式の分割による並列化アルゴリズム ESH とその評価

岩根雅彦[†] 濱田智雄[†,☆] 宇野総一[†]
小島和広[†,☆☆] 松田孝史[†]

式の変数の定義される時間，変数の参照可能時間と演算優先順位に注目した変数レベルの並列化のためのアルゴリズム ESH を提案する．ESH では同一優先順位を持つ変数間の演算を1つにまとめて未決定ノードとし原始プログラムから決定ノードと未決定ノードの混在した不完全タスクグラフを生成する．このタスクグラフからノードを取り出したとき未決定ノードであれば変数の参照可能時間の早い変数間の2項演算をプロセッサに割り当てて未決定ノードを分解し，決定ノードを生成する．決定ノードであれば Duplication Scheduling Heuristic（DSH）によってスケジューリングする．すべてのノードが決定ノードになったときスケジューリングは完了する．ESH では複数の文間に内在する並列性を最適化できるだけでなく，最適な計算木が生成される．16 個のプログラムを用いて4種類のプロセッサモデルと細粒度マルチプロセッサ MSBM 上で ESH と，ツリーハイトリダクション（THR）と DSH の併用，LR 構文解析（LR）と DSH の併用との比較を行った．その結果，すべてのプログラムで速度向上比は ESH ≥ (THR+DSH) ≥ (LR+DSH) であった．ESH の LR+DSH に対する速度向上比の平均は 1.16 であった．一方 THR+DSH の LR+DSH に対する速度向上比の平均は 1.07 であった．

Evaluation of Expression Scheduling Heuristic (ESH) Algorithm

Masahiko Iwane,[†] Tomoo Hamada,[†,☆] Souichi Uno,[†]
Kazuhiro Oshima[†,☆☆] and Takashi Matsuda[†]

Expression Scheduling Heuristic (ESH) is a class of scheduling heuristics for the basic block of the program. ESH deals with unresolvable nodes and resolvable nodes. An unresolvable node consists of two variables and the dyadic operation between them. A resolvable node consists of more than three variables and the dyadic operations with the same priority of operation. ESH resolves the incomplete task graph including unresolvable nodes and resolvable nodes generated from a basic block into the complete task graph in process of scheduling. When ESH takes the node in order of node priority, ESH schedules it using Duplication-Scheduling Heuristic if it is an unresolvable node. Otherwise, ESH selects two earliest available variables and the dyadic operation between them, assigns them to the processor and generates an unresolvable node. ESH terminates when all nodes become unresolvable nodes. ESH not only optimizes the parallelism among the expressions but also generates optimal calculation tree. ESH is compared with Tree Height Reduction (THR) and DSH, and LR parser (LR) and DSH using 16 programs on 4 processor models and MSBM multiprocessors. The speed-up ratios of all programs are ESH ≥ (THR and DSH) ≥ (LR and DSH). The average one of ESH to LR and DSH is 1.16. Otherwise, The average one of THR and DSH to LR and DSH is 1.07.

1. はじめに

並列計算機をもちいてプログラムを並列化し処理速度の向上をはかる場合にはプログラムの意味的同一性を保持したうえで並列実行可能な部分を最大限に抽出し並列実行する必要がある．とくにプロセッサ数にほぼ比例した速度向上率が得られる doall ループやベクトル演算などをプログラムから抽出することは，プログラム全体の実行速度向上に大きく寄与する．しかし一般的なプログラムには逐次実行の代入文や分岐文も数多く存在する．このような部分も並列化することによっていっそうの速度向上が期待できる．

逐次実行の代入文で構成された基本ブロックを並列化するための従来の静的スケジューリング手順は，ま

† 九州工業大学工学部電気工学科
Department of Electrical Engineering, Faculty of Engineering, Kyushu Institute of Technology
☆ 現在，松下電器産業株式会社半導体開発本部
Presently with Matsushita Electric Industrial Co., Ltd.
☆☆ 現在，九州旅客鉄道株式会社
Presently with Kyushu Railway Company

ずプログラムをプロセッサ割当ての最小単位に分割し，依存解析を行ったのちにこの最小単位をノードとするタスクグラフを作成する．そしてスケジューリングアルゴリズムによってノードをプロセッサに割り当てる．このアルゴリズムとして General List Scheduling Heuristic (GSH)[1),2)], Duplication Scheduling Heuristic (DSH)[2)], Trace Scheduling (TRS)[3),4)] などがある．これらのアルゴリズムではプロセッサの割当て単位であるノードの大きさは本質的に自由である．VLIW 計算機やスーパスカラ計算機ではこのアルゴリズムを機械命令レベルに適用[1),4)]し，OSCAR などでは文レベルに適用[5),6)]している．また，計算木の生成方法として一般的に用いられている LR 構文解析 (LR)[7)]による生成と，代入文を 2 項演算に分解して変数レベルで並列化するツリーハイトリダクション (THR)[8)]による生成がある．THR は式の計算木を完全 2 分木に近づけ木の高さをできる限り低くして並列実行可能性を最大にしている．この計算木の作成は代入文ごとに独立に行われる．しかし基本ブロックには通常複数の代入文が存在するため，これらの並列実行性を考えると THR による変数レベルの並列化は最適とはいえない．

そこで，代入文の式の変数の定義される時間と演算優先順位に注目した Expression Scheduling Heuristic (ESH) を提案する．ESH では同一優先順位を持つ変数間の演算すなわち部分式を 1 つのノードとして式を分割する．そして依存解析を行ってタスクグラフを作成する．タスクグラフからノードを取り出して，変数の取得時間の早い順に部分式を 2 項演算に分解してプロセッサに割り当てる．

本論文では，ESH に必要な基本概念について述べ，次に ESH スケジューリングアルゴリズムの概要について述べる．そして ESH により並列化した評価プログラムを 4 種類のプロセッサモデルでシミュレーションを行い，さらに MSBM マルチプロセッサ[9)]で実行する．その結果を考察し ESH の評価を行う．

2. 基本概念

2.1 前　　提

一般に手続き型言語で記述されたプログラムは代入文，制御文などで構成される．このような文系列で，その出口を除いて分岐文がなく，かつその入口を除いて外から分岐されることのない部分を基本ブロックとよぶ．基本ブロックには手続き呼び出しは含まれず代入文のみで構成される．基本ブロック内代入文系列の変数レベルでの 2 項演算を基本とした並列化を考える．並列化によって生成される変数を一時変数，一方代入文中に明示的に与えられた変数を恒久変数，一時変数および恒久変数をさすときは単に変数とよぶ．べき乗は並列化の観点から乗算に置き換え，代入文の括弧は演算精度に関係したもののみを許す．

2.2 タスクグラフ

基本ブロックを並列化するにあたって基本ブロック内の変数の定義と参照を表す有向非循環グラフであるタスクグラフを導入する．タスクグラフのノードは単項演算子をともなった変数の代入文と 2 変数演算の代入文，および演算優先順位が等しい演算子からなる 3 変数以上の多変数演算の代入文である．前者を決定ノードとよび，後者を未決定ノードとよぶ．演算優先順位はアルゴリズムが適用される言語において定義されるが，ここでは算術演算においては {×,÷}, {+,−} の順で，論理演算では ∧, ∨ の順で優先順位を定義する．ただし {} 内の演算子の優先順位は等しい．エッジはノード間の依存関係を表す．依存関係には真依存，逆依存，出力依存があるが，逆依存，出力依存関係は変数のリネーミング等により回避できることから真依存関係のみを表す．

