SIMP（単一命令流／多重命令パイプライン）方式に基づく『新風』プロセッサの低レベル並列処理アルゴリズム†

久我守弘†† 入江直彦†† 弘中哲夫††
村上和彰†† 富田眞治††

高速汎用プロセッサアーキテクチャとしてSIMP (Single Instruction stream/Multiple instruction Pipelining: 単一命令流／多重命令パイプライン) 方式を提案し，それに基づく試作プロセッサ『新風』を開発している．『新風』プロセッサは同一構造の5ステージから成る命令パイプラインを4本有し，4つの命令を同時にフェッチしてパイプライン処理する．このとき，命令レベルの並列性を利用した低レベル並列処理を行う．低レベル並列処理の処理性能は，命令間に内在するデータ依存関係および制御依存関係といったハザードへの対処法によって左右されるが，『新風』ではこれらの依存関係を実行時に検出し，この検出された依存関係にのみ従って局所データフロー的に命令実行を制御する．すなわち，オブジェクトコード上の命令出現順序とは無関係 (out-of-order) に，命令実行順序を実行時に決定する動的コードスケジューリングを採用している．動的コードスケジューリングの手法としてはCDC 6600 における Thornton のアルゴリズムや IBM 360/91 における Tomasulo のアルゴリズムなどが有名であるが，『新風』ではこれらのアルゴリズムを拡張することにより，データ依存関係だけでなく制御依存関係への対処も可能な新しい低レベル並列処理アルゴリズムを考案し採用した．本論文では，このアルゴリズムの詳細について述べる．

1. はじめに

著者らは，シングルプロセッサ上で命令レベルの並列性を利用して単一プログラムを並列実行する低レベル並列処理に着目し，新しい高速汎用プロセッサアーキテクチャとして，従来の単一命令パイプラインにおける時間並列処理と VLIW 方式に代表される空間並列処理とを融合した SIMP（単一命令流/多重命令パイプライン）方式を提案し[1),2)]，その試作プロセッサ『新風』を開発している[2)-4)]．

低レベル並列処理の性能を左右する要因として，プログラム内の命令間のデータ依存関係および制御依存関係といったハザードへの対処方法が挙げられる．一般の命令パイプライン計算機では命令の実行順序はオブジェクトコード上の命令出現順序に忠実 (in-order) であり，後続の命令が先行する命令に対して依存関係にある場合には，パイプラインインタロック制御によってハザードに対処している．しかし，パイプラインインタロック制御では依存関係があるたびにパイプラインに乱れが生じ，それによる性能低下は SIMP 方式においては単一命令パイプラインのときにも増して深刻となる．

これに対して，プログラム内の命令間に内在する依存関係を検出して並列性を最大限に発揮するように命令を並べ替えるコードスケジューリングという技法がある．これには，スケジューリングをプログラムの実行前と実行時のいずれの時点で行うかで，静的および動的コードスケジューリングの2種類が存在する．VLIW 方式[5),6)]では，コンパイラが並列性を抽出し VLIW 命令に埋め込む静的コードスケジューリングを行うが，この場合ハードウェアの制御は簡単になる反面，処理性能が最適化コンパイラの能力にのみ依存してしまう．また，メモリアドレスや分岐先アドレスなど実行時にならないと判明しない依存関係については対処できない．このような静的コードスケジューリングの欠点を補うために，動的コードスケジューリングが重要となる．動的コードスケジューリングでは，実行時にハードウェアが主にデータ依存関係を解析して，ハザードにより誤った結果が生じない範囲で命令の実行順序を入れ替える (out-of-order)．

動的コードスケジューリングの手法としては，IBM 360/91 における Tomasulo のアルゴリズム[7)]などが有名であるが，これらは多重機能ユニット方式を対象としたものである．SIMP 方式は，多重機能ユニット方式の命令フェッチ部を同時に n 命令分フェッチ可能とし，さらにデコード部を n 多重化した“拡張された多重機能ユニット計算機”として捉えることができる

† Low-level Parallel Processing Algorithms for the SIMP Processor Prototype by MORIHIRO KUGA, NAOHIKO IRIE, TETSUO HIRONAKA, KAZUAKI MURAKAMI and SHINJI TOMITA (Department of Information Systems, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University).

†† 九州大学大学院総合理工学研究科情報システム学専攻

ため，Tomasulo のアルゴリズムを拡張することで動的コードスケジューリングを実現することが可能である．

本論文では，まず『新風』プロセッサの構成と命令の処理過程について述べた後，『新風』に採用した低レベル並列処理アルゴリズム，すなわち，命令間の依存関係の検出，表現，および，それに基づいた out-of-order 実行制御などの動的コードスケジューリング・アルゴリズムについて述べる．

2. 『新風』プロセッサの概要

2.1 『新風』プロセッサの構成[3),4)]

『新風』プロセッサは1本当たりピーク性能8 MIPS の命令パイプラインを4本有する．各命令パイプラインは，命令ブロックフェッチ (IF)，命令解読 (D)，命令発行(I)，実行(E)，および，リタイア(R)の5ステージから成る．プロセッサは**図1**に示すように，以下の主要ユニットから構成される．

