RX ファミリ
DSP 機能命令を活用した行列演算プログラム

R01AN0227JJ0100
Rev.1.00
2011.03.14

要旨

この文書は、RX ファミリの DSP 機能命令を使用した行列演算プログラムを説明します。

動作確認デバイス

RX ファミリ

目次

1. はじめに

RX ファミリ CPU コア (以下 RX と略) は 16 ビット × 16 ビットの積和器を搭載しています。乗算を用いた演算式やアドレス計算で通常に用いる 32 ビット × 32 ビットの整数乗算命令 (MUL 命令) は、32 ビット × 32 ビットの演算結果 64 ビットのうち下位 32 ビットをその演算結果とします。つまり、MUL 命令の使用にあたっては、演算結果が 32 ビットを越えないという前提があります。ところが、数値データを固定小数点表現 (例えば、[1]を参照ください) で表す場合、乗算あるいは積和演算の結果のうち、有効なデータは上位側に位置するのが普通です。そのため、固定小数点表現を用いた数値データの乗算あるいは積和演算の場合に MUL 命令を用いていたのでは、演算結果が 32 ビット以内に納まる場合しか用いることができず、狭い範囲の数値データしか表現できなくなるという問題が発生します。この問題を解決するために、RX は 48 ビットのアキュムレータによる積和演算命令 (または乗算命令)、アキュムレータに格納された値の丸め演算を実行する命令、およびアキュムレータと汎用レジスタ間のデータの転送命令をサポートしています。これらの積和演算命令や丸め命令等を組み合わせることで、固定小数点表現を用いた数値データの種々の演算を高速に実現でき、DSP に匹敵するデータ処理能力を実現することができます。RX の積和演算命令の詳細は「RX ファミリ ユーザーズマニュアル ソフトウェア編 (RJJ09B0465)」を参照ください。アプリケーションノート「積和演算命令の活用方法」(R01AN0254JJ) では、これらの積和演算命令や丸め命令の使い方を説明しています。また、アプリケーションノート「積和演算の組み込み関数活用方法」(R01AN0255JJ) では、これらの積和演算命令や丸め命令を RX ファミリ C/C++コンパイラ (以下コンパイラと略) の拡張機能である組み込み関数（コンパイラ V1.01 からサポートされます）から活用する方法を説明しています。

本アプリケーションノートでは、RX の積和演算命令を活用した行列演算プログラムについて説明します。具体的には、N 行 N 列の正方行列を演算の対象とします。演算の種類としては加算、減算、乗算、逆行列を取り上げます。ただし、逆行列については、説明を簡単にするために 2 行 2 列の正方行列のみを対象とします。また、プログラム例では RX の積和演算命令をコンパイラの組み込み関数から活用する方法をとります。

【注】 [1] 森、名取、鳥居; "岩波講座　情報科学-18　数値計算", pp.1-27, 岩波書店, (1982)

2. 行列のデータ表現

本章では行列のデータ表現について説明します。本アプリケーションノートでは N 行 N 列の正方行列を演算の対象としますが、プログラム例では主に N = 2 と N = 4 の場合について説明します。逆行列の計算では N = 2 の場合、つまり 2 行 2 列の正方行列を対象とします。

RX の積和演算命令は 16 ビットの符号付きデータを対象としているので、行列の成分は 16 ビット符号付きデータで表現し、行列をこの 16 ビットデータの一次元配列で表現することにします。例えば、4 行 4 列の正方行列は、要素数が 4 × 4 = 16 の short 型の一次元配列で表現します。また、2 行 2 列の正方行列は、要素数が 2 × 2 = 4 の short 型の一次元配列で表します。以上の考え方にもとづく定義を次のプログラム例に示します。

```
#define N       4       /* 計算対象の正方行列の行数と列数 */
typedef int16_t Matrix;
typedef int16_t Matrix44[4 * 4];  /* 4 行 4 列の正方行列 */
typedef int16_t Matrix22[2 * 2];  /* 2 行 2 列の正方行列 */
```

この行列のデータ表現では行列の行ベクトルの成分が連続したアドレスに配置されます。これに対して、行列の列ベクトルの成分は飛び飛びのアドレスに配置されることに注意してください。以下、行列の成分が一次元の配列のどこに対応するかを具体例で説明します。

