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

2008－ARC－179 (1)
2008／8／5

異種命令SMTプロセッサOROCHIの実装と分析

吉村和浩[†1] 中田尚[†1] 中島康彦[†1]

我々は，2つの異なる命令セットに各々対応するフロントエンド部と，共通のバックエンド部を備える，異種命令混在実行プロセッサOROCHIを提案している．ソフトウェアシミュレータによるアイデアの検証，FPGAによる実現性の検証を経て，現在，ASICによる遅延時間の検証を行っている．本報告では，シミュレータの観点からC言語とVerilogの記述量および実行速度を比較し，また，実現性の観点からFPGA（Xilinx XC2V8000）とASIC（0.25μm）の回路規模および動作周波数を比較した．分岐予測表とキャッシュに内蔵メモリマクロを用い，全体をVerilog-HDLにより記述した場合，まず，FPGAとASICでは，乗算機能を除いてクリティカルパスの傾向に大きな差異は見られないことがわかった．また，FPGAとASICいずれの実現方法においても，OROCHIは，2種類のコアを単純並置する構成の75%の回路規模で済むことがわかった．

Implimentation and Analysis of a Heterogeneous SMT Processor

Kazuhiro Yoshimurai,[†1] Takashi Nakada[†1]
and Yasuhiko Nakashima[†1]

We have proposed a heterogeneous SMT processor OROCHI that has two frontend pipelines and a common backend pipeline to execute multiple instruction sets simultaneously. In the development of OROCHI, we have evaluated our new ideas by a software simulation first, and the processor realizability on FPGA. Now, we are evaluating the delay time on ASIC. In this paper, we compare lineages and execution speeds of C source code and Verilog-HDL source code of OROCHI from viewpoint of the simulatior. And we compare the chip area and clock frequency of FPGA (Xilinx XC2V8000) and ASIC (0.25μm) from viewpoint of the realizability. We design the logic circuit with Verilog-HDL using memory macro for PHTs (Pattern History Table) and first level caches. The result shows that there are no difference in the tendency of a critical path expect for multipliers. And the chip area of OROCHI is only 75% of a multi-core that has an ARM core and an FR-V core.

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．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[2)]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．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

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

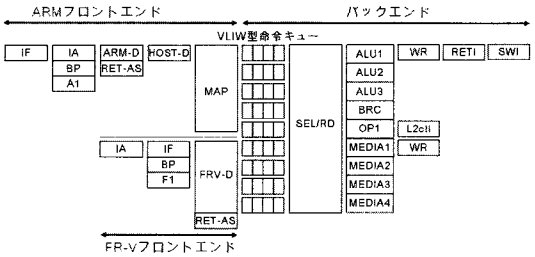

図 1 OROCHI のパイプライン

種命令セットを同時実行するプロセッサ OROCHI[3] 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 試作を目指している．

OROCHI の開発ではソフトウェアシミュレーション，RTL シミュレーション，FPGA 実機検証という段階を経て，現在，ASIC 試作を行っている．本報告では，各パイプラインステージに注目し，C 言語（以降，C と略す）で記述したソフトウェアシミュレーションと Verilog-HDL（以降，HDL と略す）で記述した RTL ソースの記述量とシミュレーション実行速度について分析し，FPGA と ASIC それぞれを対象とした論理合成の結果より遅延時間と回路規模について評価を行う．以降，第 2 章では，OROCHI のパイプラインモデルと実装モデルについて述べる．第 3 章では，C と HDL の記述量とシミュレーション実行速度について述べ，第 4 章では，OROCHI の FPGA と ASIC を対象とした論理合成の結果より遅延時間と回路規模について述べる．第 5 章では，まとめと今後の予定を述べる．

2. OROCHI の概要

OROCHI は異なる命令セットを同時に実行できるよう，一般的な SMT プロセッサを拡張したプロセッサモデルである．OROCHI は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，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．OROCHI では汎用命令セットに ARM[4] を採用し，VLIW 命令セットに FR-V[5] を採用している．本章では，OROCHI の概要について述べる．

