Vol. 42 No. SIG 9(HPS 3) 情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム Aug. 2001

オンチップマルチプロセッサ型データ駆動アーキテクチャの評価手法とその実験的検討

浦田卓治[†] 樽林亮介[††] 西川博昭[†††]

筆者らは，VLSI向きアーキテクチャとして，自己同期式エラスティックパイプラインによる動的データ駆動プロセッサのオンチップマルチプロセッサシステムを研究している．これまでの研究で，パケットの待ちが生じない定常的なデータ流量を維持するパイプライン構成が，性能向上における課題となっている．この解決には，パイプライン構成をチューニングできるプロトタイピング環境が不可欠である．本論文は，プロセッサアーキテクチャのエミュレーション環境の実現に向け，パイプライン上のパケット流を決定する転送制御機構間の通信タイミングの模擬を取り上げ，動的データ駆動プロセッサの細粒度並列処理を活用した実現法を提案している．本手法では，エミュレーションの高効率化のため，パイプライン上の個々のパケットについて，パケットを処理・転送するデータパスとパケット転送を制御するタイミングパスとを可能な限り並列に評価している．本論文では，本手法の高い並列処理性，および，オーバヘッドのない細粒度並列処理とプロセッサ通信から得られるスケーラビリティを実験的検討を通じて明らかにする．さらに，具体的なアプリケーションを対象としたエミュレーションを通じて，対話的なプロトタイピング環境に望まれる適切な応答時間を実現可能なことを示している．

An Evaluation Scheme for Super-integrated Data-driven Architecture and Its Experimental Study

Takuji Urata,[†] Ryosuke Kurebayashi[††]
and Hiroaki Nishikawa[†††]

The authors have been studying super-integrated data-driven processors based on the self-timed elastic pipeline scheme. Past studies showed necessity of a prototyping environment to tune the pipeline for maintaining high performance. This paper describes an evaluation scheme of transmission timing of each data-packet between the self-timed transfer control mechanisms as an emulation facility for the prototyping environment. To emulate the elastic pipeline efficiently, this scheme utilizes fine-grained parallelism of the dynamic data-driven processors and evaluates data-path and timing-path in parallel. Experimental results show the high parallel processing capability and scalability based on this scheme. And an emulation result of an actual application demonstrates that this scheme satisfies quick response time required in an interactive prototyping environment.

1. まえがき

これまでの電話網，放送網，およびインターネットが統合され，単一のマルチメディアネットワーキングへ移行しようとしている．それにともない，多重化されたコネクションごとの品質保証が求められるため，ネットワーク技術のみならず，コネクションを管理するプロセッサの処理能力向上も不可欠になってきている．

ネットワーク技術に関しては，WDM（Wavelength Division Multiplex）[1]等の伝送路における多重化技術，Int–Serv[2]，Diff–Serv[3]等のプロトコル技術の進展により，通信容量，通信品質が飛躍的に向上している．一方プロセッサ技術は，微細化加工技術の発展を中心に，定型処理については目覚ましい性能向上が認められるものの，本質的には逐次処理方式のノイマン型プロセッサを踏襲している．そのため，コネクションの確立・管理のような，非定型に生じる処理要求に対しては，依然として文脈切替えによる擬似多重処理で対

† シャープ株式会社IC開発本部
Integrated Circuits Development Group, SHARP Corporation
†† 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

応せざるをえない．通信処理を擬似多重に実行した場合，ネットワークからのデータ入力に対する割込み・文脈切替えのオーバヘッドが顕著になり，文脈数の増加につれて各文脈の処理時間が指数関数的に増大する恐れがある[4]．このため，通信処理の高速化を目的に，ソフトウエアのファーム化や ASIC によるハードウエア化の試みがなされているが[5]，一方で，ネットワーク向きプロセッサを研究する動きがみられる．

筆者らは，非定型処理の高効率化手法として，必要なデータが揃い次第，命令実行を可能とするデータ駆動原理に着目し，動的データ駆動プロセッサアーキテクチャCUE（Coordinating Users' requirements and Engineering constraints）を実現してきた[6]．その実装では，自己同期式エラスティックパイプライン[7]の採用により，クロック配線に関する制約の緩和とプロセッサ間の直接接続を可能にし，オンチップマルチプロセッサによる自然な処理能力向上という特長を獲得した．この VLSI 向きの特長から，オンチップマルチプロセッサ構成のカスタマイズによる，アプリケーションに即した少量多品種のチップ開発を想定している．

しかしながら，自己同期式エラスティックパイプラインの処理能力を活用するには，パケットの渋滞が生じない範囲でパイプライン上のデータ流量を高め，短いターンアラウンドタイムと高いスループットを両立させなければならない．この条件を満たすよう，アプリケーションプログラムのチューニングを行い，あるいはオンチップマルチプロセッサの構成をカスタマイズするためには，パイプラインステージ水準のプロトタイピングが不可欠である．本論文は，自己同期式エラスティックパイプラインのエミュレーションを並列化し，データ駆動アーキテクチャにマッピングする手法を提案する．

2. VLSI 向きデータ駆動アーキテクチャ

2.1 CUE アーキテクチャ

CUE 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アーキテクチャである．オンチップで相互接続される PE（Processing Element）は，世界最初のデータ駆動プロセッサであるマンチェスターデータ駆動計算機[8]と同様に，環状パイプライン構造をとる．**図 1** に示すとおり，1 つの環状パイプラインは，合流部（Joint），発火制御部（Firing Control; FC），演算部（Functional Processor; FP），命令記憶部（Program Storage; PS），分流部（Branch）の 5 つの機能ブロックから成る．

