2007－OS－106 (3)
2007／8／3

異種命令セットを同時に実行するマルチスレッディング・プロセッサの構成

須賀圭一[†] 山原幹雄[†]
中田尚[†] 中島康彦[†]

近年，携帯情報機器に対しても高度なマルチメディア処理が要求されてきており，性能向上のために専用プロセッサが搭載されている．しかし，複数プロセッサの搭載は消費電力の増大を招くため，組み込み機器として用いる場合は問題になる．そこで我々は，既存のマルチスレッド実行を拡張して，複数アーキテクチャを同時実行するOROCHIプロセッサを提案している．ただし，本プロセッサを厳密に性能評価するためにはOSの搭載が必要であり，入出力や主記憶など周辺機能を装備しなければならない．本論文では，異種命令セットを同時実行するプロセッサを想定し，OSが稼働する実験環境構築のために必要な検討課題と解決策について報告する．また，予備的評価として，クロックアキュレートなシミュレータを用いてOSを動作させ，ソフトウェア割り込みの処理形態に応じた挙動について評価を行った．

An SMT Processor for Simultaneous Execution of Heterogeneous Instructions

KEIICHI SUKA,[†] MIKIO YAMAHARA,[†] TAKASHI NAKADA[†]
and YASUHIKO NAKASHIMA[†]

Recently, mobile devices and embedded equipments are required to execute multimedia programs which have much IPL. Equipped with application specific processors, high performance can be achieved. However, this solution leads to power consumption problem. Thus we proposed a heterogeneous SMT processor OROCHI, which can support multiple instruction sets simultaneity. To estimate the processor performance under working OS code, we developed the experimental circumstance which has the peripheral, I/O and main memory. In this paper, we discuss the OS environment for an SMT processor executable with multiple instruction sets. As a consequence, we estimate cache performance under working OS code using clock accurate simulator.

1. はじめに

近年，携帯電話などの組み込み機器では，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を行うプログラムが同時に実行される環境が一般的になっている．例えば，高機能化した最近の携帯電話では，動画再生中でも電話を受けられるよう，複数のプログラムが同時に実行できることが要求されている．このような状況において，最近では，汎用プロセッサコアの他に，用途に応じた複数種類のDSP(Digital Signal Processor)コアを併置するような構成をとるプロセッサが登場してきている．例えばMP211[1]では3個のARM926コアとDSPコアを集積した構成となっている．また，Cell Broadband Engine[2]では，汎用プロセッサコアとしてのPowerアーキテクチャのPPE(PowerPC Processor Element)と単精度浮動小数点系演算を行うSPE(Synergistic Processing Element)を混載している．

このような構成のプロセッサが登場してきたのは，OSを含む様々なソフトウェア資産を有効活用しつつ，命令レベル並列度が高い一部のアプリケーションについては専用ハードウェアを活用して高速化と低電力化を図るためである．一般に，マルチメディア処理には高い命令レベル並列度が内在しており，複雑な命令発行機構により並列実行可能な命令を検出するスーパスカラ方式よりも，コンパイラ等によりあらかじめ命令スケジューリングを行うVLIW（Very Long Instruction Word）方式のほうがハードウェア機構を簡素化できるため，電力性能比が優れていると言われている．例えばFR1000[3]では4つのVLIW型プロセッサコアを集積することによって，低消費電力かつ高い処理能力を実現している．ただし，スーパスカラには過去のソフトウェア資産を利用できる利点があるのに対し，VLIW型のプロセッサにはソフトウェア資産が少ない欠点があることは言うまでもない．ソフトウェア資産の活用と低消費電力を兼ね備えるために，種類の異

