記 号 処 理 35－4
（1985. 12. 20）

参照カウント法を用いた並列ガーベジコレクタの構成法

An Architecture of a Parallel Garbage Collector
Using a Reference Counting Algorithm

大井 幹成　脇山 俊一郎　飯田 三郎　楠 菊信
Mikinari OOI　Shunichirou WAKIYAMA　Saburou IIDA　Kikunobu KUSUNOKI

豊橋技術科学大学 情報工学系
Department of Information and Computer Sciences
Toyohashi University of Technology

1．はじめに

我々は並列処理技術の観点から、Ｌｉｓｐを対象とした関数型言語向き計算機の開発を行ってきた[1]。本稿では、この計算機の並列ガーベジコレクタのアルゴリズム及びそのハードウェア構成法について報告する。本計算機は、

（1）コンパイル方式、
（2）関数に適用する引数の並列評価、
（3）データ駆動方式による引数の関数への適用、
（4）マッカーシの条件式の逐次評価、

に特徴づけられる。

並列評価の導入により、リストを表現しているセル空間へのポインタが、ハードウェア上の各所に分散する。従って不要セルを回収するガーベジコレクタには、これらのポインタ情報を効率的に収集することが必要とされる。リスト処理言語が実用に供されて以来、時間的及び空間的に効率の良いガーベジコレクタの各種のアルゴリズムが研究され、実装されてきた。これらの著名なアルゴリズムは、

（1）一括型
　1）再帰的マーキング法[2]
　2）ポインタ反転法[3]
（2）即時型
　3）参照カウント法[2]
　4）移動法[4]
（3）並列型
　5）Steeleのアルゴリズム[5]
　6）Dijkstra他のアルゴリズム[6]
　7）Kung & Songのアルゴリズム[7]
　8）日比野のアルゴリズム[8]
　9）薄他のアルゴリズム[9]

に分類される。

一括型、即時型の移動法及び並列型は、いずれも回収してはならないリストを指し示すルート（実際にはスタック等）をシステム内に保持し、このルートを起点として回収が行われる。一方、即時型アルゴリズムである参照カウント法は空間的オーバーヘッドが大きいが、このルートを必要とせずに不要セルの回収が可能である。従って本計算機の様にポインタ情報がハードウェア上の各所に分散するシステムでは、参照カウント法が最も望ましいアルゴリズムであると考えられる。

本計算機のガーベジコレクタが用いている参照カウント法は次の２点において機能が拡張されている。

（1）循環リストの回収機能
　参照カウント法は本来循環リストが回収できないが、マーキング法を併用することにより、循環リストを回収できるようにした。
（2）並列処理機能
　本計算機のセルを構成するメモリは、（コンパイルしたプログラムを格納する）関数定義メモリ及び（スタックに相当する）ノード制御メモリとは独立しており、セルへのアクセスの際にのみ用いられる。従って通常の計算機よりメモリアクセスの頻度

は低減しており、リスト処理とガーベジコレクタの並列処理が十分機能するものと考えられる。又、参照カウント法に基づいた並列ガーベジコレクタはアルゴリズムにおいても興味ある問題を提供するものと考えられる。

以下、２．において本計算機のアーキテクチャについて概観し、３．において並列ガーベジコレクタのアルゴリズムについて、４．において並列ガーベジコレクタのハードウェア構成について述べる。

２．関数型言語向き計算機

ここでは、並列ガーベジコレクタの実装対象となる計算機の概要について述べる。本計算機は、１．で述べたようないくつかの特徴を有しているが、基本的には並列ガーベジコレクタにおけるList Processor或いはMutator[6]の概念に対応している。

２－１．機械命令

利用者の定義した関数は、コンパイラにより、本計算機固有の機械命令列に変換され、実行される。１機械命令列は、１０のフィールドＯＰ，ＯＰＲ，Ｐ，ＯＰＣ，Ｆ，Ｕ，Ｖ，Ｓ，Ｄ，Ｌから構成される。１つのフィールドは、そのフィールドの意味により１ビットから数１０ビットのビット長を有する。リスト処理関数ｅｑｕａｌの場合の機械命令列の例を図１に示す。ここでは、主要フィールドについてのみ意味を説明する。

