プログラミング-言語・基礎・実践- 16－2
(1994. 3. 9)

SIMD 型超並列計算機用 HPF の実装とその性能評価

小前 晋 * 杉森 英夫 * 大谷 浩司 † 渦原 茂 ‡ 安村 通晃 ‡
* 住友金属工業株式会社 † 有限会社アックス ‡ 慶應義塾大学
〒 252 藤沢市遠藤 5322 慶應義塾大学環境情報学部安村研究室

あらまし

科学技術計算分野のプログラミングには，FORTRAN 言語が使われていることが多いため，超並列計算機上での FORTRAN の需要は高いと思われる．また，FORTRAN 77 の上位互換である Fortran 90 が ISO の国際規格となり，JIS 規格にも制定された．Fortran 90 は，配列演算などの拡張により，ベクトル計算機や並列計算機でのプログラムを記述し易い言語となっている．さらに，Fortran 90 に対して，FORALL 文，データ配置に関するコンパイラ・ディレクティブなどを拡張した HPF (High Performance Fortran) という仕様が提案されている．本稿では，HPF を SIMD 型超並列計算機 SM-1 に実装する際の問題点やその解決法，性能評価について述べる．

和文キーワード 超並列, SIMD, FORTRAN, HPF, コンパイラ

An Implementation and a Performance Evaluation of HPF for a SIMD Massively Parallel Machine

Susumu Komae * Hideo Sugimori * Koji Ohtani †
Shigeru Uzuhara ‡ Michiaki Yasumura ‡
* Sumitomo Metal Industries, Ltd. † Axe, Inc. ‡ Keio University.
Keio University Yasumura Lab. 5322 Endo Fujisawa shi.

Abstract

FORTRAN is a popular programming language in the field of science and technology and attractive to those users of massively parallel machines. Fortran 90 is the new international standard of Fortran language and new features such as array operations are added to FORTRAN 77. HPF(High Performance Fortran) is an extension to Fortran 90 providing support for high performance programming on a variety of parallel machines. HPF users can orchestrate parallel programs by using parallel constructs such as FORALL and directives for data decompostion. In this paper, we describe an implementation of HPF on SIMD architectures and a preliminary performance evaluation conducted on a massively parallel machine, SM-1.

英文 key words Massively parallel, SIMD, FORTRAN, HPF, compiler

1 はじめに

科学技術計算などの分野では，計算性能の向上が望まれている．これに応えるため，近年，超並列計算機が注目されつつある．科学技術計算分野のプログラミングには，依然として FORTRAN 言語が使われているため，超並列計算機上でも，FORTRAN が使えることが望ましい．また，FORTRAN 77 の上位互換である Fortran 90[2] が ISO の国際規格となっており，JIS 規格にもなっている．この Fortran 90 は，超並列計算機への実装を意識して，配列演算などを拡張したもので，かなり高機能な言語となっている．さらに，この Fortran 90 に対して拡張を施したものとして，HPF (High Performance Fortran)[1] がある．HPF は 40 以上の大学や企業が参加した High Performance Fortran Forum によって提案された仕様であり，主に，FORALL 文，データ配置などに関するコンパイラ・ディレクティブが拡張されている．しかし，HPF は高機能な Fortran 90 に対する拡張であるため，仕様が大き過ぎ，実現が困難である．そこで，HPF には Subset HPF という仕様も用意されており，Subset HPF は Fortran 90 のサブセットに HPF の拡張機能のサブセットを加えたものとなっている．

我々は，超並列計算機 SM-1[6] 上に Fortran 90 のサブセットである Bee-Fortran Ver1[8] を開発した．SM-1 は，住友金属工業 (株) と豊橋技術科学大学 湯浅研究室が中心となって開発した SIMD 型超並列計算機である．我々は，Bee-Fortran Ver1 に Subset HPF の機能を加えた Bee-Fortran Ver2 を開発した．本稿では，Subset HPF を実装する際の問題点とその解決について述べ，さらに，SM-1 上での性能評価について述べる．

以下では，まず Bee-Fortran 処理系の概要を説明する．次に言語仕様として，Fortran 90, HPF とそのサブセット，Bee-Fortran について説明する．そして，実装の際の問題点とその解決方法について述べ，最後に SM-1 上での性能評価について述べる．

