分散キューバッファを持つデータ駆動型プロセッサ Qv-x の性能評価

非会員

上方輝彦 (大阪大)

非会員

滝根哲哉 (大阪大)

非会員

岩田誠 (大阪大)

正員

寺田浩詔 (大阪大)

Performance evaluation of data-driven processor Qv-x
with distributed queue-buffers

Teruhiko Kamigata, Non-member(Osaka University), Makoto Iwata, Non-member(Osaka University),
Tetsuya Takine, Non-member(Osaka University), Hiroaki Terada, Member (Osaka University)

VLSI-oriented dynamic data-driven processors Qv-x is constructed by a self-timed elastic superpipeline, which has inherent buffering capability of instantaneous fluctuations in packet-flow. Thus the pipeline can smooth the packet-flow in the processor, so that it is recognized as a very promising hardware implementation method for dynamic data-driven processors. Due to the structure of programs and/or the characteristics of input packet streams, however, excessive fluctuations in packet-flow may happen, which lead to the performance degradation. To eliminate ill effects of excess fluctuations, elastic queue-buffers are also incorporated in the Qv-x. This paper proposes an allocation method of queue buffers in the Qv-x to cope with different sources of the fluctuation and provides the quantitative evaluation of buffering capability of fluctuations through extensive simulation experiments. The simulation results show that, by an allocation method, large fluctuations can be absorbed and full performance of the processor can be achieved.

キーワード: 超並列, データ駆動型プロセッサ, 自己同期緩衝パイプライン, キューバッファ

1. はじめに

超並列処理システムの実現には、ソフトウェア・ハードウェア両面に充分な考慮をはらった新しい処理パラダイムの導入が期待されている。我々は、データ駆動型処理方式をその有力な候補であると考え、動的データ駆動型1チップ・プロセッサ Qv-x の開発と、図的仕様記述から超並列実行プログラムを直接生成するための総合的なソフトウェア体系の開発を並行して進めている。本論文では、既に1チップ実現されている Qv-x プロセッサの性能向上のために行った、緩衝記憶機構の配置に関する検討結果を述べ、この形式の1チップ・プロセッサが GOPS 水準の処理性能を達成できる可能性を持つことを示す。

2. 動的データ駆動型プロセッサ Qv-x

〈2.1〉 動的データ駆動型処理方式

データ駆動型プログラムは一般に、ハードウェアプリミティブを示すノードとその処理ノードへの入力データを示すアークからなる計算グラフの形式に書かれる(図1参照)。データ駆動型処理は、各ノードはその処理に必要な入力データが全て揃ったときかつその時に限ってそのノードの処理を実行に移すという非常に自然で単純な方式である。特に、動的データ駆動型処理方式は入力アーク上に複数のデータの組(トークン)

の滞留を許し、ハードウェア構成が許す限り任意の個数のデータの組が単一のプログラムを共有し、さらにデータの組相互間の追い越しが許されるために、アーク上に単一のトークンの組しか許さない静的データ駆動型処理方式に比べて、はるかに柔軟な並列処理能力を有している。上述の動作を保証するために、動的データ駆動型処理方式では、データそのものにデータの行き先およびデータの組の識別符号からなるタグを付加したオペランドパケットの形式ですべてのデータが表現される。したがって、処理の実行は、たとえば図2に示す環状パイプライン形式で容易に実現される。入力部(IN)に到着したオペランドパケットは、単純な連想記憶である待合せ記憶部(Matching Memory: MM)

図1 計算グラフ

Fig.1. Computational Graph.

内に入り、タグの内容に従って対となるべきパケットを待合せる。対を成したオペランドパケットは演算パケットとして出力され、演算処理部 (Arithmetic & Logic Unit: ALU) に送られる。ALU では、タグの一部に含まれている演算コードにしたがって演算処理が行われ、その結果が、オペランドパケットと同じ形式の結果パケットとして、プログラム記憶部 (Program Memory: PM) に送出される。PM 内では、次の行き先と演算コードが付加され、ふたたびオペランドパケットとして出力される。この際、複数の行き先を持つ結果パケットは複製 (copy) され、各々の行き先と演算コードが付加され、前述の動作が繰り返される。ただし、これらのオペランドパケットの行き先がプロセッサ外部であれば、出力部 (OUT) で選択的に外部へ送出される。

図2 データ駆動プロセッサ Qv-x の構成例
Fig.2. An Example of Data-Driven Processors Qv-x.

