マイクロプログラム制御計算機ＱＡ－１の並列処理方式

Low Level Parallel Processing by the Microprogrammed Machine QA-1

小柳 滋　　柴山 潔　　富田 真治　　萩原 宏
Shigeru Oyanagi, Kiyoshi Shibayama, Shinji Tomita, Hiroshi Hagiwara
（京都大学　工学部）
(Faculty of Engineering, Kyoto University)

1．まえがき

図形処理、画像処理、信号処理などの応用では大量データに対して均一な処理を必要とし、これらのリアルタイム処理を可能とするために並列処理やパイプライン処理の利用が有効である。

並列処理には、いくつかの処理レベルが考えられる。たとえば、図形処理の1例としてアニメーションを考える。その基本操作は、図形を構成する多くの座標点ベクトル(x,y)に変換行列を作用させることで表わされる。従って、アニメーションの並列処理は、次の3つのレベルが考えられる。

(1) 変換行列×座標点ベクトルの基本操作を複数の演算器を用いることにより並列処理するレベル

(2) 座標点の集合をいくつかの部分に分割し各部分に対する座標点の変換を同時に複数個、並列処理するレベル

(3) ある時刻tにおける全ての座標点の変換と、時刻t+Δtにおける変換行列を求める計算などの他のタスクとを、機能分担して並列処理するレベル

これらの3つの並列処理のレベルをとり入れて、処理効率のよいシステムをいかに構成するかは、応用に強く依存する。このため、システムの専用化による高速性と、柔軟性に関するトレードオフの検討が重要である。

一般に図形処理、画像処理、信号処理用の専用計算機[4]~[7]では、(1)のレベルの並列処理をとり入れたものが多い。これらの専用計算機では、実装するハードウェアを最小限にして、効率よく特定の応用を処理することに重点がおかれている。しかるに、特定のアルゴリズムや処理方式に依存したハードウェア構成は柔軟性の欠如をもたらし、特に研究開発過程においては、処理対象の変更などに柔軟に対応できることが要請される。

また、近年のマイクロコンピュータの普及に伴ない、(2)あるいは(3)のレベルの並列処理をとり入れたマイクロコンピュータ複合体システムの研究が盛んになってきた。これらの複合体においても、高速性と柔軟性を両立させるシステム構成が必要であると考えられる。

一方、計算機に柔軟性を与える技術としてマイクロプログラミングが注目されている。マイクロプログラム制御計算機では、制御記憶を書き換えることによって応用に適したアーキテクチャを柔軟に定義することが可能であり、ハードウェアをきめ細かく制御することにより各種応用の高速処理が可能である。

我々は、低いレベルの並列処理能力をもつ計算機をマイクロプログラムで制御することにより、高速性と柔軟性とを兼ね備えた計算機ＱＡ－１を新たに設計、製作した。その設計目標は次の2点である。