PE の入出力および内部のデータはすべて，命令実行のオペランド（Operand）と見なし，命令実行に必要な情報，すなわちタグ（Tag）が付帯した，パケット（*packet*）として扱う．タグの主たる内容は，命令オペコード（Opcode），および，発火制御部 FC における待ち合わせ検出用の情報である．一般に，動的データ駆動プロセッサの待ち合わせ検出には，プログラム中の各命令に一意に対応する宛先（Destination）と，同一プログラム上を並列に流れるデータを互いに識別する色（Color）が用いられるが，CUE では，色に代えて，有意の値を与えることのできる世代（Generation）を定義している．以上から，パケットの基本構成は，宛先，世代，オペコード，オペランドである．

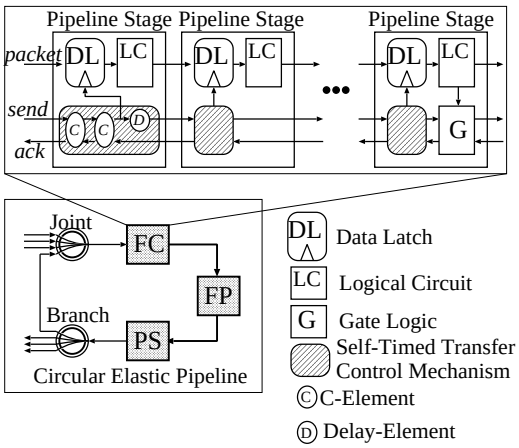


図 1 環状エラスティックパイプライン
Fig. 1 Circular Elastic Pipeline.

PE に入力されたパケットは，Joint を経て FC に入り，命令実行に必要なパケット対の発火検出が行われる．発火したパケット対のオペランドは単一のパケットにまとめられ，FP で命令が実行される．PS は，次の宛先とオペコードをフェッチするとともに，パケットの複製・消去を行う．PS から出力されたパケットは，Branch を経て，次の命令を割り当てられた PE に入力される．CUE では，PS をパイプライン終端に配置し，かつ PE 間のルーティング条件にオペコードを含めることで，PE ごとに異なる命令セットを持つ，ヘテロジニアスなマルチプロセッサを実現している．

さらに CUE は，自己同期式エラスティックパイプラインによるスーパーパイプライン構造を採用する．エラスティックパイプラインは，図 1 に示すように，自己同期式転送制御機構（Self-Timed Transfer Control Mechanism）が生成する局所的なクロック信号に基づきパケットを転送する．自己同期式転送制御機構は，送信要求（*send*）と送信許可（*ack*）とのハンドシェイクを行う C 素子（C–Element），および，データラッ

チ（Data Latch）間の論理回路（Logical Circuit）におけるパケット処理時間を保証する遅延素子（Delay–Element）から成り，近傍のステージの空き状態のみに従い，自律的にパケットの転送タイミングを決定する．また，FC におけるパケットの待ち合わせや，PS におけるパケットの複製・消去のように，論理回路の評価結果に応じてパケット流に変動を生じさせるパイプラインステージには，ゲートロジック（Gate Logic）を設け，*send*/*ack* 信号を制御する．

エラスティックパイプラインでは，データラッチへのクロック配線が極小化されるため，クロックスキューが微細化の障害となりにくい．また，パイプライン構成のカスタマイズに際しては，クロック配線の変更が極小化される．

2.2 パケット転送タイミングの評価の必要性

CUE による多重処理では，データは属する文脈をマッピングした世代とともにパケット化する．文脈切替えのたびにレジスタセットの退避・復元が必要なノイマン型プロセッサと異なり，命令実行に必要なデータをパケット内に自己充足する CUE は，文脈切替えのオーバヘッドがないため，1 つのパイプライン上に異なる文脈のデータを混在させることができる．ゆえに，パイプライン資源が十分に確保され，競合が生じない限りにおいては，パケットはパイプラインの最小転送時間，すなわち，*send* 信号の転送時間で流れる．このとき各文脈のターンアラウンドタイムは，プログラムのクリティカルパスの処理時間に依存する．

一方，環状パイプラインが入出力のための分流と合流を必然的に含むため，パケット流量の変動や，合流時のブロッキングが生じる可能性は避けられない．また，PE の入出力ポート数が Joint/Branch で制限されるため，通信経路に制約がある．これらの要因は，*send* 信号を送信してから *ack* 信号を受信するまでの待ち時間を増加させ，ターンアラウンドタイムを伸長させる可能性がある．

したがって，多重処理環境下での実時間処理を実現するためには，パイプライン上の定常的なデータ流量を維持しなければならない．そのプロトタイピング手法の 1 つとして，エラスティックパイプラインの動作特性を決定する，自己同期式転送制御機構によるパケット転送を，パイプラインステージ水準で模擬する手法を提案する．

3. エミュレータのデータ駆動型実現法

3.1 エラスティックパイプラインの評価モデル

エラスティックパイプラインの *send*/*ack* 信号の各転送時間に基づくパケット転送を模擬するには，個々のパイプラインステージを，互いに *send*/*ack*/*packet* メッセージを通信し合う，自律したモジュールとしてモデル化するのが最も直接的な手法である（**図 2** (a)）．このモデルでは，*send*/*ack* メッセージのモジュールへの到着時刻さえ分かれば，パケットの転送タイミングを決定できる．したがって，*send*/*ack* メッセージにタイムスタンプを付与することで，ある時刻におけるパケットの位置を評価する．

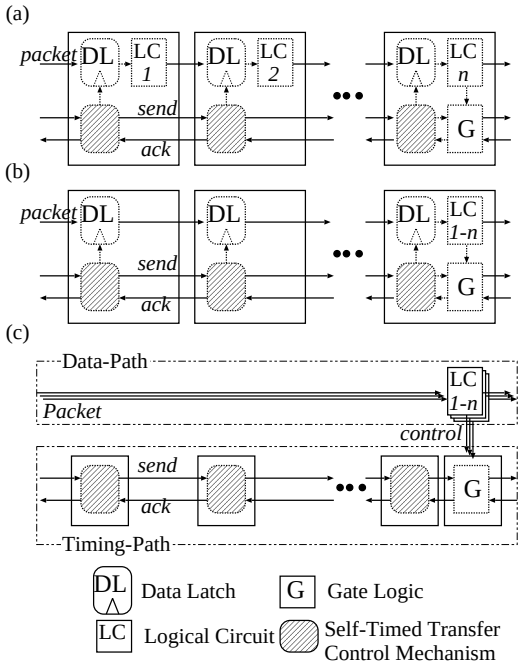


図 2 エラスティックパイプラインの評価モデル
Fig. 2 Evaluation model of Elastic Pipeline.

