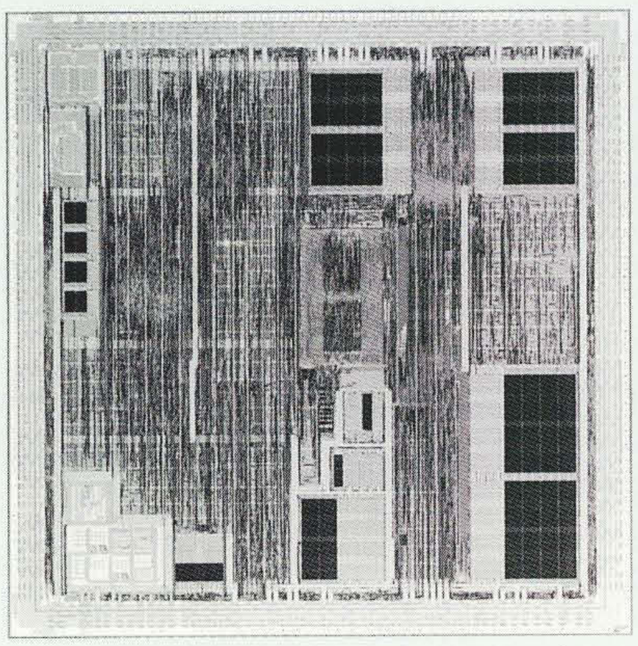
特集　ネットワーク時代を支える半導体

ネットワークを支えるプロセッサ

—高性能RISCマイコンとFIR性能—

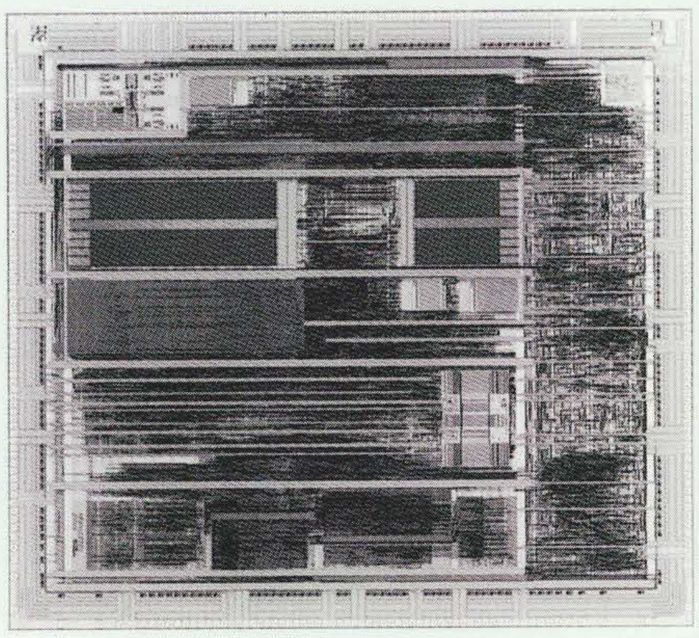
Processors for Use in Network Applications

海永正博　*Masahiro Kainaga*　斎藤靖彦　*Yasuhiko Saitô*
西本順一　*Jun'ichi Nishimoto*



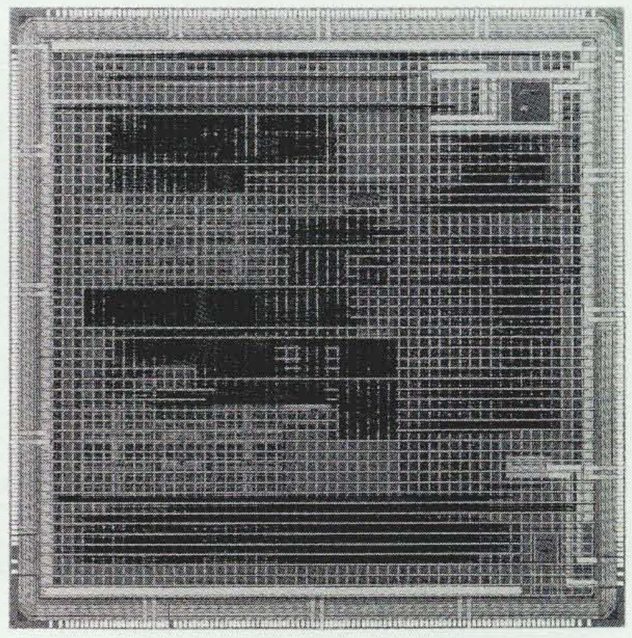
(a) SH-DSPのチップ写真

(SH7729：7.3 mm×7.3 mm)



