

ネットワーキング向きイベント駆動型 チップマルチプロセッサ CUE-v3 の性能予測

富安 洋史[†] 岡本 政信[†] 西川 博昭[†]

著者らは、データ駆動・制御駆動スレッドを命令レベルで同時・多重処理し、データ駆動方式の多重処理性を維持しつつ、時間・順序依存処理に不可避となるインライン処理能力を改善したネットワーキング向きハイブリッドプロセッサ CUE-v2 の VLSI 試作を行ってきた。現在、この CUE-v2 を基にし、低レイテンシのスレッド起動命令を拡張したイベント駆動型チップマルチプロセッサ CUE-v3 の開発を進めている。本論文では、CUE-v3 によって達成される性能を、HDL シミュレータを用いた RTL シミュレーションによって予測する。まず、CUE-v2 の特徴とそのチップマルチプロセッサ CUE-v3 について述べる。次に本論文におけるシミュレーション手法について説明し、MPLS (Multi-Protocol Label Switching) のヘッダ処理をベンチマークプログラムとして取り上げ、CUE-v2 の性能をその実時間多重実行におけるターンアラウンドタイム、多重度のパラメータとして考察した。その結果、CUE-v2 は 5 多重程度までターンアラウンドタイムを一定に維持した実時間多重処理を実現することが確認された。最後に CUE-v3 についても、その低レイテンシプロセッサ間通信を生かしたチップマルチプロセッサ構成によって、スケーラブルにスループットを向上可能であることを示す。

Performance Prediction of Event-driven Networking Processor CUE-v3

HIROSHI TOMIYASU,[†] MASANOBU OKAMOTO[†]
and HIROAKI NISHIKAWA[†]

The authors developed VLSI prototype CUE-v2 which was realized as a hybrid processor enabling simultaneous processing of data driven and control driven threads. Keeping multi processing ability inherent in original data-driven scheme, the CUE-v2 can therefore achieve higher performance for inline processing and avoid any bottleneck in sequential parts of real-time programs. This paper discusses performance prediction of event-driven chip multi-processor CUE-v3 currently being designed by connecting CUE-v2's with low latency network. The feature of CUE-v2 and CMP extensions for CUE-v3 are first introduced. The Simulation environment for CUE-v2 and experimental results of the benchmark Real-time multi processing of MPLS (Multi-Protocol Label Switching) are then described. CUE-v2 can achieve real-time multi-processing up to 5 MPLS threads keeping minimum turnaround time of a MPLS thread. After that, similar scalable performance improvement of CUE-v3 will be discussed with special emphasis on inter-processor communication scheme with minimal overhead by utilizing event-driven operation mode in data-driven scheme.

1. はじめに

近年通信ネットワーク利用者の増大にともない、ネットワークを流れるデータはその量を増している。さらにコネクション数に応じ同時に処理を行う必要があるため、ネットワーク処理の多重度も増しており、ネットワーク環境における処理は高負荷なものとなっている。また、マルチメディアサービスのような実時間処理が要求されるサービスの登場により、ネットワーク環境に求められる処理要件はさらに厳しいものとなっ

ている。

これらの要件を満たすためには、実時間多重処理を行うネットワーク処理に向くプロセッサが求められる。多くのベンダもこのことに鑑み、ネットワーク処理に特化したプロセッサを開発してきた^{1),2)}。ネットワークプロセッサに関する研究においても、先に述べたような多重処理性を持つ SMT (Simultaneous Multi-Threading)³⁾ や CMP (Chip Multi Processor)⁴⁾ が、そのアーキテクチャに向くと報告されている⁵⁾。

著者らはこれまで、データ駆動型プロセッサがネットワーキング環境向きであると考え、その設計・試作を行ってきた⁶⁾⁻⁹⁾。データ駆動型プロセッサがネットワーキング環境向けであると考え理由は、コンテク

[†] 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering,
University of Tsukuba

ストスイッチのオーバーヘッドを最小に多重処理を行うことが可能であるという特徴を持つためである。これにより、高多重処理が要求される環境においても実時間多重処理を実現することができる。

一方、ノイマン型プロセッサにおけるコンテキストスイッチのオーバーヘッドは、数 μsec であり、これは 1 万数千サイクルに相当すると報告されている¹⁰⁾。このためネットワーク環境のような高多重処理が求められる状況において、コンテキストスイッチがボトルネックとなってしまう。

この問題に対処するには、割込みとレジスタ待避のオーバーヘッドを削減することが必要である。しかし、近年の高性能マイクロプロセッサでは、パイプラインが深くなっているため、割込みのオーバーヘッドは増加傾向にある。また、メモリアクセスのレイテンシも増加傾向にある。古くから shadow register によるレジスタ待避の高速化が行われてきたが、これは多重の割込みに対処しにくい。SMT の機構を利用して割込みプロセスに新たな仮想プロセッサを割り当てればこのオーバーヘッドを削減することができるが、中断されたプロセスが使用している物理レジスタをただちに解放することができないため、多重割込みが発生するたびに使用可能な物理レジスタが制限されてしまう。これに対し、データ駆動プロセッサでは待合せに使用する物理レジスタをより小さな粒度で管理できるため、より多重度の高い処理にも対応可能である。また、この待合せ機構はプロセッサ間にも適用できるため、物理レジスタを拡張するのと同じ効果を容易に得ることができる。物理レジスタはオペランド待合せのために連想記憶機能を持つため、容量に比して規模が大きくなりやすく、しばしばクリティカルパスともなる。したがって、多重度の高い処理においてはデータ駆動プロセッサの方が拡張性に優れると考えられる。

