計算機アーキテクチャ 152−27
（２００３．３．１１）

マルチメディアネットワーキング向きデータ駆動プロセッサの命令セットアーキテクチャ

安村康平[†] 伊藤伸也[†] 榑林亮介[††]
冨安洋史[†††] 西川博昭[†††]

筆者らはデータ駆動プロセッサのオーバヘッドの無い多重処理を活かしつつ，逐次処理を高効率化したメディア・プロトコル処理向きのプロセッサを設計・試作中である．逐次処理の高効率化を実現するために，本プロセッサではデータ駆動・制御駆動プログラムを同時実行が可能である．本論文ではこのプロセッサの命令セットアーキテクチャについて論じ，データ駆動・制御駆動プログラムの双方を固定長 32 ビットに格納可能な命令形式を提案している．また，提案命令セットにより，プログラムサイズ，ダイナミックステップ数が削減できる．さらに実行ユニットについて回路規模を評価し，SIMD 演算を追加しても，回路規模の増加は約 12%であることを示す．

An Instruction Set Architecture of a Data-Driven Processor for Multimedia Networking Applications

Kohei Yasumura,[†] Shinya Ito,[†] Ryosuke Kurebayashi,[††]
Hiroshi Tomiyasu[†††] and Hiroaki Nishikawa [†††]

A data-driven processor has benefits for real-time processing such as instruction-level multi-processing without context switching overheads. However, the potential inefficiency in sequential processing of a data-driven processor has given rise to a bottleneck to overall performance. To attack this problem with maintaining the advantage of a pure data-driven processor, we are developing a processor which can execute both of data-driven and control-driven program with the same pipeline. This paper describes an instruction set architecture of this processor for multimedia networking applications. This paper presents this instruction set reduces program size and the number of dynamic steps. In addition, this paper examines the area of execution unit and shows that the effect on our SIMD implementation is about 12% increase in size.

1. はじめに

マルチメディア情報をネットワークを介して送受信することに対する需要は年々高まる傾向にあり，マルチメディアコンテンツの放送・配信を可能とするマルチメディアネットワーキング環境が望まれている．

マルチメディアネットワーキング環境では，マルチメディアコンテンツの実時間通信が要求されるため，QoS(Quality of Service) の保証が不可欠である．ネットワークノードを構成するプロセッサ技術に関しては，年々増加していくネットワークリンクの帯域を活用できる処理能力が求められている．また，単純なパケット転送だけでなく，多様化する端末の接続形態に対応するために，マルチメディアデータの形式変換[1] などのサービスが求められている．これらの要求に応えるため現在，ネットワークプロセッサやメディアプロセッサの研究・開発が進んでいる．レイヤ 2-3 におけるパケット転送処理はパケット間の並列性が豊富であるため，ネットワークプロセッサではこれを TLP(Thread Level Parallelism) として利用することが一般的である[2]．一方で，メディア処理に特化した DSP（Digital Signal Processor）などは ILP(Instruction Level Parallelism) を活用するアーキテクチャを採る場合が多い[5]．しかしメディアデータの形式変換などより高いレイヤまでを同時にサポートするには，粗粒度の並列性と細粒度の並列性を同時に抽出できることが望ましい[3]．TLP と ILP を双方を柔軟に活用できるプロセッサとして，SMT(Simultaneous Multi-Threaded processor)[4] とデータ駆動プロセッサが挙げられる．データ駆動プロセッサは文脈切り替えのオーバヘッドがなく，多重または同時に処理可能なプロセス数が SMT のようにレジスタファイル数によって制

† 筑波大学大学院博士課程システム情報工学研究科
Doctoral Program in Systems and Information Engineering, University of Tsukuba
†† 筑波大学大学院博士課程工学研究科
Doctoral Program in Engineering, University of Tsukuba
††† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

