

放送用実時間映像信号処理システム
に関する研究

八木 伸行

目次

[1] 序論	p.	1
1. 1 研究の背景と意義	p.	1
1. 2 本論文の構成	p.	2
[2] 映像信号処理に関する検討	p.	5
2. 1 映像信号のデジタル処理	p.	5
2. 2 幾何学変換	p.	8
2. 2. 1 幾何学変換回路	p.	8
2. 2. 2 アドレス変換処理	p.	9
2. 2. 3 拡大・縮小における演算語長	p.	10
2. 2. 4 回転における演算語長	p.	12
2. 2. 5 アフィン変換における演算語長	p.	13
2. 2. 6 補間処理	p.	13
2. 2. 7 補間処理の画質評価	p.	17
2. 2. 8 補間処理の演算語長	p.	21
2. 3 波形発生	p.	23
2. 3. 1 ビデオスイッチャ	p.	23
2. 3. 2 効果波形発生回路	p.	24
2. 4 映像合成	p.	27
2. 4. 1 クロマキーとビデオマット	p.	27
2. 4. 2 提案する映像合成法	p.	27
2. 4. 3 動ベクトル検出法	p.	28
2. 4. 4 シミュレーション実験	p.	31
2. 4. 5 3次元運動する剛体への適用法	p.	36
[3] 映像信号処理システムのアーキテクチャ	p.	41
3. 1 基本概念	p.	41
3. 2 並列処理方式	p.	42
3. 3 メモリ構成法	p.	48
3. 4 同期方式	p.	50
3. 5 プログラムロード方式	p.	51
[4] 実験システム R T V P	p.	53
4. 1 設計指針	p.	53
4. 2 アーキテクチャ	p.	53
4. 2. 1 概要	p.	53
4. 2. 2 映像転送路	p.	56

4. 2. 3	画像メモリ構成法	p.	56
4. 2. 4	2段階時間軸処理方式	p.	56
4. 2. 5	プログラムロード方式	p.	57
4. 3	ハードウェア構成	p.	58
4. 3. 1	プロセッシングユニット	p.	58
4. 4	ソフトウェア構成	p.	59
4. 4. 1	処理の流れ	p.	59
4. 4. 2	ソフトウェア開発環境	p.	63
4. 5	各種映像信号処理の適用と評価	p.	65
4. 6	改善すべき課題	p.	68
[5]	実用システム Picot-system	p.	73
5. 1	設計指針	p.	73
5. 2	アーキテクチャ	p.	73
5. 2. 1	概要	p.	73
5. 2. 2	並列処理方式	p.	76
5. 2. 3	メモリ構成法	p.	76
5. 2. 4	同期方式	p.	77
5. 2. 5	プログラムロード方式	p.	78
5. 3	ハードウェア構成	p.	78
5. 3. 1	プロセッサ	p.	78
5. 3. 2	プロセッサの制御機構	p.	84
5. 3. 3	プロセッサとコントローラの通信機構	p.	88
5. 3. 4	ネットワーク	p.	89
5. 3. 5	コントローラ	p.	90
5. 3. 6	プロセッサLSIとネットワークLSI	p.	90
5. 4	ソフトウェア構成	p.	94
5. 4. 1	処理の流れ	p.	94
5. 4. 2	ソフトウェア開発環境	p.	95
5. 4. 3	PicPEn	p.	100
5. 5	各種映像信号処理の適用と評価	p.	101
5. 6	適用例	p.	104
5. 6. 1	ビデオスイッチャ	p.	104
5. 6. 2	特殊効果	p.	107
5. 6. 3	画質補正	p.	107
5. 6. 4	動き検出	p.	110
5. 6. 5	映像合成	p.	112

[6] 結論	p. 117
謝辞	p. 119
参考文献	p. 121
発表論文等リスト	p. 127
付録1 プロセッサ用言語仕様概説	p. 131
付録2 ネットワーク用言語仕様概説	p. 137

[1] 序 論

1. 1 研究の背景と意義

1950年代の文字パターン認識の研究に始まったデジタル画像処理技術は、半導体技術、コンピュータ技術の進展とともに、社会のニーズの合わせて発展し、現在では医学、リモートセンシング、FA (Factory Automation)、OA (Office Automation) など幅広い分野で使われている。

このようにますます重要となる画像処理であるが、処理内容の高度化、実時間処理が求められてきている。さらに最近では、動画も扱うようになり、ビデオレートで処理することも求められつつある。このような要求に対して、汎用のコンピュータでは対応しきれないため、専用の画像処理装置が研究開発されている⁽¹⁾。

一方、放送における映像信号処理は、従来アナログで処理されてきたが、1972年のデジタルTV方式変換装置を手始めに、デジタル方式への移行が始まった⁽²⁾。当初は、アナログ回路では実現できない機能をデジタルで実現するといった方法が採られていたが、高画質、高安定などのメリットから、あらゆるスタジオ用映像機器のデジタル化が進められるようになった。ところが、これらは内部処理のデジタル化であって、入出力は相変わらずアナログ映像信号のままであった。このため、機器を接続して処理を重ねると、A/D、D/Aが何回も繰り返されることになり、高画質化などのデジタル化のメリットが生かされないままであった。そこで、トータルデジタル化、すなわち入出力も含めたシステム全体のデジタル化が、時代の要請となった。このような状況の中、1982年にCCIR (国際無線通信諮問委員会)において、デジタル規格が決められ、スタジオ設備のトータルデジタル化が図られるようになった。しかし、アナログシステムを単純にデジタルシステムに置き換えただけでは、高画質化、高安定化だけの改善に留まり、投資に対して得られるものがあまりに少ない。デジタル化によるメリットは、アナログ方式では得られなかった柔軟性に富んだシステム構成が可能になり、システム構成法の変革をもたらすことが一番大きいのではないだろうか。

従来、放送局のスタジオにおける映像信号処理は、ビデオスイッチャや特殊効果装置など専用機器を組み合わせて行っており、新しい映像効果や機能に対しては、要求ごとに新規に専用機器を開発してきた。しかし、このような方式では、放送メディアが増大し、番組制作手法が多様化し、処理内容が複雑多岐にわたるようになってきている昨今、経済性、即応性の面でとても追いつかない状況になってきている。例えば、

(1) 新しい映像表現のために次々と特殊効果が開発されてきたが、飽きられて使われなくなってしまうものも多く、コスト的にも問題がある。

(2) 画質に難点があるものの再収録の困難なビデオテープを後処理で扱いたい場合があるが、障害の状況はテープ毎に異なり、多様な障害に対応するために機器をいちいち開発することは困難であり、即応性に欠けるなどである。

これらに対して、同一のハードウェア上で、ソフトウェアを入れ換えることによって対応するようになれば、広範な放送局のニーズに、柔軟にしかもすばやく対応できるようになるはずである。ハードウェア(ワイヤードロジック)のソフトウェア化は、時代の流れに沿うものであり、機器の制御がワイヤードロジックによる制御からマイクロコンピュータによるプログラム制御に変わって、一つのコントローラでプログラムを変えることにより、きめ細かにかつ柔軟に対応できるようになったのと同じである。

プログラムで処理する装置といえば、従来よりコンピュータがある。また、用途を信号処理に限定してフィルタリング処理を高速に行えるアーキテクチャを有するDSP(Digital Signal Processor)¹³⁾も1980年代初頭に誕生し、音声やモデムなどに適用され始めていた。しかし、映像信号を処理するには、毎秒数10Mbyteものデータを扱わなければならない。1個のデータに仮に100ステップの処理を行うとしても数1000MOPS(Million Operation per second)の処理能力が要求され、現状の機器の処理能力、処理機能との間に大きな落差があった。

そこで筆者は、標準TV、HDTV(High Definition TV)、リモートセンシング画像などあらゆる画像・映像信号をプログラムにより処理できる汎用の画像・映像信号処理装置を作ること目標に画像・映像信号用のリアルタイムプロセッサの研究に着手した。この概念を実現するために、まず小規模なマルチプロセッサ構成による実験システムRTVP(Real-time video signal processor)を試作した¹⁴⁾。このRTVPの試作により実用化への糸口を見だし、得られた知見を踏まえて、映像信号用のLSI化プロセッサを開発し¹⁵⁾、マルチプロセッサ構成による大規模な実用システムPictosystem(Picture computer system)を開発した¹⁶⁾。さらに、Pictosystemを中核としたビデオスイッチャを実用化し¹⁷⁾、NHK放送センタの番組制作設備として運用している。

1. 2 本論文の構成

本論文は、研究の背景と意義について述べた第1章に引き続いて、以下に示すように構成されている。

第2章では、放送用の映像信号処理の特質について考察を加える。特に、幾何学変換、効果波形発生、映像合成を取り上げ、放送用映像信号処理装置が具備すべき演算機能、演算精度について検討する。

第3章では、プログラム可能なビデオレート映像信号処理システムのアーキテクチャについて検討し、放送用の実時間映像信号処理装置の構成法について提案する。特に、並列処理方式、メモリ構成法、プロセッサ間同期方法、プログラムロード方式について取り上げて述べる。

第4章では、第2章、第3章の検討結果を踏まえて試作した実験システムRTVPのハードウェア、ソフトウェア構成について述べる。RTVPは、プログラム可能なビデオレート映像信号処理システムの実現性の検証を行うために試作した実験システムであるが、RTVPの処理能力などを評価し、実用化する上での問題点、改善すべき点などについて述べる。

第5章では、実験システムRTVPの試作によって得られた知見に基づき、開発した実用システムPicot-systemについて述べる。Picot-systemは、14.3MHzの速度で複数のカラー動画像をビデオレート処理可能なマルチプロセッサであるが、小型化のためにプロセッサLSIとネットワークLSIの2種のLSIを開発しており、第5章ではこれらのLSIも含めハードウェア、ソフトウェア構成について述べる。また、実用化した各種応用例として、ビデオスイッチャ、画質補正、動き検出、映像合成などについても言及する。

第6章では、研究成果についてまとめ、今後の研究課題について述べることにより結語とする。

[2] 映像信号処理に関する検討

2. 1 映像信号のデジタル処理

放送における映像信号処理は、従来アナログで行われてきたが、1972年のデジタルTV方式変換装置を契機にデジタル方式への移行が始まった。当初は、アナログ回路では実現できない機能をデジタルで行うといった方法が採られていたが、現在では、高画質、高安定などのメリットから、あらゆる装置のデジタル化が進行している。

元来、映像信号処理では、『1枚の画像を1/30秒以内に途切れることなく処理する』リアルタイム性が必須であるため、アナログで1次元映像信号を直接処理していた。しかし、デジタル素子技術の進歩から、高速にデジタル処理することが可能になり、さらに大容量メモリが安価に手にはいるようになり、2次元さらに3次元(2次元+時間)で処理できるようになってきた。

デジタル映像信号処理では、映像信号の標本化に当たって、コンポジット(輝度信号に色信号を重畳したNTSC信号)で処理する場合と、コンポーネント(輝度信号と色信号が別になった[R.G.B], [Y.R-Y, B-Y], [Y, I, Q])などの信号形式)で処理する場合がある。

コンポジットで行う場合は、カラーサブキャリアの4倍の周波数である14.3MHzの標本化周波数を採用するのが一般的である。NTSC信号の信号帯域が4.2MHzであるので、標本化周波数としては、標本化定理による必要最低限の標本化周波数の8.4MHz以上あればよいが、

- ・A/D, D/Aの繰り返しによる劣化が少ない。
- ・プリフィルタ、ポストフィルタに特性の良いものを使える。
- ・サブキャリアによるビート妨害が目立たない。
- ・標本点が正方形格子状に並び、ライン間、フレーム間の処理がやりやすい。
- ・カラーエンコード、デコードの処理がやりやすい。

などの理由から通常14.3MHzが使われている¹³⁾。

また、コンポーネントの場合は、上記のような制約条件は少なく、信号帯域の2倍以上であればいくらでもよいが、通常はCCIR勧告601¹⁴⁾の13.5MHzが使われている。

HDTV (High Definition TV, 高精細度TV)については、世界中で各種の方式が提案されているが、日本ではBTA(放送技術協議会)規格¹⁵⁾の走査線数1125本/60フィールドのものが使われている。この場合は74.25MHzの標本化周波数が使用される。

表2.1に、本論文で主対象とする映像信号の規格を示す。表2.1下部

表 2.1 映像信号規格と画素数

	NTSC		HDTV
走査線数	525本		1125本
フィールド周波数	59.94Hz		60Hz
フレーム数	毎秒約30コマ		毎秒30コマ
信号帯域	4.2MHz		30.0MHz
標本化周波数	14.3MHz	13.5MHz	74.25MHz
全画素数	横910×縦525	横858×縦525	横2200×縦1125
有効画素数	横754×縦483	横720×縦483	横1920×縦1035

の有効画素数のうち、14.3MHzの項は、デジタルに関しては規格が無いため日本の電波法に準じて算出した値である。

量子化数については、要求される画質に応じて8～10bit位が選ばれる。最近は、10bitが好まれている。また、途中の処理は、12bitないし16bitで行われている^(*)。

デジタル映像信号処理機器には、ビデオスイッチャ、特殊効果装置、画質補償器、TBC (Time Base Corrector)、FS (Frame Synchronizer)、カラーコレクタ、ノイズリデューサ、TV方式変換装置など様々な機器がある^(*)が、これらを処理の観点からみると、表2.2に示す分類が考えられる。

表 2.2 映像信号処理の分類

(a) 処理の領域による分類

種類	処理内容	処理例
時空間	時空間領域の処理 (走査が行われる映像信号では空間領域の処理が含まれる)	濃度値変換、映像混合、色補正など
周波数	周波数領域の処理 (通常、時空間領域の畳み込み演算にして行う)	YC分離、輪郭補償、帯域制限、デフォーカス、ノイズリデューサ、FFTなど
座標	座標軸を変更する処理 (透視変換、曲面マッピングでは浮動小数点演算が要求される)	拡大・縮小、回転、透視変換、2次曲面へのマッピングなどの幾何学変換
準同期	書き込みと読み出しのクロックが異なる処理 (標本化周波数以下のクロック制御が必要である)	時間的ゆがみを補正するTBC、異種同期の映像信号を一致させるFSなど

表 2. 2 (つづき) 映像信号処理の分類

(b) 入力数による分類

種類	処理内容	処理例
単一 入力	1つの入力信号のみを使用する 処理 (同一画像内のライン間、 フレーム間処理も含まれる)	濃度縮変換、輪郭補償、YC分離、 幾何学変換、ノイズリデューサ、 色補正など
複数 入力	2つ以上の入力信号を同時に使 用する処理	映像合成 (10入力以上の場合も あり)、カラーマトリックス (R、G、B → Y、R-Y、B-Y)、ステレオ画像処理 など

(c) 次元による分類

種類	処理内容	処理例
0次元	それ自身のデータしか使用しない (基本的には、メモリが不要)	濃度縮変換、映像混合、コアリング 色補正など
1次元	水平方向のデータを使用する (数画素のメモリが必要)	BPFによるYC分離、水平輪郭 補償、コアリングによるノイズ除 去など
2次元	垂直方向のデータを使用する (数ラインのメモリが必要)	くし型フィルタ、垂直輪郭補償、 TBC、逐画型フィルタを使うデ フォーカスなど
3次元	時間軸方向 (フレーム間) の データを使用する (フレームメモリが必要)	幾何学変換、3次元YC分離、動き 検出、TV方式変換、ノイズリデ ューサなど

(d) 線形・非線形による分類

種類	処理内容	処理例
線形	線形演算要素による処理 (積和演算で表現できるもの)	利得調整、輪郭補償、コンポルー ション、フィルタなど
非線形	非線形演算要素による処理 (ROMまたはRAMによる $k \cdot f$ 、 f^2 、 f^{-2} が使える場合がある)	γ 補正、コアリング、白クリップ、 メディアンフィルタなど

表 2. 2 (つづき) 映像信号処理の分類

(c) データ依存度による分類

種類	処理内容	処理例
非依存	濃度値、位置に関係なく全画面で均一の処理をする	濃度値変換、輪郭補償、コンボリューション、拡大・縮小、デフォーカスなど
濃度値依存	濃度値に依存して処理を変える	適応型 Y C 分離、動き補償型 T V 方式変換器、動き補償型ノイズリデューサ、輪郭トラッキングなど
位置依存	画面上の位置に依存して処理を変える	シェーディング補正、レジストレーション補正、透視変換用の折り返し除去フィルタなど

以上の分類から、処理に必要な回路、例えばメモリの有無、人力数などについて知見が得られるが、以下では、幾何学変換、波形発生、映像合成を取り上げ、処理に必要な精度、機能、回路などについて詳しく検討する。

2. 2 幾何学変換

2. 2. 1 幾何学変換回路

従来からリモートセンシング画像など各種の画像処理において、物体の位置合わせ、レンズを含めた撮像系の幾何学歪みの補正、画像の重ね合わせなどの幾何学変換が行われてきた。一方、放送分野においても、番組制作の中で、映像表現のための特殊効果の一つとして、拡大縮小をはじめとする幾何学変換が頻繁に行われている。

この様な幾何学変換では、(a) 2次元空間上で標本化された各画素の位置を変更するアドレス変換処理、(b) 標本点の間隔が原画像に対して粗くなることによって生ずる折り返し歪を帯域制限して除去するための前置フィルタ処理、(c) 変換すべき位置に標本点が存在しないときに近傍の画素から内挿して求めるための補間フィルタ処理の主な3つの処理が必要である。幾何学変換には、拡大・縮小、平行移動などの線形な変換から、透視変換、2次以上の曲面へのマッピングなどの様々な変形があるが、これらを一画素毎に正確に計算すると、多くの時間を要しハードウェアの規模も大きくなる。実現性を重視すると、できるだけ小さいハードウェアで十分な画質を維持することが重要となる。

以下では幾何学変換回路の構成法について検討する。幾何学変換回路のうち、前置フィルタ処理に関しては他にも検討されている^{11) 12) 14)}ので、ここ

では、アドレス変換処理、補間処理について検討する。

2. 2. 2 アドレス変換処理

幾何学変換の中でも最も一般的で使用頻度の高いアフィン変換を取り上げ、アドレス演算のための回路構成と演算語長について検討する。

2次元のアフィン変換は、

$$x = A \cdot X + B \cdot Y + C \quad (2.1)$$

$$y = D \cdot X + E \cdot Y + F \quad (2.2)$$

で示すことができる。ここで、 (x, y) は変換前の座標、 (X, Y) は変換後の座標、 A, B, C, D, E, F は、変換パラメータである。この変換は、乗算器を用いて忠実に実現する方法もあるが、図2.1に示すように加算器のみで構成できる。すなわち、走査に合わせて、まず画面左上では演算値レジスタ $hacc1$ 、 $vaccl$ にそれぞれ初期値レジスタ $hint1$ 、 $vint1$ から C, F の初期値を与える。水平方向の走査に合わせ、 A が格納されている増分値レジスタ $hinc1$ 、 D が格納されている増分値レジスタ $vinc1$ を毎クロック加算する。1ラインの処理が終わったら、 Y を1増やし $X=0$ として、次のラインの処理を開始する。すなわち、 B が格納されている増分値レジスタ $hinc2$ 、 E が格納されている増分値レジスタ $vinc2$ を加える。しかし、 $X=0$ とするので、 B, E を単純に加算してはならない。このために演算値レジスタとして $hacc2$ 、 $vaccl$ を用意し、 $X=0$ のときの x, y の値を保存しておき、次のラインに移動した際に $hacc2$ 、 $vaccl$ からそれぞれ $hacc1$ 、 $vaccl$ に戻して、 B, E を加算する。この方法により、(2.1) (2.2) 式の計算が加算のみで実現できる¹¹⁾。

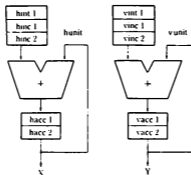


図2.1 アフィン変換回路

次に、拡大・縮小と回転に分け、演算語長について検討する。なお、アフィン変換には、この他に移動とせん断があるが、回路構成、演算語長については、拡大・縮小、回転と同様なので省略する。

2. 2. 3 拡大・縮小における演算語長

拡大・縮小は、(2. 1) (2. 2) 式において、 $B=C=D=F=0$ の場合であり、変換式は、

$$x=A \cdot X \quad (2. 3)$$

$$y=E \cdot Y \quad (2. 4)$$

となる。横 P 縦 Q 画素の画像が、横 p 縦 q 画素に縮小された場合は、変換パラメータ A, E は、

$$A=P/p \quad (2. 5)$$

$$E=Q/q \quad (2. 6)$$

である。このときに、図2. 2(a)に示すように縮小されて、しかもその位置が画面の端のとき、反対側の端がオーバーフローしないで計算可能な演算語長が整数部の所要ビット数である。これを満たす条件は、

$$2^m > A \cdot P \quad (2. 7)$$

$$2^n > E \cdot Q \quad (2. 8)$$

である。ここで、 m, n は、所要ビット数である。

(2. 5) ~ (2. 8) 式を解くと、

$$m > 2 \cdot \log_2(P) - \log_2(p) \quad (2. 9)$$

$$n > 2 \cdot \log_2(Q) - \log_2(q) \quad (2. 10)$$

となる。

525本/60Hzの標準TVを4倍のサブキャリアで標準化した場合(以下14.3MHzと略す)を考えると、有効画素数は、表2. 1によれば $P=754, Q=483$ である。 $p=1, q=1$ すなわち全面素を1点に縮小しても、オー

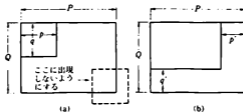


図2. 2 拡大・縮小

表 2.3 拡大・縮小に必要なビット数 (整数部)

(単位: ビット)

縮小 サイズ p, q 画素	標準テレビ				HDTV	
	14.3 MHz		13.5 MHz		74.25 MHz	
	m	n	m	n	m	n
1/4	21.1	19.8	21.0	19.8	23.8	22.0
1/2	20.1	18.8	20.0	18.8	22.8	21.0
1	19.1	17.8	19.0	17.8	21.8	20.0
2	18.1	16.8	18.0	16.8	20.8	19.0
4	17.1	15.8	17.0	15.8	19.8	18.0
8	16.1	14.8	16.0	14.8	18.8	17.0
10	15.8	14.5	15.7	14.5	18.5	16.7
20	14.8	13.5	14.7	13.5	17.5	15.7

注: 14.3 MHz のとき $P=754, Q=483$ 13.5 MHz のとき $P=720, Q=483$ 74.25 MHz のとき $P=1920, Q=1035$ としている。

表 2.4 拡大・縮小に必要なビット数 (小数部)

(単位: ビット)

縮小 サイズ p, q 画素	標準テレビ				HDTV	
	14.3 MHz		13.5 MHz		74.25 MHz	
	m	n	m	n	m	n
1/4	11.6	10.9	11.5	10.9	12.9	12.0
1/2	10.6	9.9	10.5	9.9	11.9	11.0
1	9.6	8.9	9.5	8.9	10.9	10.0
2	8.6	7.9	8.5	7.9	9.9	9.0
4	7.6	6.9	7.5	6.9	8.9	8.0
8	6.6	5.9	6.5	5.9	7.9	7.0
10	6.2	5.6	6.2	5.6	7.6	6.7
20	5.2	4.6	5.2	4.6	6.6	5.7

パフォーマンスが生じないためのビット数を求めると、 $m > 19.1$ 、 $n > 17.8$ となる。これが、整数部の所要ビット数である。その他の場合の所要ビット数を、C C I R 勧告 601 (以下 13.5 MHz と略す)、B T A 規格 H D T V (以下 74.25 MHz と略す)の標準化周波数の場合も含め、表 2.3 に示す。

一方、小数部の精度については、1倍に限りなく近い拡大・縮小ができるかどうかで決まる。図 2.2 (b) に示すように水平 p' 画素、垂直 q' 画素の縮小を想定すると、

$$1/2^{m'} < p' / P \quad (2.11)$$

$$1/2^{n'} < q' / Q \quad (2.12)$$

を満たす m' 、 n' が小数部に必要なビット数である。(2.11) (2.12) 式を変形すると、

$$m' > \log_2(P) - \log_2(p') \quad (2.13)$$

$$n' > \log_2(Q) - \log_2(q') \quad (2.14)$$

になる。 $P=754$ 、 $Q=483$ 、 $p'=1$ 、 $q'=1$ の場合は、 $m' > 9.6$ 、 $n' > 8.9$ 必要である。その他の場合の所要ビット数を表 2.4 に示す。

2.2.4 回転における演算経長

図 2.3 に示す回転は、

$$x = X \cdot \cos \theta - Y \cdot \sin \theta \quad (2.15)$$

$$y = X \cdot \sin \theta + Y \cdot \cos \theta \quad (2.16)$$

で示すことができる。ここで、 θ は回転角である。係数 $\cos \theta$ 、 $\sin \theta$ を表現するビット長が、回転の精度を決定する。

図 2.1 の回路構成を想定して、 $\sin \theta$ 、 $\cos \theta$ 値を表現するビット数に制限を加え、 $0 \sim 180^\circ$ の範囲で回転させて行ったシミュレーション結果が表 2.5 である。表 2.5 は、本来の位置 (理論値) からずれた距離の最大値を示す。これによれば、14.3 MHz の場合、蓄積誤差を 1 画素以内に抑える

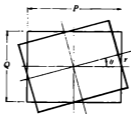


図 2.3 回転

表 2.5 回転におけるビット数と演算誤差

(単位: 画素)

ビット 数	標準テレビ		HDTV 74.25 MHz
	14.3 MHz	13.5 MHz	
1	620.3	600.6	1510.1
2	303.4	293.8	738.8
4	77.3	74.8	188.0
6	19.5	18.9	47.5
8	4.7	4.5	11.3
10	1.2	1.1	2.9
11	0.6	0.6	1.4
12	0.3	0.3	0.7
14	0.1	0.1	0.2
16	0.0	0.0	0.0

ためには、小数部に11ビット必要であることが判る。

2. 2. 5 アフィン変換における演算語長

以上を総合すると、アフィン変換の演算において必要な演算語長は、放送用途で最低限必要な1画素単位の精度を要求すると、14.3MHzの場合31(整数部20、小数部11)ビット、13.5MHzの場合30(整数部19、小数部11)ビット、74.25MHzの場合34(整数部22、小数部12)ビット必要である。しかし、これは小数点位置を固定した場合で、拡大率に応じて小数点の位置を変える方法をとれば、演算語長を小さくできる。14.3MHzを例に取り上げると、拡大率が大きい場合は、上記検討結果より整数部20ビットあれば十分で、この場合は小数部の精度はあまり必要でない。一方、拡大率が1に近い場合は、上記検討結果から小数部に11ビットあればよく、整数部は移動を考慮しても11ビットあればよい。従って、14.3MHzの場合は、1画素までの精度を要求しても計22ビットあれば十分である。

2. 2. 6 補間処理

前節で述べた(2.1)(2.2)式の幾何学変換で求めた座標(x,y)は、変換前の座標系において必ずしも格子点に一致しないため、図2.4に示す

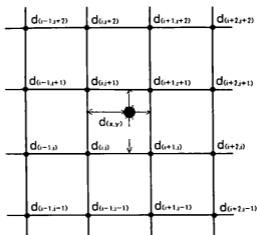


図 2.4 補間

近傍の格子点を使って補間する必要がある。補間には、以下に示すような各種の方式がある⁽¹⁶⁾⁽¹⁷⁾。

(a) 最近傍零次補間 (NEAR)

座標 (x, y) に距離的に最も近い格子点を割り当てる方法である。すなわち、座標 (x, y) の濃度値を $d(x, y)$ とすると、

$$d(x, y) = d(i, j) \\ i = [x+0.5], \quad j = [y+0.5] \quad (2.17)$$

とするものである。ここで、 $[]$ は括弧内の数値を超えない整数値を表すガウス記号である。

(b) 最大値零次補間 (MAX)

座標 (x, y) を囲む近傍 4 点の画素のうち最も信号レベルが高いものを補間データとする方法である。式で示すと、

$$d(x, y) = \text{MAX} [d(i, j), d(i+1, j), d(i, j+1), d(i+1, j+1)] \\ i = [x], \quad j = [y] \quad (2.18)$$

である。ここで、 $\text{MAX} []$ は括弧内の最大値のものをとる。

(c) 4 点線形補間 (4P)

近傍 4 点から線形に内挿して求める方法である。式で示すと、

$$d(x, y) = [f_1(y) \ f_2(y)] \begin{bmatrix} d(i, j) & d(i+1, j) \\ d(i, j+1) & d(i+1, j+1) \end{bmatrix} \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix}$$

$$i = [x], \quad j = [y]$$

$$f_1(t) = 1 - (t - [t]), \quad f_2(t) = t - [t], \quad f_3(t) = 1 - f_2(t) \quad (2. 19)$$

である。

(d) 9点2次補間 (9P)