(1) 図形処理、画像処理、信号処理など大量データの高速リアルタイム処理を目的とする応用において、充分な処理能力を提供できること

(2) 各種計算機のエミュレーション、高級言語処理など問題適応型計算機アーキテクチャ研究のための実験用計算機として利用できること

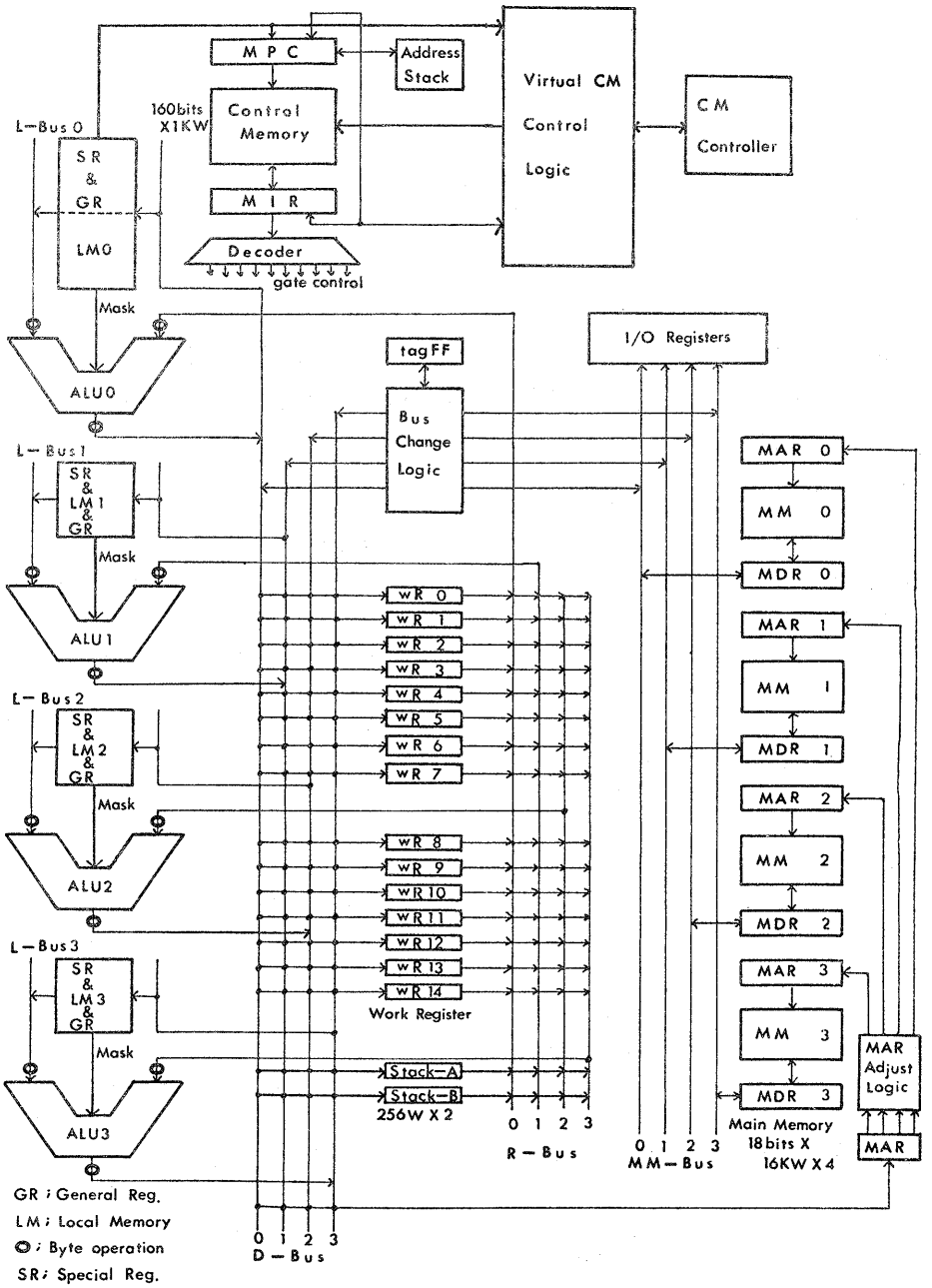
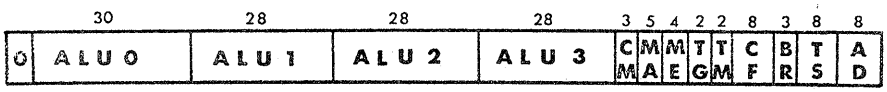


Fig. 1　QA-1の概略図

Long format ;



Short format ;

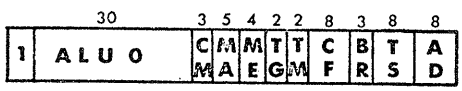


Fig. 2　マイクロ命令形式

ＱＡ－１のアーキテクチャとハードウェア構成については既に報告しているので[1]、本稿では設計目標(1)における各種応用で、ＱＡ－１の並列性がどのように生かされているかということに重点をおいて述べる。

２．ＱＡ－１の特徴

ＱＡ－１は図形処理、画像処理、信号処理などの応用における基本操作に含まれる並列性を利用して高速化をめざしている。そのため、ＱＡ－１は４個のＡＬＵ(Arithmetic and Logic Unit)を持ち、それらが１５個のＷＲ(Working Register)を共有しつつ、並列動作する構成をとっている。ＱＡ－１の概略図をFig.１に示す。

各ＡＬＵは同一能力を持ち、その右入力はＲバスを経由してＷＲに、左入力はＬバスを経由して各ＡＬＵ固有のＬＳ(Local Storage)に接続されている。その出力はＤバスを経由してＷＲ、ＬＳいずれにも接続されている。各ＡＬＵの演算、並びにＡＬＵ演算に使用されるＷＲ、ＬＳのアドレスは、マイクロ命令により指定される。マイクロ命令の形式をFig.２及びFig.３に示す。

図に示されるように、各ＡＬＵは異なるフィールドで制御される。マイクロ命令の系列は単一であるが、１マイクロ命令で複数の演算、オペランド・アドレスを指定するため、マイクロ命令レベルでのＭＩＭＤ方式と考えられる。マイクロ命令形式は、並列処理の特徴を生かすために１６０ビット水平型を採用している。更に並列処理がおこなわれない場合、すなわちＡＬＵ0のみが動作する場合には８０ビットのShort formatを用いることにより、制御記憶の節約を計っている。この２種類のformatは、マイクロ命令語の先頭ビットにより判別される。