上記のようにデータ駆動型プロセッサはスケラビリティに優れるとされており、CMP を構成することで、ターンアラウンドタイムを最短かつ一定に保ちながら、その台数効果を期待できる。このような観点に立ち、現在著者らはネットワーク環境において、実時間高多重処理を実現するために、ハイブリッドプロセッサ CUE-v2^{6),8)} をプロセッシングエレメント（以下では PE とする）としたイベント駆動型 CMP CUE-v3 の試作・設計を行っている。本論文では、CUE-v3 の設計、および性能予測について述べる。

2. ハイブリッドプロセッサ CUE-v2

2.1 CUE-v2 のアーキテクチャ

データ駆動プロセッサでは循環パイプラインに入力されるデータ流量により性能が決められる。しかし、プログラムの並列度が低い部分、特に逐次処理部分ではパイプラインをほとんど埋めることができず、性能が低下してしまう。そこで著者らは、データ駆動プロセッサにおける発火制御と、スーパスカラプロセッサにおける動的スケジューリングとはほぼ等価であるという事実に着目し、データ駆動・制御駆動の双方を共通のパイプラインで処理するプロセッサの実現を目指した^{6),8)}。データ駆動の発火制御とスーパスカラプロセッサにおける動的スケジューリングとの相違点は、命令を発火させるのに必要なデータの供給経路である。前者はつねにフロントエンド部から供給されるのに対し、後者は演算器からのフォワーディングパスから供給される。スーパスカラプロセッサにデータ駆動原理に基づく命令フェッチ、およびデータの受け渡しを追加することにより、10%程度のハードウェアを追加することで、データ駆動・制御駆動の双方を共通のパイプラインで処理するプロセッサを実現できた。これがハイブリッドプロセッサ CUE-v2 である。

CUE-v2 は、異なる性質を持つ 2 種類のスレッドを同一パイプライン上で命令レベルで同時・多重処理することにより、以下の 3 点が可能である。

1. データ駆動スレッドの実行により、データフローグラフから示される並列性を最大限に活用し、並列部を高効率に実行する。
2. さらに、データ駆動プロセッサの特長である、アーキテクチャ水準の実時間多重処理を実現する。
3. データ駆動スレッドの空き資源を活用し制御駆動スレッドの実行を行い、逐次処理部のパイプライン処理を可能とし、その実行時間を短縮する。

本論文では、データ駆動・制御駆動スレッドの各スレッドを以下のように定義する。

データ駆動スレッド

同一のカラーを有するトークンの実行シーケンス
従来のデータ駆動プロセッサと同様にデータ依存関係に基づき命令を発行する。

制御駆動スレッド

プログラムカウンタ (PC) に基づき連続的に発行される命令の実行シーケンス。

CUE-v2 では、これら 2 種類のスレッド間で相互に干渉することなく命令単位で同時・多重処理を可能とする命令フェッチポリシーを採用している。本ポリシーで

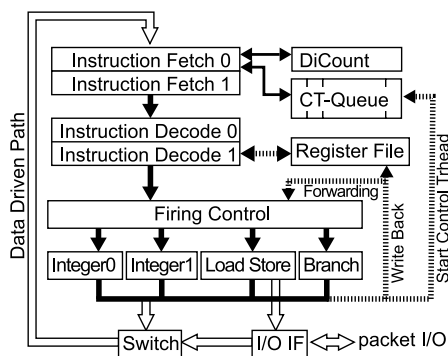


図 1 CUE-v2 パイプライン構成
Fig. 1 The pipeline of the CUE-v2.

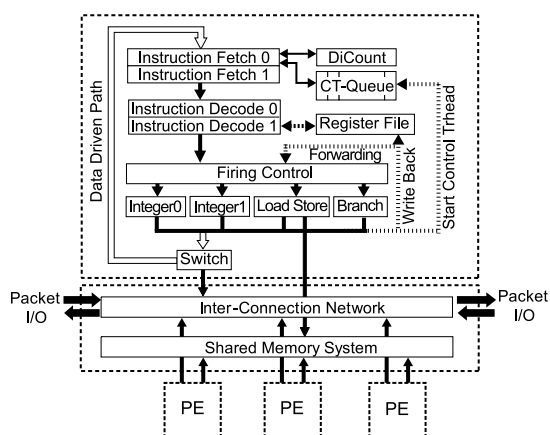


図 2 CUE-v3 チップマルチプロセッサ
Fig. 2 The structure of the CUE-v3.