以上の動作原理から明らかなように、データ駆動型処理方式では、いったんあるノードによって実行に移されたデータの組 (オペランドパケット) の処理は完全に前進的に行われ、かつ、その他のノードから処理実行に移された複数のオペランドパケットの処理実行とは全く独立に進行することが非常に大きな特徴となっている。すなわち、パイプライン段相互間の干渉 (インターロッキング) あるいは、通常のノイマン型プロセッサでは多段パイプライン処理の大きな障害となっている、分岐命令に伴うパイプラインの清掃 (フラッシング) などの処置が全く不要となるために、任意の段数のパイプライン処理が実現可能となる。この事実、プログラムが静的な構造として持っている同時並行処理性に加えてソフトウェア的なパイプライン並列性を活用できる環境では、データの転送あるいは処理に伴う遅れが処理率と全く無関係になり、遅延の隠蔽が非常に効果的に行われることを意味している [1]。

〈2.2〉 Qv-x の VLIS 実現法 Qv-x の VLSI 実現に当たっては極限集積化による超高速・低電力動作を意図した。大域的なシステムクロックはクロック分配あるいはクロックスキューなどの制約から高速化の究極的な障害になると予想されるためこれを排し、パイプライン段間転送制御にハンドシェイク方式を用いた自己タイミング型パイプラインを採用した [3, 4]。これによって、通常のノイマン型処理装置に見られるシーケンス制御のための集中制御部あるいは共有バスなど、外因性の速度阻害要因をすべて排除

することができ、高い処理能力/電力比を達成することが可能となった [2]。

自己タイミング型パイプラインは、自然なデータ緩衝能力を持つために、入・出力データの変動並びに待合せ記憶部 MM におけるオペランドパケットの生成率の揺らぎなどを吸収して、環状パイプライン内のパケット流量を平滑化する効果を持つので、動的データ駆動型処理方式には最適のハードウェア実現法となっている。しかし、前述の基本処理ブロック内で吸収できる変動量を超える揺らぎを吸収するために、実際のプロセッサでは、さらに可変長 FIFO (Queue Buffer: QB) を導入して、パケット流量変動を吸収している。この種のプロセッサに関する研究は、Qv-x の関連研究しかない。唯一、パイプライン合流部での入力側 QB 容量の評価手法が提案されている [7] が、プロセッサ全体の QB 容量に対する研究はない。本論文では、パケットの連続コピーを実現する最小構成のプロセッサにおいて、QB の配置法とその揺らぎ吸収能力の定量的評価を行い、この種のプロセッサ設計上の重要な要因を明らかにしている。

3. パケット流量の変動要因

Qv-x の能力を限界まで引き出すには環状パイプライン内のパケット流量の変動を可能な限り平滑化する必要がある。本章では、パケット流量の変動要因について述べ、定常的に処理可能なパケット流量に関する理論的評価を行う。

〈3.1〉 同時並行処理ノード数変動とコピー発生量 データ駆動型プログラムに内在する同時並行処理性は、プログラムの同一ランクに属するノードの数 (同時並行処理ノード数) によって定まる。ここで、ノードが属するランクは、そのノードが実行される時刻に相当し、そのノードに到達できる全てのソースノードからそのノードへ到達する最長経路の距離と定義する (図1参照)。一般に、同時並行処理ノード数はランク毎に異なり、結果として、単一のプログラムを1組の入力パケット群によって実行する際のパケット流量は、この同時並行処理ノード数に応じて変動する。一方、同一のプログラムが複数の入力パケットストリームに共有される場合は、単一パケット群に関する変動が時間的にずれた形で重なり合うことになる。

プログラムの同時並行処理性の変動をプロセッサ内部のパケット流量変動という観点から見ると、待合せ記憶部 MM でオペランド対を生成するために格納されることによる流量の減少と、プログラム記憶部 PM における結果パケットの複製 (コピー) による流量の増加に起因することが分かる。すなわち2入力命令の結果パケットが単一の行き先を持つ場合に流量は減少し、 n ($n = 1, 2$) 入力命令の結果パケットが $n + 1$ 以上の行き先を持つ場合は増加する。以下では、図2で示された単一環状型パイプライン構造をもつ Qv-x における、プログラムの同時並行処理性を表す特徴パラメタならびに、このパラメタによって定まる ALU における最大パケット流量を導く。