(1) マルチバンク命令キャッシュ (MBIC : Multiple-Bank Instruction Cache) :

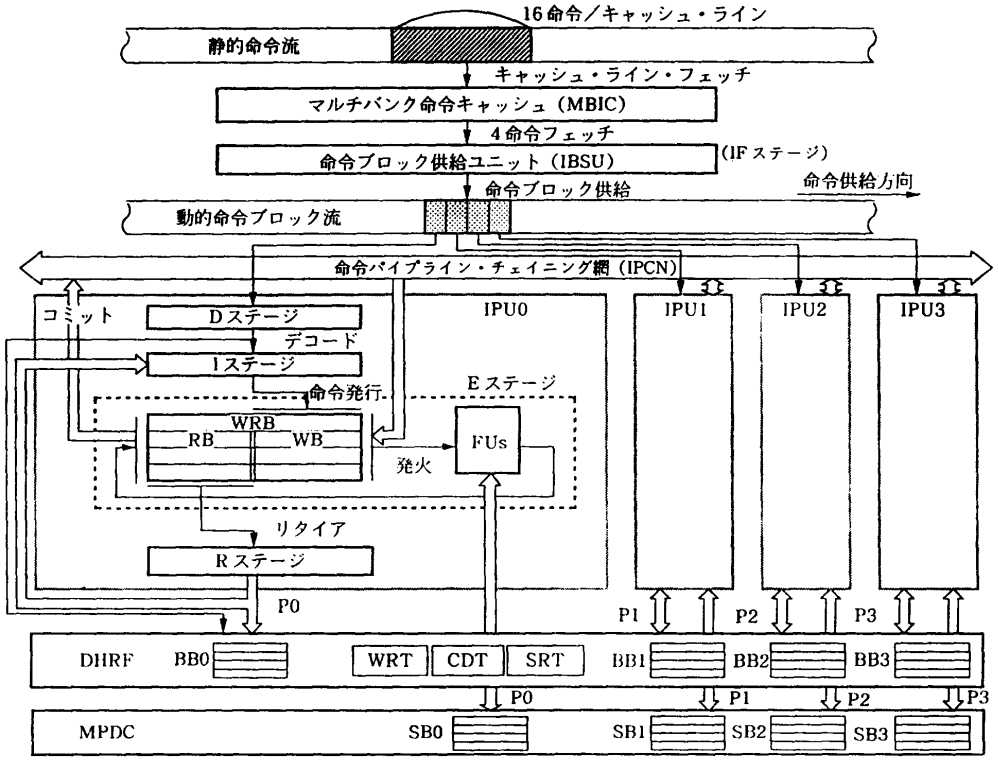
4つの命令を同時にフェッチできるように，4バンク構成の命令キャッシュである．分岐予測を可能にするため分岐先バッファを有する．

(2) 命令ブロック供給ユニット (IBSU : Instruction-Block Supply Unit) :

命令パイプラインの5ステージのうちの IF ステージに相当する．MBIC から連続する4個の命令を命令ブロック (IB) としてフェッチし，4本の命令パイプライン・ユニット (IPUs) にそれぞれ投入する．

(3) 命令パイプライン・ユニット (IPUs : Instruction Pipeline Units) :

4本の同一構成の命令パイプライン・ユニットが備えられており，各々，命令パイプラインの5ステージのうちの残り4ステージを担当する．Eステージは，整数 ALU，整数乗算器，浮動小数点数 ALU，浮動小数点数乗算器およびデータキャッシュ・アクセスリク



DHRF : 依存解析機能付きレジスタファイル　MPDC : マルチポート・データキャッシュ　WRT : 書込み予約テーブル
IPU : 命令パイプライン・ユニット　P : ポート　CDT : 制御依存テーブル
⟶ : 命令フロー　SB : ストア・バッファ　SRT : ストア予約テーブル
⟹ : データフロー　WRB : 供給・格納バッファ　BB : バイパス・バッファ

図1 『新風』プロセッサの論理構成
Fig. 1 Logical configuration of the 【ʃimpu:】 processor.

エスタと呼ぶ5つの機能ユニットを持つ．また，out-of-order 実行を実現するために，機能ユニットの前後には4エントリの供給バッファ(WB)および格納バッファ(RB)と呼ぶ対になって動作するバッファ(WRB)を有する．WB は IBM 360/91 における reservation station[7] に相当し，RB は文献8)における reorder buffer に相当する．

(4) 依存解析機能付きレジスタファイル(DHRF: Dependency-Handling Register File):

汎用レジスタ，浮動小数点レジスタなどを備えた読出しポート8，書込みポート4のレジスタファイルである．依存解析機能とは，命令間のデータ依存関係や制御依存関係を検出して，各命令の実行制御情報すなわち局所データフローグラフを作成することである．これらの依存関係を検出するために，書込み予約テーブル (WRT)，および，制御依存テーブル (CDT) を備える．また，各 IPU 内の WRB 中の命令実行結果をソースオペランドとしてバイパスできるように，バイパスバッファ (BB) と呼ぶ RB のコピーを設ける．

(5) マルチポート・データキャッシュ (MPDC: Multiple-Port Data Cache):