2 処理系の概要

2.1 SM-1

SM-1[6] は SIMD 型超並列計算機である．そのシステム構成を図 1 に示す．SM-1 は，1024 個のプロセッサ・エレメント (Processor Element 以降は PE と略す) からなる PE アレイ部と，フロント・エンド・プロセッサ (Front End Processor 以降は FE と略す) 部の二つの部分から構成される．

PE は 32×32 のメッシュ状に配置され，PE 間には通信路 (NEWS 通信路) が用意されている．メッシュの端にある PE どうしの結合を変えることによって，2D メッシュ，2D トーラス，リング・ネットワークを構成することができる．また，縦方向，横方向，もしくは全体で OR 演算を行なう 1 ビットバスが用意されている．さらに，シャッフルエクスチェンジ型の通信路 (Shuffle-Exchange 通信路) も用意されており，PE 間のデータのリダクションや，トランスポーズに利用できる．PE は 8 ビット ALU をもち，命令セットはマイクロコードで実現され，32 ビット整数演算命令や IEEE の浮動小数点数演算命令を持つ．各 PE はアクティビティと呼ばれるフラグをもち，フラグの立っている (アクティブな) PE のみが演算する．

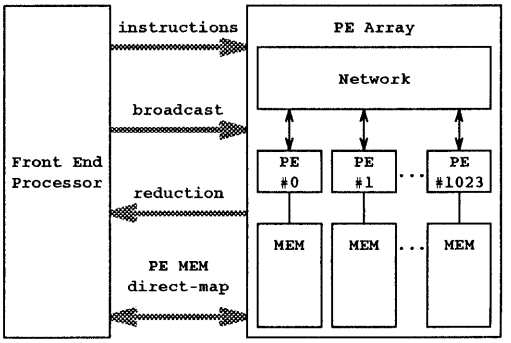

図 1: SM-1 のシステム構成

FE には Sun 社の Sparc Station を用いており，PE アレイ部は SPARC CPU のコプロセッサとして動作する．

2.2 並 C 言語

並 C 言語 [7] は，豊橋技術科学大学 湯浅研究室が中心となって開発された並列 C 言語である．並 C 言語は，SM-1 上のプログラミングの基本言語となっており，SM-1 のハードウェアに近いレベルの制御を記述することができる．PE 上のデータはパラレルクラスのデータと呼ばれる．FE 上のデータは通常の C 言語のデータと同様に扱われ，パラレルクラスと対比して，スカラクラスのデータと呼ばれる．並 C 言語は，通常の C 言語の機能に加えて，パラレルクラスのデータ同士の演算，パラレルクラスのデータのリダクション演算，PE 間の通信，PE のアクティビティを変化させるデータパラレル制御文などを持つ．並列実行の文脈におけるスカラクラスのデータは，パラレルクラスへ暗黙にクラス変換 (FE から PE へのブロードキャスト) される．

2.3 Bee-Fortran

Bee-Fortran は SIMD 型超並列計算機 SM-1 上の Fortran 90 のサブセットとして，開発された．本処理系は，Bee-Fortran コンパイラによって，Bee-Fortran のプログラムから並 C のプログラムを生成し，並 C コンパイラによって実行形式を生成する (図 2)．Bee-Fortran コンパイラは AT&T ベル研究所で開発された f2c[4] をベースにしている．

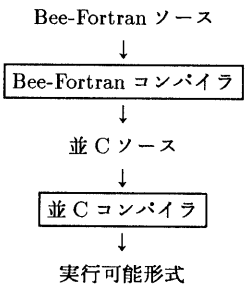

図 2: Bee-Fortran 処理系

3 言語仕様

3.1 Fortran 90

Fortran 90[2] の規格は，アメリカ規格協会 X3J3 委員会によって開発され，1991 年 2 月に ISO の国際規格に，1994 年 1 月には JIS の規格に制定された [5].