(b) SH-4のチップ写真

(SH7751：7.0 mm×7.8 mm)



(c) SH-5の評価チップ写真

(9.5 mm×9.5 mm)

日立製作所の高性能RISCマイコン「SuperH RISC engineファミリ」

日立製作所が製品化しているマイコン群の例を示す。これらマイコンのCPUコア部分は，ネットワーク処理階層の上位部分の処理を引き受ける。アプリケーションが音声符号化・復号化などであれば，演算量が大量になり，低消費電力・高性能マイコンが要求される。

最近は，携帯電話にも動画像規格“MPEG4”のサポートが求められるようになった。音楽配信も行われるようになり，復号化だけでなく，符号化にもリアルタイム処理が求められている。ネットワークでも，マルチメディア情報の送信は珍しいことではなくなってきている。携帯電話応用でも，まずマルチメディア処理性能の高さがプロセッサに要求される。余裕を持って高速に処理することは，消費電力の削減につながる。

日立製作所は，高性能RISCマイコン「SuperH RISC engineファミリ」で，SH2-DSP，SH-3，SH3-DSP，SH-4，SH-5など低消費電力で幅広い性能範囲の製品群をそろえ，さまざまなネットワーク応用機器のニーズにこたえている。

1　はじめに

ネットワーク応用では，データの流れを管理する制御処理と，音声・画像等のマルチメディア情報に圧縮・伸長などを行うマルチメディア処理が必要である。

日立製作所は，低コストの制御処理向けマイコンとして「H8シリーズ」を，より高度な制御処理およびマルチメディア処理向けに高性能RISC(縮小命令セットコンピュータ)マイコン「SuperH RISC engineファミリ」をそれぞれ製品化している。

マルチメディア情報は，通常，整数または固定小数点数で表現される。例えば，音声符号化規格G723.1は固定小数を規定し，演算では誤差を許さない。しかし，内部処理は，場合によっては浮動小数点数で行うほうがよいこともある。例えば，G728は浮動小数の演算も規定されている。したがって，浮動小数演算が可能なプロセッサで処理するほうが，演算あふれを気にすることなく処理できるので好都合な場合もある。

SuperHシリーズの中で，SH-DSPは，SH CPU本体での整数演算に加え，固定小数も高速演算できるように，16ビット固定小数点DSP(Digital Signal Processor)が付加されている。SH-4は3D画像処理などの高速化のために浮動小数点ユニット(FPU)が付加されている。またSH-5は，DSP機能を並列実行する64ビットSIMD演算器と，浮動小数点ユニットの両方を備え，SH-DSPおよびSH-4の次世代製品として提供できる。各マイコンの性能

表1　H8シリーズおよびSuperH RISC engineファミリの性能諸元

日立製作所は，低コストの制御処理向けマイコン「H8シリーズ」と，高性能RISCマイコン「SuperH RISC engineファミリ」を製品化している。

	H8 (H8/300L)	SH-DSP (SH7729R)	SH-4 (SH7750S)	SH-5
周波数(MHz)	—	200	200	400
整数/固定小数点性能(GOPS)	—	0.4	0.4	3.2
浮動小数点性能(GFLOPS)	—	—	1.4	2.8
消費電力	(400 μA/1 MHz)	1W	1W	＜2W

諸元を**表1**に示す。

H8シリーズの中でも特にH8/300Lは1.8 V低電圧動作，時計動作待機時2.8 uAtypといった低消費電力動作を実現し，電池駆動応用で評価されている。

SuperHシリーズでは，前述の各種ハードウェアを活用し，プロセッサの性能を最大限に引き出すことにより，動作周波数および動作電圧の低減を通じて消費電力の削減が可能となる。一般のソフトウェア技術者がハードウェアの詳細まで立ち入らずに最適なプログラムを開発するには，ハードウェアとソフトウェアの双方に立ち入って最適化した重要ルーチンを提供することが有効である。このような最適化によって速度が数倍になることもまれではない。

ここでは，ネットワーク応用の携帯電話，STB，ネットPDA，IP電話などの端末で中心となるマルチメディア処理の重要ルーチンであるFIR(Finite Impulse Response：有限インパルス応答)とFFT(Fast Fourier Transform：高速フーリエ変換)に関して，ソフトウェアの最適化によってSH-DSP，SH-4，SH-5の性能を最大限引き出した例について述べる。なお極力，C言語をベースに説明することとする。