ロードポート4，ストアポート4，16バンク（1バイト/バンク）構成のデータキャッシュである．DHRF 同様，各 IPU 内の WRB 中のストアデータをロードデータとしてバイパスできるように，ストアバッファ (SB) と呼ぶ RB のコピーを設ける．

(6) 命令パイプライン・チェイニング網 (IPCN: IPU Chaining Network):

IPCN は WRB 間を接続するための相互結合網であり，各 IPU がバスマスタとなるバス4本から成る．各 IPU での命令実行結果は，種々のトークンとしてすべての IPU に送られる．IPCN は IBM 360/91 における共通データバス[7]に相当する．

2.2 命令パイプライン処理過程

命令パイプラインの各ステージについて，その処理の概要を以下に示す．I，EおよびRステージにおける out-of-order 実行制御の詳細については3章および4章で述べる．

(1) IF ステージ

IBSU は，MBIC から連続する4個の命令を命令ブロックとしてプリフェッチし，各 IPU のDステージに命令を投入する．命令を投入する際，命令ブロックを識別するための命令ブロック番号(IBN)と IPU を識別するための IPU 番号 (IPN) とから成る命令識別子 (IID) を各命令に付加する．以後，命令はこの命令識別子により管理される．

(2) Dステージ

投入された命令のデコードを各 IPU で同時かつ独立に行う．命令内にある種々の命令フィールドを解読し，解読結果（操作の種類，オペランドの種類など）を DHRF へ送る．

(3) Iステージ

Dステージでの解読結果にしたがって，命令間の依存関係を DHRF において解析する．そして，ソースレジスタの現在値をレジスタファイルあるいは BB から読み出し，データおよび制御依存関係を表す実行制御情報を付加して，Eステージに対し命令を“発行”する．

以上の IF，D，Iステージは命令ブロック単位で in-order に処理される．

(4) Eステージ

発行された命令は，まず WRB に登録される．WRB 中の命令はソースオペランドがそろい次第，局所データフロー的に“発火”される．この発火順序は out-of-order である．命令実行結果は WRB にいったん格納される．WRB 中の実行結果はすべての依存関係が解消され次第，IPCN 経由で全 IPU および DHRF に送られ，後続命令のソースオペランドとして使用可能となる．この動作を“コミット”と呼び，その順序も out-of-order である．

(5) Rステージ

命令ブロックを構成する4命令がすべてコミットされたら，DHRF および MPDC において実行結果の格納を行い，その命令ブロックを命令パイプラインから“リタイア”させる．リタイアの順序は，命令ブロック単位で in-order である．

3. out-of-order 実行モデルの概要[2),4)]

3.1 要　　件

命令パイプライン方式のプロセッサには，データ依存関係，制御依存関係および不正確なマシン状態の3つの解決しなければならない問題がある．

(1) データ依存関係

データ依存関係には，1個のデータに関する先行および後続命令のアクセス（読出し/書込み）順序の組合せにより，フロー依存（書込み後読出し），逆依存（読出し後書込み）および出力依存（書込み後書込み）

の3種類がある．ある先行命令に対して依存関係にある後続命令のデータアクセスは先行命令のデータアクセス終了を待たねばならないことから，命令実行順序が in-order に制限されパイプラインに乱れも生じる．

しかし，逆依存および出力依存は真のデータ依存ではなく，データ記憶（レジスタなど）の不足に起因するアクセス衝突が原因で生じるものである．これらはデータ記憶の名前をつけ換える（renaming）ことで除去可能であり[10]，『新風』でもEステージの RB を用いたレジスタ名つけ換え（register renaming）により，レジスタに関する逆依存および出力依存を除去している[2]．

一方，フロー依存は真のデータ依存であり除去不可能である．そこで，フロー依存によるパイプラインの乱れをできる限り小さくするために，『新風』ではフロー依存関係にある命令のデータ待合せ場所として WB をEステージに設け，先行命令に対してフロー依存関係にない命令から先に（つまり out-of-order に）実行を開始可能としている．

（2） 制御依存関係

分岐命令が存在する場合，分岐するか否か（条件分岐の場合），および，分岐先アドレスが確定するまで次にフェッチすべき命令が決まらない．この制御依存関係に起因するハザードは，パイプライン中に無効なサイクルを多数生じさせるため，一般にデータ依存関係よりパイプラインの乱れを大きくする．

このペナルティを抑えるために，『新風』では分岐ターゲットバッファ（BTB）による分岐予測[11]を採用している．これにより，毎サイクル IF ステージにおいて命令ブロックをプリフェッチすると同時に BTB を検索し，次サイクルでプリフェッチすべき命令ブロックの先頭アドレスを決定する．すなわち，BTB にヒットし "Taken" と予測した場合は BTB に登録されている分岐先アドレスを，それ以外の場合は連続するアドレスを次にプリフェッチすべき命令ブロックの先頭アドレスとする．

いずれの場合でも，分岐命令が命令パイプライン内に存在する場合，それ以降のすべての命令は分岐予測に従ってフェッチされていることから条件付きモードで処理されることになる．しかし，分岐予測がはずれた際には，条件付きモードで処理中の命令を無効化し正しい命令を再フェッチしてくるという命令パイプライン復元処理が必要となる．このとき，条件付きモードで処理中のすべての命令を単に無効化するのではなく，真に不要な命令のみを選択的に無効化（選択的無効化と呼ぶ）するようにしている[2]．これにより，命令パイプラインに既に投入されている多数の後続命令（『新風』では最大 23 命令）を有効利用できる．