Fortran 90 は FORTRAN 77 に対して上位互換であり，主に以下の機能が拡張されている．

- 配列表記 (WHOLE ARRAY, ARRAY SECTION, WHERE)
- 動的メモリアロケーション
- 構造データ型と演算子のオーバーローディング
- 属性と宣言
- モジュール
- 制御文 (DO ～ END DO, DO WHILE, SELECT CASE, EXIT, ...)
- 組み込み関数

3.2 HPF

HPF (High Performance Fortran)[1] は，1992 年 3 月から 1993 年 3 月の間に 40 以上の組織が参加した High Performance Fortran Forum によって検討され，1993 年 5 月に Ver1.0 の仕様がまとめられた．Fortran 90 に対して，並列計算機への実装を意識した拡張が施されている．主な拡張部分は以下の通りである．

- データ配置などのディレクティブ
- FORALL 文 FORALL 構文
- 組み込み関数，標準ライブラリの拡張
- EXTRINSIC 手続き

3.3 Subset HPF

フルセットの HPF は Fortran 90 の仕様を含むことから，そのすべての仕様を実現するのは困難である．そのため HPF では，早期の実現が容易なように，HPF のサブセット (Subset HPF)[1] が定義されている．Fortran 90 の機能のうち，Subset HPF に含まれないものは，主に以下のものである．

- 構造データ型と演算子のオーバーローディング
- 属性の一部 (TYPE, TARGET, POINTER)
- モジュール
- 制御文の一部 (SELECT CASE)

また，HPF の拡張機能で，Subset HPF に含まれないものは，主に以下のものである．

- ディレクティブの一部 (REALIGN, REDISTRIBUTE, DYNAMIC, PURE, EXTRINSIC)
- FORALL 構文 (FORALL 文は Subset HPF に含まれる)
- HPF のライブラリと HPF_LIB モジュール

3.4 Bee-Fortran

Bee-Fortran Ver1 は，Fortran 8x などを参考に言語仕様を設計し，独自のディレクティブを持っている．Bee-Fortran Ver1 は FORTRAN 77 に対して，以下の機能が拡張されている．

- 配列表記 (WHOLE ARRAY, ARRAY SECTION, WHERE)
- 属性と宣言
- 制御文 (DO ～ END DO, DO WHILE, ...)
- Fortran 90 の組み込み関数
- 独自のデータ配置に関するディレクティブ (分割方式はサイクリック分割のみ)
- FORALL 文

Bee-Fortran Ver2 で追加するのは，主に以下の機能である．

- データ配置に関するディレクティブ (PROCESSORS, TEMPLATE, DISTRIBUTE, ALIGN)
- 組み込み関数の一部 (スカラ・ビット演算など)
- ALLOCATABLE 属性, ALLOCATE, DEALLOCATE

ただし，以下の部分に若干の制限がある．

- 文字型配列
- EQUIVALENCE 文
- COMMON 文
- 配列構成子 (array constructor)
- PROCESSORS ディレクティブ

図3 に FORTRAN 77, Fortran 90, HPF, Subset HPF, Bee-Fortran の関係を示す．

4 実装

Bee-Fortran Ver2 に追加されるデータ配置に関するディレクティブ PROCESSORS, TEMPLATE, DISTRIBUTE, ALIGN のために考慮しなければならないのは以下の点である．

- サイクリック・ブロック対応
- アラインメント処理
- 同一配置の判定

これらについて，それぞれ説明する．

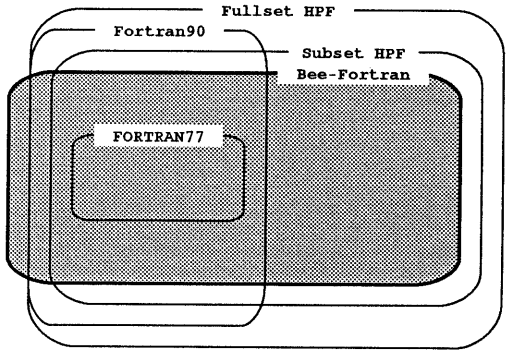

図 3: 各言語仕様の関係

4.1 サイクリック・ブロック対応

配置 Ver1 では，サイクリック分割のみのサポートとなっていたが，Ver2 では，サイクリック・ブロック分割 (図 4) をサポートする．サイクリック分割やブロック分割は，サイクリック・ブロック分割の特殊な場合として扱われる．