は、基本的にデータ駆動命令を制御駆動命令に対し優先的にフェッチ・発行する。すなわち、CUE-v2 では、データ駆動スレッドの実行時に生じる空きパイプライン資源を制御駆動スレッドに割り当てる。

本方式のプログラミングは、基本的にデータフローグラフの作成によって行う。そして、性能を制限する逐次処理部を制御駆動命令に置き換える。この方針に基づきプログラミングすることにより、データ駆動の有する多重処理能力を損ねることなく、逐次処理部分の効率化が可能となる。

CUE-v2 のパイプライン構成を図 1 に示す。CUE-v2 は、従来のフォンノイマンプロセッサの見地から見れば、2 命令同時発行 out-of-order スーパスカラに 2 種類のスレッドを管理する機構 [DiCount, CTQ (Control-driven Threads Queue)] およびデータ駆動スレッド用の循環パスを設けた構成である。逆に、従来のデータ駆動プロセッサの見地から見れば、フロントエンド部にプログラムカウンタ (PC) に基づく命令発行およびレジスタによるデータ受渡しを追加した構成である。2 種類のスレッドの実行にできるだけ共通のハードウェアを用いるため、片方でのみ使用するモジュールは最小限にとどめている。たとえば、レジスタの本数を 16 本と制限し、比較的小さい分岐予測器しか設けていない。CUE-v2 のパイプラインは、データ駆動・制御駆動の双方において、INT 命令で 7 段、LS (Load/Store) 命令で 9 段、BR (Branch) 命令 (制御駆動のみ) で 6 段である。

3. イベント駆動型 CMP CUE-v3

3.1 CUE-v2 の CMP 化

高い応答性を持つ CUE-v2 はネットワークのようなイベント駆動型処理において効率の良い処理を行うことが可能である。CUE-v2 は最大で 9.6 Gbps の

表 1 CUE-v2, CUE-v3 諸元比較
Table 1 Specifications of CUE-v2 and CUE-v3.

	CUE-v2	CUE-v3 (想定)
プロセス	180 nm	90 nm
ダイサイズ	5.0 mm 角	5.0 mm 角
周波数	100 MHz	200 MHz
IO ピン数	115	
パッケージ	BGA 292	QFP 208

スループットを持ち、ほとんどオーバーヘッドなく複数のスレッドを実行可能なことが確認できたが、実効性能では今後普及すると考えられる 10 Gbps 程度のアプリケーションに対応することは難しいと考えられた。

そこで、さらにスループットを向上させる手段の 1 つとして、著者らは CUE-v2 を CMP 化することを計画している。予定している CUE-v3 のパイプライン構造、および全体図を図 2 に示す。

CUE-v2 は TSMC 180 nm 6M1P CMOS 5 mm × 5 mm であったため、表 1 のように Star Shuttle 90 nm CMOS プロセスを用いて、5 mm × 5 mm に CUE-v2 に相当する PE を 4 個集積し、データ駆動方式のオペランド待合せ機構をプロセッサ間通信に利用することにより、ほぼリニアにスループットを向させることを目標としている。動作周波数は CUE-v2 の 2 倍程度であるため、およそ 19 Gbps × 4 のスループットとなり、実効性能においても 10 Gb Ether 等のルーティングに十分な能力を持つと予想される。さらに、個々のプロセッサエレメントは 16 個までの結合に対応しており、より大きな CMP 構成をとることにより、今後のネットワーク環境に対応可能であると考えられる。

各プロセッサエレメントは CUE-v2 の命令セット

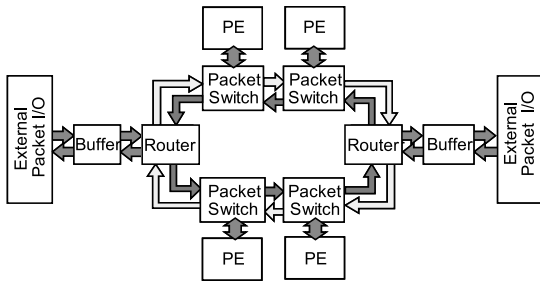


図3 CUE-v3のインターコネクションネットワーク
Fig. 3 CUE-v3 Inter-Connection network.

とパイプライン構造をそのまま引き継ぎ、さらに PE 間で互いにスレッドを起動できる命令群を付加する。これらの命令は CUE-v2 のスレッド起動命令を拡張し、PE 番号を指定可能にすることによって実現される。PE 番号の指定には命令フォーマット上の空きを利用することが可能であったため、大きくパイプラインを変更することなく、わずかなハードウェア量で実装可能である。

CUE-v3 では、必要な引数が揃っていれば PE 内と同様に 1 サイクルでスレッド起動要求を送信することができ、プロセッサ間通信路も短くレイテンシが小さいため、非常に低いオーバーヘッドで PE 間通信を行うことができる。また、元の CUE-v2 とほぼ同じプログラミングモデルで動作することができるため、高い応答性やスレッドの多重処理に対するスケーラビリティも引き継がれる。

3.2 インターコネクションネットワーク

各 PE をつなぐインターコネクションネットワークは、図3に示すような 2 重リング状のトポロジで構成される。4 個程度の PE であれば、完全結合にすることも十分考えられるが、CUE-v3 は循環パイプライン型データ駆動プロセッサの構造をそのまま持つため、1 命令ごとに必要なデータバスの幅が広い傾向がある。このようなデータバスはしばしば配線の集中によって実装が困難になるため、試作時の問題を考慮してリング状のネットワークを選択した。このネットワークについてはすでに基本的な設計を終えており、RTL シミュレーションによるデバッグを行っている。