（3） 不正確なマシン状態

不正確なマシン状態は，内部割込みや分岐命令により生じる可能性がある．これは，内部割込みの原因となった命令の実行時点，ないし，分岐命令の実行時点のプロセッサ状態（レジスタやメモリの内容）が，それ以降の命令の実行により既に変更されていて，正しい状態が保証されないことをいう．out-of-order 実行制御を行う場合，内部割込みに起因する不正確なマシン状態，すなわち不正確な割込み（imprecise interrupt）が問題となる[8),9)]．さらに，分岐命令を越えて out-of-order 実行制御を行おうとすると，問題は分岐命令にも及ぶ．しかし，この問題は，分岐予測がはずれた際の命令パイプライン復元処理の一環として対処可能である．その解決策の例として，文献8）の reorder buffer や文献9）の checkpoint repair 機構がある．『新風』では，reorder buffer に相当するEステージ内の RB によりこの問題を解決している．

3.2 特　　長

『新風』における out-of-order 実行モデルは原理的には Tomasulo アルゴリズムに基づいているが，以下の大きな拡張を施している；

① out-of-order 実行を4命令ブロック，すなわち16命令に対して行う．

② 命令間依存関係の検出を逐次的でなく，1命令ブロック中の4命令に対して同時に行う．

③ 1先行命令との間の単一のフロー依存関係だけでなく，複数（12〜15）先行命令との間の複数のフロー依存関係が，実行制御情報として表現される．これを多重依存関係表現と呼ぶ．

④ 分岐命令が16命令中に含まれていても，分岐命令の実行完了を待たずに，しかも，その分岐命令をまたいで out-of-order に実行を行う．

⑤ 命令ブロック単位の分岐予測を行う．分岐予測がはずれた場合でも，パイプライン全体を単に flush するのではなく，真に無効化すべき命令のみを選択して無効化する．これにより，多重条件付き実行モードを提供する．

⑥ 多重依存関係表現および多重条件付き実行モードを活用して，先行する命令に起因するフロー依存関係が解消され次第，命令実行を開始可能とする．

これを条件付き先行実行と呼ぶ．もし先行実行後，制御依存関係の解消により新たなフロー依存関係が生じた場合は，命令の再実行（backtrack）を行う．

⑦ 正確な割込みを保証する．

3.3 多重依存関係表現法

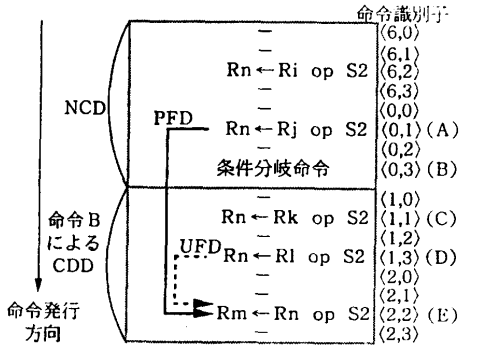
フロー依存関係と制御依存関係との相互関係に着目して，『新風』ではフロー依存関係をさらに以下の2種類に分けている；

① 最尤フロー依存関係（PFD: Probable Flow Dependency）：**図2**において，命令Eが命令Aに対してフロー依存関係にあり，かつ，命令Aに制御依存関係が存在しない場合，命令Eは命令Aに対して最尤フロー依存関係にあるという．

② 未確定フロー依存関係（UFD: Uncertain Flow Dependency）：図2において，命令Eが命令Dに対してフロー依存関係にあり，かつ，命令Dに制御依存関係（命令Bに起因）が存在する場合，命令Eは命令Dに対して未確定フロー依存関係にあるという．

したがって，ある1命令に関しては，高々1個のPFDと0個以上のUFDが存在し得る．

図2の同じ例に対して，Tomasuloアルゴリズムのタグを用いた依存関係表現を適用すると，唯一命令Dのみが命令Eに対してフロー依存関係を及ぼしていることしか表現できない．しかし，命令Dが実行されるかどうかは，分岐命令Bに依存している．したがって，もし分岐命令BがTaken（分岐する）で，かつ，命令Dと命令Eとの間に分岐したならば，命令Dはもはや実行されない命令であり，命令Dを識別していたタグも何ら意味を持たなくなる．このとき，命令Eが読み出すべきレジスタRnの値は命令Aの実行結果であるため，タグの修正が必要となる．しかし，ほとんどのout-of-order実行モデルでは，命令パイプラインをflushして正しい命令を再フェッチするので，結局命令Eはレジスタ Rn から正しい値を読み出すことになる．しかしながら，正しい命令が既に命令パイプライン中に存在するしないに関わらず，命令パイプライン中の全命令を無効にしてしまうのは効果的でない．特に『新風』の場合には，最大23個もの後続命令が既に命令パイプライン中に存在するため，分岐先命令が投入されている可能性は大きい．このような理由から，選択的無効化を採用している．選択的無効化を行う場合，Tomasuloアルゴリズムのタグ表現法のように1個のタグでは1命令つまり1個のフロー依存関係しか表現できないため不都合が生じる．そこで，最大15個の先行命令を識別するため，**図3**（b）に示すようなソース供給リスト（SSL）と呼ぶ16ビットのビットマップを用いて，複数のフロー依存関係を表現できるようにした．SSLはIステージにおいて，各レジスタ対応の書込み予約テーブル（WRT）（図3（a）参照）から作成される．WRTは，対応するレジ



NCD：非制御依存領域　PFD：最尤フロー依存関係
CDD：制御依存領域　UFD：未確定フロー依存関係

図2 最尤フロー依存関係と未確定フロー依存関係
Fig. 2 Probable flow dependency (PFD) and uncertain flow dependency (UFD).