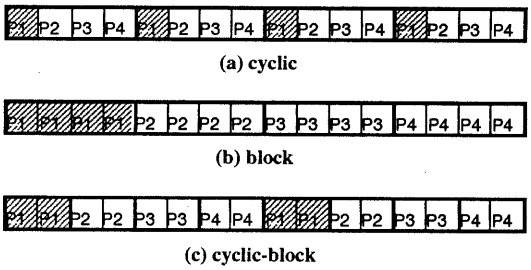

図 4: サイクリック・ブロック分割

例えば，以下のようにブロックサイズ 4 でサイクリック・ブロック分割に指定された配列 ma は各 PE 上で図 5 のように配置される．

```
      integer, parameter :: n=8205
      integer, dimension(n) :: ma
!hpf$ processors p1(1024)
!hpf$ distribute ma(cyclic(4)) onto p1
      ma = (/ (i,i=1,n) /)
      .....
```

コード生成 配列が関数の引数や返り値などにも利用されることがあるため，実行時に，各配列についてデータの先頭アドレスに関する情報の他に，データの配置状態

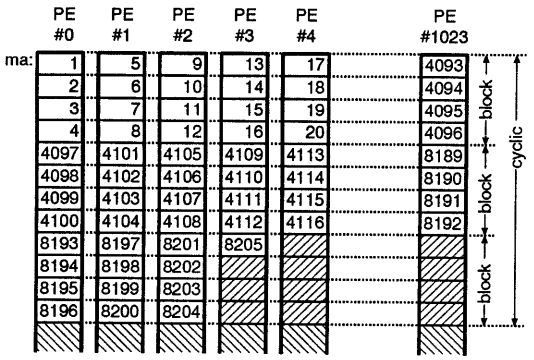

図 5: PE 上のメモリ配置

についての情報も必要である．サイクリック・ブロック対応にともなって，ブロックサイズの情報を付け加えるようにした．

配列演算式のコードでは，サイクリック分割の場合，サイクリックによる折り畳み回数の演算の繰り返すループを生成する必要がある．今回，サイクリック・ブロック分割に対応するため，このループを，折り畳み回数とブロックサイズ回数の二重ループとした．

Bee-Fortran では，データの配置状態について情報を，配列の形状 (shape) を表す構造体として扱っている．

通信ライブラリ　配列演算式で並列参照されている配列の各演算要素が同一配置でない場合，同一配置となるようにコピーしなければならない．Bee-Fortran では式の左辺の配置に合わせる方法 (Owner Computes Rule) を採用している．配列のコピーには専用の通信ライブラリを用意しており，サイクリック・ブロック分割に対応させた．

隣接 PE への通信の場合，ブロック分割であれば，サイクリック分割と比べて通信量が少なくなり，ブロックサイズ分をまとめて通信させることも容易であるため，通信時間が短く済む．

4.2 アラインメント処理

ALIGN ディレクティブによって，データ間の配置のアラインメントの指定ができる．Bee-Fortran では，基本的には，必要な大きさの領域を確保し，中間言語レベルで各要素への参照の添字式を変換することによって対応する．

同一 PE 上に配置する場合　alignee の各次元に対して，align target に対応する次元がない場合，または * が指定された場合，その次元を同一 PE 上に配置する．

```
      integer, dimension(100) :: ma
      integer, dimension(100,100) :: mb
!hpf$ processors p1(1024)
!hpf$ distribute ma(cyclic) onto p1
!hpf$ align mb(*,:) with ma(:)
      .....
```

上の例では，配列 mb の一次元目の要素が同一 PE 上に配置される (図 6).

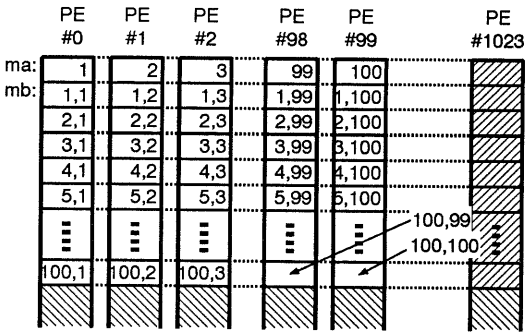

図 6: 同一 PE 上に配置する場合

PE 上に複製を持つ場合　align target の各次元に対して，alignee に対応する次元がない場合[1]，または * が指定された場合，その次元の値は各 PE に複製を持つ．