図中の Packet Switch は宛先によって使用するべきリンクを決定し、必要な場合はパケットをバッファする。Router はチップ外へリンクを拡張するものであり、External Packet I/O を通して CUE-v3 もしくは他のデバイスを接続可能である。この外部とのリンクには、CUE-v2 における経験から安定した動作を実現することを何よりも優先すべきと判断したため、

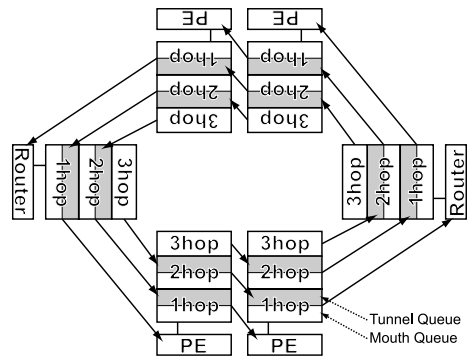


図4 仮想チャネルアクセス
Fig. 4 Virtual Channel Access.

source synch 方式の片方向バスを複数設けることとした。

各リングはデッドロックフリーを実現するために、図4のような Virtual Channel Access 方式によって構成されている。送信先との距離によってバッファを切り替えることによって、各々の通信が互いに相手を妨害することなく行われる。各 Packet Switch において、データ通過キューとなりうる 1 hop キューと 2 hop キューは、Mouth Queue と Tunnel Queue から構成されている。Mouth Queue は PE および入出力インタフェースから投入されるパケットを保持し、Tunnel Queue は隣接パケットスイッチからのパケットを保持する。この 2 つのキューから隣接する Packet Switch へ転送される際にアービトレーションを行うことによって、PE から受け取るデータと、隣接 Packet Switch から受け取るデータとの衝突を回避する。

3 hop 先の PE へ通信を行う場合、その距離は左右どちらの経路でも変わらない。しかし、動的に経路を変更すると、パケットの追越しが発生し複数のデータを送る場合に通信の終了の保証が難しくなる可能性がある。CUE-v3 のインターコネクションネットワークでは 1 ワードのデータを持つパケット単位で通信を行う単純な構成としているため、複雑な構成を用いずに転送順序を保存できる固定したルーティング方向をとることとした。このルーティング方向は各 Packet Switch ごとに全体として 2 つのリンクの通信量の不均衡が起きないように決定している。

3.3 プロセッサ間通信命令

CUE-v2 を CMP 化するにあたり、PE 間で通信を行う機構が必要である。CUE-v2 はスレッドと呼ぶプログラム小片を単位として実行するプログラミングモデルをとっており、データ駆動・制御駆動スレッドの双方を呼び出す命令を備えている。CUE-v3 ではこの

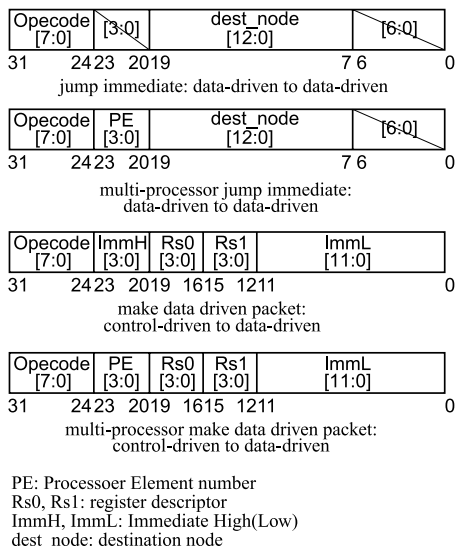


図 5 プロセッサ間起動命令の例
Fig. 5 Inter-processor activation instructions.

スレッド起動命令を拡張し、PE 間でスレッドを起動可能とすることにより、PE 間通信を行う。

図 5 に PE 間スレッド起動命令の例を示す。これらの命令は空きフィールドをプロセッサ番号とすることによって実現している。これは CUE-v2 にわずかな変更を加えるだけで実装でき、オペランド待合せ機構も PE 間で相互に利用可能である。

4. 単一プロセッサの性能評価と CUE-v3 の性能予測

4.1 評価環境

今回の性能評価では、Synopsys 社の vcs 2005.06 を用いて RTL (Register Transfer Level) シミュレーションを行った。テストパターンは CUE-v2 アセンブリ言語を用いてプログラムを記述し、それをアセンブラを用いて RTL シミュレータへの入力データ形式へと変換することによって作成した。

4.2 性能評価モデル

CUE-v2, CUE-v3 はその多重処理性能を特徴としている。この性質を検証するために、ネットワーキング環境での動作を取り上げる。実時間処理が求められるマルチメディアを利用したネットワーキングサービスの要求を満たすためには、処理多重度が増してもその性能を落とすことなく処理を行うことが求められる。この性質は、処理多重度によらず各スレッドに対して公平に資源を割り当てることができるデータ駆動型プロセッサ向きであると考えられる。これを検証するため、本論文では基本となるヘッダ処理、なか

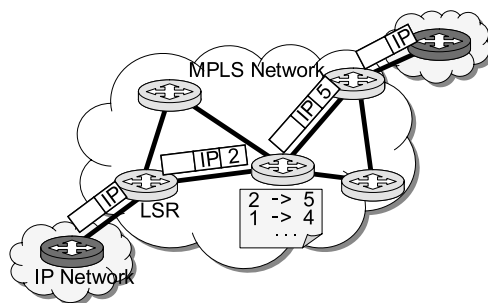


図 6 MPLS ラベルスイッチング
Fig. 6 MPLS label switching.