2　マルチメディア処理

マルチメディア処理の典型的な二つの例について以下に述べる。

2.1　FIR：有限インパルス応答

一つは，内積またはFIRフィルタの処理である。これは，**図1**の演算に相当する。

この種の演算は信号処理の基本であり，例えば，音声符号化規格やオーディオ符号化規格では各所に使用されている。

```
sum=c[0]＊x[0]+c[1]＊x[2]+c[2]＊x[2]+..+c[99]＊x[99]
```

FIR：信号処理の基本,単に内積。ただし入力xをシフトして何度も計算

(a) FIR1

```
sum0=c[0]＊x[0]+c[1]＊x[1]+c[2]＊x[2]+..+c[99]＊x[99]
sum1=c[0]＊x[1]+c[1]＊x[2]+c[2]＊x[3]+..+c[99]＊x[100]
sum2=c[0]＊x[2]+c[1]＊x[3]+c[2]＊x[4]+..+c[99]＊x[101]
sum3=c[0]＊x[3]+c[1]＊x[4]+c[2]＊x[5]+..+c[99]＊x[102]
```

BlockedFIR：FIRを(例えば四つ)まとめて計算。ロード数を$\frac{1}{4}$にできる。

(b) FIR4

図1　FIRフィルタ

FIRフィルタは，音声信号波形を加工する際などに使用される。畳み込みと呼ばれる内積型の演算である。要素数が100から200となることも多く，高速処理が重要になる。

2.2　FFT：高速フーリエ変換

もう一つは，離散フーリエ変換である。これは，時間領域の信号系列を周波数領域の信号系列に直交変換するものである。したがって，定義式は，N*N正方行列とN点ベクタの積になる。定義式どおりであると，Nの2乗個の(複素)乗算と(複素)加算が必要で，Nの増大に伴って演算量も増えていく。これを$N^*\log_2 N$に抑えるものが，有名な高速フーリエ変換FFTである。

16点の場合の，周波数間引き法と呼ばれるFFT演算方法を**図2**に示す。

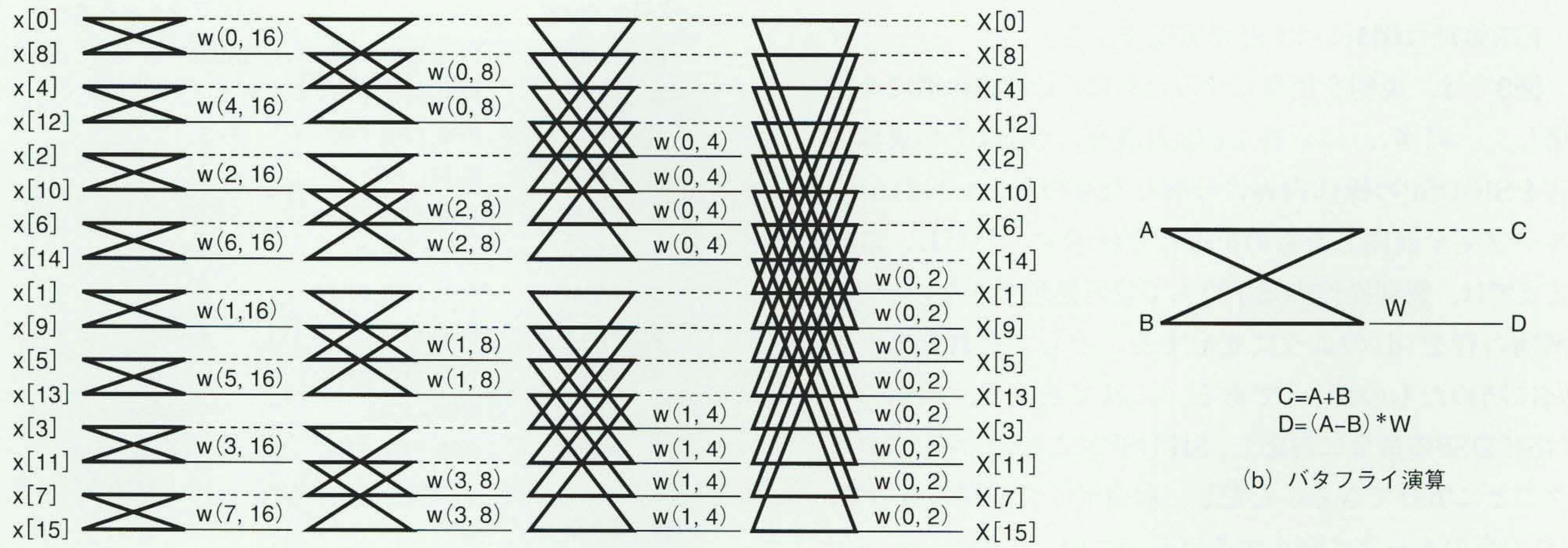
この演算は，「バタフライ」と呼ばれる基本演算だけで構成される。**図2**(b)の部分を90度回転すると「ちょう」に似ているので，バタフライ演算と呼ばれる。これは，複素数の加算と減算と乗算で構成されている。