複製部分の値を参照する場合は，代入先などに都合の良い PE を選択し，その PE の持つ値を参照する．

複製部分へ代入を行なう場合は，右辺の値の配置に都合の良い PE を選択し，その PE へ代入した後，各 PE へブロードキャストする．

```
      integer, dimension(50,50) :: ma
      integer, dimension(50) :: mb
!hpf$ processors p2(32,32)
!hpf$ distribute ma(cyclic,cyclic) onto p2
!hpf$ align mb(:) with ma(*,:)
      .....
      mb = ma(1,:)
      .....
```

上の例では，配列 mb は (32,50) の領域に配置される (図 7). 例中の代入は，配列 ma(1,:) の配置された PE 上で行なわれ，その後，配列 mb の値の複製を持つ PE へブロードキャストが行なわれる．

[1]定数が指定された場合でも，SM-1 では * が指定されたものとして扱って問題ない

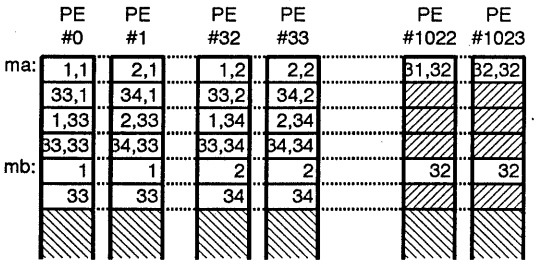

図 7: PE 上に複製を持つ場合

その他の場合 align target 側と同じ大きさの領域を確保し，中間言語レベルで各要素への参照の添字式を変換することによって対応する．

しかし，以下のような場合には，確保した領域が無駄になってしまう．

```
      integer, parameter :: P=1024
      integer, dimension(P*4) :: ma
      integer, dimension(P*2) :: mb, mc
!hpf$ processors p1(P)
!hpf$ distribute ma(cyclic(2)) onto p1
!hpf$ align mb(i) with ma(2*i)
!hpf$ align mc(i) with ma(i+N*2)
      .....
```

この場合，配列 mb，mc は，配列 ma と同じ大きさの領域を確保する必要はなく，配列 mb は cyclic(1)，配列 mc は cyclic(2) の配置のための領域を確保するだけで良い (図 8)．

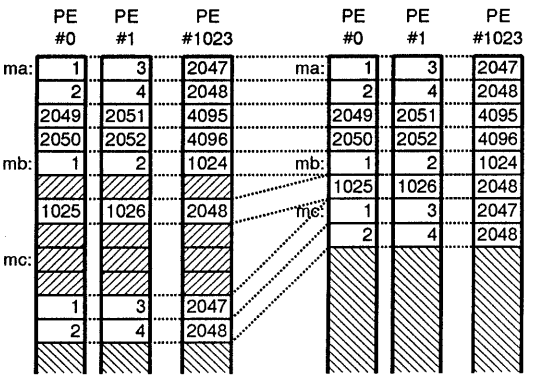

図 8: 領域の節約

このような場合を検出し，確保する領域を節約する．

4.3 同一配置の判定

配列演算式で並列参照されている配列の各演算要素が同一配置でない場合，通信ライブラリなどを用いて，同一配置となるようにコピーしなければならない．この同一配置の判定が，静的に判断できる場合はコンパイル時に行なうほうがよい．

配列のホーム (PE/FE)，配置の形状 (一次元配置 (リング状)/二次元配置 (メッシュ状)) などが明らかに違う場合は同一配置ではないとわかる．ベクトル添字である場合や，添字式に変数が含まれる場合は，同一配置となるかどうかは不明であるとする．そこで，問題となるのは，トリプレット添字の場合である．

それぞれの添字が $l_1 : u_1 : s_1$ と $l_2 : u_2 : s2$ であり，ブロックサイズが b_1，b_2 であるとすると，任意の非負整数 x について，

$$\lfloor \frac{s_1 x + l_1}{b_1} \rfloor = \lfloor \frac{s_2 x + l_2}{b_2} \rfloor$$