〈3.1.1〉 プログラムの特徴パラメタ プログラムの同時並行処理性を少数の特徴パラメタを用いて簡易に表現する。まず初めに、プログラム内に条件分岐が存在する場合には、分岐先の接続アーク数が多い側に分岐すると仮定する。また、閉路が存在する場合には、これを展開して非巡回グラフとする。このように、対象とするプログラムをプロセッサにかかる負荷がより重くなるように変形した後、以下のような特徴パラメタを定義する。

入力ソース数	N_{Input}	出力シンク数	N_{Output}
単項演算命令数	N_{1in}	二項演算命令数	N_{2in}
1行き先命令数	N_{1out}	N行き先命令数	N_{Nout}
総入力アーク数	N_{InArc}	総出力アーク数	N_{OutArc}
命令数	N_{op}	コピーの総数	$N_{SumCopy}$

これらのパラメタ間では、式 (1)、(2)、(3) が成り立つ。

$$N_{op} = N_{1in} + N_{2in} = \sum_{i=1}^{\infty} N_{iout} \quad (1)$$

$$N_{InArc} = N_{1in} + 2 * N_{2in} \quad (2)$$

$$N_{OutArc} = \sum_{i=1}^{\infty} N_{iout} \quad (3)$$

〈3.1.2〉 ALU における最大パケット流量 前節で与えられた特徴パラメタを元に ALU における最大パケット流量を導出する。まず R_{Input} を外部から入力部 IN への入力パケット流量とし、 R_{Output} を出力部 OUT から外部への出力パケット流量とする。さらに、プロセッサ内部の機能部 x と y との間のパケット流量を R_{x-y} で表す。ただし、 $x, y \in \{IN, MM, ALU, PM, OUT\}$ である。例えば R_{IN-MM} は入力部 IN と待合わせ記憶部 MM の間のパケット流量を表す。

いま、パイプラインを構成するハードウェアによって定まる最大パケット流量を R_{max} とすると、この制約から次式を得る。

$$R_{IN-MM} \leq R_{max}, \quad R_{PM-OUT} \leq R_{max} \quad (4)$$

また、ALU ではパケットの生成や消滅は起こらないため、ALU における入力パケット流量と出力パケット流量は等しく、

$$R_{MM-ALU} = R_{ALU-PM} \quad (5)$$

となる。一方、MM では二項演算命令に対するオペランド対パケットの生成のために一部のパケットが格納され、MM からの出力パケット流量が低下する。MM へ向かうパケットが MM に格納される割合は全入力アーク数に対する二項演算命令数の比 (N_{2in}/N_{InArc}) に等しい。よって、MM からの出力パケット流量 R_{MM-ALU} は、

$$\begin{aligned} R_{MM-ALU} &= R_{IN-MM} * (1 - (N_{2in}/N_{InArc})) \\ &= R_{IN-MM} * N_{op}/N_{InArc} \end{aligned} \quad (6)$$

を満たす。さらに、PM では結果パケットの連続複製により PM からの出力パケット流量は増加する。出力パケット流量の増加の割合は全命令数に対する全出力アーク数の比に等しい。よって、

$$R_{PM-OUT} = R_{ALU-PM} * N_{OutArc}/N_{op} \quad (7)$$

が成立する。

式 (4)、(6) ならびに式 (4)、(7) より、ALU における入

力、出力パケット流量はそれぞれ

$$R_{MM-ALU} \leq R_{max} * N_{op}/N_{InArc} \quad (8)$$

$R_{ALU-PM} \leq R_{max} * N_{op}/N_{OutArc}$ を満たす。よって、式 (5) より次式を得る。

$$R_{MM-ALU} = R_{ALU-PM} \leq R_{ALU_{max}} \quad (9)$$

ただし、 $R_{ALU_{max}}$ は ALU における平均パケット流量の最大値であり

$$R_{ALU_{max}} = R_{max} * \frac{N_{op}}{\max(N_{InArc}, N_{OutArc})} \quad (10)$$