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

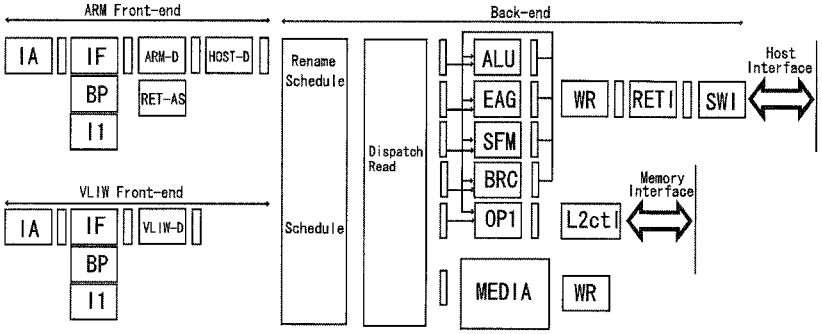

図 1 OROCHI のパイプラインモデル

なるコアを並置することは必然であると言える．

しかし，コアの単純な並置はプロセッサ全体の回路規模の増大を招くため，できるだけ資源を共有して全体の回路規模を抑えたい．このような動機に基づき，我々は，異種命令セットを同時実行する OROCHI プロセッサ[4)~6)] を提案している．目的は，ヘテロジニアス構成のマルチコアよりも小さな回路規模により同等の全体性能を達成することにあり，評価手段として，FPGA による実装および LSI 試作を考えている．ところで，異種命令セットを同時に実行するプロセッサが実環境下においてどのような振舞を示し，どれだけの性能向上が達成できるか，また，性能向上が不十分である場合にどのような問題点があるかを評価・検証するには，実運用状態を反映した評価環境が必要である．特に，OS の搭載は必須であると考えるものの，実システムと全く同じ実験環境を構築するには極めて多くの時間と労力を必要とする．本報告では，OROCHI を詳細に評価するために，どのような検討課題があり，コストをかけずにどのように解決するかについて，特に，必要な周辺機構をいかに構築するかについて述べる．引続き第 2 節では OROCHI の構成について概観する．第 3 節では評価に際しての課題について，第 4 節では解決方法として，特に OS 搭載のための機構，外部キャッシュ機構，ホスト連携機構について述べる．第 5 節では，パイプラインシミュレータを用いて，OS を搭載した場合と搭載しない場合のアプリケーションの挙動の違いについて初期評価を行う．

2. OROCHI の概要

本節では，OROCHI プロセッサの概要について述べる．OROCHI のパイプラインモデルを図 1 に示す．OROCHI は，種類の異なる命令セットを同時に実行できるよう，一般的な SMT を拡張したプロセッサモデルである．汎用命令セットにより記述されたソフトウェア資産をスーパスカラ方式により高速実行する部分と，マルチメディア処理用の命令セットに特化し，

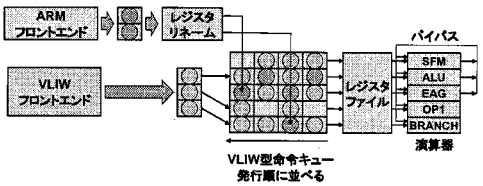

図 2 混在実行の様子

コンパイラ等によりスケジューリングが完了している命令列を VLIW 方式により効率良く実行する部分から構成される．レジスタと演算器からなるバックエンド部分を共有することにより全体の回路規模の縮小を図るとともに，レジスタリネーミング機構やリタイア機構はスーパスカラ部分のみに装備し，マルチメディア命令を実行する際には，パイプライン段数の少ない VLIW 部分のみの稼働により消費電力の抑制を図っている．

さて，命令フェッチからレジスタ読み出しの直前までは，汎用命令セット（以後，ARM[7)] を仮定）とマルチメディア処理用命令セット（以後，FR-V[8)] を仮定）のそれぞれに対してパイプラインステージが必要である．バックエンドについては，先行研究[4)] において，VLIW 型命令キューをもつ命令スケジューリング機構の検討を進めており，リザベーションステーション方式のバックエンドよりも高い性能を発揮できる見通しを得ている．なお，VLIW 型命令キューでは，命令列が効率良く流れるよう，1 サイクルにより実行可能な単純な命令（ALU 演算，シフト演算，単純なロード/ストア演算）のみを取り扱う必要がある．一方，ARM にはマルチプルロード命令やシフト機能つき第 2 オペランドをはじめとする複雑な命令があるため，このような命令キューにより混在実行するためには，単純な命令に分解する[9),10)] 必要がある．このため，各パイプラインステージは次のような構成となる．IA ステージではプログラムカウンタ（以後，PC と略す）を生成する．IF ステージでは命令キャッシュ（I1-Cache）