このとき，機能ブロックのオペレーションの模擬，すなわち *pakcet* メッセージの参照/更新については，*send*/*ack* メッセージをゲートロジックで制御できる限りにおいて，モジュールごとに細分化して行う必要はない．すなわち，図 2 (b) に示すように，ゲートロジックを扱うモジュールにおいてのみ *packet* メッセージを評価することで，モデル作成が容易になる．しかし，図 2 (b) のモデルでは，ゲートロジックを扱うモジュールがパイプラインのボトルネックとなることが明らかであり，パイプライン全体の評価効率が低下する．これらの相反する課題を解決しなければならない．

上記の課題に対し，エラスティックパイプラインが，データラッチを結ぶパケットの通信路（以下，データパス）と，転送制御機構とゲートロジックから成る *send*/*ack* 信号の通信路（以下，タイミングパス）の，

2 系統の通信路に分離できることに着目する．

データパスから見ると，タイミングパスは同期式パイプラインのクロックジェネレータに相当する．すなわち，基本的にデータパスはタイミングパスからのクロック入力を受けるのみである．例外的に，ゲートロジックではデータパスがタイミングパスを制御するが，ゲートロジック間のパケット転送に限定すれば，タイミングパスはデータパスとは独立に処理可能である．

このことから，本論文では，図 2 (c) に示すように，データパスとタイミングパスの粒度を変え，両者を可能な限り並列に評価する手法を提案する．このモデルでは，データパスは図 2 (b) と同様に機能ブロック単位で評価できるため，タイミングパスのみパイプラインステージ単位で評価すればよい．この場合，データパスの各機能ブロックで *packet* メッセージを評価した結果を，*control* メッセージとしてタイミングパスのゲートロジックに通知することで，*send*/*ack* メッセージを制御する．

3.2 応答時間の考察

環状エラスティックパイプラインの性質として，パイプライン中にパケットの存在する割合（パケット占有率）が一定値を超えるとパケットの転送時間が増加する．本評価モデルもまた，パイプラインステージ間のハンドシェイクをそのままメッセージ通信で表現するため，同様の性質を持つ．

注目するパケットが環状パイプラインを 1 周するのに要する時間（ターンアラウンドタイム）を T とする．いま，ターゲットである N 段の環状パイプライン上を n $(< N)$ 個のパケットが無限に周回しているとすると，ターゲットのパケット占有率 $o = \frac{n}{N}$ と T との関係は，次式に従う．

$$T = \max(T_s, T_{sat}, T_a), \tag{1}$$

$$T_s = \bar{s}N, \tag{2}$$

$$T_{sat} = \max(s+a)No, \tag{3}$$

$$T_a = \frac{\bar{a}No}{1-o}. \tag{4}$$

ここで，$\bar{s}$ は，1 回の *send* 転送の平均時間，$\bar{a}$ は，1 回の *ack* 転送の平均時間，$\max(s+a)$ は，1 回のハンドシェイクにかかる時間の最大値を表す．s，a は，パイプラインステージごとに，自己同期式転送制御機構中の遅延素子と配線遅延によって規程される値である．

T_s に従う領域は，パケット占有率 o が小さく，ブロッキングなくパケット転送が行われる領域である．ゆえに，T_s は，*send* 転送の平均時間 $\bar{s}$ とパイプライン長 N にのみ依存し，o によらず一定となる．また，スループット（No/T）は，o に正比例する．一方，T_{sat} に従う領域は，パイプラインのボトルネックによりパケットの渋滞が生じている領域である．すなわち，$o > \bar{s}/\max(s+a)$ となると，渋滞によってパケット転送に *ack* メッセージの待ち時間が生じるようになり，ボトルネック部分のパケット転送レートでスループットが制限されるため，ターンアラウンドタイムは増加に転じる．また，T_a に従う領域は，パケット占有率 o が高まり，*ack* メッセージの処理時間が支配的になる領域を示す．すなわち，$o > (\max(s+a)-a)/\max(s+a)$ となると，スループットも o の増加にともない減少する．

以上は実パイプラインの性質であるが，パイプラインエミュレーションの評価時間もこれに準じる．すなわち，上記の式に対して，$\bar{s}$ に *send* メッセージの平均処理時間，$\bar{a}$ に *ack* メッセージの平均処理時間，$\max(s+a)$ にハンドシェイクのエミュレーションの最大処理時間を代入すると，ターゲットのパケット占有率 o に対する，理想的な評価時間 T_{ideal} が求まる．

3.3 評価モデルのデータ駆動型実現法

筆者らは，前節の評価モデルの実現手法として，評価モデルをデータ駆動プロセッサ自身にマッピングする，自己プロトタイピング手法を提案する．特に本節では，評価モデルの並列性を最大限に活用できる，世代の割当て手法を提案する．ここでは，説明のため，エミュレーションのターゲットを，仮想プロセッサ，仮想モジュール，仮想パケットのように語頭に仮想を付けて表現する．一方，エミュレーションプログラムを実装するプロセッサに関しては，実プロセッサ，実パケットのように語頭に実を付けて表現する．また，本手法は，仮想モジュール間のメッセージを CUE の実パケットで表現する．ここで扱うメッセージとは，データパス中の仮想パケットを表す *packet*，仮想パケットの評価結果を表す *control*，および，タイミングパス中のハンドシェイクに用いる *send*，*ack* である．

