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

2004－HPC－98　(6)
2004／4／13

仮想キューマシン VQM の構成と基本性能評価

川田　宗太郎†　曽　和　将　容††

キュー計算モデルに基づく新しい並列プロセッサ「キューマシン」に、仮想的にキューを長くする「仮想キュー」を導入することを提案する。このプロセッサは、仮想レジスタの導入によってプログラムサイズが小さいまま大並列実行が可能である。スーパスカラプロセッサとしてシステムを設計し、シミュレーションによって基本的な性能評価を行った。その結果、プロセス切り替えの効率が良く、プログラムサイズを考慮するとパフォーマンスが上がることが分かった。これは限られたハードウェア資源を有効に使うプロセッサである。

Construction and Basic Performance Evaluation of Virtual Queue Machine

Sotaro Kawata† and Masahiro Sowa††

In this paper we propose a novel parallel processor, named Queue Machine, based on the queue calculation model. A "Virtual Queue" is introduced so that the queue length becomes virtually unlimited. Parallel processing can be easily performed by the proposed machine with the introduction of virtual registers while keeping small program size. We design the system as a super scalar processor and evaluate its basic performance. The results show that the processor decreases the process switching costs. In addition, programs for the proposed machine are considerably compact. Moreover, the processor makes good use of the limited hardware resources.

1. はじめに

プロセッサの計算モデルは、アキュムレータ型、レジスタ型、スタック型が良く知られている。最も代表的な仮想マシンの一つである Java[1] は、スタック計算モデル[2)3)] に基づいたマルチプラットフォームなマシンである。スタックはデータの入出力が１箇所であるため、前後の命令に依存関係が発生してしまい、並列計算のパフォーマンスが悪いという欠点がある。ところが、キューを用いればこの問題点が解決される。キューの構造上、データの入出力がお互いに独立しているためである。キュー計算モデルは Bruno ら[4)] によって研究されていたが、そのモデルの持つ並列性は、前田ら[5)] や曽和[6)] によって見出された。そして、キュー計算モデルに基づく並列プロセッサ「キューマシン」の基礎[6)] が確立され、スーパスカラプロセッサとしての設計および基本性能評価が行われてきた[7)8)9)]。キューマシンでは、キューは長ければ長いほど大並列実行が可能となる。しかしながら、物理レジスタで長いキューを実現するのは困難である。以前に設計されたキューマシン[7)] は PLD 化を目指したシミュレーション上のものであり、キューは全て物理レジスタで構築されることを想定していた。よって演算部分に利用できるキューが短く、データ退避用のキューを必要とした。

そこで筆者らは、仮想レジスタを取り入れることにより長いキューを仮想的に実現するキューマシンを提案する。このキューマシンは、キューからデータを溢れないようにするための spill コードを含める必要がないため最小限のプログラムで済み、しかも、プログラムの実行時間も物理キューのものと比べそれほど劣らない。また、ハードウェアが自動的にレジスタを選択するため、限られたハードウェア資源を有効に使う方式である。本稿では仮想キューマシンの構成と、それを基にしたシミュレーションによる性能評価を行った結果を報告する。尚、この研究内容は筆頭著者の卒業研究[8)] として成し遂げられたものである。

キューマシンは非常に新しい概念であり、まだ一般にはほとんど知られていない。しかしながら、レジスタマシンと比べて実行プログラムサイズが 1/3 程度であること、並列実行が可能であること、また、アーキテクチャが非常にシンプルであることを考慮すると、複雑化しすぎた現在のプロセッサ[10)] に代わる、新たなプロセッサ方式の１つであることは確実である。現在、汎用プロセッサへのキュー計算モデルの導入[11)12)]、C

† 東京大学 大学院新領域創成科学研究科 基盤情報学専攻
The University of Tokyo, Graduate School of Frontier Science, Department of Frontier Informatics
†† 電気通信大学 大学院情報システム学研究科 情報ネットワーク学専攻
The University of Electro-Communications, Graduate School of Information Systems, Department of Information Network Science