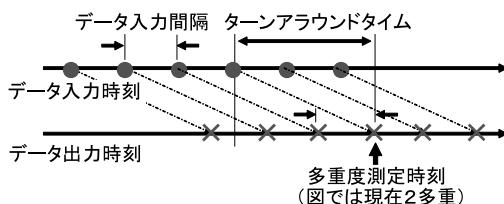


図 7 イベント駆動処理モデル
Fig. 7 Event-driven processing model.

でも、今後のネットワークにおいてその有効性が期待される MPLS (Multi-Protocol Label Switching)¹¹⁾ のヘッダハンドリングをベンチマークとして用いる。MPLS とは次世代ネットワーク NGN (Next Generation Network) において、隣接ノードとの通信を規定するレイヤ 2 と、ネットワーク間の通信を規定するレイヤ 3 との間をサポートするためのプロトコルである。これが実現されれば NGN におけるパケットハンドリングはその大部分が MPLS を利用することで行われ、IPv4, IPv6 等、数々のプロトコルを集約して通信を行うことができると考えられている。

MPLS におけるヘッダ処理は、LSR (Label Switching Router) と呼ばれるルータにおいて行われる。各 LSR では、入力パケットの経路情報に相当するラベルを新ラベルへ更新し、次のルータへ転送するという処理が行われる。MPLS パケットが転送される際の様子を、図 6 に示す。LSR に 2 というラベルを持つ MPLS パケットが入力され、LSR の持つ MPLS ラベル表に従い、適切なラベルである 5 に付け替えられ送出される。今回このラベル付け替え部分を評価モデルとして取り上げる。

具体的な性能評価方法について述べる。CUE-v2 へ 32 ビット長の MPLS ヘッダを十分な量入力し、ある入力に対応するデータの出力時刻から性能評価を行う。入力パケット時刻と、出力パケット時刻の関係から、図 7 に示すような関係が得られる。これからターンアラウンドタイム、および多重度の測定が可能とな

る．入力パケット数は，100 パケットとした．入力パケット数が少ないと，一連の処理が早く終了してしまうため，多重度の観測が難しくなり，逆に入力数が多いと冗長となってしまふ．このため，100 パケット程度が妥当と考え，100 パケットを入力とすることにした．入力されるパケットには，それぞれ 0 から 99 までの世代が一意に割り振られている．CUE-v2 へパケットが入力されると，そのパケットに付加されている世代情報をスレッド識別子として，スレッドが起動される．つまり 0 から 99 までの世代が割り振られた 100 パケットを入力するということは，CUE-v2 において 100 スレッドが起動するということになる．各スレッドでは MPLS ヘッダ処理が行われ，処理が終了すると，新ラベルが付与された 32 ビット長のパケットが出力される．ここで，入力データ間隔を変化させながら入出力されるパケット数を計測すると，CUE-v2 の過負荷状態を検出することができる．正常に処理が行われている場合は，入出力されるパケットは同数であるが，過負荷状態では十分なリソースを割り当てることができずオーバフローとなり，入力パケット数に対し出力パケット数が少なくなる．

4.3 実験結果と考察

実験結果を図 8，図 9，および図 10 に示す．

図 8 は，データ入力間隔を変えることで，平均多重処理数がどのように変動したかを表したグラフである．データ入力間隔を，16 サイクル間隔から 1 サイクルずつ短くしていき，どの時点で CUE-v2 が過負荷な状態となるかを測定した．データ入力間隔を短くすると，平均多重処理数が上がっていることが分かる．しかし，データ入力間隔を 9 サイクルに設定し実験を行った結果，多重度の向上は確認できたが，入力パケット数に対して出力パケット数が少なかった．この時点で CUE-v2 は過負荷状態に達したということが分かる．よって CUE-v2 が処理可能であるためには，入力間隔が 10 サイクル以上の間隔である必要がある．入力間隔 10 サイクル，つまり， $32 \text{ [bit]} / (1/100 \text{ [MHz]} \times 10 \text{ [cycle]}) = 320 \text{ Mbps}$ での入力データ速度が CUE-v2 の最大処理可能速度である．

次に，個々のスレッドごとの起動から終了までにかかった時間について検討する．今回作成したプログラムでは，あるスレッドは 1 つのパケットの入力で起動され，スレッドの終了時にパケットを 1 つ出力する．このためスレッド起動から終了までの時間間隔は，あるパケットのターンアラウンドタイムに相当する．図 9 に過負荷状態となる直前の 10 サイクル間隔でパケットを入力した際のターンアラウンドタイムを示す．26

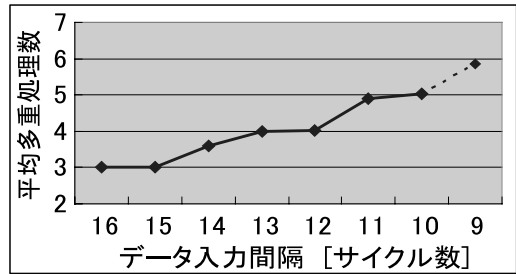


図 8 データ入力間隔に対する平均多重処理数
Fig. 8 Activated threads with data input rate.

