Vol. 28 No. 12 情報処理学会論文誌 Dec. 1987

音響ディジタル信号処理を主目的とする高速演算装置 μKIDOCH†

安倍正人†† 嶋明弘†† 上田隆††
金井浩†† 牧野正三†† 城戸健一††

筆者らは，ディジタル信号処理技術を応用して，音響および音声信号処理の研究を行っている．具体的には，音響信号処理分野では機械系の故障診断，音源位置の推定を行っており，音声信号処理の分野では不特定話者単語音声認識に関する研究を行っている．これらの処理はいずれも膨大な計算量および記憶容量を必要とし，かつ，最終的にはリアルタイムで行わなければならない．そのため，汎用計算機では演算速度，リアルタイム性および処理の連続性の点で問題があり，DSP では記憶容量，演算精度およびソフトウェアの柔軟性の点で問題がある．このため，筆者らはホスト計算機のバックエンドプロセッサとして，ディジタル信号処理を主な目的とした以下に示す3つの特徴を持つ高速演算装置 μKIDOCH を開発した．(1)ホスト計算機との間のデータ転送ネックを解消するため，パイプライン化メモリにより結合する．(2)マイクロプログラム方式によるパイプライン処理を行う．(3)データバスあるいはアドレスバスとして使える3本の 32 bit バスおよび 128 bit プログラムバスを用いることにより，複数の演算器が並列に動作する．本報では，試作した高速演算装置 μKIDOCH の特徴とクロスアセンブラを用いて作成した種々のプログラムにより検討した μKIDOCH の性能について報告する．

1. まえがき

近年のディジタル信号処理技術の応用分野は，電気通信のみならず，医学，生物学等の自然科学，さらには社会科学など多岐にわたっており，これを処理する計算機には，処理の多様化に伴う柔軟性，複雑化に伴う高速性が求められている．ディジタル信号処理で扱う演算を一般の数値演算と比較すると(1)乗算，加減算といった演算を，(2)データの内容に依存せず，常に一定のアルゴリズムで，(3)多数回繰り返す，という特徴を持っている．このことにより，ハードウェアの設計にあたっては，パイプライン処理等の高速化の手法が適用しやすく，ソフトウェアの作成にあたっては，最適化を施しやすいと言える．

以上のような背景と最近の LSI 技術の発展によって，プログラムメモリと高速演算器を内蔵した，ディジタルシグナルプロセッサ (DSP) LSI が各社から発表されている[1)~3)]．また，汎用計算機でも，従来のハードウェアに内蔵アレイプロセッサ (IAP) を付加し，自動ベクトル化の機能を持った最適化コンパイラを搭載することが多くなっている[4)]．

当研究室では，ディジタル信号処理技術を応用して，音響，音声信号処理の研究を行っている．具体的には，音響部門では，音源位置の推定システム[5),6)]の開発，機械系の故障診断システム[7)]の開発，音声部門では，不特定話者音声認識システム[8)]の構築等を行っている．これらのシステムはいずれも膨大な計算量を必要とするもので，実時間もしくはそれに準ずる速度で稼動させる必要がある．例えば，音声収録を目的としてセンサを 16 個用いて，音源位置推定システムを構築する場合には，1024 点 FFT を 16 チャネル分，連続して実時間で実行できる必要があるため，20 kHz サンプリングするとして，FFT 1 回当たり約 3 ms 以内で実行する必要がある．また，音声認識システムにおいては，LP 分析等において，演算精度が問題になり，浮動小数点演算が必要になる．また，建築音響において，部屋のインパルスレスポンスをクロススペクトル法で推定するためには，FFT の長さを十分長く取る必要があり[9)]，少なくとも 8182 点 FFT が必要になる．このような仕様をすべて満たすためには，以下のような理由から専用の高速演算装置が不可欠である．すなわち，汎用計算機を用いたシステムでは，記憶容量，演算精度の点では満足できるものであるが，特定の目的のための専用機として使用するには演算速度あるいはリアルタイム性の点でまだ問題がある．DSP を用いたシステムでは，処理速度あるいはリアルタイム性の点ではほぼ満足のいくものであるが，記憶容量が小さいため，例えば，FFT では 1024 点まで

† High Speed Computing Machine μKIDOCH for Digital Signal Processing on Acoustics by Masato Abe, Akihiro Shima, Takashi Ueda, Hiroshi Kanai, Shozo Makino and Ken'iti Kido (Research Center for Applied Information Sciences, Tohoku University).

†† 東北大学応用情報学研究センター

しか実行できないことのほかに，演算精度，およびソフトウェアの柔軟性という点で問題のある場合が多い．ただし，演算精度に関しては，最近の DSP の中には浮動小数点演算ができるものもある[3]が，依然として，メモリ容量は小さく，音響信号処理の分野には使えない場合が多い．このような理由から，従来から科学計算や信号処理を目的として，様々な並列処理のマシンが提案されて来ている[10)~15)]が，SIMD 型の計算機[10),11)]である ILLIAC Ⅳ[12)]や BSP[13)]では，任意のプログラムを書くのが難しい．また，VLIW 型の計算機[11)]である QA-Ⅱ[15)]は浮動小数点演算ができず，また同じ機能を持つ演算器が 4 台あるため，例えば，専用の最適化した FORTRAN コンパイラ等を構築するのは難しい．また，同じ VLIW 型の計算機である AP-120 B[14)]は，連続して同じ処理を行うような場合に，ホスト計算機との間の転送時間が問題となる場合が多い．このような理由から，ディジタル信号処理を主な目的とした高速演算装置の開発の必要が生じ，次のような特徴を持つ VLIW 方式のバックエンド演算システム μKIDOCH を設計，試作した[16)]．

（1） ホスト計算機との間のデータ転送ネックを解消するため，μKIDOCH とホスト計算機の間をパイプライン化メモリにより結合する．

（2） 様々な応用に柔軟に対応させるために，128 bit/word のマイクロプログラム方式によるパイプライン処理を行う．この方式を取ると，コンパイラを容易に構築できる[10)]．

（3） データバスあるいはアドレスバスとして使える 3 本の 32 bit バスを用いることにより，複数の演算器が並列に動作する．

（4） データ形式として，A/D 変換直後のデータ (12 bit 固定小数点) を効率的に扱うための CI 型 (Complex Integer : MSB 16 bit 実数部，LSB 16 bit 虚数部) を設ける．

（5） 信号処理で行われる演算はすべて複素数データの積和演算（バタフライ演算）の組合せで表現できるが，CI 型のバタフライ演算を高速に実行するためのバタフライ演算器を用意する．