近傍9点から2次多項式により補間を行うもので、ラグランジュの補間公式から補間係数を求める。式で示すと、

$$d(x, y) = [f_1(y) \ f_2(y) \ f_3(y)] \begin{bmatrix} d(i-1, j-1) & d(i, j-1) & d(i+1, j-1) \\ d(i-1, j) & d(i, j) & d(i+1, j) \\ d(i-1, j+1) & d(i, j+1) & d(i+1, j+1) \end{bmatrix} \begin{bmatrix} f_1(x) \\ f_2(x) \\ f_3(x) \end{bmatrix}$$

$$i = [x], \quad j = [y]$$

$$f_1(t) = (t - [t])(t - [t] - 1)/2$$

$$f_2(t) = 1 - (t - [t])^2$$

$$f_3(t) = (t - [t])(t - [t] + 1)/2 \quad (2. 20)$$

である。

(e) 16点標本化関数補間 (SINC)

標本化関数 $\sin(x)/x$ を用いると、周波数上で理想的な補間が行えるが、無限の標本点による畳み込みが必要になり現実的でない。そこで、近傍16点を用いた補間を考える。すなわち、

$$d(x, y) = [f(y_1) \ f(y_2) \ f(y_3) \ f(y_4)] \begin{bmatrix} d(i-1, j-1) & d(i, j-1) & d(i+1, j-1) & d(i+2, j-1) \\ d(i-1, j) & d(i, j) & d(i+1, j) & d(i+2, j) \\ d(i-1, j+1) & d(i, j+1) & d(i+1, j+1) & d(i+2, j+1) \\ d(i-1, j+2) & d(i, j+2) & d(i+1, j+2) & d(i+2, j+2) \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \end{bmatrix}$$

$$i = [x], \quad j = [y]$$

$$f(t) = (\sin \pi t) / \pi t$$

$$u = x - [x], \quad v = y - [y]$$

$$x_1 = 1+u, \quad x_2 = u, \quad x_3 = 1-u, \quad x_4 = 2-u$$

$$y_1 = 1+v, \quad y_2 = v, \quad y_3 = 1-v, \quad y_4 = 2-v \quad (2. 21)$$

を用いる。

(f) 3次スプライン補間 (SP)

コンピュータグラフィックス等では、B-スプライン補間がよく用いられているが、補間曲線が端点を除いて標本点を通らないため、今回の適用例のように4標本点内の中間の曲線を求める場合、標本点に近い部分での誤差が大きくなる。このため、3次スプライン関数を用いる。すなわち、点 t_1, t_2, t_3, t_4 ($t_1 < t_2 < t_3 < t_4$) に対して、関数値 $f(t_1), f(t_2), f(t_3), f(t_4)$ が与えられており、各区間 $[t_i, t_{i+1}]$ ($i=1, 2, 3, 4$) において3次の多項式で、1次導関数 $f'(t)$ および2次導関数 $f''(t)$ が全区間で連続で、 $S(t_i)=f(t_i)$ ($i=1, 2, 3, 4$) である関数 $S(t)$ が、3次スプライン関数である。ここで、格子間隔が一定 ($=1$) で、端条件として $f'(t_1), f'(t_4)$ を与えられた4標本点の一つの3次多項式を通る条件で、区間 $[t_2, t_3]$ での補間係数を求め、この係数を使って補間値を求める。水平方向4標本点から補間を求めた後、垂直方向に並ぶ4点の標本点から再度スプライン補間により最終結果の補間値を得る。

上記6種の補間フィルタの周波数振幅特性を、図2.5に、補間誤差を図2.6に示す。図2.5は、1次元信号を3倍に拡大する場合について計算したものである。最近傍、最大値補間の特性は零次ホールドとみなして求めた。スプライン補間は、入力信号の標本値によって補間係数に変化するので省略した。図2.6は、1次元の正弦波信号を拡大し各種補間法で補間した波形と、拡大した時に得られる周波数で作った理想的な正弦波との2乗誤差を、周波数ごとに計算したものである。具体的には、 $\cos \omega n$ ($n=1, 2, \dots$ 、

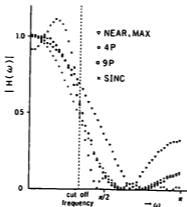


図2.5 補関関数の周波数振幅特性

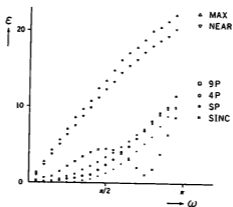


図 2.6 補間間数の補間誤差

512) の波形を時間軸で k 倍に拡大し補間して得られた信号を $s(n)$ とし、拡大率に応じて ω/k にした周波数の正弦波信号を $g(n)$ とすると、

$$\epsilon = \sqrt{\sum_{n=0}^{N-1} \{s(n) - g(n)\}^2} \quad (2.22)$$

を誤差の評価関数とした。図 2.6 は、 $k=3.5$ の場合の例である。その他の拡大率についてもほぼ同様な傾向の誤差量を持つ。

2. 2. 7 補間処理の画質評価

上述の補間法と画質との関係について調べるために主観評価実験を行った¹¹⁾。実験は、表 2.6 に示すように、各種幾何学変換における補間と画質の関係を調べる実験 1 と、各種拡大率における補間法と画質の関係を調べる実験 2 の 2 種類行った。この実験では、テレビジョンシステム評価用デジタル標準画像 (TV 学会)¹²⁾ のうち、細かい絵柄や直線の多い“チューリップ”と比較的平坦で曲線も含む“美人ボタン”の 2 種類を素材に、各種の幾何学変換を施した画像を用いて、主観評価実験を行った。評価画像は、横 $754 \times$ 縦 483 画素の画像で、ミニコンクラスのコンピュータ (MV4000) を用いて浮動小数点演算 (仮数部 24 ビット、指数部 8 ビット) で計算した。変換画像の中で縮小される部分があり、正確には折り返し歪防止用のプリフィルタが必要であるが、今回の実験が補間法を評価するためのもので、①他の要因による影響を排除するためと、②縮小率が大きくないため、

表 2.6 補間方式に関する主観評価実験

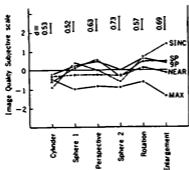
実験 1 各種幾何学変換における補間と画質の関係を調べる。

評価法	4点帰納を基準とした一対比較法 評価カテゴリ (7段階) 3:非常に良い 2:良い 1:やや良い 0:同じ -1:やや悪い -2:悪い -3:非常に悪い
評価画像	*「チューリップ」と「美人バタン」に対し、 回転 (画面中心で45°の回転)、 拡大 (横分比で1.5倍の拡大)、 円筒 (画面高の約1/2の直径と高さをもつ円柱にマッピング)、 パース (画面左部で約2倍、画面右部で1/2に拡大)、 球体1 (画面高の約1/2の直径の球体にマッピング)、 球体2 (画面高の約3/4の直径の球体にマッピング) の6種の幾何学変換を施した画像
評定者	放送技術者13名
視距離	4H (画面高の4倍)
表示条件	カラーモニタ (20インチ) 輝度:最大200nit 最小20nit モニタ背景:灰色30nit

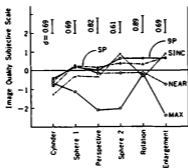
実験 2 各種拡大率における補間法と画質の関係を調べる。

評価法	一対比較法 評価カテゴリ (7段階) 3:非常に良い 2:良い 1:やや良い 0:同じ -1:やや悪い -2:悪い -3:非常に悪い
評価画像	*「チューリップ」と「美人バタン」に対し、 0.7倍、1.5倍、2.0倍、2.7倍、3.5倍の6種の拡大率で 拡大・縮小を施した画像
評定者	放送技術者13名
視距離	4H (画面高の4倍)
表示条件	カラーモニタ (20インチ) 輝度:最大200nit 最小20nit モニタ背景:灰色30nit

入れていない。実験結果を図2. 7, 図2. 8に示す。図2. 7は、各種幾何学変換における補間と画質の関係を示すもので、図2. 8は、各種拡大率における補間法と画質の関係を示すものである。(a)が“美人バタン”、(b)が“チューリップ”の場合である。縦軸は各評価画像が得た評価カテゴリの平均評点である。系列範囲法により尺度化しており、図の縦軸は画質評価心理尺度である。各幾何学処理における二つの補間法の間隔が、図中に示すdの距離より小さければ、その間には危険率5%で画質の有意差がないことを表す。

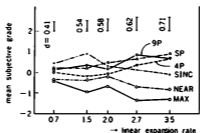


(a) "Girl"

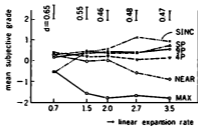


(b) "Tulips"

図2.7 主観評価実験1の結果



(a) "Girl"



(b) "Tulips"

図 2.8 主観評価実験 2 の結果

図 2.7, 図 2.8 によれば, 最近傍, 最大値補間は, 他の補間法に比べ悪い評価であった。これは, 図 2.6 において補間誤差が他の補間法に比べ大きいことも一致している。9点2次補間, スプライン補間法は全体的に良い結果を得ている。標本化間数補間は, 評価が良い時と悪い時がある。標本化間数補間の周波数特性は, 図 2.5 によれば高域が持ち上がっており, 他の補間法に比して鮮鋭に見えるため, 細かい画像のとき良い結果を与える。その反面, ノイズ成分も強調するため, 背景の平坦な部分でノイズ(曇じわのような妨害)が目立ち, 評価が悪くなる場合もあったものと考えられる。また, 4点線形補間は, 1.5倍および2倍拡大の“美人ボタン”と3.5倍拡大の“チューリップ”を除くと9点2次補間, スプライン補間との間に有意差はなく, ほぼ良い結果を得ている。

各種補間法のハードウェア化に際して回路規模に大きな影響を与える処理の複雑さを, ミニコンピュータ(MV4000)の計算時間で評価すると表 2.7 のようになった。表 2.7 は, 4点線形補間を1としたときの相対時

表 2.7 各種補間法の CPU 時間

最近傍補間	0.5
最大値補間	0.6
4点線形補間	1.0
9点2次補間	1.8
16点標本化補間	6.6
3次スプライン補間	8.2

(注) 4点線形補間を1とした相対値
拡大のためのアドレス演算も含む

間で表している。主観評価実験と演算量から評価すると、4点線形補間は、計算量が少ない(3次スプライン補間の約1/8、9点2次補間の約1/2)わりに画質が良く、ハードウェア化に適している。

2. 2. 8 補間処理の演算量

以下では、上記実験・検討結果に基づき、4点線形補間を取り上げ、必要な演算精度について検討する。(2.19)式を展開すると、

$$d(x,y) = [(1-q) \cdot \{ (1-p) \cdot d(i,j) + p \cdot d(i+1,j) \} + q \cdot \{ (1-p) \cdot d(i,j+1) + p \cdot d(i+1,j+1) \}] \quad (2.23)$$

となる。ここで、 $p=f_x(x)$ 、 $q=f_y(y)$ である。

以下では、この補間に必要なビット数について検討する。 p,q の演算量を短くした場合に倍精度の浮動小数点で補間した場合に比べ、誤差が大きくなるのは、近傍4点の値が表現できる濃度値の最大値(max)と最小値(min)になった場合である。この組合せは、

$$\textcircled{1} d(i,j) = d(i,j+1) = \max, d(i+1,j) = d(i+1,j+1) = \min \quad (2.24)$$

$$\textcircled{2} d(i,j) = d(i,j+1) = d(i+1,j) = \max, d(i+1,j+1) = \min \quad (2.25)$$

$$\textcircled{3} d(i,j) = \max, d(i,j+1) = d(i+1,j) = d(i+1,j+1) = \min \quad (2.26)$$

$$\textcircled{4} d(i,j) = d(i+1,j+1) = \max, d(i,j+1) = d(i+1,j) = \min \quad (2.27)$$

と、これらを 90° 、 180° 、 270° 回転させたボタンである。濃度レベル8ビットの場合 $\max=255$ 、 $\min=0$ で、その差は255量子化レベルとまではずである。しかし、NTSC信号の帯域は、14.3MHz標本化の場合の伝送帯域の7.15MHzよりも4.2MHzと低く、隔合う画素の濃度レベル差は255量子化レベルにならない。14.3MHz標本化の場合に

4. 2 MHz の理想ローパスフィルタを通して計算すると、最大誤差レベル差は3.8量子化レベルである。同様に計算すると、13.5 MHz の場合4.0量子化レベル、74.25 MHz の場合5.2量子化レベルである。それぞれの場合の誤差を計算したものが、表2.8である。表中の平均誤差は、 p, q を0~1の範囲で変化させた時に生ずる誤差の平均を示し、最大誤差は最も大きい誤差を示す。この誤差は、幾何学変換においてはブロック歪となり、非常に目立つ。これについては、Rico, Piper, Pieron の報告²⁰⁾があり、視覚的に敏感であることが知られている。また、視覚特性の検知限では、偽輪郭では3量子化レベル、パタン歪では1量子化レベルに誤差を抑える必要があることも報告されている²¹⁾。この条件を満たすためには、補間ビットとして平均誤差で5ビット、最大誤差で7ビット以上必要である。

ここで、ブロック歪について検討してみよう。横M倍、縦N倍の拡大率、nビット補間として、モザイク状にならない条件、すなわち同一の誤差レベルの画素が複数個生じない条件を求めると、

$$M < 2^n, N < 2^n \quad (2.28)$$

である。nについて求めると、すなわち、

$$n > \log_2(M), n > \log_2(N) \quad (2.29)$$

表2.8 補間ビット数と誤差

(単位: 量子化ビット)

補間 ビット 数	標準テレビ				HDTV	
	14.3 MHz		13.5 MHz		74.25 MHz	
	最大	平均	最大	平均	最大	平均
1	28.5	8.0	30.0	8.4	38.9	10.9
2	16.6	4.0	17.4	4.2	22.7	5.5
3	8.8	2.0	9.3	2.1	12.1	2.7
4	4.6	1.0	4.8	1.3	6.2	1.4
5	2.3	0.5	2.4	0.4	3.2	0.7
6	1.1	0.2	1.2	0.2	1.6	0.3
7	0.6	0.1	0.6	0.1	0.8	0.2
8	0.3	0.1	0.3	0.1	0.4	0.1
9	0.1	0.0	0.1	0.0	0.2	0.0
10	0.1	0.0	0.1	0.0	0.1	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0

となる。1画素を全面面に拡大した場合を想定すると、14.3MHzの場合、 $M=754, N=483$ であり、 n は10ビット以上必要である。しかし、これほどの拡大率の場合はブロック歪を除去したとしても、ボケが激しく実用に耐えうる画質ではない。筆者らの経験では、せいぜい1.6倍(面積で2.56倍)が限界である。1.6倍以下に限定すれば4ビットの精度でよく、上記の条件よりも緩いので問題ない。

以上を総合すると、1量子化レベルの誤差とするためには、平均誤差で5ビット、最大誤差で7ビット必要になる。

2. 3 波形発生

2. 3. 1 ビデオスイッチャ

ビデオスイッチャ¹²⁾は、番組制作に関わる主な映像信号処理機能を統合化した装置で、多数の素材に対して、ディゾルブ、カット、ワイプ、スーパー、DVE(Digital Video effect)などの各種の効果処理を行う。これらのなかで、シーンチェンジによく使用されるのが、カット、ディゾルブ、ワイプである。

ディゾルブは、入力映像を $A(i, j)$ 、 $B(i, j)$ 、出力映像を $M(i, j)$ とすると、

$$M(i, j) = k \cdot A(i, j) + (1-k) \cdot B(i, j) \quad (2. 30)$$

に示すように、混合比 k を、フレーム毎に少しずつ0から1に変化させることにより、映像がAからBに混合されながら切り替える処理である。これに対して、カットは、 k を0から1に1フレームで変化させることにより混合させることなく切り替える。

ワイプは、一方の画面が別の画面へ各種の図形パタンで、合成されながらその形が広がって取り切られて行く効果である。図2. 9に示すように、キー信号がディゾルブやカットのように全面面で均一でなく図形パタンとなっており、(2. 30)式の k が画面上の位置に依存して変化する

$$M(i, j) = k(i, j) \cdot A(i, j) + (1-k(i, j)) \cdot B(i, j) \quad (2. 31)$$

になっている。キー信号は、図2. 10(a)に示す波形が、(b)に示す特性を有するレベルスライス回路で2値化され、(c)に示す波形が発生される。キー信号には、境界部が急峻なハードキー(図2. 10中の実線)と、境界部が滑らかに変化するソフトキー(図2. 10中の点線)がある。

このワイプを行うための波形信号を作るのが効果波形発生回路であり、ワイプ以外にも幾何学変換を行う特殊効果装置などのマスキングにも使われている。

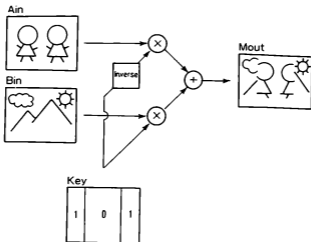


図 2.9 ワイブ

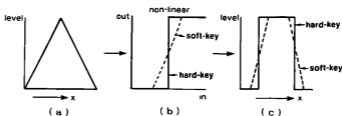


図 2.10 ワイブキー

2. 3. 2 効果波形発生回路

効果波形発生回路は、図 2. 11 に示す回路で構成できる。水平周期と垂直周期の三角波、のこぎり波などの基本波信号を元に、混合、スライスして、効果波形が生成される。図 2. 11 のスイッチ (Sw) を切り換えることにより、水平・垂直混合波形 (Sw1 側) 及び独立波形 (Sw2 側) を発生することができる。

さて、基本波の発生方法であるが、三角波を例に取り上げて回路構成と演算語長について検討する。他の波形でも大差は無い。図 2. 12 (a) は、水平周期の三角波である。水平掃線期間が終了したら、水平周期 M の半分で最

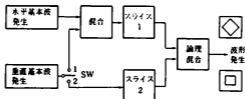


図 2.11 効果波形発生回路

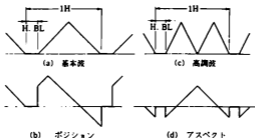


図 2.12 三角波

大値D になるように0 から加算を行いライン中央になったら減算を行う。この動作を毎ライン行うことにより水平周期の基本波が発生できる。これを式で示すと、

$$f(x) = \begin{cases} \frac{2D}{M}x & [0 \leq x < \frac{M}{2}] \\ D - \frac{2D}{M}(x - \frac{M}{2}) & [\frac{M}{2} \leq x < M] \end{cases} \quad (2.32)$$

になる。ここで、 x は水平アドレスであり、 M は水平方向の有効画素数である。垂直方向の場合は、 x を y に、 M を N (垂直方向の有効画素数) に置き換えればよい。

上記波形を位置移動させる場合には、2.12 (b) に示す波形が発生させる。画面上のどの位置からでも波形を大きくして全画面をカバーして切り替わるようにするために、波形のダイナミックレンジをすべてカバーできるようなレベルスライスを行う必要がある。このときの波形は、

$$f(x) = \begin{cases} \frac{2D}{M}(x - p) & [0 \leq x < \frac{M}{2} + p] \\ D - \frac{2D}{M}(x - p - \frac{M}{2}) & [\frac{M}{2} + p \leq x < M] \end{cases} \quad (2.33)$$

で示す式になる。ここで、 p は移動量で $-M/2 \leq p < M/2$ である。

図 2. 1 2 (c) の高調波は、増(減)分値を増やし波形を繰り返せばよい。 m 個の三角波がある場合を想定すると、

$$f(x) = \begin{cases} \frac{2mD}{M}(x - \frac{M}{m}i) & [\frac{M}{2m}i \leq x < \frac{M}{2m}(2i+1)] \\ D - \frac{2mD}{M}(x - \frac{M}{2m}(2i+1)) & [\frac{M}{2m}(2i+1) \leq x < \frac{M}{2m}(2i+2)] \end{cases} \quad (2.34)$$

に示す式になる。ここで、 $i=0, 1, \dots, m-1$ である。

図 2. 1 2 (d) のアスペクトは、水平波形と垂直波形のバイアスを変えれば得られる。式で示すと、

$$f(x) = \begin{cases} \frac{2D}{M}x + D_0 & [0 \leq x < \frac{M}{2}] \\ D - \frac{2D}{M}(x - \frac{M}{2}) + D_0 & [\frac{M}{2} \leq x < M] \end{cases} \quad (2.35)$$

になる。ここで、 D_0 は、バイアスである。

これらの (2.32) ~ (2.35) 式及びこれらの組み合わせを実行するには、図 2. 1 のアフィン変換回路にコンパレータを追加すればよい。hunit で $f(x)$ の計算を行い、vunit で x の条件計算を行う。コンパレータは、 x の条件比較を行うために使用する。(x を比較対象としないで、 $f(x)$ そのものを最大値、最小値と比較して行う方法も可能である。) この比較は、演算と同時に毎クロック行う必要があるため、演算器と兼用することはできない。また、水平同期に対しては演算をリセットする必要がある。このために、プログラマブルな演算回路においては条件比較と水平同期の同時チェックができる機能が必要である。

垂直周期の基本波形発生の場合は、クロック毎ではなく、水平同期毎に加減算を行う。1 走査線毎のワイプを実現するためには、第 1 フィールドと第 2 フィールドで初期値、最大値、最小値を変えなければならない。このためには、フレームパルスをチェックする機構が必要である。

ロークリー波形の場合は、基本波が、 x, y の 1 価関数 $f(x), g(y)$ ではなく、

x, y の 2 価関数 $f(x, y), g(x, y)$ になる。 $f(x, y)$ を例に取り上げると、ライン毎の初期値が y に依存して増加する。(2. 33) 式の p がライン毎に変化することに相当する。

これらの演算において画素単位でワイプを可能にするためには、基本波形のダイナミックレンジとして水平及び垂直の画素数分の精度が最低限必要となる。従って、14. 3 MHz、13. 5 MHz の場合 10 ビット、74. 25 MHz の場合 11 ビット必要である。

2. 4 映像合成

2. 4. 1 クロマキーとビデオマット

TV番組制作における映像合成技術は、番組演出上、不可欠なものとなっており、より適用範囲の広い自然な合成手法が求められている。従来の映像合成法には、クロマキー²⁴⁾やビデオマット²⁵⁾などがある。クロマキーは、色情報を用いた物体抽出法で動物体の抽出が実時間で行えるが、背景に青色などの特殊なスクリーンが必要であり、適用できる映像は限定される。一方、対象物の輪郭を直接タブレットなどを用いて手動で指定してキー信号を作成するビデオマットでは、幅広い映像に適用可能であるが、動物体に適用するためには全てのフレームでキー信号を作成する必要があり、作業時間がかかる欠点がある。現在のところ、背景に制約を受けず、対象物自身が移動・変形する場合や、カメラがパン・ズーム・ドリーする場合に実時間で対応できる合成方法がない。

一方、コンピュータビジョンにおいては、動物体の3次元運動を解析する研究が行われている²⁶⁾。動画像の符号化においても、ズームなどの大局的な動き補償を用いた予測符号化の研究が行われている²⁷⁾。これらは映像合成を目的としておらず、テレビ番組で要求される種々雑多なシーンに適用可能な一般的な実時間処理手法は確立されていない。

以下では、動ベクトルを用いた映像合成法を提案し、本合成法に適した動ベクトル検出法について述べることにより、後述のシステム化に必要な機能について検討する。

2. 4. 2 提案する映像合成法の概要

図2. 13の原画像のような自然風景の中を走る車の背景を別の風景に変える合成を考える。本映像合成法では次に示す手順でキー信号を作成し、合成を行う。

STEP1 最初のフレーム上で対象物(図2. 13では、車)の輪郭線をマウスやタブレットなどを用いて手動で指示し、キー信号を作る。

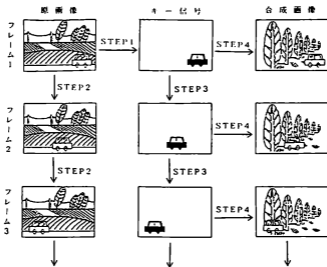


図2.13 提案した映像合成法

- STEP2 次のフレームとの間で、対象物の移動・変形などの動きを求める。
 STEP3 求めた動きに合わせて現フレームのキー信号を移動・変形する。
 STEP4 求めたキー信号を用いて対象物を抽出し、別の背景画像と合成する。
 STEP5 STEP2,3,4を繰り返し、連続した全てのフレームの合成を行う。

これらの手順のうち手動で行うのはSTEP1のみで、残りは全て自動で行われる。

この映像合成法で問題となるのは、対象物の動きの求め方である。対象物の動きを求めるためには、現フレームの対象物上の各点が次のフレームのどこに対応するかをそれぞれ検出する必要がある。しかし、対象物上の各点の動きが全く独立である場合は少なく、対象物が、(1)大きさ・形を変えず平行移動している、(2)回転・ズームを伴って移動していると仮定しても、TV番組制作ではかなり多くのシーンに当てはまる。以下では、(1)(2)に分けて検討する。

2.4.3 動ベクトル検出法

(1) 大きさ・形を変えず平行移動する場合

動物体が大きさ・形を変えず平行移動している場合、例えば物体がカメラ

の前を横切る場合や、カメラがパン・チルトする場合には、物体上の各点の動ベクトルは一樣である。

従来より動ベクトル検出法として、(1) フーリエ変換を用いる方法、(2) 相互相関関数を用いる方法、(3) 勾配法、(4) マッチング法などが提案されている^{1)~4)}が、実時間処理に向きかつ精度がよいとされるマッチング法を基本とし、動ベクトル検出を行うことにした。

マッチング法^{1)~4)}は、現フレーム S_n と前フレーム S_{n-1} 間で位置をずらしながら、類似度が最も高くなる位置を動ベクトルとする方法である。一般に、マッチング法では、画面を格子状に分割したブロック単位で検出を行い^{1)~4)}、

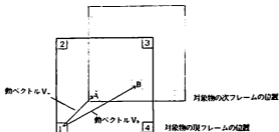
$$P_n(x, y) = \sum_i |S_n(x_i + x, y_i + y) - S_{n-1}(x_i, y_i)| \quad (2.36)$$

で示される評価関数 P_n が最小となる試行ベクトル (x, y) を動ベクトルとする。ここで、 (x_i, y_i) はブロック内の画素の位置、 Σ はブロック内の全ての画素あるいは均一に配置された検出点についての総和を表す。マッチング法では、(1) ブロック内に特徴的な絵柄が無く単調な部分を多く含む場合は、候補となる動ベクトルが複数存在する場合があるため一意に決められない、

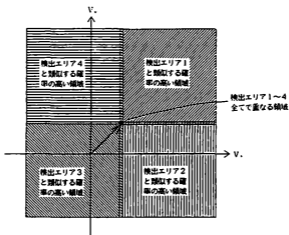
(2) 動物体と背景の両方を含む境界領域では、異なる動ベクトルをもつ代表画素の総和をとるために正確な検出ができないなどの問題がある。

そこで、本映像合成法では、抽出対象となる動物体(以下、対象物と呼ぶ)の形状が最初のフレームにおいて与えられていることを利用し、対象物上の画素をマッチングの対象領域とすることにした。さらに、演算量を減らすために対象物上の全点を用いてマッチングを行うのではなく、数ヶ所を検出点として選ぶようにした。検出点は対象物上のエッジ近辺の内側の離れた4エリア(それぞれ複数の画素を含む)に設定する。

これは、対象物上の内部では均一の明るさやテクスチャをもち、外部(背景)では異なる絵柄をもつ場合が多い点に注目したからである。図2.14(a)に示すように、4角形の対象物が動ベクトル V だけ移動した場合、現フレームのエリア1は次フィールドのエリアAと類似度が高く、(2.36)式の P_n は小さいはずである。マッチング法では探索範囲全てが候補領域であるが、対象物外部では異なる絵柄をもつ場合が多いという上記の仮定から、エリアBなどの対象物内部では類似度が高くなる(P_n が小さくなる)可能性が大きい。対象物内部であるために、エリア1の P_n が小である確率が高い、すなわち動ベクトルとして選択できる領域を示すと、図2.14(b)の 影 の部分となる。エリア2, 3, 4についても、対象物内部であるために、動ベクトルとして選択できる可能性が高い領域を示すと、図2.14(b)の 影 となる。背景に対象物内部よりも類似したエリアがあれば誤検出



(a) 4 検出エリア



(b) 動ベクトルの検索領域

図 2.14 4 検出エリアによる動ベクトル検出

の可能性もあるが、 \nearrow \searrow \swarrow \nwarrow 全てが重なる領域（4 エリアとも同一の動ベクトルの値をとる領域）をとれば、真の動ベクトルになる確率が高いのでこれを動ベクトルとして採用する。検出点として演算時間の兼ね合いで 4 エリアを選んだが、より多くのエリアを設定すれば、誤検出の可能性は減る。

このようにして求めた動ベクトルを用いて検出点を移動させ、次のフレーム間に対しても同様にして検出を行う。以降、全てのフレームに対し検出点を移動させながら検出を続ける。