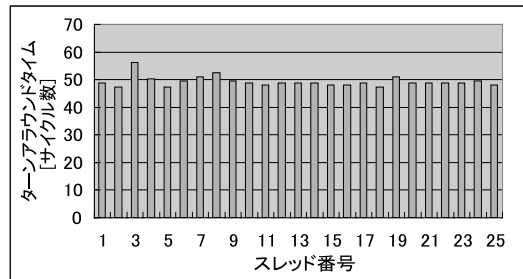


図 9 各スレッドごとのターンアラウンドタイム
Fig. 9 Turnaround time of each thread.

パケット目以降も同様にターンアラウンドタイムはほぼ一定であった．

ターンアラウンドタイムが一定であるとは，CUE-v2 内部で各スレッドごとに公平に資源が割り当てられていることを示している．3 スレッド目あたりには多少のゆれが観測されるが，その後，4，5 スレッド目でターンアラウンドタイムが短くなり，定常状態へと戻る．これは，複数の命令が実行可能な状態の場合に，FC (Firing Control) が実行する命令を選択するアルゴリズムが完全に公平ではないためである．プログラムの実行の初期には，FC の一部しか使用されておらずその使用領域が偏っているため，スレッド間でターンアラウンドタイムに非平衡が出やすい．ある程度処理が進むと FC の使用領域が分散されスレッドのターンアラウンドタイムも平均化される．

また，後述する図 10 において，多重度 1 で処理された場合のターンアラウンドタイムは，48 サイクル程度であることを示している．競合するスレッドがまったく実行されていない状態であるため，これが最短ターンアラウンドタイムである．この最短ターンアラウンドタイムと図 9 を比較すると，数サイクル程度の差異となっている．したがって，各スレッドはターンアラウンドタイムがほぼ最短，かつ一定で実行されていることが分かる．

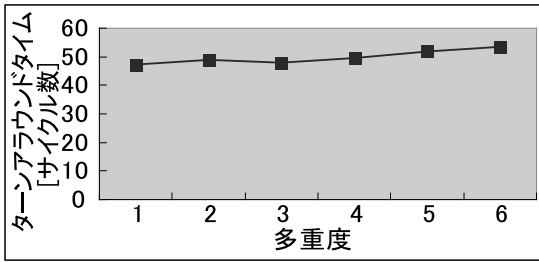


図 10 処理多重度に対するターンアラウンドタイム

Fig. 10 Turnaround time with thread multiplex.

最後に、多重度とターンアラウンドタイムとの関係を図 10 に示す。横軸に多重度を取り、縦軸は対応する多重度で処理を行った際のターンアラウンドタイムを単位をサイクル数として示している。これらの平均ターンアラウンドタイムは 50 サイクル程度と計算される。多重度が 1 のターンアラウンドタイムと、多重度が 6 のターンアラウンドタイムの差が 6 サイクル程度であり、平均ターンアラウンドタイムは 50 サイクルであることを考えると、多重度については 1 から 6 の 6 倍になっているが、ターンアラウンドタイムについては 1 割程度しか増加していない。

以上、MPLS のヘッダ処理を用いた実験結果についての考察を行った。まず、CUE-v2 への入力間隔と多重度の関係から、入力データ速度が 320 Mbps 以下であれば、最高 5 多重処理を行うこと可能であると分かった。さらにターンアラウンドタイムは多重度による変動が少なく、つねに一定で処理されることが確認できた。

4.4 CUE-v3 の性能予測

4.2 節で得られた実験結果を基に、CUE-v3 の性能予測を行う。先にも述べたように、現在、動作周波数が 100 MHz であった CUE-v2 を基に PE を開発しており、目標動作周波数 200 MHz、4 PE の CMP 構成の設計を行っている。CUE-v3 において、インターコネクションでのペナルティが 0 であると仮定すると、ターンアラウンドタイムは一定、かつ先の実験において測定された結果が実現されるため、多重度については、同時に $5[\text{多重}] \times 4[\text{PE}] = 20$ 多重、入力データ速度については $320 \text{ Mbps} \times 4[\text{PE}] \times 2[\text{倍周波数}] = 2.56 \text{ Gbps}$ 程度を受けることが可能であると予測される。1 Gbps 以上のデータ入力を受けることが可能であるということは、ギガビットイーサネットワーク環境下でネットワークプロセッサとして用いることが可能である。また、その際同時に 20 コネクションを管理し、かつターンアラウンドタイムが一定で処理可能であると予想される。

次に今回用いたプログラムに対し、実際に PE 間通信を行った場合の予測を行う。新命令の実行サイクルは、(PE における命令実行サイクル) + (PE からインターコネクションへのデータ受け渡し) + (インターコネクション通過ペナルティ) + (インターコネクションから PE へのデータ受け渡し) となる。実際にどのような通信が行われるかは、先の実験に PE 間通信に相当するスレッド起動命令を付加して CUE-v2 上でシミュレーションを行い推定した。