C=A+B

D=(A−B)*W

このFFTは，AAC(Adaptive Audio Coding)では符号化や復号化に使用されている。実際には，離散フーリエ変換でなく離散コサイン変換の部分もある。しかし，離散コサイン変換はFFTを使用して実現することも可能であり，AACではそのやり方を採用している。このため，AACではFFTがその処理の主要部を占め，その高速化が重要である。

この二つの代表アルゴリズムの高速実現法と結果の処理性能について，SH-DSP，SH-4，およびSH-5ごとに以下に述べる。周波数の違いを無視して，処理のクロック数で示す。



(a) 16点FFT

(b) バタフライ演算

図2　FFTとバタフライ演算

FFTは，離散フーリエ変換の演算量を劇的に削減する。1,024点信号の場合は，約$\frac{1}{100}$になる。

3　SH-DSPとFIR性能

SH-DSPは，SuperHマイコンにDSP機能を追加したものであり，SH2-DSPとSH3-DSPの2種がある。キャッシュ・MMUがSH2-DSPでは無く，SH3-DSPでは有るのが特徴である。DSP機能とは，加算，乗算，および演算のためのデータロードをクロックごとに実行できるようにしたものと考えればよい。

SuperHマイコンは16ビット固定長RISCマイコンであるが，DSP命令は32ビット命令で，加算，乗算と二つのロードを一つの命令で実行できる。Cの構文を借用すると，以下のように表すことができる。

a0+=m0；m0=x0*y0；x0=*r4++；y0=*r6++；

この四つの操作を1命令で指定でき，クロックごとに実行していくことができる。

また，ループオーバヘッドを0にするrepeat(繰り返し)命令も導入している。

以上の2点の活用法について，FIRを例にして以下述

```
long a0;            /* _accum a0 : */
short  *r4, *r6;  /* _fixed *r4, *r6; */
for(i=0;i<100;i++) {
     a0+=*r4++  r6++;
}
```
(a)初期コード

```
long a0;            /*  _accum a0 : */
short  *r4, *r6;  /*  _fixed *r4, *r6; */
short x0, y0;       /*  _fixed x0, y0  */
long m0;            /*  _fixed m0;  */
for(i=0;i<100;i++) {
     x0=*r4++; y0=*r6++; m0=x0*y0; a0+=m0;
}
```
(b)演算を分解したコード

```
x0=*r4++; y0=*r6++; m0=x0*y0; a0+=m0;
x0=*r4++; y0=*r6++; m0=x0*y0; a0+=m0;
x0=*r4++; y0=*r6++; m0=x0*y0; a0+=m0;   計100個
```
(c)ループ展開コード

```
                        x0=*r4++; y0=*r6++;
          m0=x0*y0;
a0+=m0;
```
(d)個別行の変形

```
                                  x0=*r4++; y0=*r6++;
                        m0=x0*y0; x0=*r4++; y0=*r6++;
a0+=m0; m0=x0*y0; x0=*r4++; y0=*r6++;

a0+=m0; m0=x0*y0; x0=*r4++; y0=*r6++;
a0+=m0; m0=x0*y0;
a0+=m0;
```
(e)ループ展開＋パイプライニング

```
                        x0=*r4++; y0=*r6++;
          m0=x0*y0; x0=*r4++; y0=*r6++;
for(i=0;i<98;i++) {
   a0+=m0; m0=x0*y0; x0=*r4++; y0=*r6++;
}
   a0+=m0; m0=x0*y0;
   a0+=m0;
```
(f)パイプライニング

```
repeat LoopStart,Loopend
        setrc  #98
   Loopstart:最初の命令
   Loopend:最後の命令
```
(g)repeat命令の使用例

図3　SH-DSPでのFIR演算(C言語で説明)