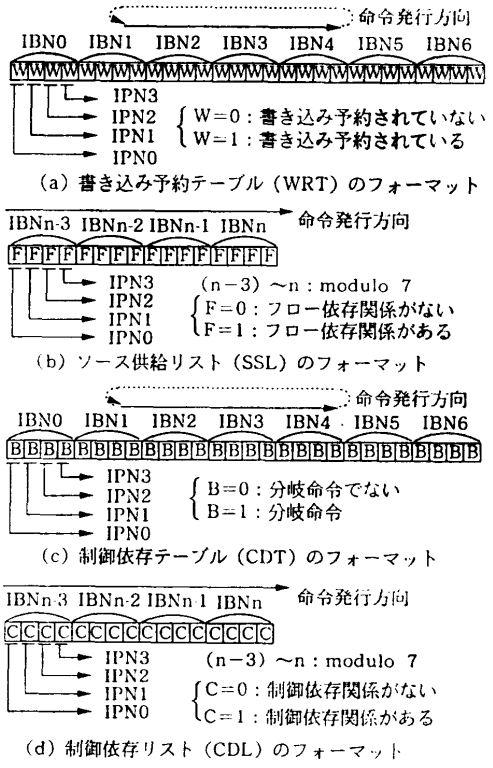


図3 WRT, SSL, CDT, および, CDLのフォーマット
Fig. 3 Formats of WRT, SSL, CDT and CDL.

スタをデスティネーションレジスタとする命令をすべて登録する．また，PFD と UFD を区別するために，図3(d)に示すような制御依存リスト (CDL) と呼ぶ16ビットのビットマップを導入する．CDL は SSL 同様，Iステージにおいて，図3(c)に示す制御依存テーブル (CDT) から作成される．CDT は，命令パイプライン中に存在するすべての分岐命令を登録する．CDL により，各命令は自分自身に対して制御依存を及ぼしている分岐命令をすべて知ることができる．

SSL と CDL とから PFD を識別するために，CDL を非制御依存領域 (NCD: No Control-dependent Domain) と制御依存領域 (CDD: Control Dependent Domain) とに分割する．NCD は CDL の最左ビット（最も先行する命令に対応）から最初に分岐命令が現れたビット位置までの領域であり，CDD はその残りのビット領域である．PFD を及ぼす命令は，SSL で NCD に対応する領域の中で最も右側に位置して（最も遅く投入された）フロー依存関係を起こしている命令である（**図4**参照）．

4. out-of-order 実行モデルの詳細[4]

『新風』の out-of-order 実行モデルにおいて，レジスタに関するフロー依存関係，および，制御依存関係を解消するアルゴリズムを示す．IF ステージから R ステージまでの，命令の状態遷移図を**図5**に示す．

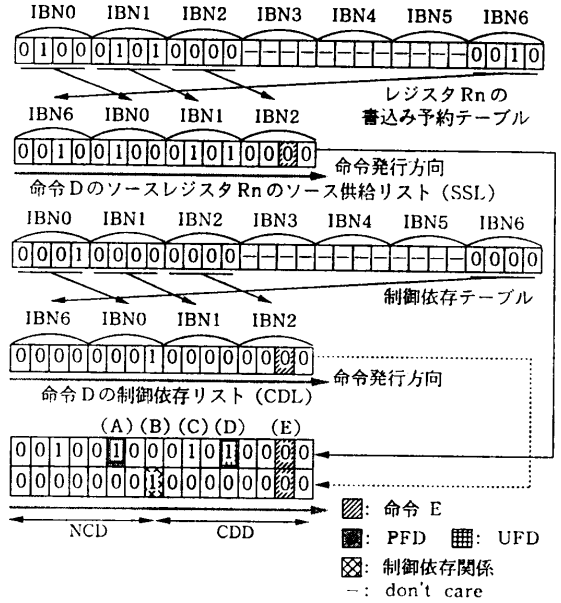


図4 最尤フロー依存関係の特定方法
Fig. 4 Mechanism of identifying PFD.