PE からインターコネクション部分へデータを渡すために 1 サイクル、インターコネクションから PE へのデータを渡すために 1 サイクル、さらにインターコネクションネットワーク通過ペナルティを、隣接ノードへは 1 サイクル費やされると仮定し、そのペナルティを予測する。最短である 1 hop ノード、つまり隣接ノードへパケットを渡す場合は、(PE からインターコネクションへのデータ受け渡し) + (インターコネクション通過ペナルティ) + (インターコネクションから PE へのデータ受け渡し) の計 3 サイクル分のペナルティが発生する。また最も遠い 3 hop ノードへデータを渡す場合には同様の計算により、5 サイクルのペナルティが発生する。これらの見積もりを基にすると、インターコネクションネットワークでは平均 4 サイクル程度のペナルティが発生するということになる。先の実験において用いた MPLS ヘッダ処理を行うプログラムを基に 1 命令にかかるサイクル数を計算すると、 $3.7[\text{サイクル}/\text{命令}]$ と計算される。これより、PE 間通信命令の実行に必要な実行サイクルは、 $3.7[\text{サイクル}] + 4[\text{サイクル}] = 7.7[\text{サイクル}]$ と計算でき、これはおよそ 2 命令分に相当すると見なせる。

このオーバーヘッドはスレッド自体の処理に比べて小さいうえに、大きく変動することがなく完全に予測可能である。しかも PE 内でも PE 間でもほぼ同じプログラミングモデルで動作するため、PE 数によって処理を代える必要がない。したがって、CUE-v3 においては CMP 全体を 1 つのプロセッサと見なして、その資源を使いきるまでスレッドを起動していくことができる。

次に PE 数増加により、スループットはどのように変化していくかについて考察する。本論文においてベンチマークとして取り上げた MPLS ヘッダ処理は、各ヘッダの処理を 1 PE に閉じて行うことを前提とするものである。このモデルにおいてインターコネクションネットワークにおけるペナルティは、外部へ処理結果を出力する場合に影響を与える。図 11 は前述したようなペナルティ仮定のもとで、先に述べた双方向き

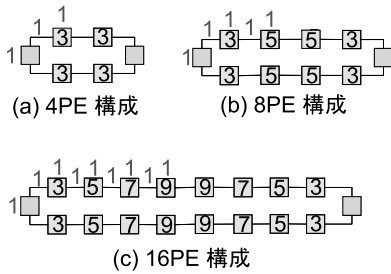


図 11 PE と外部 I/F 間通信ペナルティ
Fig. 11 Inter-processor communication costs.

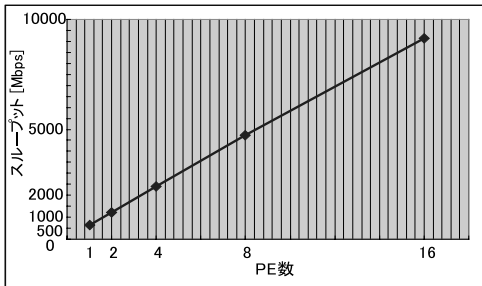


図 12 スケーラブルな性能向上
Fig. 12 Scalable performance improvement.

ング構成のインターコネクションネットワークにおいて PE 数を増した場合、PE と最も近い外部 I/F 間の通信ペナルティ変化について示したものである。数字が記入されていない箱は外部 I/F を表しており、内部に数字が記入された箱は PE を表し、その数字は最も近い外部 I/F 間通信とのペナルティ数を示している。これより、4PE 構成では平均ペナルティは 3 サイクルとなり、同様に 8PE 構成では 4 サイクル、16PE 構成では 6 サイクルと見積もることができる。このペナルティの見積りも考慮し、PE 数を増加させた場合、それともなう MPLS ヘッダ処理のスループットの変化の様子を図 12 に示す。この結果、CUE-v3 において PE 数を増加させスレッドの多重処理により、スケーラブルにスループットを向上していくことが可能である。

5. まとめと今後の課題

本論文では、ハイブリッドプロセッサ CUE-v2 を PE としたイベント駆動型 CMP CUE-v3 のアーキテクチャとその性能予測結果について述べた。

まず、CUE-v2 についてその概要と特徴を述べ、これを CMP 化した CUE-v3 について、その拡張部分を示した。次に、MPLS のヘッダ処理をベンチマークとして用いて行い、CUE-v2 の多重処理性能、ターンアラウンドタイムの 2 点についてその性能を明らかに

した。この結果、多重度が 5 程度までなら、CUE-v2 はターンアラウンドタイム一定で処理を行えることを示した。

さらに、PE 間通信を行う新命令とインターコネクションネットワークの特性を考慮し、CUE-v3 の性能予測を行った。これにより、多重度についてはスケーラブルな性能向上が予測でき、データ駆動型プロセッサのスケラビリティの高さを生かせる見通しを得られた。

以上の結果をふまえて、現在 2007 年度中の試作を目指して CUE-v3 を設計中である。今後は、さらに設計を詳細化するとともに、制御駆動スレッドを含めたベンチマークプログラムを用いた性能評価等を行っていく予定である。また、さらなる発展としてより深いパイプラインや、広帯域共有メモリシステムの付加を検討していく。

