A-05

サイクルレベルの空間及び時間冗長化技術を融合させた高信頼プロセッサの提案

A Reliable Processor Based on Cycle Level Fusion of Space and Time Redundancy Techniques

渡邊　良二 †1　　姚　駿 †1　　中田　尚 †1
Ryoji WATANABE　Jun YAO　Takashi NAKADA
嶋田　創 †1　　中島　康彦 †1
Hajime SHIMADA　Yasuhiko NAKASHIMA

1. はじめに

近年，半導体製造技術の微細化が進むにつれ，ビット反転に必要な最小電荷量が減少し，内外ノイズの影響によって発生したビット反転により，回路が一時的に誤動作を起こす一時故障の発生増加が懸念されている[1)]．また，半導体の微細化による製造ばらつきの増加で正確な加工が困難となり，同時にチップ内配線も微細化するため断線が発生しやすくなり，永久故障の発生増加も深刻な問題となる[2)]．そこで，故障の増加という問題を解決しシステム性能の継続的な改良を進めていくために，空間冗長化や時間冗長化などの冗長化技術[3)]を用いて，故障を許容するような設計を採り入れたプロセッサが多数提案されている[4)–7)]．

空間冗長化の中で幅広く使われている技術に Dual Modular Redundancy（DMR），Triple Modular Redundancy（TMR）がある．DMR では 2 つの同一の回路からの出力を比較することにより故障を検出する．一時故障であれば，正常な状態からの再実行によって，正常な動作を続けることができる．また，TMR では 3 つの同一の回路からの出力の多数決を採ることにより，故障の発生を隠蔽して実行を続けることができる．この構成は，一時故障のみならず，永久故障の発生時にも正常な動作を続けることができる点でも優れている．これらの構成の欠点として，回路面積や実行時の消費電力が 2 倍，3 倍になるという点がある．

この問題を緩和するために，我々は文献 4) においてパイプライン・レベルでの多重化度を動的に変更可能にすることにより，信頼性を確保するために必要な電力コストを削減することのできる Dynamic Adaptive Redundant Architecture（DARA）を提案した．また，我々は文献 8) において，DARA のパイプライン構成を元に，本来多重化に用いられる 2 本のパイプラインを協調動作させることによって 1 つのスーパスカラ・プロセッサとして動作させ，処理性能を向上する方法を示した．

本稿ではこれまでの研究を発展させ，DARA の空間冗長化技術に時間冗長化技術を融合させ，DMR と同等の信頼性を持ち，文献 8) のスーパスカラ構成に近い性能を発揮する新しい構成を提案する．そして，この構成が面積性能比に優れることを，回路合成結果とソフトウェア・シミュレータによるプロセッサ性能評価結果によって示す．以下では，2 章において先行研究について紹介し，3 章において提案構成について紹介する．そして，4 章で提案構成について評価結果を示し，5 章でまとめと今後の課題を述べる．

†1 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

2. 先行研究

これまでに，時間冗長や空間冗長を応用したプロセッサが多数提案されている．AR-SMT[5)] では時間冗長の技術を用い，単一の SMT（Simultaneous Multi-Threading）プロセッサによって，スレッド・レベルでの時間冗長実行を行い故障検出/訂正を可能にしている．また，空間冗長化を用いたプロセッサとして，IBM の Z990[6)] では複製されたフェッチ・ユニット，デコーダ，実行ユニットで同一の処理を実行し，実行結果をチェックポイントアレイもしくはキャッシュにコミットする際に比較することによりエラー検出を行う．CRTR[7)] では CMP（Chip Multi-Processor）内の異なるプロセッサを用いて，先行スレッド，追従スレッドと呼ばれる同一スレッドを実行し，結果を比較することにより故障の検出を行う手法が提案されている．また，我々は文献 4) において，必要とされる信頼性に応じて多重化に用いる回路数を動的に変更することによって，信頼性を確保するために必要な平均の電力量を削減する DARA を提案しており，次節で本研究のベースとなる DARA の詳細を述べる．