コンパイラの開発[13]、Queue-Java の開発[14] など、研究の裾野が広がりつつある。更に、他のアーキテクチャとの定量的な比較評価を行なうプロジェクトも進行中である。

2. 仮想キューマシン VQM の設計

本稿で提案するキューマシンは、循環配列レジスタ (後述) と仮想レジスタ (後述) とを組み合わせることにより、仮想的に長いキューを実現する仮想キューマシン (Virtual Queue Machine, VQM) である。図 1 に仮想キューマシンの概念を示す。

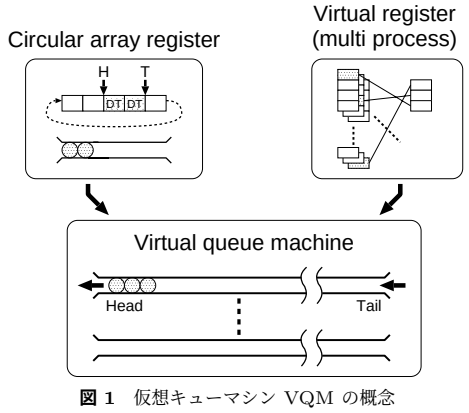

図 1 仮想キューマシン VQM の概念

2.1 キュー計算モデル

図 2 にキュー計算モデルの (a) 命令フォーマット、(b) 構文木、(c) 演算の流れを示す。ここでは式 $3*(2+5)-8=13$ を例に挙げて説明する。

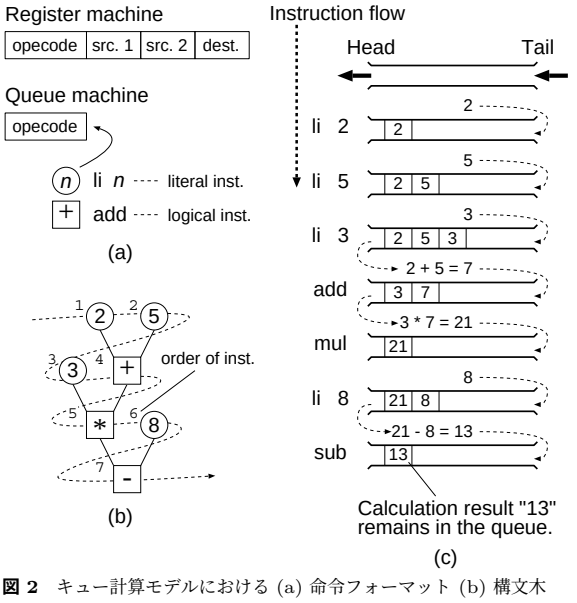

図 2 キュー計算モデルにおける (a) 命令フォーマット (b) 構文木 (c) 処理の流れ

レジスタ計算モデルの場合、命令がデータを読み書きするレジスタを明示的に指定するが、キュー計算モデルの場合、図 2(a) のように、オペランドを必要としない分、命令サイズが小さくなる。また、キュー計算モデルでは、命令はレジスタを意識せずキューのみを考慮する。命令は、キューからデータを取り出し、演算をし、再びキューにデータ (演算結果) を入れるだけである。尚、○はキューに即値を入力する命令、□は演算命令とする。命令列（プログラム）は、図 2(b) の曲線矢印が指す順番（左から右へ、上から下へ）に生成される。生成されたプログラムは逆ポーランド記法に似ているが、それとは異なりキュー計算モデル独自の並びである。図 2(c) にプログラムの進行とキュー内の演算の様子を示す。プログラムが終了すると、最期に答えの”13”がキューに残る。