（6） 複雑な処理での演算精度の確保のために浮動小数点型も設ける．

（7） プログラム作成のためにクロスアセンブラを用意する．

本報では，試作した高速演算装置 μKIDOCH の特徴とクロスアセンブラを用いて作成した種々のプログ

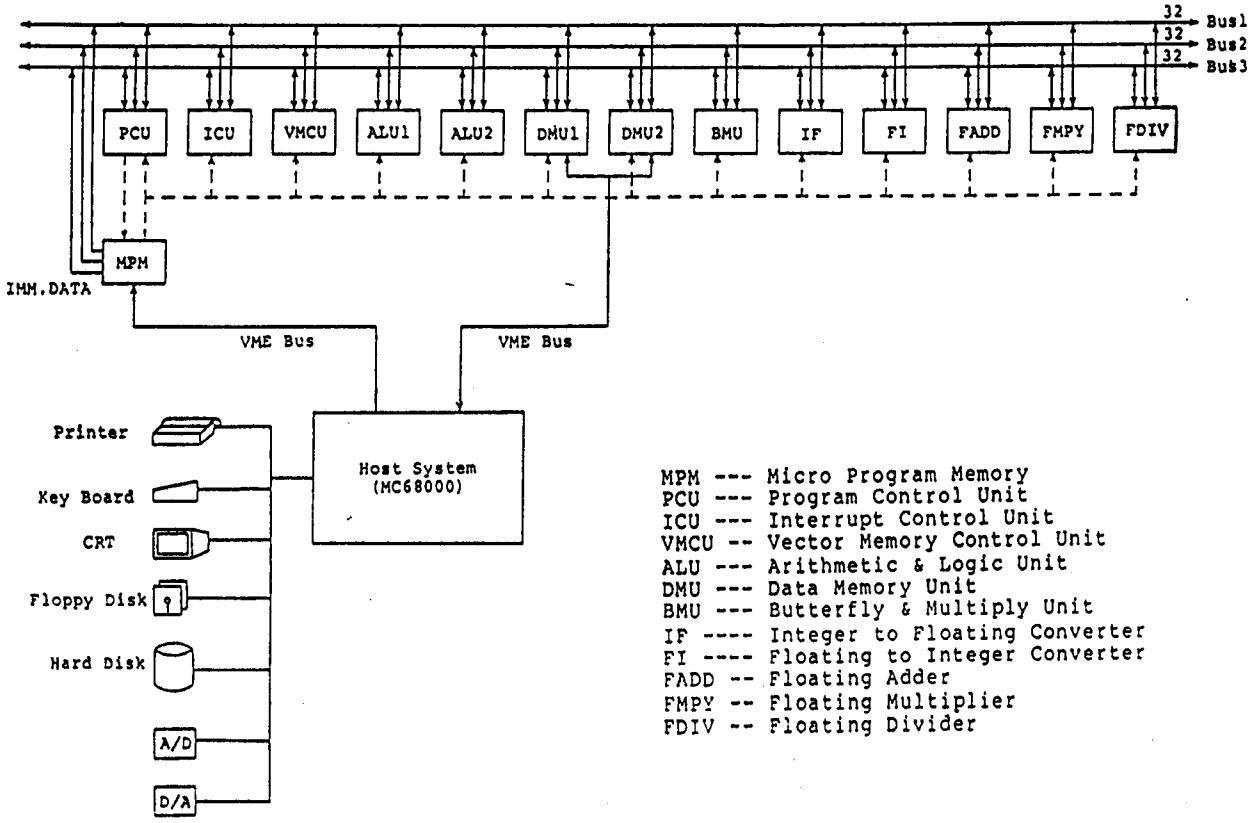

図 1 システムの全体構成
Fig. 1 Block diagram of the system.

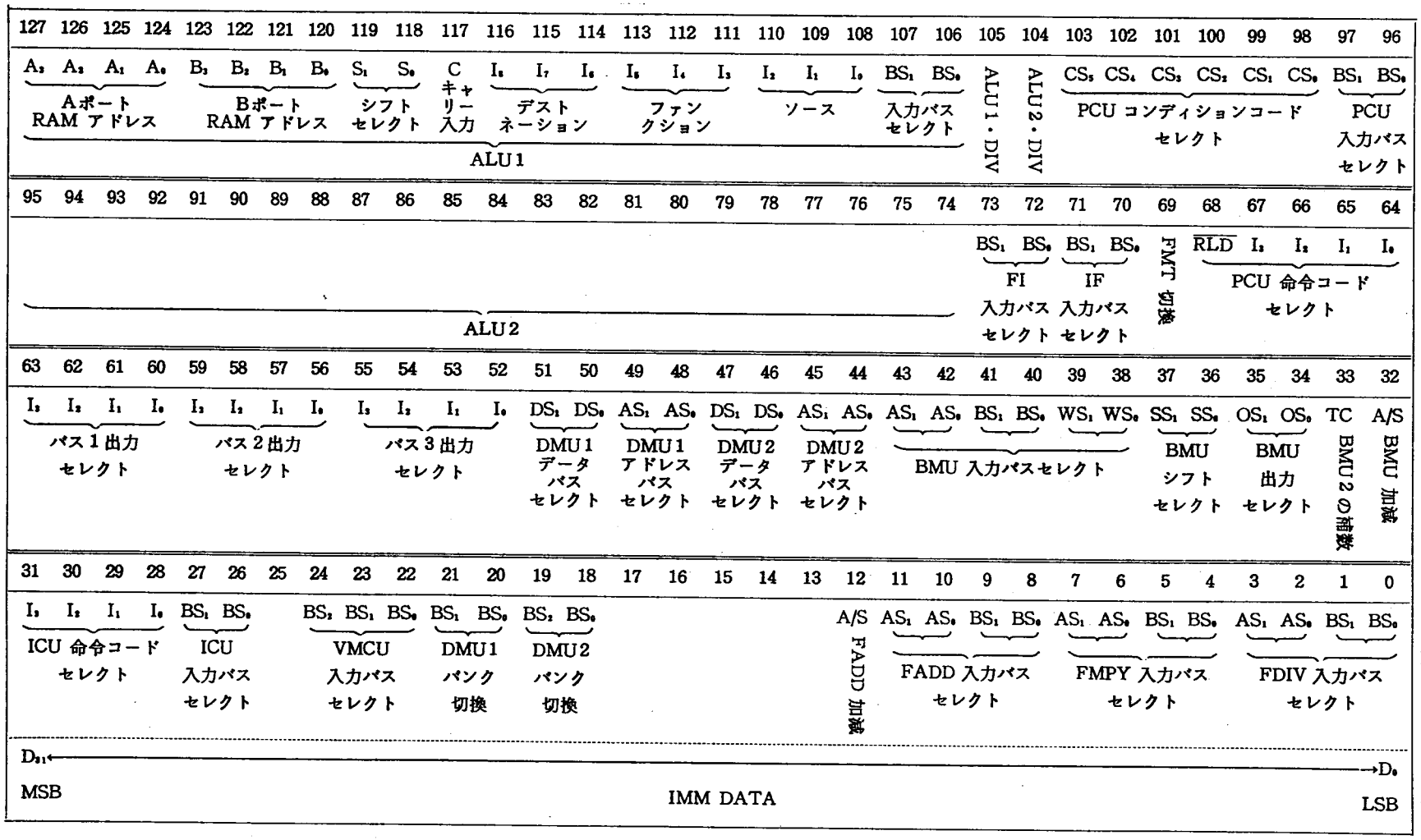

図 2 マイクロプログラムフィールド
Fig. 2 Micro-Program field of μKIDOCH.
命令フィールドの 31～0 bit は bit 69 の FMT 切換により，FMT0 と FMT1 の2つに分けられる．

ラムにより検討した μKIDOCH の性能について報告する.

2. ハードウェア構成

図1に本システムの全体の構成を示す. ホスト計算機はモトローラの 68000 を使用した市販の製品で, OS は現在は CP/M-68 K を使用している. このホスト計算機と μKIDOCH は VME バスを通じて結合されている.

μKIDOCH は独立に稼動する 14 個の装置から構成されており, 約 A3 判大のプリント基板 8 枚からできている.