2.1 DARA による高信頼実行

DARA は空間冗長化の代表的な手法である DMR，TMR を応用したプロセッサである．**図 1** に DARA の概略図を示す．DARA は，命令フェッチ（IF），命令デコード（ID），レジスタ値読み込み（RR），実行（EX），メモリアクセス（MA），レジスタ値書き戻し（WB）の 6 ステージのパイプラインが複数集まって構成される．各パイプラインは単純なスカラ・パイプラインであり，複数のパイプラインで同一の処理を実行し，各々のパイプライン・レジスタの値を比較することによってステージ・レベルでの故障検出を行う．従って，各パイプライン・ステージには故障検出を行うために，図 1 に示すような，自身と他のパイプラインが保持するパイプライン・レジスタの値を比較するための比較器と，自身のパイプライン・レジスタの値を他のパイプラインに渡すための通信路を有する．

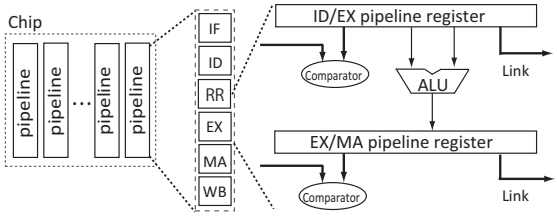

図 1 DARA の概略図

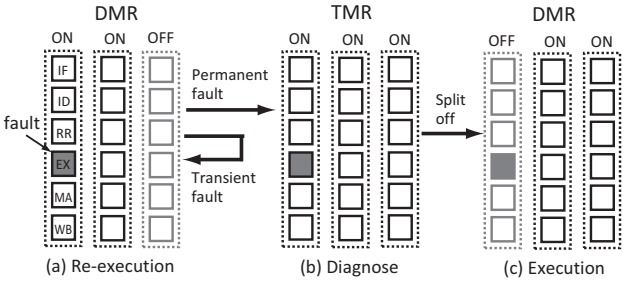

図 2 DARA における故障検出と通常実行への回復

DARA の動作モデルを**図 2** に示す．DARA は基本的な動作モードとして DMR 動作を採用している．図 1 に示したパイプライン同士を 2 つ接続し，同一の処理を実行する．故障の検出はパイプライン・ステージごとにパイプライン・レジスタの値を比較することにより，故障の発生した個所と命令を特定する．DMR 動作中に故障が検出された場合，初期の対応では故障が検出された命令の再実行を行う（図 2（a））．検出された故障が一時故障である場合，誤動作を起こした命令の再実行により訂正できる．しかし，繰返し同一の命令の再実行を行っても故障を訂正できない場合には，その故障は永久故障であると判断し動作モードを永久故障箇所特定モードに変更する．これは，基本の動作モードである DMR 構成に，3 本目のパイプラインを接続し TMR 構成とするものである（図 2（b））． TMR 構成において永久故障個所を特定した後に，故障パイプラインを切り離し，再び DMR 構成で命令の実行を継続する（図 2（c））．すなわち，DARA において TMR 構成は，永久故障が発生した個所を特定する場合にのみ用いられる．従って，DARA では一時/永久故障の両方に対応可能であり，さらに，つねに TMR 構成とする手法に比べ，平均の消費電力を抑えることが可能となる．

2.2 DARA による 2-way インオーダ・スーパスカラ実行

我々は文献 8) において，DARA の 2 本のパイプラインを用いて，1 つの 2-way インオーダ実行スーパスカラ・プロセッサを構成する方法を提案した．DARA においてあらかじめ用意されているパイプラインを 2 本用いて，故障検出のために用意されたパイプライン・レジスタの値を他のパイプラインに渡すためのネットワークで接続し，協調動作をさせる．スーパスカラ実行を行うためには，新たに次命令フェッチ機構や，レジスタ・ポート数の追加，その他に，結果フォワーディングや協調制御機構が必要である．**図 3** に 2-way のインオーダ・スーパスカラ動作の実現のために必要な構成の変更例として，追加された結果フォワーディング・ユニットを示す．ここで，重要なことはあらかじめ用意されているパイプライン・レジスタの値を渡すためのネットワークは，他のパイプラインからの結果フォワーディングに用いることが可能であり，その他にも多くの既存パイプライン資源が利用可能であるため，ハードウェアの追加量を抑えることができるという点である．