限されないという特徴をもつ．

そこで，筆者の所属する研究室では，データ駆動プロセッサ CUE-v1(Coordinating Users' requirements and Engineering constraints-version1) を研究し，データ駆動プロセッサの高い細粒度並列性がプロトコル処理・メディア処理双方に有効であることを実証してきた[6]．その一方で，データ駆動プロセッサの逐次処理を避けられない部分の非効率性や，CUE-v1 の命令セットアーキテクチャの課題が明らかになった．そこで筆者らはマルチメディアネットワーキング向きデータ駆動プロセッサとして，従来のデータ駆動プロセッサの利点であるオーバヘッドのない多重処理という特徴を維持しつつ，逐次処理を高効率化したプロセッサの LSI 設計・試作を進めている．本論文では設計・試作中のプロセッサの命令セットアーキテクチャについて論じている．以下第 2 章では，本研究で対象とするマルチメディアネットワーキング向きデータ駆動プロセッサのアーキテクチャの概要について述べる．次に第 3 章では，提案する命令セットアーキテクチャとそれらの命令を実行するための実行ユニットの実装について述べる．第 4 章ではダイナミックステップ数やプログラムサイズにより本命令セットアーキテクチャの評価を行い，さらに実行ユニットの回路規模などに基づく評価を行う．

2. マルチメディアネットワーキング向きデータ駆動プロセッサ

2.1 CUE アーキテクチャ

CUE-v1 は動的データ駆動原理を VLSI 向きに実現した，オンチップマルチプロセッサである．CUE-v1 のブロック図を図 1 に示す．プロセスルールが 0.25μm であり，パイプラインの最大処理能力は 180$Mpacket/s$ である．また，1 パケットあたり 12bit のデータを持つ．CUEv-1 は図 1 に示すパケットという単位で処理を行う．各パケットにはデータだけでなく，タグと呼ばれる文脈を識別するための識別子を持つため，異なる文脈の処理を命令単位で並列に実行することができる．

CUE-v1 の構成要素 (Processor Element;PE) はヘテロジニアスな命令セットをもっており，それぞれの PE の処理機能は，INT が論理・加減算，GNT がタグ操作，MUL・SUM が累積加減・乗算，TBL がデータをアドレスとしたメモリアクセス，そして VM がタグをアドレスとしたメモリアクセスである．

2.2 プロトコル処理・メディア処理の評価

CUE-v1 でのプロトコル処理・メディア処理において以下のような成果を得ている．

- OC-3 ATM(実効 136Mb/s) での CORBA(Common Object Request Broker Architecture) プロトコル処理
- 24bit color VGA(Video Graphics Array) サイズ

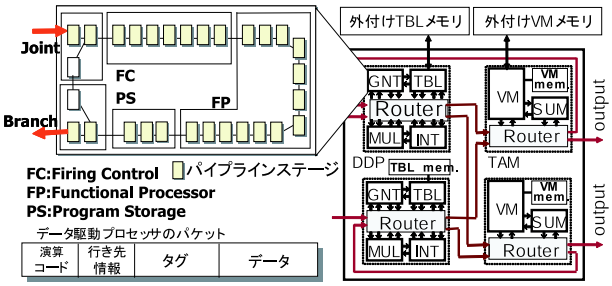

図 1 CUE-v1 ブロック図

15fps(110Mb/s 相当) での動画像圧縮処理

- 8KHz 16bit ADPCM(Adaptive Differential Pulse Code Modulation)(128kb/s) の音声圧縮処理

しかし，CUE-v1 は動画像信号処理向きに設計されたプロセッサであるため，これらの実現法からマルチメディアネットワーキング向きプロセッサアーキテクチャを実現するための主な課題として，以下の 4 つが得られた．

(1) メモリアドレッシングの自由度
CUE-v1 のメモリアドレッシングは動画像処理向きであるため，プロトコル処理等では一命令で目的のメモリ番地にアクセスできず，他の PE でアドレス計算を行う必要がある．

(2) word 長の不足
CUE-v1 は高精細動画像のデコード用にチューニングされているため，データ部のサイズは 12bit である．しかし，ADPCM では 16bit，プロトコル処理では 32bit をデータ単位として扱うことが多い．

(3) PE 間の負荷の非平衡
特定のアプリケーションに限定する場合，ヘテロジニアスな命令セットはハードウェア量を削減に有効である．しかし，CUE-v1 におけるプロトコル処理，音声圧縮処理及び動画像の処理の各 PE の動的な発火回数は図 2 のようになっており，アプリケーションによっては全く使われない PE がある．また，アプリケーションごとに命令の使用頻度に大きなばらつきがあり， PE 間に負荷の非平衡が生じている．