2.2 循環配列レジスタによるキューの実現

キューは図 3 に示す循環配列レジスタとして実現される。

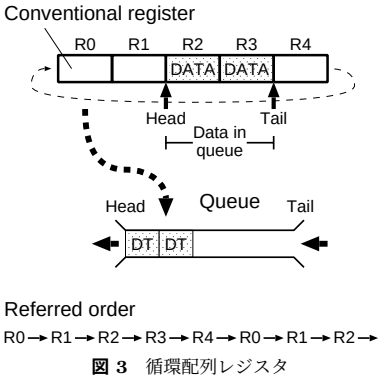

図 3 循環配列レジスタ

循環配列レジスタでは、データが挿入されると配列上の 1 エントリが割り当てられ、読み出されるとそのエントリが解放される。例として、5 エントリのレジスタを考える。キューの先頭を表すポインタ Head と、キューの末尾を表すポインタ Tail がある。データが取り出されると Head が右へ移動し、データが挿入されると Tail が右へ移動する。Head と Tail のポインタ値が 4 を超えた場合は、再び 0 へ戻る。これが繰り返され、レジスタ上でのキュー計算が執り行われる。

2.3 仮想レジスタ

キューマシンでは、キューが短いとデータが溢れて演算が行えなくなる場合がある。そこで、VQM では、仮想レジスタを導入することで長いキューを実現する。図 4 に仮想レジスタの概念を示す。

仮想レジスタの概念は、仮想メモリとほぼ同様である。いくつかのレジスタをひとまとまりにしてメモリ (キャッシュ) へ書き出したり、メモリからロードする。仮想メモリにおけるページングに相当するこの概念を、本稿は Swap と名付ける。Swap するレジスタのひとまとまりをレジスタブロック、1 ブロックのレジスタ数をブロック長と定義する。仮想レジスタと物理レジ

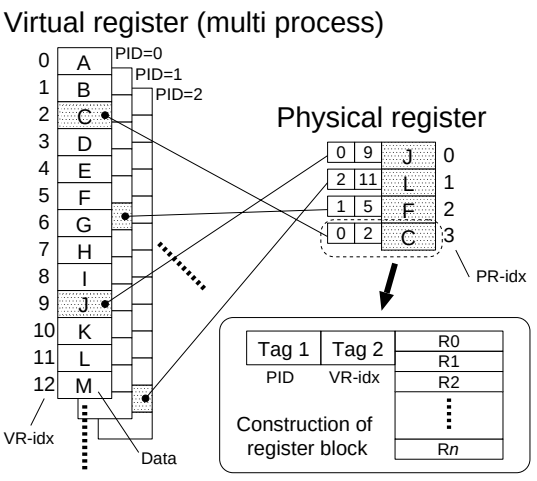

図 4 仮想レジスタの概念

スタは、それぞれインデックスを持つ。物理レジスタには Tag が 2 つあり、1 つは仮想レジスタのインデックス (VR-idx) を、もう一つはプロセス ID(PID) を保存する。これによって複数のプロセスにも対応することができる。

2.4 キューの長さとプログラムサイズ

キューが短いと、spill コードと呼ばれるメモリ退避命令をプログラムに含ませてキューからデータが溢れないようにする必要がある。よって実行プログラムのサイズが増大する。長いキューでは spill コードは必要ないため、プログラムサイズは最小で済む。この様子を図 5 に示す。

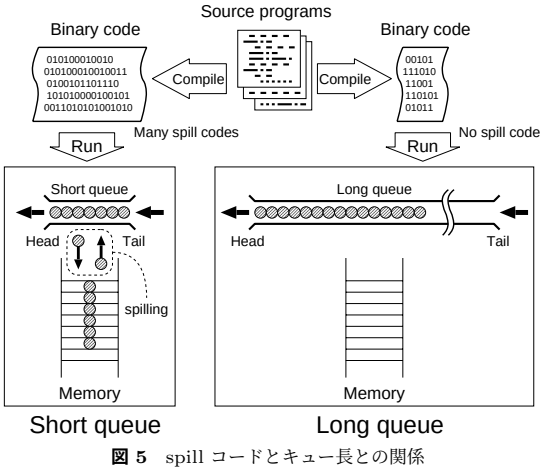

図 5 spill コードとキュー長との関係

VQM の特徴は、長いキューを仮想的に作ることができるため、spill コードのない最小限の実行プログラムで処理ができる点である。

2.5 VQM の構成

図 6 に VQM のブロック構成を、表 1 に構成要素数を示す。本研究では VQM を一般的なスーパスカラプロセッサとして設計する。命令はパイプライン（Fetch, Decode, Issue, Execute, Writeback など）を経て処理される。命令処理の管理は命令ウィンドウで行われる。