が成り立てば，同一配置である．

今回の実装では，これを用いて，次のように変形して判定している．その次元方向の PE 数を N とし，s_1 と b_1 の最大公約数を m_1，s_2 と b_2 の最大公約数を m_2 とすると，

$$\frac{b_1}{m_1} = \frac{b_2}{m_2} (mod\ N)$$

$$\frac{s_1}{m_1} = \frac{s_2}{m_2} (mod\ N)$$

$$\lfloor \frac{l_1}{m_1} \rfloor = \lfloor \frac{l_2}{m_2} \rfloor (mod\ N)$$

これらが成り立つ場合，同一配置であるとし，成り立たない場合は不明であるとする．

5 評価

Bee-Fortran Ver1 ではサイクリック分割のみであったため，アプリケーションによっては，最適なデータ配置でない場合があった．Bee-Fortran Ver2 で追加したコンパイラ・ディレクティブによって，ユーザによるデータ配置の制御が可能となった．

ライフゲーム (図 9) について，データの配置がサイクリック分割である場合とブロック分割である場合の実行時間を，超並列計算機 SM-1 上で実際に測定すると，表 1，図 10 のようになった．

SM-1 は PE 数が 32×32 個であるため，サイクリック分割もブロック分割も，n が 32 以下の場合は，ほぼ同じ結果となった．n が 32 より大きい場合は，ブロック分

```
      program lifegame
      integer, parameter :: n=256
      integer, dimension(n,n) :: a, t
!hpf$ processors p2(32,32)
!hpf$ distribute a(block,block) onto p2
!hpf$ align t(:,:) with a(:,:)
      integer c

      a = 0
      c = n/2
      a(c-3:c+3,c-3:c+3) = 1

      do i=1,100

      t(2:n-1,2:n-1) = a(2:n-1,2:n-1)
     &   + a(2:n-1,1:n-2) + a(2:n-1,3:n)
      t(2:n-1,2:n-1) = t(2:n-1,2:n-1)
     &   + t(1:n-2,2:n-1) + t(3:n,2:n-1)
     &   - a(2:n-1,2:n-1)
      where (t(2:n-1,2:n-1) == 3)
     &   a(2:n-1,2:n-1) = 1
      where (t(2:n-1,2:n-1) > 3)
     &   a(2:n-1,2:n-1) = 0
      where (t(2:n-1,2:n-1) < 2)
     &   a(2:n-1,2:n-1) = 0

      enddo

      end
```

図 9: ライフゲームプログラム (セル数 256 × 256, ブロック分割の場合)

表 1: ライフゲームの実行時間 [sec]

n	セル数	サイクリック	ブロック
20	400	0.41	0.41
32	1024	0.41	0.41
100	10000	4.31	1.96
256	65536	15.78	6.35
512	262144	60.76	23.81
1024	1048576	239.08	93.28

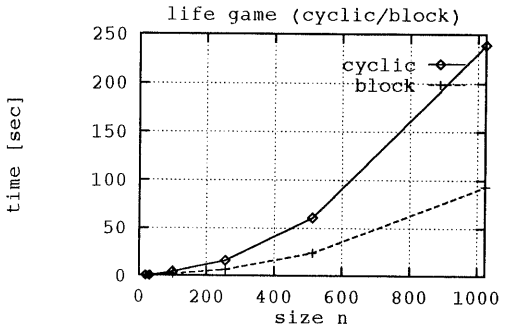

図 10: ライフゲームの実行時間

割の実行時間が短いという結果となった. このプログラムは隣接要素との通信が多いため, サイクリック分割よりもブロック分割の方が通信量が少なくなり, このような結果となった.

最後に, このプログラムをブロック分割にした場合, Bee-Fortran コンパイラによって生成される並 C のコードの一部 (DO ループ中の最初の配列演算式) を図 11 に示す. 配列演算式で並列参照されている配列が同一配置ではないので, 左辺の配列 t と同じ形状 (shape_1) の一時配列 i_9 を用意し, a(2:n-1,3:n) をコピーする. 同様に一時配列 i_10 を用意し, a(2:n-1,1:n-2) をコピーする. 次に, 演算をする要素を指定するマスク L_6 を作る. そして, ブロックサイズの回数のループ中で, pif 文によりマスク L_6 が真である PE 上で演算を行なう. この場合は, サイクリックによる折り畳みがないので, 二重ループとはなっていない.