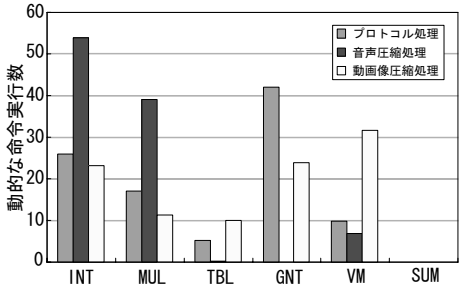

図 2 各 PE における動的発火回数

(4) 逐次処理の非効率性

CUE-v1 でデータ依存関係のある命令の実行を開始するには，そのソースオペランドを生成する命令の実行完了を待たなければならない．すなわち，環状パイプライン一周分の遅延が生じる．そのため，メディア処理におけるビットストリーム出力処理など，逐次処理を避けられないアルゴリズムの場合，パイプラインの利用効率が悪い．

2.3 データ駆動・制御駆動プログラムを多重実行するプロセッサアーキテクチャ

逐次処理を高効率化するために，本研究で対象とするプロセッサアーキテクチャはデータ駆動図式によって表現したプログラム（以下，データ駆動プログラム）と実行順序に基づき命令を逐次的に並べて表現したプログラム（以下，制御駆動プログラム）を同一パイプライン上で命令単位に多重処理する．処理対象の並列化が可能な箇所についてはデータ駆動プログラムにより表現し，並列化が困難な箇所については制御駆動プログラムにより表現する．データ駆動プログラムの命令はデータ依存関係に基づき発行される．一方，制御駆動プログラムの命令は PC(Program Counter) に基づき逐次的に発行される．このように異なる駆動原理に基づき発行されるデータ駆動・制御駆動プログラムの各命令は，相互に干渉することなくパイプラインを共有し，実行される．

本プロセッサアーキテクチャのパイプライン構成を図 3 に示す．本アーキテクチャは，8 種の機能ブロック，すなわち，命令フェッチ部 (Instruction Fetch;IF)，命令デコード部 (Instruction Decode;ID)，パケット転送スイッチ (Switch)，発火制御部 (Firing Control;FC)，整数論理演算部 (INTeger;INT)，ロードストア部 (Load/Store;L/S)，分岐部 (BRanch;BR) 及びレジスタ書き込み部 (Write Back;WB) からなる．CUE-v1 で生じていた PE 間の負荷の非平衡を解消するために，6 種あった実行ユニットを INT と L/S に集約している．FC はスーパスカラにおけるリザベーションステーションとしても機能する．制御駆動プログラム実行時には 2 命令同時発行かつ out-of-order 実行となる．

3. 命令セットアーキテクチャと実行ユニットの設計

3.1 メディア処理・プロトコル処理向き命令

一般にメディア処理は画一的な処理を連続するデータに対し適用する．そのため，1word を複数のフィールドに分けて各々一つの変数として演算する命令，すなわち SIMD 演算が有効である[7]．一方，ネットワークプロセッサは，パケット転送処理を効率良く行う命令を備えている．例えば，入力オペランドのデータを，上位 bit から検査していき，何 bit 目が始めて 1 になるか（もしくは，0 がいくつ続くか）を出力する命令がある．以下，

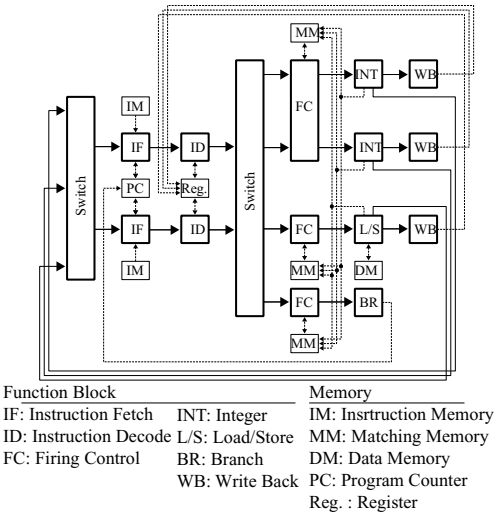

図 3 本プロセッサのパイプライン構成

この命令をリーディングゼロ命令を呼称する．これは IP でパケットの転送先を決定するのに有効な命令である．転送先を決定する際，IP パケットの行き先アドレスと転送先候補のアドレスをマッチさせ，最も長くマッチした転送先を選ぶ[8]．よって，マッチした bit 数を知ることができる命令が有効にである．