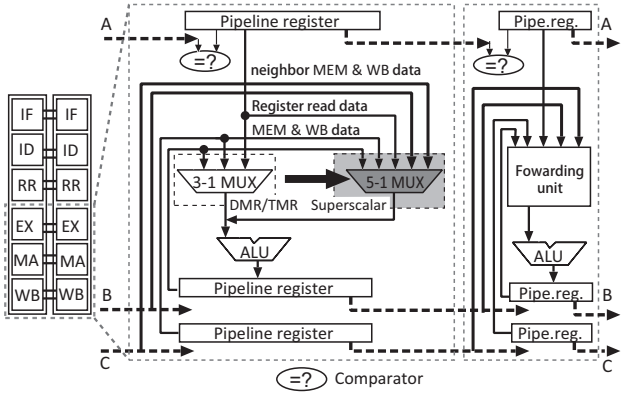

図 3 DARA を用いた 2-way インオーダのスーパスカラ・プロセッサ

3. 空間/時間冗長を融合したプロセッサの提案

DARA では，故障状況に応じた適応的な多重化を行うことにより一時/永久故障の両方に対応可能であった．また，DARA には豊富なパイプライン資源に加え，各々のパイプライン同士がネットワークを通じてパイプライン・レジスタの値を伝搬できるという特徴がある．そこで，本章では上記のような DARA のパイプライン構成を利用し，信頼性を維持しつつ，コスト対性能比を向上させる新しい構成の提案を行う．

3.1 提案構成の概要

本節では，DARA のパイプラインを 3 本接続した構成をもとに，信頼性を維持しながらも，プロセッサの処理性能を向上する構成を提案する．

DARA において，あらかじめ用意されているパイプラインのうち 3 本を協調動作させることにより，最大で 3 命令を同時に実行できる．そこで，本提案では 3 本のパイプラインを用いて，1 クロック・サイクル中にデータ依存のない 2 命令の同時実行を行う．2 命令のうち 1 命令は複製し，2 本のパイプラインで実行することにより空間冗長実行を適用し，残りの 1 命令は次サイクルとの間で時間冗長実行を行う．各命令はフェッチ時にその命令を DMR 動作による空間冗長実行を行うか，1 クロック・サイクル分の時間冗長実行を行うかが決まる．以降，パイプラインが空の状態からの実行を例示する．3 本のパイプラインにおいて，最初の命令は複製し 2 本のパイプラインを用いて空間冗長実行する．このとき，2 番目の命令は残り 1 本のパイプラインが利用可能であるため時間冗長実行の 1 度目の実行を行う．そして，次のサイクル

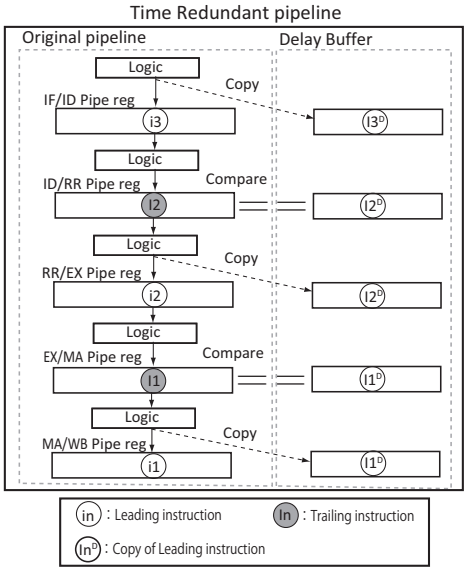

図 4 DARA パイプラインを用いた時間冗長実行の構成

において 1 本のパイプラインを用いて 2 番目の命令を再度実行することにより，1 クロック・サイクル分の時間冗長実行を行い故障検出を行う．3 番目の命令は最初の命令と同様に，残りの 2 本のパイプラインを利用することにより空間冗長実行を行う．上記のように，空間冗長実行と時間冗長実行を融合することにより，3 本のパイプライン資源を最大限利用し，信頼性を維持しながら性能を向上する構成を提案する．

本節では，まず DARA のパイプラインでの時間冗長実行の実現について説明し，続いて 3 本のパイプラインを協調動作させることによって，信頼性を維持し性能を改善する提案構成の詳細とその動作例について示す．