CUE で相互干渉なく並列処理するには文脈ごとに一意の世代を与えればよいが，性質の異なるタイミングパスとデータパスに対し，文脈も異なる手法で与える．以下，CUE の実パケットを $name(generation, data)$ と表記する．$name$ は実パケットが表すメッセージ名，$generation$ は世代，$data$ はオペランドを表す．また，転送制御機構やパイプラインステージ間のゲートロジック等，すべての仮想モジュールに一意の識別子 ID_f を与えておく．

3.3.1 データパス中の世代割当て

データパスのエミュレーションでは，同時に複数の *packet* メッセージを評価するため，メッセー

ジごとに異なる文脈とすべきである．したがって，メッセージごとに一意の識別子 ID_p を，仮想モジュールの識別子 ID_f とともに，メッセージを表現する実パケットの世代とする．すなわち，データパス中のメッセージは，$packet((ID_f, ID_p), data)$，$control((ID_f, ID_p), data)$ と表現できる．ただし，処理の局所性から ID_p は仮想プロセッサ内で一意とし，仮想プロセッサの命令記憶部（PS）の評価時に更新するとした．

3.3.2 タイミングパス中の世代割当て

仮想プロセッサの 1 つのパイプラインステージには，たかだか 1 個の仮想パケットしか存在しない．そのため，エミュレーションプログラムでは，仮想プロセッサのパイプラインステージごとに一意の世代をマッピングすれば，すべてのメッセージを並列に評価できる．したがって，タイミングパスのエミュレーションにおいては，ID_f を実パケットの世代とし，メッセージの生成時刻を表すタイムスタンプ t を実パケットのオペランドとする．また，データパスから適切な $control$ メッセージを受けられるよう，$send$ メッセージには対応する $packet$ メッセージの ID_p を，実パケットのオペランドとして与えておく．すなわち，タイミングパス中のメッセージは，$send(ID_f, (t, ID_p))$，$ack(ID_f, t)$ と表現できる．

この世代の割当てに基づき，タイミングパスにおけるハンドシェイク処理をデータ駆動図式（Data-Driven Scheme; DDS）[9]で表したものを，**図 3** に示す．まず，$send(ID_f, (t, ID_p))$ と $ack(ID_f, t)$ の同期をとった後，互いの t を比較し，より大きな t をハンドシェイクの成立時刻と見なす（図中，n0）．CUE では，この同期処理は発火制御部（FC）における実パケットの待ち合わせとして実現でき，さらに，引き続き演算処理部（FP）で t の比較演算を実行することで，同期検出から比較までの一連の処理を 1 命令で実現する．次に，テーブル（Send/Ack Delay）を参照し，ステージごとの転送時間を t に加算する（図中，n1）．最後に $send/ack$ メッセージをそれぞれ隣接する仮想ステージまたは仮想ゲートロジックに転送するが，これは，仮想パイプラインの構成を保持するテーブル（Send/Ack Pipeline Connection）に基づく ID_f の更新と，ID_f による分岐によって実現される（図中，n2–3）．

さらに，エミュレーションプログラムにおいて，$send$ メッセージの世代に ID_p を一時的に与えることで，仮想ゲートロジックにおける $send$ メッセージと $control$ メッセージの同期処理もまた，実パケットの待ち合わせとして簡潔に実現できる．

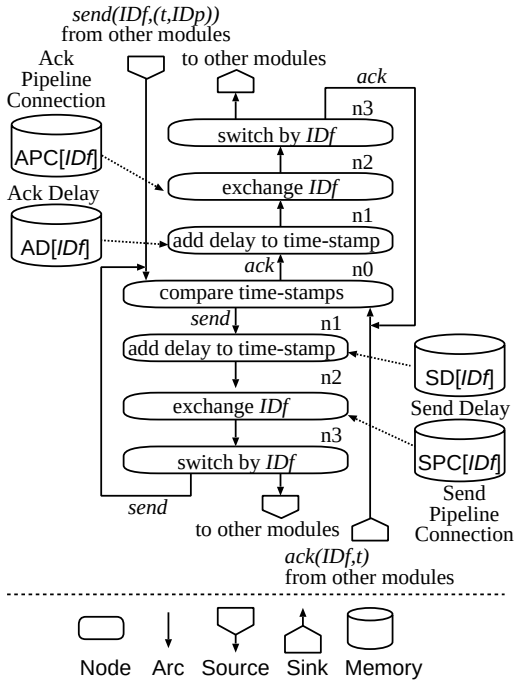


図 3 データ駆動図式による，自己同期式転送制御機構の動作記述
Fig. 3 A behavioral description of Self-Timed Transfer Control Mechanism with data-driven scheme.

3.3.3 *null* メッセージ

本手法では，仮想パイプライン中の仮想 Branch が 1 経路にのみ $send$ メッセージを送信する一方，仮想 Joint はすべての $send$ メッセージが揃うまで評価できない．ゆえに，適当な外部入力がない場合に仮想 Joint でデッドロックを生じる可能性を避けるため，指定時刻までメッセージが到着しないことを保証する $null$ メッセージ[10]を用いた．これにより，一度仮想 Joint にすべてのメッセージが揃えば，すべての経路につねに 1 つ以上のメッセージが生成されることが保証されるため，デッドロックは回避される．

本手法では，$null$ メッセージを特殊な $send$ メッセージとして扱う．タイミングパスにおいて $null$ メッセージと ack メッセージの同期が行われた際，ack メッセージは消費せず，$null$ メッセージのみ次のステージへ転送する．さらに，$send$ メッセージと $null$ メッセージは互いに追い越してはならないが，仮想パイプラインの渋滞や仮想 Joint での待ち合わせによって，$send$ または $null$ メッセージが先行する $null$ メッセージに追いついた場合，先行する $null$ メッセージの破棄を許すとした．これにより，$null$ メッセージ数の爆発が自律的に防がれる．

表 1 パケット流情報
Table 1 Packet Flow Information.