2.1 パイプラインモデル

OROCHI のパイプラインモデルを**図 1** に示す．VLIW 型命令キュー直前までは，それぞれの命令セットに対してパイプラインステージが必要であり，これらをそれぞれ ARM フロントエンドと FR-V フロントエンドと呼ぶ．バックエンドについては，これまでに，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行った結果，既存の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ことを報告している[6]．VLIW 型命令キューでは，命令列が効率良く流れるよう，1 サイクルで実行可能な単純な命令（ALU 演算，分岐，単純なロード/ストア演算）のみを取り扱う必要がある．しかし，ARM では，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，VLIW 型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[8] 必要がある．

このため，各フロントエンドは次のような構成となる．ARM-IA ステージではプログラムカウンタを生成する．ARM-IF ステージでは ARM 命令 1 次キャッシュ（A1-Cache）を読み出すと同時に分岐予測を行う（BP）．その後，ARM-D および HOST-D ステージにおいてデコードと命令分解を行う．また，ARM-D ステージにはリターンアドレススタック（RET-AS）を備えている．MAP ステージでは分解された ARM 命令の依存関係の解消と命令スケジューリングを行う．同様に，FRV-IA，FRV-IF ステージおよび FRV-D を設け，FR-V 命令 1 次キャッシュ（F1-Cache）から読み出した命令列をデコードする．

バックエンドは，命令発行を行う SEL/RD ステージ，算術論理演算を実行する ALU ステージ，ロードストアを実行する OP1 ステージ，分岐命令を実行する BRC ステージ，マルチメディア演算用の MEDIA ステージ，レジスタ書き込みを行う WR ステージ，アウトオブオーダ実行された ARM 命令を並び替える RETI ステージから構成される．ただし，FR-V 命令についてはリネームは行わず，そのまま VLIW 型命令キューに格納し，RETI ステージも備えていない．OP1 はデータ 1 次キャッシュ（D1-Cache）ミスが発生すると L2ctl ステージを介して，2 次キャッシュ（L2-Cache）へアクセスする．SWI ステージでは，システムコールの実行をプロセッサに通知する．また，ホスト PC からのコマンドを受け取りプロセッサを制御し，プロセッサ情報をホスト PC へ通知する．

異種命令を混在実行する様子を**図 2** に示す．FR-V 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．VLIW 型命令

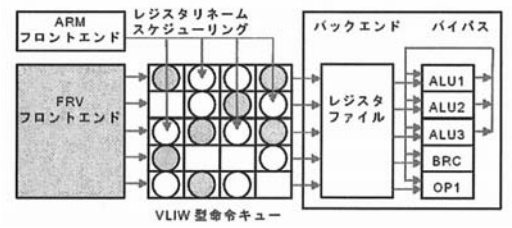

図 2 VLIW 型命令キュー

キューから発行された命令はレジスタの読み出し値または ALU からのバイパス値とともに演算器へ投入される．本実行方式により，演算器の使用効率向上を実現している[7]．

2.2 実装モデルと評価環境

新しいプロセッサを開発するにあたり，ソフトウェアシミュレーションによりシステム全体の動作を確認した後，主記憶，I/O 装置，バスなどプロセッサコア以外を準備することはたいへん手間のかかることであり，研究対象であるプロセッサコアをハードウェアで検証するまでに膨大な時間がかかるという問題がある．そこで OROCHI の開発では，段階的な評価環境[9] を用いている．OROCHI の実装モデルと評価環境を**図 3** に示す．(a) は内蔵キャッシュを含むプロセッサコアを FPGA 上に実装するモデル，(b) はプロセッサコアを ASIC 上に搭載するモデルを示している．いずれの場合も，主記憶および I/O 装置はホスト上のものを利用する．L2-Cache は FPGA ボード上に搭載した SSRAM 上に構築し，ホスト PC は PCI 空間上に写像された FPGA 内空間を通じて L2-Cache およびプロセッサ内レジスタを参照する．

OROCHI のソフトウェアシミュレータはホスト PC 上で主記憶および I/O 装置のエミュレートを行う．これにより，主記憶や I/O 装置などのハードウェアを用意することを省略できるため，プロセッサコアの評価に円滑に入ることができる．また，ASIC 試作後でも同じインターフェースで評価を行うことができる．

3. C 言語と Verilog-HDL の比較

OROCHI のソフトウェアシミュレータはパイプラインの動作を RTL レベルで記述したクロックアキュレートなパイプラインシミュレータであり，実際のプロセッサコアの動作を忠実にシミュレートすることができる．本章ではこのソフトウェアシミュレータと HDL により記述された RTL の比較について述べる．