3.1.1 サイクルレベル時間冗長実行の追加

本項では，DARA のパイプラインにおいてサイクル・レベルの時間冗長実行を実現する方法について述べる．**図 4** に DARA のパイプラインをもとに構成した**時間冗長実行パイプライン**を示す．時間冗長実行実現のため，もとのパイプライン構成に対してディレイ・バッファ（Delay Buffer）が追加されている．時間冗長実行パイプラインでは，フェッチされた命令と同一の命令を，次のサイクルにもう一度フェッチし 1 サイクル分だけ遅れた時間冗長実行を行う．最初に実行する命令を**先行命令（Leading instruction）**，続いて実行する冗長命令を**追従命令（Trailing instruction）**と呼ぶ．時間冗長パイプラインでは，各ステージで実行された先行命令の結果をディレイ・バッファに蓄える．そして，次のサイクルに実行される追従命令の結果と，ディレイ・バッファ内に蓄えられている先行命令の結果を比較することによって故障検出を行う．なお，パイプライン・レジスタとディレイ・バッファの内容の比較は，通常の DARA の動作と同様に次のステージで行われる．これは，各ステージのクロック・サイクル時間を増加させないためである．

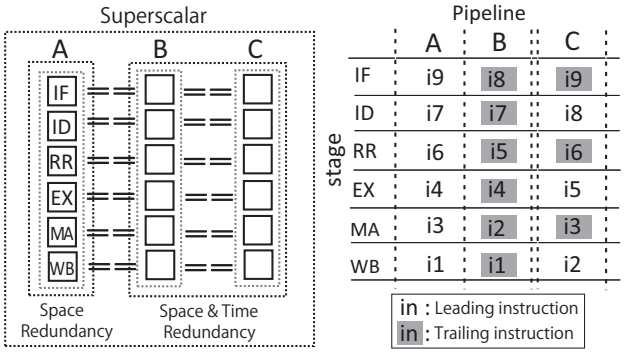

図 5 提案構成の構成図と命令の実行モデル

3.1.2 3 本のパイプラインを利用した空間及び時間冗長を融合したスーパスカラの構成

3.1.1 項で提案したパイプラインの時間冗長実行，文献 8) で提案した 2 本のパイプラインを用いたスーパスカラ実行のアイデアを用いて，3 本のパイプラインを用いた信頼性/性能を両立するアーキテクチャを構成する．

図 5（a）に提案構成の構成図を，図 5（b）に各パイプライン・ステージでの命令の実行モデルを示す．図 5（a）に示すように，パイプライン A は空間冗長実行を行う命令のみで用いられ，パイプライン B，C は空間冗長実行と時間冗長実行を行う命令の両方で用いられる．また，図 5（b）の中で空間冗長実行されている命令は i1，i3，i4，i6，i7，i9 であり，時間冗長実行されている命令は i2，i5，i8 である．図 5（b）に示すように，命令 i2，i5，i8 は異なるパイプラインを用いて時間冗長実行されている．このように，異なるパイプラインで時間冗長実行を行う理由は，3.1.1 項で示したような，同一のハードウェアで 1 クロック・サイクル分のみ間隔をあけて時間冗長実行を行った場合，組み合わせ回路やラッチの一時的なビット反転が複数サイクルにわたって生じる場合に対応できないためである．複数サイクルにおよぶビット反転は半導体製造技術の微細化に伴って増加すると考えられており，永久故障発生時も同様に時間冗長実行では対応できない．すなわち，このような複数サイクルに及ぶ一時故障や，永久故障を検出できない状況を回避するために時間冗長実行における先行命令，追従命令を異なるパイプラインで実行する構成を採用する．

こうして構成された図 5（b）のような実行モデルにより，命令ごとに空間冗長，時間冗長どちらかの冗長実行を適用することによって，命令の実行における誤動作を検出し，信頼性を保証する．また，本構成では図 5（b）に示すような動作を行うことによって，最大で 2 クロック・サイクル（CC）あたり 3 命令実行可能（最大 IPC1.5）なスーパスカラ・プロセッサの構成にもなっており，処理性能も改善することができる．