(2) 拡大・縮小、回転を伴う場合

動物体が画面上で回転する場合や、カメラがズームする場合などでは、物

体上の各点の動ベクトルは一樣ではない。この場合は、動物体が画面上で拡大・縮小および回転などの2次元アフィン変換に基づいて移動・変形していると考えられる。すなわち、対象物上の点 (x_n, y_n) と、この点の次フレームでの対応点 (x_{n+1}, y_{n+1}) は2次元アフィン変換を用いて、

$$x_{n+1} = a \cdot x_n + b \cdot y_n + c \quad (2.37)$$

$$y_{n+1} = d \cdot x_n + e \cdot y_n + f \quad (2.38)$$

で表すことができる。ここで、 a, b, c, d, e, f は、変換パラメータである。また、点 (x_n, y_n) の対応点 (x_{n+1}, y_{n+1}) は、 (x_n, y_n) の周辺の複数画素を使い、(2.36)式のマッチング法を用いて求める。3個の特異でない (x_n, y_n) と (x_{n+1}, y_{n+1}) の対応(動ベクトル)が求まれば、6個の変換パラメータ a, b, c, d, e, f を一意に決定できるが、検出精度を上げるために、多数の点を用いて最小二乗法¹³⁾で求める。すなわち、 k 個の対応点群 $(x_n(i), y_n(i)) - (x_{n+1}(i), y_{n+1}(i))$ ($i=1, 2, \dots, k$)に対して、

$$E_x = \sum_{i=1}^k [(x_{n+1}(i) - a \cdot x_n(i) - b \cdot y_n(i) - c)^2] \quad (2.39)$$

$$E_y = \sum_{i=1}^k [(y_{n+1}(i) - d \cdot x_n(i) - e \cdot y_n(i) - f)^2] \quad (2.40)$$

が、最小となる a, b, c, d, e, f を求める。ここで、 Σ は、全ての対応点についての総和である。

このようにして求めた変換行列によって各検出点を移動させ、移動した点を新たな検出点として、次のフレーム間に対して同様に検出を行う。以降、全てのフレームに対し検出点を移動させながら検出を続ける。

2. 4. 4 シミュレーション実験

(1) 大きさ・形を変えず平行移動する場合

静止画像の上に、静止物体をフレーム毎に位置を変えて合成しノイズを加えて作成した動ベクトルが既知の疑似的な動画像(20フレームの白黒画像)を用い、動画像シミュレータ¹³⁾上で実験を行った¹³⁾。実験では、対象物の形状、明るさ、絵柄、動き、背景S/N、検出点の数、位置などのパラメータを変えて行った。実験結果の1例を図2.15に示す。図2.15は、6種類のテクスチャを有する対象物に対して、動き検出実験を行った結果で、検出点の数をパラメータにして、対象物の初期フレームからの移動量の理論値 (x_n, y_n) と検出値 (x_e, y_e) の累積移動量誤差 $\sqrt{(x_n - x_e)^2 + (y_n - y_e)^2}$ の最大値を示す。図2.15によれば、検出点を対象物上のエッジ近辺の離れた4エリアに設定した場合、各エリア15画素以上使用すれば、物体の明るさ、絵柄に影響されず、±2画素未満、すなわち、上下左右(以上誤差1画素)、斜め(誤差 $\sqrt{2}$ 画素)の8近傍点内の誤差で動き検出が行えることがわかる。

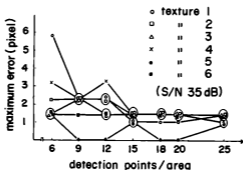


図2.15 各検出エリア当りの画素数と検出誤差

次に、実際に撮影して得た動画像に対し、映像合成実験を行った。実験は、固定したカメラの前を動物体が横切る画像（動画像1）、カメラをパンさせて撮影した画像（動画像2）、動物体が画面中央に位置するように追跡して撮影した画像（動画像3）の各々10フレーム（20フィールド）程度連続するカラー動画像に対して行った。使用したカラー動画像は、コンポーネント信号（R, G, B）で、動ベクトル検出の評価関数として、(2.36)式をコンポーネント・カラー画像に対応させた

$$\begin{aligned}
 P_r(x, y) = & \sum \{ |R_r(x, x, y, y) - R_{n-1}(x, y, y) | \\
 & + |G_r(x, x, y, y) - G_{n-1}(x, y, y) | \\
 & + |B_r(x, x, y, y) - B_{n-1}(x, y, y) | \} \quad (2.41)
 \end{aligned}$$

を用いて動ベクトル検出を行った。ここで、R, G, BはRGB各コンポーネントの映像信号レベルを表す。図2.16は1例で、(a)は初期フィールドに輪郭線を与えた画像、(b)は20フィールド後の合成画像である。なお、合成は、合成の境界部でちらつきが見えなくなるように、2値のハードキーを5×5のウィンドウで平滑化して多値にしたソフトキーを用いて行った。図2.16(a)で輪郭線上の○で示した部分が検出エリアである。図2.17は、図2.15と同様に検出エリア当りの画素数をパラメータとした場合の動ベクトルの検出結果である。最大誤差の計算は、図2.15の場合の理論値(x, y)の代わりに、輪郭線近辺の全画素を用いて行った検出値(x, y)を基準値として行った。すなわち、それぞれの画素数で検出した検出値(x, y)との累積移動量誤差 $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ の最大値である。この(x, y)は、筆者が目視で検出した動きとも一致している。動ベクトル検出の評価



(a) 撮影画像+輪郭線+検出点 (○印)



(b) 合成画像

図2.16 映像合成実験

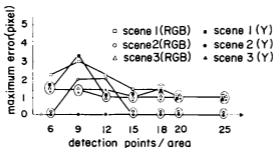


図2.17 各検出エリア当りの画素数と検出誤差 (撮影画像の場合)

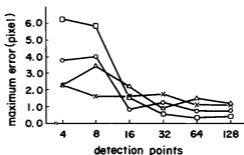
関数として、輝度信号 Y を用いた

$$P_i(x, y) = \sum_j |Y_i(x_i, y_i, t_j) - Y_j(x_j, y_j)| \quad (2.42)$$

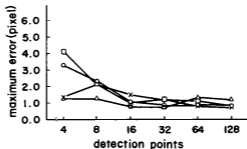
でも動ベクトル検出を行ったが、図2.17に示したように、(2.41)式の場合に比べ大きな差は見られなかった。従って、(2.42)式を用いた方が演算量の面で有利である。実験の結果、検出点を Y 信号において動物体上の輪郭線近辺のできる限り離れた4エリア(各エリア15画素以上)に設定すれば、撮影画像においても、 ± 2 画素未満の誤差で検出でき、合成処理を行った場合にも問題ないことが明らかになった。

(2) 拡大・縮小、回転を伴う場合

(1)と同様に、動ベクトルが既知の動画(20フレームの白黒画像)



(a) 拡大の場合



(b) 回転の場合

図2.18 検出エリアの数と検出誤差

を作成して実験を行った。検出点は、動物体内にてできるだけ大きい四角形を指定して、この四角形の2本の対角線上に等間隔に配置した。これは、全ての検出点を手動で与えるとする作業量が多いので、できるだけ自動的に行う部分を増やすための方策である。各検出点（エリア）当り、近傍16×16画素を用いて動ベクトル検出を行った。実験結果の1例を図2-18に示す。図2-18は、検出エリアの数をパラメータとして、初期フレームで設定した四角形の各頂点が各フレームで対応する位置を、求めた変換行列で算出した値（実験値）と真の位置（理論値）とで比較したものである。最大誤差とは実験値と理論値との距離の20フレーム中の最大値である。各種画像に対して、回転・ズーム量、検出エリアの位置、検出エリアの数など各種のパラメータを変えて行った実験の結果、対象物上のできる限り大きな四角形を設定し、この四角形の対角線上に32点以上の検出点を設定すれば、±2画素未満の誤差に収まることがわかった。

次に、実際に撮影して得た動画像に対し、映像合成実験を行った。実験は、カメラをズームしてボスタを撮影した画像、メトロノームの棒の先に小物体



(a) 撮影画像+検出点



(b) 合成画像

図2.19 映像合成実験（拡大・縮小、回転を伴う場合）

を貼り付けて動かし、これを静止したカメラで撮影した画像などで行った。図2. 19が、実験に用いた画像の1例である。図2. 19(a)は、ポストの前に置かれたメトロノームの先に取り付けた小物体が動いている画像で、検出エリアを重ねて表示したものである。(b)は別の背景と合成した20フィールド後の画像である。手動で全てのフレームについて切り出した場合に比べ±1画素程度のずれがあるが、動きげけにより、さほど気にならない。撮影画像に対しても、作成画像と同程度の誤差で動物体を抽出できることが明らかになった。

2. 4. 5 3次元運動する剛体への適用法

本節では、これまでの条件を拡張し、

- ・車などのように物体自体は形を変えない剛体で、
- ・滑らかに(突然向きを変えたりしない)移動し、
- ・物体あるいはカメラの3次元運動によって変形して見える動きで、
- ・フレーム間であまり激しく形を変えない

場合(これらの仮定は、特殊なものではなく多くの動画像に当てはまる)を対象とし、

『対象領域を2次元アフィン変換が成り立つと仮定できる小領域に分割し、

各領域ごとに変換行列を算出して動きを求める』

という方法を検討する。

回転・ズームを伴う場合では、多数の検出点(エリア)で動ベクトルを求め最小二乗法を用いることによって、検出誤差の影響を減らすことができたが、対象領域をいくつかの小領域に分割すると、多数の検出点を設定することが困難になり、十分な検出精度を保つことができない。さらに、複数のフレームにわたる場合は、注目点の各フレームでの正確な対応点が求められないと、フレーム間では小さな誤差でも蓄積されて、最終的には真の対応点からはずれてしまうことがあり、うまく行かない。

そこで、初期フレームにおいて注目点(始点)を手動で設定していたこれまでの補助情報に加え、注目点(始点)の最終フレーム上の位置(終点)も与えることにした。これは、カットの最初と最後のフレームのみ与えればよいので、さほど妥当性を欠いてはいない。

注目点の動きは、図2. 20に示すように始点から終点までの様々な軌跡で表すことができる。この軌跡の中から、最適な軌跡を求めればよい。最適な軌跡を求める問題は、図2. 21の示す多段ネットワークを用いて表現することができる。節点は画面上の位置を表し、枝は動きを示す。始点から終点までの経路が、軌跡となる。最適な軌跡を求めることは、ネットワークの

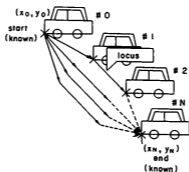


図 2.20 軌跡

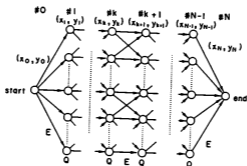


図 2.21 多段ネットワーク

最短経路を求めることと等価である。この問題は、動的計画法¹⁾を用いると、高速に解くことができる。

最短経路を求める評価関数として、初期フレームとの近似度と最終フレームとの近似度を表すの評価関数 Q 、(各節点の重みに相当)と、物体の動きの滑らかさを表す評価関数 E 、(各枝の重みに相当)を使用する。 Q は、図 2.22 に示すように、

$$Q_k(x, y) = (1-v) \cdot P1_k + v \cdot P2_k \quad (v=k/N)$$

$$P1_k = \sum_j |S_k(x_0+j, y_0+i) - S_k(x, j, y, i)|$$

$$P2_k = \sum_j |S_k(x, j, y, i) - S_k(x_0+j, y_0+i)| \quad (2.43)$$

を用いた。P1_k は初期フレームと現フレームの近似度を、P2_k は現フレームと最終フレームの近似度を、S_k(x, y) は第k フレームにおける位置(x, y)の映像信号レベル、 $\sum \sum$ は注目点近傍の全画素に対する総和を、Nはフレーム間の数(全フレーム数はN+1である)を示す。Q_kは、初期フレームと最終フレームに類似しているほど、小さな値となる。E_kは、図2.23に示すように、

$$E_k(\alpha, d) = 1 - \exp\left[-\frac{1}{2}\left(\frac{\alpha^2}{\theta^2} + \frac{d^2}{L^2}\right)\right] \quad (2.44)$$

を用いた。αは直進を0とした回転角、dは移動量、θは回転角の重み係数、

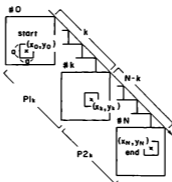


図2.22 評価関数Q_k

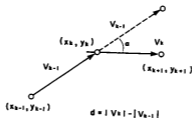


図2.23 評価関数E_k

L は移動量の重み係数である。E は、動き量が少なく直進するほど小さな値になる。

この評価関数 Q と、重み係数 ϵ を掛けた E を加算した目標関数

$$\sum_{t=0}^{N-1} [Q_t \cdot (X_t, Y_t) + \epsilon \cdot E_t(a, d)] \quad (2.45)$$

が最小になるパスを求めると、これが最適軌跡である。

本検出法の妥当性を評価するために、本制約条件に適合する撮影画像（レールを走るおもちゃの汽車を撮影した映像、交差点をまがる車をビルの屋上から撮影した映像）に対して動き検出実験を行った。実験結果の 1 例を図 2.24 に示す。図 2.24 (a) は初期フレーム、(b) は最終フレームのそれぞれで対応する注目点 (+ 印) を指定したものである。(c) は、本稿で提案した方法を適用した結果で、(d) は、従来のマッチング法を適用した結果である。実験では、 ϵ は点の重み Q の最大値と最小値の差、 $\theta = 60^\circ$ 、 $L = 6$ [画素] とした。従来のマッチング法では、注目点が指定した場所から外れ

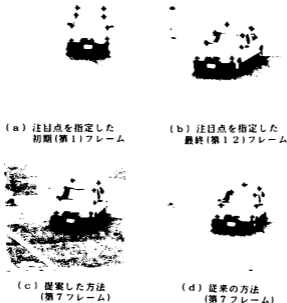


図 2.24 検出結果

てしまう場合が多いのに対し、本手法では、注目点が指定した場所から大きく外れることなく追跡できた。本手法の誤差は、筆者が目視で検出したものに比べ、全てのフレームにおいて約2画素の誤差に収まった。

[3] 映像信号処理システム

のアーキテクチャ

3.1 基本概念

映像信号（動画）処理の特徴は、膨大な情報量の画像を扱うことにある。例えば、NTSC画像の場合、毎秒数10Mbyteものデータを処理しなければならない。これを、汎用のコンピュータで処理することは困難である。その理由は、ノイマン型を採用しているために、①制御が逐次的である、②CPUとメモリのデータ転送幅が狭いなど、大量の画像データを扱うのに適した構造をしていないためである。この点に関しては、高速に並列演算が行えるスーパーコンピュータにしても同じである。しかし、映像信号処理では、全画面に同じ処理を施す場合が多い、近傍のデータを参照する場合が多いなど一般のデータ処理と異なる特徴があり、映像信号処理の特質を生かしたシステム構成とすれば、高速化が可能となる。本章では、映像信号処理のためのシステム構成について検討を行う。

システム構成法を検討するに当たり、画像・映像信号処理に必要な機能について考察する。

(1) 幅広い応用分野

放送、通信、工業、医学用など広範な動画処理、映像信号処理に適用できるようにする。これらの応用では、次に示すような処理がしばしば行われている。

- ・点演算（濃度値変換、閾値処理など）
- ・近傍演算（空間フィルタ、輪郭抽出など）
- ・幾何学変換（拡大縮小、回転、テクスチャマッピングなど）
- ・色処理（色補正、カラーマトリックスなど）
- ・フレーム間処理（動ベクトル検出など）
- ・画像間処理（画像合成、パタン・マッチングなど）

(2) ビデオレート処理

2.1で述べたように、NTSCに対しては、14.3MHzあるいは、13.5MHz、HDTVに対しては、74.25MHzの速度の画素を漏れなく処理することが要求される。ビデオレート処理とは、1フレームの処理が1フレーム期間（1/30秒）内で終了することであり、さらに次々と入力される画像に対して途切れなく処理を施して出力するものであり、処理時間の関係から入力や出力の流れを止めるといったことは許されない。

(3) 複数系統のカラー動画の処理

放送用では、複数系統のカラー動画を同時に処理することが求められて

いる。信号形式は、コンポジット信号（NTSC、PAL、SECAM）あるいは、コンポーネント信号（[R,G,B]、[Y,R-Y,B-Y]、[Y,I,Q]）である。

（４）16ビット処理

データ幅8～10ビットの入力画像に対して、処理ステップ数を重ねた場合にも十分な演算精度を保つ必要がある。一方、高速処理のためには、少ない演算語長の方が好ましい。ハードウェア規模も考慮して、通常16ビットで処理している。

（５）プログラム制御

システムの大半の機能がプログラムで制御される方が好ましい。これにより、従来の専用映像機器と異なり、同一のハードウェアで日々変わる番組制作に対して機能を変更したり、新しい効果を作り出すことができる。さらに、これらのことが、工場の開発・設計技術者によってではなく、番組制作スタジオの技術者によって、現場で行うことができる。

（６）拡張性・運用性

システムの拡張が容易で、新たな機能要求にも柔軟に対応できる構成であること。簡単な処理には、小規模なシステム構成が可能で、複雑な処理には新規にハードウェアを開発するのではなく、小規模システムと同じ基板を使用し、ただ数を増やすだけで対応できることが望ましい。これにより、予備基板の種類が少なくて済み、保守、運用性の向上がはかれる。

3. 2 並列処理方式

画像処理は、2次元状のデータ構造や、そのデータ量の膨大さのために、並列処理技術を用いた画像処理専用のアーキテクチャが提案されている。大別すると、パイプライン方式、完全並列方式、局所並列方式、マルチプロセッサ方式になる。

（１）パイプライン方式

図3. 1に示すように、同じ時間で処理が完了する異なる機能を有する演算器を多段に接続し、各演算器で処理を行いながら、順に前から後へパイプラインのように送って処理を重ねる方式。この方式を採用しているものに、Cytocomputer⁴¹⁾、彗駄天⁴²⁾、TIP⁴³⁾、FLIP⁴⁴⁾などがある。

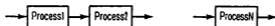


図3. 1 パイプライン方式

(2) 完全並列方式

図3. 2に示すように同じ演算要素(多くの場合2進演算器)を画像の構造に合わせて多数配置して全画素を同時に処理を行う方式。この方式をとるものに、MPP¹⁾、CLIP²⁾、AAP³⁾などがある。

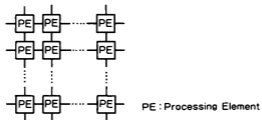


図3. 2 完全並列方式

(3) 局所並列方式

図3. 3に示すように、局所的アレイ(例えば3×3)を全画面あるいは処理対象領域に走査して処理する方式。この方式を採用するものには、TOSPIX¹⁾、ISP²⁾、RISP³⁾などがある。

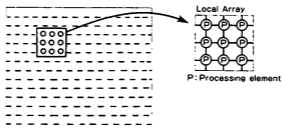


図3. 3 局所並列方式

(4) マルチプロセッサ方式

各々のプロセッサが各々のプログラムにより動作する方式。多くの場合、完全独立には動作させず、同期をとりながら処理させる。この方式をとるものに、MACSYM¹⁾、ZMOB²⁾、SIPS³⁾、RAPID⁴⁾、VSP⁵⁾などがある。

(1)は、固定のパイプライン構造のため柔軟性に欠ける欠点がある。章

駄夫⁴⁴⁾では、処理モジュールの接続を変えられるネットワークを採用し、パイプライン構造を可変にできるようにしているが、処理形態はパイプラインのままであり、用意した処理モジュールの機能以外は実現できない。負荷の大きいまとまった処理を複数のプロセッサに分散して行うような用途には向いていない。また、(2)は、画素数に相当する多くのプロセッサ数が必要となり、かつプロセッサ間の接続が固定的で離れたプロセッサ間の通信に時間がかかり、制御もSIMD (Single-Instruction Multiple-Data) のために柔軟性に欠ける。(3)は、画像処理で多用される近傍画素演算 (空間積和演算) に適しており、小規模のハードウェアで構成できるため、商用の画像処理装置に多くみられる方式であるが、幾何学変換のように近傍画素以外のデータを扱う大局的な処理には、向いていない。

3. 1で述べた機能を実現するためには、制御方式は多少複雑となるが、処理に応じて接続および処理形態を自由にえられる柔軟性に優れたマルチプロセッサ方式の方が望ましい。

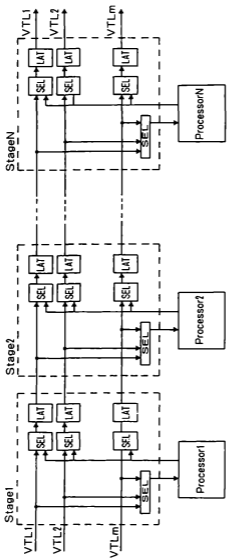
一口に、マルチプロセッサ方式といっても、プロセッサ間の結合形態の違いから、様々な方式がある⁴⁵⁾⁴⁶⁾。1つの画像を細かなブロックに分割しブロック毎にプロセッサを割り当て、これを並列に動作させることがVSP⁴⁷⁾などでは、行われている。この場合は、直並列変換を行うことにより、少ない本数のバスでプロセッサ間接続が実現できる。しかし、複数の映像信号間や色信号間の処理を行う場合には、複数の映像信号を通すための多数のバスが必要である。また、処理によっては、処理途中の中間映像のためにバスが必要である。一方、多くのフレームに渡る処理では、プロセッサを多段に接続する必要がある。バスの本数が、プロセッサの処理能力以上に、システムの処理能力を制約しかねない。このため、プロセッサ間の接続は、自由であることが望まれる。しかし、プロセッサ間をすべて結ぶとなると、プロセッサ数 n に対して、接続数 L は、

$$L = C_n = n(n-1)/2 \quad (3.1)$$

のように n^2 のオーダーで増大し、あまりに多くの線が必要になり、製作上問題である。

そこで、次に示す2つの構成法を考案した。

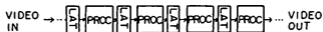
Configuration A 図3.4に示すように、プロセッサは、複数の単方向性の映像転送路によって結合される。映像転送路上では、映像データは、走査された映像信号列となっている。各プロセッサは、映像転送路から必要なデータが自分のプロセッサに流れて来たタイミングに信号列から選択して入力し処理する。処理結果は、映像信号列となるタイミングに映像転送路に出



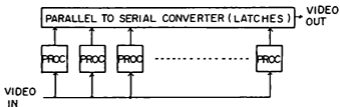
VTL: Video Transfer Line, SEL: Selector, LAT: Latch
(映像転送路)

図 3. 4 構成法 Configuration A

力する。手前のプロセッサから来たデータをそのまま通過させ、次のプロセッサで処理させることもできる。映像転送路には、プロセッサ毎にラッチが挿入されており、映像信号の標本化周波数のクロックで動作している。セレクトとラッチの動作により、図3. 5に示すようにパイプライン・モードにも、パラレル・モードにも適応できる。この方法によれば、転送路への出力は映像信号列となるように決められているので、バスアクセス競合もなく、入出力もスムーズに行える。このため、プロセッサの台数の増大に比例した処理能力の向上を望める。また、映像転送路は、ラッチによって規制されているため、通常バス方式で生じる線路長の増大によるプロパゲーション・ディレイの問題がない。物理的には、プロセッサの接続台数に制限がなく拡張が容易である。さらに、映像転送路では、データは映像信号列となっているために、映像転送路の間にD/A変換器を接続することにより、処理途中の状態をピクチャモニタで直接見ることができる。これにより、操作性、保守性を向上させることができる。この構成法は、4章で述べる実験システムRTVPで採用した。



(a) パイプライン・モード



(b) パラレル・モード

図3. 5 動作モード

Configuration B 図3. 6に示すように、16個のプロセッサをひとまとまり(クラスタ)にして、この中では完全自由に接続できるようにし、このクラスタを多段に接続される。クラスタ間は、16本の単方向性のバスで接続した。

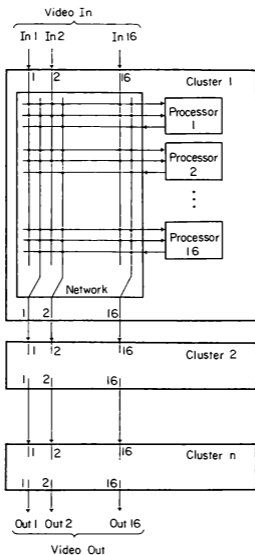


图 3. 6 構成法 Configuration B

ネットワークは、32入力（プロセッサからの出力16とクラスタ外からの入力16）48出力（プロセッサへの入力32とクラスタ外への出力16）のクロスバー・マトリックスであり、入力と出力の任意の接続が可能である。図3.7に示すように、1つの入力を同時に複数の出力に分配すること（ブロードキャスト・モード）や、複数の入力を時分割に1つの出力に集中すること（コンセントレイト・モード）が可能である。これにより、プロセッサをパイプライン状に接続することや、アレイ状に接続することも、さらに両者を組み合わせた形態をとることもできる。クラスタをまたがって、複数の映像入力を並行して供給したり、処理結果の映像や処理途中の映像も出力でき、処理能力をあまり犠牲にすることなく、パイプライン処理やパラレル処理を実現することができる。また、クラスタ内では、フィードバックループを構成することも可能で、巡回型の処理も容易に行える。この構成法は、5章で述べる実用システムPicot-systemで採用した。

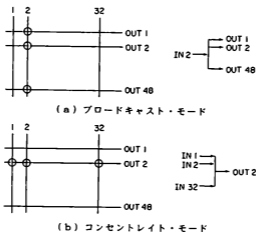


図3.7 接続モード

3.3 メモリ構成法

マルチプロセッサシステムにおいては、メモリの配置の方法によって、処理能力は大きく変化する。メモリ配置の方法を大別すると、

- (1) 共有方式
- (2) 分散方式

になる。

メモリを1ヶ所に集中させ、これを複数のプロセッサで共有する共有方式では、複数のプロセッサが同時にメモリにアクセスすると、メモリアクセス競合が生じて、処理能力が低下する。このため、メモリを分割し複数バスやクロスバー接続にして、同一メモリに同時にアクセスする確率を下げたり、バスの伝送容量を増やしたり、アービタを設けたりして、競合を緩和する工夫がされている。しかし、画像処理のように、メモリが大容量でかつ頻りにアクセスされる処理では、必要な接続線は膨大なものになる。クロスバー・スイッチの場合の規模は、データ幅 d ビット、アドレス幅 a ビット、プロセッサ数 p 、メモリ数 m とすると、接続線数はプロセッサとメモリを分離して配置した場合、 $(d+a) \times p \times m$ となり、大規模になる。

これに対して、各プロセッサにローカルメモリをもたせてデータを分散して格納する分散方式は、メモリアクセスによる競合は生じない。しかし、自分のメモリにないデータが必要になった場合、必要なデータをもつプロセッサと通信を行わなければならない。この通信が、あまり頻りに行われると速度低下を招く。画像を固定的にブロックに分割してメモリ分散させてしまうような方式をとると、幾何学変換に際し、プロセッサ間で通信が頻りに起こることになり、効率が下がる。

このため、分散方式を基本とするが、プロセッサが処理に必要とするデータは、各プロセッサのメモリに全て確保しておく方式とした。必要なデータの配置は、処理の内容によって前もってわかっているため、処理に先立ち自分のメモリに格納しておく。処理のやり方によっては、プロセッサが必要とするデータが複数のプロセッサで同じであることが起こりうるが、その場合は、別々のプロセッサに同じ内容のデータが確保されることになる。一見無駄のようであるが、メモリアクセス競合や通信手続きが不要のため、プロセッサ数に比例した処理能力向上が望める。メモリの容量は、幾何学変換を行うのに最低必要な1フレームあるいは1フィールド（インクレス走査している映像信号では、1フレームの半分の容量）とすることにした。

この考え方は、離散系シミュレータ KDDS-1⁽⁶⁾ のマルチリードメモリや行列計算用のシステム⁽⁶⁾ のブロードキャストメモリに似ている。書き込みに比べ読み出しの頻度が多い場合に有効であり、前述のように近傍演算の多い映像信号処理の特質に類似している。しかし、マルチリードメモリ方式やブロードキャストメモリ方式と異なり、メモリの書き込みに際してその都度コピー動作を行うことはしない。処理は出力画像を基本に行うために、自分と同じ出力対象領域を処理しているプロセッサは他にない。このため、処理結果をほかのプロセッサに伝えなくても、同一のフレームあるいはフィ

ールド内の処理に限れば問題は生じない。プロセッサの処理がフレームあるいはフィールド単位で完了した後、処理を行ったプロセッサから書き換えられた部分のデータを集めて、矛盾の無い完成したデータとする。

このメモリ構成法は、実験システム R T V P、実用システム P i c o t e r s y s t e m の両者で採用した。

3. 4 同期方式

多数のプロセッサを使用するマルチプロセッサシステムにおいては、処理過程でさまざまな遅れが生ずるので、プロセッサ間の同期の問題について考慮する必要がある。例えば複数のプロセッサを図 3. 8 に示す系統で接続して処理を行った場合、以下の解決すべき問題が生ずる。

Problem 1 バス A、B、C 間で位相にずれがあるため、これを補正する必要がある。

Problem 2 同じシステム同期信号で全てのプロセッサをプログラム制御すると、多数のプロセッサを通った場合、映像のみが遅れ同期と合致しなくなる。このため、画面の表示部分が欠落し、有効画素数が不足することがある。

Problem 1 のためには、各プロセッサに遅延量補償要素を設ければ良い。後述の P i c o t e r では、各プロセッサに最大 1 ラインの容量を持つ可変遅延をもたせている。 **Problem 2** のためには、次の 2 つの対策を講じた。