行き先 PE 番号（PE#）
宛先ノード番号（Node#）
世代（Generation）
オペコード（Opcode）
複製・消去の有無（Copy Flag）

3.4 パケット流情報

データパスでは，待ち合わせに入ることによる仮想パケットの消去（FC），オペランドの複製による新規仮想パケットの生成（PS），および，演算結果と命令記憶に依存する仮想パケットの分岐（FP，PS，Branch）を決定し，その結果を *control* メッセージとして仮想ゲートロジックに通知する．これらの「振舞い」は本来，各機能ブロックにおいて，*packet* メッセージおよび機能ブロック固有のメモリ内容から決定される．しかしながら，プログラムの論理的正当性はその実行ハードウエアと切り離せることから，仮想パケットの振舞いを，エミュレーション実行前にプログラムと入力パターンに基づいて評価しておくことで，データパス評価の高効率化を図る．ここでは，エミュレーション前に得ておくべき情報は，**表 1** に示す項目に限定される．これらを，パケット流情報（Packet Flow Information）と呼ぶ．

3.5 データ駆動型実現法の利点

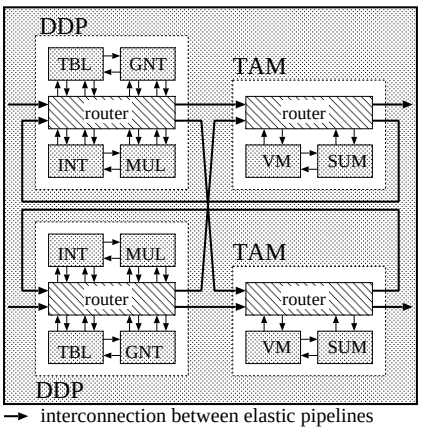
本論文がターゲットとするエラスティックパイプラインは，プロセッサどうしの接続を含むすべての隣接するステージ間に *send* と *ack* のハンドシェイクが存在する．そのため，並列エミュレーションに際しては，どのように分割しても細粒度のメッセージ通信が残る．ゆえに，細粒度のプロセッサ間通信が必須である．

CUE による並列エミュレーションの実現は，(1) PE 間通信は，互いのエラスティックパイプラインを直接に接続するため，通信処理のオーバヘッドがなく，(2) データ駆動原理の受動的性質により，PE 間メッセージの受信から処理の起動までのオーバヘッドがないことから，高いスケーラビリティが期待できる．

4. エミュレーション方式の実験的検討

4.1 実験目的

ここでは，3 章に述べたエミュレーション手法の有効性を，実験的に検証する．すなわち，(1) データパスとタイミングパスの並列評価を効率良く実現可能なこと，(2) 複数の仮想パケットの並列評価が効率良く実現され，かつ，計算資源に対するスケーラビリティを得られること，および，(3) 対話的プロトタイピング環境として妥当な応答時間を実現可能なことを示す．



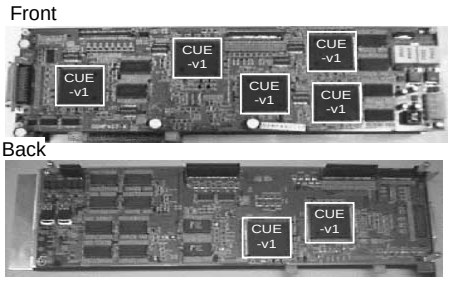
DDP : super-integrated Data-Driven Processor
composed of 4 heterogeneous processing elements
INT : Integer logical operation, branch control
MUL: Multiplier, static accumulator
GNT: Generation number manipulation
TBL : Lookup table reference
TAM: Tag Addressable Memory
composed of 2 heterogeneous processing elements
VM : Video memory reference
SUM: Summation

図 4 CUE–v1 ブロック図
Fig. 4 Block diagram of CUE–v1.

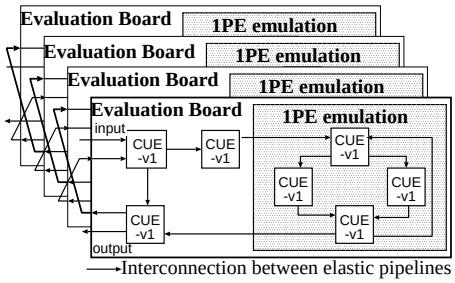
4.2 実験環境

実験には，CUE アーキテクチャに基づく，ヘテロジニアスな 12 の PE から成るオンチップマルチプロセッサ CUE–v1 を用いた．CUE–v1 は，**図 4** のブロック図に示したように，DDP（super-integrated Data-Driven Processor）および TAM（Tag Addressable Memory）の 2 種類のオンチップマルチプロセッサをさらに集積したもので，INT，MUL，GNT，TBL，VM，および SUM の 6 種の PE から成る．ここでは 6 種の PE のうち，実験でエミュレーション対象とする INT，MUL，GNT についてのみ説明するが，CUE–v1 の詳細は文献 6)，11) を参照されたい．INT は整数・論理演算を行う PE，GNT は世代操作用 PE，MUL は乗算・静的累算用 PE である．これら 3 つの PE は，いずれも図 1 の構成をとり，環状パイプラインの長さは，INT が 19 段，GNT が 25 段，MUL が 23 段である．CUE–v1 プロセッサ内のパケット転送レートの最大値は，実測で約 180 MHz である．

図 5 に，実験環境の全体図を示す．7 つの CUE–v1 を実装した同図 (A) の評価ボードを，同図 (B) の構成で 4 枚接続した．本章の実験は，すべてこの構成で行っており，1 枚の評価ボードで 1 つの仮想 PE をエミュレーションする．各評価ボード上では，図 5 の網



(A)Overview of Evaluation Board



(B)Experimental Environment for 4PEs Emulation

図 5 実験環境の全体図
Fig. 5 Overview of emulation system.