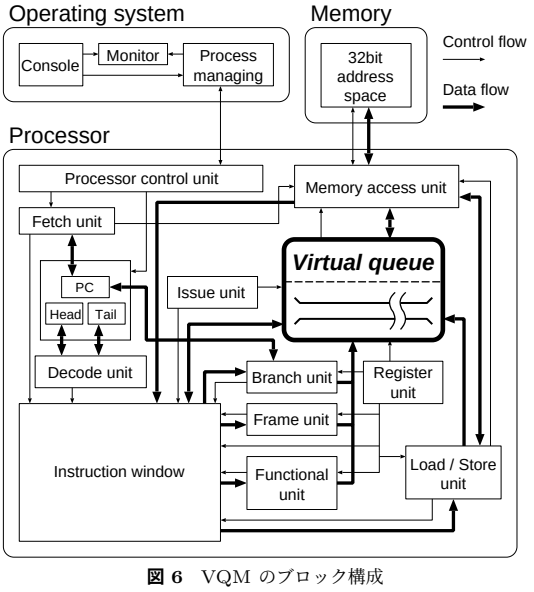

図 6 VQM のブロック構成

表 1 VQM の構成要素数

Default patameters of VQM	
Word size in bits	32
Instruction fetch width	4
Queue length (in words)	320
Block length (registers)	8
Number of register blocks	4
Swap buffer entries	20
Instruction window entries	20

Arithmetic logic unit	
Integer	4
Single-precision fp	4
Double-precision fp	4
Logic operation	4
Type convertor	4

Swap latency per register	1 [cycle]
Process switching latency	10 [cycle]

Queue control unit	4
Branch unit	1
Frame access unit	1
Memory access unit	1
Load unit	1
Shift unit	1

VQM の基本構成は一般的なスーパスカラプロセッサと代わらないが、Head と Tail の指すポインタ値が使用するレジスタ番号であり、プロセッサ内で動的に使用するレジスタが決まる点がこれまでのプロセッサと異なる。レジスタリネーミングなどのステージは、VQM には必要ない。Queue length は仮想キューの長さである。プログラムはこの長さを越えないように設計される。物理キューのハードウェア量は、Block length と Number of register block の積である。Swap buffer は、Swap 要求のあるレジスタブロックのインデックスを一時保存するキューである。Swap による遅延時間は、1 レジスタにつき 1 サイクルを想定する。プロセス切り替えは 10 サイクルを見積もる。

3. パフォーマンス

設計した VQM は、様々な特徴を兼ね備えている。それぞれの特徴に応じた評価項目に分類することが有効と考え、次の 5 つの項目で性能評価を行った。それぞれ、キューマシンの並列実効性の検証、効率の良いブロック長の評価、Swap 戦略、プロセス切り替えのコスト、仮想レジスタの効果である。評価を行うために、VQM の設計に基づいたシミュレータを作成し、これを使用してサイクル数、IPC の値、及び Swap の回数を測定した。また、評価項目に相応しいベンチマークを作成した。評価項目により VQM の構成要素数を多少変化させるが、特に指定が無い限り、表 2 に従う。

3.1 ベンチマークプログラム

VQM 用のコンパイラが存在しないため、構文木からアセンブラを起こし、ベンチマークプログラムを作成した。表 2 に評価に使用するベンチマークプログラムとその特徴を示す。作成したベンチマークは、*intmix, li-drop, dblong, no-spill, use-spill* の 5 種類である。