各々の命令は以下のトリガにより状態を遷移する．

4.1 発行 (issue)

Iステージにおいて各々の命令を発行するにあたり，DHRF では以下の処理を行う；

① デスティネーションレジスタに対応する WRT エントリに書込みを予約する．ソースレジスタに対応する WRT エントリを基に SSL を作成する．

② 発行する命令が分岐命令である場合 CDT に登録する．また，各々の命令について CDT を基に CDL を作成する．

③ ソースオペランドの現在値をレジスタファイルまたは BB から読み出す．

④ ソースオペランド，SSL および CDL を WRB の最後尾エントリに登録することで命令を発行する．WRB が一杯の場合インタロックする．WRB の構成を**図6**に示す．

発行された命令は，発行済み (issued) 状態に入る．

4.2 発火 (fire)

(1) 発火ルール

WRB 中の発行済み状態の各命令について，各 IPU

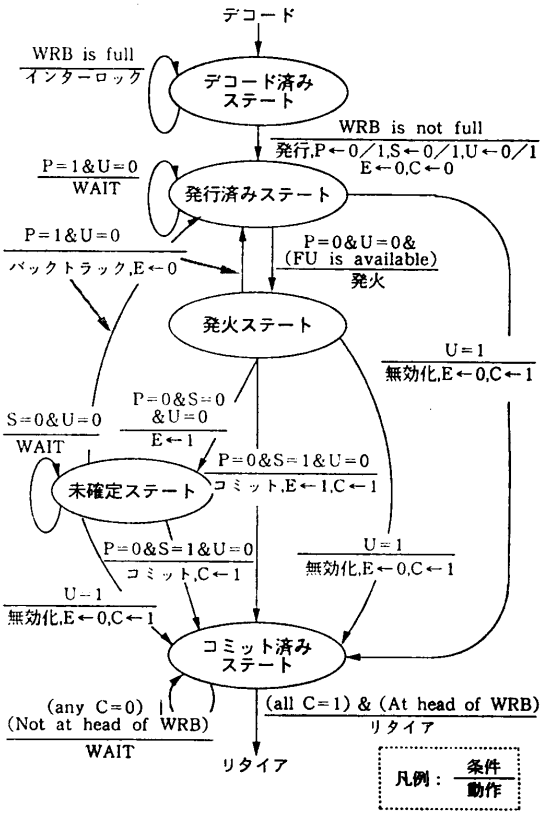
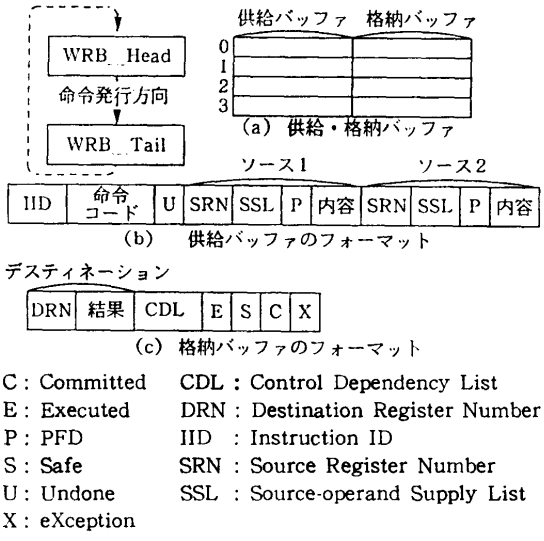


図5 命令の状態遷移図
Fig. 5 State diagram of an instruction.



C : Committed　CDL : Control Dependency List
E : Executed　DRN : Destination Register Number
P : PFD　IID : Instruction ID
S : Safe　SRN : Source Register Number
U : Undone　SSL : Source-operand Supply List
X : eXception

図 6 供給・格納バッファの構成
Fig. 6 Configuration of waiting and reorder buffer (WRB).

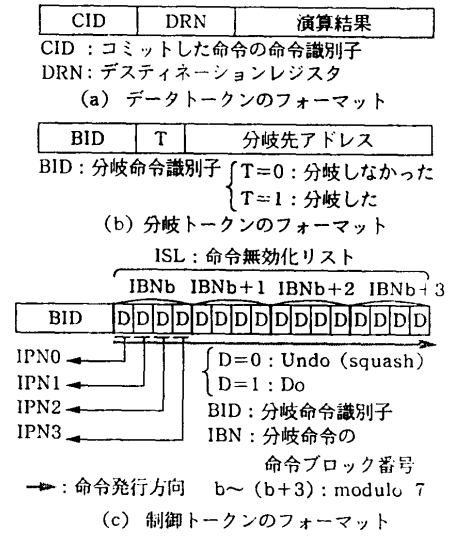


図 7 トークンのフォーマット
Fig. 7 Formats of tokens.

は，命令を発火できるかどうかチェックする．以下の発火ルールを満たす命令が発火される；

① すべてのソースオペランドが有効である．すなわち，WRB の P (PFD) ビットが '0' である (PFD が存在しない)．

② 命令が無効化/実行/コミットされていない．すなわち，WRB の U(Undone)，E(Executed)，C (Committed) ビットがすべて，'0' である．

③ 命令が使用する演算ユニットが使用可能である．

もし，複数命令が発火可能である場合には，命令発行が早かった順に発火される．また，発火可能な命令が1個もない場合は，どの命令も発火されない．

(2) データ依存関係の解消方法

PFD が存在する場合，命令は発火不可能である．このとき IPCN をモニタしてソースオペランドを受け取り，PFD を解消しなければならない．データ依存関係は，以下のように解消される；