とくに決定ノードのみで構成されるグラフを完全タスクグラフ，未決定ノードを含むグラフを不完全タスクグラフとよぶ．このようなタスクグラフにおいて，ノード N_i からノード N_j に向かうエッジがあるときノード N_i をノード N_j の親ノード，ノード N_j をノード N_i の子ノードとよぶ．タスクグラフ中で親ノードを持たないノードを入口ノード，子ノードを持たないノードを出口ノードとよぶ．ノード N_i の代入文を実行する時間をノード実行時間 $T_e(N_i)$ で表し，未決定ノードのノード実行時間はその演算を逐次実行したときの実行時間をとる．ノード N_i から出口ノードの間の経路で，経路に存在するノードの実行時間の和の最大値をノード N_i の優先度とよぶ．**図 1** (a) は基本ブロック，図 1 (b) は基本ブロックに対応する依存解析後の不完全タスクグラフ，図 1 (c) は不完全タスクグラフを ESH によってスケジューリングした結果の完全タスクグラフを示す．図 1 (b) のノード N1, N3 は未決定ノード，図 1 (b) のノード N2 と図 1 (c) のすべてのノードは決定ノードである．また変数 X_i は恒久変数，V_i は一時変数である．

2.3 2 項演算とタプル表現

演算優先順位が等しい演算子で構成される未決定ノードの代入文では右辺の式の評価順序に拘束はない．すなわち，結合則，可換則をどのように適用してもかまわない．基本ブロックの実行開始時間を 0 とし

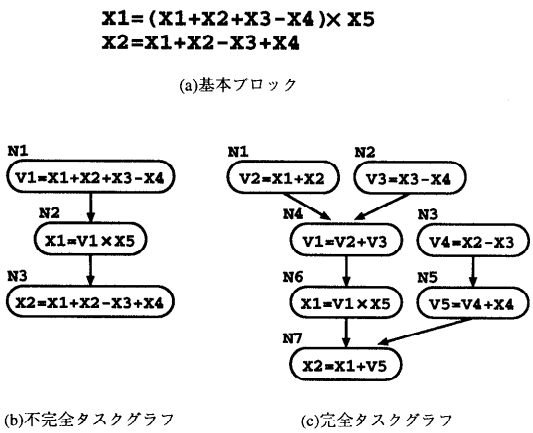


図1 タスクグラフ
Fig. 1 Examples of a task graph.

表1 演算子の変換則
Table 1 Transformation rules of operators.

	OP_i	OP_j	OP_q	OP_r
空を含む演算	ϕ	α	ϕ	α
	α	ϕ	ϕ	α
算術演算	$+$	$+$	$+$	$+$
	$+$	$-$	$+$	$-$
	$-$	$+$	$-$	$-$
	$-$	$-$	$-$	$+$
	$\times$	$\times$	$\times$	$\times$
	$\times$	$\div$	$\times$	$\div$
	$\div$	$\times$	$\div$	$\div$
	$\div$	$\div$	$\div$	$\times$
論理演算	$\wedge$	$\wedge$	$\wedge$	$\wedge$
	$\vee$	$\vee$	$\vee$	$\vee$

$\alpha = \{+, -, \times, \div, \wedge, \vee\}$

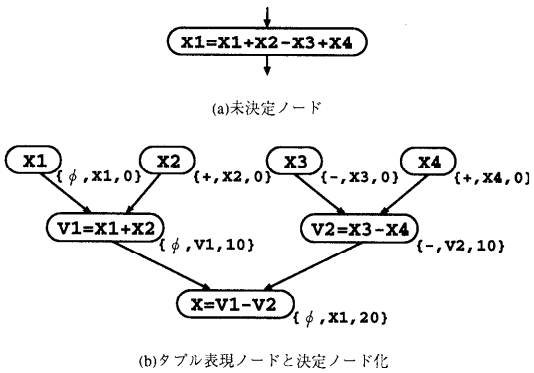


図2 未決定ノードの分解
Fig. 2 Partitioning resolvable node.

たときに，式に現れる変数 X_i がプロセッサ PU_j で定義される時間を変数 X_i の定義時間 $T_d(X_i, PU_j)$ とよぶ．変数 X_i がプロセッサ PU_j で定義されてプロセッサ PU_k で参照可能となる時間を変数 X_i の取得時間 $T_g(X_i, PU_k)$ とよび次式で与える．

$$T_g(X_i, PU_k) = T_d(X_i, PU_j) \quad (j = k) \quad (1)$$

$$T_g(X_i, PU_k) = T_d(X_i, PU_j) + T_c(PU_j, PU_k) \quad (j \neq k) \quad (2)$$

ここで，$T_c(PU_j, PU_k)$ はプロセッサ PU_j とプロセッサ PU_k との通信時間である．すると取得時間の小さい変数に対する演算から順次評価していけば代入文の実行が最も早く完了する可能性が大きい．また基本ブロックの代入文を図1で示したように決定ノードと未決定ノードに分解していくので負符号または否定演算子をともなった変数は決定ノードとして扱われる．それゆえ未決定ノードには負符号，否定演算子をともなった変数は含まない．

式(3)に示した未決定ノードの代入文において変数 X_0 の前に形式的な空なる演算子 $OP_0(\equiv \phi)$ をおけば演算子 OP_i と変数 X_i を組として取り扱える．

$$X_n := X_0 OP_1 X_1 \cdots OP_i X_i \cdots \cdots OP_j X_j \cdots OP_m X_m \quad (3)$$

取得時間の最小と次に小さい2つの変数 X_i，X_j の演算を考えると変数 X_i，X_j に対する式(3)の部分式は式(4)となる．

$$OP_i\ X_i\ OP_j\ X_j = OP_q\ V_q \quad (4)$$

$$V_q := (X_i\ OPr\ X_j) \quad (5)$$

このとき演算子 OP_i，OP_j，OP_q，OP_r には**表1**に示した関係がある．

式(3)は変数が1つ減る式(6)となって，式(5)で表された決定ノード N_q が生成され，このノードの演算実行時間は OP_r によって規定される．

$$X_n := X_0 OP_1 X_1 \cdots OP_q V_q \cdots OP_m X_m \quad (6)$$

未決定ノードの入力変数に，演算子 OP_i，変数 X_i，変数 X_i の取得時間 $T_g(X_i, PU_k)$ をタプル $(OP_i, X_i, T_g(X_i, PU_k))$ として与える．2つのタプルを用いて未決定ノードから決定ノードを生成する．生成された決定ノードの一時変数にも同様のタプル $(OP_q, V_q, T_g(V_q, PU_k))$ を与える．なお，一時変数 V_q の定義時間 $T_d(V_q, PU_k)$ は次式で与えられる．

$$T_d(V_q, PU_k) = \max\{T_g(X_i, PU_k), T_g(X_j, PU_k), T_s(PU_k)\} + T_e(N_q) \quad (7)$$

ここで，$T_s(PU_k)$ はプロセッサ PU_k での実行開始可能時間である．これらのタプルと式(3)～(7)によって統一的に完全タスクグラフが生成できる．**図2**(a)は未決定ノード，図2(b)は生成された決定ノードからなる完全タスクグラフを示す．図2において入力変数の取得時間は0，加減演算時間を10とした．

```
algorithm ESH(BB,GC)
  input:  BB /* 基本ブロック */
          GC /* スケジューリング前の
                ガントチャート */
  output: GC /* スケジューリング後の
                ガントチャート */
  (1) 基本ブロックからタスクグラフ TG を作成.
  (2)TG の各ノードのノード優先度を計算.
  (3)while(TG ≠空) begin
  (4)  TG で親ノードのすべてがスケジューリング済の
       ノードで最もノード優先度が高いものを
       選び CN とする.
  (5)  if(CN は未決定ノード？) then
  (6)   RESNODE(TG,GC,CN)
          /*未決定ノードを決定ノードに分解*/
       else
  (7)   DSH(TG,GC,CN)
          /*決定ノードを DSH でスケジューリング*/
       endif
  (8) タスクグラフを更新する.
  (9)end while
end algorithm
```

図3 ESH アルゴリズム
Fig. 3 ESH algorithm.