表 2 ベンチマークプログラム

Program	Size [Byte]	Feature
intmix	33	This program is composed from many integer instructions. Used queue length is under 32.
li-drop	9	This program has simple activity of only input and output. The highest parallelism can be performed.
dblong	68	Double-precision floating-point calculation is done in this program. Queue length is over 32.
use-spill	122	Both programs perform the same tasks but with different queue length requirements. The first one, "use-spill", creates many spill codes and it does not need a long queue. The other program, "no-spill", makes no spills, but needs a longer queue.
no-spill	72	

3.2 キューマシンの並列性

キューマシンの持つ並列性の検証を行った。実験では 320 個の物理レジスタのみで構成される物理キューマシンを想定した。命令のフェッチ幅を 1〜5 までとし、命令ウィンドウのサイズを 10 エントリ、20 エントリの 2 通りとして、それぞれ 500 命令読み込んだところの IPC の測定を行った。使用したベンチマークは、*intmix, li-drop, dblong* である。図 7 に命令フェッチ幅に対する並列実行度を示す。

実験の結果、命令フェッチ幅を 1 以上に設定したものは、いずれのベンチマークにおいても、IPC が 1 以上になった。このことから、キューマシンに並列実効性があることが確かめられた。*li-drop* は並列実行度が突出しているが、これは原理上、最高の並列度が出せるベンチマークであるため、その特徴が顕著に現れている。命令ウィンドウのエントリ数を増やすと、多くの命令が一度に管理できるため、並列実行できる命令の先読みが数多くできる。このこともグラフから読み取れる。

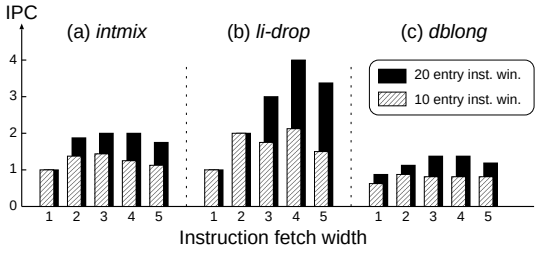

図 7 命令フェッチ幅に対する並列実行度

3.3 ブロック長

効率の良いブロック長の評価を行った。実験では、32 個の物理レジスタ、320 個の仮想レジスタによる VQM を想定した。ブロック長をそれぞれ 32, 16, 8, 4, 2, 1 レジスタとし、前回と同様の条件で IPC を測定した。ただし、命令ウィンドウは 20 エントリである。図 8 にその実験結果を示す。

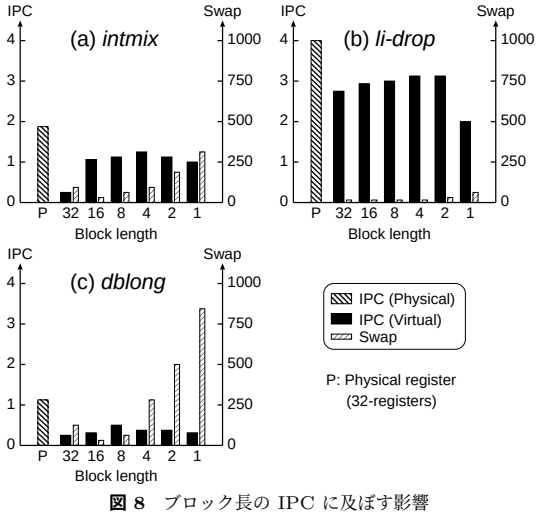

図 8 ブロック長の IPC に及ぼす影響

実験結果から、(a) ではブロック長を 4 にしたものが僅かに良い性能を示し、(b) では 1 を除いて殆ど変わらず、(b) では 8 が最も良い性能を示している。また、Swap の回数は、ブロック長が小さくなるほど増える傾向を示している。参考のために、物理レジスタのみで構成されたキューマシンでの結果を示す。(a) と (c) では、仮想レジスタに比べて物理レジスタの方が IPC は大きいが、(b) ではあまり差が出ない。並列実行が可能なプログラムでは、仮想レジスタが有利であることが分かる。