一般に、並列処理システムにおいては共有メモリへのアクセスの競合により、効率の低下をもたらす。ＱＡ－１においては、ＷＲへのアクセスの競合が生じないような構成をとっている。即ち、各ＡＬＵは任意のＷＲを同時にアクセスすることができる。複数のＡＬＵの演算結果が同一ＷＲに書き込まれる場合には、その論理和が書き込みデータとなる。

マイクロプログラム制御計算機においては、その演算が主にレジスタレベルとなるため、ＡＬＵとＭＭ(Main Memory)との結合方式が重要な問題である。特に、図形処理、画像処理のように大量データの高速処理が必要となる応用では、ＭＭへの高速かつ柔軟なアクセスが可能でなければならない。ＱＡ－１においては、ＭＭはサイクルタイム３５０ns，容量１６k語(１語１８ビット)のＭＯＳＩＣメモリ４バンクから構成されている。また、４個のＡＬＵの配列順位がハードウェアで固定されているため、種々の語長のデータを効率よくアクセスするためには、４バンクに分割されたＭＭと４個のＡＬＵが順序正しく接続されねばならない。そのため、ＱＡ－１ではＢＣＬ(Bus Change Logic)を装備してこれを実現している。

ＱＡ－１のＣＭ(Control Memory)はアクセスタイム40nsのバイポーラＩＣメモリで構成され、容量は１０２４語×１６０ビットが実装されている。種々の応用分野に適用させるためには、この容量では充分とは言えないが、語長が１６０ビットと長いため、実装するハードウェア規模やコスト面からの制限も強い。そのため、ＣＭを仮想記憶方式としている。そのバックアップメモリとしてＭＭを用い、ハードウェアで制御することにより高速化している。ＣＭは８個のブロック(128語)に分かれ、ブロック単位で入れ替えがおこなわれる。

以上が、ＱＡ－１のハードウェア構成の主な特徴であるが、ＱＡ－１の設計にあたって基本的に拡張可能性を重視している。すなわち、将来の変更、追加が可能なようにハードウェアはできる限りモジュール化されている。

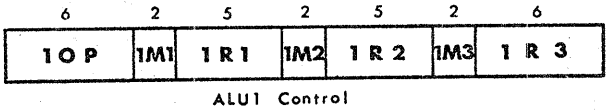


Fig.３　ＡＬＵの制御部

3. 応用例

3.1 図形処理（アニメーション）

図形の取扱いには、ベクトルや行列演算に関する処理が多い。2次元図形の運動は、パラメータwを導入したホモジニアス座標(x, y, w)を用いて、3行3列の行列を作用させることで表わされる。例えば、図形の平行移動、回転、拡大／縮少は次のようになる。

・平行移動

(x, y)方向に(a, b)だけ平行移動する

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

・回転

原点を中心としてαだけ回転する

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \leftarrow \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

・拡大／縮少

原点中心にk倍だけ拡大／縮少すると

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \leftarrow \begin{pmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

これらの変換を組み合わせることにより、簡単な2次元図形のアニメーションができる。我々はFig.4に示されるシステムを用いてアニメーション・システムを作成している。すなわちタブレットより図形を入力し、その運動をリアルタイムで処理しつつグラフィック・ディスプレイに表示する。

一般にグラフィック・ディスプレイで図形がなめらかに運動するように人の目で認識されるためには、1秒に20～50コマの運動図形の生成が必要とされる。我々は千本のベクトルより成る図形の運動をリアルタイムで処理することを目標とした。そのためには、1ベクトル当りの変換を約20μsec以内で処理しなければならない。

本システムにおいてQA－1は変換行列の計算、座標点の変換の処理を受け持ち、変換された図形はDMA転送によりHITAC 10－IIを経由してVarianに送られる。変換行列の計算は3行3列の行列積の計算が中心であり、これを3個のALUを用いて並列処理している。また座標点変換は、3行目は計算不要であるため、

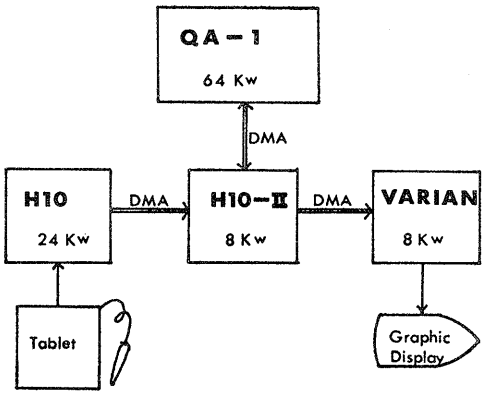


Fig.4　システム構成

2行3列の行列と3次元ベクトルの積で表わされ、2点の変換を同時に処理することができる。この時の各ALUの演算は次のようになる。

$$\text{ALU}\phi : x_{2n}^{t+1} \leftarrow a_{11} x_{2n}^{t} + a_{12} y_{2n}^{t} + a_{13}$$

$$\text{ALU}1 : y_{2n}^{t+1} \leftarrow a_{21} x_{2n}^{t} + a_{22} y_{2n}^{t} + a_{23}$$

$$\text{ALU}2 : x_{2n+1}^{t+1} \leftarrow a_{11} x_{2n+1}^{t} + a_{12} y_{2n+1}^{t} + a_{13}$$

$$\text{ALU}3 : y_{2n+1}^{t+1} \leftarrow a_{21} x_{2n+1}^{t} + a_{22} y_{2n+1}^{t} + a_{23}$$