3. ESH アルゴリズム

3.1 概　　要

ESH では基本ブロックを構成する代入文を式の評価順序に従って決定ノードと未決定ノードが混在する不完全タスクグラフを作成する．次に不完全タスクグラフのすべてのノードにノード優先度をつける．親ノードがすべてプロセッサ割当てを終了したノードのなかでノード優先度の最も高いノードから順に，そのノードが決定ノードであれば DSH によりプロセッサに割り当てる．未決定ノードであれば決定ノードを1つ生成し，プロセッサ割当てを行う．未決定ノードから1つの決定ノードを分解したもう一方のノードに対してノード優先度を再計算する．すべてのノードが決定ノードになるまでこの操作を繰り返す．プロセッサ割当てはガントチャートで表現する．**図3** に ESH のスケジューリング概要を示す．図3 の DSH(TG,GC,CN) はタスクグラフ TG，ガントチャート GC，スケジュールの対象ノード CN を入力として，DSH によって対象ノード CN をプロセッサに割り当てる．その結果をガントチャート GC に書き込む．RESNODE(TG,GC,CN) は未決定ノードのスケジューリングを行う．

GSH や DSH では与えられたタスクグラフのノードをそのままの形でスケジューリングするが，ESH ではノードを生成してタスクグラフを再構成しながらプロセッサに割り当て，完全タスクグラフが完成したときにはスケジューリングが完了する．

3.2 未決定ノードの並列化

式 (3) に示した未決定ノードの代入文を式 (4) によって式 (5)，(6) で示した2つの代入文に分解する．まず式 (3) のすべての入力変数の取得時間をプロセッサごとに算出する．プロセッサ PU_k における入力変数の取得時間の最も小さい2変数 X_i，X_j の取得時間を $T_g(X_i, PU_k)$，$T_g(X_j, PU_k)$ とする．表1 の変換則によって演算子 OP_r の演算時間が演算子 OP_j の演算時間より長くなるなら，取得時間の最も早い変数と3番目に早い変数をとる．以下同様である．そのプロセッサで式 (5) の代入文を実行したときの一時変数 V_q のそのプロセッサでの取得時間 $T_g(V_q, PU_k)$ を DSH でスケジューリングすることによって算出する．すべてのプロセッサについてそのプロセッサでの取得時間 $T_g(V_q, PU_k)$ を求め，そのなかで最小の取得時間 $T_g(V_q, PU_s)$ が得られるプロセッサ PU_s に式 (5) で生成された決定ノードを割り当てる．そして未決定ノードの代入文を式 (6) に変更し，変更された未決定ノードのノード優先度を計算し直す．**図4** に未決定ノードから決定ノードの生成アルゴリズムを示す．

```
algorithm RESNODE(TG,GC,UN)
  input:  TG   /* タスクグラフ */
          GC   /* ガントチャート */
          UN   /* 未決定ノード */
  output: GC   /* 未決定ノード UN から決定ノード
                  が生成されてスケジューリング
                  された結果のガントチャート*/
  (1)  for j:=1 to no_of_PU do begin
         / *PU の数まで繰り返す */
  (2)    未決定ノード UN の入力変数 X0,X1,…, Xm の
         プロセッサ PUj における取得時間を計算して
         タプルを作成.
         TSj={(OP0,X0,Tg(X0,PUj)),(OP1,X1,Tg(
              X1,PUj)),…, (OPm,Xm,Tg(Xm,PUj))}
  (3)  end for
  (4)  for j:=1 to no_of_PU do begin
  (5)    TSj のなかで取得時間の最小のタプルと次
         のタプルを TP1,TP2 とする.
         この時演算子の変換則により演算時間が
         増えるようならば次に小さいものをとる
         (以下同様)
  (6)    TP1,TP2 から決定ノード NEW_NODEk
         (一時変数 Vk を定義する代入文) と
         そのタプル TPk を候補として生成し TG の
         更新.
  (7)   DSH(TG,GC,NEW_NODEk)
  (8)   Vk の取得時間を Tg(Vk,PUj) とする.
  (9)   GC と TG を (6) の実行前に戻す.
  (10) end for
  (11) min{Tg(Vk,PU1),…, Tg(Vk,PUno_of_PU)}
       を Tg(Vk,PUs) とする.
  (12) この Tg(Vk,PUs) に対応する決定ノード
       NEW_NODEk と未決定ノードから決定ノード
       NEW_NODEk を除去したノード P_NODE を TG に
       追加，未決定ノード UN を TG から削除.
  (13) P_NODE の優先度を再計算し TG, GC を更新.
end algorithm
```

図4 未決定ノードの分解アルゴリズム
Fig. 4 Algorithm of partitioning resolvable node.

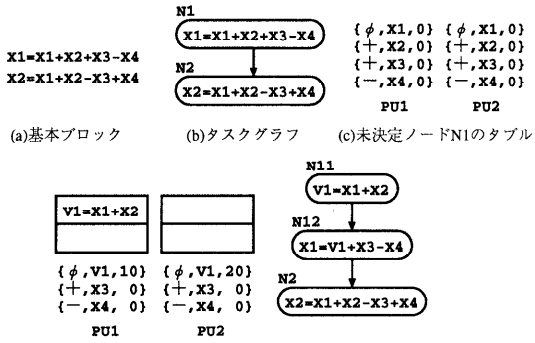


図5 スケジューリング例（N1のスケジューリング過程）
Fig. 5 The example of scheduling (scheduling process of N1).

3.3 簡単なスケジューリング例

図5(a)に示した基本ブロックをESHを用いてスケジューリングする．2台のプロセッサを使用し，2変数の加減算は10単位時間，プロセッサ間の通信に10単位時間が必要とする．基本ブロックの実行開始前には恒久変数はすでに定義されているとする．まず基本ブロックから図5(b)に示したタスクグラフを作成する．タスクグラフは2つの未決定ノードN1，N2で構成される．ノード優先度を求めた結果，ノードN1をはじめにスケジュールする．図5(c)にプロセッサごとに作成されたノードN1の入力変数 X_i のタプルを示す．図5(c)のタプルからV1=X1+X2はPU1に割り当てる．決定ノードN11と未決定ノードN12に分解し，未決定ノードN12のノード優先度を再計算する．次に未決定ノードN12のスケジューリングを行う．図5(d)のタプルからV2=X3−X4をPU2に割り当てる．未決定ノードN12を決定ノードN13とN14に分解し，ノードN14のノード優先度を再計算する．次に図5(e)をもちいてノードN14をスケジューリングする．このときDSHによりV2=X3−X4がPU1に割当ての変更が行われる．ノードN1のスケジュール後のガントチャート，タスクグラフを**図6**に示す．なお，ガントチャートの1コマは10単位時間である．次にノードN2をスケジュールする．まずノードN2の入力変数 X_i のタプルをプロセッサごとに作成する．PU2におけるX1の取得時間は通信時間が

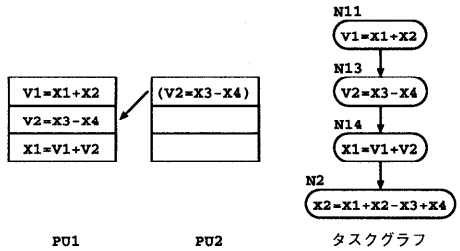


図6 スケジューリング例（N1のスケジューリング後）
Fig. 6 The example of scheduling (after scheduling N1).

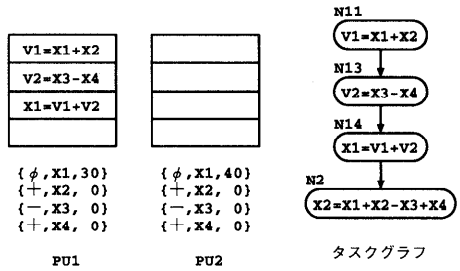


図7 スケジューリング例（N2のスケジューリング前）
Fig. 7 The example of scheduling (before scheduling N2).

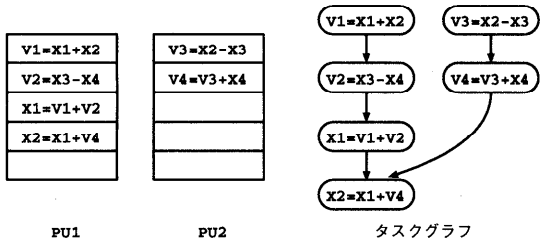


図8 スケジューリング例（N2スケジューリング後）
Fig. 8 The example of scheduling (after scheduling N2).