を読み出すと同時に分岐予測を行う（BP）．その後，ARM-D，HOST-D ステージにおいて命令分解を行う．同様に VLIW 型命令に対しても IA および IF ステージを設け，VLIW-D において VLIW 型命令のデコードを行う．演算器は 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行う EAG，ロードストアを実行する OP1，分岐命令を実行する BRC，マルチメディア演算用の MEDIA から構成される．

バックエンドは依存関係の解消，命令スケジューリングを行う Rename/Schedule ステージ，命令発行を行う Dispatch/Read ステージ，各種演算を行う Execute ステージからなる．ただし，VLIW 型命令についてはリネームは行わず，そのまま命令キューに格納する．同様にリタイアステージも備えていない．異種命令を混在実行する様子を図 2 に示す．VLIW 型命令は 1 語中に含まれる複数語の命令が同時に命令キューの同一列に格納され，演算器の入力バッファのエントリーに向かってシフトされる．単純な内部命令に分解された ARM 命令は VLIW 型命令キューの空き部分に投入され混在実行される．本実行方式により，演算器の使用効率向上を目指している．

3. 評価環境構築の課題と解決方法

さて，冒頭に述べたように，OROCHI を定量的に評価するためには，実運用状態を反映した評価環境が必要であるものの，環境構築に際しては次に挙げる課題がある．

OS 搭載の必要性　プロセッサアーキテクチャ研究に一般的に用いられるソフトウェアシミュレータでは OS 機能をホストがエミュレートする方式が一般的である．比較のために，OS がない環境でもアプリケーションが動作する仕組みは備えたい．しかし，既存ソフトウェア資産の代表は OS であり，前者と比較するためにも OS の搭載は必須である．

ハードウェア量削減の必要性　最近の FPGA は大容量化が著しいとは言え，スーパスカラコアや VLIW コアを全て収容し，さらに二次キャッシュまで搭載できるほどの余裕はない．二次キャッシュ機能は FPGA の外に配置するとしても，相対的に低速過ぎると性能評価にならない．

開発コスト削減の必要性　OS を搭載するために実環境と同様のシステムを構築しようとすると，プロセッサの他に，外部キャッシュ，主記憶，外部記憶，I/O 装置，タイマ，割り込みなどを一通り装備する必要があり，まともに取り組むと極めて多くの時間と労力を必要とする．

簡便なデバック環境の必要性　仮に独立したシステムとして構築できたとしても，内部状態の確認やデバッグに極めて多くの時間と労力を必要とする．主目的がプロセッサの性能評価であるのに，これでは本末転倒である．システムの状態を可能な限り，かつ，単純な機構により観測可能としたい．

次に，以上の課題を解決する方法について述べる．基本的アイデアは，従来のソフトウェアシミュレータと同様，アプリケーションプログラムはホストの主記憶上に展開しつつ，外部キャッシュやプロセッサコアは切り離して，ホストの PCI 空間に写像した FPGA ボード上に搭載する点にある．このような構成とすることにより，当初はクロックアキュレートなパイプラインシミュレータとしてスタートし，OS を含むシステム全体の動作を確認した後にプロセッサ部分のみを FPGA へ移行するインクリメンタルな環境構築が可能となる．