3.1 記 述 量

各パイプラインステージの C と HDL の記述量を**表 1** に示す．実装される ALU は 3 個であるが，ここでは 1 個分の記述量を示している．同様に MEDIA も 4 個であるが，1 個分の記述量を示している．全ステージの合計では，HDL の記述量は C の 4 倍程度になっているが，ステージによってはその比率が大きく異なっているものがある．

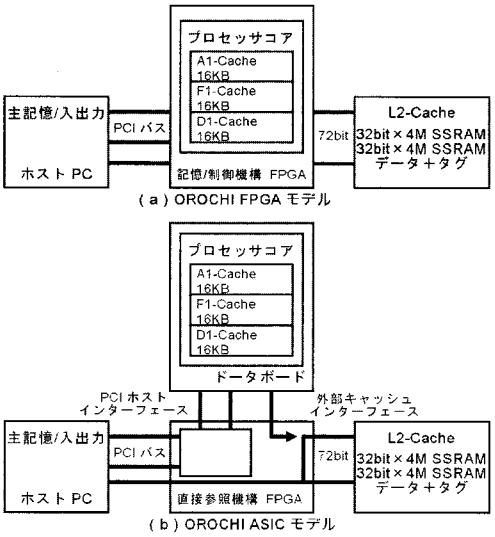

図 3 OROCHI の実装モデルと評価環境

フロントエンドでは，ARM-IA，FRV-IA はプログラムカウンタを生成する加算器が主な記述であるため，C，HDL ともに 150 行から 200 行程度であり，記述量は少ない．ARM-IF，FRV-IF はそれぞれ C では 300 行程度と他のステージと比べて記述量は少ないが HDL ではそれぞれ 1949 行，4658 行となり，他のステージより多い．これは，それぞれ A1-Cache，F1-Cache を持っており，L2-Cache との通信のためにステートマシンの記述が必要になるからである．また，ARM-IF は ARM 命令を同時 2 命令フェッチし，ARM-D へ最大 4 命令供給するのに対して，FRV-IF は FR-V 命令を同時 4 命令フェッチし，FRV-D へ最大 8 命令供給する．つまり，フェッチ命令数は記述量の比と等しい．ARM-D は C では 4502 行，HDL では 5063 行と記述量はほぼ等しい．これは，ARM 命令を単純な命令に分解するために C，HDL ともに if 文を多用して類似した記述が多いからである．一方，FRV-D は C では 1222 行に対して，HDL では 5295 行と多い．これは，FR-V 命令のデコードを行うために，C では 4 命令を for 文で処理しているのに対して，HDL では VLIW 命令の発行規則に従った 4 つのデコードモジュールを記述しているからである．MAP&RETI は C では 1599 行，HDL では 7703 行となり，他のステージより多い．これは，ARM 命令のアウトオブオーダ実行のために，レジスタリネーミング機構，VLIW 型命令キューの空きエントリのチェック，ポート競合の回避，リタイヤ機構の制御を行うモジュールを記述しているからである．

表 1　各パイプラインステージの記述規模と論理合成結果

	C	HDL	FPGA 33MHz				ASIC 制約なし			
			LUTs		BRAM	delay	Logic		Cache	delay
	[lines]	[lines]		[%]		[ns]		[%]		[ns]
ARM-IA	159	178	170	0.2	—	7.9	2674	0.3	—	1.7
ARM-IF	334	1949	1274	1.7	10	11.8	19533	2.4	80919	5.2
ARM-D	4502	5063	8060	10.9	—	20.3	73943	9.2	—	3.6
HOST-D	148	704	5483	7.4	—	12.4	57033	7.1	—	2.8
FRV-IA	159	214	119	0.2	—	6.5	2892	0.4	—	1.2
FRV-IF	326	4658	9133	12.0	10	28.7	74561	9.3	91393	6.4
FRV-D	1222	5295	4202	5.7	—	7.0	37284	4.7	—	3.1
MAP&RETI	1599	7703	23710	32.0	—	27.8	271811	34.0	—	5.8
SEL/RD&WR	2777	13285	12604	17.0	—	16.7	174947	21.9		3.8
ALU	429	1243	1665	2.2	—	2.6	29840	3.7	—	8.7
BRC	344	402	311	0.4	—	3.8	4012	0.5	—	1.6
OP1	1420	16751	6936	9.4	12	28.0	42878	5.4	102563	5.9
MEDIA	120	368	429	0.6	—	4.7	8528	1.1	—	2.8
Total	13539	57813	75383	100	32	—	799936	100	274875	—