掛けで示した 4 つの CUE–v1，すなわち 4 × 12PE でエミュレーションプログラムを実行し，残る 3 つの CUE–v1 は，評価ボード間のパケット転送と，テストパターンの入力処理のみに使用した．仮想 PE 間のルーティングは，エミュレーションプログラムによりソフトウエア的に制御しているため，仮想 PE 間の接続構成が評価ボード間の物理的な接続構成に制約されることはない．このように，1 つの仮想 PE のエミュレーションにつき 4 つの CUE–v1 を使用するのは，CUE–v1 固有の資源制約が，評価モデルの並列性を損なわないようにするためである．すなわち，評価ボード上の CUE–v1 のパケット占有率が 3.2 節に述べた T_{sat}，T_a 領域に達することによる，エミュレーション効率の低下を避けた．なお，エミュレーション対象の仮想 PE は，エミュレーション結果を実機の動作と比較できるよう，CUE–v1 を構成する PE のうちから取り上げる．

仮想 PE の構造データおよびターゲットプログラムは，CUE–v1 内部のメモリに分散配置しておく．テストパターンは，入力処理専用に割り当てた CUE–v1 上に貯めておき，順次投入していく．応答時間は，処理の開始時に実パケットを 1 つ出力し，最後の結果出力までの間隔を外部から計測して求めた．

本実験に固有の制約を述べておく．実験に用いた CUE–v1 は，本来はマルチメディアネットワーキング用途であり，メディアストリームの表現手法である，連続した世代を持つ実パケットの入力に対して，最も効率的に FC が機能するようチューニングされている．そのため，識別子 ID_p，ID_f のような，連続性が保証されない世代の場合，処理効率の著しい低下を招く．ゆえに，本実験では，連続した世代への動的変換，および，FC の容量に合わせた実パケットの分岐・分配処理をエミュレーションプログラムに追加した．これらのオーバヘッドによりターンアラウンドタイムは増加するが，すべて実パケットごとに独立して施される処理であるため，本論文の主題である，データ駆動型エミュレーション手法の並列処理性およびスケーラビリティを損なうものではない．

表 2 データパスとタイミングパスの処理時間の比較
Table 2 Evaluation time of Data-Path and Timing-Path.

機能ブロック（パイプライン段数）		データパス	タイミングパス
Joint	(4)	1.46	6.62
FC	(11)	1.37	12.19
FP	(1)	0.52	2.42
PS	(1)	2.98	2.46
Branch	(2)	0.74	3.14
計	(19)	6.34	26.83

4.3 評価結果

4.3.1 データパスとタイミングパスの並列評価

本実験は，1 つの仮想パケットが仮想環状パイプライン上で周回したときの，データパスとタイミングパスそれぞれの処理時間を，機能ブロックごとに測定した．評価対象となる仮想 PE には，CUE–v1 を構成する PE のうち，パイプライン段数が 19 段と最も短い INT を選択した．

表 2 に，機能ブロックごとの評価時間を，仮想パイプライン 1 段の最小タイミングパス評価時間を 1 として正規化した値で示す．ここで最小タイミングパス評価時間とは，仮想パイプラインに十分に空きがある状態で，タイミングパス中の自己同期式転送制御機構が *send* メッセージを受信してから，出力となる *send*/*ack* メッセージを送信し終えるまでに要する最小時間である．PS 以外では，機能ブロック単位で，データパスの処理時間がタイミングパスの処理時間より短いことが分かる．

4.3.2 細粒度並列処理のスケーラビリティ

前節で示したように，エミュレーションの効率は，タイミングパスの処理時間に依存する．したがって本実験では，タイミングパスにおいて，各パイプラインステージのハンドシェイクを効率良く並列評価できることを，理論値と実験結果の比較から明らかにする．INT をターゲットとし，仮想環状パイプライン上を周

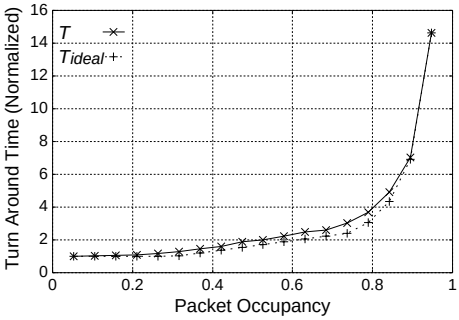


図 6 1PE エミュレーションのターンアラウンドタイム
Fig. 6 Turn around time of emulation (1PE).

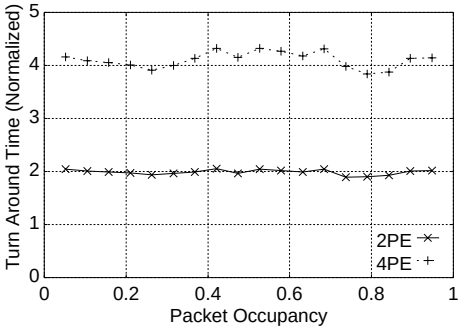


図 7 資源追加に関するスケーラビリティ
Fig. 7 Scalability in resource extention.