SH-DSPはrepeat(繰り返し)命令をサポートしている。(f)のfor文に対応する部分をrepeat命令を使用することでFIR処理を高速化できる。

べる。

FIR処理は**図3**に示す形で実現できる。

図3では，説明を簡単にするために，C言語の構文を借用した。同図(a)は素朴なFIR処理を，(b)はその演算内容をSH-DSPの操作内容に分解したものを，(c)は(b)をループ完全展開したものをそれぞれ示す。ただし，このままでは，個別操作の完了待ちで並列処理できないので，個別の行を(d)のように変形する。そしてそれを上下方向に詰めたものが(e)である。これであると，個別の行はSH-DSPの命令に対応し，SH-DSPはこの命令を1クロックごとに実行できる。ただし，命令サイズが膨らむので，(f)の形のループに修正するほうがよい。この際，repeat命令を使用する。repeat命令の使用例を同図(g)に示す。ループ進入の前に，ループ範囲と繰り返し数を設定する。ループに進入すると，繰り返し数の更新と条件分岐処理は，プログラムで明示される表の演算処理と並列に自動処理される。したがって，ループオーバヘッドは0となる。結局，二つのロードと一つの乗算と一つの加算を，クロックごとに実行できることになる。

4　SH-4とFIR性能

SH-4はスーパスカラをサポートし(2並列)，SH-3に浮動小数演算機能を追加したものである。浮動小数レジスタは表(プログラムに明示される)に16本と裏(暗黙に使用される)に16本がそれぞれ用意されている。裏のレジスタを4*4正方行列，表の四つの連続レジスタを4要素ベクタと考え，線形変換を行う命令(FTRV)を用意している。この命令は4クロックピッチで実行でき，さらにスーパースカラ方式によってデータのロード·ストアと並列実行できるので，最大1.4 Gflops(浮動小数演算毎秒，200 MHz時)の演算性能となる。FTRV命令は16個の乗算と12個の乗算を4クロックピッチに行うので，(16+12)/4*200 MHz=1.4 Gflopsとなるからである。

FIR処理は，SH-4の場合，一括処理(**図1**(b)参照)のほうが性能を多く引き出せる。アセンブリコーディング(線形変換命令FTRVまたは4要素ベクタ間内積命令FIPR使用)では800 Mflops程度の性能となる。また，Cコーディングであっても400 Mflopsを実現することができる。

性能を引き出すようにCコーディングした一括処理(FIR4)のコンパイル結果での，機械語の実行をトレースしたものを**図4**に示す。

図4に示すように毎クロック積和命令FMACが実行されており，400 Mflops(200 MHz時)の性能が達成されている。一括処理の**図1**(b)では，データアクセスが重複している。この重複を有効活用してデータロードの数を削減し，積和命令FMACを毎クロック実行できるループ構造にできたからである。

クロック	命令		パイプラインステージ
0	FMOV.S	FR6, FR0	EAS
0	FMAC	FR0, FR7, FR1	F--S
1	FMAC	FR0, FR9, FR2	F--S
1	FMOV.S	@R6+, FR7	EAS
2	FMAC	FR0, FR4, FR3	F--S
2	FMOV.S	@R5+,FR10	EAS
3	FMAC	FR0, FR8, FR12	F--S
3	FMOV.S	FR5, FR0	EAS
4	FMAC	FR0, FR9, FR1	F--S
4	FMOV.S	@R6+,FR9	EAS
5	FMAC	FR0, FR4, FR2	F--S
.....			
14	BF/S	L271	E
15	FMAC	FR0, FR4, FR12	F--S

図4　SH-4でのFIR4処理
FIR4処理をパイプラインシミュレータで表示している。右端の数字がクロックを示す。個別のクロックごとにFMAC命令が実行されることがわかる。

5　SH-5とFIR性能

SH-5では，SIMD(単一命令複数データ)方式の新規(命令セット)アーキテクチャを採用している。SH-4互換モードもサポートしており，SH-4のプログラムも走行できる。汎用レジスタは，64ビット幅で64本を用意している。64ビットのレジスタを16ビット四つ，または8ビット八つに分割することにより，個別のフィールドごとに同一の演算を実行することができる。これがSIMDの特徴である。

演算命令には，フィールドごと加算やフィールドごとの比較などがある。これらは，ベクタ演算に分類できる(**図5**参照)。

上記のほかに，還元(reduction)型の命令もサポートしている。還元命令とは，ベクタとベクタに作用し，スカラを生成する命令のことである。代表例は内積である。例えば，2バイトごとの積和命令MMULSUM.WQでは，四つの乗算と三つの加算と一つの累算(加算)を行って一つのスカラ値を生成する(**図6**参照)。