変換行列の要素$(a_{11} \sim a_{23})$を重複してLSに格納し、座標値$(x_{2n}^{t}, y_{2n}^{t}, x_{2n+1}^{t}, y_{2n+1}^{t})$をWRに格納することにより、4個のALUはほとんど無駄なく動作する。

我々は、本システムを用いて花の回りを蝶々が羽根を開閉しつつ飛び回るアニメーションを製作した。座標点数は512点であり、これをリアルタイムでなめらかに表示することができることを確認した。

3.2 画像処理

画像前処理の例として、濃淡画像の線図形化の手法であるラプラス演算子による操作を示す。この操作では、Fig.5に示すように2次元画像の画素Eとその隣接点B, D, H, Fに対して、B＋D＋H＋F－4Eの演算を行なうことにより線図形の画素E'が得られる。この操作は通常全画面に対して行なわれる。また、FSSなど画像入力装置による濃淡画像は高々8ビットでディジタル化されている。

(1) 2次元画像の主記憶内での配置

1画素8ビットの2次元画像の2行画素分に

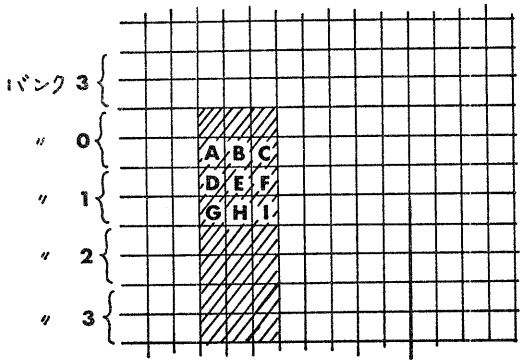


Fig.5 2次元画像の主記憶内での配置

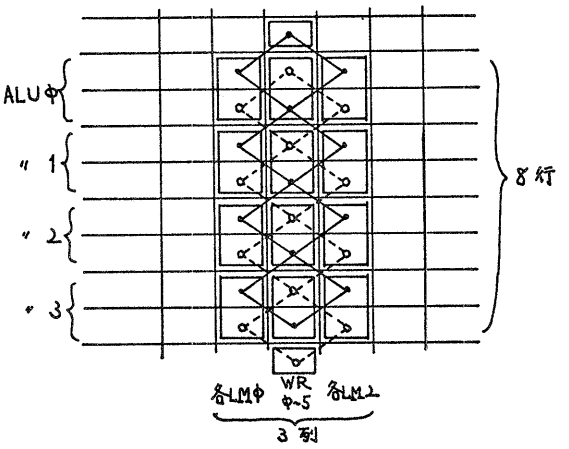


Fig. 6 ラプラス演算子

対応するデータが16ビットデータ幅の主記憶の各バンクにFig.5に示すように格納される。たとえば、画像の1，2行，3，4行，5，6行および7，8行のデータは、それぞれ主記憶のバンクφ，1，2および3に格納され、以下この順にデータ配列をとるものとする。

(2) 2次元画像に対するラプラス演算

QA－1では語長64ビットの4バンク主記憶装置を採用したので、8画素に対応するデータが同時にアクセスされ、各ローカルメモリの2番地（LM2）に格納される。Fig.5の斜線部で示す主記憶から読み出された8行3列の画素は、各ALUのローカルメモリLMφ，1，2に格納される。演算実行時における各画素データのレジスタ内での配置をFig.6に示す。ここで、WRφ，5に対応する画素は演算開始時に同時に主記憶より読み出され、またWR1～4にはLM1より1マシンサイクルでデータの転送がなされる。Fig.6の実線及び点線は、ラプラス演算を2度に分割して実行することを示している。たとえばALUφでは、最初LMφ，1の上位8ビットとWRφの下位8ビット、およびWR1との間でラプラス演算を行ない、引き続いてLMφ，1の下位8ビット、WR2の上位8ビット、およびWR1の間で演算を行なう。その後、2つの8ビットの演算結果を16ビット長データにパックして主記憶に書込む。ALU1～3も同時に全く同様の演算を行なう。この並列演算によって8画素分の縁図形が得られる。