μKIDOCH はパイプライン処理[17]により高速化を図っている. 1つのパイプラインは, セグメントと呼ばれるさらに細かい種々の演算ユニットの連続体として構成され, これらは同期化されたタイミング信号の制御を受ける. 2つのセグメント間にはラッチレジスタが入れられ, これがセグメントの入出力データを保持する. セグメント S_i での遅延時間 τ_i, ラッチレジスタでの遅延時間を τ_l とすると, 同期クロックの最小周期は,

$$\tau = \mathrm{Max}\{\tau_i\} + \tau_l \qquad (1)$$

である必要がある. また, このパイプラインは n 個のセグメントから構成されているものとすれば, 滞在時間は $n\tau$, 処理時間は τ となる.

μKIDOCH の特徴の1つに, 内部バスが3本あることがあげられる. これにより, 例えば次に用いるデータメモリのアドレスの転送, 前に与えたアドレスのメモリ内容の転送および演算器からの出力の3つのデータの転送が同時に行える. さらに, バス入力の制御が 2 bit のマイクロプログラムで効率良く行うことができることも特徴の1つである. すなわち, (0) ラッチしない, (1) バス1からデータをラッチする, (2) バス2からデータをラッチする, (3) バス3からデータをラッチするの4通りがわずか 2 bit で行える. もし, バス入力の制御にマイクロプログラムの bit 数を増やして, 例えば 3 bit にすると, ラッチしない場合を除いて, バスの数を7本に増やさないと効率が悪く, また増やすとコストがかかりすぎ, 現実的でない.

2.1 制御装置 (MPM, PCU, ICU, VMCU)

MPM はマイクロプログラムメモリで, 4 k×128 bit の RAM で構成されている. 図2に, マイクロ命令の構成図を示す. PCU はプログラムコントロールユニ

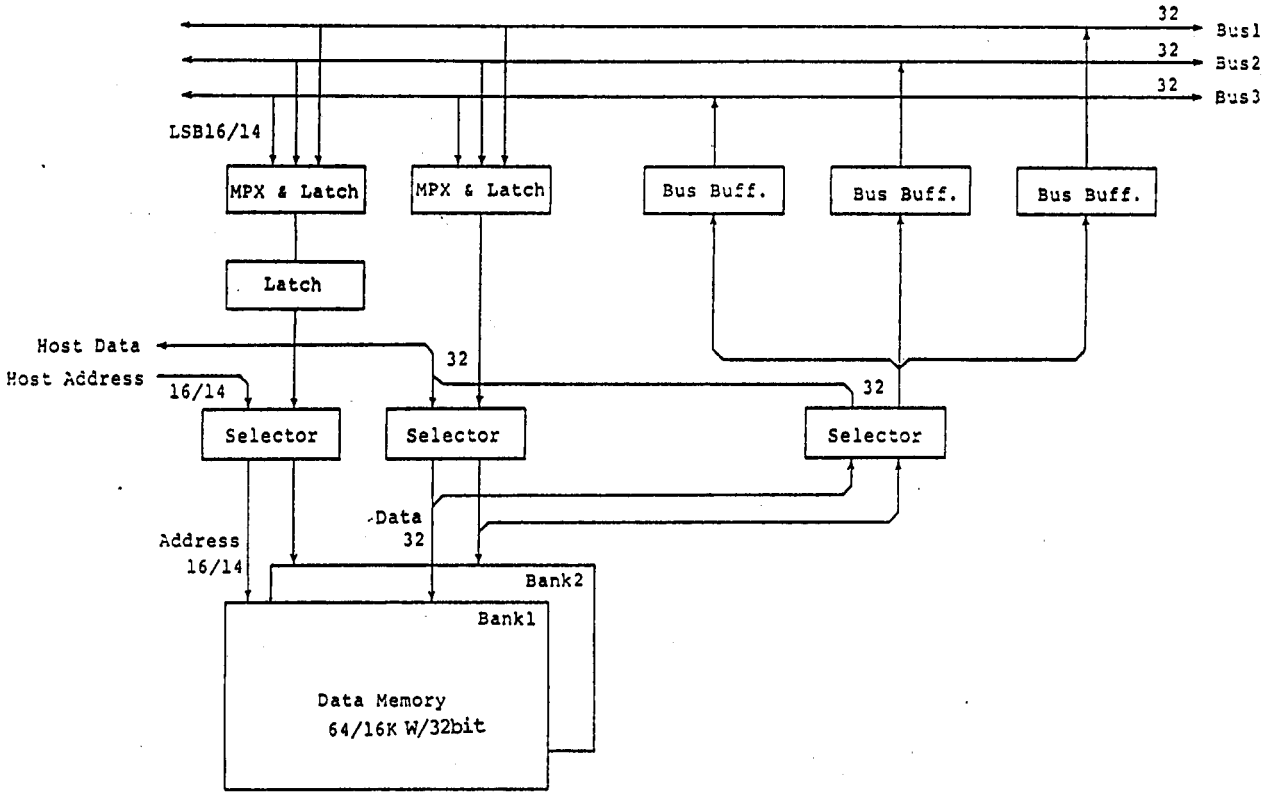

図 3 DMU 構成図
Fig. 3 Block diagram of Data Memory Unit.

ットで，AMD 社のプログラムシーケンサ Am 2910 を中心に構成されている．ICU は割込み制御ユニットで，各演算装置からのオーバフローとアンダフローおよびホスト計算機からの割込みを監視し，必要があれば次に実行するマイクロプログラムのアドレスを VMCU（割込みベクトルメモリ制御ユニット）から読み出し，PCU に送る．

2.2 記憶装置 (DMU 1, DMU 2)

DMU 1 および DMU 2 は，図 3 に示すように，32 bit/Word で，それぞれ 64 kWord・2 Bank，16 kWord・2 Bank のデータメモリユニットである．各メモリユニットの 2 つのバンクのうち，どちらか一方は μKIDOCH がアクセスし，他方はホスト計算機に接続されており，ホスト計算機から見れば単なるメモリである．バンクの切り替えはマイクロプログラムによって行われる．1 回のメモリアクセスには 2 ステップ必要であるが，内部はパイプライン化されており，1 ステップごとに連続したアクセスが可能である．

各メモリユニットが 2 つのバンクを持つと，例えば A/D 変換と FFT を連続して行う場合には事実上データの転送に要する時間が 0 とみなせる．すなわち，A/D 変換に要する時間，FFT に要する時間およびホスト計算機で後処理する時間のうち一番長い時間を 1 フレームの長さとすると，図 4 において，フレーム 1 ではホスト計算機に接続されている A/D 変換器から DMA（ダイレクトメモリアクセス）コントローラにより直接 DMU 1 のバンク 1 にデータが転送される．フレーム 1 とフレーム 2 の境目でバンクが切り替えられる．フレーム 2 では，μKIDOCH はフレーム 1 で与えられたデータを FFT し，結果を DMU 2 に出力する．その間，A/D 変換器に付属した DMA コントローラは次のデータを DMU 1 のもう 1 つのバンクに転送する．次のフレームとの境目で再びバンクが切り替えられる．フレーム 3 ではホスト計算機は DMU 2 にある FFT された結果を直接後処理することができる．また，その間に μKIDOCH はフレーム 2 で与えられた DMU 1 にあるデータを FFT し，DMA コントローラは次のデータを DMU 1 の他のバンクに転送する．このように，連続して同一の演算を行う場合には，DMA コントローラ，μKIDOCH およびホスト計算機が完全に並列動作し，データの転送時間は実質的に 0 とみなせる．