3.4 Swap 戦略

Swap させるレジスタブロックの戦略を変えて、そのパフォーマンスの評価を行った。図 9 にその実験結果を示す。

実験では、アクセスが最も新しいもの、2 番目に新

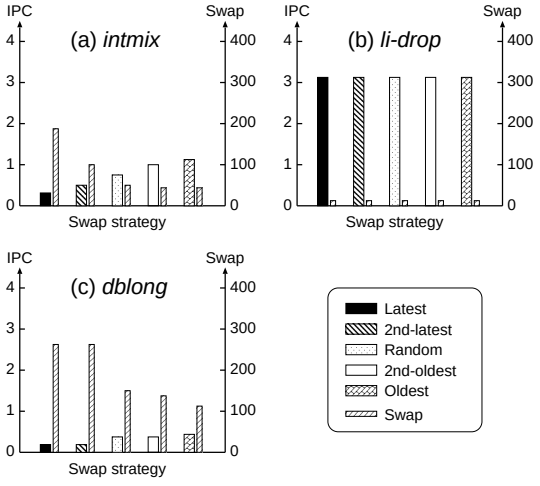

図 9 Swap 戦略の IPC に及ぼす影響

しいもの、ランダム、最も古いものから 2 つ前、そして最も古いものの 5 つで IPC と Swap の回数を調べた。(a) と (c) からは、最も古いものを Swap するのが効率が良いことが分かった。(b) は戦略に対して変化がなかった。

3.5 プロセス切り替えのコスト

構築した VQM は、マルチプロセスに対応する。そこでプロセス切り替えを行った場合の評価を行った。想定するプロセス切り替えの概念を、図 10 に示す。

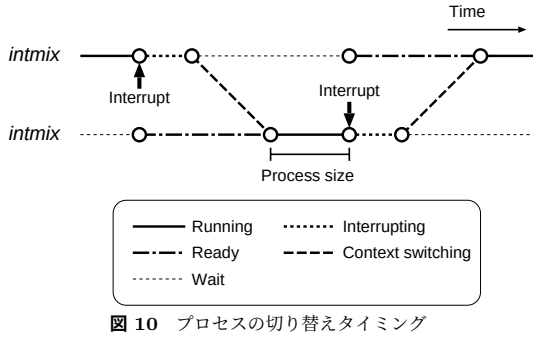

図 10 プロセスの切り替えタイミング

プロセスを特定する情報は Head, Tail, Frame, PC, PID である。プロセス切り替えでは、これらの情報の退避に 10 サイクル必要であると想定した。物理レジスタのみのプロセッサでは、プロセス切替えの際はレジスタの内容も退避させる必要があり、32 サイクルの遅延が必要と想定する。VQM では、レジスタの内容を退避させる必要は無い。従って、仮想レジスタの場合、1 回のプロセス切替えに 10 サイクルの遅延で済み、物理レジスタ方式の場合はレジスタと合計して 42 サイクルの遅延が必要となる。ベンチマークには *intmix* を用いた。測定は、まず *intmix* を 2 つ立ち上げ、一定の命令数を読み込んだ時点でプロセスを交互に切替え、500 命令読み込んだ時点で、IPC の値を読んだ。切替える命令数の間隔はそれぞれ、500 命令, 250 命令, 125 命令, 83 命令, 62 命令, 50 命令, 42 命令とした。よって、切替えの回数はそれぞれ、切替えなし、1 回, 3 回, 5 回, 7 回, 9 回, 11 回となる。図 11 にプロセス粒度の IPC に及ぼす影響を示す。