3.2 提案構成の動作モデル

本節では，3.1.2 項で構成した提案構成が，実際にどのよう

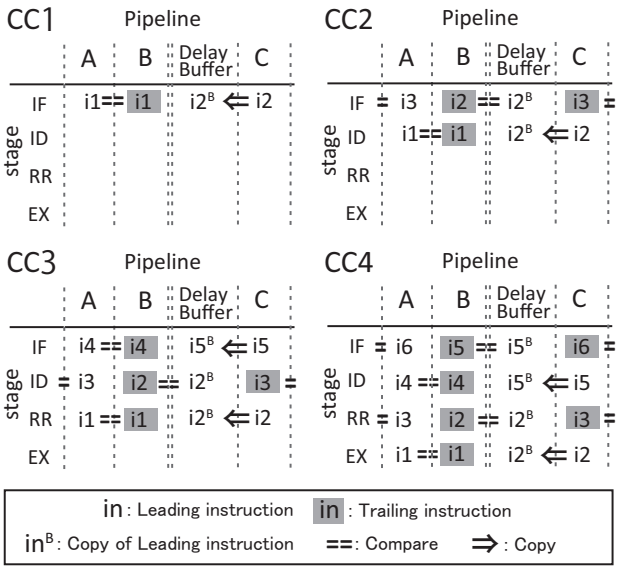

図 6 提案プロセッサ内の実行命令の流れ

に命令を実行しているのかについて説明する．続いて，時間冗長実行による性能低下を最小限に抑えるためのデータ依存や分岐命令の取り扱い方について説明を行う．

3.2.1 通常動作

提案構成における命令の実行モデルについて説明する．**図 6** に提案構成のクロック・サイクル（CC）ごとの命令の実行状況を示す．まず，図 6 の CC1 においてパイプライン A，B では同一の命令 i1 がフェッチされ，パイプライン C では次の命令 i2 がフェッチされる．このとき，パイプライン C でフェッチされた先行命令 i2 の結果はディレイ・バッファに蓄えられる．次に，CC2 においてパイプライン A，C では次の命令 i3 がフェッチされるが，パイプライン B では再度 i2 が追従命令としてフェッチされる．このとき，ディレイ・バッファに蓄えられていた先行命令 $i2^B$ とフェッチしてきた追従命令 i2 を比較することによって，フェッチした命令の正しさを確認する．CC3 においても同様に，CC2 のパイプライン C の ID ステージでディレイ・バッファに保存された i2 の先行命令の結果と，CC3 のパイプライン B の ID ステージで実行された i2 の追従命令の結果を比較することにより ID ステージの故障を検出することができる．以降のステージでも同様の処理により故障を検出することが可能である．故障を検出した場合，故障が検出された命令と，後続命令をすべてフラッシュし，命令を再実行する．

3.2.2 命令間に依存関係がある場合

AR-SMT や CRTR では，時間冗長実行する命令において，最初に実行した命令の結果の正しさを確認する前にその結果を投機的に用いることにより，命令間に存在する依存関係等を解消し，性能低下を最小限に抑える手法が採られている．そこで，本提案構成においても同様に命令間に依存関係がある場合に時間冗長パイプラインで実行された先行命令の結果を，正確性が確認される前に投機的に後続の命令で用いることによって，性能低下を最小限に抑える手法を用いる．

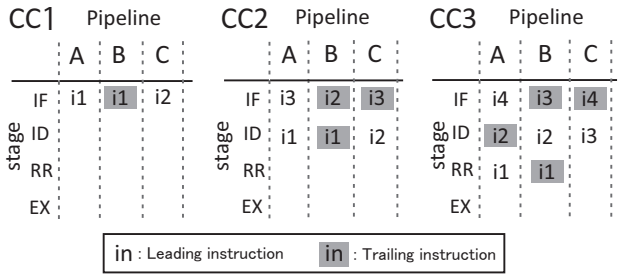

図 7 命令間に依存関係がある場合の動作

まず，図 6 の CC4 において時間冗長実行している i2 と，空間冗長実行している i3 の間にデータ依存がある場合を考える．i2 の実行結果の正確性を確認後に i3 が i2 の結果を用いるためには，まず CC4 の次のサイクル CC5 において EX ステージで実行される i2 の追従命令とディレイ・バッファ内の i2 の先行命令の結果を比較し，命令 i2 の演算結果が正しいことを確認する．その後，CC6 において i3 に正しいことが確認された i2 の結果をフォワーディングすれば，i3 に使われるフォワーディング・データは正しいと言える．しかし，i3 は CC5 で実行される i2 の追従命令の演算が完了した次のサイクル CC6 で実行されるため，i2 の時間冗長実行によって 1 サイクル分性能は低下する．そこで，CC4 において演算された i2 の先行命令の結果を i3 にフォワーディングし，CC5 でそのデータを用いて i3 の演算を行えば 1 サイクル分の性能低下を防止することができる．この場合，i2 の先行命令からのフォワーディング・データは正しさが確認されていない．しかし，i2 の先行命令の演算結果が間違っていた場合には CC5 で実行される i2 の追従命令との結果比較により誤動作は検出できる．従って，i2 及びその後続命令はすべてフラッシュされ，再び i2 のフェッチから行われるため，投機的にデータをフォワーディングすることによる信頼性への影響はなく，性能低下も防止できる．