3.2 命令セットの概要

前節で述べたメディア処理・プロトコル処理向きの命令セットを考慮し，命令セットを決定した．表 1 に本プロセッサの命令セットの概要を示す． INT では算術・

表 1 命令セット概要

INT	L/S	BR
算術・論理演算 シフト，ジャンプ， リーディングゼロ タグ操作	タグ及びデータ部による メモリアクセス	制御駆動実行用 分岐命令

論理演算，シフト，ジャンプ命令の他にタグ操作，リーディングゼロ命令を行う．L/S はメモリアクセスを行う．詳細は 3.2.2 で述べる．また BR は制御駆動用の分岐命令を行う．さらに，算術演算とシフトは SIMD 演算が可能である．これについては 3.2.1 で詳しく述べる．またデータ駆動命令から制御駆動命令を起動する命令，制御駆動命令からデータ駆動命令を起動する命令を用意する．なお，本プロセッサでは，CUE-v1 が備えていた命令のうち，メディア・プロトコル処理において使用頻度が少なかった命令（算術結果のシフト，ローテートなど）を削除している．

3.2.1 SIMD 演算

加算，減算命令はそれぞれ 8bit,16bit ずつの SIMD 演算命令が可能である．また，オーバフローを起こした場合は飽和処理を行う命令を持つ．飽和処理とは，オーバフロー/アンダーフローした際にその最大値/最小値

を自動的に代入する演算である．本プロセッサはワード長が 32bit であり，それを 8bit×4，16bit×2 として SIMD 演算を行う．この演算を行うためには各々の変数を 1word に埋め込む，または取り出す必要がある．これを効率良く実行するために，32bit のフィールドに対して，任意のデータを 8bit，または 16bit ごとに埋め込む pack 命令，または取り出す unpack 命令を用意する．また同一の 32bit フィールドに埋め込まれた，変数間の演算も可能にした．隣合う 8bit 同士の乗算と，隣合う 16bit 同士の乗算が可能である．また隣り合う 16bit 同士を加算する命令も加えた．これにより積和演算のステップ数を削減することが可能になる．また TCP・UDP/IP のチェックサム計算にも有効な命令である．

3.2.2 メモリアドレッシング

CUE-v1 は動画像信号処理向きのプロセッサであった．そのためメモリアドレッシングの方法が非常に限定的である．データ駆動プログラムによるプロトコル処理では，24bit のタグの上位 12bit をストリーム識別子として利用し，下位 12bit をストリーム中の要素の識別子として使用している．このストリーム識別子をメモリアドレッシングの際にベースアドレスとして使用するなど，タグを操作してアドレスとする場合が多い．しかし．CUE プロセッサのメモリアドレッシングは図 1 中の VM はタグ+即値のみ，TBL はデータ部のみのアドレッシングとなっており，各 PE でメモリアドレッシングが固定されている．VM はタグ全ての領域をベースアドレスとして使用するため，ストリーム識別子をデータ部に移す処理やマスクをとる処理を，INT や GNT で行わなければならず，プログラムのステップ数が増加する．

そこで本プロセッサはタグをシフトしてベースアドレスとするアドレッシングモードを備える．図 4 にタグを利用したメモリアクセスの方法を示す．タグをベースアドレスとして使用する場合，タグごとに一定のメモリ領域を確保できるようにするために，タグをシフトさせる．シフト幅は専用のシフトテーブルに格納しておく．シフトテーブルのアドレスはオフセットで使用するデータの上位 3bit を用いることとする．またデータ部にオフセットを加算するアドレッシングモードを追加した．本プロセッサのアドレッシングモードをまとめると図 5 のようになる．