Measure A 処理に応じて後段のプロセッサほど遅れたシステム同期信号を供給する。システム同期信号は、処理に応じてプロセッサ毎に変えるのが望ましい。しかし、これはハードウェア製作上も制御上も大きな困難を生じるので、個々のプロセッサではなく、ある程度まとまった数のプロセッサ（後述の P i c o t e r s y s t e m では、16 台）で共通にする。このため、

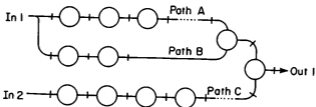


図 3. 8 バスの違い

プロセッサによっては、映像とシステム同期が完全には一致しない場合が生じ、有効画面が欠落することがある。これを避けるために、水平同期信号の幅を本来の同期信号規格よりも狭くする。

Measure B 処理される映像に同期情報を付けて、この同期情報を使ってプロセッサを制御する。この同期情報は、プロセッサのプログラム制御回路に入力し、これを用いてプロセッサを動作させることにより、映像と同期を合致させた処理を行うようにする。この対策は、RTVP、Picot-system共に採用しているが、具体的な構成法については4章、5章で述べる。

3. 5 プログラムロード方式

放送においては、演出効果を上げるために「色変換を行った後に幾何学変換を行う」といったように、次々と処理内容を変えて行くのが普通である。これを行っても画面上に切り替えノイズ等の妨害を生じてはならない。この要求を満たすためには、映像信号の垂直掃線期間に切り替えを行うようにしなければならない。しかし、垂直掃線期間は、約1m秒と短いために、長いプログラムのロードは困難である。このため、RTVPでは、プログラムメモリ全てをダブルバッファ化し、切り替えは垂直掃線期間に行うが、プログラムのロードは垂直掃線期間以外でも可能とした。変更されたプログラムは、変更の直後に来た垂直掃線期間に実行を開始する。Picot-systemでは、プログラムメモリはダブルバッファとしないで前もって異なる処理を行うプログラムモジュールを多数プログラムメモリにロードしておき、実行したい処理のプログラムの先頭番地と必要なパラメータのみを与えるようにした。この先頭番地とパラメータを受け渡すためのメモリは、ダブルバッファ化し、いつでもロードできるようにした。

[4] 実験システム R T V P

4. 1 設計指針

R T V P (Real-time video signal processor) は、複数の同一仕様のプロセッサを並列動作させることで映像信号をビデオレートで処理することが可能であるかを検証するための実験システムとして設計した。従って、処理の対象画像も放送品質ではなく、3. 58 MHz と低レートでかつ2入力の白黒映像と小規模のものとした。また、プロセッサ台数も、並列処理やソフトウェアによる機能変更の容易性など基本的な事柄を検証するに足るプロセッサ数8台とした。従って、放送で使われる実用的な処理を全てについてはチェックはしないこととした。このため、将来のLSI化について配慮した回路構成とするが、LSIの開発は行わず市販の個別部品を使用して製作した。

4. 2 アーキテクチャ

4. 2. 1 概要

R T V Pは、図4. 1に示すように、3本の映像転送路(V T L, Video transfer Line)により多段に結合されたプロセッシングユニット群とホストコンピュータ(ミニコンピュータMV4000)から構成される。映像転送路は、それぞれ16bit幅である。2系統の映像信号入力は、それぞれA/D変換部によって標準化周波数3. 58 MHz、量子化数8bitにデジタル化され、3本の映像転送路のうちの2本に供給される。プロセッシングユニットは、映像転送路から映像データを取り込んで処理し、処理結果を再び映像転送路に出力する。プロセッシングユニット群で処理された映像データは、映像転送路の終端でマルチプレクサによって1つが選択され、D/A変換部によってアナログ映像信号に変換される。

映像転送路では、映像信号の標準化周期(280ns)で、データが図4. 1中の左から右へ流れていく。一方、プロセッシングユニットは、140nsのクロックサイクルで動作している。従って、各プロセッシングユニットは、1画素当たり2ステップに相当するプログラムの処理を施して映像転送路に出力する。

ホストコンピュータは、各プロセッシングユニットのマイクロプログラムの作成およびプログラムメモリへのロード、ルックアップテーブルのデータ作成および各プロセッシングユニットのルックアップテーブルメモリへのロード、映像転送路の終端にあるマルチプレクサなどマイクロプログラム以外のパラメータの制御、マイクロプログラムのデバック用に用意したシングルステップ動作などの制御を行う。

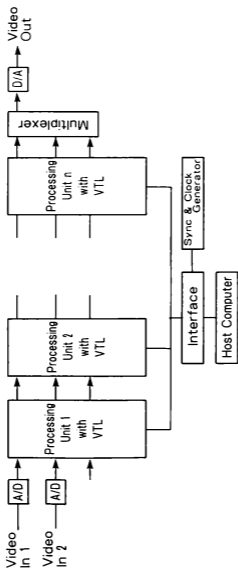


図 4. 1 RTVPの構成図

表4. 1に主な仕様を、図4. 2にプロセッシングユニット8台構成のシステムの写真を示す。

表4. 1 RTVPの仕様

映像入力	標準TV (白黒信号) 2系統 各8bit
映像出力	標準TV (白黒信号) 1系統 10bit
標準化周波数	3. 58MHz
データ語長	16bit
処理速度	140ns/pixel, 33ms/frame
映像転送路	3本 各16bit (映像信号12bit+同期信号4bit)
プロセッシング ユニット数	8台
プロセッシング ユニット仕様	ALU (16bit) 乗算器 (16×16→32bit) ルックアップテーブル (2 ¹² ×12bit) 画像メモリ (128k×12bit) マイクロプログラム制御 (256w×272bit) クロックサイクル 140ns



図4. 2 RTVPの外観写真

4. 2. 2 映像転送路

RTVPでは、3. 2で提案したConfiguration Aを採用した。映像転送路は、映像入力より1本多い3本とした。映像転送路の転送サイクルは280nsで、データ幅は16bitである。このうち、映像情報は12bitで、残り4bitは3. 4のMeasure Bで述べた同期情報（水平同期HD、垂直同期VD、フレームパルスFP、フラッグFLAG）に割り当てた。映像転送路のセレクトとラッチは、プロセッシングユニットに含めた。これにより、プロセッシングユニットの増設のみでシステム規模の拡大が可能である。また、プロセッシングユニット間にD/A変換器を直接接続することにより、各ユニットの出力を見ることができ、処理途中の状態をTVモニターで容易に確認できる。

4. 2. 3 画像メモリ構成法

画像メモリは、3. 3に述べたように、各プロセッシングユニットに持たせた。プロセッシングユニット当りのメモリ容量は128k×12bitである。本システムでは、映像信号を3. 58MHzで標本化しているために、1フレームに相当する容量である。画像メモリは、140nsのサイクルタイムでランダムアクセスが可能である。映像信号の標本化周期、映像転送路の転送サイクルは、280nsの周期なので、1サイクルにつき2回のメモリアccessが可能である。通常は、前半をプロセッシングユニット側の読み出しサイクル、後半を映像信号転送路からの書き込みサイクルとしている。書き込みと読み出しは別々のアドレス発生回路が行う。映像転送路内では、データは走査線構造をしていることから、その書き込みアドレス発生回路はカウンタ回路だけでよい。映像信号とともにアドレスカウンタのクリアのための水平同期パルスと垂直同期パルスを転送し、インタフェースの簡素化をはかる。この機構は、次節で述べる時間軸補正処理にも役立っている。

4. 2. 4 2段階時間軸処理方式

放送局内の映像信号と局外の中継車などから送られてくる映像信号を合成する場合、同一の同期信号で駆動されていないために、直接合成処理することはできない。このような非同期の映像信号を同時に扱うために時間軸補正（フレームシンクロナイズ）機能を実現した。

入力映像信号の標本化周期を t_{in} 、水平同期の周期を t_{H} 、垂直同期の周期を t_{V} 、システムを駆動している標本化周波数を f_{in} 、水平同期信号を f_{H} 、垂直同期信号を f_{V} 、とすると、入力信号を基準同期に変換するためには、

$$t_{in} \rightarrow t_{H}, t_{in} \rightarrow t_{V}, t_{H} \rightarrow t_{in} \quad (4. 1)$$

のように変換しなければならない。これらの値には次の関係がある。

$$t_{1,0} = 525 \times t_{1,1} \quad (4. 2)$$

$$t_{1,0} = n \times t_{1,1} = (n1+n2) t_{1,1} \quad (4. 3)$$

$$t_{1,1} = 525 \times t_{1,2} \quad (4. 4)$$

$$t_{1,1} = n \times t_{1,2} = (n1+n2) t_{1,2} \quad (4. 5)$$

ここで、 n は 1 ライン中の画素数であり、 $n1$ は水平有効画素数、 $n2$ は水平補綴期間中の画素数を示す。3. 5 8 M H z の標本化の場合には、 $n = 227.5$ 、 $n1 = 189$ 、 $n2 = 38.5$ である。

従来の F S や T B C では、この変換をすべて 1 つのバッファメモリで実現しているが、その制御回路はかなり複雑である。それは、位相や周期の異なる 2 系統のクロックを同時に扱わなければならないためである。これを簡略化するために、

$$\text{STEP1 } t_{1,0} \rightarrow t_{1,1} \quad (4. 6)$$

$$\text{STEP2 } t_{1,1} \rightarrow t_{1,2}, t_{1,0} \rightarrow t_{1,2} \quad (4. 7)$$

のように、標本化周波数だけをまず変換し、次に水平と垂直の各周期を変換するという 2 段階方式で時間軸処理する。中間段階 (STEP1 と STEP2 の間) では、 $t_{1,0}$ と $t_{1,1}$ がそのまま、 $t_{1,1}$ だけ $t_{1,2}$ に変更されるため、(4. 5) 式の関係が成立しなくなる。そこで、

$$t_{1,0} = (n1+n2') t_{1,1} \quad (4. 8)$$

のように水平補綴期間の画素数 $n2'$ を削ったり増やしたりして調整する。これにより、A/D 変換部では、STEP1 のクロックの調相を行うので済み、大きなメモリは不要となり、映像転送路も含めたプロセッシングユニットは、同一のクロックで駆動することが可能となる。STEP2 に関しては、4. 2. 3 で述べたように映像転送路の同期パルスによってメモリへの書き込みを行い、読み出しをシステム同期で行うことにより自動的に STEP2 が実現される。

4. 2. 5 プログラムロード方式

3. 5 で述べたように、画面上に妨害を与えないで処理内容を変更するために、プログラムの切り替えは垂直補綴期間に行う。R T V P では、後述のようにプログラムは、1 プロセッシングユニット当たり 2 5 6 ワード (各 2 7 2 b i t) のマイクロプログラムである。垂直補綴期間 (約 1 m s) 内に 8 台のプロセッシングユニットにロードするためには、約 7 0 M B y t e / s e c の転送レートが要求されるが、この転送レートを実現するのはハードウェアの設計上大きな困難を伴う。

そこで、各プロセッサにバッファメモリを設け、ホストコンピュータからバッファメモリへは垂直補綴期間に限らず、随時プログラムのロードができ

るようにした。ホストコンピュータとバッファメモリ間は16bit幅のバスで転送されるので遅いが、バッファメモリとプログラムメモリ間は272bit幅のバスで転送されるため、短い垂直掃線期間でも容易に転送が完了する。また、バッファメモリはプログラムメモリの4倍の容量を確保したため、あらかじめ機能の異なるプログラムをバッファメモリにロードしておけば、バッファメモリからどの部分をプログラムメモリに転送するのかわかるだけでプログラム切り替えが可能になった。さらに、プロセッシングユニットは、並列動作させるためプロセッサのプログラムの内容はほとんど同じでパラメータのみ異なるといった場合が多いので、ホストコンピュータから各プロセッシングユニットにブロードキャストモードでロードできるようにした。

4. 3 ハードウェア構成

4. 3. 1 プロセッシングユニット

プロセッシングユニットPU (Processing Unit) は、図4. 3に示すように映像転送路VTL (Video Transfer Line) も含んだ構造をしている。デジタル化された映像信号をVTLより書き込み制御回路WC (Write Controller) を用いて、画像メモリIM (Image Memory) に書き込む。書き込まれた画像データをアドレス演算部XP (X address Processing part)、YP (Y address Processing part) を用いて読み出す。読み出されたデータは、データ演算部DP (Data Processing part) で処理し、結果をVTLに出力する。これらの制御は、PU内の制御部CTL (Controller) が行う。VTLは、セレクトクとラッチで構成されている。試作では3本のVTLを設けた。

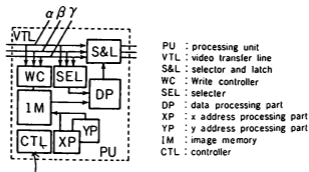


図4. 3 プロセッシングユニットの構成図

1Mは1フレームの容量をもつ。1Mへのデータ書き込みは、VTL上を映像信号と共に送られてきた同期信号でWCが行う。読み出しは140nsのサイクルでランダムアクセスが可能であり、書き込みと独立して、XP、YPが行う。

DP、XP、YPは同一仕様となっており、ALU(16bit)、乗算器(16×16→32bit)、ルックアップテーブル(4096w×12bit)の演算要素を有する。これらは、複数バスによって結合されており、相互にデータ交換が可能である。単一共通バスに比べ信号線が増える欠点はあるが、演算要素を並行して使用することができるためスループットは良くなる。この複数バスを使うことにより、アドレス演算部XP、YPはアドレス処理だけするといったように固定的な使い方でなく、処理に余裕がある場合は負荷分散させることができる。

プロセッシングユニットは、プロセッシングユニット内の制御部CTLによって動作する。このCTLには、RAMで構成されたプログラムメモリがあり、このプログラムに従って動作する。このプログラムメモリをホストコンピュータから書き換えることによって、各種の処理を行う。

CTLは、水平型のマイクロプログラム制御方式を採用している。1PUにつき272bit×256stepの容量である。このマイクロプログラム制御方式はデータ量が大きくなる欠点はあるが、全ての演算要素を同時に制御できる利点がある。プログラムメモリは、ホストコンピュータから見ると16ビット幅となるように分割され、ルックアップテーブルと共に連続したアドレスをもつメモリ空間に配置した。PU番号、アドレス、データ(各16bit)をひとつのバケットとすることにより、変更部分のみをホストコンピュータより転送することが可能となる。また並列動作させるため、各PUのプログラムのほとんどは同じで、パラメータのみ異なるケースが多いので、ブロードキャストモードの転送も可能とした。プログラム変更は、映像信号に切り替えノイズを生じさせないように垂直帰線期間に行うが、前述のように、ホストコンピュータからのプログラム転送は常時行えるように、バッファを設けている。このバッファは4バンクすなわち1k×272bitあり、この範囲内であればバンク切り替えのみでプログラム変更ができる。

1台のPUは、ほぼB4版大の基板を8枚(DP、XP、YP各2枚、その他2枚)使用して実現した。

4.4 ソフトウェア構成

4.4.1 処理の流れ

RTVPは、3.58MHzの標準化レートに対して、7.16MHzの

命令サイクルであるので、実時間処理を行うためには、1PUでは1画素当り2ステップの命令しか実行できない。しかし、PUを並列に動作させることにより、等価的に1画素当りPU数×2ステップのプログラムを実行することができる。PU群は、アレイ的にもパイプライン的にも動作させることができる。この並列動作を始めとするRTVPの動作方式について、混合処理とグラディエント処理を例にとりて説明する。

(1) 混合処理

2つの映像入力AとBの混合処理は、

$$M(i, j) = k \times A(i, j) + (1-k) \times B(i, j) \quad (0 \leq k \leq 1) \quad (4.9)$$

で示すことができる。ここで、 $A(i, j)$ は映像Aのアドレス (i, j) における濃度値、 $B(i, j)$ は映像Bのアドレス (i, j) における濃度値、 $M(i, j)$ は混合結果Mのアドレス (i, j) における濃度値、 k は混合比である。この処理を実現する方法は、いろいろ考えられるが、ここでは図4.4に示すようにタスクを分割して行う場合を考える。すなわち、Aを k 倍してCとして出力するTASK1、Bを $(1-k)$ 倍してDとして出力するTASK2、このCとDを加算しMを出力するTASK3である。この処理のフローチャートは、図4.5に示すようになる。すなわち、

TASK1 IMには映像Aを書き込むようにしておく。まず、乗数 k を乗算器のパラメータ p に置数する(step1)。IMのアドレス x, y をどちらも1とし、垂直同期VDをチェックする(step2)。垂直掃線期間終了し、かつ水平掃線期間が終了したら(step3)、アドレス (x, y) の画像データIM (x, y) を読み出し k 倍すると共に、 x を1増やす(step4)。次に乗算して得られた結果CをVTL α に出力すると共に、水平同期HDのチェックを行う(step5)。step4とstep5を繰り返し、水平掃線期間になったら x を1として、 y を1増やす(step6)。つまり、次のラインに移るわけである。水平掃線期間をチェックして、水平掃線期間が終了したら、再びloop1にはいる(step7,8)。

TASK2 TASK1と同様の処理を映像Bに対して行い、結果Dを出力する。ただし、乗数 p は $1-k$ とする(step1)。

TASK3 CとDを加算する処理で、VTL α とVTL β からデータを取り込み(step3,4)。これらを加算して(step5)、結果をVTL γ に出力する(step6)。これを水平掃線期間になるまで続ける(loop3)。垂直掃線期間、水平掃線期間のチェックは、TASK1、TASK2と同様に行う(step1,2,7,8)。

ここで、出力のためのループに何ステップ必要なのかによって割り当てるPU数が決まる。TASK1については、loop1がstep3とstep4のステップに



図 4. 4 2 映像の混合処理

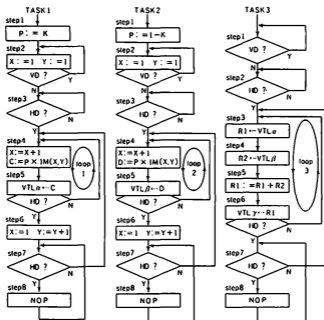


図 4. 5 混合処理のフローチャート

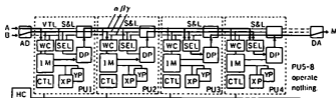


図 4. 6 混合処理の信号系統

対応しており 1画素当り 2 ステップとなっているので、1個のPUで処理できる。TASK2 についても、TASK1 と同様であり、1個のPUで処理できる。TASK3 については、loop3 に step3~6 の 4 ステップ必要であるので、2個のPUが必要である。

信号の流れを図 4. 6 に示す。図 4. 6 では、TASK1 に PU1 を、TASK2 に PU2 を、TASK3 に PU3、PU4 を割り当てている。PU3 と 4 は、VTI. α より C を VTI. β より D を取り込み加算して出力する処理を同期して並列に行う。出力のための step6 は、4 ステップ毎に繰り返される。4 ステップ毎に PU3 が M₁, M₂, M₃, M₄, ... を、PU4 が M₁, M₂, M₃, M₄, ... を順に VTI. γ に出力すると、VTI. γ 上には M₁, M₂, M₃, M₄, M₁, M₂, M₃, M₄, ... と順に隙間のない映像信号系列となる。この際、PU3 と PU4 の出力は同時に行えばよい。このように、PU3 と 4 は、同じタスクをアレイ的に処理している。これに対し PU1 と PU3、4 との関係、及び PU2 と PU3、4 の関係は、パイプライン的に動作していると言える。

(2) グラディエント

グラディエントのひとつである Roberts オペレータは、

$$g(i, j) = |f(i+1, j) + f(i, j+1) - f(i, j)| + |f(i+1, j) - f(i, j+1)| \quad (4. 10)$$

で示される。ここで、 $f(i, j)$ はアドレス (i, j) における濃度値、 $g(i, j)$ はアドレス (i, j) におけるグラディエント値である。1画素の計算に 4 ステップ必要なため、7PU を並列動作させる。基本的な考え方は混合処理と同じであるので詳細な説明は省略するが、ここでは特に注意すべき点として、絶対値を求める処理について述べる。絶対値を求める処理は、符号をチェックし、負の場合は符号を変え正の場合は何もしないようにすれば求められるが、正と負の場合でステップ数が異なる。このため、画素毎にステップ数変動することになり、複数PU間で同期がとれなくなる。そこで、図 4. 7 に示すようにどのパスを通っても同一ステップ数となるように何もしないステップ

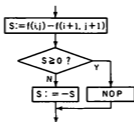


図 4. 7 条件分岐

ブ (NOP) を挿入して対応する。

4. 4. 2 ソフトウェア開発環境

RTVPは、1PU1ステップ当り272ビットの幅をもつ水平型のマイクロプログラムで制御するが、ゲートレベルに到るハードウェアとその動作に関する理解が必要であり、プログラミングは簡単ではない。更に、PU内の演算要素をパイプライン式にも動作させるために、その前の命令の影響を受ける。またPU群を効率よく動作させることも考慮してプログラムを作成しなければならない。

このための言語として、マイクロアセンブラを作成した。開発したマイクロアセンブラでは、PU内部を機能ブロックごとに分け、マイクロインストラクションとして定義し直し、命令群を作成した。表4. 2に命令群を、図4. 8に記述例の一部を示す。PU内には、多くの演算が含まれているので、並列処理のためには1ステップに多くの命令を記述しなければならない。そこで、ゲート動作の細部や使用しない演算部については、記述しなくても処理系(クロスアセンブラ)が自動的に行うようにして、プログラミングの煩雑さを軽減した。この処理系は、ホストコンピュータ上にインプリメントした。このクロスアセンブラは、ソースからRTVP用のオブジェクトを生成する。この際、PUを並列動作させると、複数のPUの同じアドレスに同じプログラムがある場合が多いので、同じプログラム部分を抽出し、ブロードキャストモードのバケットに変換して、転送バケット数を減らすようにした。また、クロスアセンブラと共に、ローダ、コントローラを作成し、プログラムの転送、バンク切り替えなどの制御を行えるようにした。

表4. 2 R T V Pのマイクロアセンブラ命令群

機 能	命 令 群
プログラム制御に関する命令	CONT JMP LDCT RPCT
映像転送路に関する命令	VTL
画像メモリに関する命令	IM
画像メモリのアドレスを制御する命令	SXY HXY
映像転送路への出力に関する命令	SD HD
X P, Y P, D P間のデータ交換の命令	MVS MVH
LUTに関する命令	LUT
乗算器に関する命令	MPI MPY
ALUに関する命令	LDI ADI SBI SRI ANI ORI LDA ADA SBA SRA ANA ORA

```

* sample program
UNIT2/[IM B & VTL B]
01:LDI D0,4000
02:JMP SFD,02 & LDI X0,0000 & LDI Y0,0000
03:JMP NSHD,03
04:JMP SHD,04
05:ADI X0,0001 & MPY DALU,IM & SKY ALU,ALU & HD LUT & VTL C
06:JMP NSHD,05 & LUT MSP & HXY ALU,ALU & SD LUT & VTL C
07:LDI X0,0000 & ADI Y0,0001
08:JMP NSHD,05
09:JMP MC,08
/
UNIT3/
01:CONT

```

図4. 8 プログラム記述例 (-部)

4. 5 各種映像信号処理の適用と評価

先にも述べたように、RTVPは1個のPUにつき1画素当り2ステップの命令を実行でき、n個のPUでは1画素当り $2 \times n$ ステップに相当する命令を実行することができる。従って、ビデオレート処理するためには、処理の総ステップ数に応じて必要な数のPUを割り当てることになる。RTVPに、種々の映像信号処理を適用した場合の必要PU数を、表4. 3に示す。なお、混合処理については、4. 4. 1では並列動作の説明のために冗長な処理形態をとり4PUを使用した。が、(4. 9)式は、PU内の乗算器とALUをパイプライン化して使用することにより4ステップ毎に計算できるため、表4. 3では2PUと記述してある。また、動きベクトル検出は、代表点マッチング法¹⁾を適用した場合、アフィン変換は最近傍補間法を適用した場合、テクスチャマッピングは、映像を球体の表面にマッピングした場合に必要なPU数を示す。処理例を図4. 9に示す。(a)は混合処理、(b)はワイプ処理、(c)はネガポジ反転、(d)は幾何学変換例である。

通常、画像処理装置の性能を評価するには、①典型的な画像処理のベンチマークプログラムの処理時間で比較する方法^{1)~6)}、②単位時間や画素

表4. 3 各種映像信号処理のRTVPへの適用

処理内容		処理に必要なPU数
濃度値 変換	ゲイン調整	1
	γ補正	1
	しきい値調整	1
複数 映像間 演算	混合処理	2
	マスキング	2
	横ワイプ	1
近傍 画素 演算	水平微分	2
	平滑化(2×2)	5
	グラディエント	7
時間軸 処理	フレームシンクロナイズ	1
	フレーム間差分	3
	動きベクトル検出	6
幾何学 変換	アフィン変換	1
	テクスチャマッピング	5



(a) 混合処理



(b) ワイプ



(c) ネガポジ反転



(d) 幾何学変換

図4.9 処理例

当りの演算回数を評価するMOPS、PIXOSを用いて比較する方法。③演算並列度、画素並列度、近傍並列度、ビット並列度などの評価尺度の積で比較する方法などが提案されている。④が、ヒネオレート処理装置であるRTVPでは、すべての処理が1フレーム当り1/30秒で完了することが前提であるので、④のように処理時間で評価することかできない。また、RTVPはPIE数に比例して単位時間当りの演算回数(ステップ数)を向上させることかできるため、④のような評価法は意味を持たない。⑤については、アーキテクチャの評価になってシステム性能評価とはなりにくい。

一般に、並列処理ではプロセッサ数を増やすことによって処理能力を向上させるが、アーキテクチャに依存して向き不向きの処理があり、プロセッサ数に比例した性能を出すことかできないのが普通である。例えば、完全並列方式では近傍データを扱うのには向いているが、幾何学変換を行うためにはプロセッサ間でデータのやり取りが必要になり効率か下がることかあけられる。

そこで、RTVPの処理能力を評価するために、RTVPに適用した各種処理を汎用のコンピュータにも適用して、処理時間を計測し、処理内容の違

いによる処理時間の変化のようすとRTVPにおける必要PU数とを比較した。表4.4は、画像のコピー処理を基準にして、ミニコンピュータでの各種映像信号処理の相対処理時間Tと、RTVPにおける必要なPU数/T(=R)を比較したものである。ノイマン型のコンピュータでは処理速度は処理手順の質よりも処理手順の量に依存しており、処理内容によるばらつきは少なく、汎用性は高いと言える。従って、PU数の増加率がミニコンピュータにおける処理時間の増加率に近いほど汎用であると言える。評価は、14.3MHz標準化に準じた画素数754×483の画像に対して行った。入力画像配列IM1を出力画像IM2にコピーする処理を基準にして処理時間を示した。仮想記憶によるメモリスワッピングの影響を小さくするために、画像の配列は2個に限定し、2入力必要なものについてはIM1とIM2を人力画像としIM2を出力画像とした。

表4.4 RTVPとミニコン(MV4000)の処理能力比較

処理内容		ミニコン上の(*) 相対処理時間 T	R = $\frac{\text{必要PU数}}{T}$
画質 変換	ゲイン調整	1.3	0.8
	γ補正	1.3	0.8
	しきい値調整	1.3	0.8
複 数 映 像 演 算	混合処理	2.0	1.0
	マスキング	1.5	1.3
	横ワイプ	1.5	0.7
近 傍 演 算	水平差分	1.5	1.3
	平滑化(2×2)	2.4	2.1
	グラディエント	2.6	2.7
時 間 軸 処 理	フレームシンクロナイズ	— (**)	— (**)
	フレーム間差分	1.5	2.0
	動きベクトル抽出	1.9	3.2
幾 何 学 変 換	アフィン変換	1.3	0.8
	テクスチャマッピング	1.8	2.2
画像のコピー		1.0	1.0

(*) 画像のコピーを基準(1)とした。

(**) フレームシンクロナイズのシミュレーションは行えない。

表4. 4によれば、濃度値変換、複数映像間演算、アフィン変換などは、 $R=0.7 \sim 1.3$ とミニコンピュータと同程度であるのに対し、近傍演算や時間軸処理に関しては、 $R=1.3 \sim 3.2$ とミニコンピュータに比べ処理効率が良くない。

4. 6 改善すべき課題

RTVPの試作、各種映像信号処理の適用実験から、次の問題点や改善すべき点が明らかになった。

(1) 表4. 4からもわかるように、近傍演算や時間軸処理では、ミニコンピュータに比べ処理効率が良くない。これは、VTLあるいはIMより2つ以上のデータを同時に取り込んで処理できないためである。この点については、A.L.U.に使用したマイクロプロセッサAMD 2901が2つの入力のうち、一方は内蔵のレジスタに固定されており外部より2つのデータを同時に取り込めないことに起因する。RTVPでは、実験システムということで市販のマイクロプロセッサを使用したため実現できなかったが、1度に外部から2つ以上のデータが取り込めるよう演算部とすれば、