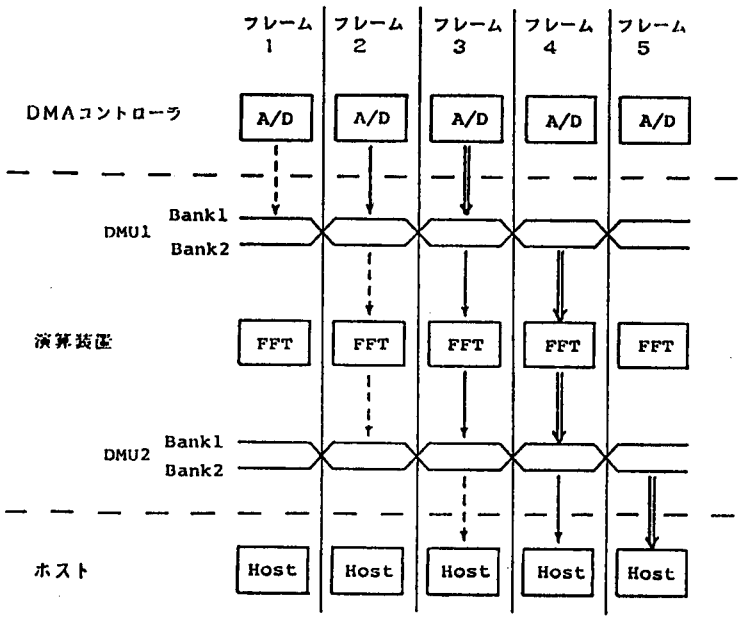

図 4 DMA コントローラ，演算装置およびホストとの間でのパイプライン処理

Fig. 4 Pipeline processing among DMA controller, processor and host computer.

2.3 算術論理演算ユニット (ALU 1, ALU 2)

ALU 1, ALU 2 はそれぞれ AMD 社の 4 bit のビットスライスマイクロプロセッサ Am 2901 C を 8 個用いて 32 bit 構成としたものである．演算結果のコンディションは PCU から参照でき，条件 JUMP 等が行える．ALU 1 および ALU 2 は固定小数点データの演算（加減算および除算）のみでなく DMU 1 あるいは DMU 2 のアドレス計算にも用いられる．

2.4 バタフライ演算ユニット (BMU)

信号処理演算は一般に積和演算の繰り返しから構成されている．特に 2 を基数とした FFT ではバタフライ演算（複素数の積和演算）の繰り返しから構成されている．汎用計算機あるいはマイクロプロセッサ等では，このバタフライ演算を逐次的に行っているために時間がかかる．それに対して，BMU は図 5 に示すように 16 bit の乗算器を 4 個および 16 bit の ALU 7 個を並列に動作させて，(1)CI 型のバタフライ演算，(2) INT 型の符号なし乗算を行うユニットである．内部的には(1)入力データのシフトおよび乗算，(2)各乗算器の出力の加算，の 2 つのセグメントに分かれており，滞在時間 2 ステップ，処理時間 1 ステップで実行できるようになっている．

BMU を用いたバタフライ演算では，実数部，虚数

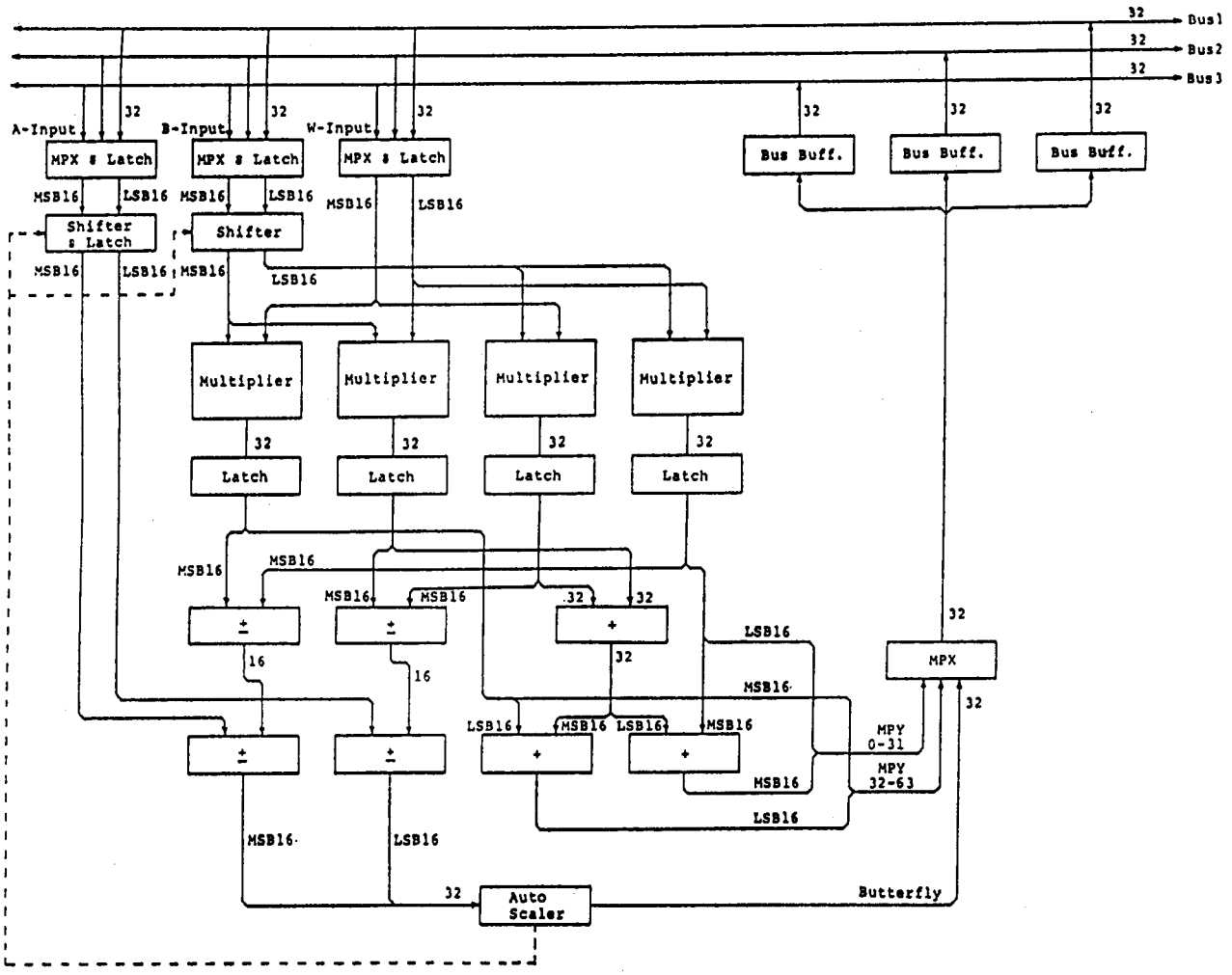

図 5 BMU 構成図
Fig. 5 Block diagram of Butterfly & Multiply Unit.