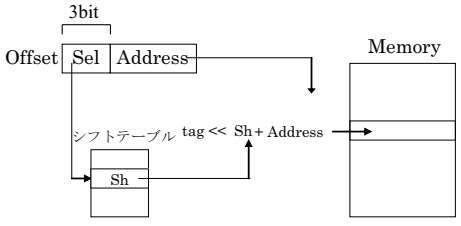

図 4 タグを利用したメモリアクセス

CUEプロセッサ	本プロセッサ
タグ+オフセット（即値）	データ駆動実行
データ部	タグシフト+オフセット（即値）
	タグシフト+オフセット（右データ）
	タグシフト+オフセット（左データ）
	右データ　+オフセット（即値）
	制御駆動実行
	タグシフト+オフセット（即値）
	タグシフト+オフセット（レジスタ）
	レジスタ　+オフセット（即値）

図 5 メモリアドレッシングの比較

3.3 データ駆動実行と制御駆動実行の双方を可能にする命令形式

本プロセッサではデータ駆動命令と制御駆動命令と実行の制御方式の異なる 2 種類の命令形式が必要になる．この 2 種類の命令形式をできるだけコード効率を良くプログラムメモリに格納することが求められる．本プロセッサは実装の制約から命令長を 32bit とした．またフェッチとデコードのハードウェアを簡単にするため，命令長を固定長とした．これらの制約を満たしつつ，各命令形式に必要なフィールドを決定する．

3.3.1 データ駆動命令と制御駆動命令

データ駆動命令と制御駆動命令は，その動作原理の違いから命令形式に含める情報が異なる．双方の命令形式にはデータ駆動命令と制御駆動命令を区別するフラグを用意する．また命令の種類を識別するオペコード，条件分岐を行うための条件コード (Condition Code;CC) 等のフィールドが必要になる．データ駆動命令は 2 つもしくは 1 つのオペランドをとる．本論文ではこのオペランドをそれぞれ左オペランド (O_L)，右オペランド (O_R) と呼ぶ．1 オペランド命令は O_L のみとする．データ駆動命令では能動的にオペランドを指定する必要はないが，計算結果が次命令で O_L となるか O_R となるかを指定するフラグが必要になる．また，あて先アドレスを保持しておく必要がある．一方データ駆動命令ではレジスタを指定するフィールドが必要になる．

3.3.2 データ駆動命令の命令形式

図 6 にデータ駆動命令の命令形式を示す．データ駆動命令形式の場合，最も大きい制約が行き先ノード (Destination Node;DN) である．データ駆動命令では常に次命令のアドレスを指定しなければならない．プロトコル処理・メディア処理の各プログラムの規模は約 2000 ノード必要であるため，少なくとも 4000 ノード〜8000 ノード分必要である．これを指定するためには 12〜13 ビット必要になる．さらに，分岐を行うには，真と偽双方の行き先を保持しておかなければならず，他のフィールドを勘案すると，命令長が足りない．そこで以下の 2 つの方法を採用する．まず一つ目はあて先アドレスを現アドレスの相対アドレスで表現する方法である．より大きなとび先を指定するために，ジャンプ命令を用意する．2 つ目は計算結果が 条件コードで指定される条件

において真であれば DN の命令を発行し，偽であれば DN+1 に示す命令を発行する方法である．この方法により行き先を一つだけ保持しておけば良い．これらの方法により DN フィールドは 8bit にできた． また，2 オペランド形式では，Immediate フィールドを Function として利用することによってオペコードを拡張する．

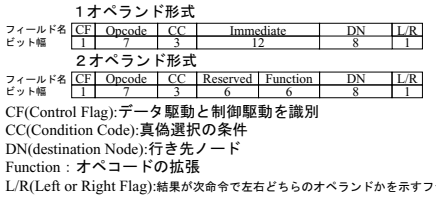

図 6 データ駆動命令形式

3.3.3 制御駆動命令の命令形式

図 7 に制御駆動命令の命令形式を示す．制御駆動命令は PC に従い，格納されたアドレス順に発行される．制御駆動命令ではオペランドのソース・デスティネーションレジスタを示すフィールドが必要になる．制御駆動モード用のレジスタは 16 本とし，3 オペランド形式とする．従ってレジスタ指定に必要なビット数は $4 \times 3 = 12$bit となる．制御駆動命令でのジャンプ命令は，指定レジスタが CC を満たしたときのみジャンプする．