① 各サイクルごとに，最大4個までのデータトークンが IPCN により送られてくる．データトークンは**図 7**(a)に示すように，コミットした命令の命令識別子(CID)，実行結果およびデスティネーションレジスタ番号 (DRN) が含まれる．

② すべてのデータトークン中の CID でインデクスされる SSL ビットをリセットする．

③ もし，リセットされた SSL ビットが PFD を示していて，かつ，P ビットが '1' であれば，待っていたソースオペランドが到着したことを意味する．よって，データトークン中の実行結果を WRB のソースオペランドフィールドに格納すると同時に P ビットをリセットする．

(3) 実行完了後の状態遷移

発火した命令の実行が完了したら，その結果は WRB 中の結果フィールドに格納される．この時点で，以下のチェックがなされる；

① もし，P ビットが '1' であれば，新たな PFD が出現したことを意味する．このとき，その命令は発行済み状態にバックトラックされる．

② 逆に，P ビット '0' であれば E ビットをセットする．さらに，S (Safe) ビットが '1' であれば，制御依存関係がすべて解消されたことを意味するので，その命令は直ちにコミットされ，コミット済み状態に入る．

③ 一方，S ビットが '0' で制御依存関係が解消していない場合は，解消するまで命令は未確定状態に入る．

(4) 制御依存関係の解消方法

制御依存関係は以下のように解消される．

① 分岐命令がコミットされると，図7(b)に示す分岐トークンが IBSU に送られる．

② IBSU は，分岐トークンによって無効化すべき命令を決定する．そして，図7(c)に示す制御トークン中の命令無効化リスト (ISL) により，無効化すべき命令を全 IPU に知らせる．

③ 各 IPU は制御トークンを受け取ると，無効化すべき命令に対応する WRB の U ビットをセットし

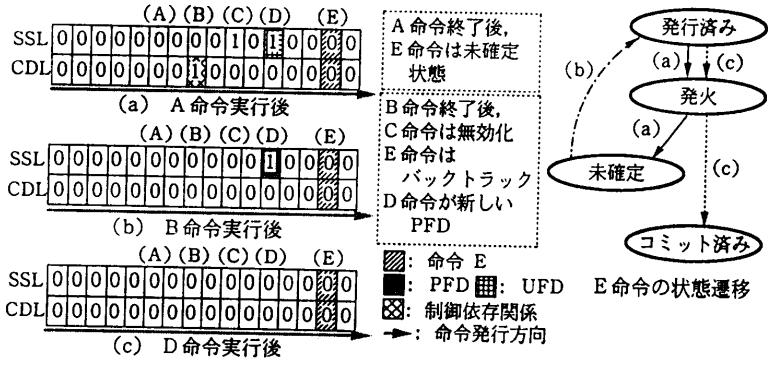


図 8 out-of-order 命令実行の例
Fig. 8 Example of out-of-order instruction execution.

て，その命令が無効化されたことを示す．

④ また，SSL および CDL を ISL でマスクする．これにより，無効化すべき命令に対応するビットがリセットされる．

⑤ さらに，制御トークン中の BID でインデクスされる CDL ビットをリセットする．

⑥ もし，CDL ビットがすべて '0' となれば，制御依存関係が解消されたことを意味し，Sビットをセットする．このとき，新たな PFD が現れた場合にはPビットをセットし，バックトラックを起こす．

4.3 バックトラック (backtrack)

命令発火は，先行する分岐命令の実行完了/未完了に関わらず行うことができる（条件付き先行実行）．いったん発火された命令に対しても，制御依存関係の解消に伴って新たな PFD が現れる可能性がある．このとき，その命令は発行済み状態にバックトラックされ，新しい PFD が解消されて再発火可能となるまで待つ．

4.4 無効化 (undo)

無効化された命令は WRB 中の Uビットがセットされる．そして，Eビットをリセット，Cビットをセットして，直ちにコミットされてコミット済み状態に入る．

4.5 コミット (commit)

命令をコミットするためのルールは以下のとおりである；

① 命令の実行が完了し，Eビットがセットされている．

② PFD が残っていない．すなわち，正しいオペランドを使用して演算を行った．

③ 制御依存がすべて解消している．すなわち，Sビットがセットされている．

以上の条件を満たしている場合，WRB 中のCビットをセットし，その命令をコミットする．コミットした命令の実行結果は，データトークンにより IPCN を経由して各 IPU，DHRF 内の BB，および，MPDC 内の SB に送る．また，コミットした命令が分岐命令の場合，分岐トークンを IPCN 経由で IBSU に送る．

4.6 out-of-order 命令実行の例

図2の命令流において，命令 A，B の順に実行終了し，命令Bが命令CとDとの間に分岐した場合の実行過程を図8に示す；

① 命令A，Bがまだコミットしていないとき，命令Eは図4に示す SSL および CDL を持つ．

② PFD である命令Aがコミットしたとき，その結果を用いて命令Eは発火する．命令Eの実行終了後は，まだ制御依存関係が解消されていないため未確定ステートに入る（図8(a)参照）．

③ 命令Bがコミットし命令CとDとの間に分岐すると，命令Cは無効化され命令Dが新しい PFD となる．したがって，命令Eはバックトラックを起こし再び発行済みステートとなる（図8(b)参照）．