3.3 信号処理

ディジタル信号処理を高速におこなうアルゴリズムとしてFFT（Fast Fouvier Transform）が普及しているが、我々はQA－1のファームウェアでFFTをインプリメントしている。FFTのアルゴリズムとしてCooley Tukeyアルゴリズムをとりあげ、これをin place形で実行する場合を考える。標本数を$N=2^n$のFFTの第ℓステージの演算は、次式の計算の繰返しである。

$$X_\ell(k) = X_{\ell-1}(k) + W^P \cdot X_{\ell-1}(k+2^{n-\ell})$$

$$X_\ell(k+2^{n-\ell}) = X_{\ell-1}(k) - W^P \cdot X_{\ell-1}(k+2^{n-\ell})$$

但し回転因子は

$$W^P = \exp\left(-i\frac{2\pi P}{N}\right)$$

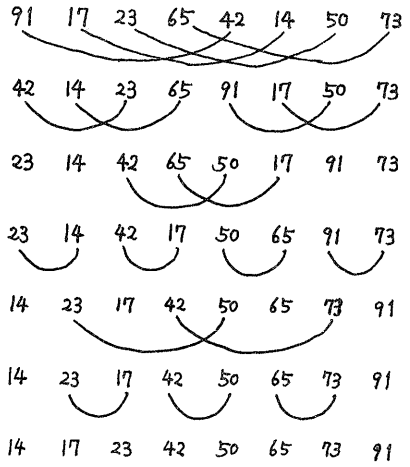
であり、Pはk／$2^{n-\ell}$の逆2進表示となる。

データ長を実数部、虚数部各16ビットの32ビットとすると、QA－1では主記憶4バンク64ビットが同時にアクセスできるため、$X_\ell(k)$、$X_\ell(k+2^{n-\ell})$と$X_\ell(k+1)$、$X_\ell(k+2^{n-\ell}+1)$とを同時に計算すると、主記憶のアクセス回数を減少させることができるので効率がよい。上記のFFTの基本演算は4個のALUを用いて並列化することができ、乗算器などの機能を用いて効率よく実行することができる。計算は16ビットの固定小数点演算で行ない、オーバーフローが生じた時には、マイクロ命令レベルの割込み機能を用いて、現在実行中のステージにおけるデータセットを右シフトすることにより処理している。

3．4　ソーティング

大量データの高速処理の簡単な例としてソーティングを考える。現在の所、ＱＡ－１には２次記憶がないため、internal sortingを試みた。また、プログラムの単純化のため、１６ビットデータをin place形で小さい順に並べ換える例を考える。

４個のＡＬＵを効率的に用いるアルゴリズムとして、Batcher's Method[8]を採用した。このアルゴリズムでは、各ステージにおけるすべての比較が並列に実行できる特徴があり、ＱＡ－１上のインプリメントも比較的単純に効率よくできる。データ数が8個の場合の、本アルゴリズムによるデータの処理過程を以下に示す。



3．5　初等関数計算（ＣＯＲＤＩＣ）[9]

数値処理の簡単な例として、ＣＯＲＤＩＣアルゴリズムによる初等関数計算を考える。このアルゴリズムはFig.7に示されるように乗算を用いずに加減算とシフト操作のみで初等関数が計算できる所に特徴がある。従って、乗算器を備えたＱＡ－１では、多項式近似と比較して高速ではないが、ファームウェア化による効率の比較に興味があったのでインプリメントした。

精度は２４ビットで演算は３２ビットで行なう。Fig.7の同一box内にある演算は並列処理可能である。しかるに、ＱＡ－１ではＡＬＵのデータ幅が１６ビットであるので、ＡＬＵ２個を連結して３２ビットの演算をおこなう。そのため、ＱＡ－１では3個以上の演算を同時に実行することはできない。

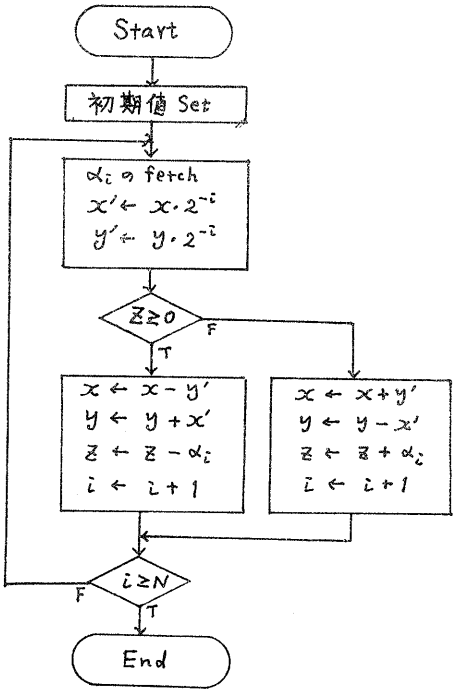


Fig.7　CORDIC アルゴリズム

4．ＱＡ－１の評価

第３章で説明した応用例をＱＡ－１のファームウェア、ＨＩＴＡＣ８３５０のファームウェア、及びＨＩＴＡＣ１０のアセンブリ言語でインプリメントした場合の処理速度と実行ステップ数をTable 1に示す。また、ＱＡ－１のファームウェアにおける各種評価データをTable 2に示す。