3.1 搭載する OS および搭載時/非搭載時の切替え

現在，組み込み機器はますます高性能化してきており，高機能な汎用 OS の採用が必須となってきている．組み込み OS としては，ITRON，Windows CE，VxWORKS[11] などがあるが，本稿では，MMU を持たない環境を想定し，μClinux(2.4.32)[12] を用いた．OS 搭載時に必要な機能は，コンパイラが生成する一般命令の他に，制御命令，制御レジスタ，割り込み機能，入出力機能，タイマー機能である．なお，OS 搭載時と非搭載時とで動作を変更する必要があるプロセッサ機能はシステムコール命令（ARM では SWI 命令）である．OS 搭載時は SWI 命令がリタイアする際にベクタ割り込みを発生する．一方，非搭載時はプロセッサの動作を中断してホストがエミュレートする．ただし後述するように，エミュレートの際には内蔵キャッシュのダーティラインに関して特別な考慮が必要である．

3.2 記憶階層と I/O の実現

ハードウェア量削減のために内蔵キャッシュを含むプロセッサコアは FPGA 上に配置し，外部キャッシュは同一ボード上の専用メモリ上に配置するとしても，ホストの外部記憶や I/O 機能を利用して主記憶とのデータ転送を行うためには，プロセッサの主記憶をホストの主記憶空間上に写像しておくのが都合がよい．特に，システムコールをホストがエミュレーションする場合，内蔵キャッシュのダーティラインを追い出す必要がある以外は，プロセッサ側のアドレスをホスト側のアドレスに変換してホスト上でシステムコールを発行するだけでよい．すなわち，ソフトウェアシミュレータにおける一般的な仕組みをそのまま用いることができる．同様に，OS 搭載時に必要となる I/O については，主記憶上にノンキャッシャブル領域を設定し，ホストの I/O 装置をそのままプロセッサに見せることにより容易に実現することができる．以上により，FPGA 上に必要となるハードウェア量を削減できるのみならず，評価システム全体の開発コストを削減することもできる．

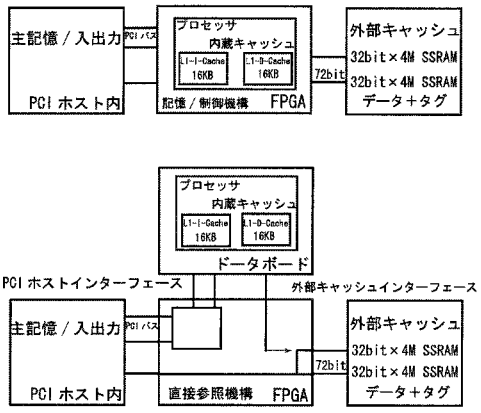

図 3 論理インターフェース

3.3 ホストとの連携による内部の観測

外部キャッシュから外の部分をホスト上に配置することにより，ホスト側からの状態観測も容易になる．ただし，システムコールには，主記憶の内容だけでなくレジスタ値の参照および更新が必要となることから，ホストからプロセッサ内部のレジスタ値を読み書きする仕組みの追加が必要である．ただし，レジスタ値の読み書きはテストのために必要であることから，特にシステムコール専用のインタフェースを追加することなく既存の仕組みを利用して実現することができる．

4. 評価環境の詳細

本節では，以上の考えに基づき詳細化した評価環境について述べる．

4.1 全体の構成

図 3 に全体構成を示す．上側は内蔵キャッシュを含むプロセッサ機能を FPGA 上に実装する形態，下側はプロセッサ機能を専用 LSI パッケージに搭載する形態を示している．いずれの場合も，ホスト上の主記憶および I/O 装置を利用する．外部キャッシュは FPGA ボード上に搭載した SSRAM 上に構築する．ホストは PCI 空間上に写像された FPGA 内空間を通じて外部キャッシュおよびプロセッサ内レジスタを参照する．図 4 にブロック図，図 5 に外観を示す．各モジュールは次の通りである．

CPLD FPGA のコンフィグレーション制御に用いる．

SSRAM 36bit × 1M word の ZBT-SSRAM を計 8 個搭載しており，36bit × 4M word × 2 組の外部メモリとして使用できる．この上に外部キャッシュタグおよびキャッシュラインを配置する．