部はそれぞれ 16 bit の固定小数点の形式になっており，演算の途中でオーバフローがおきる可能性がある．これを防ぐために，BMU にはオートスケーリング機能が付加されている．すなわち，このまま演算を続けた場合に次のステージでオーバフローが生じる可能性があることを実数部および虚数部それぞれに対して上位 2 bit の排他的論理和を取ることにより検出し，次のステージではA入力とB入力を 1 bit ダウンシフトさせるというものである．

2.5 浮動小数点演算ユニット (FADD, FMPY, FDIV, IF, FI)

FADD, FMPY, FDIV, IF, FI はそれぞれ浮動小数点加減算ユニット，浮動小数点乗算ユニット，浮動小数点除算ユニット，固定小数点から浮動小数点へのデータ変換ユニットおよび浮動小数点から固定小数点へのデータ変換ユニットで，TTL, PAL (Programmable Array Logic)[18] および 16 bit×16 bit の乗算器で構成されている．FADD 内部では(1)桁合せ，(2)加減算の 2 つのセグメントに，FMPY 内部では(1)乗算，(2)正規化の 2 つのセグメントに分かれており，どちらも滞在時間 2 ステップ，処理時間 1 ステップで演算を行う．また，FDIV は 10 ステップ，FI, IF はそれぞれ 1 ステップで演算を行う．

3. μKIDOCH のアセンブラ言語

μKIDOCH は 128 bit のマイクロプログラムによって各演算器，メモリ等の機能とバスの結合状態を制御している．このように長大なマイクロプログラムを直接 2 進数によって記述することは非人間的な作業であり，単純な誤りの原因にもなる．そこで，ホスト計算機を用いた μKIDOCH 用のクロスアセンブラを製作した．

図 6 (a)に示したものは，128 bit のマイクロプログラムの 1 命令に相当するものであり，アセンブラで

記述された一連のアセンブラプログラムは，その集りとして記述される．

＜ラベル＞は，マイクロプログラム上でのこの命令のアドレスを保持する．

＜ユニット名＞には，{ALU 1, ALU 2, PCU, ICU, VMCU, DMU 1, DMU 2, BMU, FI, IF, FADD, FMPY, FDIV} のうち1つを指定することができる．機能については，それぞれのユニット名に応じて，種々のものが設けられている．

＜疑似ユニット名＞には，{ORG, BSO 1, BSO 2, BSO 3, IMM, FMT, IRT 68 K} がある．ORG は，生成されるマイクロプログラムの先頭アドレスを指示するもので，アセンブラプログラムの第一行目に書く必要がある．BSO n はバス n に出力するユニットを指定するもので，＜疑似機能＞の部分には＜ユニット名＞を記述する．IMM, FMT, IRT 68 K はそれぞれ，マイクロプログラムに内包する 32 bit のデータ，マイクロプログラムフィールドのフォマット変換，ホスト計算機への割込みを指示するものである．図6(b)は，アセンブラで記述したマイクロプログラムの一例である．

△＜ラベル＞:
△＜ユニット名＞△＜機能＞，＜機能＞，…，＜機能＞，△；＜コメント＞
△＜疑似ユニット名＞△＜疑似機能＞△／

ただし△は，任意個の空白を示す．

図6(a) アセンブラプログラムの書式
Fig. 6(a) Format of assembly program.

```
以下の計算を i=0 から 1023 まで繰り返す

  a[i] = a[i] + b[i] * c[i]

ただし a,b,c はそれぞれ
  DMU1 の    0 番地以降
  DMU2 の    0 番地以降
  DMU2 の  400 番地以降
に記憶しているものとする．

IMM     0                          ; a,b の先頭番地( 0x0 )を
BSO1    IMM                        ;  バス1 を通して
ALU1    IB1.DZ.OR.RAMF.B15         ;  ALU1 のレジスタ15および
ALU2    IB1.DZ.OR.RAMF.B15 /       ;  ALU2 のレジスタ15に入れる

IMM     400                        ; c の先頭番地( 0x400 )を
BSO1    IMM                        ;  バス1 を通して
ALU2    IB1.DZ.OR.RAMF.B14 /       ;  ALU2 のレジスタ14に入れる

IMM     3FF                        ; ループの反復回数を
BSO1    IMM                        ;  バス1 を通して
PCU     IB1.RLD                    ;  PCU のレジスタに入れる
ALU2    A15.ZA.ADD.C1.RAMA.B15     ; b の番地を ALU2 から
BSO2    ALU2                       ;  バス2 を通して
DMU2    IA2 /                      ;  DMU2 へ送る

ALU2    A14.ZA.ADD.C1.RAMA.B14     ; c の番地をALU2 から
BSO1    ALU2                       ;  バス1 を通して
DMU2    IA1 /                      ;  DMU2 へ送る

BSO2    DMU2                       ; DMU2 は b の値をバス2 へ送り
FMPY    IA2 /                      ;  FMPY は これをラッチする

ALU1    A15.ZA.ADD.C1.RAMA.B15     ; a の番地を  ALU1 から
BSO1    ALU1                       ;  バス1 を通して
DMU1    IA1                        ;  DMU1 へ送る
BSO2    DMU2                       ; DMU2 は c の値をバス2 へ送り
FMPY    IB2                        ;  FMPY は これをラッチする
PCU     PUSH /                     ;

ALU2    A15.ZA.ADD.C1.RAMA.B15     ;
BSO2    ALU2                       ;
DMU2    IA2 /                      ;

ALU2    A14.ZA.ADD.C1.RAMA.B14     ;
BSO1    ALU2                       ;
DMU2    IA1                        ;
BSO2    DMU1                       ; DMU1 は a の値をバス2 へ送り
BSO3    FMPY                       ; FMPY は b*c の結果をバス3 へ送り
FADD    IA2,IB3 /                  ;  FADD はこれら 2 つのデータをラッチする

BSO2    DMU2                       ;
FMPY    IA2                        ;
FADD    ADD/                       ;

ALU1    A15.ZA.ADD.C1.RAMA.B15     ;
BSO1    ALU1                       ;
DMU1    IA1                        ;
BSO2    DMU2                       ;
FMPY    IB2                        ;
BSO3    FADD                       ; FADD は a+b*c をバス3 へ送り
DMU1    ID3                        ;  DMU1 はこれをラッチする
PCU     RFCT /                     ;
```

図6(b) アセンブラで記述したマイクロプログラムの一例
Fig. 6(b) An example of Assembler program.

4. ホスト計算機からの処理の依頼

ホスト計算機は初めに μKIDOCH のマイクロプログラムメモリ (MPM) にプログラムをロードしておき，起動をかける．その後，ホスト計算機は必要なデータをデータメモリ1 (DMU 1) あるいはデータメモリ2 (DMU 2) にストアし，μKIDOCH に割込みをかける．μKIDOCH は必要な演算を実行後，ホスト計算機に割込みをかけて処理の終了を知らせ，ホスト計算機からの確認を受けたあとにメモリのバンクを切り替えることにより演算結果をホスト計算機に引き渡す．

5. μKIDOCH の性能の評価

本章では，まず，μKIDOCH とほぼ同じ目的で設計されたと思われる AP-120 B と μKIDOCH との比較を行い，続いて，当研究室で行っている研究を応用して種々のシステムを構成するのに必要な 1024 点 FFT と 29 チャネル帯域フィルタ群を用いて，製作した μKIDOCH の性能評価を行う．

5.1 AP-120 B との比較検討

(1) AP-120 B のバスは(a)アドレス専用バス，(b)データメモリあるいはレジスタファイルから演算器 (FALU：浮動小数点 ALU と FMPY：浮動小数点乗算器) への入力バスおよび，(c)演算器から演算器，データメモリあるいはレジスタファイルへのリザルトバスという異なる機能を持つ3本のバスから構成されている．このため，AP-120 B では，FALU と FMPY とから同時に出力を得ることができない．また，2以上の次元を持つ配列のアドレッシングには固定小数点の乗算が必要であるが，AP-120 B ではアドレスの計算に固定小数点の乗算を実行することができないため，2次元以上の配列を扱うのは不利である．これに対して，μKIDOCH はどんな情報も流すことが可能な汎用バス3本で構成されているので，プログラミングの自由度が大きい．また，μKIDOCH には固定小数点乗算器があり，任意の次元の配列を扱うことができる．