まず、2 行 2 列の正方行列

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

の成分は、次の順番 (左から右) で一次元配列に格納されます。

$$a,b,c,d$$

次に、4 行 4 列の正方行列

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

の成分は、次の順番 (左から右) で一次元配列に格納されます。

$$a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p$$

最後に、具体的なプログラム例をいくつか示します。次の行列

$$A = \begin{bmatrix} 2 & -7 \\ -1 & 3 \end{bmatrix}$$

は、次のプログラムに対応します。

```
Matrix22 A = {
    2, -7,
   -1,  3,
};
```

次に示す行列 B は、

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ -1 & -2 & -3 & -4 \\ -5 & -6 & -7 & -8 \end{bmatrix}$$

次のプログラムに対応します。

```
Matrix44 B = {
    1,  2,  3,  4,
    5,  6,  7,  8,
   -1, -2, -3, -4,
   -5, -6, -7, -8,
};
```

3. 行列の加減乗算

本章では N 行 N 列の正方行列の加算、減算、乗算について説明します。

3.1 加算

行列の加算は、2 つの行列の対応する成分同士を加算します。例えば、2 行 2 列の正方行列の加算は次の通りです。

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

N 行 N 列の正方行列の加算プログラム matrix_add を次に示します。matrix_add は 2 つの行列 a、b (いずれも配列の先頭アドレス) を引数にとり、a と b の和の計算結果を a に設定します。

```
/*
  N 行 N 列の正方行列の和の計算
  結果は a に設定する
*/
void matrix_add (Matrix a[], Matrix b[])
{
    int i;

    for  (i = 0; i < N * N; i++)  {
        a[i] += b[i];
    }
}
```

3.2 減算

行列の減算は、加算と同様に、2 つの行列の対応する成分同士の差を計算します。例えば、2 行 2 列の正方行列の減算は次の通りです。

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a-e & b-f \\ c-g & d-h \end{bmatrix}$$

N 行 N 列の正方行列の加算プログラム matrix_sub を次に示します。matrix_sub は 2 つの行列 a、b (いずれも配列の先頭アドレス) を引数にとり、a と b の差の計算結果を a に設定します。

```
/*
  N 行 N 列の正方行列の差の計算
  結果は a に設定する
*/
void matrix_sub (Matrix a[], Matrix b[])
{
    int i;

    for  (i = 0; i < N * N; i++)  {
        a[i] -= b[i];
    }
}
```

3.3　乗算

行列の乗算は、加減算と比較するとやや複雑な計算が必要になります。行列の乗算は、同じ順番にある左側の行列の行ベクトルの成分に、右側の行列の列ベクトルの成分を掛けてから足して計算します。ここで積和演算を活用することができます。例えば、2 行 2 列の正方行列の乗算は次の通りです。

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a \times e + b \times g & a \times f + b \times h \\ c \times e + d \times g & c \times f + d \times h \end{bmatrix}$$

ここではN行N列正方行列同士の乗算を考えているので、乗算結果は同じN行N列正方行列になります (上の例では乗算結果は 2 行 2 列の正方行列になります)。

N 行 N 列の正方行列の乗算プログラム matrix_mul を次に示します。matrix_mul は 3 つの行列 a、b、res (いずれも配列の先頭アドレス) を引数にとり、a と b の積を res に設定します。

```
#include <machine.h>

/*
  N 行 N 列の正方行列の積の計算
  結果は res に設定する
*/
void matrix_mul (Matrix a[], Matrix b[], Matrix res[])
{
    int i, j;
    int16_t *q = b;
    int16_t col[N];

    for  (i = 0; i < N; i++)  {
        for  (j = 0; j < N; j++)  {
            col[j] = q[N * j];
        }
        q++;
        for  (j = 0; j < N * N; j += N)  {
            res[j] = (int16_t) macl (a + j, col, N) ;
        }
        res++;
    }
}
```

このプログラムでは、効率よく計算を実行するために 2 つの工夫をしています。まず、RX の積和演算 (組み込み関数 macl) を利用するために、乗算記号の右側の行列の列ベクトルの成分を作業用配列 col にコピーして連続したアドレスに配置してから、組み込み関数 macl を呼び出しています。次に、作業用配列 col にコピーした列ベクトルを有効に使い回すために、乗算の外側ループを列ベクトルの順番として処理しています。これに対して、乗算の外側ループを行ベクトルの順番、内側ループに列ベクトルの順番とした場合は、毎回同じ列ベクトルを作業用配列にコピーしなければなりません。

なお、上に示した乗算プログラムは、N 行 N 列の正方行列一般について正しく動作するように書かれています。もし、N がある特定の小さな数に決まっている場合は、プログラムの内側ループを展開 (loop unrolling) することで、より速いプログラムに書き直すことが可能です。