なお，SH-5では分岐高速化のくふうをしている。分岐アドレスを登録する命令と実際に分岐する命令の二つに分岐命令を分割したことにより，ループオーバヘッドを

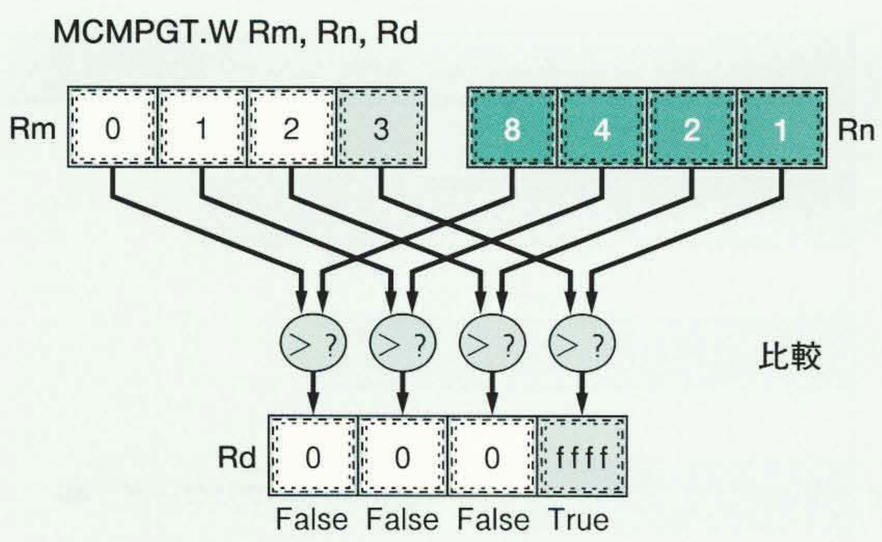


図5　ベクタ型SIMD命令の例

64ビットレジスタを16ビット四つに分割し，それぞれに大小比較し，それらの結果を出力レジスタに設定している。

削減することができる。

また，2バイトごとの積和命令を有効に活用すれば，**図1**(a)のFIR1を高速に実行することができる。ここでは，Cでのコーディング法を示す。組み込み関数(sh_media_MMULSUM_WQ)を使用する場合の方法を**図7**に示す。

図7のFIR1_1は高速化をねらったもので，4倍にループ展開している。forループ内で，64ビットのデータをロード後，ある組み込み関数を呼んでいる。この組み込み関数は，MMULSUM.WQ命令に置き換わる。同図のFIR1_1のアセンブリ オブジェクト コードを**図8**に示す(ループ本体だけ)。

