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

2007－ARC－172 (5)
2007－HPC－109 (5)
2007／3／1

VLIW型命令キューを持つOROCHIの命令スケジューリング機構

片岡晶人† 中西正樹†
山下茂† 中島康彦†

スーパースカラプロセッサは，複雑な命令発行機構や演算器周辺回路によって命令レベル並列度を向上させているものの，消費電力の削減が問題となっている．本稿では命令をVLIW型命令キューの適切な位置に登録することにより，命令発行機構や演算器周辺回路を簡素化できる手法を提案する．提案モデルと対抗モデルのパイプラインシミュレータによるIPC測定と，対抗モデルの遅延時間の評価と，提案モデルの遅延時間の見積もりを行った結果，提案手法は既存の集中命令ウィンドウ型スーパースカラより21%性能向上できることがわかった．

Instruction scheduling method for OROCHI with VLIW Instruction Queue

Akihito KATAOKA,† Masaki NAKANISHI,†
Shigeru YAMASHITA† and Yasuhiko NAKASHIMA†

Complex issue logic and reservation station have improved the instruction level parallelism in superscaler processors, though the compicated logic increases the power consumption. In this paper, we propose method for instruction scheduling to appropriate position in VLIW instruction queue to simplify issue logic. We evaluated IPC using a pipeline simulator, and we estimated the delay time of the pipeline model written in Verilog-HDL. The result shows our method can speed up in 21 % to the super scaler processor with centralized instruction window.

1. まえがき

近年，携帯端末機器などの組み込みシステムでは，命令レベル並列度を期待できないOSなどの制御プログラムと，高い命令レベル並列度を期待できるマルチメディア処理プログラムを同時に実行する環境が一般化してきている．並列性が高いマルチメディア処理プログラムの実行には，並列処理能力が高く，低消費電力であるVLIWプロセッサが有利である．しかしOSなどの制御プログラムの実行には，ソフトウェア資産の豊富な，業界標準の汎用プロセッサが有利である．このためマルチメディア処理とOSを同時実行する環境では，一般に複数種類のプロセッサの並置が行われている．

一方，我々の研究グループでは，バックエンドをVLIW型の構成としつつ，複数のフロントエンドを有するSMT(Simultaneous Multi-Threading)プロセッサOROCHI[1]を提案している．OROCHIでは汎用の命令セットとマルチメディア処理用の命令セットに各々対応するフロントエンド，例えばARM[2]とFRV[3]といった組み合わせを用意することで異種命令セットの同時実行を図る．

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

OROCHIの構成方法を検討するにあたり，まず汎用命令用のフロントエンド，VLIW型バックエンドを備えるOROCHIについて評価することにした．VLIW型バックエンドはスーパースカラのように複雑な命令発行機構を必要とせず，VLIW型の命令キューに汎用の命令をスケジューリングする構成とできるため，低消費電力化や動作周波数の向上を期待できる．

本研究では，ARM用のフロントエンドとVLIW型のバックエンドからなるOROCHIモデル，およびARM用のフロントエンドとスーパスカラ型のバックエンドからなるARM-SSモデルの2種類のプロセッサを実際に設計し，両者を比較することを目的とした．本稿ではOROCHIモデルとARM-SSモデルのシミュレータによるベンチマークプログラムの走行結果を示し，またARM-SS命令発行機構の論理合成結果を用いて，提案するOROCHIモデルの優位性を示す．

2. 従来型スーパースカラと提案手法の概要

従来型のスーパースカラには，Pentium Pro[4]のP6アーキテクチャ[5]やPentium 4[6]のNetBurstアーキテクチャに代表されるリザベーションステーション方式と，MIPS R10000[7]に代表される集中命令ウィンドウ方式がある．