で与えられる。すなわち、ALU における平均パケット流量の最大値は、パイプラインの物理的最大パケット流量 R_{max} 、プログラムの入力アークの総数 N_{InArc} 、出力アークの総数 N_{OutArc} ならびに総命令数 N_{op} によって決定される。また、式 (8) より、 $N_{InArc} \geq N_{OutArc}$ の時には、IN のパイプライン合流部から MM のパケット格納部までのパイプラインがボトルネックになり、逆に、 $N_{InArc} \leq N_{OutArc}$ の時には、PM のコピー部から OUT の分岐部までのパイプラインがボトルネックになることが分かる。

〈3.2〉 入・出力パケット流量の変動 Q_{v-x} は入力パケットによって起動される受動的プロセッサである。入力パケットの投入間隔は処理内容あるいは処理要求によって定まる外的要因であり、これを制御するためには、プロセッサの外部において制御機構を導入する必要がある。

一方、結果パケットの行き先がプロセッサ外部である場合は出力部へ送られる。プロセッサ外部に何らかの資源競合がある場合、結果パケットが直ちに外部へ出力されず、プロセッサ内に留まる可能性がある。これはパイプラインのパケット流量の平滑化の阻害要因となる。

以下では、特定の単一プログラムに対するストリーム型処理を行う場合の、プログラムに依存した入力データの理論的な最大入力可能パケット流量を導く。

〈3.2.1〉 理論的最大入力可能パケット流量 プログラムを一度だけ実行するために必要な入力パケット組を入力単位としたときの、入力パケット組の流量を $R_{SetInput}$ で表すと、定義より

$$R_{SetInput} = R_{Input}/N_{Input} \quad (11)$$

である。1つの入力パケット組がプロセッサに投入されると、ALU にはプログラムの全命令 (N_{op} 個) だけの処理を要求する。一方、ALU における最大パケット流量 $R_{ALU_{max}}$ は式 (10) で与えられるため、 $R_{SetInput}$ は次式を満たさなければならない。

$$R_{SetInput} * N_{op} \leq R_{ALU_{max}} \quad (12)$$

よって、式 (10)、(11)、(12) より、 R_{Input} は

$$R_{Input} \leq R_{Input_{max}} \quad (13)$$

を満たす。ただし $R_{Input_{max}}$ は理論的最大入力可能パケット流量であり

$$R_{Input_{max}} = R_{max} * \frac{N_{Input}}{\max(N_{InArc}, N_{OutArc})} \quad (14)$$

で与えられる。すなわち、理論的最大入力可能パケット流量はパイプラインの最大パケット流量 R_{max} 、プログラムの入力アーク数の総数 N_{InArc} 、出力アーク数の総数 N_{OutArc} ならびに入力数 N_{Input} によって決定される。

4. QB 配置方式とその評価

前章で導出した理論的最大性能は、パイプライン内のパケット流量の平均値に関するものである。実際の実行時には、プログラムの同時並行処理性や入力パケット系列の到着状況によって、パケット流量が変動する。既にVLSI実現しているQv-1[5]やQv-2m[6]では、プロセッサ内部を巡回するパケットが入力パケットと合流する際に、内部巡回パケットを一時的に退避するキューバッファ（Queue Buffer：QB）機構を導入していた。しかし、同時並行処理性の変動が大きいプログラムに対しては、自己同期型パイプライン機構が有している緩衝能力では不足し、理論的最大入力可能パケット流量での実行が不可能であった。これは、環状パイプライン上のボトルネックで発生するパケットの停滞が逆方向に伝搬するためである。

そこで本章では、前章で明らかにしたボトルネックが存在する2箇所にQBを配置する方式を提案し、この分散QB配置を採用したQv-xの性能をパケットフローシミュレーションにより評価し、理論的最大性能を達成できることを示す。

〈4.1〉 QB 配置方式 前章で明らかにしたように、例えば、図2の構成では、入力部とプログラム記憶部内のパケット複製部がボトルネックになる。したがって、図3に示すように、QBを入力部と出力部の間、ならびに、プログラム記憶部の直前に分散配置する。これによって、パケット流量の変動を、配置したQBの容量分だけ緩衝することが可能になる。

図3 分散QBを持つQv-x
Fig.3. Qv-x with distributed queue-buffers.