以下、このデータについて簡単に説明する。図形処理においては、乗算の比率が高い。Ｈ－８３５０では、ハードウェア乗算器を装備していないため、乗算に費す時間が大部分となる。ＱＡ－１では、乗算の高速化のみならず、多倍長演算、マスク処理などを有効に生かされ、ＡＬＵの使用率はほぼ4に近い。

画像処理では、バイト演算機能が大きな役割を果たす。ＱＡ－１，Ｈ－８３５０共にハードウェアによるバイト選択機能を持つ。また、主記憶の語長がそれぞれ６４ビット、３２ビットであるので、主記憶参照回数を減らすことができる。

信号処理の評価はまだおこなっていないが、ＦＦＴの主要部分では4個のＡＬＵはほとんどfullに使用される。この場合も、ハードウェ

ア乗算器、倍長演算機能、主記憶の語長などの効果により、かなりの効率向上が期待できる。

ソーティングにおいては、論理がマシン命令レベルに適しているため、ファームウェア化しても実行する命令数はあまり変わらない。また主記憶参照率が高いため、QA－1での効率向上の比率は他の例よりも低くなる。

CORDICでは、32ビット演算をおこなうためQA－1の並列度は2となるが、多重シフト機能、定数発生機能が強化されているためH－8350と比較して、かなりの効率向上が達成できる。

以上に述べたように、ファームウェア化による効率向上は、実装されているハードウェア機能に強く依存する。QA－1は、高速性と柔軟性を兼ね備えることを設計目標としているのでハードウェアによる特殊化を避けている。しかし、ハードウェアコストの低廉化に伴ない、特殊目的のハードウェアユニットを付加して、一層の効率向上を達成することも検討する必要があろう。以下では、QA－1で実装されているハードウェアユニットと、それを制御するマイクロ命令の構成が、各応用例にどのように生かされるかについて述べる。

・4個のALU

前章で述べた応用例では、4個のALUの使用率はかなり高い。これらの例では本質的な並列性を利用しているので、マイクロプログラムの作成は、さほど困難なことではない。また、ALUを連結して倍長演算を行なうことができるので、幅広く応用することができる。

ALUの個数の評価はきわめて重要な問題である。QA－1の設計においては、ハードウェアの規模（マイクロ命令長、WRの数、制御記憶、主記憶の構成、容量など）を考慮し、かつ大量均一データの高速処理と共に汎用機としての問題適応型マシンを効率よく実現できる数として4個と決定した。

・Working Register（WR）

QA－1では、15個のWRが4個のALUに共有されている。この構成により、レジスタ間の無駄な転送を減少させることができ、レジスタレベルの並列処理を真に有効なものとしている。Table 2に示されるように、前述の例において大部分のWRが使用されている。WRの数は多い程効率は向上し、マイクロプログラムの作成も容易になる。しかしながら、ハードウェア規模を考慮に入れると、WR15個は妥当と考えられる。

・MIMD制御

4個のALUは異なるフィールドにより制御される。前章で述べた応用においては、その大部分でALUは同じ演算を行なう。しかしながら、1割～2割の比率で異なる演算を行なっている。これらは、レジスタ類の初期設定、主記憶のアドレス計算、繰り返し回数のカウント等によく用いられる。前述の例のように本質的に並列処理可能な場合でも、MIMDによる柔軟性の効果は大きい。更に、より広い応用分野に適応させるためには、MIMDの効果は大きいと考える。特に、エミュレーションにおいて、命令のfetchルーチンでオペランドアドレス計算等を並列に実行することができる。

・主記憶の構成

マイクロ命令レベルでの並列処理において、主記憶とALU間の結合は重要な問題である。特に、画像データのような2次元記憶空間をもつものに対して効率よくアクセスできる機能を持たせなければならない。QA－1では4バンクの主記憶をもち、それらを連続したものとしてみるモード（ノーマル・モード）と、それらを切り離されたものとしてみるモード（スペシャル・モード）の2種類のアクセス方法を用意している。ノーマル・モードでは、BCLにより倍長データを効率よくアクセスできる。画像処理ではスペシャル・モードを用いることにより、効果をあげている。