(2) AP-120 B の演算器 FALU および FMPY はデータメモリとレジスタファイルに対して対等であり，信号処理で最も良く用いられる積和演算を特に意識して作られてはいない．これに対して，μKIDOCH では，16 bit 固定小数点用ではあるが，BMU と呼ばれるバタフライ演算器が用意してある．ただし，16 bit 固定小数点といっても，ブロックフローティング FFT ができるので，ダイナミックレンジとしては 60 dB 以上が得られ，音響信号処理としては十分な演算精度が得られる．

(3) AP-120 B も μKIDOCH もともにバックエンドプロセッサであり，使用するためにはホスト計算機が必要であるため，ホスト計算機との間のデータ転送が必要となる．このデータ転送の効率化に関しては，AP-120 B はなんらの対策も用意していないが，μKIDOCH ではパイプライン化メモリを用意して処理を連続して行う場合にはデータ転送の時間を実効的に0にすることができる．

5.2 1024点 FFT

図7に μKIDOCH で FFT を実行する際のアルゴリズムを示す．FFT のアルゴリズムは，時間間引き型とし，データのアドレス計算およびループの制御はすべて加減算，シフト演算のみで処理できるようにしてある．また，2つのメモリユニットを持っているという μKIDOCH のハードウェアの特徴を生かすために，一方のメモリユニットを読み出し専用，他方を書込専用とし，ステージが変わるたびにこれらを反転させるという方法を用いる．

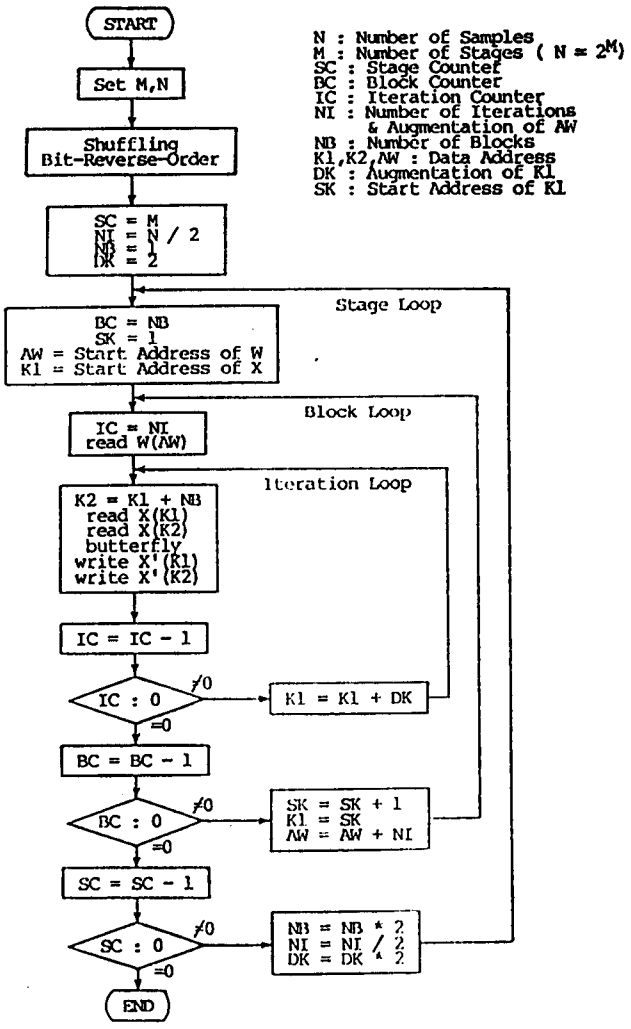

図7 FFT のアルゴリズム
Fig. 7 Algorithm of FFT.

プログラムは Stage Counter, Block Counter, Iteration Counter を制御変数とする3重ループからなる．このうち，最も内側のループ（メモリアクセスも含めた1回のバタフライ演算に相当）は最も多く実行されるものであり，この部分の処理速度をいかに速くするかが FFT 全体の実行速度を決定する．図8に FLOAT 型のバタフライ演算のタイミングチャートを示す．バタフライ演算を連続的に行う場合には，図中で0～2ステップで行う演算を12～14ステップのところで行わせることが可能である．そのため，実効的には1バタフライ演算当たり12ステップで処理できる．ただし，タイミングチャート中の &An は変数 An のメモリユニット上でのアドレスを表している．このときの各バスおよび演算器の稼働率を表1に示す．なお，CI 型は3ステップでバタフライ演算が実行でき

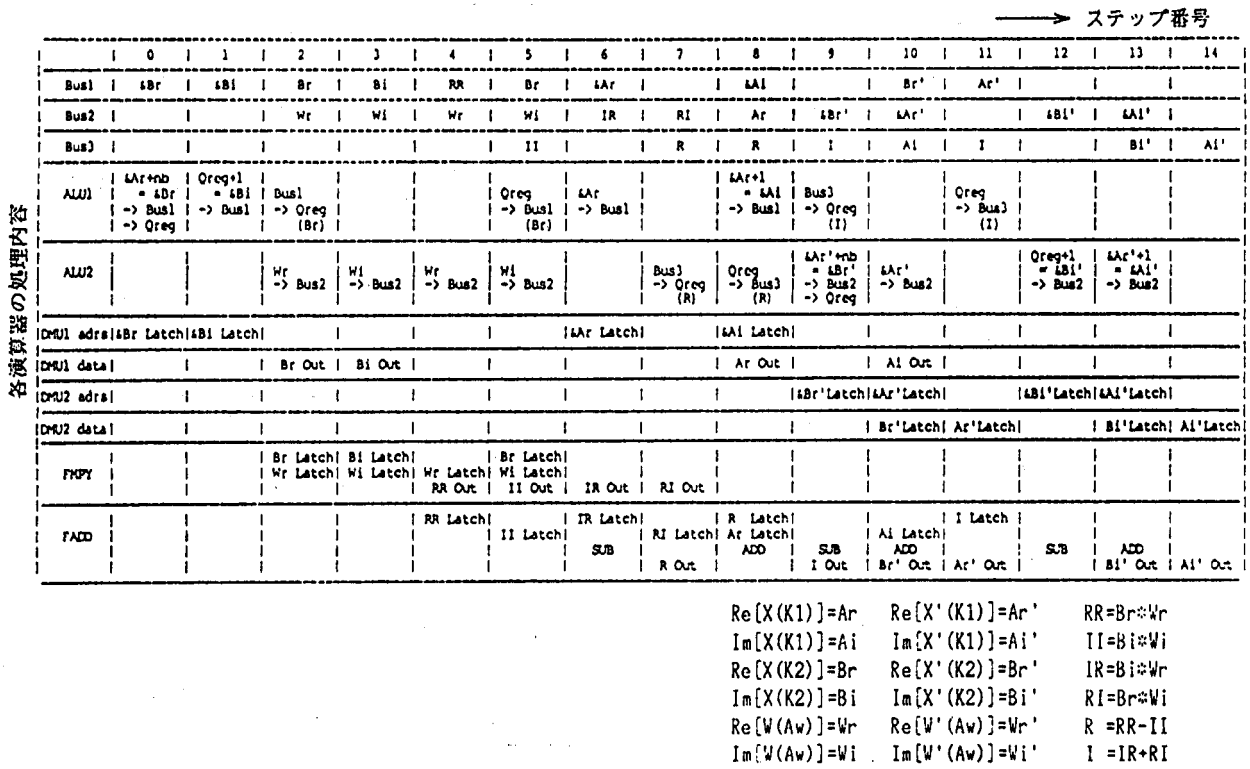

図 8 FLOAT 型バタフライ
Fig. 8 Butterfly program for FLOAT type data.