本論文では、以上の方式に従って構成されたプロセッサを、分散QBを持つデータ駆動型プロセッサQv-xと呼ぶ。

〈4.2〉 シミュレーション評価 分散QBを持つQv-xの性能評価を行うために、プロセッサ内のパケット流を環状パイプラインの各段間の転送時間で離散化した、シミュレーション・モデルを構築した。このモデルでは、QBの配置箇所の効果を測定するために、各QBの容量は十分にあるとした。さらに、外部へ送られたパケットは外部資源を優先的に使用できるものとし、プロセッサ内に滞留することはないとした。

3章で導出したALUにおける理論的最大パケット流量（最大ALUパケット流量）を性能評価基準として、その達

成度を観測するために、与えられたプログラムに対して最大ALUパケット流量を測定する。また、パイプライン自身の緩衝能力、すなわち、環状パイプラインの段数を変化させてシミュレーション実験を行った。これは、自己同期型パイプライン自体が本来有している緩衝能力に対して、各QBによる緩衝効果を明らかにするためである。

本評価では、図3で示された分散QBを持つQv-x以外に、表1に示されるいずれか一方のQBのみを持つプロセッサとQBを持たないプロセッサの3種の比較対象を用意して、上記のシミュレーションにより定量的な比較評価を実施する。

表1 比較対象プロセッサ
Table1. Processors for comparative purpose

	QB1	QB2
分散QBを持つQv-x（提案モデル）	有り	有り
集中QBを持つQv-x（モデルA）	有り	—
集中QBを持つQv-x（モデルB）	—	有り
QBを持たないQv-x（モデルC）	—	—

一方、被評価プログラムは、同時並行処理性およびパイプライン並列性が様なものに加え、これらの変動が極めて大きいものを多数用意し、分散QB配置方式の評価が汎用的に行えるように配慮した。図4は、その一例で、同時並行性およびパイプライン並列性の変化が大きい、以下のような特徴パラメータを持つテストプログラムに対するシミュレーション結果である。

特徴パラメータ： $N_{Input} = 8$, $N_{Output} = 8$, $N_{Op} = 61$, $N_{InArc} = 80$, $N_{OutArc} = 80$, $N_{1in} = 42$, $N_{2in} = 19$, $N_{1out} = 46$, $N_{2out} = 13$, $N_{4out} = 2$

図4 最大ALUパケット流量
Fig.4. Maximum packet flow in ALU.

図4より、分散QBを持つQv-xは、一定の入力間隔でデータを入力する場合には十分なQB容量があれば理論的最大ALUパケット流量を処理できることがわかる。さらに、上述の多様なプログラムに対するシミュレーション結果でも分散QBを持つQv-xに関しては同様の結果が得られた。一方、比較モデルAでは、同時並行処理性変動の小さいプログラムのみ理論的最大ALUパケット流量を処理

できることがわかった。また、比較モデルBと比較モデルCでは、同時並行処理性変動を持つプログラムは理論的最大ALU パケット流量で処理できないことがわかった。

さらに、実用的なプログラムの一例として、有限体積法による流体計算で用いられる SIMPLE 法において、速度を導出する連立一次方程式をヤコビ反復法で解くプログラム CFD に対する結果を示す。このプログラムの特徴パラメータは、 $N_{Input} = 12$, $N_{Output} = 9$, $N_{op} = 159$, $N_{InArc} = 216$, $N_{OutArc} = 213$, $N_{lim} = 102$, $N_{2im} = 57$ である。

図5は実線が一定間隔で入力パケットを投入した場合を示していて、点線が入力パケット間隔に最大50%変動がある場合の最大ALU パケット流量を示している。この図より、3つの比較モデルでは最大ALU パケット流量で処理できないのに対して、提案モデルでは最大ALU パケット流量で処理可能であることが示されている。

図5 CFD を実行した場合の入力間隔の変動による影響
Fig.5. Influence of variable input intervals on maximum input rate for CFD program.

他の評価プログラムに対しても入力パケットの入力間隔を変動させてシミュレーション実験を行った結果、上記と同様の結果が得られた。

以上より、提案した分散QBを持つQv-xは、十分なQB容量を持つ場合、実行するプログラムの同時並行処理性変動や入力パケットの入力間隔の揺れに依存せず、理論的な最大パケット流量で安定して動作することが確認された。

5. QB 容量とその評価

実際にプロセッサを設計する際には、分散配置されたQBの適切な容量を決定する必要がある。しかし、前章に述べたシミュレーション評価の過程で、各QB内に滞留するパケット数(QB長)の最大値は、プログラムの性質や環状パイプライン段数に依存して、変動することがわかった。本章では、環状パイプライン段数に依存しないQB容量の十分量を推定し、この見積り方法が妥当であることをシミュレーション実験を通して明らかにする。