現状のSH-5ではスーパースカラを採用してないので，

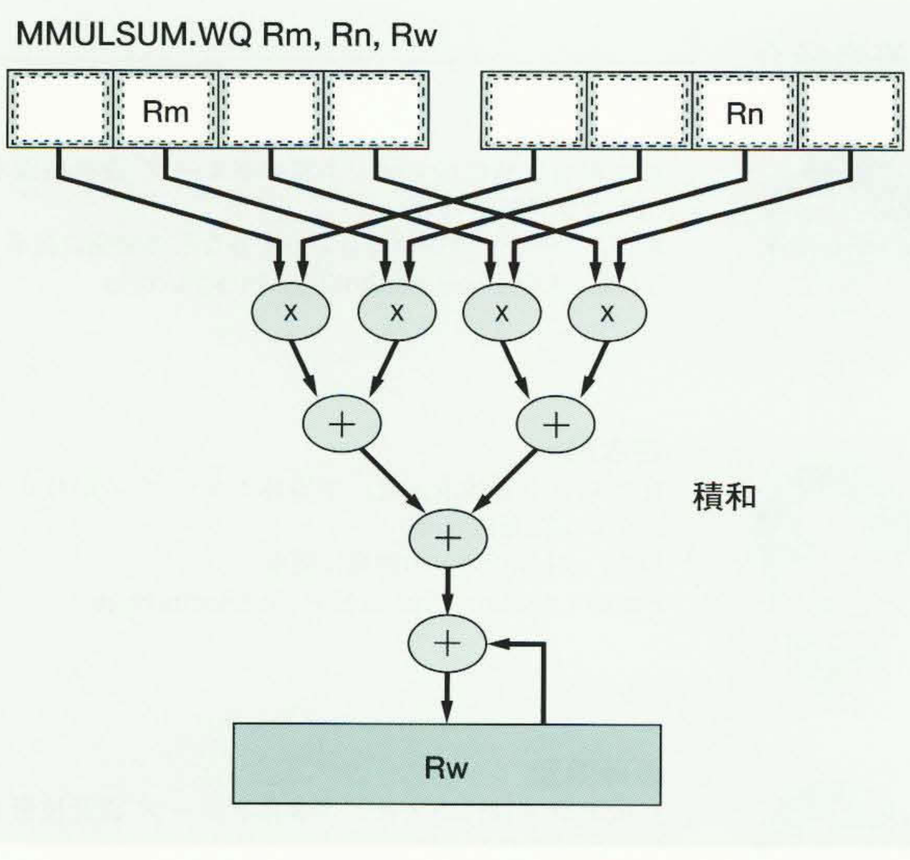


図6　還元型SIMD命令の例

64ビットレジスタを16ビット四つに分割し，おのおので乗算を行い，それらを加算し，さらに出力レジスタの内容に足し込んでいる。出力は一つのスカラ値である。

```
#include<shmedia.h>
long long FIR1_2(long long*cp0,long long*ip0)
{     int i;
      long long t0,t1,sum=0;
      long long *cp1=cp+8,cp2=cp+16,cp3=cp+24;
      long logn *ip1=ip+8,ip2=ip+16,ip3=ip+24;

      for(i=0;i<8;i++) {
            t0=cp0[i];
            t1=ip0[i];
            sum=sh_media_MMULSUM_WQ(t0,t1);
            t0=cp1[i];
            t1=ip1[i];
            sum=sh_media_MMULSUM_WQ(t0,t1);
            t0=cp2[i];
            t1=ip2[i];
            sum=sh_media_MMULSUM_WQ(t0,t1);
            t0=cp3[i];
            t1=ip3[i];
            sum=sh_media_MMULSUM_WQ(t0,t1);
      }
      return sum;
}
```

図7　SH-5向けFIRコード

SH-5のマルチメディア専用命令を活用する組み込み関数使用例を示す。sh_media_MMUL_SUM()の呼び出しが，MMULSUM命令に置き換えられる。

1命令に1クロックピッチを要する。**図8**からわかるように，ループオーバヘッドはループの4倍展開で軽減されている。さらに，分岐命令のうちの実際に分岐する命令だけがループ内に置かれており，この命令も1クロックで実行するので，ループ内の命令数(15個)と同じクロック数でFIR処理を行えることになる。積16個，加算16個を15クロックで処理することができる。

```
L12:  LD.Q      R2,0,R6
      LD.Q      R3,0,R8
      MMULSUM.WQ R6,R8,R7
      LD.Q      R2,8,R6
      LD.Q      R3,8,R8
      MMULSUM.WQ R6,R8,R9
      LD.Q      R2,16,R8
      LD.Q      R3,16,R7
      MMULSUM.WQ R8,R7,R9
      LD.Q      R2,24,R8
      LD.Q      R3,24,R7
      ADDI.L    R2,8,R2
      ADDI.L    R3,32,R3
      MMULSUM.WQ  R8,R7,R9
      BGTU  R5,R2,TR4
```

図8　FIR1_2のアセンブリ オブジェクト コード(ループ本体だけ)

マルチメディア命令と高速分岐命令を有効活用し，高速処理が実現されている。

C.r=A.r+B.r
C.i=A.i+B.i
D.r=(A.r−B.r)*W.r−(A.i−B.i)*W.i
D.i=(A.i−B.i*W.r+(A.r−B.r)*W.i

(a) 定義(実数表現)

$$\begin{pmatrix} C.r \\ C.I \\ D.r \\ D.i \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ W.r & -W.i & W.r & -W.I \\ W.I & W.I & -W.I & -Wi \end{pmatrix} * \begin{pmatrix} A.r \\ A.i \\ B.r \\ B.i \end{pmatrix}$$