回する仮想パケット数を変化させながら，仮想パイプライン 1 周にかかる時間を計測して，仮想パイプラインのパケット占有率とエミュレーションのターンアラウンドタイムの関係を求めた．

図 6 の横軸に仮想パイプラインのパケット占有率 o，縦軸にターンアラウンドタイム $T(o)$ を，仮想パケット数 n が 1 のときの値で正規化して示す．比較のため，3.2 節で述べた $T_{ideal}(o)$ をあわせて示している．$T_{ideal}(o)$ のパラメータは実験からあらかじめ，$\bar{s} = 7.8$（μsec），$\max(s+a) = 25.44$（μsec），$\bar{a} = 6.34$（μsec）と求めた．

実験結果では，T_{sat} の領域で理想値と実験結果に若干の差が生じている．これは，*null* メッセージの処理時間が顕在化すること，および，実験環境の CUE–v1 の実パイプラインが混雑することで，実パイプラインの合流部等でブロッキングが生じることによる．これらのオーバヘッドは実験環境に依存するが，現状でターンアラウンドタイムの 26%以下に抑えられており，本手法が各仮想パケットを効率良く並列評価できているといえる．

次に，エミュレータのスケーラビリティを評価するため，評価対象の仮想 PE 数 k を 1，2，4 と変化させ，同様の実験を行った．本実験のターゲットとして，複数の INT を相互接続して形成した仮想的な環状路について，仮想パケットを無限に周回させる．先の実験と同様に，注目する仮想パケットがターゲットの環状路を 1 周する時間をターンアラウンドタイムとした．すなわち，ターゲットの環状路全体のパイプライン段数を N とすると，INT 単体のパイプライン段数が 19 であるから，$N = 19k$ となる．したがって，資源の追加がオーバヘッドにならないならば，そのターンアラウンドタイムは k に正比例する．

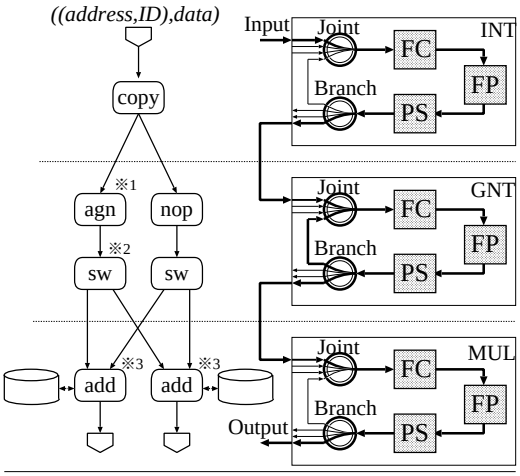
実験結果を**図 7** に示す．図 7 の横軸は仮想パイプラインのパケット占有率 o である．縦軸は，各 o における，1PE エミュレーションのターンアラウンドタイムとの比であり，1PE の場合を 1 としている．図 7 では，o に対してターンアラウンドタイムの揺らぎが現れているが，これは，*null* メッセージの消去タイミングが一定でないことに起因する．しかしながら，ほぼ仮想 PE 数 k に正比例した値を中心に揺らいでおり，資源追加に関するオーバヘッドは極小化されているとみてよい．ゆえに，本エミュレーション方式が優れたスケーラビリティを実現しているといえる．

4.3.3 エミュレーション効率

式 (1)〜(4) に CUE–v1 の *send*/*ack* 信号の転送時間を代入し，CUE–v1 に対するエミュレータのターンアラウンドタイム比を求めた．ここで，エミュレータの *send* メッセージの平均処理時間 $\bar{s}$ を 1 とした場合，CUE–v1 のパラメータは，$\bar{s} = 5.35 \times 10^{-4}$，$\bar{a} = 9.49 \times 10^{-5}$，$\max(s+a) = 7.10 \times 10^{-4}$ となる．その結果，CUE–v1 に対するエミュレータのターンアラウンドタイム比は，約 2,000 倍から約 8,500 倍と見積もられた．以下では，実際にアプリケーションをエミュレーションした場合の応答時間が，この予測値で近似できることを例証する．

ターゲットプログラムの例として，**図 8** に示す TCP/IP 通信におけるチェックサム計算処理をエミュレーションする．このプログラムは，CUE–v1 を構成する PE のうち，INT/MUL/GNT の 3 つの PE でストリーム状のパケット列を処理するものである．実行時のパケットの流れには，パケットの複製，パケットの合流および分岐，待ち合わせによるパケットの消費が含まれる．すなわち，CUE のパケット流に対するエミュレーションの主要な要素をすべて含んでいる．

このプログラムに対し，一般的な長さの TCP セグメントを入力した際の応答時間を評価した．エミュレーションプログラムの実行は，4.2 節の図 5 に示したように，1 つの仮想 PE の評価につき 4 つの CUE–v1,



※1 assign *address* at *generation* to *data*
※2 switch by *ID* at *generation*
※3 sum *data* with a memory

図 8 実プログラム例
Fig. 8 Sample application.

表 3 エミュレーション効率
Table 3 Efficiency of emulation.