・ハードウェア乗算器

図形処理、信号処理においては乗算の比重はきわめて高い。これらの処理をリアルタイムでおこなうことをめざすためには、乗算の高速化は不可欠である。従って、我々は16ビット×16ビットのハードウェア乗算器を各ALUに設置した。これにより、図形処理の高速化が達成されている。

・バイト演算機能

画像処理では8ビットデータの処理が多く、1語16ビットに2画素を格納する場合には、1語から1画素をとり出す機能や、処理された2画素を1語にパックする機能を効率よくおこ

				QA-1 firmware	H-8350 firmware	H-10 Assembler
図形処理	座標点変換 $\begin{pmatrix} a_{11} a_{12} a_{13} \\ a_{21} a_{22} a_{23} \end{pmatrix} \times \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$	1点の変換	処理速度	7.2 μs	104 μs	669 μs
			実行 Step	19	260	217
	変換行列の積	(3×3) 行列	処理速度	32.1 μs	612 μs	2250 μs
			実行 Step	89	1530	600
画像処理	Laplacian 法 B+D+F+H−4E	1画素の処理	処理速度	0.9 μs	4.6 μs	103 μs
			実行 Step	2	10	31
	Gradient 法 |A+B+C−G−H−I| +|A+D+G−C−F−I|	1画素の処理	処理速度	1.65 μs	7.3 μs	209 μs
			実行 Step	4	18	60
ソーティング	Batcher's Method	1回の compare	処理速度	1.6 μs	5.4 μs	41.3 μs
			実行 Step	3.25	11	13
CORDIC	SIN, COS の計算		処理速度	63 μs	244 μs	3000 μs
			実行 Step	157	610	800

Table 1　効率の比較

	プログラム サイズ		平均 ALU 使用率	WR 使用総数	主記憶 参照率
	long format	short format			
座標点変換	47	3	3.7	14	5%
変換行列積	33	1	3.2	12	0%
Laplacian 法	13	3	3.4	12	19%
Gradient 法	30	3	3.5	14	9%
FFT	42	5	3.2	13	18%
SORTING	33	7	3.1	14	31%
CORDIC	53	2	2.7	6	0%

Table 2　QA-1 firmware の評価データ

なうことが必要である。従って、QA－1においては、マイクロ命令のM1, M2, M3部によりALUの入出力バス上のデータに対してバイト修飾する機能をもたせた。また、この機能は8ビット論理シフトと同じ働きをするので、CORDICのように多重シフトが必要なアルゴリズムにおいては、効果的に利用することができる。

・定数発生機能

マイクロプログラムの作成において、定数演算はかなりの比率で必要となる。たとえば、レジスタの初期値の設定、マスク処理、カウンタ類の更新などにおいて定数演算が必要となる。QA－1においては、各ALUが16ビット定数を発生する機能を持ち、更に5ビット定数と任意のレジスタとの直接演算が可能である。この機能により、マイクロプログラムの作成が容易になり、かつ実行効率もよくなる。

・多重シフト機能

CORDICの高速化を計るためには、多重シフトを効率よく行なうことが最も重要である。更に、エミュレーションを考える時、1語中の任意のフィールドを取り出して処理を行なうためには、多重シフト機能は重要である。そのため、QA－1においては4ビットまでの論理、算術シフト機能が1マイクロ命令で実行でき、またバイト演算機能を併用すれば8～12ビットシフトも1マイクロ命令で実行できる。更にALUの連結による倍長シフトも可能である。この機能によりCORDICの高速化はもとより、他の種々の応用に用いられる。

・Disable Flip-Flop

QA－1のように単一のマイクロ命令系列で複数の演算を制御する場合には、ALUの演算をデータにより制御する機能が必要となる。たとえば、4個のレジスタWRφ～3の内容の絶対値をWRφ～3に格納する例を考える。数は2の補数表示の整数とし、先頭ビットが符号を表わす。このときのマイクロプログラムを次に示そう。

最初のステップではDisable F.FにWRφ～3の符号ビットを反転した値をセットする。

次のステップではWRφ～3の2の補数がWRφ～3に格納される。但し、Disable F.Fが1であるALUに関してはこの演算は実行されず、従ってその時のWRには値は代入されず前の値を保持している。この2ステップのマイクロ命令により、WRφ～3の絶対値がWRφ～3にそれぞれ格納されることになる。

```
(φ) ← (WRφ)
(φ) ← (WR1)
(φ) ← (WR2)
(φ) ← (WR3)
  Set D=¬ALU\S

(WRφ) ← −(WRφ)
(WR1) ← −(WR1)
(WR2) ← −(WR2)
(WR3) ← −(WR3)
  Reset D
```