必要であるので40単位時間となる．**図7**にN2のスケジュール前のガントチャート，タプル，タスクグラフを示す．図7のタプルからノードN1と同様にスケジューリングすれば**図8**に示すガントチャート，タスクグラフが得られる．

4. 実　　験

4.1 シミュレーション

4.1.1 プロセッサモデルと評価プログラム

表2に示した演算時間と通信時間を持った4種類のプロセッサモデル上で，**表3**で示した特性の評価プログラムをもちいて，LRとDSHによる並列化，THRとDSHによる並列化，ESHによる並列化における速度向上比の比較を行った．評価プログラムのなかで，Exam1–4はESHの評価のために作成したプログラ

```
void Exam1{
  int x1=1,x2=2,
      x3=3,x4=4;

  x1=x1+x2+x3-x4;
  x2=x1+x2-x3+x4;
}
```

(a)Exam1

```
void Exam2{
  int x1=1,x2=2,x3=3,
      x4=4,x5=5,x6,x7;

  x1=x1+1;
  x2=x2/4;
  x3=10*x3;
  x4=x4+20;
  x6=x1*x2*x3/x5;
  x7=x1+x2+x4;
}
```

(b)Exam2

```
void Exam3{
  float x1,x2,x3,
        x4,x5,x6;

  x1=2.0*3.0;
  x2=x1*3.0;
  x3=x2*4.0;
  x4=x3*5.0;
  x5=x4/x1/x2*x3;
  x6=x5/x1*x2*x3;
}
```

(c)Exam3

図 9 評価のために作成したプログラム
Fig. 9 Example programs made for evaluation.

表 2 プロセッサモデルの性能
Table 2 Specification of processor models.

モデル番号	加算		減算		乗算		除算		S
	I	R	I	R	I	R	I	R	
Model1	1	1	1	1	1	1	1	1	0
Model2	1	1	1	1	1	1	1	1	1
Model3	1	2	1	2	2	4	2	4	0
Model4	1	2	1	2	2	4	2	4	1

I：整数，R：実数，S：同期・通信（単位は単位時間である）

表 3 評価プログラムの特性
Table 3 Characteristics of programs for evaluation.

Exam	プログラム名	依存	量
1	評価のために作成 Fig9 (a)	有	4
2	評価のために作成 Fig9 (b)	有	4
3	評価のために作成 Fig9 (c)	有	4
4	評価プログラム Ex4[10]	有	6
5	Whetstone Module1	有	4
6	Whetstone Module2	有	4
7	Whetstone Module6	有	4
8	Flops Loop1	有	4
9	Flops Loop3	有	3
10	Flops Loop4	有	8
11	Flops Loop6	有	8
12	Flops Loop7	有	8
13	Livermore Kernel1	無	3
14	Livermore Kernel5	無	0
15	Livermore Kernel7	無	9
16	Livermore Kernel9	無	10

依存：文間依存，量：未決定ノード量

ムで，その他は Whetstone，Flops，Livermore ベンチマークプログラムからランダムに選んだプログラムである．並列化の効果をあげるために，ベンチマークプログラムのなかの括弧をプログラムの精度が変わらない範囲で除去した．また，Exam1–3 をもちいて実験結果の詳しい考察を行うので，そのプログラムリストを**図 9** に示す．なお，表 3 の文間依存の有無は未決定ノードを持った文がそれ以前の文に真依存するかどうかを示し，未決定ノード量は未決定ノードに含まれる変数の数の最大値を示す．

4.1.2 シミュレーション結果

表 4 に最大 2 台，3 台，4 台，8 台のプロセッサをもちいた場合の速度向上比を示した．表 4 の速度向上比は当該モデルプロセッサ 1 台による逐次実行時間に対する比率である．なお，表 4 の最大並列度は当該アルゴリズムで並列化できる最大プロセッサ台数で，プロセッサ台数が最大並列度よりも大きい個所は最大並列度による速度向上比を与えた．また，表 4 からプロセッサ台数にわたって速度向上比の平均をとった結果を**表 5** に示す．

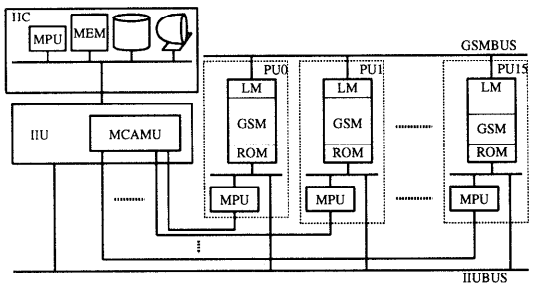


図 10 MSBM のシステム構成
Fig. 10 Hardware organization of MSBM.

4.2 並列計算機による実測

4.2.1 細粒度マルチプロセッサ MSBM[9]

ESH を評価するために，包含検索機能付き機能メモリによるハードウェアバリア同期機構（MCAMU）とマルチキャスト機能付きのグループ共有メモリ（GSM）を持った細粒度マルチプロセッサ MSBM を用いた．MSBM は**図 10** に示すようにホストコンピュータ（HC），統合インタフェース（IIU）と 16 台のプロセッサ（PU）で構成される．HC はユーザインタフェース，入出力管理などを行う．さらに IIU のための BIOS レベルの基本ソフトウェアが稼働しており，PU 管理やバリアグループ（バリア同期をとりあう PU の組）管理などを行うタスク管理ソフトウェアの基礎となっている．タスク実行のまえに必要数の PU でプロセッサグループ（単にグループ）を形成し，プログラムを転送するとともに MCAMU にそのタスクで使用する全バリアグループを登録する．そして実行終了後に PU の解放とともにバリアグループを MCAMU から削除する．IIU は HC と PU とのコード/データ転送機能と MCAMU を持っている．PU は MPU と GSM

表4 評価プログラムの速度向上比（その1）
Table 4 Speedup ratios of benchmarks (No.1).

