率}}$$

$$= \frac{IPC\times\text{周波数}}{PC2\text{個で同時に}\text{Error}\text{が起こる確率}}$$

$$= \frac{IPC\times\text{周波数}}{(PC2\text{個で同じ時,場所で}\text{Fault}\text{が起こる確率})^2\times AVF_{pc}}$$

$$= \frac{IPC\times\text{周波数}}{Comb(N_{b_{PC}},1)\times\left(\frac{PC\text{で}\text{Fault}\text{が起こる確率}}{N_{b_{PC}}}\right)^2\times AVF_{pc}}$$

$$= \frac{IPC\times\text{周波数}}{E_b^2\times N_{b_{PC}}\times AVF_{pc}}$$

図 6 スレッド冗長実行時の MITF

3.3 命令冗長実行時の MITF

命令冗長実行では，クラスタ 2 個を用いた冗長実行で実現する．命令冗長実行時にエラーが起こる場合は，クラスタ 2 個で同じ間違った結果が出力された時，またはクラスタ以外でフォールトが起こった時である．スレッド冗長実行時と同じように，エラーはクラスタ 2 個で同じ時間に同じビットでフォールトが起こった時のみ考慮する．命令冗長実行ではクラスタ以外は冗長実行されないため，スレッド冗長実行時よりも MITF は小さくなる．命令冗長実行時の MITF の式を図 7 に示す．CL はクラスタを表している．

$$MITF_{instruction} = \frac{\text{実行命令数}}{\text{Error数}} = \frac{\text{実行命令数}}{\text{実行時間}\times\text{Errorする確率}} = \frac{IPC\times\text{周波数}}{\text{Error}\text{する確率}}$$

$$\text{Error}\text{する確率} = CL2\text{個で同時に}\text{Error}\text{が起きる確率}$$

$$+\text{その他で}\text{Error}\text{が起きる確率}$$

$$= (CL2\text{個の同じ時,場所で}\text{Fault}\text{が起こる確率})\times AVF_{CL}$$

$$+(\text{その他で}\text{Fault}\text{が起きる確率}\times AVF_{other})$$

$$= \left\{Comb(N_{b_{CL}},1)\times\left(\frac{CL\text{で}\text{Fault}\text{が起こる確率}}{N_{b_{CL}}}\right)^2\times AVF_{CL}\right\}$$

$$+(\text{その他で}\text{Fault}\text{が起きる確率}\times AVF_{other})$$

$$= (E_b^{\,2}\times N_{b_{CL}}\times AVF_{CL}) + (E_b\times N_{b_{other}}\times AVF_{other})$$

つまり

$$MITF_{instruction} = \frac{IPC\times\text{周波数}}{(E_b^{\,2}\times N_{b_{CL}}\times AVF_{CL}) + (\text{その他で}\text{Fault}\text{が起きる確率}\times AVF_{other})}$$

図 7 命令冗長実行時の MITF

4. 性能と信頼性のトレードオフ

性能と信頼性のトレードオフ評価には，上述した評価指標 MITF を用いる．(冗長実行なしの MITF)：(スレッド冗長実行の MITF)：(命令冗長実行の MITF) は図 8 の式で表される．

$$\frac{IPC_{PC}\times f}{E_b\times N_{b_{PC}}\times AVF_{PC}} : \frac{IPC_{PC}\times f}{E_b^{\,2}\times N_{b_{PC}}\times AVF_{PC}} : \frac{IPC_{CL}\times f}{E_b^{\,2}\times N_{b_{CL}}\times AVF_{CL} + E_b\times N_{b_{other}}\times AVF_{other}}$$

図 8 MITF の比

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．E_b については，文献 4) よりビットあたりのソフトエラー率を 0.01FIT とする．また，キャッシュと TLB が ECC(Error Correcting Code) によってエラーから守られている場合とそうでない場合を考慮する．ECC で守られている場合，キャッシュのエラー率は 10^4 倍に減少する[5]．総ビット数 N_b はマルチ・クラスタ型コア・プロセッサの面積[2] で近似を行う．表 1 に回路規模を示す．表はマルチ・クラスタ型コア・プロセッサのプロセッサコアの構成要素の面積を表している．表中の I$，D$，TLB は ECC によりエラーから守られている要素である．表中の OoOexe,RFs,Func units はクラスタの構成要素であり，表はクラスタ二個分の面積を表している．この表より，プロセッサコアとクラスタとその他の総ビッ

ト数 N_b の比を求めた．ECC なしでは (プロセッサコア):(クラスタ):(その他) = 1:0.18:0.62，ECC ありでは (プロセッサコア):(クラスタ):(その他) = 1:0.26:0.47 となった．

AVF は全ての場合において同じと仮定した．

表 1 プロセッサコアの回路規模[2)]

D cache	13.0
I cache	10.4
TLB	4.4
Fetch, BrPred	4.5
Decode	1.7
OoOexe	20.2
RFs	5.8
Func units	13.0
Misc	2.4
Routing	26.4
Total	101.8 (mm^2)

続いて IPC の見積り方法を述べる．

4.1 IPC の評価

シミュレーションにはプロセッサの動作を模擬したシミュレータである M5[6)] を使用した．命令セットは Alpha である．クラスタ，プロセッサコア，プロセッサ全体の構成を表 2 に示す．表の左はプロセッサの要素を表し，右はクラスタ，プロセッサコアのエントリ数または演算器の個数を表している．今回の評価では，冗長実行時の比較にかかる時間は考慮していない．冗長実行なしとスレッド冗長実行はプロセッサコア 1 台での IPC を，命令冗長実行はクラスタ 1 つでの IPC を適用する．