上のプログラムを N = 4 の場合、つまりに書き直した例を次に示します。この例は 4 行 4 列用ですが、5 行 5 列あるいは 6 行 6 列などについても同じ方法で書き換えることができます。

```
#if N == 4
/*
  4 行 4 列の正方行列の積の計算 (N == 4 のとき専用)
  結果は res に設定する
*/
void matrix44_mul (Matrix a[], Matrix b[], Matrix res[])
{
    int i;
    int16_t *q = b;
    int16_t col[N];

    for  (i = 0; i < N; i++)  {
        col[0] = q[N * 0];
        col[1] = q[N * 1];
        col[2] = q[N * 2];
        col[3] = q[N * 3];
        q++;
        res[0 * 0] = (int16_t) macl (a + N * 0, col, N) ;
        res[N * 1] = (int16_t) macl (a + N * 1, col, N) ;
        res[N * 2] = (int16_t) macl (a + N * 2, col, N) ;
        res[N * 3] = (int16_t) macl (a + N * 3, col, N) ;
        res++;
    }
}
#endif /* N == 4 */
```

4. 逆行列の計算

本章では逆行列の計算について説明します。一般に、逆行列を計算することは簡単ではありません。特に組み込みシステムでは、計算によってその場で逆行列を計算することは効率的ではありませんので、可能な限り事前に逆行列を計算しておき、アプリケーションでは求めておいた逆行列と乗算を使って必要な計算をしてください。

たとえば、A、B を行列 (あるいはベクトル) とするとき、次の計算

$$A^{-1} \times B$$

をしなければならないとします。このとき A の逆行列をプログラムで計算するのではなく、あらかじめ

$$C = A^{-1}$$

となる行列 C を計算しておいた上で、アプリケーションでは次のように

$$C \times B$$

単なる乗算として処理するのが効率的です。

なお、2 行 2 列の正方行列については逆行列を求める公式がありますので、逆行列を計算するプログラム例を以下に示します。

4.1 2 行 2 列の正方行列の逆行列の計算

2 行 2 列の正方行列

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

の逆行列A^{-1}は、次の公式で与えられます。

$$\mathbf{A}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

ただし、次の条件を満たす場合にのみ が逆行列を持つことに注意してください。

$$ad - bc \neq 0$$

この公式を使って 2 行 2 列の正方行列の逆行列を計算するプログラム matrix22_inv を次に示します。matrix22_inv は、2 行 2 列の正方行列 a の逆行列を計算し res に格納します。正常終了の場合、戻り値として 0 を返しますが、逆行列が存在しない場合は戻り値として–1 を返します。なお、このプログラムは行列の成分を符号付き整数と見なして計算するので、入力データの値によっては正しい結果が得られない場合があることに注意してください。

```
/*
  2x2 の正方行列の逆行列の計算
  結果は res に設定し、戻り値として 0 を返す
  ただし、逆行列が存在しない場合は-1 を返す
 */
int matrix22_inv (Matrix a[], Matrix res[])
{
    int32_t det = (int32_t)a[0] * (int32_t)a[3] - (int32_t)a[1] * (int32_t)a[2];

    if (det == 0) {
        return -1;      /* エラー：逆行列は存在しない */
    }
    res[0] = (int16_t)( a[3] / det);
    res[1] = (int16_t)(- a[1] / det);
    res[2] = (int16_t)(- a[2] / det);
    res[3] = (int16_t)( a[0] / det);
    return 0;
}
```

5. サンプルプログラム

本アプリケーションノートで説明した行列の乗算と逆行列を計算するプログラムを使った例を次に示します。この例では、4 行 4 列の行列の積と 2 行 2 列の逆行列の計算を実行します。

```
static Matrix44 a = {
     1,  2,  3,  4,
     5,  6,  7,  8,
    -1, -2, -3, -4,
    -5, -6, -7, -8,
};

static Matrix44 b = {
    5,  6,  7,  8,
    1,  2,  3,  4,
    5,  6,  7,  8,
    1,  2,  3,  4,
};

static Matrix22 c = {
     2, -7,
    -1,  3,
};

void main (void)
{
    Matrix44 d;
    Matrix22 e;

    /* 行列 a と行列 b の積を d に格納 */
    matrix_mul (a, b, d) ;
#if N == 4
    /* 行列 a と行列 b の積を d に格納 (4x4 専用版)  */
    matrix44_mul (a, b, d) ;
#endif
    /* 行列 c の逆行列を e に格納 */
    matrix22_inv (c, e) ;
}
```

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

すべての商標および登録商標は，それぞれの所有者に帰属します。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.03.14	—	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

© 2011 Renesas Electronics Corporation. All rights reserved.

Colophon 1.0