図 7 制御駆動命令形式

3.4 INT 実行ユニットの概要

INT の命令を実行する INT 実行ユニット（以下実行ユニット）の概要を図 8 に示す． 実行ユニットは，演算を行う Operation Unit，第二オペランドを選択する Operand Selector，どの計算結果を出力するかを示す Result Selector，行き先ノードを生成する Destination Generator，出力するタグを選択する Tag Selector，オペランドをバイパスするどうかを選択する Bypass Selector からなる．入力信号として FC より第一，第二オペランド（図中 Operand0,Operand1），タグ (図中 Tag)，あて先ノード番号 (Distination Node)，ID でデコードされた演算器を制御する，Control Signal，Select Signal が送られてくる．

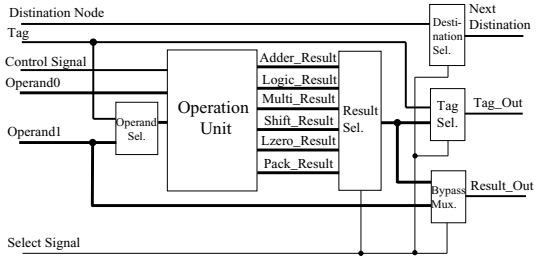

図 8 実行ユニット全体図

データ駆動命令でのタグに対する演算を行う演算器を Operand 同士の演算を行う演算器と共通化するために，Operand Selector がある．また，データ駆動命令では CC と計算結果に基づき Destination Node を決定する必要がある．その役割を担うのが Destination Generator である．

演算を行う Operation Unit は，加減算を行う Adder，論理演算を行う Logic，乗算を行う Multiplier，シフトを行う Shifter，リーディングゼロ命令を行う Leading Zero，パック，アンパック演算を行う Pack・Unpack からなる．以下，SIMD 演算を行う手法を Adder を例に説明する．SIMD 演算は 8 ビットの加減算器を 4 つ並べ，Carry を伝播させるかを判定するセレクタをはさむことによって実現する．このセレクタは，パック演算を判断する packed_flag，SIMD 演算の幅を指定する width_flag，減算を指定する Substract_flag の 3 つの信号によって制御される．

4. 命令セットアーキテクチャの評価

4.1 命令種類数の比較

表 2 に命令種類数の比較を示す．本プロセッサは SIMD 演算やリーディングゼロ命令を追加しているが，3.2 で述べたように，メディア処理・プロトコル処理のデータ駆動型実現法で使用頻度の低い命令を削除しているため，命令の種類を約 10%削減できている．

表 2 命令種類数の比較

CUE-v1 の命令種類数

PE 名	命令数
INT	18
GNT	43
MUL	8
SUM	6
TBL	106
VM	21
計	202

本プロセッサのデータ駆動命令種類数

PE 名	命令数
INT	120
L/S	61
計	181

4.2 プログラムサイズとダイナミックステップ数

本プロセッサは一命令で目的のアドレスを参照するために，タグをシフトしてベースアドレスとして利用できるアドレッシングモードを追加した．その結果，CUE-v1 に比べ，TCP/IP プログラムの全 1036 ノード中，73 ノードを削除することができ，プログラムサイズを約

7%小さくできる．

次にリーディングゼロ命令の効果を評価するために，IP の経路選択プログラムを用いて，プログラムサイズ，ダイナミックステップ数の観点から比較を行う．CUE-v1 ではネットワークアドレスの短い方から ID1〜4 を振り，4 つのアドレスを同時に比較する．マッチしたアドレスの中で一番 ID の大きいネットワークアドレスを選択することによって最長一致による経路選択を実現している．一方本プロセッサアーキテクチャでは，行き先 IP アドレスと候補の IP アドレスの排他的論理和をとる．その値に対してリーディングゼロ命令を行い．一致したビット数によってネットワークアドレスを決定する．また評価の対象をリーディングゼロ命令の効果に絞るために，CUE-v1 の Word 長を 32bit として評価した．