2.1 リザベーションステーション方式

リザベーションステーション方式では，仮のソースオ

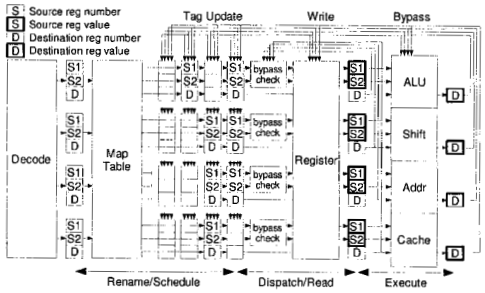

図 1　VLIW 型命令キュー方式

ペランド値を読み出し，演算器毎に存在するリザベーションステーションにキューイングする．キューイングされた命令は，真のレジスタ値が依存関係にある命令から供給されるまでリザベーションステーションで待機する．依存関係にある命令が実行されると，その結果がリザベーションステーションにフォワーディングされ，待機している命令は真のレジスタ値を得ることができる．そして Execute ステージにおいて，全てのソースオペランドが使用可能となった命令が選択されて演算器に投入される．この方式ではリザベーションステーションにエントリ数 ×2 個のソースオペランド値を保持することになるため，大きな記憶容量を必要とする．また各演算器から全てのリザベーションステーションの全エントリへ実行結果をフォワーディングするため回路も必要である．一方，1 サイクルでリザベーションステーションの全エントリから発行可能な命令を選択して演算器に投入するため，Execute ステージの遅延時間が大きくなる．

2.2　集中命令ウィンドウ方式

集中命令ウィンドウ方式では，1 つの命令ウィンドウにキューイングされた命令は，ソースオペランドを供給する命令が実行されるまで命令ウィンドウで待機する．依存関係にある命令が実行されると，デスティネーションレジスタ番号が命令ウィンドウに通知される．命令ウィンドウは通知されたデスティネーションレジスタ番号と，ソースレジスタ番号の一致比較を行い，Select/Read ステージにおいて，命令ウィンドウから全てのソースオペランドが使用可能となった命令を選択し，ソースレジスタ値を読み出した後に演算器の入力バッファに投入する．この方式ではフォワーディングが不要となるものの，デスティネーションレジスタ番号とソースレジスタ番号の一致比較回路は必要である．また発行可能な命令の選択とソースレジスタ値の読み出しを 1 サイクルで行うため，Select/Read ステージの遅延時間が大きくなる．

2.3　提案手法 (VLIW 型命令キュー方式)

以上のように，従来型のスーパースカラではリザベーションステーション，フォワーディング機構，レジスタ番号の一致比較回路，および，リザベーションステーションまたは命令ウィンドウの任意のエントリから命令を選択して発行する機構が必要であり，これらはプロセッサの電力消費量を上げる要因の 1 つとなっている．また多数のエントリから発行可能な命令を探し出す回路は遅延が大きく動作周波数に悪影響を与える．そこで，フォワーディング機構や複雑な命令発行機構が不要となる VLIW 型命令キュー方式を提案する．図 1 に示す方式では，Rename/Schedule ステージにおいて命令間の依存関係を検査し，VLIW 型命令キューの適切な位置にスケジューリングする．次に，Dispach/Read ステージにおいて，VLIW 型命令キュー右端の全エントリが発行可能かどうかを検査し，発行可能であればレジスタ値を読み出し，演算器の入力バッファに投入する．

VLIW 型命令キューは演算器ごと入り口が分かれており，各エントリはデコーダ側から演算器側に向かうシフトレジスタである．命令発行は常に右端のみから行い，複雑な命令発行機構は無い．このため動作周波数や回路面積の面で有利である．一方，命令のキューイングは任意の位置に可能であり，依存関係が無ければ既に命令が入っている位置より右側，すなわち追い越してキューイングすることができる．命令の順序を並び替えてスケジューリングできるため，既存のスーパースカラと同等の命令レベル並列度が期待できる．ロード命令がキャッシュミスを起こした場合，後続命令がロードストアユニット (Cache) を使用するか，ロード結果を使用するまで命令発行を継続できる．またロード命令とロード結果を使用する命令を離してスケジューリングすることにより，影響を軽減できる．同一タイミングでスケジューリングされる命令間に依存関係がある場合は，命令順に先頭の命令から見て後方となるようスケジューリングすることが必要である．このために必要な情報は，従来型スーパースカラもレジスタリネーミング機構が生成しているため，新たな機構は必要ない．