次に，**図 7** において空間冗長実行している i1 と，時間冗長実行している i2 の間にデータ依存がある場合を考える．この場合，i1 は空間冗長実行されているため，結果を投機的にフォワーディングすることができず，i2 は必ずストールしなければならない．そこで，このような場合には i1 と i2 の間の依存関係を CC2 の ID ステージにおいて，デコードした結果から検出し，図 7 の CC3 に示すように i2 を時間冗長実行から空間冗長実行に切り替える．CC2 から CC3 にかけて先行命令 i2 をパイプライン C からパイプライン B に移すことは，パイプライン・レジスタの値を渡すための通信路を用いることにより容易に実現でき，他の命令も同様の方法でパイプライン間を移動することができる．

3.2.3 分岐命令

時間冗長実行している命令が分岐命令の場合は 3.2.2 項で述べた，命令間に依存関係がある場合と同様に投機的な実行を行うことにより性能低下を防ぐことができる．図 6 の CC4

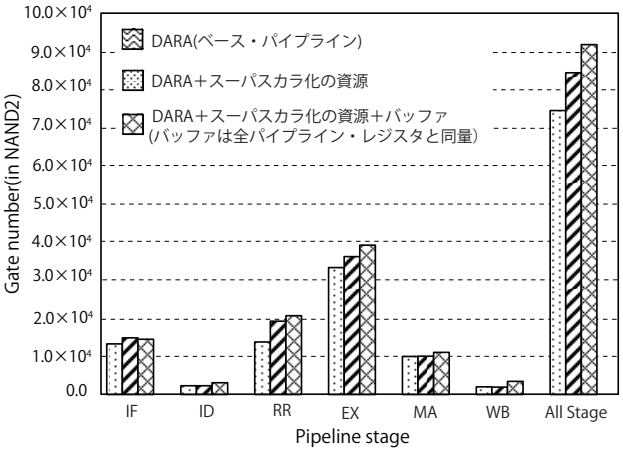

図 8 各パイプライン構成におけるステージごとの面積評価

において i2 が分岐命令である場合，CC4 で実行される i2 の先行命令の分岐先アドレスの計算結果から投機的に次の命令のフェッチを開始することによって 1CC 分の性能低下を防止できる．投機的に分岐した後に，CC5 で実行される i2 の追従命令によって計算された分岐先アドレスと，CC4 でディレイ・バッファに蓄えられた i2 の先行命令の分岐先アドレスを比較することにより誤動作は検出できる．すなわち，データ依存の場合と同様に，結果が間違っていれば後続の命令をすべてフラッシュし，再実行を行うことによって投機的実行による信頼性への影響はない．

4. 評　　価

3 章で提案した構成について面積及び性能について評価を行い，提案構成の有効性について検証する．

4.1 面積評価

本節では 3 章において提案した構成について面積評価を行う．提案構成の基本となる DARA の各パイプラインは，命令長 16-bit の RISC 命令セットをもとに 6 ステージで設計されている．命令/データ・キャッシュは共用されており，今回使用したキャッシュ・サイズは 4Kbyte と設定した．上記のように設計された DARA のパイプラインをもとに，文献 8) で提案した DARA の 2 本のパイプラインをもとにした 2-way インオーダ・スーパスカラ構成と，3.1.2 項で構成を提案した信頼性/性能を両立する提案構成について面積評価を行った．RTL 実装には Verilog-HDL を用い，Synopsys Design Compiler において Rohm 0.18μm cell library を用いて論理合成を行った．