表 3 プログラムサイズとダイナミックステップ数の比較

	CUE-v1	本プロセッサ
プログラムサイズ (bit)	1088	384
ダイナミックステップ数	19	12

表 3 に両プログラムのプログラムサイズとダイナミックステップ数を示す．CUE プロセッサの場合に比べ，本プロセッサはプログラムサイズは 1/3，ダイナミックステップ数は 2/3 にすることができた．

4.3 実行ユニットの回路規模の評価

実行ユニットの回路規模を評価する．ここではハードウェア記述言語 Verilog-HDL を用いて RTL(Register Transfer Level) 記述を行った．さらにこれを Synopsys 社の Design Compiler を用いて論理合成し，面積を評価した．ここで用いたライブラリは Artisan 社から提供されてる TSMC 社 0.18μm プロセス用のスタンダードセルライブラリである．なお Design Compiler による論理合成を容易にするため，論理合成の難しい部分は直接論理式レベルで記述した．

表 4 に実行ユニット全体の回路規模を示す．SIMD 実装が無い場合も同時に示した．2 入力 NAND 換算では全体は約 5700 ゲートの規模となる．SIMD 演算機能による回路規模の増加は Adder では 33%の程度，Multiplier では 4%程度，Shifter では 72%程度の回路規模の増加となるが，全体では約 12%の増加になる．これは十分許容できる範囲内で SIMD 機能を追加することができたと言える．またリーディングゼロ命令の追加によるハードウェアの追加は演算器全体の 2%程度である．

表 4 実行ユニットの面積 μm^2

	SIMD 実装有り	SIMD 実装無し	増加率
Adder	5179	3895	33%
Multiplier	37119	35629	4%
Shifter	5894	3430	72%
Logic	971	971	—
Pack,Unpack	755	—	—
Leading zero	1174	1174	—
その他セレクタ等	5511	5511	—
全体	56783	50610	12%

5. おわりに

本論文はマルチメディアネットワーキング環境向きのデータ駆動プロセッサにおける命令セットアーキテクチャを明らかにした．さらに CUE-v1 に比べ，命令種類数を削減し，プログラムサイズ，ダイナミックステップ数も削減できることを示した．加えて実行ユニットの回路規模の評価を行い，SIMD 演算の追加による演算器全体の回路規模の増加を約 12%程度に抑えることができることを示した．今後は試作した LSI 上で本プロセッサのプロトコル処理，メディア処理のスループット，多重処理性能を評価していく予定である．

謝辞 本研究の一部は，民間との共同研究「マルチメディアネットワーキング向きデータ駆動プロセッサの研究」によるものである．

参 考 文 献

1) A. Fox, S. D.Gribble, E. A.Brewer, and E. Amir, “Adapting to Network and Client Variability via On-Demand Dynamic Distillation,” 7th Int’l Conf. on Architectural Support for Programming Languages and Operating Systems,pp.160-170,1996.
2) Intel Corp, “Intel IXP2800 Network Processor,” product brief,2002.
3) P. Crowley, M. E. Fiuczynski, J. Baer, and B. N. Bershad, “Caracterizing Processor Architecture for Programmable Network Interfaces,” Proc. of the 2000 International Conference on Supercomputing, pp. 54-65, May 2000.
4) S. J. Eggers, J. S. Emer, H. M. L. Jack, L. L. Rebecca, L. S. Dean, and M. Tullsen, “Simultaneous Multithreading: A Platform for Next-Generation Processors,” IEEE Micro, pp. 12-19, Oct. 1997.
5) F. Arakawa, O. Nishi, K. Uchiyamaa,and N. Nakagawa, “SH4 RISC Multimedia Microprocessor,” IEEE Micro, pp. 26-34, Mar. 1998.
6) H. Nishikawa, and R. Kurebayashi, “ A Networking Oriented Data-Driven Processor: CUE,” International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems 2002, IEEE Computer Society Press, pp. 103-111, 2002.
7) T. Huff, “Internet Streaming SIMD Extensions,” IEEE Computer, Vol.32, No.12, pp. 26-34, Dec. 1999.
8) T.V. Lakshman, and D.Stiliadis, “High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching,” Proc. of ACM SIGCOMM ’98, pp. 203-214, Sep. 1998.