このように，Rename/Schedule ステージの遅延時間がスーパースカラの命令発行機構の遅延時間より小さくできると見込まれ，全体として動作周波数の向上が期待できる．

3.　パイプラインモデル

提案モデルの優位性を示すため，ARM 命令を実行する 2 種類のプロセッサをパイプラインシミュレータおよび Verilog-HDL により実装した．ARM は命令語により指定可能なレジスタは 16 本であるものの，割り込み処理のために裏レジスタを持っており，レジスタ総数は 31 本である．論理レジスタを物理レジスタの一部にマッピングする物理レジスタ方式により実現すると，普段は使われない裏レジスタを常に物理レジスタにマッピングしておく必要があり，レジスタリネーミングの効果が低下する．このためスーパースカラモデルは，リオーダーバッファを兼ねる集中命令ウィンドウ方式とした．論理レジスタを分けることで効率の良いリネーミングが可能となる．

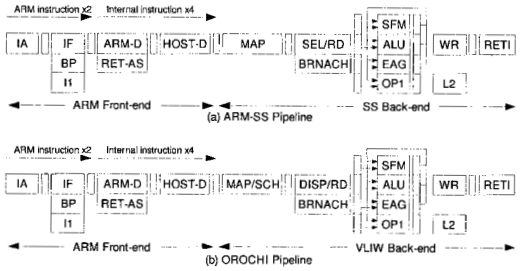

図 2　パイプライン構成

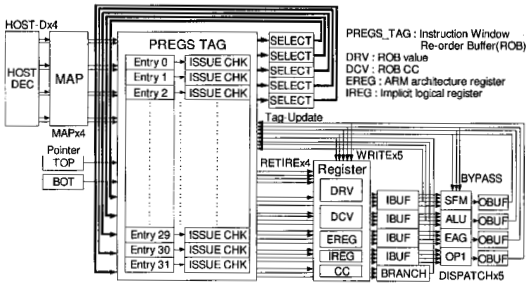

図 3　ARM-SS モデルのバックエンド

3.1　パイプライン構成

既存手法を用いた ARM-SS モデルと，提案手法 (以下 OROCHI モデル) のパイプライン構成を図 2 に示す．IA ステージから HOST-D ステージまでは，ARM-SS モデル，OROCHI モデルに共通のフロントエンドである．IA ステージは PC を生成する．IF ステージは I1 キャッシュにアクセスし，連続する ARM 命令を 2 つ同時にフェッチする．また G-share 分岐予測を行う．ARM-D，HOST-D ステージは複雑な ARM 命令を内部命令に分解[8)9)] する．また ARM-D ステージでは ARM の 15 番レジスタに書き込む命令 (分岐命令) を検出し，リターンアドレススタックを使って分岐先アドレスを予測する．

ARM-D ステージ，HOST-D ステージの命令分解機構は，ARM 命令 1 個を平均 2 個の内部命令に分解する．1 サイクルごとに最大 4 個の内部命令への分解が可能である．また，分解後の命令は ARM のアーキテクチャレジスタ (EREG:Explicit Register) とは別に 6 個の内部命令用レジスタ (IREG:Implicit Register) を使用する．条件コードについても同様に ARM アーキテクチャの条件コード (LCOND) とは別に，内部命令用のシフターキャリー (LSFTC) を持つ．いずれのレジスタもバックエンドにおけるレジスタリネーミングの対象となる．