（１）フィールドＯＰ(Operation code)は、関数名をコード化したものであり、関数名ごとに一意のコードがコンパイラにより付与される。

（２）フィールドＯＰＲ(OPeRand)は、ＯＰで定められた関数に適用する引数又は関数の評価結果である値を保持する。図１においてＯＰＲの＃印は、関数に引数を適用する際の実引数の埋め込み位置を示す、

（３）フィールドＵ(Undefined number)は、この機械命令で使用する引数の未評価数を示す。引数は並列に評価されるので、評価が完了するごとにＵの値は１つずつ減じられ、データ駆動

```
(DEFUN EQUAL (X Y) (COND ((ATOM X) (EQ X Y))
                         ((ATOM Y) NIL)
                         ((EQUAL (CAR X) (CAR Y))
                          (EQUAL (CDR X) (CDR Y)))
                         (T NIL)))

    P   OP     OPC   U   V   S   D   L     F   OPR-0        OPR-1

*(ATOM X)
    1   EQUAL  0     1   0   0                  #OPR-0       #OPR-1
    0   CONDX        1   0   1   1   F
    0   ATOM          0   0   1   0   OPR-0     #OPR-0
        GC                                     #OPR-0(+1)

*(EQ X Y)
    0   EQ            0   0   0                  #OPR-0       #OPR-1

*(ATOM Y)
    1   EQUAL  2     1   0   0                  #OPR-0       #OPR-1
    0   CONDX        1   0   1   1   F
    0   ATOM          0   0   1   0   OPR-0     #OPR-1
        GC                                     #OPR-1(+1)

*NIL
    0   NOP           0   1   0                  NIL
        GC                                     #OPR-0(-1)   #OPR-1(-1)

*(EQUAL (CAR X) (CAR Y))
    1   EQUAL  4     1   0   0                  #OPR-0       #OPR-1
    0   CONDX        1   0   1   1   F
    0   EQUAL  0     2   0   1   1   OPR-0
    0   CAR           0   0   1   0   OPR-0     #OPR-0
    0   CAR           0   0   1   0   OPR-1     #OPR-1
        GC                                     #OPR-0(+1)   #OPR-1(+1)

*(EQUAL (CDR X) (CDR Y))
    0   EQUAL  0     2   0   0
    0   CDR           0   0   1   0   OPR-0     #OPR-0
    0   CDR           0   0   1   0   OPR-1     #OPR-1

*T/NIL
    0   NOP           0   1   0                  NIL
        GC                                     #OPR-0(-1)   #OPR-1(-1)
```

図１．機械命令列

方式に則りＵ＝０となった段階でこの機械命令が起動される。

（４）ＯＰがＧＣである機械命令は、ガーベジコレクタ用の命令であり、関数を起動する際、関数本体のコピーにより生じる引数の増減がＯＰＲの括弧内に示される。

２－２．ハードウェア構成

ハードウェア構成を図２に示す。構成は２つのバス、Ａ－バス及びＤ－バスに演算モジュール群を接続する方式に基づいている。Ａ－バス及びＤ－バスのビット幅は、それぞれ１２８ビットであり、機械命令列はパケットとしてこのバスを用いて転送される。図中の矢印は、パケットの転送方向を示す。パケットの形式は、図３に示す通り機械命令列の形式に準じる。演算モジュール群は、関数適用部ＦＡ(Function Applier),条件式評価部ＥＣＥ(Evaluater of Conditional Expression),基本演算部ＦＯ(Fundamental Operator),ノード制御部ＮＣ(Node Controller)及びフロントエンドプロセッサ部ＦＥＰ(Front End Processor)に大別される。