・順序制御

マイクロ命令の順序制御は、その効率を左右し、かつマイクロプログラムの書き易さにも影響を及ぼす。QA－1においては、2方向分岐4方向分岐、レジスタの内容による間接分岐、マイクロ・サブルーチンへのリンク等の分岐方法を用意している。また、仮想記憶記憶方式におけるページングによる分岐の制限、2種類のマイクロ命令形式による複雑なアドレッシングは、HITAC－10上で作成されたマイクロプログラム・アセンブラにより完全に吸収することができる。

また、マイクロ命令レベルでの割込み機能により、特殊状態の検出が容易におこなえる。

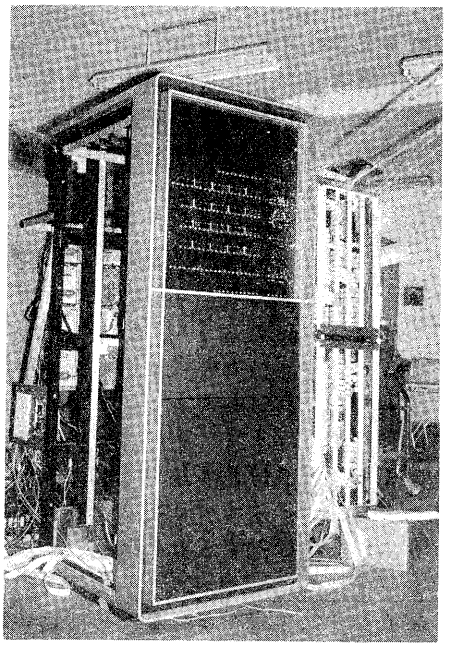
5. むすび

QA－1は低いレベルの並列性、すなわち図形処理、画像処理、信号処理などにおける基本操作内に含まれる本質的な並列性を効率よく利用することにより、処理の高速化を達成している。3、4章で述べたように、これらの応用分野において、ほぼ満足できる処理速度の向上が実現できた。更に、QA－1はこれらの応用のみに適する専用計算機でなく、ALUを連結して64ビットマシンを実現できることなどの汎用性をもっている。今後、高級言語マシン、各種計算機のエミュレーションなどのマイクロプログラミングの応用分野について実験をおこない、QA－1がいかに役立つかについて調べる

予定である。

最後に、ＱＡ－１のサポート・ソフトウェアについて若干考察する。現在、ＨＩＴＡＣ１０上で作成されたマイクロプログラム・アセンブラ[3]と、ＨＩＴＡＣ８３５０上で作成されたマイクロプログラム・シミュレータ[2]が動作している。そして、ＨＩＴＡＣ１０上で会話型マイクロプログラム・デバッガーと、任意の形式の機械語を定義することができるアセンブラを作成している。今後の課題として、マイクロプログラムの並列化を自動的におこなうソフトウェアの開発を検討している。

なお、ＱＡ－１の外観を右に示す。そのハードウェア規模は、ＴＩ社のＳＳＩ，ＭＳＩ，ＬＳＩ約４５００個で構成されている。ハードウェアの設計、製作には、約６人年要した。



本研究は、文部省科学研究費によって作成した。

〔謝辞〕

ＱＡ－１のハードウェア、ソフトウェア両面におけるサポート・システムを設計、製作していただいた本学大学院生の石田享君、山崎勝弘君、本学卒業生の野村伸介君（現日本電気）、宮脇保裕君（現ＮＣＲ）に深く感謝します。

〔参考文献〕

（１）小柳，柴山，富田，萩原　「図形処理用マイクロプログラム制御計算機」　情報処理学会計算機アーキテクチャ研 CA15-1 Jun. 1975

（２）山崎，他　「マイクロプログラム制御計算機ＱＡ－１の構成とそのマイクロプログラムシミュレータ」　情報処理学会第17回全国大会 27 Dec. 1976

（３）石田，他　「マイクロプログラム制御計算機ＱＡ－１のマイクロプログラム・アセンブラ」　情報処理学会第17回全国大会 28 Dec. 1976

（４）H. Webber　「SIMALE」　ACM Sigmicro 3 P25 1974

（５）B. Kruse, et.al　「A Parallel picture processing machine」　IEEE Trans. C-22, 12 P1075 1973

（６）V. Tantillo, et.al　「A Research Oriented Dynamic Microprocessor」　IEEE Trans. C-22, 11 P996 1973

（７）相磯，他　「高速パイプライン信号処理装置のマイクロプログラム制御」　信学論（D） 58-D, 6 P328 1975

（８）D.E. Knuth　「The Art of Computer Programming Vol. 3 Sorting and Searching」　Addison-Wesley Publishing Co. P111 1973

（９）J.S. Walther　「A unified algorithm for elementary functions」　AFIPS Conf. Proc. SJCC Vol.38 P379 1971