ARM-SS モデルでは MAP ステージ以降が，OROCHI モデルでは MAP/SCH ステージ以降が各々バックエンドである．演算器周辺の構成は同一であり，ALU，シフトと乗算命令を実行する SFM，アドレス計算と乗算の補助演算を実行する EAG，ロードストア命令を実行する OP1 から構成される．SFM，ALU，EAG の結果は，各演算器にバイパスできる．これらの演算器は必ず 1 サイクルで動くので，入力バッファの内容を見て次の命令を発行できる．このようにバイパスを使えるようにするために命令分解を行っている．OP1 からのバイパスはキャッシュミスによって結果を供給できない場合があるので設けていない．また分岐命令を実行する BRANCH は特別であり，命令発行と同じステージ (ARM-SS モデル:SEL/RD ステージ，OROCHI モデル:DISP/RD ステージ) において分岐予測結果と分岐結果が一致しているかどうかの判定を行う．

3.2　ARM-SS モデルのバックエンド構成

ARM-SS のバックエンドの構成を図 3 に示す．分解された命令はまず，マップテーブル (MAP) にてレジスタリネーミングされる．マップテーブルは論理レジスタとリオーダーバッファの対応関係を記憶している．レジスタリネーミングされた命令は，命令ウィンドウ (PREGS_TAG) の最後尾 (TOP ポインタの位置) にキューイングされる．命令ウィンドウは 32 エントリあり，TOP と BOT の 2 つのポインタで制御される環状バッファになっている．レジスタリネーミングの結果，ソースオペランドが論理レジスタ，リオーダーバッファのどちらにあるのか，使用可能かというタグ情報が得られる．このタグ情報に基づいて各エントリに存在する命令発行判定機構 (ISSUE_CHK) は，そのエントリが発行可能か判定し発行許可信号を出力する．このとき命令をどの演算器に発行するか，次サイクルで演算器間のバイパスが使用できるかも判定している．

バイパスチェックの方法は演算器入力バッファ (IBUF) にある SFM，ALU，EAG に投入される命令のデスティネーションレジスタ番号とソースレジスタ番号の一致比較によって行う．また，タグ情報の更新と命令発行判定機構は同じタイミングで動作するので，演算器出力バッファ (OBUF) の内容も考慮してソース使用可否を決定する必要がある．各エントリの発行許可信号は演算器ごとの発行機構 (SELECT) へ送られ，演算器に発行する命令を選択する．このとき 1 つの演算器に対して 2 つ以上の命令が発行可能になることがあり，命令選択アルゴリズムが回路規模や IPC に影響を与える．ARM-SS モデルでは，回路の簡単化のため，エントリの物理的な順番によって発行する命令を選択する方式を採る．この結果，発行機構は単純なプライオリティエンコーダとできる．この方法は回路が単純になるが，IPC が低下することが分かっている．そこでパイプラインシミュレータにより，エントリの命令順によって発行する命令を選択する理想的な方式について IPC を測定する．このモデルを ARM-SS(ideal) モデルとする．なお，ARM-SS(ideal) モデルを回路で実現しようとすると，基本的には発行許可信号を並び替えるエントリ数分の入力を持つマルチプレクサがエントリ数分必要となる．このため回路化は