④ 命令Dがコミットすると，命令Eはその結果を用いて再び発火する．命令Eの実行終了後は，SSL および CDL はすべて '0' でありコミットルールを満たすためコミット済みステートに入る（図8(c)参照）．

5. おわりに

以上，『新風』プロセッサの概要，および，低レベル並列処理を遂行するための out-of-order 実行モデル各種アルゴリズムについて述べた．ここで述べた低レベル並列処理アルゴリズムの一部を用いて，ソフトウェア・シミュレーションによる簡単な性能予測を行った結果，最適化されていないオブジェクトコード* に対しても約 15 MIPS の性能が得られている[3]．

『新風』プロセッサの詳細な性能予測，および，実機での性能評価の報告は別の機会に譲りたい．

謝辞 初期設計に携わっていた五島龍宏氏（現在東芝(株)），現在我々とともに設計・開発を行っている権五鳳，音成幹，原哲也の各氏，および，日頃御討

* Whetstone, Quick-sort, Eratosthenes-sieve, Fibonacci など．

論いただく富田研究室の皆様に感謝いたします.

なお，本研究は一部文部省科研費試験研究による.

参 考 文 献

1) 村上ほか：SIMP（単一命令流/多重命令パイプライン）方式の構想，情報処理学会研究報告，88-CA-69-4 (1988).
2) Murakami, K. et al.: SIMP (Single Instruction stream/Multiple instruction Pipelining): A Novel High-Speed Single-Processor Architecture, *Proc. 16th Annual Int. Symp. Computer Architecture*, pp. 78-85 (1989).
3) 入江ほか：SIMP（単一命令流/多重命令パイプライン）方式に基づく『新風』プロセッサの高速化技法および性能予測，情報処理学会研究報告，88-ARC-73-11 (1988).
4) 久我ほか：SIMP（単一命令流/多重命令パイプライン）方式に基づく『新風』プロセッサの低レベル並列処理アルゴリズム，並列処理シンポジウム JSPP '89 論文集，pp. 163-170 (1989).
5) Fisher, J. A.: Very Long Instruction Word Architectures and the ELI-512, *Proc. 10th Annual Int. Symp. Computer Architecture*, pp. 140-150 (1983).
6) 富田眞治：マイクロプログラム制御方式の動向，情報処理，Vol. 28, No. 12, pp. 1540-1552 (1987).
7) Tomasulo, R. M.: An Efficient Algorithm for Exploiting Multiple Arithmetic Units, *IBM J. Res. Dev.*, Vol. 11, pp. 25-33 (1967).
8) Sohi, G. S. et al.: Instruction Issue Logic for High-Performance, Interruptable Pipelined Processors, *Proc. 14th Annual Int. Symp. Computer Architecture*, pp. 27-34 (1987).
9) Hwu, W. W. et al.: Checkpoint Repair for Out-of-Order Execution Machines, *Proc. 14th Annual Int. Symp. Computer Architecture*, pp. 18-26 (1987).
10) Keller, R. M.: Look-Ahead Processors, *Comput. Surv.*, Vol. 7, No. 4, pp. 177-195 (1975).
11) Lee, J. K. F. et al.: Branch Prediction Strategies and Branch Target Buffer Design, IEEE Comput., Vol. 17, No. 1, pp. 6-22 (1984).

(平成元年5月30日受付)
(平成元年9月12日採録)

久我 守弘（正会員）

1965年生．1987年福岡大学工学部電子工学科卒業．1989年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．現在，同大学院博士課程に在学中．並列処理，計算機アーキテクチャの研究に従事.

入江 直彦（学生会員）

1965年生．1988年九州大学工学部電子工学科卒業．現在，同大学大学院総合理工学研究科情報システム学専攻修士課程に在学中．並列処理，最適化コンパイラの研究に従事.

弘中 哲夫（学生会員）

1965年生．1988年山口大学工学部電気工学科卒業．現在，九州大学大学院総合理工学研究科情報システム学専攻修士課程に在学中．並列処理，ベクトル処理，計算機アーキテクチャの研究に従事.

村上 和彰（正会員）

1960年生．1982年京都大学工学部情報工学科卒業．1984年同大学院工学研究科情報工学専攻修士課程修了．同年富士通(株)本体事業部に入社．主として汎用計算機Mシリーズのアーキテクチャ開発に従事．1987年九州大学工学部助手，1988年同大学院総合理工学研究科情報システム学専攻助手，現在に至る．計算機アーキテクチャ，スーパコンピューティング，並列処理等の研究に従事．著書「計算機システム工学（共著，昭晃堂)」．電子情報通信学会，ACM，IEEE-CS 各会員.

富田 眞治（正会員）

1945年生．1968年京都大学工学部電子工学科卒業．1973年同大学院博士課程修了．この間，零交さ波による音声合成の研究に従事．工学博士．同年京都大学工学部情報工学教室助手．1978年同助教授．1986年九州大学大学院総合理工学研究科教授，現在に至る．計算機アーキテクチャ，並列処理システムなどに興味を持つ．著書「並列計算機構成論」「計算機システム工学」「並列処理マシン」など．電子情報通信学会，IEEE, ACM 各会員.