PCI コントローラ FPGA とホストを PCI ローカルブリッジにより接続し，ホストの主記憶空間に SSRAM を対応付けることにより，ホストが直接 SSRAM の内容を参照/変更できるようにする．

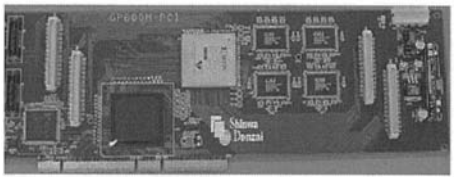

図 5 FPGA ボードの外観

表 1 記憶階層の構成

Capacity	I1-Cache	16KB
	D1-Cache	16KB
	L2-Cache	16MB
Way	I1-Cache	4
	D1-Cache	4
	L2-Cache	1
Line Size	I1-Cache	64B
	D1-Cache	64B
	L2-Cache	64B
BlockRAM Size	I1-Cache	32 Blocks
	D1-Cache	32 Blocks

ドータボード 専用 LSI を搭載する際に用いる．合計 4 個の高密度コネクタにより FPGA ボードと接続する．

4.2 一次キャッシュ

本節では，前述した課題の解決方法のうち，特に複雑な記憶階層の構成と動作について述べる．命令用とデータ用に独立した一次キャッシュ（I1, D1）は FPGA 内に配置される内蔵キャッシュに対応する．表 1 に記憶階層の構成を示す．一次キャッシュは FPGA 内の BlockRAM を利用して構築する．まず，図 6 にロード命令の動作概要を示す．

(a) キャッシュにヒットした場合，一次キャッシュからデータをそのまま読み込む．

(b) キャッシュミスが検出された場合，入れ替え対象ラインが way の中から LRU アルゴリズムによって決定され，もし二次キャッシュとの不整合を示すダーティビットがセットされている場合は，ダーティラインの追い出しが行われる．

(c) 追い出しが完了するかダーティビットがセットされていない場合は，有効ラインの読み出し要求が外部キャッシュインターフェースに送出される．外部キャッシュから到着した有効ラインは，内蔵キャッシュの入れ替え対象ラインに書き込まれる．

次に，図 7 にストア命令の動作概要を示す．内蔵キャッシュへの書き込みが発生するのはストア命令がリタイアする時である．本プロセッサモデルでは，8 エントリ分のストアバッファを用意しており，ストアすべきアドレスおよびデータがリタイアまでの間，一時的に保存されている．