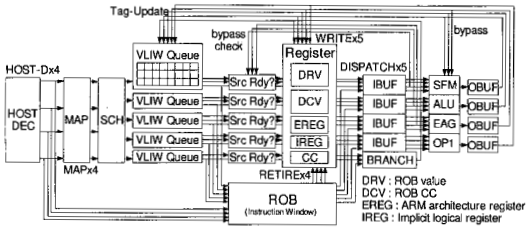

図 4　OROCHI モデルのバックエンド

困難と判断した．発行機構のアルゴリズムについてはいくつかの折衷案が考えられるが，本稿では議論の対象としない．

発行機構は，選択したエントリ番号のワード線のみをアクティブにする．これを用いて命令ウィンドウから命令発行に必要なオペコードやタグ情報を読み出す．その後タグ情報に基づいて，リオーダーバッファ (DRV) や論理レジスタ (EREG, IREG) からソースオペランド値を読み出し，演算器入力バッファに格納する．命令の発行は演算器 4 個に同時発行可能で，命令 1 個は 2 個のソースオペランドを使用するため，リオーダーバッファには 8 個の読み出しポートと 4 個の書き込みポートが必要である．それに加えてリタイアのために 4 個の読み出しポート，リンク付き分岐命令のために 1 個の書き込みポートが必要となる．分岐命令はリーオーダーバッファ(DCV) または条件コードレジスタ (LCOND) から条件コードを読み出し，分岐実行 (BRANCH) で分岐予測が正しかったかを判定し，その結果をリオーダーバッファに格納する．このように，発行機構，命令ウィンドウからのソースレジスタ番号読み出し，リオーダーバッファまたは論理レジスタからの読み出しからの読み出しという 3 つのオーバーラップできない動作が存在する．分岐命令ではさらに分岐実行の動作が必要である．よって SEL/RD ステージ内のパスがプロセッサ全体の動作周波数を決定するとみて間違いない．

3.3 OROCHI モデルのバックエンド構成

OROCHI モデルのバックエンド構成を図 4 に示す．分解された命令は ARM-SS モデルと同様にレジスタリネーミングされる．その結果を用いてスケジューリング機構 (SCH) が，VLIW 型命令キューの適切な位置に命令をキューイングする．またスケジューリングと同時に，リオーダーバッファ(ROB) へのキューイングも行う．このとき VLIW 型命令キューのエントリにリオーダーバッファのエントリ位置を記憶しておく．VLIW 型の命令キューはソースオペランドのタグ情報とリオーダーバッファ上での位置を 8 エントリ分記憶している．このようなキューが各演算器 SFM, ALU, EAG, OP1, BRANCH 毎にあるので，全体のエントリ数は 8 エントリ ×5 個となっている．しかし実際に使えるエントリ数はリオーダーバッファのエントリ数の 32 個までとなる．前述のように命令の発行は右端のみからとなっており，右端には発行判定機構 (ARM-SS モデルの ISSUE_CHK に相当) が接続されている．発行判定機構が発行を許可すると，ソースオペランドが読み出され，演算器入力バッファに投入される．

VLIW 型命令キューは任意の位置に命令をキューイングできるが，適切な位置へのスケジューリングには制限がある．まず，デコーダからの情報に従って，命令を投入すべき演算器に繋がっているキューを選択する際，同じキューを使用する命令が 2 個以上あるときは，命令順で早いものを優先する．この結果衝突した命令とそれ以降の命令のスケジューリングを 1 サイクルで行うことはできない．次にマップテーブルからの情報に従って，全てのソースオペランドが供給される位置を見つける．ただし条件を満たす位置がない場合，具体的には VLIW 型命令キューの左端の命令がソースオペランドを供給する場合はスケジューリング失敗となる．結果として，ROB のエントリに空きがあるのにスケジューリングに失敗することがあり，ARM-SS モデルよりバックエンドへの命令の供給能力が劣ることが懸念される．なお，ソースオペランドが供給される位置を見つけるためには，VLIW 型命令キューの各エントリのデスティネーションレジスタ番号とソースレジスタ番号の一致比較を行い，できるだけ演算器に近いエントリを探すことになる．但しバイパスが使用できない演算器，すなわちロード命令からのオペランド供給は 1 サイクル遅れることを考慮する必要がある．

4. 性能評価

4.1 シミュレータによる IPC の評価

ARM-SS モデルおよび OROCHI モデルのシミュレータにより MiBench を走行させたときの内部命令の IPC を図 5 に示す．なお，キャッシュ容量などの条件は ARM-SS モデル，ARM-SS(ideal)，OROCHI モデルとも共通であり表 1 に示すとおりである．ARM-SS(ideal) とは前述の命令順に基づいて発行するモデルである．