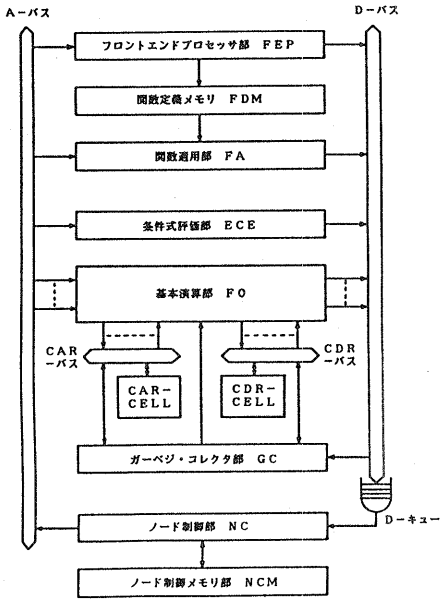

図2．ハードウエア構成

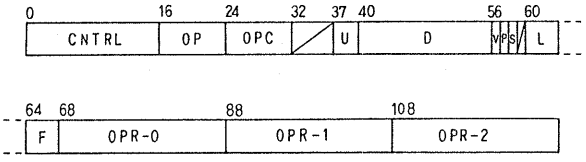

図3．パケットの形式

（1）関数適用部

関数適用部は、本体及び関数定義メモリＦＤＭ(Function Definition Memory)から構成される。ＦＤＭには計算に先立ち、使用するすべての関数の機械命令列がロードされる。本体は、Ａ－バスより関数に適用する引数を受け取り、この関数に対応する機械命令列を逐次ＦＤＭから読みだしながら引数を機械命令内の該当するフィールドＯＰＲに埋め込みＤ－バスへ送出する。即ち、関数適用部は関数本体のコピーを生成する。

（2）条件式評価部

条件式評価部は、条件式述部の真偽に応じ対応する条件式値部又は次の条件式述部の起動を行う。

（3）基本演算部

基本演算部の演算モジュール群は、リスト操作のための基本関数（ｃａｒ，ｃｄｒ，ｃｏｎｓ等）、加減算及び数値の大小比較演算より構成されている。これらのモジュールは、いずれもＡ－バスから入力したパケットのＯＰＲを実引数として、定められた演算を行った後、結果を出力パケットとしてＤ－バスへ出力する。

（4）ノード制御部

ノード制御部は、本体及びノード制御メモリＮＣＭ(Node Control Memory)から構成され、関数の評価過程において生成或いは消滅する機械命令を多進木の節に対応させ、この多進木を制御する。即ち、関数の評価は、根から開始し、節の伸縮を繰り返し、根に戻った時点で終了する。この際、機械命令のフィールドＵが０でない命令はＮＣＭに貯えられ、０となった時点でＡ－バスへ送出される。

（5）フロントエンドプロセッサ部

フロントエンドプロセッサ部は、計算開始のパケットを生成すると共に、計算終了時に、Ａ－バスから関数の評価結果を受け取る。又ＦＥＰは、プログラムをコンパイルし、機械命令列をＦＤＭ上へ格納する。

３．並列ガーベジコレクタ：アルゴリズム

ここでは、前述した計算機をｍｕｔａｔｏｒと呼び、このｍｕｔａｔｏｒの処理と並行して不要セルを回収するプロセッサをｃｏｌｌｅｃｔｏｒと呼ぶ。図４に示すようにｍｕｔａｔｏｒとｃｏｌｌｅｃｔｏｒは、Ｄ－バス及びＧ－キューを介して接続され、Ｃｅｌｌ－Ｓｐａｃｅへアクセスする。Ｃｅｌｌ－Ｓｐａｃｅは、Ｐａｓｃａｌ風言語で次のように定義される。

```
Var  CS : array[1..cmax] of
          record  left : 1..cmax;
                  right: 1..cmax;
                  rrc  : integer;
                  crc  : integer;
                  mark : boolean  end.
```

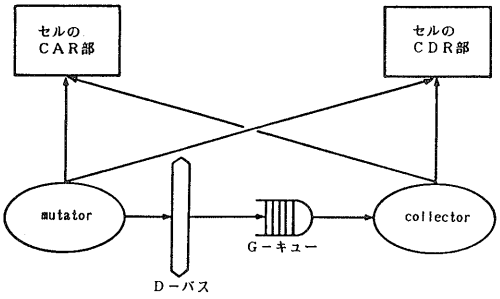

図4．ｍｕｔａｔｏｒとｃｏｌｌｅｃｔｏｒ

Ｃｅｌｌ－Ｓｐａｃｅの基本単位であるセルの形式を図５に示す。本システムでは不要セルの回収は、２つの参照カウンタｒｒｃ(root reference counter)及びｃｒｃ(cell reference counter)を用いて行われる。

ｒｒｃは、セル空間外から該当セルへの参照数を記録する。即ち、従来のルート（スタック等）を起点としたマーク付けの代わりに、ｒｒｃはルートに対応する機能を果たす。このことはセル空間の空間的オーバヘッドを増加させるが、ハードウェア上に分散しているセルへのポインタを一括して表現できる点に特徴がある。