図 8 に各パイプライン構成において，パイプライン・ステージ単位で論理合成を行った結果を示す．また，**表 1** に各々の構成における面積を，DARA のベース・パイプラインを基準として比較した結果を示す．図 8 の結果から DARA のベース・パイプラインから 2-way のインオーダ・スーパスカラにするためには，新たに次命令フェッチや結果フォワーディングなどのハードウェアが必要なため IF，RR，EX ステージ

表 1 DARA をベースとしたの面積比率評価

Pipeline 構成	評価値		
	ゲート数 (NAND2)	比率	バッファ追加による増加率 (%)
DARA	74583	100.0	–
＋ Superscalar 化の資源	84455	113.2	–
＋ Superscalar 化の資源 ＋全パイプライン・ レジスタ分のバッファ	91908	123.2	10.0

で面積の増加が見られ，パイプライン全体として 13.2%の面積増加となった．特に RR ステージでは，スーパスカラによる 2 命令の同時実行を実現するために必要なレジスタ・ポート数の追加が影響し，39.6%の面積増加が発生した．

続いて，3.1.1 項で述べた，時間冗長実行を実現する構成を検討する．表 1 よりすべてのパイプライン・レジスタと同一のサイズのディレイ・バッファを用意し，時間冗長実行を実現するには，ディレイ・バッファの追加によって 10.0%分の面積増加が必要である．すなわち，提案構成を実現するためには DARA のベース・パイプラインに対し，ディレイ・バッファに加えスーパスカラ化の資源も必要であるため，合計で 23.2%の面積増加が必要であることが分かった．

4.2 性能評価

本節では，3.1 節において提案した，2 サイクルで 3 命令実行可能な提案構成が，DARA のベース・パイプライン構成や通常の 2-way インオーダ・スーパスカラ構成に対してどの程度の性能向上を達成できるのかを検討する．性能評価では，上記の各構成に対するサイクル・アキュレートのソフトウェア・シミュレータと Stanford ベンチマークを用いて評価した．

図 9 に各構成においてベンチマークを実行した結果を示す．図 9 から，各構成での平均の IPC を見ると，2-way インオーダ・スーパスカラ構成では DARA のベース・パイプライン構成に対し 29.4%性能が向上しており，提案構成では 21.3%性能が向上した．すなわち，この結果から 2 サイクルで 3 命令実行できる提案構成では，2-way インオーダ・スーパスカラ構成での性能改善率に対して，73.3%分の性能改善を達成できていることが分かった．さらに，図 9 において，各ベンチマークにおける 2-way インオーダ・スーパスカラ構成の性能と提案構成の性能について見ると，Intmm，Trees では提案構成と 2-way インオーダ・スーパスカラ構成においての性能差は殆どないことが分かる．しかし，Bubble，Towers，Perm においては比較的大きな性能差が見られる．この原因は，DARA のベース・パイプライン構成での性能と 2-way インオーダ・スーパスカラ構成にした場合の性能の改善度に着目すると，Bubble，Towers，Perm においては性能が大幅に改善されており，プログラム中の命令レベル並列度が高く 2 命令同時実行できるパターンが多く存在していることが分かる．しかしながら，提案構成においては 2 サイクルで 3 命令実行という制約があるために，2-way インオーダ・スーパスカラ構成では 2 命令同時実行できる状況においても，提案構成では 2 命

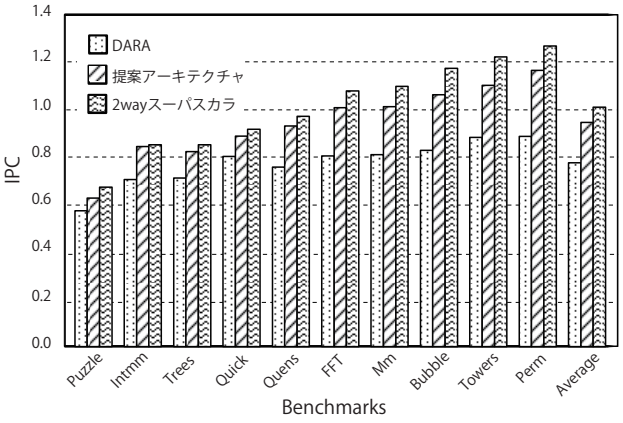

図 9 提案構成の性能評価

表 2 提案構成の面積性能比評価

Pipeline 構成	評価値		
	ゲート数 (NAND2)	平均 IPC	面積性能比 (正規化値)
提案構成	275724	0.947	0.660
DMR-スーパスカラ	337820	1.011	0.575