〈5・1〉 QB1 容量の十分量の推定 QB1のQB長は外部から入力パケットが投入される時のみ増加する可能性がある。ここでは、ストリーム型の処理を行なうプログラムの実行を対象として、十分多くの入力パケット組を理

論的最大入力パケット流量で1パケットずつ等間隔にプロセッサに入力する場合を想定して議論を進める。

定常状態では、1つの世代の入力パケットが入力され、それによってQB1に滞留する周回パケットが、次の世代が入力されるまでに解消されなければ、QB1に滞留するパケットが限りなく多くなりプロセッサは破綻に向かう。したがって、定常状態ではQB1は次世代の入力までに入力される前の状態に戻る必要がある。

一方、プログラムの実行開始時からプロセッサ内のパケット流が安定するまでにQB1に溜るパケット数は1世代の入力パケット数を超えないので、QB1の長さはプログラムの入力数の2倍以上にはならないと推定できる。したがって、QB1の容量の十分量は、式(15)と推定できる。

$$QB1 \text{ の容量の十分量} = N_{Input} * 2 \tag{15}$$

〈5・2〉 QB2 容量の十分量の推定 QB2のQB長は、パケットが複製され、かつ、QB2に連続的にパケットが到着する場合に長くなる。そこで、最もQB2のQB長が伸びる可能性がある状態を考えると、プログラムのほとんど全てのランクに1つ以上のパケット組が存在する状態である。プロセッサ内のパケット流が安定した状態では、次の世代の到着までに同じ状態まで戻らなければ、QB2に滞留するパケットが限りなく多くなりプロセッサは破綻に向かう。よって、QB2のQB長の最大値は「コピー数の総数*2」で抑えられると推定できる。

よって、QB2の容量の十分量は、コピー数の総数を $N_{SumCopy}$ とすると、式(16)と推定できる。

$$QB2 \text{ の容量の十分量} = N_{SumCopy} * 2 \tag{16}$$

〈5・3〉 シミュレーション評価 3.2節と同様に、実用的なプログラムの一例として、流体計算で用いられるプログラムに対して理論的最大入力パケット流量で入力して最大QB長を評価した。各QBの最大QB長は、入力部に対するQBを持たないモデルBとCでは、そのQB長は増加し続けた。一方、モデルAと分散QBを持つQv-xのQB長は有限の値に抑えられ、各QB容量の十分量の推定値を超えていないことを確認した。また、表2は、プログラムCFDとテストプログラムWORSEの最大QB長の実測値と推定値の比を示している。この他にも、3.2節と同様に様々なプログラムについてシミュレーション実験を行い、各QBの最大長は十分量の推定値で抑えられることを確認した。

表2 最大QB長の実測値と推定値の比
Table2. Ratio of experimental maximum QB length to estimated one.

実測値／推定値	CFD	WORSE
QB1	0.25	1.00
QB2	0.10	0.79

6. おわりに

本論文では、アーキテクチャ構成の最適化の観点から、分散QBを持つQv-xを提案した。適当な容量のバッファを環状パイプラインのボトルネックとなる機能部に分散配置することによりパケット流量の変動を吸収し、定常的に理論的最大性能を達成できることを確認した。

この論文で得られた結果から、一般的に次のようなことが言える。すなわち、Qv-xプロセッサを設計する上で、理論的最大性能を達成するためには、パケット流のボトルネックになり得る (i) パイプラインの合流部において、合流の優先度が低い側の入力パイプライン上、ならびに (ii) パケットの複製が生じる箇所に、QBを分散して配置すべきである。

次期開発予定のQv-xプロセッサは2GOPSの最大ピーク性能を有している。このプロセッサに上記の分散QB配置方式を適用すれば、1~2GOPSの実効性能を常に維持できるプロセッサの実現が可能であることが判る。

また、ここで得られたQB長の十分値は、QBの容量が与えられた時のプログラムの静的な最適化手法において、最適化後のプログラムが満たすべき具体的な特徴パラメタの指標を与えている。また、入力制御を考える際にも、平均入力パケット流量に関して具体的な目標値を与えている。今後、本論文で得られた結果を基に、これらの最適化手法、ならびに、入力制御法について検討する予定である。