ソフトウェア開発環境については、現在のところ、アセンブラ・ローダのみの未整備な状態にあるが、CUE-v3 の開発に並行して、従来、CUE-p ならびに CUE-v1 を対象に開発してきた実時間実行システム RESCUE (Realtime Execution System for CUE-series data-driven processors)^{12),13)} の拡張を図りたいと考えている。

謝辞 本研究の一部は、総務省戦略的情報通信研究開発推進制度 (SCOPE)、半導体理工学研究センター (STARC)、ならびに文部科学省の支援による共同研究によって遂行したものである。また、本論文中における CUE-v2、CUE-v3 の開発、および性能評価に用いたツールは、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、シノプシス株式会社の協力により提供されたものである。

参考文献

- 1) Intel Corporation: Intel IXP2350 Network Processor Product Brief.
- 2) Cisco 7200 Series Routers Data Sheets.
- 3) Tullsen, D.M., Eggers, S. and Levy, H.M.: Simultaneous Multithreading: Maximizing on Chip Parallelism, *Proc. 22nd Annual International Symposium on Computer Architecture*, Santa Margherita Ligure, Italy, pp.392-403 (1995).
- 4) Olukotun, K., Nayfeh, B.A., Hammond, L., Wilson, K. and Chang, K.: The Case for a Single-Chip Multiprocessor, *Proc. 7th International Symposium*, Cambridge, MA, pp.2-11 (1996).
- 5) Crowley, P., Ficuzynski, M.E., Baer, J.-L. and

Bershad, B.N.: Characterizing Processor Architectures for Programmable Network Interfaces, *Proc. 2000 International Conference on Supercomputing*, Santa Fe, New Mexico, pp.54–65 (2000).

- 6) Ito, S., Kurebayashi, R., Tomiyasu, H. and Nishikawa, H.: A Processor Architecture for Simultaneously Processing Dataflow and Control-flow Threads, *Proc. 15th IASTED International Conference on Parallel and Distributed Computing and Systems*, pp.339–344 (2003).
- 7) 青木一浩, 工藤慎也, 西川博昭: ボトルネックのないレイヤ 2/3 間インタフェースのデータ駆動型実現法とその実験的検討, 電子情報通信学会論文誌 D-I, Vol.J87-D-I, No.5, pp.591–598 (2004).
- 8) 伊藤伸也, 野本祥平, 富安洋史, 西川博昭: データ駆動・制御駆動スレッドを同時・多重処理するプロセッサ CUE-v2 の LSI 試作, 電子情報通信学会論文誌 D-I, Vol.J88-D-I, No.2, pp.113–124 (2005).
- 9) Nishikawa, H.: Design Philosophy of a Networking-Oriented Data-Driven Processor: CUE, *IEICE Transactions on Electronics*, Vol.E89-C, No.3, pp.221–229 (2006).
- 10) Nellans, D., Balasubramonian, R. and Brunvand, E.: A Case for Increased Operating System Support in Chip Multi-Processors, *2nd IBM Watson Conference on Interaction between Architecture, Circuits and Compilers (P=ac2)*, Yorktown Heights (Sept. 2005).
- 11) Rosen, E., et al.: IETF RFC 3031.
- 12) 西川博昭, 我孫子泰祐: タグ操作を許すデータ駆動プログラムの開発・再利用支援手法, 電子情報通信学会論文誌 D-I, Vol.J85-D-I, No.3, pp.294–302 (2002).
- 13) 樽林亮介, 西川博昭: データ駆動プロセッサによる実時間処理のためのプログラム割当手法, 電子情報通信学会論文誌 D-I, Vol.J86-D-I, No.10, pp.721–732 (2003).

(平成 19 年 1 月 22 日受付)

(平成 19 年 5 月 9 日採録)



富安 洋史 (正会員)

平成元年九州大学工学部電気工学科卒業。平成 3 年同大学大学院総合理工学研究科修士課程修了。博士(工学)。九州大学助手を経て、現在、筑波大学大学院システム情報工学研究科講師。並列計算機アーキテクチャの研究に従事。IEEE 会員。



岡本 政信 (学生会員)

平成 17 年筑波大学第三学群情報学類卒業。平成 19 年同大学大学院博士前期課程修了, 同年同大学院博士後期課程に進学。現在同課程に在学中。並列計算機アーキテクチャの研究に従事。プロセッサアーキテクチャ, ネットワーク環境等に興味を持つ。IEEE 学生会員。



西川 博昭 (正会員)

昭和 51 年大阪大学工学部電子工学科卒業。昭和 59 年同大学大学院工学研究科博士課程修了。工学博士。日本学術振興会奨励研究員, 大阪大学助手, 講師, 筑波大学助教授を経て、現在、筑波大学大学院システム情報工学研究科教授。平成 6 年 7 月~7 年 8 月, 平成 9 年 11 月~12 月, 平成 10 年 4 月~5 月 MIT 招聘研究員, 平成 10 年 3 月~4 月 USC 招聘教授。データ駆動型超分散システムとその仕様記述環境等の研究に従事。平成 15 年 IASTED Best Paper Award in the area of Processor Architecture 受賞。電子情報通信学会会員, IEEE シニア会員。