ｃｒｃは、セル空間内における該当セルへの参照数を記録する。

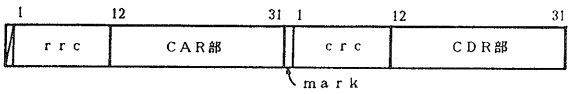

図5．セルの形式

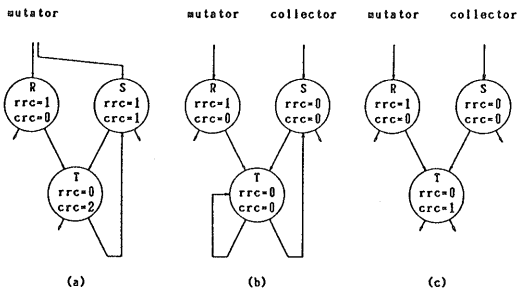

図6．ｒｒｃとｃｒｃ

図６-(a)にｒｒｃ，ｃｒｃの例を示す。セルＲ，Ｓ，Ｔは、円で示されている。

３－１．ｍｕｔａｔｏｒ

ｍｕｔａｔｏｒは、各種のリスト処理を行うが、これらは以下に述べる５つの極めて単純な手続きから構成される。

Ｐ１：ｃａｒ（ｘ）（またはｃｄｒ（ｘ））

Ａ－バスからセルへのポインタｘを受け取りｚ：＝ｃａｒ（ｘ）（又はｚ：＝ｃｄｒ（Ｘ））を行った後、ポインタｘ及びｚをＤ－バスへパケットとして転送する。

Ｐ２：ｃｏｎｓ（ｘ，ｙ）

Ａ－バスからセルへのポインタｘ及びｙを受け取り、フリーセルへｘ及びｙを書き込む。その後、ポインタｘ及びｙをＤ－バスへ送出する。

Ｐ３：ｒｐｌａｃａ（ｘ，ｙ）（又はｒｐｌａｃｄ（ｘ，ｙ））

Ａ－バスからセルへのポインタｘ及びｙを受け取り、ｒｐｌａｃａ（ｘ，ｙ）（又はｒｐｌａｃｄ（ｘ，ｙ））を行う。その後、ｙ及び切り離されたポインタｚをＤ－バスへ送出する。

Ｐ４：ポインタ数増加

Ｐ５：ポインタ数減少

これらの手続きは、関数定義部で関数本体のコピーが生成された際に、ポインタ数の増減をｃｏｌｌｅｃｔｏｒが把握するために用いられる。コピーされたポインタｘ及びポインタ数の増減ｎをＤ－バスに送出する。これは、図１のフィールドＯＰがＧＣである機械命令に対応している。

図７にこれらの手続きを示す。図中の手続きｒｅｍｏｖｅ－ｃｅｌｌ（ｚ）は、フリーセルｚを獲得することである。セルｚのフィールドｒｒｃ，ｃｒｃ及びｍａｒｋはこの手続き内で１，０，及びｆａｌｓｅにそれぞれ初期化されるものとする。

```
begin
  case P of
       P1: begin z:=CS[x].left (or z:=CS[x].right);
                 send (P1, x, z) to D_BUS end;
       P2: begin remove_cell(z);
                 CS[z].left:=x; CS[z].right:=y;
                 send (P2, x, y) to D_BUS end;
       P3: begin z:=CS[x].left (or z:=CS[x].right);
                 CS[x].left:=y (or CS[x].right:=y);
                 send (P3, y, z) to D_BUS end;
       P4: begin send (P4, x, n) to D_BUS end;
       P5: begin send (P5, x, n) to D_BUS end
  end
end
```