入力レート※1	8.5 Mpackets/sec（136 Mbits/sec）
データ長※2	266 packets（532 bytes）
CUE–v1 での応答時間 (1)	31.77 μsec
エミュレーションでの応答時間 (2)	62.14 msec
(2)/(1)	1,956

※ 1: ATM セルのヘッダ，および IP ヘッダを除く
※ 2: TCP セグメントの標準データ長 512 bytes
＋TCP ヘッダ 20 bytes

すなわち，3 × 4 CUE–v1 を用いた．また CUE–v1 での応答時間を，1 つの CUE–v1 内の INT/MUL/GNT に図 8 に示されるとおりにプログラムをマッピングし実行することで求めた．ネットワーク層を ATM（Asynchronous Transfer Mode）と想定して実験した結果を**表 3** に示す．見積りのとおり，エミュレーションの応答時間は，実プロセッサの応答時間の約 2,000 倍と分かる．

マルチプロセッサアーキテクチャである CUE を最適化する目的に照らせば，各仮想プロセッサが最大スループットを発揮する近辺の負荷状態（パケット占有率 o がおよそ 0.7 以下）のエミュレーションが主となる．この仮定の下では，CUE–v1 に対するターンアラウンドタイムの比は，約 2,000 倍から約 5,000 倍である．エミュレーションの具体例として，ATM ネットワーク上の TCP/IP について述べれば，標準的な長さである 512 bytes 程度の TCP セグメントであれば，その CUE–v1 で実行した場合のターンアラウンドタイムは 88 μsec 程度である．このことから，TCP/IP を 440 msec 以内でエミュレーション可能である．さらに，本実験で用いた CUE–v1 に固有の制約を取り除き，負荷分散や動的な世代の更新のオーバヘッドを解消することで，エミュレーション効率はさらに向上する．

本論文に示した自己プロトタイピング手法は，動的データ駆動プロセッサの原理的な細粒度並列処理性を活用し，実プロセッサ，仮想プロセッサともに性能，規模を拡大させていく，自己発展的なエミュレーション環境を実現できる．これにより，実プロセッサに対する仮想プロセッサのターンアラウンドタイムの比をつねに維持できるため，対話的なプロトタイピング環境を十分実現可能である．

5. むすび

本論文は，パイプライン構成をチューニングできるプロトタイピング環境の実現に向け，自己同期式エラスティックパイプラインのデータ流を決定する転送制御機構間のパケット転送の評価モデルを示し，データ駆動原理に適した実現手法を提案した．

CUE アーキテクチャの特長である，データの自己充足性とエラスティックパイプラインの自律性の高さは，PE や機能ブロックの部品化を可能としている．したがって，プロセッサのチューニングに際し，既存の部品を再利用し，VLSI の最小デザインルールに応じて $send/ack$ 信号の転送時間の予測値を設定することで，パイプラインステージ水準であっても，信頼性のあるプロトタイピングが可能である．

このようなプロトタイピング環境が実現可能なのは，元来データ駆動原理が並列処理を指向し，同時に，CUE アーキテクチャも徹底的に並列処理を指向しているためである．今後，エミュレーションを経て実現した CUE プロセッサを用いてさらに次世代のプロセッサ開発が可能な，自己発展的なプロトタイピング環境の構築を目指していく．

謝辞 本研究にあたって，データ駆動プロセッサの評価システムに関して数々のご支援をいただいた，シャープ株式会社宮田宗一氏，ならびに，同木原誠一郎氏に深く感謝いたします．本研究の一部は，文部科学省科学研究費（基盤研究 B（2）11480060，萌芽的研究 11878049）の援助を受けて行ったものである．

参考文献

1) Ramaswami, R.: Multiwavelength Lightwave

Networks for Computer Communication, *IEEE Commun. Mag.*, pp.78–88 (1993).
2) White, P.: RSVP and integrated services in the Internet: A tutorial, *IEEE Commun. Mag.*, Vol.34, pp.100–106 (1997).
3) Nichols, K., et al.: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, Internet RFC 2474 (1998).
4) 石井啓之, 西川博昭, 小林秀承, 井上友二：TINA型高品質マルチメディアネットワークの実現法の検討, 信学論（B-I）, Vol.J80-B-I, No.6, pp.457–464 (1997).
5) 長田孝彦, 東海林敏夫, 山下博之, 塩川鎮夫：マルチメディアコンテンツ転送向け高性能TCP/IP通信ボードの構成と評価, 情報処理学会論文誌, Vol.39, No.2, pp.347–355 (1998).
6) Nishikawa, H. and Miyata, S.: Design Philosophy of Super-Integrated Data-Driven Processors: CUE, *Proc.1998 International Conference on Parallel and Distributed Processing Techniques and Applications*, pp.415–422 (1998).
7) 岩田　誠, 宮田宗一, 寺田浩詔：自己タイミングスーパパイプライン型データ駆動プロセッサ, 信学論（D-I）, Vol.J81-D-I, No.2, pp.62–69 (1998).
8) Gurd, J.R., Kirkham, C. and Watson, I.: The Manchester Prototype Dataflow Computer, *Commun. ACM*, Vol.28, No.1, pp.34–52 (1985).
9) Dennis, J.B.: Dataflow Schemas, *Project MAC*, pp.187–216, MIT (1972).
10) Misra, J.: Distributed discrete-event simulation, *Computing Surveys*, Vol.18, No.1, pp.39–65 (1986).
11) Muramatsu, T., Shichiku, R.T., Miyata, S. and Nishikawa, H.: Super-Integrated Data-Driven Processors Realizing Hyper-Distributed System Environment, *Proc. 1998 Int. Conf. on Parallel and Distributed Processing Techniques and Applications*, pp.461–468 (1998).

(平成13年2月 9 日受付)
(平成13年5月24日採録)

浦田　卓治（正会員）

平成7年筑波大学第三学群情報学類卒業．平成12年同大学大学院博士課程工学研究科単位取得後退学．同年，シャープ（株）入社．現在，同社IC開発本部ネットワークシステムLSI開発センターにて，データ駆動型プロセッサの設計開発に従事．電子情報通信学会，IEEE各会員．

樽林　亮介（学生会員）

平成10年筑波大学第三学群情報学類卒業．現在，同大学大学院博士課程工学研究科在学中．データ駆動プロセッサアーキテクチャの研究に従事．電子情報通信学会，IEEE各会員．

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，筑波大学電子・情報工学系教授．平成6年7月〜7年8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．昭和61年度高柳賞受賞．電子情報通信学会，IEEE各会員．