E	アルゴリズム	M	P	プロセッサ台数 2	3	4	8	E	アルゴリズム	M	P	プロセッサ台数 2	3	4	8
1	LR + DSH	1	1	1.00	1.00	1.00	1.00	5	LR + DSH	1	2	1.06	1.06	1.06	1.06
		2	1	1.00	1.00	1.00	1.00			2	2	1.06	1.06	1.06	1.06
		3	1	1.00	1.00	1.00	1.00			3	2	1.05	1.05	1.05	1.05
		4	1	1.00	1.00	1.00	1.00			4	2	1.05	1.05	1.05	1.05
	THR + DSH	1	3	1.50	1.50	1.50	1.50		THR + DSH	1	4	1.33	1.33	1.33	1.33
		2	2	1.20	1.20	1.20	1.20			2	3	1.23	1.23	1.23	1.23
		3	3	1.50	1.50	1.50	1.50			3	4	1.25	1.25	1.25	1.25
		4	2	1.20	1.20	1.20	1.20			4	4	1.21	1.21	1.21	1.21
	ESH	1	3	1.50	2.00	2.00	2.00		ESH	1	4	1.78	1.78	1.78	1.78
		2	2	1.50	1.50	1.50	1.50			2	3	1.46	1.46	1.46	1.46
		3	3	1.50	2.00	2.00	2.00			3	4	1.54	1.54	1.54	1.54
		4	2	1.50	1.50	1.50	1.50			4	4	1.49	1.49	1.49	1.49
2	LR + DSH	1	4	1.80	2.25	2.25	2.25	6	LR + DSH	1	2	1.08	1.08	1.08	1.08
		2	4	1.80	1.80	1.80	1.80			2	2	1.08	1.08	1.08	1.08
		3	4	1.75	1.75	1.75	1.75			3	2	1.06	1.06	1.06	1.06
		4	4	1.75	1.75	1.75	1.75			4	2	1.06	1.06	1.06	1.06
	THR + DSH	1	4	1.80	2.25	3.00	3.00		THR + DSH	1	4	1.44	1.44	1.44	1.44
		2	4	1.80	2.25	2.25	2.25			2	4	1.36	1.36	1.36	1.36
		3	4	2.00	2.33	2.33	2.33			3	4	1.33	1.33	1.33	1.33
		4	4	2.00	2.00	2.00	2.00			4	4	1.30	1.30	1.30	1.30
	ESH	1	4	1.80	2.25	3.00	3.00		ESH	1	4	1.68	1.93	1.93	1.93
		2	4	1.80	2.25	2.25	2.25			2	4	1.58	1.74	1.74	1.74
		3	4	2.00	2.33	2.33	2.33			3	4	1.68	1.68	1.68	1.68
		4	4	2.00	2.00	2.00	2.00			4	4	1.63	1.63	1.63	1.63
3	LR + DSH	1	1	1.00	1.00	1.00	1.00	7	LR + DSH	1	6	1.94	2.22	2.22	2.22
		2	1	1.00	1.00	1.00	1.00			2	5	1.71	1.81	1.81	1.81
		3	1	1.00	1.00	1.00	1.00			3	6	2.00	2.53	2.68	2.68
		4	1	1.00	1.00	1.00	1.00			4	6	1.84	2.18	2.28	2.28
	THR + DSH	1	3	1.25	1.25	1.25	1.25		THR + DSH	1	6	1.94	2.40	2.61	2.61
		2	2	1.11	1.11	1.11	1.11			2	5	1.71	1.81	1.81	1.81
		3	3	1.25	1.25	1.25	1.25			3	6	1.91	2.68	3.02	3.02
		4	3	1.25	1.25	1.25	1.25			4	6	1.84	2.18	2.28	2.28
	ESH	1	3	1.43	1.67	1.67	1.67		ESH	1	6	1.94	2.40	2.61	2.61
		2	3	1.25	1.43	1.43	1.43			2	5	1.82	1.94	1.94	1.94
		3	3	1.43	1.67	1.67	1.67			3	6	2.00	2.68	3.02	3.02
		4	3	1.43	1.60	1.60	1.60			4	6	1.92	2.18	2.41	2.41
4	LR + DSH	1	4	1.20	1.20	1.20	1.20	8	LR + DSH	1	5	1.64	2.00	2.25	2.25
		2	4	1.20	1.20	1.20	1.20			2	5	1.50	1.64	1.80	1.80
		3	4	1.16	1.16	1.16	1.16			3	5	1.61	2.07	2.23	2.23
		4	4	1.16	1.16	1.16	1.16			4	5	1.61	1.93	2.07	2.23
	THR + DSH	1	9	1.56	1.56	1.56	1.56		THR + DSH	1	6	1.64	2.00	2.25	2.25
		2	9	1.50	1.50	1.50	1.50			2	5	1.38	1.64	1.80	1.80
		3	9	1.44	1.44	1.44	1.44			3	6	1.61	2.07	2.23	2.42
		4	9	1.42	1.42	1.42	1.42			4	6	1.61	1.93	2.07	2.32
	ESH	1	9	1.90	2.58	2.58	2.58		ESH	1	6	1.64	2.00	2.25	2.25
		2	9	1.80	2.12	2.12	2.12			2	5	1.50	1.64	1.80	1.80
		3	9	1.91	2.10	2.10	2.10			3	6	1.61	2.07	2.23	2.42
		4	9	1.83	2.05	2.05	2.05			4	6	1.61	1.93	2.15	2.32

E：Exam，M：プロセッサモデル，P：最大並列度

で構成され，割り当てられた逐次コードを実行する．PU はバリア同期命令を実行したとき MCAMU に同期要求を通知するとともに実行をいったん停止する．MCAMU では同期が成立していればその旨をバリアグループを構成する PU に知らせ，PU は実行を再開する．GSM はグループごとに独立した共有メモリである．同一グループに属する PU 間の通信は GSM への Read/Write によって行われる．なおバリアグループの設定は1つのタスクに1バリアグループとして実験を行った．

4.2.2 実測結果

シミュレーションの場合と同様に，表3のプログラムをもちいて，LR と DSH による並列化，THR と DSH による並列化，ESH による並列化における速度向上比を MSBM 上で実測した．MSBM はバリアによって同期を行う．そのため ESH および DSH で生成されたガントチャートに対して，演算の実行順序を守りながらバリア同期命令の挿入を行う．本来同期が必要でないプロセッサに対してはダミーのバリアを挿入する．なおバリア同期の挿入は Zaafrani のアルゴリズム[11]を用いた．

また，MSBM が共有メモリ型マルチプロセッサであることから一時変数の共有変数（一時共有変数）および恒久変数は共有メモリに，一時変数の局所変数（一時局所変数）は局所メモリに割り当てた．また**表6**に MSBM の特性を表す四則演算実行時間，バリア同期命令実行時間，プロセッサ間通信時間を示す．

表7に実測結果を示す．速度向上比は MSBM1 台

表4 評価プログラムの速度向上比（その2）
Table 4 Speedup ratios of benchmarks (No.2).

E	アルゴリズム	M	P	プロセッサ台数 2	3	4	8	E	アルゴリズム	M	P	プロセッサ台数 2	3	4	8
9	LR + DSH	1	4	1.80	2.25	3.00	3.00	13	LR + DSH	1	4	1.62	1.91	1.91	1.91
		2	4	1.50	1.80	2.25	2.25			2	4	1.40	1.62	1.75	1.75
		3	4	1.83	2.75	2.75	2.75			3	4	1.69	1.81	1.81	1.81
		4	4	1.69	2.00	2.44	2.44			4	4	1.69	1.81	1.81	1.81
	THR + DSH	1	4	1.80	2.25	3.00	3.00		THR + DSH	1	4	1.62	1.91	1.91	1.91
		2	4	1.50	1.80	2.25	2.25			2	4	1.40	1.62	1.75	1.75
		3	4	1.83	2.75	2.75	2.75			3	4	1.69	1.81	1.81	1.81
		4	4	1.69	2.00	2.44	2.44			4	4	1.69	1.81	1.81	1.81
	ESH	1	4	1.80	2.25	3.00	3.00		ESH	1	4	1.62	1.91	1.91	1.91
		2	4	1.50	1.80	2.25	2.25			2	4	1.62	1.62	1.75	1.75
		3	4	1.83	2.75	2.75	2.75			3	4	1.69	1.81	1.81	1.81
		4	4	1.69	2.44	2.44	2.44			4	4	1.69	1.81	1.81	1.81
10	LR + DSH	1	9	1.81	2.37	2.71	2.71	14	LR + DSH	1	2	1.44	1.44	1.44	1.44
		2	8	1.73	2.11	2.53	2.53			2	2	1.44	1.44	1.44	1.44
		3	9	1.89	2.61	3.09	3.23			3	2	1.59	1.59	1.59	1.59
		4	9	1.86	2.57	3.09	3.16			4	2	1.59	1.59	1.59	1.59
	THR + DSH	1	11	1.81	2.53	2.92	4.22		THR + DSH	1	2	1.44	1.44	1.44	1.44
		2	10	1.73	2.24	2.53	3.17			2	2	1.44	1.44	1.44	1.44
		3	11	1.89	2.61	3.23	4.53			3	2	1.59	1.59	1.59	1.59
		4	11	1.86	2.57	3.16	4.39			4	2	1.59	1.59	1.59	1.59
	ESH	1	13	1.81	2.53	2.92	4.22		ESH	1	2	1.44	1.44	1.44	1.44
		2	11	1.73	2.24	2.53	3.17			2	2	1.44	1.44	1.44	1.44
		3	13	1.89	2.70	3.47	4.85			3	2	1.59	1.59	1.59	1.59
		4	13	1.86	2.57	3.40	4.39			4	2	1.59	1.59	1.59	1.59
11	LR + DSH	1	12	1.90	2.57	3.11	3.68	15	LR + DSH	1	13	1.90	2.64	3.36	3.52
		2	12	1.84	2.36	2.81	3.47			2	11	1.85	2.38	2.74	3.36
		3	12	1.91	2.63	3.28	4.20			3	13	1.93	2.74	3.42	4.31
		4	12	1.88	2.53	3.13	4.03			4	13	1.93	2.70	3.24	4.31
	THR + DSH	1	24	1.84	2.45	3.11	4.53		THR + DSH	1	14	1.90	2.64	3.21	4.63
		2	24	1.84	2.36	2.81	3.69			2	11	1.80	2.56	2.96	3.70
		3	24	1.84	2.56	3.18	4.77			3	14	1.92	2.69	3.47	5.21
		4	24	1.84	2.53	3.13	4.46			4	14	1.92	2.69	3.36	4.95
	ESH	1	24	1.90	2.57	3.11	4.92		ESH	1	14	1.90	2.64	3.36	4.63
		2	24	1.84	2.36	2.81	3.69			2	11	1.85	2.56	2.96	3.90
		3	24	1.91	2.63	3.28	5.00			3	14	1.93	2.74	3.48	5.21
		4	24	1.88	2.57	3.19	4.67			4	14	1.93	2.70	3.36	4.95
12	LR + DSH	1	12	1.90	2.57	3.11	3.68	16	LR + DSH	1	8	1.75	2.15	2.33	2.33
		2	12	1.84	2.36	2.81	3.47			2	8	1.64	2.15	2.15	2.15
		3	12	1.91	2.63	3.28	4.20			3	8	1.75	2.36	2.68	2.70
		4	12	1.88	2.53	3.13	4.03			4	8	1.75	2.25	2.57	2.59
	THR + DSH	1	24	1.84	2.45	3.11	4.53		THR + DSH	1	8	1.87	2.55	2.80	3.50
		2	24	1.84	2.36	2.81	3.69			2	8	1.75	2.33	2.54	3.11
		3	24	1.84	2.56	3.18	4.77			3	8	1.87	2.55	3.11	4.03
		4	24	1.84	2.53	3.13	4.46			4	8	1.87	2.52	3.08	3.89
	ESH	1	24	1.90	2.57	3.11	4.92		ESH	1	8	1.87	2.55	3.12	4.00
		2	24	1.84	2.36	2.81	3.69			2	8	1.75	2.33	2.80	3.11
		3	24	1.91	2.63	3.28	5.00			3	8	1.91	2.55	3.24	4.42
		4	24	1.88	2.57	3.19	4.67			4	8	1.87	2.52	3.09	3.89