図7．ｍｕｔａｔｏｒのアルゴリズム

３－２．ｃｏｌｌｅｃｔｏｒ

ｍｕｔａｔｏｒによりＤ－バスへ送出されたパケットは、一度Ｇ－キューへ貯えられる。ｃｏｌｌｅｃｔｏｒは、これらのパケットをＧ－キューから取り出し、２つの参照カウンタを更新し、不要リストを構成するセルを回収する。例えば、図６(a)においてｍｕｔａｔｏｒがｃｄｒ（Ｒ）を行った場合、セルＲ及びＴのｒｒｃは、ｃｏｌｌｌｅｃｔｏｒによりそれぞれ１づつ減少、増加させられる。その後、ｃｏｌｌｅｃｔｏｒは、Ｒを入口としてリストをたどりながら子のｃｒｃを１減じ、ｒｒｃ＝ｃｒｃ＝０のセル（セルＲ）を回収する。このリストたどりは、ｒｒｃ＞０のセル（セルＴ）に出会った場合終了する。このアルゴリズムを図８に示す。図中において、手続きｒｒｃ－ｕｐ（ｘ）或いはｃｒｃ－ｕｐ（ｘ）はセルｘのｒｒｃ又はｃｒｃを１増加させるものであり、ｒｒｃ－ｄｏｗｎ（ｘ）或いはｃｒｃ－ｄｏｗｎ（ｘ）は、１減少させるものである。手続きａｐｐｅｎｄ－ｃｅｌｌ（ｘ）は、回収したセルｘをフリーリストにつなぐ。

上記のリストたどりにおいて、ｃｒｃ＞０のセル’ｘ’に出会った場合、このセルは回収可能な循環リストに含まれている可能性があるので、マーキング法を併用して回収の可否を調べる。最初にｒｒｃ＝０の条件のもとで、ｃｒｃを１づつ減じつつ、’ｘ’から到達可能なすべてのセルにマークを付ける。次にこのマークづけられたセルの集合のうちｃｒｃ＞０のセルを探し出し、該当セル及びその子孫のマークを消す。最後に’ｘ’から到達可能なマーク付けられたセルを回収する。このアルゴリズムを図９に示す。Ｔ－ＳＰＡＣＥは作業用領域であり、ｃｒｃ＞０のセルの探索に用いられる。｛ｐ｝を有する文については次に述べる。

３－３．並列性

ｍｕｔａｔｏｒ及びｃｏｌｌｅｃｔｏｒが並列処理を行った場合、３－１，３－２で述べたアルゴリズムのままでは、生きているリストの回収が生じる場合がある。このことはｍｕｔａｔｏｒが３－１のＰ３で述べたｒｐｌａｃａ，或はｒｐｌａｃｄ演算を行ってリスト構造を書き換えた場合、この書き換えの副作用をｃｏｌ

```
begin
  repeat
    begin while G_QUEUE is empty do nothing;
          remove (P, x, y) from G_QUEUE;
          case P of
               P1: begin rrc_down(x); rrc_up(y);
                         collect(x) end;
               P2: begin rrc_down(x); crc_up(x);
                         crc_down(y); crc_up(y) end;
               P3: begin rrc_down(x); crc_up(x);
                         crc_down(y);
                         collect(y) end;
               P4: begin for i:=1 to y do rrc_up(x) end;
               P5: begin for i:=1 to y do rrc_down(x);
                         collect(x) end
          end
    end
  forever
end
procedure collect(x);
begin if (x <> NIL) and (CS[x].rrc = 0) then
        if CS[x].crc = 0
          then {zero crc}
               begin crc_down(CS[x].left);
                     collect (CS[x].left);
                     crc_down(CS[x].right);
                     collect (CS[x].right);
                     append_cell(x) end
          else {crc greater than zero}
               begin crc_up(x); m:=0;
                     mark(x); turn_off; reclaim(x) end
end
```

図8．ｃｏｌｌｅｃｔｏｒのアルゴリズム

図9．循環リストの回収

```
procedure mark(x);
begin if (x <> NIL) and (CS[x].rrc = 0) then
          begin crc_down(x); m:=m+1; T_SPACE[m]:=x;
                if not CS[x].mark then
                   begin CS[x].mark:=true;
                         mark(CS[x].left);
                         mark(CS[x].right) end
          end
end
procedure turn_off;
begin for i:=1 to m do
          if CS[T_SPACE[i]].crc > 0 then
             turn_off_sub(T_SPACE[i]);
      for i:=1 to m do crc_up(T_SPACE[i]);
      set no. of packets inserted into G_QUEUE to n; {p}
      for i:=1 to n do                               {p}
          if G_QUEUE[i].P = P3 then                  {p}
             begin turn_off_sub(G_QUEUE[i].y);       {p}
                   tuen_off_sub(G_QUEUE[i].z) end    {p}
end
procedure turn_off_sub(x);
begin if (x <> NIL) and CS[x].mark then
          begin CS[x].mark:=false;
                turn_off_sub(CS[x].left);
                turn_off_sub(CS[x].right) end
end
procedure reclaim(x);
begin if x <> NIL then
          if CS[x].mark
             then begin CS[x].mark:=false;
                        reclaim(CS[x].left);
                        reclaim(CS[x].right);
                        append_cell(x) end
             else crc_down(x)
end
```