表 2　シミュレーション実行環境と実行速度

	ソフト	RTL
CPU	Xeon 3.80GHz	Xeon 3.80GHz
Memory	2GB	6GB
OS	FreeBSD 6.2R	CentOS 4.6
平均実行速度 [kHz]	131	1.3

バックエンドでは，SEL/RD&WR は C では 2777 行に対して，HDL では 13285 行と記述量は多い．これは，C ではレジスタアクセスは単に配列参照であるのに対して，HDL では ARM の汎用レジスタと一時レジスタ，および，FRV の汎用レジスタとメディアレジスタの本体が記述してあり，さらにそれらのレジスタの読み出しと書き込みを行うモジュールを記述しているからである．ALU，BRC，MEDIA いった演算器は 1 サイクルで実行可能な命令のみを処理することと，C，HDL ともに算術演算子で主な記述ができるため他のステージと比べて記述量は少ない．OP1 は D1-Cache を持っているため，ARM-IF，FRV-IF と同様キャッシュを制御するためのステートマシンが大きいからである．それに加えて，C では配列へのアクセスとシフト演算子と論理演算子によりデータのマージを行っているのに対して，HDL ではロードストア値のマージを行うモジュールの記述が複雑である．

以上のことから，HDL ではステートマシンを持つモジュールや並列性の高いモジュールを持つステージで，C と比較して記述量が多くなっていることがわかる．

3.2 実行速度

一般的にプロセッサアーキテクチャに関する研究では SimpleScalar[10] などのソフトウェアシミュレータにより評価が行われている．しかし，アイデアの正確な性能や妥当性を評価するためには ASIC または FPGA に実装して評価しなければならず，そのためにはハードウェア記述言語で記述し，RTL シミュレーションを行わなければならない．また，アイデアが複雑になればソフトウェアシミュレーション，RTL シミュレーションともに実行時間が増大し問題となる．本節では OROCHI のシミュレーション実行速度について述べる．

RTL シミュレーションは Cadence 社の NC-Veirilog 05.10-p004 を使用する．ソフトウェアシミュレータからメモリイメージ，レジスタへの書き込み値と書き込みレジスタ番号，メモリへのストア値とストアアドレスをログとして出力し，それをテストベンチで読み込み，比較して検証を行う．ベンチマークには Stanford Benchmark の 10 個のベンチマークプログラムを ARM 向けにコンパイルしたものを使用している．コンパイルオプションは -msoft-float -O2 である．使用したツールは gcc-4.1.1，binutils-2.11（リンカのみ），binutils-2.17（リンカ以外）である．実行環境と Stanford Benchmark の平均実行速度を**表 2** に示す．ソフトウェアシミュレーションでは，平均実行速度は 131kHz となり，RTL シミュレーションの平均実行速度は 1.3kHz となる．ソフトウェアシミュレーションは RTL シミュレーションより 100 倍以上速いため，開発初期の様々なアイデアの検証ではソフトウェアシミュレータは有利であるが，プロセッサコアに搭載するアイデアと仕様が決定している開発中期では RTL シミュレーションでの検証に時間がかかることが問題となる．Stanford Benchmark は基本的な動作を確認するための小さなプログラムであるため，さらに大きなプログラムの検証にはそれ以上の時間がかかってしまう．そこで，FPGA での検証を円滑に行うことができれば，より大きなプログラムによる検証を高速に行うことができる．最終的には，OS の動作とステレオ画像処理などのマルチメディア処理を FPGA 上で実運用に近い形で検証することができる．

FPGA での検証では，シミュレーションのように

検証に必要な情報を容易に取得する柔軟性が必要である．OROCHIのソフトウェアシミュレータはプロセッサコアから，実行サイクル数，キャッシュヒット回数，キャッシュミス回数，レジスタやキャッシュの内容などのパフォーマンス情報を取得するコンソールとしても動作する．ソフトウェアシミュレータにこのような機能を付加することで開発の最終段階まで活用することができ，検証期間の短縮に大きな効果がある．パフォーマンス情報はプロセッサコアの検証だけでなく，ASIC試作後の評価にも使用することができる．また，開発者はソフトウェアシミュレーションからASICの評価に至るまで統一的なインターフェースで作業を行うことができる．