E：Exam，M：プロセッサモデル，P：最大並列度

表5 各プロセッサモデルのプロセッサ台数にわたる平均速度向上比
Table 5 Average speedup ratios over number of processors for processor models.

Model	アルゴリズム	Exam 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	LR	1.00	2.13	1.00	1.20	1.06	1.08	2.15	2.04	2.51	2.40	2.81	2.81	1.84	1.44	2.85	2.14
	THR	1.50	2.51	1.25	1.56	1.33	1.44	2.39	2.04	2.51	2.87	2.98	2.98	1.84	1.44	3.09	2.68
	ESH	1.88	2.51	1.61	2.41	1.78	1.87	2.39	2.04	2.51	2.88	3.13	3.13	1.84	1.44	3.14	2.89
2	LR	1.00	1.80	1.00	1.20	1.06	1.08	1.78	1.69	1.95	2.22	2.62	2.62	1.63	1.44	2.58	2.02
	THR	1.20	2.14	1.11	1.50	1.23	1.36	1.78	1.69	1.95	2.42	2.68	2.68	1.63	1.44	2.75	2.43
	ESH	1.50	2.14	1.39	2.04	1.46	1.70	1.91	1.69	1.95	2.42	2.68	2.68	1.69	1.44	2.82	2.50
3	LR	1.00	1.75	1.00	1.16	1.05	1.06	2.47	2.03	2.52	2.70	3.00	3.00	1.78	1.59	3.10	2.37
	THR	1.50	2.25	1.25	1.44	1.25	1.33	2.65	2.08	2.52	3.06	3.08	3.08	1.78	1.59	3.32	2.89
	ESH	1.88	2.25	1.61	2.06	1.54	1.68	2.68	2.08	2.52	3.23	3.21	3.21	1.78	1.59	3.34	3.03
4	LR	1.00	1.75	1.00	1.16	1.05	1.06	2.14	1.96	2.14	2.67	2.89	2.89	1.78	1.59	3.04	2.29
	THR	1.20	2.00	1.25	1.42	1.21	1.30	2.14	1.98	2.14	2.99	2.99	2.99	1.78	1.59	3.23	2.84
	ESH	1.50	2.00	1.56	2.00	1.49	1.63	2.23	2.01	2.26	3.06	3.08	3.08	1.78	1.59	3.24	2.84

のPU上で逐次実行した時間を1とした．表7からプロセッサ台数にわたって速度向上比の平均をとった結果を**表8**に示す．なお，最大並列度を超えるプロセッサ台数の速度向上比の扱いはシミュレーションと同様である．

5. 考　　察

5.1 評価プログラムの特性と速度向上比

表5，表8からプロセッサモデルにわたって速度向上比の平均をとった結果を**図11**に示す．すべての評価プログラムにおいて，ESHの速度向上比がTHR+DSHとLR+DSHを上回っているか等しい．特にESHのLR+DSHに対する速度向上比の最大はExam4のときの1.82，THR+DSHに対する最大は同様にExam4のときの1.45である．

評価プログラムの特性と速度向上比の関係を見ると，未決定ノードを持った文がそれ以前の文に真依存していれば，ESHがTHR+DSHより良い結果となる場合が多い．このような依存関係がなくても，未決定ノー

表6 MSBMの性能
Table 6 The specification of MSBM.

加算	整数演算	4.67
	浮動小数点演算	18.50
減算	整数演算	4.67
	浮動小数点演算	19.56
乗算	整数演算	10.94
	浮動小数点演算	19.56
除算	整数演算	12.61
	浮動小数点演算	25.00
アドレス修飾	配列のアドレス修飾	3.22
同期	バリア同期命令実行	1.33
通信	整数マルチキャスト	1.00
	浮動小数点マルチキャスト	1.39

単位は単位時間である

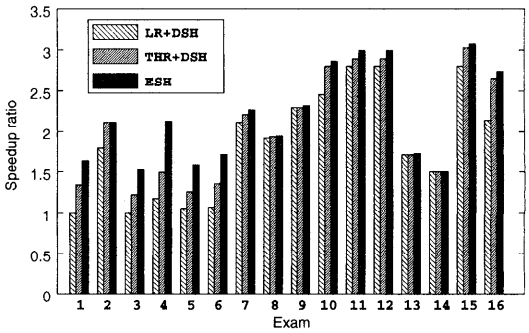


図11 評価プログラムにおける平均速度向上比
Fig. 11 Average speedup ratios for benchmarks.

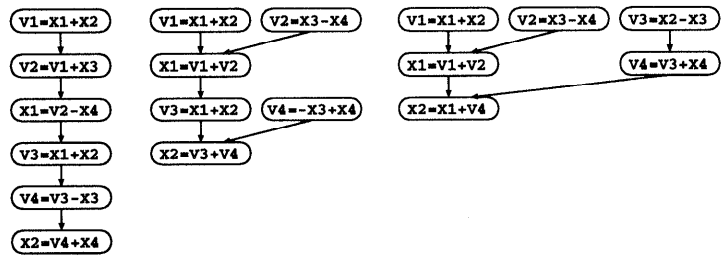
表7 MSBM上での速度向上比
Table 7 Speedup ratios on MSBM.