謝辞 御指導、御支援頂いたQv-x研究の関係者各位、ならびに御協力頂いた寺田研究室の各位に感謝します。特に、Qv-xプロセッサを提供していただいた、シャープ(株)と三菱電機(株)の関係各位に深謝します。なお、本研究の一部は、文部省科研費(試験B-1 06555110, 一般B-2 05452363, 重点知能の極限集積化), ならびに、新エネルギー・産業技術総合開発機構(NEDO)の提案公募型最先端分野研究開発の援助を受けて行われたものである。

(平成8年3月7日受付、同8年5月10日再受付)

参考文献

- [1] Arvind and R.A.Iannucci: "A Critique of Multiprocessing von Neumann Style," The 10th Annual Symp. on Comp. Architect., pp.426-436 (1983).
- [2] 芳田真一, 紫竹リカルド毅史, 松浦康弘, 村松剛司, 岡本俊弥, 宮田宗一: 「映像信号処理向きデータ駆動プロセッサ」, 信学技法, DSP95-109, ICD95-158 (Oct. 1995).
- [3] T.Tamura, S.Komori, F.Asai, H.Tsubota, H.Sato, H.Takata, Y.Seguchi, T.Tokuda and H.Terada: "Self-Timed Clocking Design for a Data-Driven Microprocessor," IEICEJ Trans., Vol.E-74, No.11, pp.3757-3765 (Nov. 1991).
- [4] 佐藤尚和, 小守伸史, 浅井文康, 田村俊之, 瀬口禎治, 徳田健, 寺田浩詔: 「可変遅延素子を用いた自己同期パイプライン」, 信学技法, ICD93-48, pp.31-38 (June 1993).
- [5] S.Komori, K.Shima, S.Miyata, T.Okamoto and H.Terada: "Data-driven microprocessor," IEEE Micro, pp.45-59 (June 1989).
- [6] S.Komori, T.Tamura, F.Asai, H.Tsubota, H.Sato, H.Takata, Y.Seguchi, T.Ohno, T.Tokuda and H.Terada: "A 50 MFLOPS Superpipelined Data-Driven Microprocessor," Proc. of IEEE Int'l Solid State Circuit Conf., pp.92-93 (Feb. 1991).
- [7] 小守伸史, 高田英裕, 田村俊之, 浅井文康, 山崎哲男, 嶋憲司, 西川博昭, 浅田勝彦, 寺田浩詔: 「巡回パイプライン方式データ駆動形プロセッサの性能評価方法に関する一提案」, 電子情報通信学会論文誌, Vol.J71-D, No.8, pp.1553-1559 (Aug. 1988).

上方 輝彦

(非会員)1972年1月3日生まれ。1994年大阪大学工学部情報システム工学科卒業。1996年同大学院博士前期課程修了。同年同大学院博士後期課程入学、現在に至る。データ駆動プロセッサの研究に従事。情報処理学会、電子情報通信学会、各会員。

岩田 誠

(非会員)1963年生。1986年阪大・工・電子卒。1991年同大学院博士課程単位取得後退学。同年大阪大学工学部情報システム工学科助手、現在に至る。データ駆動パラダイムを核とした、ソフトウェア環境およびVLSI向きデータ駆動アーキテクチャの研究に従事。情報処理学会、電子情報通信学会、IEEE-CS各会員。

滝根 哲哉

(非会員)1961年生。1984年京大・工・数理卒。1989年同大学院博士後期課程修了。同年、京大・工・数理助手。1994年阪大・工・情報システム講師、同年同助教授となり現在に至る。1991~1992年カリフォルニア大アーバイン校客員研究員。待ち行列理論並びに情報システムの性能評価に関する研究に従事。工博。日本OR学会、電子情報通信学会、情報処理学会、システム制御情報学会、IEEE各会員。

寺田 浩詔

(正員)1933年生。1956年愛媛大学工学部電気工学科卒業。1961年大阪大学大学院通信工学専攻博士課程修了。同年大阪大学助手となり、講師、助教授を経て、1976年同教授(電子工学科)、1989年から同情報システム工学科教授となる。1993年から2年間大阪大学大型計算機センター長を併任。交換機などの実時間高度並列処理向きの言語・アーキテクチャの研究に従事し、1989年、電子情報通信学会業績賞、小林記念特別賞を受賞。電子情報通信学会副会長。情報処理学会関西支部長。テレビジョン学会会員。IEEEシニア会員。