スムージング	5PU	→	3PU
グラディエント	7PU	→	4PU
フレーム間差分	3PU	→	2PU
動きベクトル	6PU	→	4PU

など、かなり改善することができる。演算部と画像の入出力部及び画像メモリは密につながっている必要がある。

(2) VTLが3本では少ない。例えば、図4. 10に示すように処理形態を考える。すなわち、

- ①映像入力Aを複数のPUで処理して、処理結果CをVTLに出力する。
- ②映像入力Bを複数のPUで処理して、処理結果DをVTLに出力する。
- ③CとDを複数のPUで処理して、処理結果EをVTLに出力する。
- ④AとEを複数のPUで処理して、処理結果FをVTLに出力する。
- ⑤BとEを複数のPUで処理して、処理結果GをVTLに出力する。
- ⑥FとGを複数のPUで処理して、処理結果HをVTLに出力する。
- ⑦Hを最終の処理結果として、映像出力に出力する。

この場合、A～Hの8種の映像をVTLに載せる必要があるが、うまく割り当てても、最低5本のVTLが必要である。この主な原因は、映像を後段のPUで使うために、途中のPUが使用しないにもかかわらず、VTL上を転送しなければならぬからである。処理形態が複雑になると、さらに多くのVTLが必要である。試作では白黒映像であったが、カラー化するとRGB

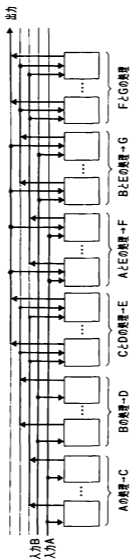


図4. 10 多くのVTLが必要な場合

コンポーネントの場合3倍のVTLが必要となり、回路規模が大きくなってしまふ。VTLの機能を単純な転送路ではなく、可変構造を有するものにしたと使用効率が上がらない。

(3) PU内ではフィードバックループを構成できるが、PU間ではフィードバックループを構成できない。3本のVTLのうち、出力の1本を入力のない1本に接続すれば、フィードバックループが1応構成できる。しかし、2つのPU間でフィードバックループを構成したい場合にも余分なパスを通す必要がある。(2)と同様、柔軟性のあるVTLが望ましい。

(4) 絶対値演算は、グラディエントや動ベクトル検出などに頻繁に使用されるために、ソフトウェアではなくハードウェアで実現した方が高速化できる。同様なものに、NAM (Non Additive Mixture) などにおける最大値、最小値演算がある。

(5) 演算のオーバーフローをチェックして、白黒反転を生じないようにクリッピング動作をさせようとすると、余分のプログラムステップが必要となり、リアルタイム性に欠ける。各演算器にクリッピング回路を持たせるべきである。

(6) 乗算器は16bit×16bitして32bit出力のものであるが、乗算結果を上位ワードと下位ワードの2回に分けて読み出す方式のLSIが入手できなかった。その他の演算器は16bit処理のため、どちらか一方を使うことになる。しかし、ゲイン調整を行う場合には、1.0前後の係数を映像データに乗ずる必要があるが、下位ワードを使うと1.0、2.0など整数倍しか扱えず、上位ワードを使うと1未満の小数が扱えるだけで、1.2倍などの係数が精度よく扱えない。このために、1.0前後で小数部を含む係数が扱えるような出力モードがあると便利である。

(7) レジスタがALUのみに付属しているために、乗算器の係数を設定するために、ALUを動作させなければならず、無駄が生じる。能力を十分生かすためには、各演算器にレジスタを設けるべきである。

(8) アドレス計算が16bit精度では不足である。これは、2.2の検討でも明らかである。RTVPでは、アドレス演算器XP、YPにデータ演算器DPと同じ仕様の演算回路を使用したためである。アドレス計算に適した演算回路と精度とすべきである。

(9) 映像信号を扱うために、HD、VDなどの同期パルスをチェックする必要があるが、ALUのフラッグをチェックしようとすると同期パルスがチェックできない。交互にチェックすることで対応すると、1クロック分の遅れが生じる。少なくとも2つのフラッグが同時にチェックできるようにすべきである。

(10) VTLの制御プログラムがPUのプログラムと混在して記述されているために、プログラム構造が分かりにくい。処理そのものの記述とPU間の結合に関する記述は、別々に行える方がよい。

(11) ホストコンピュータとPU間は、ホストコンピュータのシステムバスを延長して接続したが、バスはあまり延長することができない点と接続本数が多くなる点から、PU数の増大には限度がある。LAN (Local Area Network) などの採用が必要である。

(12) PU数が増えると、全てのプログラムを1フィールド内で送れなくなる。前もって、ロードしておく必要がある。この際、1kwのプログラムメモリの容量では不足である。同様のことがルックアップテーブルでも通える。

RTVPの試作により、上記の問題点、改善すべき点などの知見が得られ、ビデオレート映像信号処理システムの構成上のキーとなるノウハウが蓄積できたため、これらをもとに実用システムを作ることにした。

[5] 実用システム

P i c o t - s y s t e m

5. 1 設計指針

RTVPの試作結果を踏まえて実用システムPicot-systemを開発した。Picot-systemを開発するに当たり、以下の目標を設定した。

NTSC信号を対象とする。信号形式は、コンポジット信号（NTSC信号）あるいは、コンポーネント信号（[R,G,B], [Y,R-Y,B-Y], [Y,I,Q]）である。コンポジット、コンポーネントの両方を扱うために、標準化周波数は、14.3MHzを採用する。プロセッサの処理速度もこの標準化周波数と同じにする。すなわち、サイクルタイム70nsである。演算語長は16ビットとする。この仕様であれば、放送に使用しても十分な画質である。

この基本システムに加え、実用的なシステムとして放送用ビデオスイッチャを開発し、放送現場で稼働させ検証を行う。このために、ビデオ入出力は、最低必要な4~8程度の入力数、2~3程度の出力数に対応できるようにする。また、プロセッサ数は、通常使われている特殊効果などを含む3効果列程度に対応できるように、200台程度用意する。このために、主要部は、LSI化して小型化を進める。さらに、専用の操作卓とアプリケーション・ソフトウェアを作成する。

5. 2 アーキテクチャ

5. 2. 1 概要

Picotシステムは、図5.1に示すように多段に接続されたクラスタ群、ホストコンピュータ（ラップトップ・パーソナルコンピュータJ3100）、コントロールパネル（制御卓）から構成される。クラスタ間は、単方向性バス16系統で接続されている。初段のクラスタの入力に映像信号を入力し、クラスタ群で処理し、最終段のクラスタの出力に処理結果を出力する。映像入力、映像出力とも最大16系統接続できる。標準化周波数は、14.3MHzである。クラスタは、図5.2に示すように16台の同一仕様のプロセッサを中心に構成される。16台のプロセッサの入出力48系統（各プロセッサは、2入力1出力になっている）及び、クラスタの外部入力16系統、外部出力16系統が、ネットワークで結合されている。各系統は、同期情報（水平同期HD、垂直同期VD、フレーム同期FP）を多重化できるように、16bitのデータに同期ビットを付加した17bit幅である。クラスタ内には、コントローラがあり、プロセッサとネットワークを含む

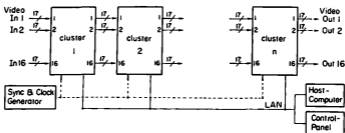
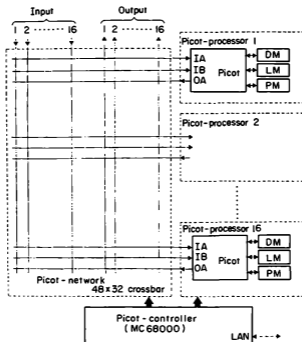


図 5. 1 Picot-systemの構成図



DM: Data Memory (256Kx16b) PM: Program Memory (4Kx32b)
 LM: Line Memory (1Hx16b)

図 5. 2 Picotクラスタの構成図

ラスタの状態を管理する。

A/D, D/Aは、通常8ビット程度であるので、データのダイナミックレンジを確保でき、かつ演算精度も保つことができるように、16ビットの上位から5ビット目から割り当てるようにした。これにより、データレンジとして±8倍までの余裕があり、かつ小数部に4ビットの精度を保つことができる。

コントロールパネルは、ジョイスティック(X, Y, Zの3軸型)、スイッチ群などから構成されている。ホストコンピュータでは、主にプログラムの開発とシステム全体の制御を行う。クラスタ群、ホストコンピュータ、コントロールパネルは、すべて1本のLANで接続されており、ホストコンピュータあるいはコントロールパネルからクラスタ群に、プログラム、パラメータ、コマンドを送り、これをコントローラが解析し、プロセッサやネットワークを制御することによって、各種の動画像処理を行う。

表5. 1に主な仕様を、図5. 3に4クラスタ構成のシステムの写真を示す。

表5. 1 Picot-systemの主な仕様

映像入力	標準TV (RS170A) 1G系統 ([NTSC], [R, G, B], [Y, R-Y, B-Y]) など
映像出力	標準TV (RS170A) 1G系統 ([NTSC], [R, G, B], [Y, R-Y, B-Y]) など
標準化周波数	14. 3 MHz
データ語長	16 bit
処理速度	70 ns/pixel, 33ms/frame
プロセッサ数	16 × n (n: クラスタ数)
制御方式	マイクロプログラム制御
処理機能	点演算 (濃度値変換, しきい値処理など) 色処理 (色補正, クロマキーなど) 複数映像間演算 (ミックス, ワイプなど) 近傍画素演算 (輪郭強調, ノイズ除去など) 幾何学変換 (アフィン変換, 透視変換, マッピングなど) 時間軸処理 (フレーム間処理, 動き検出など)



左から、本体(A/D, D/A, クラスク)、ホストコンピュータ、コントロールパネル

図5.3 Picot-systemの外観写真

5. 2. 2 並列処理方式

Picot-systemでは、3. 2で提案したConfiguration Bを採用した。RTVPで採用したConfiguration Aでは、①フィードバックループが構成しづらい、②前段のプロセッサから後段のプロセッサにデータを渡すためだけに多くのVTLを使用するなどの欠点を回避するためである。

Configuration Bでは、16個のプロセッサをひとまとまり(クラスク)にして、この中では任意の接続ができるようにし、このクラスクを多段に接続して構成する。クラスク間は、16本のバスで接続されている。これにより、クラスクをまたがって、複数の映像入力を並行して供給したり、処理結果の映像や処理途中の映像も出力でき、処理能力をあまり犠牲にすることなく、パイプライン処理や並列処理を実現することができる。

5. 2. 3 メモリ構成法

画像メモリは、3. 3で述べたように、各プロセッサにメモリをもたせる方式とした。必要なデータの配置は、処理の内容によって前もってわかっているのて、処理に先立ちそれぞれのプロセッサのメモリに格納しておく、プ

ロセッサが必要とするデータが複数のプロセッサで同じであることが起こりうるが、その場合は、別々のプロセッサに同じ内容のデータが確保されることになる。この方式は、メモリアクセス競合や通信手続きが不要のため、プロセッサ数に比例した処理能力向上が望める。画像メモリの容量は、幾何学変換を行うのに最低必要な1フィールド（インクレーズ走査しているNTSC信号では、1フレームの半分の容量、256kw）とした。

データメモリへの書き込みと読み出しはRTVPと異なり同じアドレス演算器を使用する。従って、書き込みと読み出しは同時に行えない。これは、標準化周期が70nsのために同時に書き込みと読み出しを行うためにはサイクルタイムが最悪でも35nsのメモリが要求され、コストアップにつながるからである。これは、幾何学変換する場合に、同じメモリに書き込みながら読み出すと、読み出しより早くなる追越しが起きて、1つの画面に異なる時刻の画面が合成されて表示される現象を防ぐ効果もある。

5. 2. 4 同期方式

3. 4で述べたように、多数のプロセッサを使用するマルチプロセッサシステムにおいては、処理過程でさまざまな遅れが生ずるので、プロセッサ間の同期対策が必要である。

Picot-systemでは、次に示す3つの方法を採用した。

(1) 各プロセッサに、最大7画素の可変遅延要素と、最大1ライン(910画素)の可変遅延要素(ラインメモリ)及びフィールドメモリをもたせた。これをソフトウェアで制御することにより、ずれを吸収できるようにした。また、フィールドメモリを位相合わせのために使用するのは最終処理出力のみとした。

(2) システム同期信号を遅らせるために、クラスタ内の同期信号分配回路に可変遅延回路(最大1024クロック)を設けた。システム同期信号は、本来プロセッサ毎に変えるのが望ましいが、処理のやり方によって供給すべき同期信号を変える必要があり現実的でない。そこで、クラスタ内は同じシステム同期信号とした。このため、プロセッサによっては、映像とシステム同期が完全には一致しない場合が生じ、有効画面が欠落することがある。これを避けるために、水平同期信号の幅を同期信号規格よりも狭くした。Picot-systemにおける処理の遅延量Dは、 $(14 + 2 \times e) \times p + 2$ × cである。ここで、cはシステムを構成するクラスタ数(図5. 3の例では、4台)、pは処理に割り当てるプロセッサ数、eは処理に割り当てるプロセッサ内の演算器数である。従って、1クラスタ当りの最大の遅延量Dは162(e=3, p=16, c=1の時)である。このため、1段目のクラ

スタに対して、2段目は162クロック、n段目は $162 * (n - 1)$ クロック遅れたシステム同期信号を供給すればよい。通常、全ての演算器を使用することや、全てのプロセッサをバイブライン接続して使用することは希であるので、経験的には、Dは66 ($e = 2, p = 8, c = 1$ の時)でも、問題はないと考えている。5. 6. 1で述べるビデオスイッチャではこの値に設定している。

(3) 映像を同期情報付きで送るために、プロセッサ間及びクラスタ間の接続を、16ビットの映像データに同期ビットを付加した17ビット幅にした。同期情報は、プロセッサのプログラム制御回路に入力でき、これにより、映像と同期に合致してプロセッサを制御することができる。

5. 2. 5 プログラムロード方式

3. 5で述べたように、プログラムのロードは垂直補給期間内に行う。しかし、RTVPのように毎垂直補給期間に強制切り替えを行うことはせず、プロセッサがRESTARTと名付けられた命令を実行した場合のみに行うようにした。これは、フィールドをまたがる処理、例えばフレーム単位の処理やフィールドとは関係なく処理が完了するまで行うタイプの処理に対応するためである。

実用システムでは、数百台のプロセッサを接続することになるため、1プロセッサ当たり200ステップのプログラムとしても、データ量は100kbyteを超える。これをロードするためには、1フィールド(1/60秒)全てを費やしたとしても、数Mbyte/sec以上の転送レートが要求される。この転送は、パラメータの計算など他の処理を行わねばならないホストコンピュータには処理しきれない。そこで、複数種類のプログラムを前もってプロセッサにロードしておき、この中からどれを実行するのかを指示する方式とした。この指示のためには、実行したいルーチンの先頭アドレスとともにパラメータを渡す必要がある。このデータのやり取りのために、プロセッサ内に専用のインタフェース・レジスタを2セット(各8w×16bit)設けた。これにより、メインプログラムが、サブルーチンを呼ぶような形式で、多数のプロセッサを制御することができる。

5. 3 ハードウェア構成

5. 3. 1 プロセッサ

プロセッサは、14.3MHz(サイクルタイム70ns)の速度で映像信号を16bit精度で処理する。構成を図5.4に示す。入力ポートを2系統、出力ポートを1系統もつ。データ幅は、同期情報を送受するためのビ

ットも含め17bitである。ALU (Arithmetic Logic Unit)、乗算器、アドレス演算回路HAU (Horizontal Addressing Unit)及びVAU (Vertical Addressing Unit)、シーケンサ等からなる。全ての演算は、1サイクル(70ns)で実行できる。これらは、複数バスにより接続されており、パイプライン動作や並列動作が可能である。

(1) 演算回路

ALUは、16bit(2の補数)精度で処理する。加減乗除演算機能に加えて映像信号処理に不可欠な絶対値、最大値、最小値演算機能を行なう。また、オーバーフローに対しては、自動的にクリップがかかる。これにより、視覚的に目立つ白黒反転を防ぐことができる。

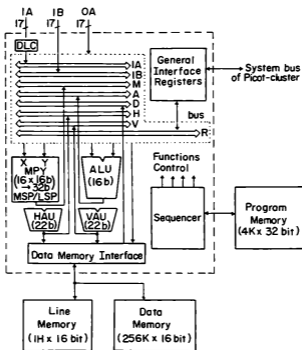


図5. 4 プロセッサの構成図

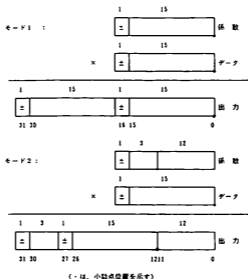


図5. 5 乗算器の演算モード

乗算器は、16bit×16bitで行い16bitの出力を得る。2次のBoothのアルゴリズムを採用しており、ALUと同様1サイクルで求めることができる。演算モードは、図5. 5に示すように通常の乗算モード1に加え、係数の絶対値を最大8に抑えて小数部に12bit割り当てることにより、演算時の有効桁数が多くなるモード2を用意した。

(2) アドレス回路とデータメモリ

H A UとV A Uは、アフィン変換が可能なアドレス演算回路である。図5. 6に示すように、同一の回路構成をしており、それぞれ、加減算回路とコンパレータ2個から成る。2. 2で検討を加えた図2. 1に初期値レジスタhint2, vint2を加えた構成をしており、それぞれ(2. 1)(2. 2)式の演算を毎クロック行うことができる。演算、レジスタの選択は、マイクロプログラム制御のシーケンサで行う。しかし、同じ演算を繰り返す場合は、命令を繰り返すように記述しなくても演算は自動的に継続するようになっている。更に、演算と同時に比較を行うためにコンパレータを2つ設けた。このコンパレータは、前もって比較値をセットしておけば毎クロック比較しフラグを出力する。コンパレータ1の場合、H A U, V A Uそれぞれのデー

タが比較値レジスタ $hcmp1$, $vcmp1$ 未満のときにフラッグ $hauf1$, $vauf1$ が立ち、コンパレータ2の場合、比較値レジスタ $hcmp2$, $vcmp2$ を越えるときそれぞれフラッグ $hauf2$, $vauf2$ が立つ。これらのフラッグは、シーケンサの判定条件に設定できる。この条件と同期信号（水平同期 HD 、垂直同期 VD 、フレームバースト FP ）を同時にチェックでき、優先度も任意に設定できるため、2. 2. 2. 3で述べた基本波形発生や幾何学変換などにおいて、遅滞なく処理を進めることができる。

HAU , VAU の演算語長については、2. 2. 2. 3の検討結果があり、一方ハードウェア規模を小さくし、かつ演算速度を1サイクル70ns以下に抑える必要があったため、22bitとした。これは、標準TV信号を14.3MHzで標本化した場合に1画素単位の精度を満たす語長である。標準的には、整数部を16bit、小数部を6bitとしている。

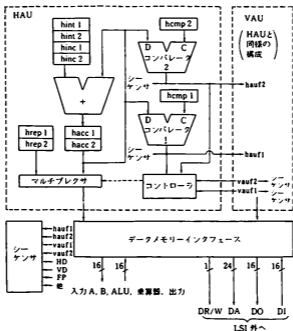


図5. 6 HAU, VAUの構成図

H A U と V A U を使用すると、演算と並行してデータメモリへのアクセスが可能である。データメモリは、256K×16bitの容量(1024×256画素の画像に相当)で、1サイクルで書き込み又は読み出しができる。

H A U と V A U の各22ビットの演算出力から、アドレス幅18bit、容量256kwのデータメモリにアクセスするためには、アドレスの変換が必要である。NTSC信号用フィールドメモリを想定すると、横1024画素×縦256画素となるので、H A U から10bit、V A U から8bit抜き出して連結し18bitとすればよい。この方式では、上位ビットが切り捨てられるために縮小時に、H A U、V A U 内の演算のオーバーフローよりも先に、出力アドレスが循環し同じ画像が繰り返し出力されてしまう。これを避けるためには、H A U、V A U からデータメモリへの出力をそれぞれ0~1023、0~255の範囲に抑えなければならない。これをプログラムで行っては複数のプログラムステップを必要とするので、ハードウェア的にクリップされるようにした。すなわち比較値レジスタhcomp1、hcomp2、vcomp1、vcomp2の値を有効画素の端に設定し、すべてのフラッグhauf1、hauf2、vauf1、vauf2が0となる有効画素内ではhacc、vaccをそのまま出力し、それ以外の条件となる有効画素外にはみ出した時、hrep1、vrep1の値が出力されるモードを設けた。hrep1、vrep1の画素に0を設定しておけば、繰り返しはなくなり黒が出力されることになる。この他に、

① 全てのフラッグhauf1、hauf2、vauf1、vauf2が0の時、hrep1、vrep1、それ以外の時は、hrep2、vrep2を出力する。(マスキングパターンを発生させるために使用する。)

② hauf1=1の時hrep1、hauf2=1の時hrep2、それ以外の時haccを出力する。V A U も同様に動作する。(H A U、V A U 独立のクリップで、データメモリをルックアップテーブルに使用の際にテーブルの参照可能範囲を越えないように制限するために使用する。)

などのモードも用意した。

なお、データメモリは、ルックアップテーブルとしても使用でき、16bit入力16bit出力の非線形演算器が4個用意できる。テーブル内容をロードするためには、後述のコントローラを使用する。データメモリは、コントローラのメモリ空間にマッピングされており、書き込み読み出しは容易である。また、ルックアップテーブルは、幾何学変換の際の4点補間にも使用できる。このために、H A U、V A U の小数部を連結して、データメモリにアクセスするモードも用意した。(2、23)式を変形すると、

$$d = (1-q)(1-p) \cdot d(i, j) + (1-q)p \cdot d(i+1, j) + q(1-p) \cdot d(i, j+1) + qp \cdot d(i+1, j+1) \quad (5. 1)$$

のようになる。補間係数 $(1-q)(1-p)$, $(1-q)p$, $q(1-p)$, qp を前もって計算しデータメモリにロードしておき、これをルックアップテーブルとして参照するようにすれば、プロセッサ1台で(5.1)式の各項を計算できる。

この他、複数のプロセッサを使ってアフィン変換より複雑なアドレス発生を行いたい場合がある。このために、他のプロセッサからアドレスを受け入れられるようにH A Uの出力の代わりに入力Aを、V A Uの代わりに入力Bをデータメモリのアドレスに接続するモードも用意した。

(3) 入出力回路

2系統ある入力のうち1系統は、水平方向の画素間演算が容易になるように最大7画素の遅延がプログラムで可変にできるようになっている。また、垂直方向の画素間演算のためには、最大1ラインまで遅延量を可変にできるラインメモリを有している。出力は、A L U、乗算器、H A U、V A Uの演算結果、データメモリ、ラインメモリの出力に加え、A L Uの演算フラッグを直接出力することができる。これにより、例えばあるプロセッサでA画像とB画像を比較したフラッグを画素単位で出力すると、次のプロセッサでは、このフラッグによって処理を画素単位に切り換えることができ、プロセッサ間で運動したプログラム制御を行うことが可能である。なお、入出力は、5.2.4で述べたように、同期情報を付加するために17bit幅になっている。

(4) シーケンサ

プロセッサは、内蔵のシーケンサによってマイクロプログラム制御される。シーケンサは、マイクロプログラムメモリの内容をデコードして、各処理ユニット、入出力、及びシーケンサ自身を制御する。プログラムは、 $4k \times 32$ bitの容量をもつプログラムメモリに格納する。本プロセッサは、画像、特に映像信号を主対象としているため、映像信号処理をかなり意識した制御方式を採用しているが、これについては、5.3.2で述べる。

(5) コントローラとのインタフェース

プロセッサ内には、各演算器内にあるレジスタとは別にL S I内で共通に使用できるインタフェース・レジスタが2セット(各 $8w \times 16$ bit)ある。このレジスタは、コントローラとのインタフェースも兼ねており、レジスタをバッファとして、データなどの送受を行うことができる。これについては、5.3.3で述べる。

なお、プロセッサは、プログラムメモリ、データメモリ、ラインメモリを除いたほとんどの部分をC M O Sゲートアレイを用いて、L S I化した。このL S Iについては、5.3.6で述べる。図5.7にプロセッサ・ボードの写真を示す。約 $20cm \times 10cm$ の大きさである。

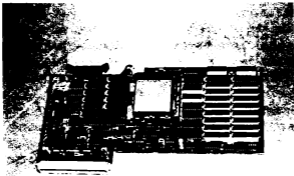


図5. 7 プロセッサの外観写真

5. 3. 2 プロセッサの制御機構

本プロセッサは、画像、特に映像信号を主対象としているため、映像信号処理をかなり意識した制御方式を採用している。

プロセッサは、前述のように多くの処理ユニットを有しており、これら全てを一度に動作させようとする、100bit幅以上のマイクロ命令が必要となる。プログラムメモリを外付けとするためには、マイクロ命令用に多くのピンを割り当てることはできない。しかし、ピン数を抑えるためにマイクロ命令のビット幅を抑えると、全ての演算器を同時には有効に使えなくなり、効率が下がる。ここに、相反する要求があった。

そこで、次に示す手段を講じた。

(1) 映像信号処理においては、『フレーム内(あるいはライン内)の全ての画素に対して、同じ処理を行う場合が多い』ことに着目し、一旦セットした演算器の命令は、新たに演算器に関する命令がセットされるまで、その演算器は、同じ命令の処理を続けるようにした。

(2) 映像信号処理においては、『演算結果フラグや同期信号などの複数の条件をチェックしてから演算を行う場合が多い』ことに着目し、一度に最大2つの条件をチェックでき、これに基づいて分岐するようなシーケンス命令とした。

(3) また、同じ処理を続けることが多い点を踏まえ、通常のプロセッサが有するCONTINUE、JUMP命令に加え、自番地にループす

表 5. 2 インストラクション・セット

標準の命令形式 (32bit)

シーケンス命令(16bit)	ファンクション命令(16bit)
----------------	------------------

シーケンス命令 (16bit)

R J C CC1のとき jump, CC2のとき continue, それ以外 repeat
 R C J CC1のとき continue, CC2のとき jump, それ以外 repeat
 R J J CC1のとき jump 1, CC2のとき jump 2, それ以外 repeat
 C J J CC1のとき jump 1, CC2のとき jump 2, それ以外 continue
 ・ jump 1, jump 2 は, それぞれ別のジャンプアドレスが記述できる.
 ・ CC1, CC2を設定しないことも可能である. 従って,
 R 無条件 repeat
 C 無条件 continue
 J 無条件 jump
 R J CC1またはCC2のとき jump, それ以外 repeat
 R C CC1またはCC2のとき continue, それ以外 repeat
 C J CC1またはCC2のとき jump, それ以外 continue
 も記述可能である.

ファンクション命令 (16bit)

NOP 何もしない
 ALU ALUに関する命令
 MPY 乗算器に関する命令
 HVAU HAU, VAUに関する命令
 DMI, DMB データメモリに関する命令
 DLT, TMC ライン遅延と番線遅延に関する命令
 OUT 出力に関する命令
 MGS, MGG, MFG 演算レジスタ及び汎用レジスタ間のデータ転送命令

例外の命令形式 (32bit)

ファンクション命令(32bit)

シーケンス命令 (0bit)

記述されていないが, continueを実行する.