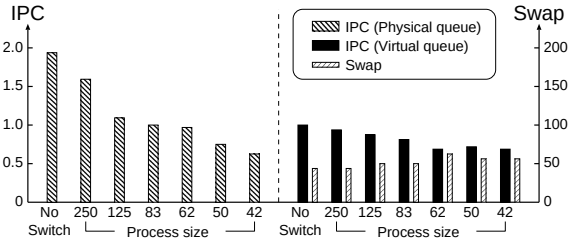

図 11 プロセス粒度の IPC に及ぼす影響

結果から、仮想レジスタの場合は、プロセス粒度を小さくしてもあまり性能は低下しないことが分かった。この理由は、粒度が小さい場合はプロセス番号が切り替わらずに、以前のプロセス番号が残るレジスタブロックが多くなるため、Swap を行なわずに処理が続けてできるからである。これは仮想レジスタの特徴であり、実験の結果からこの特徴が明らかとなった。これに対し、従来の方式ではプロセス粒度が小さくなるに連れ、性能が低下していった。このことから、仮想レジスタにおけるプロセス切替えは効率が良いといえる。

3.6 仮想レジスタの効果

キューは短いが全て物理レジスタのみで構成されるキューマシンと、キューの長い VQM とで、同じ目的を達成するベンチマークを用い、仮想レジスタを導入することの効果を調べた。ベンチマークが処理し終えたときのかかったサイクル数とプログラムサイズの積を、かかったコストとして評価する。spill コードが必要な短いキューを物理レジスタで構成し、spill コードの要らない長いキューを仮想レジスタとする。同じ物理レジスタ数にもかかわらず、仮想レジスタの方がコストを低く抑えられれば、仮想レジスタの効果を確かめることができる。

表 3 仮想レジスタの評価に用いる比較対象の条件

	Benchmark	Size [Byte]	Queue len. [word]	Phy. reg. [entry]	Fetch size [inst]	Inst. win. [entry]
(i)	*use_spill*	122	8	8	2	3
(ii)	*no_spill*	72	320	8	2	3
(iii)	*use_spill*	122	8	8	4	20
(iv)	*no_spill*	72	320	8	4	20
(v)	*no_spill*	72	320	16	4	20
(vi)	*no_spill*	72	320	32	4	20

表 3 に示される 6 つの条件のプロセッサを想定して実験を行った。実験結果を図 12 に示す。(a) は命令ウィンドウのサイズを変えた場合のコストの比較、(b) は物理レジスタの個数を変えた場合の比較である。(a)

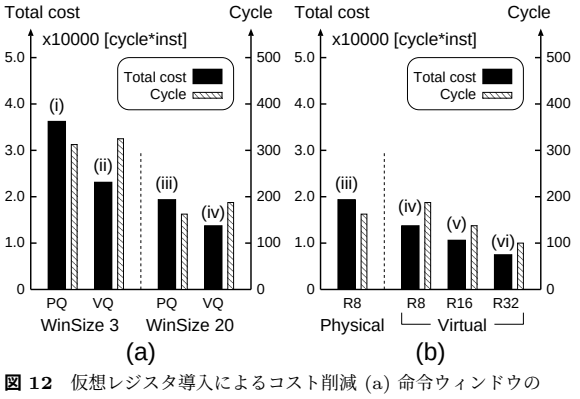

図 12 仮想レジスタ導入によるコスト削減 (a) 命令ウィンドウの大きさを変えた場合 (b) 物理レジスタの個数を変えた場合

からは、物理レジスタと仮想レジスタ共に、命令ウィンドウサイズを大きく取った方がコストを低く押さえられることが分かる。また、同じ命令ウィンドウサイズなら、仮想レジスタの方がサイクル数は多くなっているが、コストが低くなっている。(b) からは、仮想レジスタ中の物理レジスタを少し増やせば。少し少ない物理レジスタのみのプロセッサよりもサイクル数を少なくできることが分かる。