表 2 プロセッサ構成

	CL , PC , Processor
IQ	16 , 32 , 64 entry
Int units	2 , 4 , 8
Fp units	2 , 4 , 8
Ld/St	1 , 2 , 4

ベンチマークプログラムには SPEC2000 より 10 種類を用い，先頭から 5 億命令をシミュレーションした．シミュレーションの結果を図 9 に示す．縦軸は IPC を，横軸は各ベンチマークを表している．評価の結果，クラスタが 1 つの時の IPC は平均で 1.21，プロセッサコア 1 台（クラスタ 2 つ）の IPC は平均で 1.98 であった．

4.2 MITF の見積り

上記の仮定と IPC の評価結果を図 8 の式に代入し，MITF を求める．メモリ機構 (キャッシュ，TLB) が ECC によってエラーから守られている場合と守られていない場合の，冗長実行なし，スレッド冗長実行，命令冗長実行の MITF を求める．

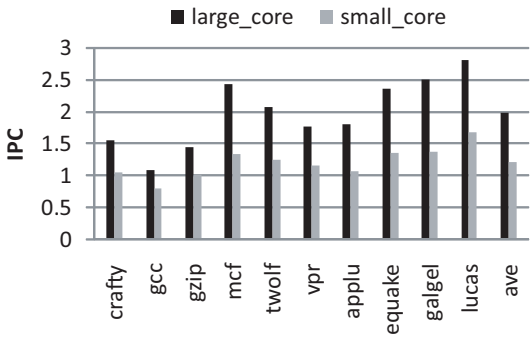

図 9 IPC の評価結果

4.2.1 ECC なしの場合

ECC なしのマルチ・クラスタ型コア・プロセッサの MITF を表 3 に示す．全ての値は，冗長実行なしの MITF の値により正規化されている．表の行の non は冗長実行なしの従来通りの実行を，thread はスレッド冗長実行を，instruction は命令冗長実行を表している．

命令冗長実行では冗長実行なしに比べ MITF は平均 1%の向上に留まった．また，mcf や galgel などのベンチマークでは，MITF は冗長実行しない時より低下している．原因はクラスター台での IPC が，計算資源の欠乏により大幅に低下するためである．

4.2.2 ECC ありの場合

ECC ありのマルチ・クラスタ型コア・プロセッサの MITF を表 3 に示す．本節の頭で述べたように ECC を用いる時のエラー率は非常に小さいので，キャッシュ，TLB でエラーは起こらないと仮定した．ECC ありの場合ではスレッド冗長実行で 100%，命令冗長実行で 31% 向上した．ECC ありではクラスタが冗長実行で，メモリが ECC で保護されており，エラーに脆弱な部分が ECC なしに比べ減少する．そのため命令冗長実行では，MITF が 31%に向上している．

5. まとめと今後の課題

本稿では，性能・電力・信頼性・ばらつきに考慮したアーキテクチャであるマルチ・クラスタ型コア・プロセッサの信頼性を検討した．見積りの結果，冗長実行なしと比べて，ECC なしの場合に MITF が，スレッド冗長実行で 100 倍，命令冗長実行で 1%向上した．ECC ありの場合で MITF が，スレッド冗長実行で 100 倍，命令冗長実行で 31%向上した．

今後の課題として，詳細な評価を行う必要がある．まず，AVF の見積りである．スレッド冗長実行，命令冗長実行の MITF の式からわかるように，AVF は信頼性指標である MITF に大きな影響を与える．AVF を見積ることで MITF のより正確な評価をおこなうことができる．

また本稿では，冗長実行後の比較にかかる時間を考

表 3　ECC なしの MITF

	crafty	gcc	gzip	mcf	twolf	vpr	applu	equake	galgel	lucas	ave
non	1	1	1	1	1	1	1	1	1	1	1
thread	100	100	100	100	100	100	100	100	100	100	100
instruction	1.08	1.19	1.15	0.89	0.97	1.07	0.96	0.93	0.88	0.96	1.01

表 4　ECC ありの MITF

	crafty	gcc	gzip	mcf	twolf	vpr	applu	equake	galgel	lucas	ave
non	1	1	1	1	1	1	1	1	1	1	1
thread	100	100	100	100	100	100	100	100	100	100	100
instruction	1.40	1.55	1.49	1.15	1.26	1.39	1.24	1.20	1.15	1.25	1.31

慮していないため，実行結果の比較にかかる時間を考慮した評価をする必要がある．

今回は性能と信頼性のトレードオフの評価を行った．この他に，消費電力と信頼性のトレードオフの検討，ばらつきの検討をする必要がある．

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) 小野寺，ばらつきを克服する設計技術，回路とシステム軽井沢ワークショップ，2006.
2) T. Sato and A. Chiyonobu, Multiple Clustered Core Processors, 13th Workshop on Synthesis and System Integration of Mixed Information Technologies, 2006.
3) 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. 43, No. 6, 2004.
4) E. Normand, Single Event Upset at Ground Level, IEEE Transaction on Nuclear Science, Vol. 24, No. 6, 1996.
5) M. Sugihara, T. Ishihara, K. Hashimoto, and M. Muroyama, A Simulation-Based Soft Error Estimation Methodology for Computer System, 7th International Symposium on Quality Electronic Design, 2006
6) N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, The M5 Simulator: Modeling Networked Systems, IEEE Micro, Vol. 26, No. 4, 2006.