ファンクション命令 (32bit)

LDR レジスタにイミディエイト値をロードする命令
 LDA IAMにアドレステーブルをロードする命令
 LDP IPMにアドレステーブルをロードする命令

る命令 REPEAT を用意した。

(4) シーケンサを 14、3MHz で動作させるため、3 段のパイプライン構造としたが、条件分岐が発生した場合にも、パイプライン動作が崩れて処理が中断しないようにする機構を加え、実時間処理を可能にした。

以上の手段により、実時間性を損なうことなく、マイクロプログラム幅を、32 bit とすることができた。表 5. 2 に、命令表を示す。全ての命令が、32 bit 幅の 1 ワード命令である。ほとんどの命令が 16 bit 幅のシーケンサ命令と 16 bit 幅のファンクション命令からなる。以下では、上記の (1) ~ (4) を実現しているシーケンサの動作について説明する。

シーケンサは、図 5. 8 に示す構成をしている。通常、PC (Program Counter) から与えられたプログラムアドレスを AL (Address Latch) でラッチする。ラッチされたアドレスは次のサイクルで、プログラムメモリにフェッチされ、IL (Instruction Latch) にラッチされる。ラッチされたプ

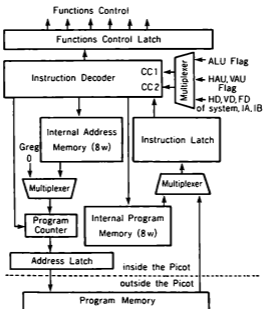


図 5. 8 シーケンサの構成図

プログラムは、次のサイクルで I D (Instruction Decoder) によりデコードされ、F C L (Function Control Latch) でラッチする。この次のサイクルが実行サイクルである。すなわち、(4) で述べたように、フェッチ、デコード、実行の3段のパイプライン構造である。F C L (Function Control Latch) は、以前に実行した全ての演算器 (A L U、乗算器、H A U、V A U など) の入出力や演算機能などの制御情報を保持している。例えば、A L U に関するマイクロ命令がデコードされた場合は、A L U に関する部分の F C L が書き換えられるのみで、そのほかの演算器に関する部分の F C L は前の状態が保持されている。このため、(1) で述べたように保持されている制御情報で常に全ての演算器を動作させることができる。

一般にパイプライン構造を採用すると、条件分岐が発生した場合、パイプラインが崩れ処理が中断する。その結果、映像信号に妨害を与えてしまう。そこで、条件分岐が発生しても処理が中断しないように、I P M (Internal Program Memory) と I A M (Internal Address Memory) をシーケンサ内に設けた。これらは、L S I 内のメモリが L S I 外のメモリに比べ高速にアクセスできることを利用したもので、I P M にはジャンプ先の命令を、I A M にはジャンプ先+1のアドレスを格納しておく、それぞれ8ワードの容量があり、ジャンプ処理に先立つ垂直、水平掃線期間にあらかじめ格納しておく。

これらの動作について、図5. 9に示す。命令がCONTINUEの場合

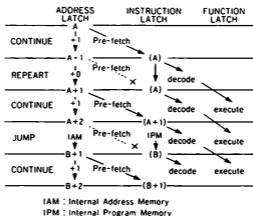


図5. 9 シーケンサの動作フロー

は、PCは+1され、pre-fetchがそのまま生き、ILに命令がラッチされる。REPEAT命令の場合は、PCはホールドされ、ILにはpre-fetchされた命令を放棄して前サイクルの命令がホールドされる。JUMP命令の場合は、IAMからジャンプアドレスをPCにロードし、ILにはpre-fetchされた命令を放棄してIPMからマイクロ命令をロードする。これにより、ジャンプしても、パイプライン制御がくずれない。

また、IDの条件入力にはALU、HAU、VAUのフラッグ、前出の入出力ポートから抜き出された同期情報の中から、2つの条件(cc1、cc2)を設定できる。これにより、(2)で述べたライン単位、フレーム単位の同期をとりながら比較演算を行う処理が可能となった。また、この同期情報を使用すると、プロセッサ間の同期を取ることができ、並列動作に有効である。

5. 3. 3 プロセッサとコントローラ間の通信機構

プロセッサは、上述のシーケンサにより自立的に動作するが、クラスタに付属するコントローラの制御を受け入れる機能がある。

画像処理においては、処理結果に応じて処理内容を変更したい場合があり、放送においては、演出効果を上げるために色変換を行った後に幾何学変換を行うといったように次々と処理を変えて行くのが普通である。しかも、この機能変更を画面上に妨害を与えないで行うことが要求される。

これを実現する機構は、5. 2. 5で述べたRESTART命令とインタフェース・レジスタを用いて実現されている。このレジスタ2セット(A、B)は、必ず一方がプロセッサ側に、もう一方がコントローラ側に向いている。いま、仮にAセットがプロセッサに、Bセットがコントローラに向いているとする。コントローラは、Bセットの0番地にスタート番地を、1～7番地にパラメータを与える。その後、垂直帰線期間になり、RESTART命令がデコードされると、シーケンサはレジスタのAセットとBセットを入れ替え、入れ替わったBセットの0番地の内容をシーケンサのプログラムカウンタ(PC)にロードする。シーケンサは、この値をスタート番地として動作を開始する。パラメータが必要であれば、1～7番地から取り込む。一方コントローラは、自分側に向いたAセットにスタート番地等を書き込めば、次の処理を指示できる。この切り替えは、プロセッサが垂直ブランキング期間内にRESTART命令を発しない限り行われぬし、RESTART命令が発せられてもコントローラから次の処理の指示を与えていなければ行われぬ。従って、不用意に処理が中断して、画面が乱れることはない。インタフェース・レジスタの切り替えを利用すると、プロセッサがインタフェー

ス・レジスタにデータを書き込んでRESTART命令を発すれば、コントローラにデータを返すこともできる。

また、リセット端子を用いてプロセッサをリセット状態にすると、コントローラからプログラムメモリとデータメモリの書き込みと読み出しを行うことができる。従って、ホストコンピュータは、LANを通してコントローラと通信することにより、制御だけでなく、データとプログラムをプロセッサに送り込み、処理を行わせ、その結果を吸い上げることも可能である。

5. 3. 4 ネットワーク

ネットワークは、32入力（プロセッサからの出力16とクラスタ外からの入力16）48出力（プロセッサへの入力32とクラスタ外への出力16）のクロスハブ・マトリックスであり、入力と出力間で任意の接続が可能である。1つの入力を同時に複数の出力に分配することも、複数の入力を時分割に1つの出力に集中することも可能であり、プロセッサをパイプライン状に接続することも、アレイ状に接続することも、さらに両者を組み合わせた形態をとることもできる。このネットワークは、14.3MHzの速度で動作する。ネットワークの切り替えは、プロセッサと同様ネットワークに付属するシーケンサにより、マイクロプログラム制御される。ネットワークの各接続を制御するためには240bit（=48×5）幅のマイクロ命令が必要である。しかし、多くの画像処理では1フィールド内同じ処理を行う場合か

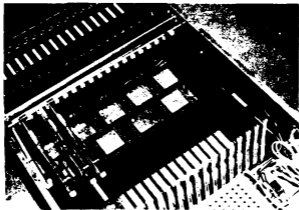


図5. 10 ネットワークの外観写真

多く、1フィールド内ほとんどのルートを固定しておいても問題無いので、1命令では、ひとつの出力をひとつの入力に接続するという制御を行うことにした。従って、全ての出力を確定するのに48命令必要になる。しかし、一度セットされると、新たな命令によって接続が変更されるまでルートは固定されているため問題はない。このように、フィールド単位を基本としているが、領域分割を用いたプロセッサの並列動作においては、画素単位やライン単位に切り換える必要がある。このために、出力1系統については、画素単位やライン単位の切り替え機能を付加した。マイクロプログラムメモリの容量は、128w×16bitである。

ネットワークの主な部分は、CMOSゲートアレイを用いてLSI化した。このLSIについては、5.3.6で述べる。図5.10にクラスタ内に実装されたネットワークの写真を示す。クラスタは、幅約40cm×奥行き約60cm×高さ約20cmの大きさで、プロセッサを挿入するマザーボード上に実装されている。

5.3.5 コントローラ

プロセッサとネットワークは、5.3.3及び5.3.4で述べたように、それぞれが有するシーケンサにより自立的に動作するが、コントローラ(MC68000)によっても制御できる。コントローラは、ホストコンピュータ、コントロールパネルから、LANを通して送られてきたコマンド、データ、プログラムを解析して、プロセッサ及びネットワークに指示を与える。また、プロセッサの処理結果を解析して、プロセッサに新たな指示を与えたり、ホストコンピュータにデータを送り返すなどの処理を行う。

5.3.6 プロセッサLSIとネットワークLSI

プロセッサとネットワークの主な部分は、LSI化されている。プロセッサに関しては、プログラムメモリ、データメモリ、ラインメモリを除いたほとんどの部分が、ネットワークに関しては、プログラムメモリも含めたほとんどの部分がLSI化されている¹⁾。プロセッサLSIの構成図を図5.11に、ネットワークLSIの構成を図5.12に示す。

プロセッサLSI(Picot)は、本システムのプロセッサとして使用することを主目的に設計されているが、単独のDSP(Digital Signal Processor)として、あるいは複数使用して各種映像機器に組み込んで使用することも可能である。データメモリに関しては、最大16Mwまで接続できるようにアドレス幅が24bitになっている。HAUとVAUの各22bitの演算出力から、データメモリにアクセスするために、5.3.1で述べ

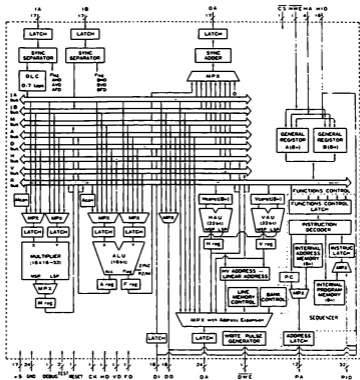


図5. 11 プロセッサLS1のブロック図

たHAU, VAUからそれぞれ10bit, 8bitを抜き出して18bitにするモードに加え, HAU, VAUからそれぞれmビットを抜き出して連結し2mビット列とするアドレス変換回路を付加した. 用意したモードは, 512×512画素 (m=9), 1024×1024画素 (m=10), 2048×2048画素 (m=11), 4096×4096画素 (m=12)の4つである. これにより, 工業用によく用いられる512×512画素から, HDTVを超える大きな画像まで対応できる. また, データメモリには, RAMとROMを混在して使用できるため, 特殊関数をテーブル化してROMとして搭載することも可能である. プログラムメモリも同様に, RAMだけでなくROMも使用可能であるので, ROM化して専用プロセッサとして

表5. 3 プロセッサLSIの主な仕様

Technology	1.5 μ m CMOS gate array
Chip size	14.95 mm \times 14.95 mm
Number of devices	30,000 gates
Power supply	5 V
Power dissipation	1.0 W
Package	223-pin PGA
Clock frequency	14.3 MHz

表5. 4 プロセッサLSIの処理能力

Interframe differenciation	20ms(1Picot)
Noise reduction (2 \times 2)	80ms(1Picot), 20ms(4Picots)
Convolution (3 \times 3)	180ms(1Picot), 20ms(9Picots)
Binarization, Gray-scale conversion	20ms(1Picot)
Maximum detection (3 \times 3)	180ms(1Picot), 20ms(9Picots)
Affin transformation	20ms(1Picot)
Ensemble between images	20ms(1Picot)

各種機器に組み込むことも可能である。

プロセッサLSIは、1.5 μ m HCMOS 2層配線のチャンネルレス型のゲートアレイを用いて試作した。回路規模が30Kゲート程度となったが、複数バスを採用しているなどの要因により配線領域が大きくなることを考慮し、rawゲート数130Kのものを用いた。レイアウトは自動で行ったが、高速度を要するところは配線長が短くなるようにブロック毎に大まかに指定して行った。プロセッサLSIの主な仕様を表5. 3に、LSIの処理能力を表5. 4に示す。

ネットワークLSI (Picot-net) は、本システムのネットワークを小型化するために設計したものである。32入力48出力のネットワークであるが、ピン数の関係からデータ幅が17bitではなく、2bit幅

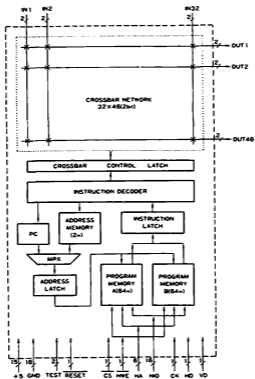


図 5. 1 2 ネットワーク L S I のブロック図

表 5. 5 ネットワーク L S I の主な仕様

Technology	1.5 μ m CMOS gate array
Chip size	14.95 mm \times 14.95 mm
Number of devices	28,500 gates
Power supply	5 V
Power dissipation	1.0 W
Package	223-pin PGA
Clock frequency	37.125 MHz

である。このため、1クラスタにつき9個使用する。ネットワークL S IはプロセッサL S Iと同じデバイス技術を用いて実現した。ネットワークL S Iの主な仕様を表5. 5に示す。プロセッサL S Iの仕様とはほぼ同じであるが、動作周波数は37. 125MHzと高速である。ただし、Pico-t-systemでは、プロセッサに合わせ14. 3MHzで動作させている。

この2種のL S Iは、Pico-t-system以外にも、すでに使用されている。プロセッサL S Iは、東芝のNTSC及びHDTV用効果波形発生装置²⁴⁾に、ネットワークL S Iは、NHK及びJSBの映像音声分配切替システム²⁵⁾に組み込まれて使用されている。

5. 4 ソフトウェア構成

5. 4. 1 処理の流れ

Pico-t-systemの処理の流れを、アフィン変換を例に取り上げて説明する。図5. 13に、処理系統を示す。アフィン変換は、(2. 1) (2. 2)式で実現できる。(2. 1)式の演算をHAUで、(2. 2)式の演算をVAUで行うことができるので、静止画であれば1プロセッサでできる。しかし、動画像を通す必要があるため、2プロセッサを使用する。すなわち、プロセッサのデータメモリのサイクルタイムは、1サイクル(70ns)なので、書き込みと読み出しを同時にできない。そこで、2台のプロセッサをインターリーブして並列動作させる。プロセッサ1は、第1フィールドでメモリに書き込み、第2フィールドで読み出す。プロセッサ2は、その逆で第2フィールドで書き込み、第1フィールドで読み出す。書き込み時は、アドレスを変換しないで書き込み、読み出し時にアフィン変換したアドレスをアドレス回路で発生させてメモリを読み出す。プロセッサ3と4は、プロセッサ1と2で1フィールド遅れとなった画像をさらに1フィールド遅

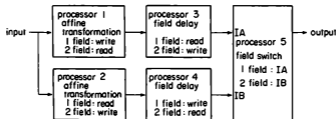


図5. 13 アフィン変換の信号の流れ

らせることでインタレース走査している映像の位置を正規にするためのものである。プロセッサ5は、フィールド毎に書き込み読み出しが切り換えられるプロセッサ3と4の中から、読み出しモードになっているプロセッサを選択して出力する処理を行う。以上は、モノクロ画像の場合であるので、RGBのカラー画像の場合は、3倍の15プロセッサ必要である。

また、コントロールパネル上のジョイスティックの値を、水平拡大率、垂直拡大率、回転角、水平移動量、垂直移動量に割り当て、コントローラではこれらの値から(2.1)(2.2)式のA～Dのパラメータを算出し、プロセッサ1とプロセッサ2のインタフェースレジスタに渡すようにする。これにより、コントロールパネルのジョイスティックを動かすと、リアルタイムで拡大・縮小、回転、移動を行うことができる。

5.4.2 ソフトウェア開発環境

Picot-systemを効率的に動作させるために構築したソフトウェア環境について述べる。ソフトウェア開発は、ホストコンピュータを中心に行う。ソフトウェアは、開発系、実行系、デバッグ系に大別できる。以下、これらについて順に述べる。

(1) 開発系

プログラムを開発するためのソフトウェア群である。プログラム開発は、図5.14の手順に従って行う。まず、プロセッサ、ネットワーク、コントローラの3種のプログラムを作り、これをアSEMBル、リンクし、パッケージ化する。更にこれをLANを通してダウンロードする。プロセッサとネットワークは、アーキテクチャに適合した専用の言語を開発した。コントローラは、68000用のC言語を用いる。また、プロセッサ内のデータメモリにロードするルックアップテーブルを作成するツールも用意した。

プロセッサに関しては、映像信号処理をかなり意識したプロセッサの制御構造を簡易に表現できる図5.15に示すような専用の言語(言語仕様は、付録1参照)でプログラム開発を行う。図5.15は、画面を左右に分割して、画面左側にIA入力を右側にIB入力をワイプして表示するためのプログラム例である。以降はコメントである。分割位置の判定は、HARの演算フラッグhauf2を用いて行う。画面左からカウントを開始しhcap2を超えてhauf2が立ったら、出力を入力IAからIBに切り替える。①は、プロセッサ番号の宣言文である。②は、コントローラがこの関数をコールするために使うモジュール名である。③～⑥は、有効画面部分が始まる前の垂直掃線期間に行う初期値の設定などである。⑦は、垂直掃線期間が終了するのを待っている。⑧～⑩は、有効画面中の処理である。⑪は、一連の処理が終了し、

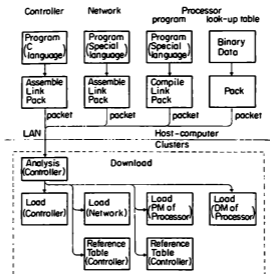


図 5. 14 開発系ソフトウェア

```

picot4 [
"sample1":
    dic(0);
    ldr(hint1, 0);
    ldr(hinc1, 1.0);
    mgs(greg1, hcmp2);
    while(svd) out(clr);
loop:
    while(shd) au(hint1/reg1);
    au(hau/reg1, hinc1);
    while(!shd) {
        while(!hauf2) out(la);
        while( hauf2) out(lb);
    }
    if (!svd) goto loop;
    return;
]

```

① processor number
 ② module name
 ③ set delay line compensator
 ④ set hint1(initial value of hau)
 ⑤ set hinc1(increment value of hau)
 ⑥ set hcmp1(comparison value of hau)
 ⑦ wait for active image area
 ⑧ process for active image area
 ⑨ set hau hint1
 ⑩ start incrementation of hau
 ⑪ no code generated
 ⑫ repeat/cont(hauf2)/jmp⑬(shd)
 ⑬ repeat/jmp⑫(!hauf2)/cont(shd)
 ⑭ no code generated
 ⑮ jump ⑨ if !svd
 ⑯ control returns to controller
 ⑰ end of program for Picot4

図 5. 15 プロセッサのプログラム例

制御の主導権をコントローラに返す命令である。これにより、次の処理をコントローラから受けることができる。フィールドをまたがって処理したい場合は、return文を書かなければ、コントローラに制御が戻ることはない。

ここで、同期信号を用いたシステム制御と一度に2つのフラッグをチェックできる機能を明示的にかつ分かりやすく記述するために設けたwhile文について述べる。図5.15の①～④に示す書式で記述する。外側のネスト(入れ子)の条件が優先順位が高い条件1(cc1)になり、内側の条件が条件2(cc2)になる。①④は、マイクロコードは生成されない。②③はcc1、cc2を設定する2条件3分岐のコードが生成される。②は、cc1にシステムHDが、cc2にhauf2がセットされる。HDが来たとき③にジャンプし、HAUの演算フラッグであるhauf2が立つと次の行④に移る。それ以外の場合は、②を繰り返す。一方③は、cc1にシステムHDが、cc2に!hauf2(!はnotを示す)がセットされる。HDが来たとき次の行④に移り、hauf2が立たないとき③にジャンプする。それ以外は、②を繰り返す。ネストは、ハードウェアの制約から最大2であるので、それ以上設定しようとするときアセンブル時にエラーとなる。値hcmp2が894以上の場合、一般にはHDを読み飛ばしてしまうおそれがあるが、ここでは優先度の高いcc1にHDがセットされているために、hcmp2の値が依存せず④に飛ぶことができ、プロセッサの特徴である実時間処理の機能を簡易に表現できる。shd(システムのHD)やsvd(システムのVD)の代わりにahd(入力IAのHD)、avd(入力IAのVD)をセットすると、システムではなく画像にタグとしてつけられた同期情報を使ってデータ駆動的に動作できる。

また、5.3.2で述べた実時間性を維持するための機構IAMとIPMに関しては、IAMについてはジャンプ先のアドレス+1を、IPMについてはジャンプ先のプログラムを全ての処理に先だちロードしておく必要がある。このためのマイクロコードを自動的に生成して挿入する言語処理系とし、プログラムの負担を抑えた。

ネットワーク用ソフトウェアは、プロセッサと類似した図5.16に示す制御構造の言語(言語仕様は、付録2参照)で記述する。図5.16中のwhile文では水平同期HDの期間以外でプロセッサ2と3の出力を画素単位に切り替える例である。

以上のプログラムは、アセンブル、リンクされ、さらにLAN上へ送出するためにパケット化される。パケットには、クラス番号、プロセッサ番号、モジュール名などの識別情報も含まれる。これらの情報により、プログラムのダウンロードを行う。

```

network[                'program for network
"sample2";             'module name
  net(in1 , pu1a);      'input1 → processor1's IA
  net(in2 , pu1b);      'input2 → processor1's IB
  net(pu1 , pu2a);      'processor1's OA → processor2's IA
  net(pu2 , pu3a);      'processor2's OA → processor3's IA
  while(!shd) {        'switch pixel by pixel
    net(pu2 , out2);    'processor2's OA → output2
    net(pu3 , out2);    'processor3's OA → output2
  }
  return                '
]                       '

```

図5. 16 ネットワークのプログラム例

コントローラでは、コントロールパネルの値を解析して、プロセッサやネットワークを制御する。プログラムはC言語で開発する。詳しくは、次で述べる。

(2) 実行系

実行系は、クラスター群を動作させて、動画像処理を実際に行うためのソフトウェア群である。人間系とのマッチングをとるためには、処理の応答時間は1~2フィールド(16~33m秒)以内が望ましいので、実時間性を重視した設計方針を取った。OS (Operating System) のオーバーヘッドが大きい汎用のパーソナルコンピュータでは、実時間性に問題があるため、ホストコンピュータでは、処理の開始、停止だけを行い、実行時は専用のリアルタイムOSを搭載したコントローラが主体となる方式にした。

図5. 17が実行時のシーケンスである。最初に、ホストコンピュータからスタートをコントローラに指示する(図5. 17中①)。コントローラは、VDに同期してコントロールパネルにパケットの送信要求を出す(図5. 17中②)。コントロールパネルは卓面の状態をパケット化して出力する(図5. 17中③)。データは、加工されることなくジョイスティックなどの値そのものが出力される。コントローラは、パケットを受信(図5. 17中④)、解析(図5. 17中⑤)し、データ処理して(図5. 17中⑥)、プロセッサとネットワークにパラメータを与える(図5. 17中⑦)。プロセッサとネットワークは、パラメータを受け取り、次のVDがきたらプログラムを切り替え(図5. 17中⑧)、処理を開始する(図5. 17中⑨)。以降⑩

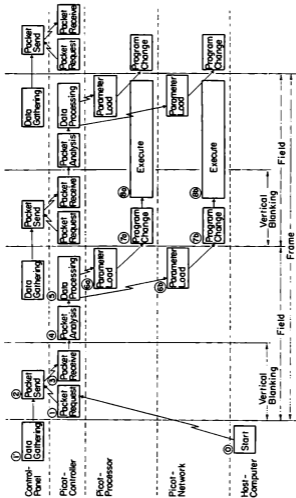


図5. 1 7 実行時の制御の流れ

～④の処理を繰り返す。

このうち、プログラマは、バケット解析とデータ処理の部分（図5. 17中④⑤）をプログラムするだけでよい。その他の処理は、コントローラのOSが行う。また、コントローラのOSは、プログラムのダウンロード時にバケットに付属した差別情報に基づいてモジュール名と絶対番地の対応表を作成するようにしている。これにより、コントローラのCプログラムでは、絶対番地ではなくモジュール名でプロセッサやネットワークのプログラムを関数と呼ぶようにコールできる。前もって、ネットワークとプロセッサのプログラムメモリに、平滑化、微分などの基本ルーチンをライブラリ化してロードしておけば、これをコールするコントローラのCプログラムを作成するだけでも、1連の画像処理を実現できる。

(3) デバッグ系

デバッグ系は、ソフトウェア開発を支援するためのソフトウェア群である。プロセッサのレジスタ、データメモリの内容、コントローラの状態、コントローラパネルから受信したバケットの履歴をホストコンピュータに引き上げることができ、ICE (In Circuit Emulator) 等を使用しなくても、システムの状態をチェックできる。これによりデバッグ作業の効率化が計れた。

5. 4. 3 PicPEn

Pico-le-systemのようなマルチプロセッサでは、ソフトウェア開発が複雑になる。5. 4. 2で述べた開発環境では、実現したい大きな処理を各プロセッサがビデオレートで処理可能な小さなタスクに分割し、プロセッサに割り当てるのは、プログラマの負担である。このタスク分割を行うためには、システムの内部構成について細かな知識が必要である。アプリケーション・ソフトウェアを増やすためには、システムのハードウェアの知識を持たなくても、簡単にプログラムが作成できるツールが必要である。このようなツールとして、PicPEn (Picot Programming Environment) を開発した¹⁴⁾。PicPEnでは、実現したい処理をホストコンピュータ上で図5. 18に示すようにシグナルフローグラフの形で記述すれば、自動的に前述のプロセッサ及びネットワーク用マクロアセンブラ、コントローラ用Cプログラムを生成するツールである。アイコン化されたモジュールは、差分、混合、フレームディレイなどの処理を行うものである。モジュールは、階層化されており、よりプリミティブなモジュールで記述されている。従って、モジュールは、プロセッサと1対1に対応しているのではなく、機能を記述している。モジュールを分解あるいは統合し、タスクをプロセッサに割り当てる作業は、PicPEnが自動的に行う。この際、

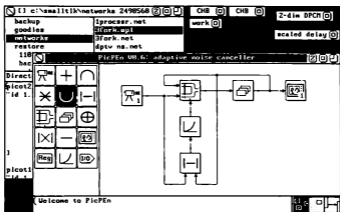


図 5. 18 PicPENの表示画面

- (1) プロセッサは2入力、1出力である。
- (2) プロセッサには、乗算器、ALUが各1個ある。
- (3) プロセッサでは、フレームメモリとルックアップテーブルの共用はできない。

などの制約条件を考慮しつつ、自動的にタスクをプロセッサに割り当てる。また、PicoL-systemでは、パイプライン化のために多くのディレイが生じ、異なるバスを通った時の遅延差が問題になるが、PicPENがディレイ量を計算して必要に応じて自動的にディレイを挿入し、遅延差を補償する¹⁾。これらの処理は、クラスタをまたがっても解析可能になっている。

PicPENは、ホストコンピュータ上のSmalltalk上にインプリメントした。それは、Smalltalkが高度なグラフィック機能を有することや、オブジェクトの概念が階層化したモジュール構造などにマッチしているためである。

5. 5 各種映像信号処理の適用と評価

PicoL-systemでは、入出力される映像信号の標準化周期とプロセッサの命令サイクルが同じであるため、1画素当り1ステップの命令の実行を基本としている。しかし、プロセッサの並列・パイプライン動作により、プロセッサ数に比例したステップ数に相当する処理を実行でき、複雑な

表5. 6 各種映像信号処理のPicot-systemへの適用

処理内容		必要プロセッサ数 ^{*)}
点演算	適度値変換	3
	閾値処理	3
近傍演算	ラブラション(3×3)	9
	スムージング(2×2)	4
幾何学変換	アフィン変換	15
	テクスチャマッピング	33
色処理	色補正	3
	クロマキー合成	18
フレーム間処理	フレーム間差分	21
	動物体抽出	18
映像間処理	映像混合	6
	ワイプ処理	3

*) カラー映像信号(85170A準拠, RGB, 525本/60Hz, (79×1-1)の場合)

表5. 7 プロセッサの処理能力

model ⁽¹⁾	SUN		NEWS	
	elaps	ratio	elaps	ratio
addition, subtraction	0.27sec	8.1	0.49sec	15.0
maximum, minimum	0.24sec	7.2	0.63sec	19.2
absolute	0.41sec	12.6	0.76sec	23.4
multiplication(mpy)	0.88sec	27.0	1.43sec	43.8
multiplication & addition	0.91sec	27.0	1.62sec	43.8
read or write frame-memory	0.35sec	10.8	1.84sec	56.4
interframe difference	0.39sec	12.0	0.62sec	18.9
affine transformation	1.57sec	48.0	3.07sec	90.9
affine & mpy & addition	2.34sec	71.7	4.35sec	133.2
convolution(3*3) ⁽³⁾	7.35sec	25.0	13.13sec	44.6

(注1) SUN : Sun-3/260 (6649Dhrystones/sec, 3.8MIPS)

NEWS : NWS-830 (4242Dhrystones/sec, 2.4MIPS)

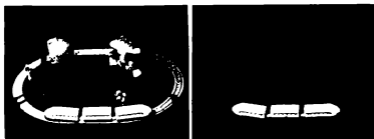
VAX11/780 (1757Dhrystones/sec) を1MIPSとして換算

(注2) Picot-processor (フレームメモリ付) 1台で可能な処理を想定し, 白黒画像(754*483画素)を対象にした。

(注3) convolutionには, Picot-processor 9台必要なので, ratio(対Picot比)は, 1/9倍して算出した。



(a) テクスチャ・マッピング



(b) 動物体抽出

図5.19 処理例

処理もビデオ処理できる。従って、処理の複雑度によって必要なプロセス数も決定される。表5.6に、主な映像信号処理を行った場合に必要なプロセス数を示す。処理例を、図5.19に示す。(a)は、球体の表面へのテクスチャマッピング、(b)は、動物体抽出の例である。

Pico-Systemの処理能力を評価するために、Pico-Systemでビデオ処理できる処理を、通常のコンピュータで処理した場合の処理時間を計測することにより比較した。表5.7に処理結果を示す。elapsは処理に要した時間であり、ratioは処理時間の対Pico-Systemのプロセッサ比である。実験では、標準TV方式の14.3MHzに標準化した画像、すなわち有効画素数が横754画素×縦483画素の白黒画像を対象とした。また、処理内容は、1フル・ム分のカメラを有するプロセッサ1台で、ビデオレート(1/30秒以内)で処理可能なものを主対象とした。評価したコンピュータは、Sun-3/260とNEWS830

である。プログラム作成に当たっては、レジスタ変数、ポインタを使うなどして、極力高速化を図った。表5. 7によれば、画像処理の内容により違いはあるが、Sun (3. 8MIPS)の最大72倍、NEWS (2. 4MIPS)の最大133倍の高速処理ができる。このことから1台のプロセッサは、最大300MIPS程度の汎用コンピュータに相当する画像処理能力を発揮できることになる。

上述のプロセッサの処理能力から計算すると、16台のプロセッサを有するクラスタは、1クラスタ当り4, 800MIPS、4クラスタでは19, 200MIPS、12クラスタでは57, 600MIPSの処理能力になる。しかし、プロセッサの全ての演算要素をいつも使うとは限らないので、これらの値は最大値ととらえるべきであろう。アフィン変換を例に取り上げる。表5. 6によれば、アフィン変換はプロセッサ15台で実現できる。処理内容の内訳は、表5. 7のaffine transformationの処理6台、read or write memoryの処理6台、addition3台であるので、ratioを加算平均して求めると、15台をフルに使ったばあいの約1/3の能力の1525MIPSに相当する能力を使用している。