4. おわりに

本研究では、仮想レジスタを導入した並列実行のキューマシンを設計し、そのシミュレーションによって基本的な性能評価を行なった。評価の結果から次のことを確認した。(1) キューマシンの並列実行が可能であること。(2) 最も効率の良いブロック長は 4 個もしくは 8 個であること。(3) 最も古いレジスタブロックを Swap すると効率が良いこと。(4) プロセスの粒度を小さく取ればプロセス切替えに効果が出ること。そして、(5) 仮想レジスタを導入するとコストが削減できるということである。

本研究で得られた一番大きな結果は、プログラムが要求するキューの長さほど、物理レジスタの数を用意しなくても、少ないレジスタで用が足せる上、プログラムサイズも短いままで済むこと、なおかつ、それほど速度も低下しないプロセッサが作れることを、シミュレーションで確かめられたことである。本研究は、限られたハードウェア資源で、効率良く処理が実行できることを証明した。VQM 用のコンパイラが開発されれば、スケジューリングなどの評価項目で、さらに研究の幅が広がると考えられる。利用するレジスタの選択や Swap 戦略などにニューラルネットワークなどの学習理論を取り入れることや、仮想レジスタの物理的構成に光インターコネクションの技術を取り入れることで、更に効率が増すものと考えられる。

謝辞

前田敦司先生 (筑波大学) には、研究の初期の段階で VQM 用の命令セットを設計して頂き、研究を大変潤滑に進めることができた。廣瀬明先生 (東京大学) には、研究に対する深いご理解を頂き、発表の機会を与えて頂いた。また、在籍当時の曽和研の皆さんには色々とお世話になった。この場を借りてお礼を申し上げます。

参 考 文 献

1) http://java.sun.com
2) R.Bigelow, "Structured Code via Stack Machine", COMPUTER, Vol.38, No.3, pp.67 (1975).
3) Koopman, Philip J., Jr.(著), 藤井敬雄 (訳), 田中清臣 (監訳), "スタックコンピューター—CISC, RISC とスタックアーキテクチャ", 共立出版 (1994).
4) R.Bruno and V.Carla, "Data Flow on Queue Machine", 12th Int. IEEE Symposium on Computer Architecture, pp.342-351 (1985).
5) 前田敦司, 中西正和, "新しい計算モデル キューマシンとその並列関数型言語への応用", 情報処理学会論文誌, Vol.38, No.3, pp.574-583 (1997).
6) M.Sowa, "Fundamental of Queue Machine", Sowa Lab. Tech. Rep. SLL97302 (1997).
7) 鈴木均, 岡本秀輔, 前田敦司, "キューマシン計算モデルに基づくスーパスカラプロセッサの実装と評価", 情報処理学会 HPC 研究会, 研究報告, No.075-16 (1999).
8) 川田宗太郎, "並列キューマシンの設計とシミュレーションによる性能評価", 電気通信大学 電子工学科 卒業論文 (2000).
9) M.Sowa et al, "Proposal and Design of a Parallel Queue Processor Architecture(PQP)," IASTED-PDCS2002, Cambridge, USA, pp.554-560 (2002).
10) 月刊アスキー編集部 (著), "標準 PC ハンドブック／なぜ CPU は速くなるのか?", ASCII (2003).
11) B.A.Abderazek, K.Nikolova, and M.Sowa, "FARM-Queue Execution Model: Towards an Alternative Computing Paradigm", Proceedings of IPSJ Symposium, Yokohama, pp.99-100 (2000).
12) B.A.Abderazek et al, "On the Design of Register-Queue Based Processor Architecture (FaRM-rq)," Lecture Note in Computer Science, Springer-Verlag, Vol.2745, pp.248-262 (2003).
13) 奥村義智, 吉永努, 曽和将容, "キューマシン用並列化 C コンパイラ", 情報処理学会 ARC 研究会, 研究報告, No.149-22 (2002).
14) L.Q.Wang et al, "QJAVAC: Queue-Java Compiler Design for High Parallelism Queue Java Bytecode", ITC-CSCC2003, Kang-Woo Do, Korea, pp.900-903 (2003).