まず全体的に，ARM-SS(ideal) モデルが最も IPC が高く，OROCHI モデルは平均的に IPC が低いことが分かる．rijndael(1) と CRC32 では OROCHI モデルが逆転しているが，3 つのモデル間の差は少なく，これらは誤差の範囲と言える．susan(1) などでは OROCHI モデルと ARM-SS(ideal) モデルの差が大きい．全体的に見て IPC が高いもの，すなわち並列度が高いものについて IPC に差が出ていることが分かる．平均値では，ARM-SS(ideal) モデルに対して 5.7%の低下，ARM-SS モデルに対して 0.3%程度の低下となり，OROCHI モデルと ARM-SS モデルの間にはほとんど差が無い．

OROCHI モデルの IPC が低くなる要因は 2 つ考えられる．1 つはバックエンドへの命令供給能力である．ARM-SS モデルでは SFM 命令を 4 個同時に命令ウィンドウにエンキューできるが，OROCHI モデルでは同じ演算器に対しては 1 命令ずつしかスケジューリン

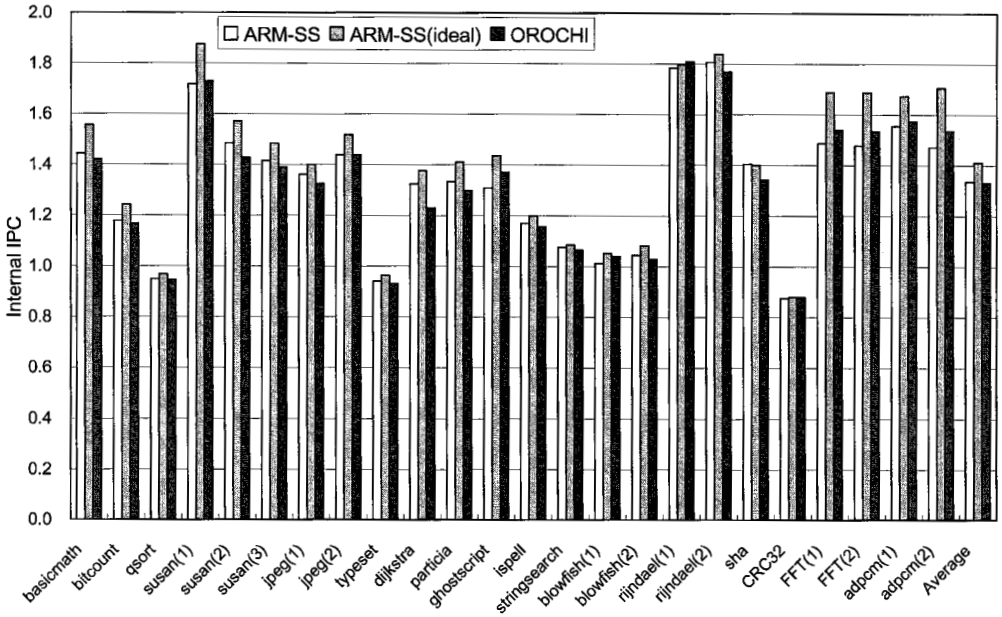

図 5 内部命令の IPC

表 1 シミュレーション条件

分岐予測 (gshare)	pht:2bit × 8K entry
	PC[14:2] xor ghr<<7 による選択
RET-AS	8entry
物理レジスタ	32entry
ストアバッファ	8entry
キャッシュライン	64byte
I1 キャッシュ	4way, 16KB, ミス時 10cycle
D1 キャッシュ	4way, 16KB, ミス時 10cycle
L2 キャッシュ	4way, 2MB, ミス時 100cycle

グできない．演算器は 1 種類につき 1 個しかないので，命令が完了するスピードは変わらないように見えるが，実は後続命令を機会が減っており IPC 低下の要因と考えられる．