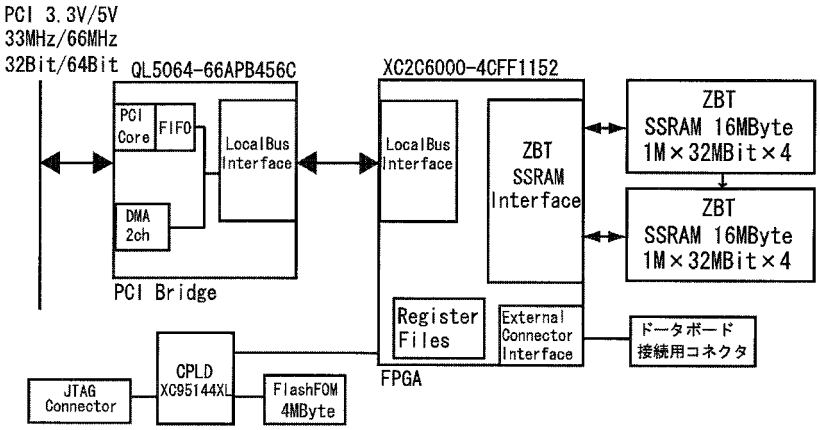

図 4 FPGA ボードのブロック図

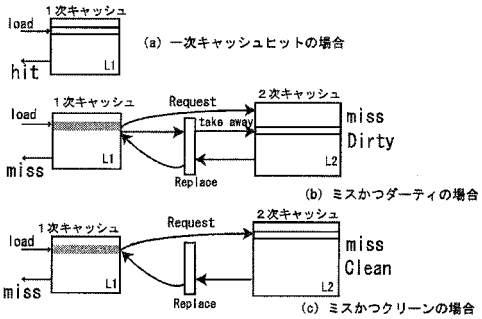

図 6 一次キャッシュにおけるロード命令の動作

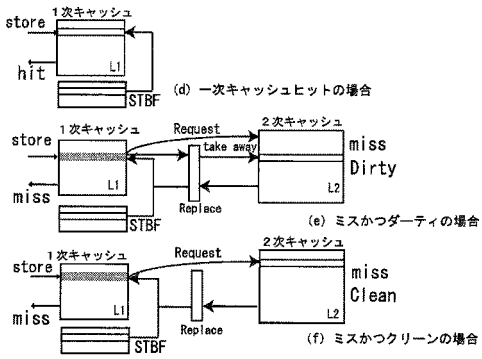

図 7 一次キャッシュにおけるストア命令の動作

(d) 一次キャッシュへの書き込み時にキャッシュヒットした場合は，ストアバッファの先頭エントリが内蔵キャッシュに書き込まれる．このとき，前述したダーティビットがセットされる．

(e) キャッシュミスが検出された場合，ロードの場合と同様，入れ替え対象ラインが way の中から LRU アルゴリズムによって決定され，もし二次キャッシュとの不整合を示すダーティビットがセットされている場合は，ダーティラインの追い出しが行われる．

(f) 追い出しが完了するかダーティビットがセットされていない場合は，ロードの場合と同様，有効ラインの読み出し要求が外部キャッシュインターフェースに送出される．外部キャッシュから到着した有効ラインとストアバッファのエントリが，内蔵キャッシュの入れ替え対象ラインに書き込まれる．

4.3 二次キャッシュと主記憶

外部キャッシュは命令とデータを共用するユニファイドキャッシュ（L2）であり，物理アドレスを用いたダイレクトマップ方式としている．SSRAM 空間はホストから直接参照/更新することができるので，もし FPGA の容量に余裕がない場合や，タイミング依存による異常な挙動を応急処置により収束させたい場合には，一次キャッシュから二次キャッシュへの追い出しや読み込み操作をソフトウェアとしてホスト側に実装することもできる．一方，主記憶はホストの管理下にあるため，主記憶と二次キャッシュ間の転送はホスト側に実装される．このような代替手段を設けることにより，評価環境の開発コスト削減を図っている．

5. 現状および初期評価

前述のように，これまでにインクリメンタルな環境構築を行ってきており，現在，クロックアキュレートなパイプラインシミュレータ上で，OS 搭載時と非搭載時について，ARM 命令に基づくアプリケーションプログラムが動作する状況にある．μClinux(2.4.32) の有無によって，アプリケーションの挙動がどのように

表 2 ミス率の比較結果 (単位%)

	OS	L2 ミス回数 (%)	I1 ミス	D1 ミス	SB ヒット
Queens	有	2768 (29.4)	2.1	4.6	0.9
	無	35 (94.6)	0.2	0.2	9.5
FFT	有	2746 (28.3)	0.7	3.1	0.6
	無	167 (96.5)	0.0	0.2	1.7
Intmm	有	2733 (25.4)	0.6	2.0	0.1
	無	327 (99.2)	0.0	0.1	0.0
Quick	有	2765 (24.8)	0.4	2.4	0.3
	無	349 (53.9)	0.0	0.3	0.1
Perm	有	2799 (28.7)	0.3	0.6	0.0
	無	24 (85.7)	0.0	0.0	0.0
Bubble	有	2809 (29.3)	0.4	1.2	12.5
	無	53 (91.4)	0.0	0.0	12.6
Towers	有	2757 (28.4)	0.3	0.5	0.1
	無	31 (79.5)	0.0	0.0	0.0
Mm	有	2844 (19.2)	0.1	2.3	0.1
	無	720 (96.8)	0.0	0.6	0.0
Puzzle	有	2820 (8.0)	0.1	2.7	0.0
	無	495 (3.3)	0.0	1.4	0.0
Trees	有	23528 (15.0)	0.2	3.1	0.3
	無	43161 (65.2)	0.0	10.3	0.0