表 1 FFT 実行時の演算装置各部の稼働率
Table 1 Working ratio of each processing unit at execution of FFT.

CI 型の場合

	Bus	ALU	DMU1	DMU2	BMU
バタフライ	8/9 =89%	6/6 =100%	2/3 =67%	2/3 =67%	2/3 =67%

FLOAT 型の場合

	Bus	ALU	DMU1	DMU2	FADD	FMPY
バタフライ	29/36 =81%	18/24 =75%	4/12 =33%	4/12 =33%	6/12 =50%	4/12 =33%

表 2 各種システムでの 1024 ポイント FFT の処理時間
Table 2 Processing time of 1024 point complex FFT.

	msec.	データ形式
AP-120 B	7.13	38 bit 浮動小数点
μKIDOCH (CI 型)	3.5	16 bit Block Floating
μKIDOCH (FLOAT 型)	10.9	32 bit 浮動小数点
Cray-1	8.98	64 bit 浮動小数点
Cyber 750	24	64 bit 浮動小数点
MELCOM COSMO 700S	320	32 bit 浮動小数点
VAX 11/780	360	32 bit 浮動小数点
IBM 370	404	32 bit 浮動小数点
PDP 11/60	566	32 bit 浮動小数点

る．1ステップは 150 ns で動作するので，1024 点の複素数データの FFT を実行するのに，CI 型の場合は約 3.5 ms, FLOAT 型では 10.9 ms がかる．なお，表 2 には，代表的な計算機について，FFT の計算時間を示している．

5.3 巡回型ディジタルフィルタの構成

当研究室の音声認識システムの音響処理部では，24 kHz でサンプリングされた音声信号を 1/6 オクターブ間隔に並べた 29 チャネルから成る Q=6 の帯域フィルタ群によって周波数分析を行っている[19]．現在のところ，このフィルタ群は汎用計算機上で，ソフトウェアで構成されており，その処理のためには実時間の 20 倍程度の時間が必要であるが，μKIDOCH を用いると実時間処理が可能となる．ここでは，29 チャネル帯域フィルタ群を FLOAT 型で構成したときの結果を示す．この場合，1チャネル当たり滞在時間 14 ステップ，処理時間 9 ステップ (1.35 μs) となる．29 チャネルでは，メモリユニットのバンク切替を含めても 41 μs (サンプリング周波数換算で 24.2 kHz 相当) で実行できるので，十分に実時間処理が可能となる．また，そのときの各バスおよび演算器の稼働率を表 3 に示す．

表 3 フィルタリング実行時の演算装置各部の稼働率
Table 3 Working ratio of each processing unit at execution of filtering.

Bus	ALU	DMU1	DMU2	FADD	FMPY	IF
24/27 =89%	11/18 =61%	5/9 =56%	4/9 =44%	3/9 =33%	4/9 =44%	1/9 =11%

5.4 パイプライン化メモリの効用

図 4 に示した処理を連続して行う場合について，パイプライン化メモリの有効性を調べた．

データ転送を含めた A/D 変換に要する時間，FFT の時間，データ転送を含めたホスト計算機による後処理の時間をそれぞれ，T_{AD}, T_{FFT}, T_{HOST} とすると，パイプライン化メモリを用いない場合には，1 組の処理時間 T_N は

$$T_N = T_{AD} + T_{FFT} + T_{HOST} \quad (2)$$

となる．また，データの抜けを無くして処理を連続して行うためには，A/D 変換器には大容量のバッファメモリを付ける必要がある．一方，パイプライン化メモリを用いて連続的に処理する場合には，A/D 変換とホスト計算機による後処理はともに VME バスをアクセスするのでバスが競合する．そのため，1 組当たりの処理時間 T_P は，

$$T_P = \max(T_{AD} + T_{HOST}, T_{FFT}) \quad (3)$$

となり，連続して処理する場合にも，A/D 変換器には小容量のバッファメモリを付けるだけで，比較的容易に構成できる．図 9 は A/D 変換器のサンプリング周波数を 40 kHz, VME バスのデータ転送速度を 500 ns/サンプル，T_{FFT}=16 ms (1024 点複素数浮動小数点 FFT，固定小数点から浮動小数点への FORMAT 変換，および時間窓の掛け算を含む) としたときの T_{HOST} をパラメータとした T_N と T_P の値をプロットしたものである．この図から，パイプライン化メモリが有効であることがわかる．すなわち，サンプリング周波数 40 kHz で 1024 点 FFT を連続的に行うためには，1 組の処理（図 4 におけるフレームの長さ）を約 25 ms 以内に行わなければならない．その場合，なんらの対策も取らないときには，ホスト計算機に与えられる後処理の時間 T_N は 8 ms しかないが，パイプライン化メモリを用いると，A/D 変換のデータ転送の時間約 1 ms を除いて後処理の時間 T_P として約 24 ms も使うことができる．

5.5 FFT 演算の高速化のための μKIDOCH のアーキテクチャの改善

μKIDOCH のように各装置間のデータ転送をすべて内部バスで行うアーキテクチャでは，バスネックが問題となることがある．すなわち，μKIDOCH では出力できる装置が 14 (ただし，頻繁に使う装置は 7 程度) あるのにたいし，バスの数が 3 本しかないことが問題となる．これを解決するためには，例えば，(1) FMPY と FADD の間に積和演算のバイパスを付ける，(2) ALU 1 と ALU 2 の間に専用バスを設ける，(3) DMU にオートインクリメント等のアドレス計算機能を付ける等の方法がある．これらの方法を用いるとバスネックの問題は片付くが，新たにマイクロプログラムメモリのビット数を増やさなければならないのでコスト増になるという問題がある．さらに，信号処理において一番よく使うと思われる演算である FFT においては，以下に示す方法を用いた方が良い結果が得られるので，μKIDOCH では，これらの手法は用いていない．

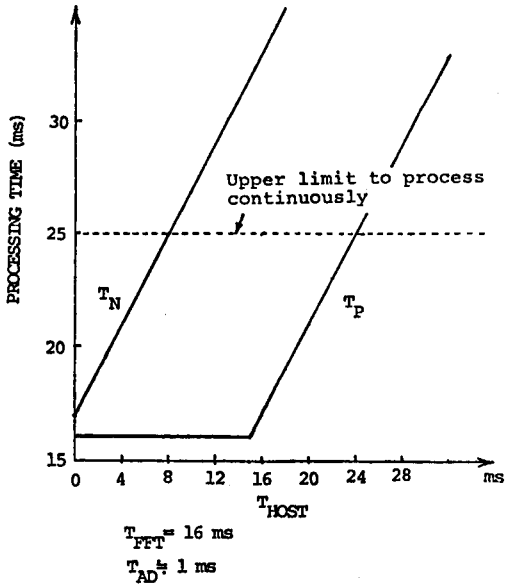

図 9 パイプライン化メモリの効果
Fig. 9 Effectiveness of pipeline memory.