もう 1 つは OP1 がキャッシュミスしたときの挙動である．ARM-SS モデルでは，OP1 に命令を投入できなくなるものの，他の演算器へは発行可能である．一方，OROCHI モデルでは 2 個目の OP1 命令と同じタイミングにスケジューリングされた命令の発行もできなくなる．これについては OP1 の演算器入力バッファをキュー構造にすることで対応可能であると考えられる．より深刻なのは，キャッシュミスしたロード命令の結果を必要とする命令と同じタイミングにスケジューリングされた命令が発行できなくなることである．この問題を解決するにはスケジューリング時に何らかの方法でキャッシュミスを予測する必要がある．

4.2 ARM-SS モデルの遅延時間

論理合成ツール Synopsys 社の Design Compiler 2006.06-SP2 を用い，CMOS セルライブラリに日立 0.18 μ m を使用して，Verilog-HDL で記述した ARM-SS モデルの合成を行い，遅延時間の評価を行った．測定結果を表 2 に示す．遅延制約の設定は文献[10] を参考に，全体の合成は動作周波数 1ns を目標値とし，平坦化を行って速度方向へ合成をするように指定した．なお個々のモジュールの遅延時間は参考値である．set_max_delay 0 all_outputs() を指定して合成しており，その結果を全体の合成には使用していない．

SEL/RD ステージにおいてクリティカルパスとなっているのは，命令ウィンドウの物理レジスタ番号から命令発行判定機構，命令発行機構，命令ウィンドウ読み出し，物理レジスタ読み出し，分岐実行を経て分岐実行結果の書き込みへ至る経路で，5.36ns という遅延時間になっている．現在のパイプライン構成では BRANCH を SEL/RD ステージにて実行しており，クリティカルパスになっている．BRNACH を別ステージとした場合，SEL/RD ステージの遅延時間は 4.95ns 程度になると考えられる．

4.3 OROCHI モデルの遅延時間の概算

ARM-SS の Verilog モデルの合成結果から，提案モデルの遅延時間の概算値を求める．OROCHI モデル遅延時間を概算するにあたり，クリティカルパスは MAP/SCH ステージではなく DISP/RD ステージにあると仮定した．マップテーブルの遅延時間は 1.77ns となっており，これにスケジューリング機構の遅延時間を加えても DISP/RD ステージと同じ程度の遅延時間に収まると考えられる．DISP/RD ステージでは，命令発行判定機構，ソースレジスタの読み出し，分岐実行，分岐予測結果の書き込みの経路がクリティカルパ

表 2 ARM-SS モデル 各モジュールの遅延時間

モジュール	遅延時間 [ns]
命令発行判定機構 (ISSUE_CHK)	0.92
命令選択機構 (SELECT)	0.83
命令ウィンドウ読み出し (PREGS_TAG)	1.44
物理レジスタ読み出し (PREGS_VAL_DRV)	1.76
分岐実行 (BRANCH)	(1.18)
分岐結果書き込み (PREGS_TAG)	(0.56)
合計	(6.69)
合計 (BRANCH を別ステージで処理)	4.95
全体 (平坦化後)	5.36

表 3 OROCHI モデルの遅延時間 (概算値)

モジュール	遅延時間 [ns]
命令発行判定機構 (ISSUE_CHK)	0.92
物理レジスタ読み出し (PREGS_VAL_DRV)	1.76
分岐実行 (BRANCH)	(1.18)
分岐結果書き込み (PREGS_TAG)	(0.56)
合計 (概算値)	(4.42)
合計 (BRANCH を別ステージで処理)	2.68

表 4 IPC と遅延時間による総合評価

モデル	IPC	Delay[ns]	IPC× freqency[MHz]
ARM-SS	1.335	5.36	249.1
OROCHI(概算値)	1.331	4.42	301.2