6 おわりに

我々は, SIMD 型超並列計算機 SM-1 に Subset HPF を実装し, その実装に際して, 考慮しなければならない点について述べた. また, 適切なデータ配置を選択することによる効果を, SM-1 上で評価した.

現在, Bee-Fortran Ver2 は, 組み込み関数の一部を除いてほぼ完成している. 組み込み関数を, サイクリック・ブロック分割に対応することが今後の課題である. また, データ配置をコンパイラで自動的に行なうことも, 今後の課題である.

最後に, 並 C 言語, SM-1 などの開発にあたられた湯淺助教授をはじめ, 豊橋技術科学大学湯淺研究室の皆さん, 住友金属工業 (株) の皆さんに感謝します.

```
    … 省略 …

/* Main program */ BFMAIN__()
{

    … 省略 …

  for (i = 1; i <= 100; ++i) {

    BF_make_integer_PEarray(&i_9, i_9_body,
      &shape_1);
    SET_SEC_TRIPLET(sec_3[0], 1, 254, 1);
    SET_SEC_TRIPLET(sec_3[1], 1, 254, 1);
    SET_SEC_TRIPLET(sec_2[0], 2, 255, 1);
    SET_SEC_TRIPLET(sec_2[1], 3, 256, 1);
    BF_copy(&i_9, sec_3, &a, sec_2,
      sizeof(integer));

    BF_make_integer_PEarray(&i_10, i_10_body,
      &shape_1);
    SET_SEC_TRIPLET(sec_4[0], 2, 255, 1);
    SET_SEC_TRIPLET(sec_4[1], 1, 254, 1);
    BF_copy(&i_10, sec_3, &a, sec_4,
      sizeof(integer));

    BF_make_activity(L_6, real_t.shape, sec_3);

    for (c_idx = 0; c_idx < 64; c_idx++) {
      pif (L_6[c_idx]) {
        real_t.body.PEinteger[c_idx] =
          i_9.body.PEinteger[c_idx] +
          i_10.body.PEinteger[c_idx] +
          a.body.PEinteger[c_idx];
      }
    }

    … 省略 …

  }
} /* BFMAIN__ */
```

図 11: 生成された並 C コード (ライフゲーム)

参考文献

[1] High Performance Fortran Language Specification Version 1.0, High Performance Fortran Forum, *May 3, 1991*

[2] Fortran 90, ANSI X3J3, ISO/IEC 1539 *1991*

[3] JIS Programming Language FORTRAN, JIS X 3001-1982 (Reaffirmed 1987)

[4] S.I.Feldman et al., A Fortran-to-C Converter, AT&T Bell Laboratories Computing Science Technical Report NO.149, *August 2, 1990*

[5] M.Metcalf, J.Reid 著, 西村 他訳: bit 別冊 詳解 Fortran 90, 共立出版, *December, 1993*

[6] 松田, 湯淺: SIMD 型超並列計算機 SM-1(仮称) の概要とその性能, 情報処理学会研究報告, 92-ARC-87, *August, 1992*

[7] 貴島, 湯淺: SIMD 型超並列プログラミング言語「並 C」とそのコンパイラ, 情報処理学会研究報告, 92-PRG-9, *October, 1992*

[8] 大谷, 小前, 杉森, 渦原, 安村: 超並列計算機用 Fortran コンパイラの設計と試作, 電子情報通信学会研究報告, COMP92-94, *May 11, 1993*

[9] 渦原, 小前, 杉森, 大谷, 安村: Bee-Fortran のマルチターゲット・コンパイル方式, 情報処理学会研究報告, 93-PRG-13, *August 19, 1993*

[10] 杉森, 小前, 大谷, 渦原, 安村: 超並列計算機用言語 Bee-Fortran の通信ライブラリの最適化方式の検討, 第 47 回全国大会, 5D-1, *October 8, 1993*

[11] 小前, 杉森, 大谷, 渦原, 安村: 超並列計算機用言語 Bee-Fortran における配列配置方式の検討, 第 47 回全国大会, 5D-2, *October 8, 1993*