(b) 行列表現

図9　バタフライ演算の例
複素数の演算を実数演算で表現した。(b)の演算は，四次元線形変換命令FTRV命令の機能に対応する。

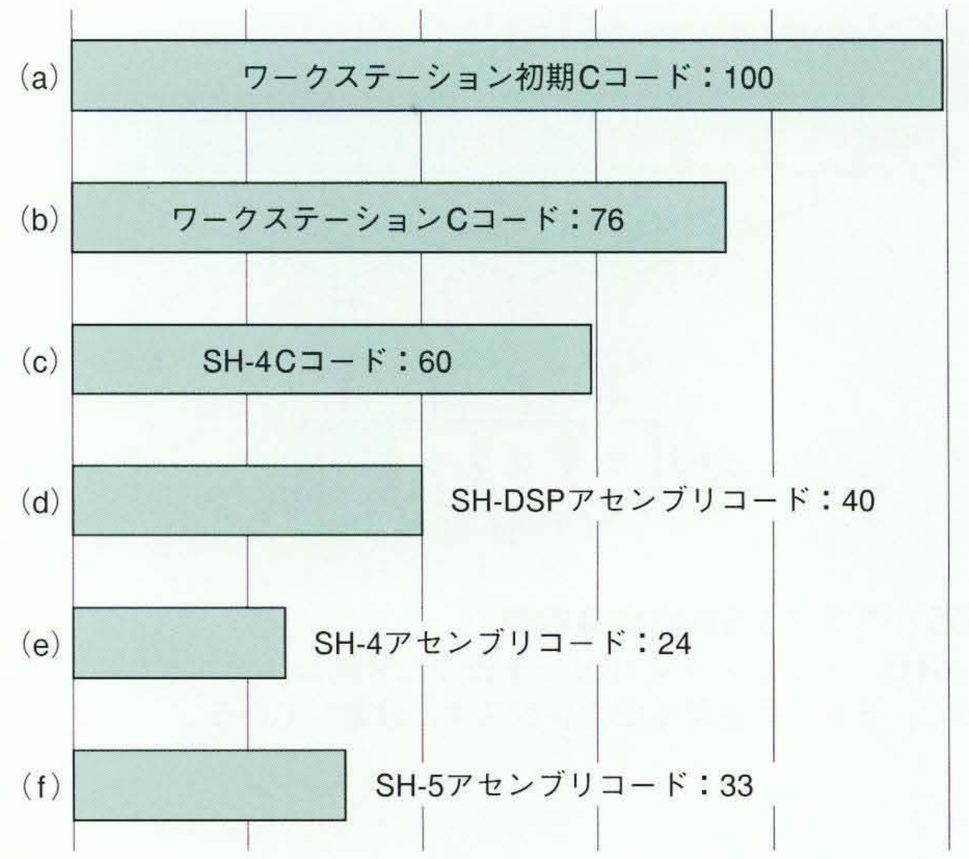


図10　離散フーリエ変換"FFT"の所要クロック数(相対値)
教科書的なFFTソースコードをあるワークステーションで走行させた場合のクロック数を100として，各種のプロセッサの性能を比較した。

6　FFT性能

FFTの性能は，結局，**図2**(b)のバタフライ演算を何クロックピッチで実行していけるかで決まる。

高速処理を実現するためには，コーディング上のくふうが要求され，処理も複雑になる。そこで，高速化の発想を容易に理解できるSH-4の場合についてだけ以下に述べる。

FFTのバタフライ演算を実数演算に書き換えたものを**図9**(a)に示す。これを行列表現すると同図(b)になり，四次元ベクタの線形変換になる。したがって，四次元線形変換命令FTRVが有効に適用される。

FFTの処理性能を**図10**に示す。同図(a)は，基本的なCコードをあるワークステーション(2並列スーパスカラ)でコンパイルし，走行させた場合のクロック数である。これを100として正規化した。(b)は，コーディングにくふうした場合のワークステーションクロック数で，76と高速になった。(c)はそのCコードをSH-4で走行させた場合のもので，もっと速い60である。(d)は，SH-DSPのアセンブリコーディングのもので40，(e)はSH-4のアセンブリコーディングのもので24，(f)はSH-5のもので33である。なお，(c)はAACの実システムに使用されている。

7　おわりに

ここでは，日立製作所の高性能RISCマイコンのうち，SH-DSP，SH-4，SH-5のネットワークとマルチメディア応用への適合性についてプログラマの観点から分析し，マルチメディア応用の二つの典型演算(FIRとFFT)へのこれらマイコンの対処法と処理性能について述べた。

日立製作所は，今後も，これら高性能マイコンのコストと消費電力のいっそうの低減を図っていく考えである。

参考文献

1) 海永：RISCマイコン Super Hシリーズ，bit, Vol. 31, No. 5 (1999)

執筆者紹介

海永正博
1973年日立製作所入社，半導体グループ 設計技術本部 CPUコア開発部 所属
現在，アーキテクチャ，コンパイラなどの開発に従事
E-mail：kainaga-masahiro @ sic. hitachi. co. jp

西本順一
1992年日立製作所入社，半導体グループ 設計技術本部 CPUコア開発部 所属
現在，SH-DSPコアの開発に従事
E-mail：nishimoto-juninchi @ sic. hitachi. co. jp

斎藤靖彦
1992年日立製作所入社，半導体グループ 設計技術本部 CPUコア開発部 所属
現在，アーキテクチャ，コンパイラなどの開発に従事
E-mail：saito-yasuhiko @ sic. hitachi. co. jp