E	アルゴリズム	P	プロセッサ台数 2	3	4	8	E	アルゴリズム	P	プロセッサ台数 2	3	4	8
1	LR	1	1.00	1.00	1.00	1.00	9	LR	4	1.78	2.21	2.76	2.76
	THR	3	1.23	1.30	1.30	1.30		THR	4	1.78	2.21	2.76	2.76
	ESH	3	1.29	1.53	1.53	1.53		ESH	4	1.77	2.21	2.76	2.76
2	LR	4	1.56	1.63	1.63	1.63	10	LR	9	1.61	2.20	2.59	2.66
	THR	4	1.74	1.98	1.98	1.98		THR	11	1.72	2.32	2.70	3.81
	ESH	4	1.74	1.98	1.98	1.98		ESH	13	1.75	2.36	2.91	3.83
3	LR	1	1.00	1.00	1.00	1.00	11	LR	12	1.84	2.38	2.86	3.57
	THR	3	1.22	1.23	1.23	1.23		THR	16	1.73	2.19	2.81	4.03
	ESH	3	1.43	1.60	1.60	1.60		ESH	16	1.84	2.38	2.86	4.34
4	LR	4	1.18	1.18	1.18	1.18	12	LR	12	1.85	2.40	2.89	3.62
	THR	9	1.49	1.50	1.52	1.52		THR	16	1.74	2.20	2.83	4.09
	ESH	9	1.76	2.36	2.39	2.41		ESH	16	1.85	2.40	2.89	4.41
5	LR	2	1.05	1.05	1.05	1.05	13	LR	4	1.48	1.58	1.58	1.58
	THR	4	1.29	1.29	1.29	1.29		THR	4	1.48	1.58	1.58	1.58
	ESH	4	1.69	1.69	1.69	1.69		ESH	4	1.53	1.51	1.58	1.58
6	LR	2	1.06	1.06	1.06	1.06	14	LR	2	1.44	1.44	1.44	1.44
	THR	4	1.34	1.34	1.34	1.34		THR	2	1.44	1.44	1.44	1.44
	ESH	4	1.69	1.76	1.76	1.76		ESH	2	1.44	1.44	1.44	1.44
7	LR	6	1.60	1.88	2.16	2.25	15	LR	13	1.71	2.23	2.58	3.04
	THR	6	1.71	1.93	2.20	2.32		THR	14	1.80	2.42	2.94	3.88
	ESH	6	1.73	2.06	2.27	2.42		ESH	14	1.80	2.42	2.94	4.21
8	LR	5	1.54	1.79	2.02	2.11	16	LR	8	1.55	1.83	2.05	2.01
	THR	6	1.56	1.85	2.04	2.08		THR	8	1.75	2.28	2.57	3.20
	ESH	6	1.56	1.85	2.04	2.08		ESH	8	1.75	2.28	2.57	3.20

E: Exam，P: 最大並列度

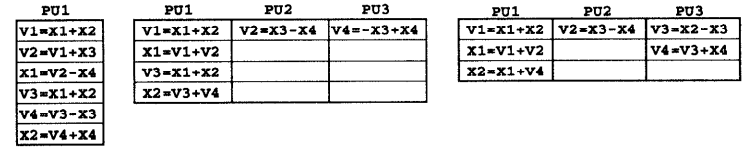
表8 MSBMのプロセッサ台数にわたる平均速度向上比
Table 8 Average speedup ratios over number of processors on MSBM.

アルゴリズム	Example 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LR	1.00	1.61	1.00	1.18	1.05	1.06	1.97	1.86	2.38	2.26	2.66	2.68	1.55	1.44	2.39	1.86
THR	1.28	1.92	1.22	1.50	1.29	1.34	2.04	1.88	2.38	2.63	2.69	2.71	1.55	1.44	2.76	2.45
ESH	1.47	1.92	1.56	2.23	1.69	1.74	2.12	1.88	2.38	2.72	2.86	2.89	1.55	1.44	2.85	2.45



(a)LR+DSH (b)THR+DSH (c)ESH

図 12 Exam1 のタスクグラフ
Fig. 12 The task graphs of Exam1.



(a)LR+DSH (b)THR+DSH (c)ESH

図 13 Exam1 のガントチャート
Fig. 13 The gantt charts of Exam1.

ドに含まれる変数が多い評価プログラムでは速度向上比の改善が見られる．このような依存関係と未決定ノードの変数の多さが速度向上比に与える影響を，評価プログラム Exam1–3 を中心に検討する．

Exam1 では変数 X2 への代入文は変数 X1 への代入文に真依存している．これを 3 台のプロセッサモデル 1 でのスケジューリングを考える．このときの LR+DSH，THR+DSH，ESH のタスクグラフとガントチャートを**図 12** と**図 13** に示す．LR+DSH では生成されたタスクグラフのすべてのノードに真依存が存在するので並列化できない．THR+DSH と ESH の第 1 文のスケジューリングは同一である．THR+DSH では第 2 文を単独で計算木の高さが最低になるようにスケジューリングしているが，ESH では変数の取得時間を考慮して 2 つの文全体の高さが最低になるようにスケジューリングしている．その結果，ESH の LR+DSH，THR+DSH に対する速度向上比は 2.00，1.33 であった．

Exam2 では変数 X6 への代入文は変数 X1，X2，X3 への代入文に真依存し，変数 X7 への代入文は変数 X1，X2，X4 への代入文に真依存している．これを 3 台のプロセッサモデル 1 でのスケジューリングを考える．変数 X1，X2，X3，X4 への代入文は 2 項演算であるので，LR+DSH，THR+DSH，ESH とも同じスケジューリング結果となった．また，変数 X6，X7 への代入文では LR+DSH と THR+DSH のスケジューリング結果は異なったが，実行時間は同一であった．ESH では変数 X6，X7 への代入文の右辺は未決定ノードであるが，変数の取得時間によって決定ノードに分解した結果 THR+DSH とまったく同じスケジューリング結果となった．

Exam3 では変数 X2, ..., X6 への代入文は 1 つ前の文に真依存している．これをプロセッサモデル 1 の 3 台のプロセッサでスケジューリングすると，Exam1 と同じ理由により ESH，THR+DSH，LR+DSH の順で良い速度向上比を得た．ESH の LR+DSH と THR+DSH に対する速度向上比はそれぞれ 1.67，1.34 であった．

Exam1 と同様な理由で ESH が THR+DSH より速度向上があった評価プログラムは Exam3，4，5，6，7，11，12 であった．プロセッサモデルによって異なるが，Exam2 と同じ理由により速度向上がなかった評価プログラムは Exam8 であった．Exam9 はほとんど未決定ノードがないため速度向上比は ESH，THR+DSH，LR+DSH にあまり差がなく，Exam14 は未決定ノードがまったくないため速度向上比は同じであった．Exam10，14，15，16 では依存関係はまったくないが，未決定ノードに含まれる変数の多さに従って ESH の速度向上比は LR+DSH や THR+DSH より良くなっている．

なお，MSBM による評価では Exam8，9，13 の速度向上比が LR+DSH や THR+DSH に比べて良くないことがある．この場合も，ESH のスケジューリング直後の速度向上比は LR+DSH や THR+DSH より

表9 プロセッサモデルにおける平均速度向上比
Table 9 Average speedup ratios over benchmark for processor models.

Model	Algorithm	プロセッサ台数				対 Algorithm	プロセッサ台数			
		2	3	4	8		2	3	4	8
Model1	LR+DSH	1.55	1.85	2.06	2.14	THR/LR	1.07	1.08	1.10	1.24
	THR+DSH	1.66	1.99	2.27	2.66	ESH/THR	1.05	1.11	1.10	1.10
	ESH	1.75	2.20	2.49	2.93	ESH/LR	1.13	1.19	1.21	1.37
Model2	LR+DSH	1.47	1.67	1.82	1.94	THR/LR	1.04	1.08	1.07	1.13
	THR+DSH	1.53	1.80	1.95	2.19	ESH/THR	1.08	1.07	1.08	1.06
	ESH	1.65	1.93	2.10	2.33	ESH/LR	1.12	1.16	1.15	1.20
Model3	LR+DSH	1.57	1.92	2.11	2.29	THR/LR	1.06	1.07	1.09	1.20
	THR+DSH	1.67	2.06	2.29	2.75	ESH/THR	1.06	1.08	1.08	1.08
	ESH	1.77	2.22	2.47	2.97	ESH/LR	1.13	1.16	1.17	1.30
Model4	LR+DSH	1.54	1.81	2.02	2.21	THR/LR	1.06	1.06	1.06	1.16
	THR+DSH	1.63	1.92	2.15	2.56	ESH/THR	1.07	1.08	1.07	1.06
	ESH	1.74	2.08	2.31	2.72	ESH/LR	1.13	1.15	1.14	1.23
MSBM	LR+DSH	1.45	1.67	1.86	1.99	THR/LR	1.08	1.08	1.09	1.19
	THR+DSH	1.56	1.81	2.03	2.36	ESH/THR	1.07	1.10	1.08	1.09
	ESH	1.67	1.99	2.20	2.58	ESH/LR	1.15	1.19	1.18	1.30