変化するかを見るために，本環境を用いて初期評価を行った．各々の環境において，Stanford Benchmark を実行し，前述したキャッシュに関する挙動を調査した．OS を含む全てのロードモジュールは，gcc-4.1.1 (march=armv4 -msoft-float -O2)，binutils-2.17 により生成した．結果を表 2 に示す．なお，SB hit は Store Buffer にヒットした率を表している．

OS 搭載時と非搭載時では，L2 のミス率に変化が見られた．OS 非搭載時では二次キャッシュへのアクセス回数が少なく，キャッシュが有効に働く前にアプリケーションは終了してしまう．このため，OS 搭載時に比べてミス率が大きくなっている．一方，OS 搭載時は，タイマー割り込みなどの命令列が定期的に実行され，二次キャッシュアクセス回数自体が多いため前述したミス率は低く見える．逆に，I1-cache，D1-cache のミス率に関しては，OS 搭載時の方がヒット率が低い．これは割り込み処理などの命令列が実行される際にキャッシュラインが追い出されるためと考えられる．このように，OS 搭載時と非搭載時ではキャッシュの挙動が大きく変わることがわかった．すなわち、異種命令セットを混在実行する OROCHI においては，この様な状況においていかにキャッシュヒット率を向上させるかが大きな課題であるといえる．

6. おわりに

本稿では，一般的な SMT を拡張して，複数アーキテクチャを同時実行する OROCHI プロセッサについての構成を述べ，さらにプロセッサ試作のための実験環境，および，キャッシュ構成について述べた．また，現況のパイプラインシミュレータを用いてシステムを初期評価し，キャッシュの挙動について調査した．現在，OROCHI プロセッサ，および，比較対象とするスーパスカラプロセッサの Verilog モデルを設計，実装している．今後，これらの実験環境を用いて，さらに評価を進めていく予定である．

7. 謝　辞

本研究の一部は科学研究費補助金（基盤研究 (B) 課題番号 19300012），および，半導体理工学研究センターとの共同研究による．

参考文献

1) S.Torii, et al, "A 600MIPS 120mW 70uA Leakage Triple-CPU Mobile Application Processor Chip", ISSCC, pp.136-137, (2005)
2) Dac Pham, et al , "The Design and Implementation of a First Generation CELL Processor", ISSCC, p.184- 592 Vol. 1 (2005)
3) Shiota et al, "A 51.2GOPS, 1.0GB/s-DMA Single-Chip Multi-Processor Integrating Quadruple 8-Way VLIW Processor". ISSCC, 2005, p.18-19.
4) 片岡 晶人, 中西 正樹, 山下 茂, 中島 康彦, "VLIW 型命令キューを持つ OROCHI の命令スケジューリング機構" 情処研報 2007-ARC-172, p.25-30, Mar. (2007)
5) 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, p.55-60 (2006)
6) Hajime Shimada, Takashi Shimada, Takekazu Tabata, Tomoya Kojima, Kenji Kise, Yasuhiko Nakashima, Toshiaki Kitamura:"Outline of OROCHI: A Multiple Instruction Set Executable SMT Processor", IWIA (2007)
7) ARM Architecture Reference Manual, ARM Limited, ARM DDI E (2000)
8) FR550 Series Instruction Set Manual Ver.1.1, 富士通株式会社 (2002)
9) 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, p.77-82 (2006)
10) 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, p.61-66 (2006)
11) VxWORKS, http://www.windriver.com/
12) μClinux, http://www.uclinux.org/