ｌｅｃｔｏｒが認識するまでにタイムラグがあることに起因する。手続きＰ３に関与するポインタは、つながれたポインタｙと切り離されたポインタｚの２つであり、誤った回収はｙとｚに応じた２つの場合が考えられる。例えば図６(a)から出発してｍｕｔａｔｏｒ（Ｍ）とｃｏｌｌｅｃｔｏｒ（Ｃ）が、次のような系列で手続きを実行したとする。

場合１：つながれたポインタによる誤った回収

Ｍ：ｃｄｒ（Ｓ）の実行
Ｃ：ｃｏｌｌｅｃｔ（Ｓ）の開始
Ｍ：Ｐ４（Ｒ，１）に引き続きｒｐｌａｃａ（ｃｄｒ（Ｒ），ｃｄｒ（Ｒ））の実行
Ｃ：ｍａｒｋ（Ｓ）の開始

場合２：切り離されたポインタによる誤った回収

Ｍ：ｃｄｒ（Ｓ）の実行
Ｃ：ｍａｒｋ（Ｓ）から復帰
Ｍ：ｒｐｌａｃｄ（ｃｄｒ（Ｒ），任意のポインタ）の実行
Ｃ：ｔｕｒｎ－ｏｆｆの開始

ｃｏｌｌｅｃｔｏｒが手続きｒｅｃｌａｉｍ（Ｓ）を開始する直前のセルＲ，Ｓ及びＴの関係を図６(b)（場合１）及び図６(c)（場合２）に示す。場合１ではＳとＴが、場合２ではＳが誤って回収される。この誤った回収を防ぐため、図９の｛ｐ｝を付した文が必要とされる。

４．並列ガーベジコレクタ：ハードウェア構成

３－２及び３－３で述べたアルゴリズムは、ミニコンピュータＵ－２００上で実行される。セルのｃａｒ部・ｃｄｒ部及びＧ－キューへのアクセスは，Ｕ－２００の共通バスＵ－バス（データ転送のビット巾：１６ビット）を介して行われる。この共通バスには、ハードウェア上次の機能が要求される。

（１）セル（ｃａｒ部又はｃｄｒ部）へのアクセス

リード時には、Ｕ－バスからセルのアドレスを送出し、リードした内容をＵ－バスから取り込む。ライト時には、Ｕ－バス上にセルのアドレス及び書き込みデータを送出する。

（２）Ｇ－キューへのアクセス

ｍｕｔａｔｏｒがＤ－バスへ送出したバケットは、Ｇ－キューへ貯えられた後、Ｕ－バスを介して順次取り出される。Ｄ－バスとの間のインターフェイスを図１０に示す。Ｇ－キュー本体はＳ－ＲＡＭにより構成され、３つのカウンタｔｏｐ　ｃｏｕｎｔｅｒ，ｂｏｔｔｏｍ　ｃｏｕｎｔｅｒ及びｄｉｆｆｅｒｅｎｃｅ　ｃｏｕｎｔｅｒにより制御される。