5. 6 適用例

5. 6. 1 ビデオスイッチャ

プログラム可能な映像機器の番組制作現場への導入効果を検証する目的で、Pico-tec-systemをベースにして、ビデオスイッチャを実用化した。現在、番組制作用の設備としてNHK放送センターで試験運用している¹¹⁾。12クラスタ構成のシステムである。図5. 20に外観写真を、表5. 8に主な仕様を示す。

本システムは、カラーコレクタ、3列の効果列(Mix/Effect)などかなり大規模のビデオスイッチャに相当する機能を全てソフトウェアで実現している。図5. 21に従来形のビデオスイッチャによる構成法に置き換えた場合の構成図を示す。ディゾルブ、ワイプ、クロマキー、拡大縮小、デフォーカス、エッジスーパーなどの一般的な機能に加え、マット合成機能、ウェーブ効果などの新しい機能もプログラムした¹²⁾。

番組制作においては、幾何学変換などの特殊効果が多用されている。このため、システムの処理能力を上げるために、幾何学変換時に生じる折り返し除去用のフィルタボードを開発した¹³⁾。このフィルタボードは、水平24タップ、垂直9タップのFIR(Finite Impulse Response)フィルタである。プロセッサと同形状のボードとし、置き換えられるようにした。本スイッチャでは、各クラスタに1台挿入している。



左から、ホストコンピュータ、本体(A/D、D/A、クラスタ)、操作卓
 図5. 20 ビデオスイッチャの外観写真

表5. 8 ビデオスイッチャの主な仕様

映像入力	アナログコンポジット(NTSC)8系統 標本化数14.3MHz、量子化数9bit
映像出力	アナログコンポジット(NTSC)3系統 (ON-AIR, NEXT, MONITOR) 標本化数14.3MHz、量子化数10bit
内部信号形式	コンポーネント(Y, R-Y, B-Y) Y : 14.3MHz, 16bit(2's Complement) R-Y, B-Y : 7.15MHz, 16bit(2's Complement) 時間軸多変化
処理速度	70ns/pixel, 33ms/frame
クラスタ数	12
プロセッサ数	196 (うち16台は、フィルタ処理専用)
機能	ColorCorrector (色補正) ゲイン、ペダスタル、ガンマ、ヒュー、サチュレーションなど Mix/Effect 1 (効果列1) ディゾルブ、NAM、マツト合成 ワイプ(四角、円、菱形、星、ランダム、ポジション、高画速、変調波など) 特殊効果(拡大・縮小、デフォーカス、モザイク、ウェーブなど) Mix/Effect 2 (効果列2) ディゾルブ、NAM、ワイプ、クロマキー DownStream (ダウンストリーム) ディゾルブ、NAM、ワイプ、エンジススーパー、プレビュー、モニタ機能

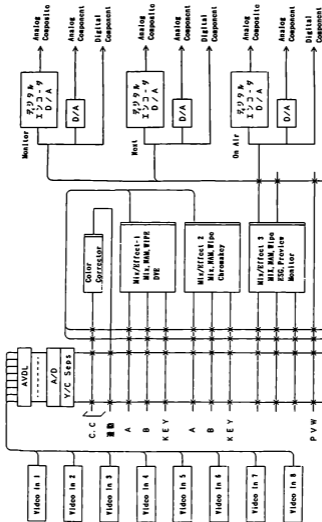


図5. 21 従来のビデオスイッチャに置き換えた場合の構成図

本システムは、ホストコンピュータ上のキーボードだけからでもコントロール可能であるが、制作スタジオでの日々の運用を考慮し、専用の操作卓（図5. 20右側）を用意した。従来型のビデオスイッチャの操作卓の操作性（マン・マシン・インタフェース）を継承すると共に、Picot-systemのもつ柔軟性の両者を整合するように作られている。大型E.L.パネルを使用しており、メニュー画面などもソフトウェアにより変更可能で拡張性に優れている。

本システムは、機能が固定の専用機器を組み合わせて作られていた従来の番組制作システムとは大きく異なるもので、導入後も新機能の追加が容易である。現在も、現場の番組制作技術者により、新しい効果が作成されており、拡張性、柔軟性が確認された。

5. 6. 2 特殊効果

演出効果を上げるために、シーンチェンジなどでは特殊効果と呼ばれる映像効果が頻繁に使われる。特殊効果は、次々と新しい効果が求められる一方で飽きやすい効果でもある。これまでも、様々な特殊効果機器が開発されてきたが、今では見向きもされない効果機器も少なくない。ハードウェアではなくソフトウェアで効果を実現できるPicot-systemは、このような一過性の強い特殊効果などの用途に適したマシンであり、ハードウェアが陳腐化することがない。

Picot-systemで実現した特殊効果の1例を、図5. 22に示す。(a)は前景の青色部分を抜き取り別の背景を合成した効果。(b)は2つの映像を波状的に切り換えてゆく効果の途中経過。(c)は正弦波を用いてアドレスを変調してマッピングした効果。(d)は乱数を使ってアドレスを変調しシーンチェンジに使った効果。(e)は非直線位相の残像効果。(f)は背景に縮小画像をはめ込んだ効果である。(a)~(d)は2クラスタ。(e)は3クラスタ。(f)は4クラスタ使用した。

5. 6. 3 画質補正

画質補正処理として、①ノイズ除去、②色補正、③フリッカ妨害軽減などをPicot-system上で試みた。

①の例では、信号成分のフレーム間相関を利用したタイプのノイズリデューサを実現した。動きの検出精度を上げるために、フレーム間差分を水平方向にフィルタをかけて雑音成分と動き成分の識別を行う手法を用いた。

②の例では、ゲイン、バズスタル、ガンマ、ルミナンス、サチューション、ヒューを変更できるカラーコレクタを実現した。このうち、非線形演算であ



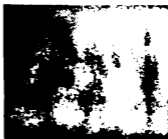
(a) クロマー



(b) ワイズ



(c) ウェーブ



(d) ハースト



(e) 映像



(f) DVE

図5. 2.2 効果例

るガンマに関しては、256種類のガンマ・テーブルを1kw単位でデータメモリ(256kw)にロードして使っており、 $\gamma=0.5\sim 2.0$ まで可変にできる。ゲインについては0~150%、ベDESTALについては $\pm 50\%$ までと、通常のカラコレクタに比べ可変範囲を大きくした。入出力がY:R-Y:B-Y(4:2:2)信号に対し、1クラスタで実現できた。

③の例では、1988年のソウル・オリンピック放送において、生じたフリッカ妨害を軽減させた。オリンピック放送においては、限られた映像伝送回線を最大限に使用するために、1つの回線に2つの映像を多重化して伝送する2画面伝送装置が通常使用されている。ソウル・オリンピック放送で使用された2画面伝送装置においては、送信側でフィールド・オフセット・サブサンプリングにより、斜め方向の解像度を落として、2つの映像を多重化する方式を採用していた。ところが、伝送回線の特性不良から、受信側で正しくリサンプルできず、サンプル位相が崩れたために、縦方向のエッジ部分に特に大きなフリッカ妨害を生じた。妨害の発生が発覚したのがオリンピック開催の直前のために改良を加える時間的余裕が無く、また既存の機器でも取り除くこともできなかった。

そこで、Pico-tec-systemを使用して妨害の軽減を試みた。現象を検討したところ、フィールド間で高域成分を除去すれば有効であることがわかり、図5.23に示す系統で除去することにした。処理は、コンポジットのままで行い、1クラスタで実現することができた。フィールド間のフリッカ妨害のために静止画像にすると効果が分かりにくい。図5.24に処理前と処理後の画像を示す。現象の検討からプログラム作成、デバッグまで3日程度で完成した。ソフトウェアによる映像信号処理の有効性が放送現場で確認される結果となった。

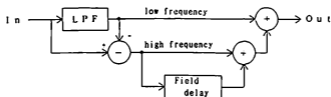


図5.23 フリッカ軽減の系統図



(a) 処理前



(一部拡大)



(b) 処理後



(一部拡大)

図5.24 フリッカ軽減例

5.6.4 動き検出

動き検出は、方式変換装置、ディジタル伝送装置などではよく使われているが、番組制作機器ではこれまであまり使われていない。画像処理的な手法を番組制作に導入すると新しい映像表現が可能となる。その1例として、動き検出を用いた映像効果を試みた。

2.4.3で述べた動き検出法のひとつであるマッチング法をPico Systemにインプリメントした場合、(2.36)式を計算するのに要する時間(フィールド数)は、

$$N = \frac{(S_1 \times N_1 + S_2) \times N_2 + S_3 + S_4}{f \times f} \quad (5.2)$$

である。ここで、

S_1 : 1点のマッチングに要するステップ数 (=14)

S_2 : マッチングに付随して行われる処理に要するステップ数 (=66)

- S.:初期化などに要するステップ数 (=22)
 S.:動ベクトル決定などに要するステップ数 (=36)
 N.:マッチング対象の画素数
 N.:検索対象となる範囲 (画素数)
 f.:演算速度すなわち標準化周波数 (=14.3MHz)
 f.:フィールド周期 (=1/60sec)

である。これを、グラフで示すと図5. 25になる。図5. 25によれば、検索範囲を水平方向-16~+15画素、垂直方向-8~+7ラインとし、マッチング点数を60画素とした場合、2フィールドでマッチング処理が完了する。従って、動ベクトル検出は、メモリへの書き込みに1フィールド、マッチング処理に2フィールド、マッチング結果に基づいてパラメークを変更する処理に1フィールドの合計4フィールド必要である。これをビデオレートで実現するためには、図5. 26に示すように4プロセッサを時間をずらして並列動作させる。図5. 27にこの方式で行った動き検出の例を示す。図5. 27 (a)は、顔を自動的に追跡して+印で表示した例である。最初に顔の部分に指定して、その後はマッチング法により自動的に追跡した。+印の代わりにモザイクを用いた例が、図5. 27 (b)である。従来より人の顔などをぼかすためにモザイクなどを用いているが、この動き検出手法を用いると最初に指定するだけでその後は自動で追跡しモザイクをかけること

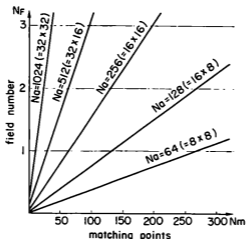


図5. 25 所要フィールド数

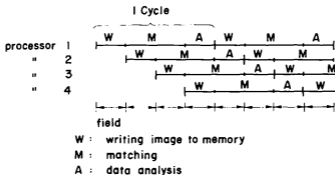


図5. 26 4プロセッサによる並列処理



(a) +印表示例



(b) モザイク表示例

図5. 27 動き追跡例

ができる。

このPicot-systemによる動き検出は、コンピュータグラフィックスで生成したキャラクターのリアルタイム制御¹⁾やマラソンランナーのピッチ計測²⁾にも使われた。

5. 6. 5 映像合成

2. 4で述べた映像合成法をPicot-system¹⁾でヒヨレットで処理させることを試みた。

(1) 大きさ・形を変えず平行移動する場合

2. 4. 4のシミュレーション結果に基づき、各エリア15画素の4エリ

アを設定して、撮影映像を用いて映像合成実験を行った。実験では、検索範囲を水平方向-16～+15画素、垂直方向-8～+7ラインとした。この場合は、図5、25より2フィールドでマッチング処理が完了することがわかる。従って、動ベクトル検出は、メモリへの書き込みに1フィールド、マッチング処理に2フィールド、マッチング結果に基づいてキーを变形する処理に1フィールドの合計4フィールド必要になるため、4プロセッサを割り当てればよい。動ベクトルは4フィールド遅れて検出されるので、映像合成のためには原画像を4フィールド遅らせる処理が必要である。この処理に加え、キー作成（入力）、検出点の指定の処理などを含めて合計3クラスを使用した。撮影映像を用いて映像合成実験を行ったところ、シミュレーションと同程度、すなわち±2画素未満の検出精度の合成映像が得られた。

(2) 回転・ズームを伴う場合

2. 4. 4のシミュレーション結果によれば、32ヶ所以上の検出エリア（各エリア16×16画素）を設定し、最小二乗法により変換パラメータを求める必要がある。ところが、この処理をそのままPicot-system上にインプリメントしても、プロセッサ台数に比べ演算量が多くビデオレート処理することは難しい。このため、回転を除いた拡大・縮小、移動する動きに限定して実時間動作を検証することにした。回転を除くと、(2. 37) (2. 38)式は、

$$X = a \cdot x + b \quad (5. 3)$$

$$Y = a \cdot y + c \quad (5. 4)$$

となり、推定するパラメータは半分の3個に減る。マッチングにおける各エリアの検出点数を60画素、検索範囲を水平方向-12～+11画素、垂直方向-5～+4ラインとすれば、図5、25よりマッチング処理が1フィールドで行える。メモリ書き込み、結果の解析を合わせ3フィールドで処理が完了する。従って、3プロセッサで1検出エリアの動ベクトル検出が行え、12プロセッサで4エリアの動ベクトル検出ができる。このやり方では、8個の方程式から3個のパラメータを最小二乗推定することになる。映像合成処理（4フィールド・ディレイも含む）、RGBからYを作る処理、キーを变形しソフトキー化する処理に必要な36プロセッサを合わせ、合計3クラスを使用した。

撮影映像を用いて映像合成実験を行ったところ、検出精度はあまり良くなかった。シミュレーション実験に比べ、演算点数がかなり少ないためと思われる。検出点の位置をテクスチャがはっきりしているところなどマッチングがとり易いところ（顔の場合は目頭など）に設定すれば、精度が若干上がるが精度向上のためには、プロセッサ数を増やす必要がある。なお、評価間数

として、距離最小を示す

$$E = \sum [(X - a \cdot x - b) + (Y - a \cdot y - c)] \quad (5. 5)$$

を用いた。

(3) 応用例—タイトル合成—

上記予備実験を踏まえ、具体的な応用例を試みた。ニュース番組などでは、

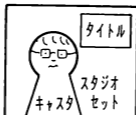


図5. 28 ニュースなどでよく見られるシーン

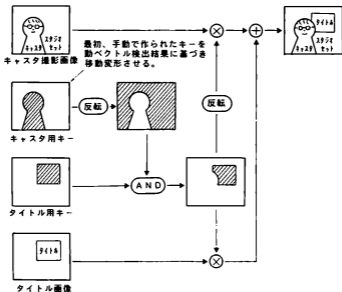


図5. 29 タイトルをキャストと背景の中間に
位置するように見せるキー合成



(a) 通常のタイトル合成



(b) 提案した方法によるタイトル合成
(頭部とタイトルの境界に注目)

図5. 30 合成画像

図5. 28に示すようなキャストとタイトルを配したシーンがよく見られる。この場合、通常キャストを撮った映像の上にタイトルを重ねているために、キャストが動くとキャストがタイトルの後方に隠れてしまう。このシーンのイメージとしては、キャストと背景の中間にタイトルが存在するのように見えるのが自然である。これを実現するために、図5. 29に示すようにキャスト用のキーとタイトル用のキーを組み合わせて合成する。このうち、キャスト用のキーを、動ベクトルを検出し移動・変形して作れば、キャストが動いても、キャストの後ろにタイトルがあるように見せることができる。本手法をP i c t o r - s y s t e mによりビデオレートで処理した例を図5. 30に示す。2. 2で仮定した動きに必ずしも適合しない人間が抽出対象であるため、キャストを切り出すキーが完全には得られない。この誤差を補うために、キャストとタイトルの境界をハードキーではなくソフトキーで処理した。これらの要因によりキャストとタイトルの境界部が曖昧であるが、見タイトルがキャストと背景の中間に位置しているように見せることができた。

画像・映像信号をプログラムによりビデオレートで処理できる実時間映像信号処理システムについて述べた。本論文の要旨をまとめると、以下のようになる。

第1章：研究の背景と意義について述べた。

第2章：放送用の映像信号処理の特質について考察を加えた。特に、幾何学変換、効果波形発生、映像合成を取り上げ、放送用映像信号処理装置が具備すべき演算機能、演算精度について明らかにした。

第3章：プログラム可能なビデオレート映像信号処理システムのアーキテクチャについて検討した。放送用の実時間処理システム用に、マルチプロセッサ型の並列処理方式、各プロセッサが処理に必要なデータを全て確保しておく分散型のメモリ構成法、データに同期情報を付加した映像データによるプロセッサ間同期方法、画面上に乱れを生じないプログラムロード切り替え方式などについて提案した。

第4章：第2章、第3章の検討結果を踏まえて、実験システムRTVPを開発し、各種映像信号処理を適用して評価した。RTVPは、プログラム可能なビデオレート映像信号処理システムの実現性の検証を行うために試作した実験システムのため、標準化周波数3.58MHzの白黒映像信号2系統しか扱えないものであったが、映像転送路、分散型メモリ構成法などにより、プログラム可能なビデオレート映像信号処理システムの実現の目処が得られるとともに、実用化する上での問題点、改善すべき点などについての知見が得られた。

第5章：実験システムRTVPの試作によって得られた知見に基づき、14.3MHzの速度で複数のカラー動画をビデオレート処理可能なマルチプロセッサであるPicot-systemを開発した。この際、小型化のためにプロセッサLSIとネットワークLSIの2種のLSIを開発した。さらに、Picot-systemをベースにビデオスイッチャや画質補償装置などを開発した。Picot-systemは、動画像・映像信号処理のアルゴリズムの研究に使用されているばかりでなく、NHKスペシャル(NHK)、ソウルオリンピック(NHK)、大阪国際女子マラソン(関西TV)などの番組にも活用された。また、Picot-systemを構成する2種のLSIもそれぞれPicot-system以外の機器に組み込まれて使用されている。

研究を開始した当初は、映像信号をプログラム可能な装置でビデオレート処理することは試みられていなかったが、現在は各所で行われるようになって

てきている。研究の方向性に誤りがなかったことが確認された形となった。現在多くのシステムがパーソナルコンピュータのように各所に使われるようなレベルにまで達していないが、本研究は先駆けとしても意味があったと考える。

映像信号をプログラムで処理することの意義は、ソウルオリンピック放送などでの使用で確認されたが、まだ放送現場で活用されているとは言い難い。ハードウェア信仰の強い現場での意識の変革も必要であるが、欲しい機能が簡単に実現できる開発環境を整えることも必要である。特に、緊急を要する仕事が多い番組制作現場では、マンマシンインタフェースの充実が重要であろう。

現在、HDTVレートまで処理可能なシステムの開発に着手している。今後は、映像ワークステーション、すなわち標準TV、HDTVなどあらゆる画像・映像信号統合的にビデオレート処理でき、かつワークステーションのようにウィンドウやマウスなどを使って手軽に扱えるヒューマンインタフェースの優れたコンパクトなシステムの構築が必要であろう。

【謝辞】

本論文をまとめるに当たって、ご指導いただいた京都大学工学部応用システム科学教室英保茂教授、八村広三郎助教授に深謝いたします。

本研究はNHK放送技術研究所において行ったものである。本研究を進めるに当たり当初からご指導いただいたNHK放送技術研究所画像研究部村上敬之助主任研究員、榎並和雅主任研究員に感謝します。また、Picot-systemの開発を共同で進めていただいたNHK放送技術研究所画像研究部福井一夫主任研究員、TV方式研究部矢島亮一氏、ご協力いただいた(株)東芝小向工場の佐々木信之主務、星野浩二氏、春川和弘氏、小暮勝氏、高田龍一郎氏、金野雄二氏、船橋清一郎氏、同総合研究所高比呂志研究主務、同半導体事業本部山田秀喜氏をはじめ関係各位に感謝いたします。

本研究を進めるに当たり有益な御助言いただいたNHK放送技術研究所香沢淳之助主幹研究員、画像研究部二宮佑一郎氏、藤原正雄前副部長、野口英男副部長、同視覚情報研究部三橋哲雄主任研究員、矢野澄男研究員、同物性素子研究部小林希一主任研究員、明智和幸研究員、同技術局大場吉延前副部長、金井清福前副部長、川野順一郎氏、同放送技術局三上繁実氏、鈴木彰氏、武藤光明氏、指田忠男氏をはじめ関係各位に感謝いたします。また、日頃御討論いただく画像研究部の皆様にも感謝いたします。

本研究の機会を与えて頂いたNHK放送技術研究所杉本昌徳前所長、泉竹博所長、吉川重夫前次長、西沢台次担当局長、吉野武彦担当局長、横山克哉前主任研究員(現、日本ビクター)、石田順一前次長(現、NHK-E)に深謝いたします。

[参考文献]

- (1) 木戸出, 坂上: 'パイプライン方式と完全並列方式が増えた最近の画像処理装置', 日経エレクトロニクス, 1982, 07, 19号, p. 179-212 (1982)
- (2) 小特集: '最近のデジタルテレビジョン技術', T V誌, Vol. 36, No. 1, pp. 2-34 (1982)
- (3) 特集: 'D S P (デジタル信号処理プロセッサ)', 情報処理, Vol. 30, No. 1, pp. 1290-1375 (1989)
- (4) 横並, 田中, 村上: '実時間映像信号処理システムのアーキテクチャ', 信学論, Vol. J68D, No. 4, pp. 885-892 (1985)
- (5) 八木, 矢島, 横並, 他: '実時間映像信号処理L S I - P i c o t のアーキテクチャ', 信学論, Vol. J72C-II, No. 5, pp. 346-353 (1989)
- (6) 八木, 矢島, 横並, 他: '実時間動画画像処理システムPicot-system', 情報研報, CV58-2 (1989)
- (7) 川野, 八木, 福井, 他: '番組制作におけるP I C O Tの実用化', 1989年T V大全, 14-1 (1989)
- (8) 八木: 'デジタル信号処理デバイスと基本機能回路', N H K - E S セミナ [テレビジョン信号・デジタル処理技術] 資料, p. 68-101 (1990)
- (9) 'Encoding Parameters of Digital Television for Studios', CCIR Rec. 601 (1982)
- (10) '1125/60高精細度テレビジョン方式スタジオ規格', 放送技術開発協議会, BTA S-001 (1986)
- (11) [日本放送協会編: '放送におけるデジタル技術', p. 163-219, 日本放送出版協会 (1982)
- (12) N H K 放送技術研究所: 'デジタルテレビ技術', p. 259-277, 日本放送出版協会 (1990)
- (13) 金次: '映像の幾何学変換用フィルタBLIF', 第17回画像工学コンファレンス, pp. 237-240 (1986)
- (14) 飯田, 岡田, 西村, 他: '実時間3次元特殊効果装置', T V技術, IPA87-3 (1987)
- (15) 八木, 福井, 横並, 他: '映像信号処理L S I - P i c o t の演算回路に関する一検討', T V誌, Vol. 44, No. 9, pp. 1269-1278 (1990)
- (16) Pratt: 'Digital image processing', p. 93-120, Wiley-Interscience Publication (1987)
- (17) 谷内田編: 'コンピュータビジョン', p. 51-54, 丸善 (1990)

- (18) 櫻井, 八巻, 八木, 他: '画像信号の幾何学変換のための補間フィルタと画像に関する一考察', 信学論, Vol. J69D, No. 11, pp. 1617-1623 (1986)
- (19) 日下, 湯山: 'テレビジョンシステム評価用デジタル標準画像', T V誌, Vol. 39, No. 10, pp. 160-163 (1985)
- (20) 和田, 大山, 今井編: '感覚知覚心理学ハンドブック', p.206-223, 誠信書房 (1969)
- (21) 二宮: '画像信号処理と視覚特性', テレビ誌, Vol. 40, No. 4, pp. 280-286 (1986)
- (22) 日本放送協会編: 'テレビジョン番組制作技術', p.81-107, 日本放送出版協会 (1983)
- (23) 日本放送協会編: 'テレビジョン番組制作技術', p.99-102, 日本放送出版協会 (1983)
- (24) 日本放送協会編: 'テレビジョン番組制作技術', p.128-133, 日本放送出版協会 (1983)
- (25) 吉田, 梶井: 'ビデオマットによる動画合成技術', T V技報, TEBS88-19 (1988)
- (26) 谷内田編: 'コンピュータビジョン', p.167-188, 丸善 (1990)
- (27) M.Hoetter: 'Differential estimation of the global motion parameters zoom and pan', Signal Processing, Vol. 16, No. 3, pp. 249-265 (1989)
- (28) 吹抜: '画像のデジタル信号処理 (増補版)', p. 221-227, 日刊工業新聞社 (1985)
- (29) 二宮: 'フレーム間符号化における動き補正', 信学技報, IE78-6 (1978)
- (30) 二宮, 黨, 杉山: 'M U S E用動きベクトル検出装置', T V技報, TEBS103-5 (1985)
- (31) 情報処理学会編: '情報処理ハンドブック', p. 123, オーム社 (1989)
- (32) 櫻井, 井沢, 平林, 鈴木: '汎用動画画像シミュレータ', 信学技報, IE87-73 (1987)
- (33) 八木, 田中, 櫻井: '初期画像を補助手段とした動きベクトル検出法の検討', 昭63信学春季全大, D-517 (1988)
- (34) 田中, 八木, 櫻井: '動きベクトルを用いた映像合成手法', 1988年 T V全大, 19-6 (1988)
- (35) 田中, 八木, 櫻井: '大きさ・向きを変える物体の動きベクトル検出法の検討', 昭64信学春季全大, D-109 (1989)
- (36) 西川, 三宮, 沢木: '最適化', p. 83-93, 岩波書店 (1982)
- (37) 八木, 田中, 櫻井: '動物体抽出法に関する一検討', 1990年 T V全大,

- (38) 八木, 田中, 櫻並: '動き抽出法に関する一検討', T V 技報, ICS91-24, AIPS91-4 (1991)
- (39) 坂上, 木戸出: 'イメージプロセッサの最近の動向', 信学誌, Vol. 67, No. 1, pp. 90-98 (1984)
- (40) 前田: '画像処理マシン', 情報処理, Vol. 28, No. 1, pp. 19-26 (1987)
- (41) 吉田, 長谷部: '高速画像処理技術', T V 誌, Vol. 42, No. 3, pp. 241-246 (1988)
- (42) 櫻並, 八木: '映像信号の汎用リアルタイム処理', T V 誌, Vol. 43, No. 12, pp. 1314-1319 (1989)
- (43) R. Loughheed, B. McCubbrey, S. Sternberg: 'CYTOCOMPUTER: Architecture for parallel image processing', Workshop Picture Data Description and Management, pp. 281-286 (1980)
- (44) 直井, 古明地, 太田, 他: '構造可変型ビデオレートカラー画像処理システム「草駟天」', 信学論, Vol. J73D-II, No. 10, pp. 1751-1760 (1990)
- (45) 天満, 溝口, 花木: '可変バイライン方式の画像処理プロセッサ T I P - 1', 信学論, Vol. J68D, No. 4, pp. 853-860 (1985)
- (46) K. Luetjen, P. Gemmer, H. Ischen: 'FLIP: A flexible multiprocessor system for image processing', 5th ICPR, pp. 326-328 (1980)
- (47) K. E. Batcher: 'Design of a massively parallel processor', IEEE Tran. Com, Vol. C-29, No. 9, pp. 836-840 (1980)
- (48) M. J. B. Duff: 'Review of the CLIP4 image processing system', AFIPS Conference, National Computer Conference, pp. 1055-1060 (1984)
- (49) 近藤, 杉山, 中島: '2次元アレープロセッサ (A A P 2) とプログラミング言語', 信学論, Vol. J71D, No. 8, pp. 1399-1406 (1988)
- (50) 木戸出, 白男川, 角谷, 他: '高速画像処理装置 T O S P I X - II', 第15回画像工学コンファレンス論文集, pp. 133-136 (1984)
- (51) 福島, 小林, 平沢, 他: '画像処理 V L S I - Image Signal Processor のアーキテクチャ', 信学論, Vol. J66C, No. 12, pp. 959-966 (1983)
- (52) 森, 丸山, 青野, 他: 'リアルタイム画像プロセッサ RISP - II', 信学技報, SSD86-96 (1986)
- (53) K. Inagaki, T. Kato, T. Hiroshima, et al: 'MACSYM: A hierarchical parallel image processing system for event-driven pattern understanding of documents', Pattern Recognition, Vol. 17, No. 1, pp. 85-108 (1984)