令を同時に実行できない場合がある．そのため，2-way インオーダ・スーパスカラ構成で同時実行できる命令の割合が増えるにしたがって，提案構成では同時実行できない状況が増え，性能に差が生じたと考えられる．

4.3 提案構成の面積性能評価

DARA のパイプライン構成を利用して信頼性を維持し，性能を改善する単純なアイデアとして，3.1 節で提案した構成の他に 4 本のパイプラインを用いて性能を改善する構成がある．まず，4 本のパイプラインのうち 2 本ずつをペアとして 2 本の DMR 構成のパイプラインを作る．そして，2 本の DMR パイプラインを文献 8) で提案した方法により 2-way インオーダ・スーパスカラ化することにより，すべての命令で空間冗長実行を行えるため信頼性を低下することなく，さらに性能も改善できる．

本節では，4 本のパイプラインを利用した構成を DMR-スーパスカラ構成と呼び，3.1 節で提案した提案構成と共に面積性能比を評価した．3 本のパイプラインを利用した提案構成では，3 本すべてのパイプラインにスーパスカラ実行と時間冗長実行の機能を持たせた構成を提案構成として評価を行った．

表 2 に各構成での面積性能比の評価結果を示す．表 2 における面積性能比は，DARA の基本構成である DMR 構成の性能，面積で正規化した値となっている．この結果から各構成における面積性能比は提案構成で 0.660，DMR-スーパスカラ構成では 0.575 と，提案構成は 4 本のパイプラインを用いて単純な空間冗長構成とする構成に比べ面積性能比が 14.9%良いことが分かった．

5. ま と め

本稿では，冗長化度変更可能なアーキテクチャである DARA のパイプライン構成をもとに拡張を行い，3 本のパイプラインを利用して信頼性を維持しながら，性能を改善できる新しい構成の提案を行った．面積評価において DARA のベース・パイプラインに対し，時間冗長実行とスーパスカラの機能を付け加えるためには，パイプラインの面積が 23.2%増加することを確認した．また，面積性能比の評価において，提案構成は単純に空間冗長化によって信頼性を確保し，性能を改善する DMR-スーパスカラ構成に比べ面積性能比を 14.9%改善することができた．

今後の研究において，複数パイプラインを協調動作させた場合のクリティカル・パスの増加や，時間冗長実行のための最適なバッファ配置などの検討を行っていく予定である．

謝　　辞

本研究の一部は科学技術振興機構・戦略的創造研究推進事業の協力により行われたものである．また，本研究は東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社，日本ケイデンス株式会社，ローム株式会社の協力で行われたものである．

参 考 文 献

1) P. Shivakumar, et al. “Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic”, In Proceedings of the 2002 International Conference on Dependable Systems and Networks, pp. 389-398, 2002.
2) J. Srinivasan, et al. “The Impact of Technology Scaling on Lifetime Reliability”, In Proceedings of the 2004 International Conference on Dependable Systems and Networks, pp. 177-186, 2004.
3) S. Mitra, et al. “Robust System Design with Built-In Soft Error Resilience”, IEEE Computer, vol. 38, no. 2, pp. 43-52, 2005.
4) Jun Yao, et al. “A Scalable Pipeline Design for Modularizing High Dependable Framework via Spatial Redundancy”, In Proceedings of the DA symposium 2008, pp. 169-174, 2008.
5) F. Rotenberg, et al. “AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors”, In Proceedings of the 29th Annual International Symposium on Fault-Tolerant Computing, pp.84-91, 1999.
6) P. Meaney, et al. “IBM z990 soft error detection and recovery”, In Proceedings of the IEEE Transactions on Device and Materials Reliability, vol. 5, no. 3, pp. 419-427, 2005.
7) M. Gomaa, et al. “Transient-Fault Recovery for Chip Multiprocessors”, In Proceedings of the 29th International Symposium on Computer Architecture, pp. 98-109. 2003.
8) R. Watanabe, et al. “Implementation and Evaluation of Superscalar Processor Based on Dynamic Adaptive Redundant Architecture”, In Proceedings of COOL Chips XIII, pp. 195, 2010.