4. FPGAとASICの比較

本章ではOROCHIをFPGAおよびASIC向けに論理合成した結果について述べる．FPGA向けとしてXilinx社のXC2V8000を対象とし，Synplicity社のSynplify Pro 9.2を使用して動作周波数を33MHzに設定して論理合成を実行した結果を示す．また，ASIC（0.25μm）を対象にし，Synopsys社のDesign Compiler 2007.03-SP4を使用して制約なしに設定し，できるだけ高い動作周波数を達成することを期待して論理合成を実行した．論理合成の結果を表1に示す．

4.1 回路規模

表1のLUTsはFPGAのロジックのみの規模を表しており，キャッシュはXilinx社FPGAのBlock Select RAM（BRAM）で構成されている．Logicはロジックのみの規模であり，Cacheはキャッシュのみの規模を表している．単位はともに1NANDゲートである．すなわち，これらの合計がコア全体の回路規模となる．1次キャッシュは当初の設計では，A1-Cache，F1-Cache，D1-Cacheそれぞれ16KBの4wayセットアソシエイティブキャッシュとなっていた．しかし，ASICでは，メモリ幅は最大64bitであり，8個つなげ1wayを構成するとデッドスペースが大きくなり論理をすべて実装することができない問題が起きたため，キャッシュラインは512bit幅のまま，A1-Cache，F1-Cache，D1-Cacheの読み書きをそれぞれ64bit幅，256bit幅，128bit幅で数サイクルかけて行う．また，それぞれダイレクトマップ，ダイレクトマップ，4wayセットアソシエイティブキャッシュ構成になっている．表1の値はFPGAとASICともに仕様変更後のものである．表1の比率は総ロジックに対する各ステージのロジック比を表している．SEL/RD&WRは，ASICでは21.9%に対してFPGAでは17.0%と比率が低い．これは，FPGAの論理合成ツールがレジスタファイルをXilinx社FPGAに搭載されているDistributed Select RAMへ置き換えるため，使用するLUTが削減されたからである．また，ALU，MEDIAといった算術論理演算は，FPGAに比べてASICの比率が高い．これは，論理合成ツールの最適化が影響していると考えられる．その他のステージでは，ロジックの比率はFPGAとASICでほぼ同じである．キャッシュに関しては，ARM-IF，FRV-IF，OP1で同じ16KBでもCacheの値が異なっている．これはキャッシュの規模が読み書きのbit幅とway数の構成に依存して増加しているからである．

4.2 遅延時間

表1のdelayは遅延時間を表す．FRV-IFは，FPGAでは遅延時間が28.7nsとなり，クリティカルパスとなっている．これは，クリティカルパスがFR-Vフロントエンドにあり，OROCHIの特徴であるバックエンドにクリティカルパスがないということを意味する．つまり，OROCHIの設計はVLIWプロセッサの回路と同等以下の複雑さになっていると言える．ARM-IA，FRV-IA，FRV-D，BRC，MEDIAといったFPGAにおいて遅延時間が10ns未満となるステージはクリティカルパスの候補ではない．これらのステージは，ASICでも4ns未満となり，クリティカルパスの候補になっていない．しかし，ALUは，FPGAでは遅延時間が2.6nsに対してASICでは遅延時間が8.7nsとクリティカルパスになっている．これは，FPGAの論理合成ツールがALUの乗算機能をLUTではなく，Xilinx社FPGAに搭載されている専用乗算器に置き換えたため，遅延時間が改善したからである．ALUの次に遅延時間の長いステージはFRV-IFとなり，遅延時間は6.4nsとなり，FPGAと同じステージがクリティカルパスとなっている．

ASICのメモリの遅延時間はデータシートより5ns程度であるため，キャッシュを持つARM-IF，FRV-IF，OP1の遅延時間は5nsより短くなることはない．クリティカルパスの候補となるこれらのステージの遅延時間は5nsから6ns程度になっており，メモリの遅延時間とほぼ等しいことからこれらのステージの設計が妥当であったと言える．