信号処理の計算では，例えば，ディジタルフィルタのようにデータを反復的に用いることがよく起こる．このような場合，後で同じ値を使うのであるから，そのデータを取って置くことができれば，バスの効率の点で望ましい．FFT のバタフライ演算においてもこのことが当てはまる．すなわち，加算が 6 回，乗算が 4 回必要なため，バスネック等の障害が全然ない場合には，浮動小数点データに対しては 6 ステップで実行できるはずである．しかし，実際にはデータアドレスの計算を 8 回 (入力 A, B, 出力 A′, B′ : それぞれ実数部と虚数部)，さらに，回転因子の入力を 2 回行わな

ければならないので，ステップ数は増え，図8のタイミングチャートでは，12ステップかかっている．この原因は，(1)メモリユニットにはアドレス記憶領域が1つしかなく，また，FMPY や FADD には入力データの記憶領域が1つずつしかないため，後で同じ値を使うとわかっていても，それを取っておけず，再転送しなければならないこと，(2) FMPY および FADD は入力データを与えてから3ステップ目ではじめて結果が利用できるため，他の装置とタイミングがうまく合わず，演算器が利用できないステップが生じるためである．もしも，(1)での問題である記憶領域がそれぞれ2つになると，ステップ数は7になる．さらに，(2)が解決できれば6ステップとなり，理論上の最高速度で1バタフライが実行できることになる．μKIDOCH の次期機種では(1)の手法を採用する予定である．

6. むすび

当研究室で行っている音響信号処理や音声認識等の研究を効率的に行うために信号処理用高速演算装置 μKIDOCH を開発した．本演算装置はホスト計算機との間のデータ転送の仕方に特徴を持つ．すなわち，パイプライン化メモリにより処理を連続的に行う場合には実質的にデータ転送の時間が0とみなせるため，本演算装置とホスト計算機は高度の並列処理が可能となっている．現在のところ，ソフトウェアとしては，FFT とディジタルフィルタのみであるが，行列演算等を順次増やしていく予定である．さらに，現在，μKIDOCH 用のC言語風の高級言語を開発中であり，出来上がり次第発表する予定である．

謝辞 本演算装置を製作する際にいろいろ御協力を頂いた(株)小野測器の小野隆彦氏，前田利重氏および佐々木徹氏に感謝いたします．また，有益な討論をしていただいた東北大学城戸研究室の皆様に感謝いたします．

参考文献

1) TMS 32010 User's Guide, Texas Instruments (1983).
2) Hagiwara, Y., Kita, Y., Miyamoto, T., Toba, Y., Hara, H. and Akazawa, T.: A Single Chip Digital Signal Processor and Its Application to Real-Time Speech Analysis, *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-31, pp. 339-346 (1983).
3) μPD 77230, 日本電気(株).
4) COSMO 700 S システム説明書，三菱電機(株).
5) Abe, M., Nagata, Y. and Kido, K.: A New Method to Locate Sound Sources by Searching the Minimum Value of Error Function, *Proceedings of IEEE International Conference on ASSP*, 18 B. 2. 1, pp. 933-936 (1986).
6) 上田 隆，安倍正人，城戸健一：直線配列マイクロホンを用いた CCSM による音波到来方向推定の実験的検討，信学技報 EA 84-38 (1984).
7) Kanai, H., Abe, M. and Kido, K.: Detection of Slight Defects in Ball Bearings by Non-periodic Analysis, *J. Acoust. Soc. Jpn.(E)*, Vol. 7, No. 4, pp. 219-228 (1986).
8) Makino, S. and Kido, K.: Recognition of Phonemes Using Time-spectrum Pattern, *Speech Communication*, Vol. 5, pp. 225-237, North Holland (1986).
9) Abe, M., Kim, C. D. and Kido, K.: Investigation of the Effect of a Time Window on the Accuracy of an Estimated Impulse Response, *J. Acoust. Soc. Jpn.(E)*, Vol. 7, No. 5, pp. 269-277 (1986).
10) 高橋義造：並列処理マシン開発の現状，情報処理，Vol. 28, No. 1, pp. 10-18 (1987).
11) 富田真治：並列計算機構成論，昭晃堂，東京 (1986).
12) Bouknight, W. J. et al.: The ILLIAC IV System, *Proceedings of IEEE, 1972 April*, pp. 369-388 (1972).
13) Kuck, D. J. and Stokes, R. A.: The Burroughs Scientific Processor (BSP), *IEEE Trans.*, Vol. C-31, No. 5, pp. 363-376 (1982).
14) Hagiwara, H. et al.: A Dynamically Microprogrammable Computer with Low Level Parallelism, *IEEE Trans.*, Vol. C-29, No. 7, pp. 577-595 (1980).
15) Charlesworth, A. E.: An Approach to Scientific Array Processing: The Architectural Design of the AP-120 B/FPS-164 Family, *IEEE Comput.*, Vol. 14, No. 9, pp. 18-27 (1981).
16) 上田 隆，安倍正人，城戸健一：信号処理用高速演算装置のソフトウェア作成言語と目的プログラムの最適化について，第 32 回情報処理学会全国大会論文集，No. 5 S-10, pp. 271-272 (1986).
17) 堀越：コンピュータの高速演算方式，近代科学社，東京 (1980).
18) MMI PAL FAMILY, MMI ジャパン.
19) 三輪譲二，城戸健一：母音認識による分析フィルタの特性の検討，第8回東北大学応用情報学研究センターシンポジウム予稿集，pp. 53-62 (1983).

(昭和62年6月18日受付)
(昭和62年11月11日採録)

安倍 正人（正会員）

昭和27年生．昭和56年東北大学大学院工学研究科修了．現在同大学応用情報学研究センター助手．工学博士．音響ディジタル信号処理，計算機アーキテクチャなどの研究に従事．IEEE，電子情報通信学会，日本音響学会，日本騒音制御工学会各会員．

嶋 明弘

昭和34年生．昭和58年東北大学工学部電気工学科卒業．昭和60年同大学院修士課程（情報工学専攻）修了．現在同大学院博士課程在席．音響信号の計算機処理および信号処理用演算装置の研究に従事．日本音響学会会員．

上田 隆（正会員）

昭和35年生．昭和59年東北大学工学部機械工学第二学科卒業．昭和61年同大学院情報工学専攻修士課程修了．現在NTT無線システム研究所勤務．ディジタル移動通信の研究に従事．電子情報通信学会会員．

金井 浩（正会員）

昭和33年生．昭和56年東北大学工学部通信工学科卒業．昭和61年同大学院博士課程修了．工学博士．同年東北大学情報処理教育センター助手，現在に至る．音響・振動信号等のディジタル信号処理と機械系診断への応用に関する研究に従事．昭和62年度石川賞受賞．日本音響学会，電子情報通信学会，日本機械学会各会員．

牧野 正三

昭和22年生．昭和44年東北大学工学部電子工学科卒業．昭和49年東北大学大学院工学研究科修了．同年東北大学電気通信研究所助手．昭和55年東北大学応用情報学研究センター助手．昭和62年同助教授．音声の自動認識に関する研究に従事．日本音響学会，電子情報通信学会各会員．

城戸 健一（正会員）

1926年4月15日生．1948年3月東北大学工学部電気工学科卒業．東北大学電気通信研究所助手，同大学工学部助教授を経て，1963年同大学電気通信研究所教授，1976年応用情報学研究センター教授，センター長，現在に至る．音声自動認識，ディジタル信号処理とその応用に関する研究に従事．著書：音響工学(電子通信学会編，コロナ社)，ディジタル信号処理入門（丸善），電子計算機原論 上 下（丸善），過渡現象論（朝倉書店）等．工学博士．日本音響学会，電子情報通信学会，電気学会，計測自動制御学会，IEEE，AESなどの会員，アメリカ音響学会フェロー．