- (54) T. Kushner, A. Y. Wu, A. Rosenfeld: 'Image processing on ZMOB',
Computer Architecture for Pattern Analysis and Image Database
Management, pp. 88-95 (1981)
- (55) 長谷部, 加藤, 伊藤, 他: 'マルチプロセッサ型画像処理システム
S I P S', 情処研報, CV39-5 (1985)
- (56) 政木, 堀, 内野, 他: 'リアルタイム画像処理システム R A P I D',
情処研報, CV50-2 (1987)
- (57) I. Inmitani, H. Harasaki, T. Nishitani, et al: 'A real-time video
signal processor suitable for motion picture coding
applications', IEEE Trans., Vol. CAS-36, No. 10, pp. 1259-1266 (1989)
- (58) 高橋: '並列処理のためのプロセッサ結合方式', Vol. 23, No. 3,
pp. 201-209 (1982)
- (59) 黒川, 相磯: '並列処理の諸問題 - 結合方式', 情報処理, Vol. 27, No. 9,
pp. 1005-1021 (1986)
- (60) 中川, 小林, 相磯: 'データ駆動型離散系シュミレータ K D S S - I',
信学論, Vol. J65D, No. 3, pp. 386-393 (1982)
- (61) 小畑, 金田, 前川: 'ブロードキャストメモリ結合形マルチマイクロプロ
セッサシステムの試作', 情処学論, Vol. 24, No. 3, pp. 351-356 (1983)
- (62) 櫻並, 八木, 村上: '2段階時間軸処理方式の提案と実時間映像処理シス
テム R T V P への適用', 昭和60信学部門全大, 171 (1985)
- (63) 田中, 櫻並, 村上: '実時間映像処理システム R T V P
制御とソフトウェア', 昭和60信学全大, 1276 (1985)
- (64) 八木, 櫻並, 村上: '実時間映像処理システム (R T V P) による
各種画像処理', 昭和60信学部門全大, 172 (1985)
- (65) 久野: '画像処理・画像理解におけるベンチマーク', 情報処理,
Vol. 31, No. 3, pp. 343-351 (1990)
- (66) K. Preston: 'The abingdon cross benchmark survey', IEEE Computer,
Vol. 22, No. 7, pp. 9-18 (1989)
- (67) 八木, 福井, 櫻並, 他: '動画像を実時間で処理するシステムにおける処理
遅れと同期の問題について P i c o t - s y s t e m の対応法',
昭64信学春季全大, D-264 (1989)
- (68) 八木, 矢島, 櫻並, 他: '実時間映像信号処理 L S I - P i c o t',
信学技報, IC088-35 (1988)
- (69) N. Yagi, R. Yajima, K. Enami, et al: 'Chip set for real-time video
signal processing', IAPR Workshop on Computer Vision,
pp. 237-240 (1988)

- (70) 斉藤, 上田, 佐々木: 'プログラム動作を行うプロセッサによる映像信号処理 (ワイズキー発生器への適用)', 放送技術, Vol. 44, No. 3, pp. 217-222 (1991)
- (71) 特集: '放送用大規模映像・音声分配システム', 東芝レビュー, Vol. 45, No. 12, pp. 927-956 (1990)
- (72) 高橋, 萬: 'NHK放送センタデジタル切替, 分配システム', TV技報, BF091-2 (1991)
- (73) 磯村, 松尾, 広田: 'J S Bのデジタル分配システム', TV技報, BF091-13 (1991)
- (74) M. Ott, 櫻並, 羽鳥, 他: 'P i c P E n - 実時間映像信号処理装置 P i c o tのためのプログラミング環境', TV誌, Vol. 44, No. 11, pp. 1570-1578 (1990)
- (75) M. Ott, N. Yagi, K. Fukui, et al: 'Timing in synchronous system', 1989年TV全大, 14-4 (1989)
- (76) N. Yagi, K. Fukui, K. Enami, et al: 'Real-time video signal processing system for dynamic images', SPIE Vol. 1199, Proc. Visual Communications and image processing IV, pp. 866-877 (1989)
- (77) 八木, 福井, 櫻並, 他: '実時間動画画像処理システム P i c o tの処理能力評価', 1989信学秋季全大, D-107 (1989)
- (78) 三上, 櫻並, 八木, 他: 'P i c o tシステムを用いたデジタルビデオスイッチャの開発', 1988年TV全大, 19-2 (1988)
- (79) 佐々木, 小暮, 富田, 他: 'P I C O Tを用いた新しいスイッチャ効果の開発', 1989年TV全大, 14-2 (1989)
- (80) 八木, 福井, 櫻並, 他: '実時間動画画像処理システム P i c o tとその応用', 画像電子学会研究会予稿, 89-01-06 (1989)
- (81) 指田, 武藤, 川野: 'P i c o tの現場での運用', TV技報, BF089-6 (1989)
- (82) 林, 八木, 福井, 他: 'P i c o tによるノイズ軽減処理', 1989年TV全大, 14-3 (1989)
- (83) 林, 井上, 藤原, 他: '時間軸圧縮による映像2チャンネル同時伝送装置', TV誌, Vol. 35, No. 8, pp. 631-636 (1981)
- (84) 末岡, 大場, 平川: '2画面伝送装置', TV誌, Vol. 43, No. 2, pp. 132-134 (1989)
- (85) 八木, 福井, 櫻並, 他: 'CGキャラクターの撮影画像を用いた実時間操作', 1990年信学秋季全大, D-416 (1990)
- (86) 佐々木, 富田, 並川, 他: 'P i c o tを用いたマラソンランナーのピッチ,

ストライド計測システムの開発*, 1990年TV全大, 12-3 (1990)
1990年TV全大, 12-4 (1990)

[発表論文等リスト]

[学会論文]

- ・ 榎並, 田中, 村上: '実時間映像信号処理システムのアーキテクチャ',
信学論, Vol. J68D, No. 4, pp. 885-892 (1985)
(上記文献中の著者名'田中'は, 八木の旧姓)
- ・ 榎並, 八巻, 八木, 他: '画像信号の幾何学変換のための補間フィルクと両質に
関する一考察', 信学論, Vol. J69D, No. 11, pp. 1617-1623 (1986)
- ・ K. Murakami, K. Enami, N. Yagi: 'A proposed universal signal-processing
system', SMPTE Journal, Vol. 96, No. 6, pp. 527-531 (1987)
- ・ K. Enami, N. Yagi, K. Murakami: 'Real-time video signal processor',
SMPTE Journal, Vol. 96, No. 12, pp. 1158-1165 (1987)
- ・ 榎並, 八木, 矢島, 他: '汎用映像信号処理システム P i c o t システム',
T V 誌, Vol. 43, No. 2, pp. 178-186 (1989)
- ・ 八木, 矢島, 榎並, 他: '実時間映像信号処理 L S I - P i c o t の
アーキテクチャ', 信学論, Vol. J72C-II, No. 5, pp. 346-353 (1989)
- ・ N. Yagi, R. Yajima, K. Enami, et al: 'An architecture of real-time video
signal processing LSI-Picot', Electronics and Communications
in Japan, Part 2, Vol. 73, No. 6, pp. 70-78 (1990)
- ・ 八木, 福井, 榎並, 他: '映像信号処理 L S I - P i c o t の演算回路に
関する一検討', T V 誌, Vol. 44, No. 9, pp. 1269-1278 (1990)
- ・ M. Ott, ..., 八木, 他: 'P i c P E n - 実時間映像信号処理装置 P i c o t の
ためのプログラミング環境', T V 誌, Vol. 44, No. 11, pp. 1570-1578 (1990)
- ・ N. Yagi, K. Fukui, K. Enami, et al: 'Programmable real-time video signal
processing system', SMPTE Journal, Vol. 100, No. 10 (1991)
- ・ 八木, 田中, 榎並: '映像合成のための動きベクトル検出法に関する一検討',
T V 誌, Vol. 45, No. 10 (1991)

[国際会議]

- ・ K. Enami, N. Yagi, K. Murakami: 'Architecture of a real-time video
signal processor', ICASSP '86, pp. 793-796 (1986)
- ・ K. Enami, N. Yagi, K. Murakami: 'Universal video signal processing
system', 128th SMPTE technical conference (1986)
- ・ N. Yagi, R. Yajima, K. Enami, et al: 'Chip set for real-time video signal
processing', IAPR Workshop on Computer Vision, pp. 237-240 (1988)
- ・ N. Yagi, K. Fukui, K. Enami, et al: 'Real-time video signal processing
system for dynamic images', SPIE Vol. 1199, Proc.

- Visual Communications and image processing IV, pp.866-877(1989)
- ・ K. Fukui, K. Enami, N. Yagi, et al: "Picot: picture computer",
132nd SMPTE technical conference(1990)

【解説】

- ・ 櫻並, 八木: "映像信号の汎用リアルタイム処理",
TV誌, Vol. 43, No. 12, pp. 1314-1319 (1989)
- ・ 八木, 櫻並: "映像機器構成法の新しい流れ",
放送技術, Vol. 44, No. 4, pp. 374-381 (1991)
- ・ 櫻並, 八木: "並列リアルタイム映像信号処理システム",
信学誌, Vol. 74, No. 6, pp. 573-576 (1991)
- ・ 八木, 福井, 櫻並, 他: "映像コンピュータ Picot", NKK 技研 R & D,
No. 16 (1991)

【学会研究会】

- ・ 櫻並, 八巻, 八木, 他: "画像補間フィルタと画質の評価",
画像電子学会研究会予稿, 86-01-01 (1986)
- ・ 八木, 矢島, 櫻並, 他: "実時間映像信号処理 LSI-Picot",
信学技報, ICD88-35 (1988)
- ・ 八木, 矢島, 櫻並, 他: "実時間動画像処理システム
Picot-system", 情報研報, CV58-2 (1989)
- ・ 八木, 福井, 櫻並, 他: "実時間動画像処理システム Picot とその応用",
画像電子学会研究会予稿, 89-01-06 (1989)
- ・ 佐々木, 小暮, 八木, 他: "汎用映像処理システム Picot の開発",
TV 技報, BF089-5 (1989)
- ・ 八木, 田中, 櫻並: "動き抽出法に関する一検討",
TV 技報, ICS91-24, AIPS91-4 (1991)
- ・ 櫻並, 福井, 八木, 他: "実時間映像信号処理装置 Picot", TV 技報,
IPC91-13, AIPS91-24 (1991)

【学会全国大会】

- ・ 櫻並, 田中, 村上: "実時間映像処理システム R T V P
ハードウェア構成", 昭和60信学全大, 1277 (1985)
 - ・ 田中, 櫻並, 村上: "実時間映像処理システム R T V P
制御とソフトウェア", 昭和60信学全大, 1276 (1985)
 - ・ 村上, 櫻並, 田中, 他: "放送における映像処理汎用システムの提案と試作",
昭60TV全大, 8-5 (1985)
- (上記3文献中の著者名"田中"は, 八木の旧姓)
- ・ 八木, 櫻並, 村上: "実時間映像処理システム (R T V P) による

- 各種画像処理', 昭和60信学部門全大, 172(1985)
- ・ 櫻並, 八木, 村上: '2段階時間軸処理方式の提案と実時間映像処理システム R T V P への適用', 昭和60信学部門全大, 171(1985)
 - ・ 櫻並, 八巻, 八木, 他: '画像補完法と画質に関する一考察', 昭61信学全大, 1240(1986)
 - ・ 櫻並, 八木, 矢島, 他: '実時間映像信号処理システムの提案 P i c o t システム', 昭63信学全大, D-201(1988)
 - ・ 八木, 矢島, 櫻並, 他: '実時間映像信号処理 L S I - P i c o t', 昭63信学全大, D-202(1988)
 - ・ 八木, 矢島, 櫻並, 他: '実時間映像信号処理 L S I - 制御機構', 昭63信学全大, D-203(1988)
 - ・ 櫻並, 八木, 矢島, 他: '実時間映像信号処理用並列プロセッサ結合 L S I', 昭63信学全大, D-204(1988)
 - ・ 矢島, 八木, 櫻並, 他: '実時間映像信号処理システム P i c o t における処理方式', 昭63信学全大, D-205(1988)
 - ・ 八木, 田中, 櫻並: '初期画像を補助手段とした動きベクトル検出法の--検討', 昭63信学春季全大, D-517(1988)
 - ・ 八木, 矢島, 櫻並, 他: '実時間映像信号処理装置 P i c o t システムの放送への応用', 1988年 T V 全大, 19-1(1988)
 - ・ 三上, 櫻並, 八木, 他: 'P i c o t システムを用いたデジタルビデオスイッチャの開発', 1988年 T V 全大, 19-2(1988)
 - ・ 櫻並, 八木, 矢島, 他: 'P i c o t システムを用いたデジタルビデオスイッチャの制御', 1988年 T V 全大, 19-3(1988)
 - ・ 矢島, 八木, 櫻並, 他: 'P i c o t システムにおけるソフトウェア開発', 1988年 T V 全大, 19-4(1988)
 - ・ 田中, 八木, 櫻並: '動きベクトルを用いた映像合成手法', 1988年 T V 全大, 19-6(1988)
 - ・ 八木, 福井, 櫻並, 他: '動画像を実時間で処理するシステムにおける処理遅れと同期の問題について P i c o t - s y s t e m の対応法', 昭64信学春季全大, D-264(1989)
 - ・ 田中, 八木, 櫻並: '大きさ・向きを変える物体の動きベクトル検出法の--検討', 昭64信学春季全大, D-109(1989)
 - ・ 川野, 八木, 福井, 他: '番組制作における P I C O T の実用化', 1989年 T V 全大, 14-1(1989)
 - ・ 佐々木, …, 八木, 他: 'P I C O T を用いた新しいスイッチャ効果の開発', 1989年 T V 全大, 14-2(1989)

- ・ 林, 八木, 福井, 他: 'P i c o tによるノイズ軽減処理',
1989年TV全大, 14-3(1989)
- ・ M. Ott, N. Yagi, K. Fukui, et al: 'Timing in synchronous system',
1989年TV全大, 14-4(1989)
- ・ 八木, 福井, 櫻並, 他: '実時間動画処理システムP i c o tの
処理能力評価', 1989信学秋季全大, D-107(1989)
- ・ 福井, 八木, 櫻並: 'P i c o tによるステレオ画像処理とその応用例',
1990年TV全大, 12-5(1990)
- ・ 佐々木, 八木, 福井, 他: 'P i c o tを用いた並列処理の各種例と分類',
1990年TV全大, 12-6(1990)
- ・ 八木, 田中, 櫻並: '動物体抽出法に関する一検討',
1990年TV全大, 13-8(1990)
- ・ 八木, 福井, 櫻並, 他: 'CGキャラクタの撮影画像を用いた実時間操作',
1990年信学秋季全大, D-416(1990)
- ・ 八木, 櫻並: '動ぼけを考慮した映像合成法',
1991年信学春季全大, D-454(1991)
- ・ 猪山, 福井, 八木, 他: '両眼立体視によるリアルタイム映像抽出手法',
1991年信学春季全大, D-493(1991)

1. プログラムの形式

- ・プログラムの形式は以下の通り。

```

(例) picot1[           `プロセッサの指定
      'MODULE1':       `モジュール名の指定 (''の中,24char以内)
      dlc(0):          `以下, 実行文
      ldr(hint1,0):
      ldr(hincl,1,0):
      mgs(greg1,hcmp2):
      while(svd) out(clr):
loop:
      while(shd) au(hint1/reg1,hincl,hw0,hoff):
      au(hau/reg1,hincl,hw0,hoff):
      while(!shd) {
          while(!hau2) out(la):
          while( hau2) out(lb):
      }
      if(!svd) goto loop:
      return:
]           `プロセッサ1のプログラムMODULE1の終り
picot2[     `プロセッサの指定
      'MODULE2':   `モジュール名の指定 (''の中,24char以内)
      .....       `以下, 実行文
      .....
      .....
]           `プロセッサ2のプログラムの終り
    
```

このように、ひとつのソースファイル中に複数のプログラムが記述できる。

例えば、picot1が複数存在しても、かまわない。

- ・コメント、モジュール名以外は、小文字を使用する。
- ・各行の `以降は、コメントが記述できる。

2. 制御命令

- ・制御命令には以下のものがある。

goto LABEL

LABELのついた文へ無条件にジャンプする。

ラベルは、label:のように記述する。

(例) loop:

.....

goto loop: `loopにジャンプする。

if (FLAG) goto LABEL

FLAGがonなら、LABELのついた文にジャンプする。

FLAG

shd: システムのHD, svd: システムのVD, sfd: システムのFP

ahd: 入力IAのHD, avd: 入力IAのVD, afd: 入力IAのFP

bhd: 入力IBのHD, bvd: 入力IBのVD, bfd: 入力IBのFP

z: ALUのゼロ・フラッグ, pz: ALUのプラス/ゼロ・フラッグ

hauf1: HAUのフラッグ1, hau2: HAUのフラッグ2

vaufl: VAUのフラッグ1, vau2: VAUのフラッグ2

上記の条件FLAGの否定として, !が使える。(例: !shd, !svdなど)

(例) loop:

```
.....
    if (shd) goto loop: 'shdのときloopにジャンプする.
while (FLAG) FUNC_STATEMENT
FLAGがonの間, FUNC_STATEMENTが実行される.
FLAG
    shd: システムのHD, svd: システムのVD, sfd: システムのFP
    ahd: 入力IAのHD, avd: 入力IAのVD, afd: 入力IAのFP
    bhd: 入力IBのHD, bvd: 入力IBのVD, bfd: 入力IBのFP
    z: ALUのゼロ・フラッグ, pz: ALUのプラス/ゼロ・フラッグ
    haufl: HAUのフラッグ1, hau2: HAUのフラッグ2
    vaufl: VAUのフラッグ1, vau2: VAUのフラッグ2
```

上記の条件FLAGの否定として, !が使える。(例: !shd, !svdなど)

(例) while (shd) alu(...): 'shdの間 alu(...) を実行する.

while (FLAG) {FUNC_STATEMENTS}

FLAGがonの間, FUNC_STATEMENTSが実行される.

while文は, 2重の入れ子まで許される.

FLAG

```
shd: システムのHD, svd: システムのVD, sfd: システムのFP
ahd: 入力IAのHD, avd: 入力IAのVD, afd: 入力IAのFP
bhd: 入力IBのHD, bvd: 入力IBのVD, bfd: 入力IBのFP
z: ALUのゼロ・フラッグ, pz: ALUのプラス/ゼロ・フラッグ
haufl: HAUのフラッグ1, hau2: HAUのフラッグ2
vaufl: VAUのフラッグ1, vau2: VAUのフラッグ2
```

上記の条件FLAGの否定として, !が使える。(例: !shd, !svdなど)

(例) while (svd) { 'svdの間 { から } を実行

```
.....
    alu(...);
    .....
}
while (svd) {
    .....
    while (shd) alu(...): 'shdかつsvdの間 alu(...) を実行
    .....
}
while (svd) {
    .....
    while (shd) { 'shdかつsvdの間 { から } を実行する.
    .....
}
}
return
```

ルーチンから抜け出る. インストラクションセットのRESTARTを実行する.

(例) return: 'RESTART命令が実行される.

3. ファンクション命令

nop

何もしない

out (MPX)

出力を制御する関数

MPX

la : 外部入力IA	lb : 外部入力IB
alu : ALUの出力	aluf : ALUのフラッグ
mpy : 乗算器の出力	dm : Date Memoryの出力
hau : HAUの出力	vau : VAUの出力
greg : 汎用レジスタ	clr : 0
h : H D	V : V D
VH : V DとH D	FV : F PとV D
FVH : F PとV DとH D	

(例) out(alu): `出力0AにALUの出力を出力する。

alu (INA, INB, FUNC, FL, L/H)

算術論理演算器を制御する関数

INA 入力選択

la : 外部入力IA	lb : 外部入力IB
mpy : 乗算器の出力	hau : HAUの出力
acons : レジスタ	dm : Date Memoryの出力

INB 入力選択

la : 外部入力IA	alu : ALUの出力
mpy : 乗算器の出力	vau : VAUの出力
dm : Date Memoryの出力	

FUNC 演算機能

clr : クリア (all 0)	prt : プリセット (all 1)
sba : b-a	sab : a-b
add : a+b	xor : a`b
or : a b	and : a&b
max : max(a,b)	min : min(a,b)
absa : a	absb : b
pa : a	pb : b
ma : -a	mb : -b

FL フラッグ

z : zeroフラッグ pz : plus/zeroフラッグ

L/H 演算結果のラッチ、ホールド

lch : ラッチ hld : ホールド

(例) alu(lb, la, add, z, lch): `LB+LAを計算し、ラッチする。
`zeroフラッグを選択。

mpy (INA, INB, FMT, L/H)

乗算器を制御する関数

INA 入力選択

la : 外部入力IA	lb : 外部入力IB
alu : ALUの出力	dm : Date Memoryの出力

INB 入力選択

la : 外部入力IA	lb : 外部入力IB
acons : レジスタ	dm : Date Memoryの出力

FMT 小数点位置

lsl: 演算モード1 msl: 演算モード2

L/H 演算結果のラッチ、ホールド

lch: ラッチ hld: ホールド

(例) mpy (la, lb, lsl, lch): LA×LBをモード1で計算し、ラッチする。
au (HFUNC, HINC, HMODE, HAREP, VFUNC, VINC, VMODE, VAREP)

算術演算器HAUとVAUを制御する関数

HFUNC 演算機能

hnop : 何もしない

hreg1/reg2: hreg1←hreg2

hint1/reg1: hint1←hreg1

hint2/reg1: hint2←hreg1

hau/reg1 : hreg1 + HINC → hreg1

hau/reg2 : hreg1 + HINC → hreg2

hau/reg12: hreg1 + HINC → hreg1, hreg2

HINC 増分値レジスタ

hinc1: HINC1 hinc2: HINC2

HMODE リプレイス・モード

hm1: モード1 hm2: モード2 hm3: モード3 hm4: モード4

HAREP 自動リプレイス

hoff: マニュアル・モード hon: オート・モード

VFUNC 演算機能

vnop : 何もしない

vreg1/reg2: vreg1←vreg2

vint1/reg1: vint1←vreg1

vint2/reg1: vint2←vreg1

vau/reg1 : vreg1 + VINC → vreg1

vau/reg2 : vreg1 + VINC → vreg2

vau/reg12: vreg1 + VINC → vreg1, vreg2

VINC 増分値レジスタ

vinc1: VINC1 vinc2: VINC2

VMODE リプレイス・モード

vm1: モード1 vm2: モード2 vm3: モード3 vm4: モード4

VAREP 自動リプレイス

voff: マニュアル・モード von: オート・モード

(例) au(hau/reg1, hinc1, hm0, hoff, vint1/reg1, vinc1, vm1, von):

 HREG1←HREG1+HINC1, マニュアルモード0を選択。

 VREG1←VINT1, 自動リプレイスモード1を選択

dm(ADR, DAT, R/W)

データメモリのアクセスを制御する関数

ADR

la : 外部入力IA

lb : 外部入力IB

alu : ALEの出力

mpy : 乗算器の出力

hau : HAUの出力

vau : VAUの出力

lmc : ラインディレイ

lvau : 補間パラメータ

lv9 : VAU(9bit), HAU(9bit)

lab9 : IB(9bit), IA(9bit)

lv10 : VAU(10bit), HAU(10bit)

lab10 : IB(10bit), IA(10bit)

lv11 : VAU(11bit), HAU(11bit)

lab11 : IB(11bit), IA(11bit)

hvl2: VAU(12bit),HAU(12bit) lab2: IB(12bit),IA(12bit)
DAT
la: 外部入力IA lb: 外部入力IB
alu: ALUの出力 mpy: 乗算器の出力

R/W
r: read w: write

(例) dm(hau,alu,w): アドレスHAUにALUのデータを書き込む。

dmb(BANK)

データメモリのバンクを制御する関数

BANK バンク番号 0~255

(例) dmb(255): データメモリのバンク255を選択する。

dlc(DL)

遅延回路を制御する関数

DL 遅延量 0~7

(例) dlc(4): デレイ量を4に設定する。

tmc(DL)

ラインディレイを制御する関数

DL 遅延量 7~912

(例) tmc(500): デレイ量を500に設定する。

ldr(SREG,DATA)

定数レジスタの値をセットする命令

SREG 定数レジスタ

hint1: HAUの初期値レジスタ1	hint2: HAUの初期値レジスタ2
hincl: HAUの増分値レジスタ1	hint2: HAUの増分値レジスタ2
hcml: HAUの比較値レジスタ1	hcml: HAUの比較値レジスタ2
hrep1: HAUの置換値レジスタ1	hrep2: HAUの置換値レジスタ2
vint1: VAUの初期値レジスタ1	vint2: VAUの初期値レジスタ2
vincl: VAUの増分値レジスタ1	vint2: VAUの増分値レジスタ2
vcml: VAUの比較値レジスタ1	vcml: VAUの比較値レジスタ2
vrep1: VAUの置換値レジスタ1	vrep2: VAUの置換値レジスタ2
acon: ALUのレジスタ	acon: 乗算器のレジスタ

(例) ldr(hint1,0): hint1に0を置数する。

(注) この命令は、強制的にcontinueになるため、while文などの制御命令中に記述することはできない。

mgr(GREG,SREG)

汎用インタフェースレジスタから定数レジスタにデータをコピーする命令

GREG 汎用インタフェースレジスタ

greg1 ~ greg7

SREG 定数レジスタ

hint1: HAUの初期値レジスタ1	hint2: HAUの初期値レジスタ2
hincl: HAUの増分値レジスタ1	hint2: HAUの増分値レジスタ2
hcml: HAUの比較値レジスタ1	hcml: HAUの比較値レジスタ2
hrep1: HAUの置換値レジスタ1	hrep2: HAUの置換値レジスタ2
vint1: VAUの初期値レジスタ1	vint2: VAUの初期値レジスタ2
vincl: VAUの増分値レジスタ1	vint2: VAUの増分値レジスタ2
vcml: VAUの比較値レジスタ1	vcml: VAUの比較値レジスタ2
vrep1: VAUの置換値レジスタ1	vrep2: VAUの置換値レジスタ2

acons: ALUのレジスタ mcons: 乗算器のレジスタ

(例) mgs(greg1, hint1): 'greg1をhint1にコピーする.

mfg (FREG, GREG)

関数レジスタから汎用インタフェースレジスタにデータをコピーする命令

FREG 関数レジスタ

alu : ALUの出力	mpy : 乗算器の出力
la : 外部入力1A	dm : Data Memoryの出力
hau : HAUの出力	vau : VAUの出力
hreg1: HAUのREG1	vreg1: VAUのREG1
hreg2: HAUのREG2	vreg2: VAUのREG2

GREG 汎用インタフェースレジスタ

greg1 ~ greg7

(例) mfg(alu, greg1): 'aluレジスタをgreg1にコピーする.

mgx (GREG, GREG)

汎用インタフェースレジスタから汎用インタフェースレジスタにデータをコピーする命令

GREG 汎用インタフェースレジスタ

greg1 ~ greg7

(例) mgx(greg1, greg2): 'greg1をgreg2にコピーする.

【付録 2】 ネットワーク用言語仕様概説

1. プログラムの形式

- ・プログラムの形式は以下の通り。

```
(例) network[          `ネットワークの指定
      'MODULE1':      `モジュール名の指定 (''の中,24char以内)
      net(in1,pu1a):  `以下, 実行文
      net(in2,pu1b):
      net(pu1,pu2a):
      net(pu2,pu3a):
      while(!shd) {
          net(pu2,out2):
          net(pu3,out2):
      }
      return:        `ネットワークのプログラムMODULE1の終り
network[            `ネットワークの指定
'MODULE_NAME':    `モジュール名の指定 (''の中,24char以内)
.....          `以下, 実行文
.....
.....
]                  `ネットワークのプログラムの終り
```

このように、ひとつのソースファイル中に複数のプログラムが記述できる。

- ・コメント、モジュール名以外は、小文字を使用する。
- ・各行の ` 以降は、コメントが記述できる。

2. 制御命令

- ・制御命令には以下のものがある。

goto LABEL

LABELのついた文へ無条件にジャンプする。

ラベルは、label:のように記述する。

```
(例) loop:
.....
      goto loop: `loopにジャンプする。
```

if (FLAG) goto LABEL

FLAGがonなら、LABELのついた文にジャンプする。

FLAG フラッグ

shd: システムのHD, svd: システムのVD, sfd: システムのFP
上記の条件FLAGの否定として、!が使える。(例: !shd,!svdなど)

```
(例) loop:
.....
      if (shd) goto loop: `shdのときloopにジャンプする。
```

while (FLAG) FUNC_STATEMENT

FLAGがonの間、FUNC_STATEMENTが実行される。

FLAG フラッグ

shd: システムのHD, svd: システムのVD, sfd: システムのFP
上記の条件FLAGの否定として、!が使える。(例: !shd,!svdなど)

```
(例) while(shd) net(...): `shdの間 net(...)を実行する。
```

while (FLAG) {FUNC_STATEMENTS}

FLAGがonの間、FUNC_STATEMENTSが実行される。

while文は、1重の入れ子までしか許されない。

FLAG フラッグ

shd: システムのHD, svd: システムのVD, sfd: システムのFP

上記の条件FLAGの否定として、!が使える。(例: !shd,!svdなど)

(例) while(svd){ `svdの間 { から } の命令を実行する、

.....

}

halt

その行で停止する。

(例) halt: `ここ以下の行を実行しない。

return

ルーチンから抜け出る。インストラクションセットのRESTARTを実行する。

(例) return: `RESTART命令が実行される。

3. ファンクション命令

・ファンクション命令には、以下のものがある。

net (IN,OUT)

ネットワークの入出力を接続する関数

IN 入力

in1~in16 クラスタ外部からの入力

pul~pul6 プロセッサ1~16の出力

OUT 出力

out1~out16 クラスタ外部への出力

pula~pula6a プロセッサ1~16の入力1A

pulb~pul6b プロセッサ1~16の入力1B

(例) net(in1,pula): `in1をpulaにつなぐ。