良い結果を得ている．MSBMでは1タスクに1バリアグループを設定しているので，本来バリア同期をとる必要のないプロセッサにもバリア同期命令をESHによるスケジューリング後に挿入する．このときに速度向上比の逆転が生じており，ESHによるものではない．

5.2 プロセッサモデルと台数

表4，表7から評価プログラムにわたる速度向上比の平均を算出し，さらに各スケジューリングアルゴリズム間での速度向上比を算出したものが**表9**である．表9からプロセッサ台数が増加するにつれて速度向上比も増加している．ESHのLR+DSHに対する速度向上比の最大はプロセッサモデル1の8台のときで速度向上比は1.37である．ESHのTHR+DSHに対する速度向上比の最大はプロセッサモデル1の3台のときで速度向上比は1.11であった．

次に，プロセッサモデルと速度向上比，プロセッサ台数と速度向上比を検討するために，表9をプロセッサ台数にわたって速度向上比の平均を算出した結果を**表10**に，プロセッサモデルにわたって速度向上比の平均を算出した結果を**表11**に示す．表10より，プロセッサモデルの中ではモデル1，MSBM，モデル3，モデル4，モデル2の順にESHがTHR+DSHおよびLR+DSHに対して良い速度向上比を示している．表11より，プロセッサが8台のときを除いてはESHのTHR+DSHに対する速度向上比とTHR+DSHのLR+DSHに対する速度向上比はほとんど変わらない．プロセッサが8台のときのESHのTHR+DSHに対する速度向上比は1.08，THR+DSHのLR+DSHに対する速度向上比は1.18であった．LR+DSHでは6個の評価プログラムの最大並列度が8以上であり，残りの10個の評価プログラムの速度向上比は最大並列度の速度向上比で代表している．またLR+DSHの最大並列度はESHやTHR+DSHの最大並列度に比べて小さい．このことがESHのTHR+DSHに対する速度向上比とTHR+DSHのLR+DSHに対する速度向上比の差が大きい原因である．

表10 各プロセッサモデルにおけるアルゴリズム間での速度向上比
Table 10 Speedup ratios among algorithms for processor models.

対 Algorithm	Model				
	Model1	Model2	Model3	Model4	MSBM
THR/LR	1.12	1.08	1.11	1.09	1.11
ESH/THR	1.09	1.07	1.07	1.07	1.09
ESH/LR	1.22	1.16	1.19	1.16	1.21

表11 各プロセッサ台数におけるアルゴリズム間での速度向上比
Table 11 Speedup ratios among algorithms for number of processor.

対 Algorithm	プロセッサ台数			
	2	3	4	8
THR/LR	1.06	1.07	1.08	1.18
ESH/THR	1.07	1.09	1.08	1.08
ESH/LR	1.13	1.17	1.17	1.28

一般に細粒度並列処理で使用されるプロセッサ台数は2台から4台で，その範囲では，ESHのLR+DSHに対する平均速度向上比は1.16，ESHのTHR+DSHに対する平均速度向上比は1.08，THR+DSHのLR+DSHに対する速度向上比は1.07であった．以上より，細粒度並列処理において，代入文の式の評価に一般的に使用されているLR構文解析法に対してツリーハイトリダクション法では7%しか速度向上が得られなかったが，本手法では16%の速度向上を得ることができ，有効性が確認できた．

6. むすび

変数レベルの並列化アルゴリズムESHについて述べた．ESHは複製というDSHの利点を活用しつつ，決定ノードと未決定ノードが混在した不完全タスクグ

ラフに対して変数の取得時間に基づいて2項演算をプロセッサに割り当てる．すべてのノードが決定ノードになったときすなわち完全タスクグラフが生成されたときスケジューリングが完了する．これによって複数の文間で内在している並列性を最適化し，計算木を最適に生成していくことができる．

16個の評価プログラムを用いて，4種類のプロセッサモデル上と MSBM マルチプロセッサ上で ESH と，THR と DSH の併用および LR と DSH の併用との比較を行った．これらすべてのプログラムにおいて速度向上比は ESH≧(THR と DSH の併用)≧(LR と DSH の併用) であり，ESH の有効性が確認できた．

今後の課題は，このアルゴリズムをスーパスカラプロセッサ，VLIW プロセッサ，細粒度マルチプロセッサ用コンパイラに実装することである．

謝辞 本研究にご助力いただいた（株）東芝日野工場木村勝彦氏に深謝します．

参 考 文 献

1) Johnson, M., 村上和彰（監訳）：スーパースカラプロセッサ，日経 BP 出版センター (1994).
2) El-Rewini, H., Lewis, T. and Ali, H.: *Task Scheduling in PARALLEL and DISTRIBUTED SYSTEMS*, Prentice Hall (1994).
3) Fisher, J.A.: Trace Scheduling: A Technique for Global Microcode Compaction, *IEEE Trans. Comput.*, Vol.C-30, No.7 (1981).
4) 富田眞治：並列計算機構成論，昭晃堂 (1987).
5) 笠原博徳：並列処理技術，コロナ社 (1991).
6) 尾形航他：スタチックスケジューリングを用いたマルチプロセッサシステム上での無同期近細粒度並列処理，情報処理学会論文誌，Vol.35，No.4，pp.522–531 (1994).
7) Aho, A.V., Sethi, R. and Ullman, J.D. 原田賢一（訳）：コンパイラ—原理，技法，ツール，サイエンス社 (1990).
8) 村岡洋一：超並列処理コンパイラ，サイエンス社 (1990).
9) 岩根雅彦ほか：細粒度マルチプロセッサ MSBM，情報処理学会論文誌，Vol.37，No.6，pp.1196–1205 (1996).
10) 岩根雅彦ほか：式の分割による並列化アルゴリズム ESH，情報処理学会研究会報告，Vol.96，No.80，pp.179–184 (1996).
11) Zaafrani, A., Diets, H.G. and O'Keefe M.T.: Static Scheduling for Barrier MIMD Architectures, *Int. Conf. Parallel Processing*, pp.II.187–II.194 (1990).

(平成8年9月13日受付)
(平成9年7月 1 日採録)

岩根 雅彦（正会員）

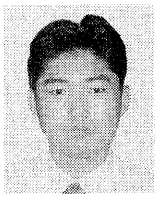
1946年生．1968年京都大学工学部数理工学科卒業．同年（株）東芝に入社．1980年同社退社後，トヨタ自動車（株），豊田工業大学を経て，1988年より九州工業大学工学部教授．工学博士．計算機アーキテクチャの研究に従事．電子情報通信学会会員．

濱田 智雄

1971年生．1994年九州工業大学工学部電気工学科卒業．1996年同大学大学院工学研究科電気工学専攻博士前期課程修了．同年松下電器産業（株）に入社．半導体の論理回路設計，DSP の研究開発に従事．

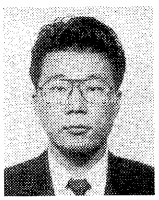
宇野 総一（学生会員）

1974年生．1997年九州工業大学工学部電気工学科卒業．現在，同大学大学院工学研究科電気工学専攻博士前期課程に在学中．自動並列化コンパイラの研究開発に従事．ハイ・パフォーマンス・コンピューティングにも興味を持つ．



小島 和広（正会員）

1971年生．1995年九州工業大学工学部電気工学科卒業．1997年同大学大学院工学研究科電気工学専攻博士前期課程修了．同年九州旅客鉄道（株）に入社．信号通信関係に興味を持つ．



松田 孝史（正会員）

1973年生．1995年九州工業大学工学部電気工学科卒業．1997年同大学大学院工学研究科電気工学専攻博士前期課程修了．同年より同大学工学部助手．計算機アーキテクチャの研究に従事．ヒューマン・インターフェイス，自然言語処理にも興味を持つ．