★ｔｏｐ　ｃｏｕｎｔｅｒは、キューに書き込まれているバケットの先頭アドレスを指す。

★ｂｏｔｔｏｍ　ｃｏｕｎｔｅｒは、バケットを次に書き込むキューのアドレスを指す。

★ｄｉｆｆｅｒｅｎｃｅ　ｃｏｕｎｔｅｒは、ｂｏｔｔｏｍ　ｃｏｕｎｔｅｒとｔｏｐ　ｃｏｕｎｔｅｒの値の差を示し、現在キューに書か

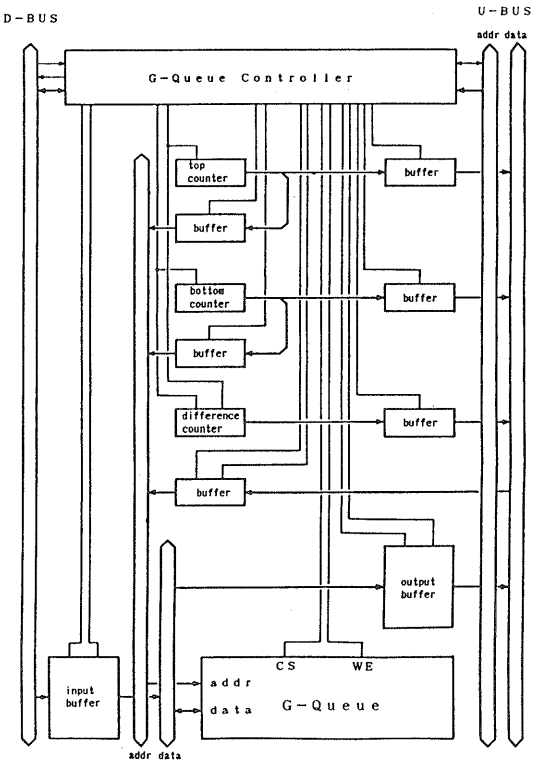

図１０．ガーベジ・コレクタのハードウエア構成

れているバケット数を示す。

Gーキューのアクセスに際し、ｃｏｌｌｅｃｔｏｒが必要とする機能は、次の通りである。

★キューからのバケットの取り出し

この機能は、ｄｉｆｆｅｒｅｎｃｅ　ｃｏｕｎｔｅｒの値が０でない場合、ｔｏｐ　ｃｏｕｎｔｅｒの示す値をキューのアドレスとしてキューの内容を取り出すことにより行われる。

★キュー内のサーチ

この機能は、生きたセルの誤った回収を防ぐため必要である。即ち、図９の｛ｐ｝を付した文を実行する際、ｔｏｐ　ｃｏｕｎｔｅｒを起点としてｄｉｆｆｅｒｅｎｃｅ　ｃｏｕｎｔｅｒの示すバケット数だけキュー内をサーチする。このサーチにおいて、ｒｐｌａｃａ或いはｒｐｌａｃｄ演算が行われているバケットを見つけた場合は、該当するセルのマークをはずして誤った回収を防ぐ。

ハードウェア本体は，ＴＴＬ－ＩＣを用いて構成されている。これらのＩＣは、約２０＊３０ｃｍ^{2}の大きさの基板５枚上に搭載され、ラッピングにより布線されている。

５．まとめ

本稿では、我々が開発した関数型言語向き計算機上の並列ガーベジコレクタのアルゴリズム及びその構成法について報告した。今後は、このアルゴリズムの正当性の検証及び本アーキテクチャの性能評価を行う予定である。これらに関しては、順次、別途報告する。

参考文献

（１）大井　幹成他：関数型言語向き計算機アーキテクチャの一方式，東北大通研シンポジウム　「新しい計算機アーキテクチャ」，pp.13-17 (1985).

（２）Knuth,D.E.:The Art of Computer Programming,Vol.1:Foundamental Algorithms, Addison-Wesley,Reading,Mass (1973).

（３）Schorr,H. and Waite,W.:An Efficient Machine-independent Procedure for Garbage Collection in Various List Structure,Comm,ACM,10 (8) pp.501-506 (1967).

（４）Baker,H.G.:List Processing in Real Time on a Serial Computer,Comm.ACM.21(4) pp.280-294 (1978).

（５）Steele,G.L.:Multiprocessing Compactifying Garbage Collection,Comm.ACM, 18 (9),pp.495-508 (1975).

（６）Dijkstra,E.W. et al.:On-the-fly Garbage Collection:An Exercise in Cooperation,in Lecture Note in Computer Science 46,pp.43-56,Springer-Verlag,New York (1976).

（７）Kung,H.T. and Song,S.W.:An Efficient Garbage Collection System and its Correctness Proof,Proc.18th Annual Symposium on Foundation of Computer Science pp.120-131 (1977).

（８）Hibino,Y.:A Parallel Garbage Collection Algorithm and its Application to Lisp,Trans.IECE Japan,E63 (1),pp.1-8 (1980).

（９）薄　隆他：マルチマイクロプロセッサによるＬｉｓｐマシン，信学技報，EC 78-33,pp.31-38 (1978).