スになると思われるため，表 3 に示した概算値 4.95ns が得られる．なお，ARM-SS と同様に BRANCH を別ステージとすると，DISP/RD ステージの遅延時間は短くなり，2.68ns 程度になると思われる．また，OROCHI 用の命令発行判定機構は ARM-SS 用に比べて簡単になるため，このモジュールの遅延時間は短くなる．さらに，全体の合成時に平坦化を指定することで，遅延時間の改善が可能である．

4.4 総合評価

概算した最大動作周波数と IPC の平均値との積を求めた結果を表 4 に示す．この結果から OROCHI モデルは ARM-SS モデルより 21%程度性能向上していることがわかる．クリティカルパスを短くするために BRANCH を別ステージとすると，OROCHI モデルの遅延時間はさらに短くなり，ARM-SS モデルとの性能差は大きくなる．しかし IPC が低下するため IPC× 周波数が一概に良くなるとは言えない．

5. あとがき

本稿では VLIW 型命令キューによって命令レベル並列処理を実現する方法について述べ，これを利用した OROCHI プロセッサのバックエンドの構成方式を示した．集中命令ウィンドウ型の ARM-SS モデルと OROCHI モデルのパイプラインシミュレータに MiBench を走行させた結果，OROCHI モデルの IPC が平均して 0.3%程度の低下に抑えられることがわかった．また ARM-SS の Verilog モデルを論理合成し遅延時間を求め，その結果を元に OROCHI モデルの遅延時間の概算し，性能 (IPC× 動作周波数) 求めたところ，21%程度性能を向上できることがわかった．

本稿の OROCHI モデルの遅延時間は概算値に留まっている．また OROCHI モデルのスケジューリング機構の遅延時間が，命令発行判定機構と物理レジスタ読み出しより短いと仮定している．今後，OROCHI モデルを Verilog-HDL を用いて実装して論理合成を行い，正確な遅延時間を評価する必要がある．また命令発行ステージ (ARM-SS モデル:SEL/RD ステージ,OROCHI モデル:DISP/RD ステージ) にて実行している BRANCH を別ステージに移動させることにより，遅延時間を短かくできることが分かっている．しかしその結果 IPC は低下すると思われる．今後，この点についても評価していく必要がある．

6. 謝　辞

本研究は，半導体理工学研究センターとの共同研究によるものである．加えて，東京大学 VDEC を通じて，Synopsys 社，Cadence 社，日立製作所の協力で行われたものである．

参考文献

1) 島田 貴史, 田端 猛一, 北村 俊明, "異種命令セット同時実行プロセッサ OROCHI の構成", 情処研報 2006-ARC-170,pp.55-60 (2006)
2) ARM Architecture Reference Manual, ARM Limited, ARM DDI E (2000)
3) FR550 Series Instruction Set Manual Ver.1.1, 富士通株式会社 (2002)
4) David B. Papworth, "Tuning the Pentium Pro microarchitecture", IEEE Micro, Vol.16,No.2,pp.8-15 (1996)
5) L. Gwennap, "Intel's P6 Uses Decoupled Superscalar Design", Microprocessor Report, Vol.9,Np.2,pp.9-15 (1995)
6) Glenn Hinton, Dave Sager, Mike Upton, Darrell Boggs, Doug Carmean, Alan Kyker, Patrice Roussel, "The Microarchitecture of the Pentium4 Processor", Intel Technology Journal,Q1 (2001)
7) Kenneth C. Yeager, "The MIPS R10000 Superscalar Microprocessor", IEEE Micro, Vol.16,No.2,pp.28-40 (1996)
8) 中島 康彦, "ARM アーキテクチャ向け命令分解型スーパースカラ", 情処研報 2006-ARC-168,pp.77-82 (2006)
9) 小島 和也, 中島 康彦, "OROCHI 評価用集中命令ウィンドウ型スーパースカラの設計", 情処研報 2006-ARC-170,pp.61-66 (2006)
10) STARC, "STARC RTL 設計スタイルガイド Verilog HDL 版 第 2 版", STARC (2006)