4.3 プロセッサコアの比較

OROCHI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を目標としている．そこで，ARMコアとFR-Vコアを仮定してOROCHIの回路規模について評価を行った．そのために，OROCHIからFR-V部分を切り離したものをARMコアとし，同様にOROCHIからARM部分を切り離したものをFR-Vコアとした．ARMコア，FR-Vコア，OROCHIの論理合成結果を**表3**に示す．表3のマルチコアはARMコアとFR-Vコアの合計を表している．

この結果から，ASICの制約なしでは，OROCHIはARMコアとFR-Vコアを並置したマルチコアプロセッサと比べて，ロジックとキャッシュを含めて，76%の回路規模，98.7MHzの動作周波数を達成して

表 3 OROCHI のプロセッサコア比較

	FPGA 33MHz					ASIC 制約なし			ASIC 80MHz		
	LUTs		BRAM		freq.	Total		freq.	Total		freq.
		[%]		[%]	[MHz]		[%]	[MHz]		[%]	[MHz]
ARM コア	59384	—	22	—	35.1	813666	—	99.9	693478	—	83.5
FR-V コア	52610	—	22	—	35.1	645685	—	100.8	578637	—	83.5
マルチコア	111994	100	44	100	—	1459351	100	—	1272115	100	—
OROCHI	77969	70	32	73	34.7	1115885	76	98.7	960331	75	82.9

いる．この ARM コアと FR-V コアは VLIW 型命令キューといった OROCHI の特徴を持つものであり，一般的な ARM コアや FR-V コアの回路規模がこれらより大きいものであれば，OROCHI の規模はさらに小さいと言える．FPGA では，OROCHI はロジックで 70%，キャッシュで 73%の回路規模，34.7MHz の動作周波数を達成している．

OROCHI は 2.2 節で述べたように ASIC をドータボードを介して FPGA に接続するためクロックは FPGA から供給する形になる．動作周波数は 66MHz となり，20%のマージンを想定して論理合成は動作周波数を 80MHz に設定している．論理合成結果を表 3 の ASIC 80MHz に示す．OROCHI の回路規模はマルチコアプロセッサの 75%となる．また，動作周波数は 82.9MHz を達成し，タイミング制約を満たしている．

5. ま と め

異種命令 SMT プロセッサ OROCHI の開発はソフトウェアシミュレーションによる検証から ASIC 試作まで段階的な開発を行っている．本報告では，ソフトウェアシミュレーションと RTL について記述量と実行速度について述べた．また，FPGA と ASIC を対象とした論理合成を行った結果，OROCHI は，ARM コアと FR-V コアを並置するマルチコアプロセッサと比べて FPGA では回路規模は 72%，動作周波数は 34.7MHz となり，ASIC では回路規模は 75%，動作周波数は最大 98.7MHz となることがわかった．クリティカルパスの傾向は各パイプラインステージで比較すると乗算機能を除いて差異は見られないことがわかった．今後は，ASIC の実運用での消費電力の評価を行う予定である．

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．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，日本ケイデンス株式会社およびシノプシス株式会社の協力で行われた．

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) T. Shiota, et al, “A 51.2GOPS, 1.0GB/s-DMA Single-Chip Multi-Processor Integrating Quadruple 8-Way VLIW Processor”, ISSCC, p.18-19 (2005)
3) Shimada, H., Shimada, T., Tabata, T., Kojima, T., Kise, K., Nakashima, T., Kitamura, T.:“Outline of OROCHI: A Multiple Instruction Set Executable SMT Processor”, IWIA (2007)
4) ARM Architecture Reference Manual, ARM Limited, DDI 0100E (2000)
5) FR550 Series Instruction Set Manual Ver.1.1, 富士通株式会社 (2002)
6) 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 (2007)
7) 中田尚，中島康彦，“異種命令混在実行のための VLIW 型命令キューの設計”，情処研報 2007-ARC-175, pp.89-94 (2007)
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) 須賀圭一，山原幹雄，中田尚，中島康彦，“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”，情処研報 2007-OS-106, p.17-22 (2007)
10) Austin, T., et al, “SimpleScalar: An Infrastructure for Computer System Modeling”, IEEE Computer, pp.59-67 (2002)