同期構造を埋め込んだ SIGMA-1 用高級言語 DFCII†

関口智嗣†† 島田俊夫†† 平木 敬††

電子技術総合研究所において，次世代のスーパーコンピュータを目指した科学技術計算用データ駆動計算機 SIGMA-1 の開発を進めてきている．ユーザ用高級言語として，これまでに DFC (Data Flow C) の開発を行った．DFC は一般のユーザが容易に SIGMA-1 を利用するための言語という設計の立場をとり，C言語との互換性を維持しつつ，単一代入則の導入により副作用の排除を行ったものである．しかし，並列処理において必要な同期記述能力に関して，DFC は変数の定義参照関係による同期能力しか実現されていない．したがって，式相互，文相互，文内部相互，関数相互の同期といった変数の依存関係が全くないものに関しては，疑似的な変数を工夫して用いる必要がある．われわれは，C言語のシンタクスを利用して，並列処理可能な部分をコンパイラによりデータ依存関係として抽出するような言語 DFCII (Data Flow C II) の設計を行った．プログラムの字面から，逐次解釈を行うため，文相互，文内部相互，関数間の同期を記述することが可能となる．さらに，SIGMA-1 のようなデータ駆動計算機では，データ依存関係からデータ駆動計算機が並列性を自然に抽出して並列実行を行うことができる．本論文ではこの DFCII について，SIGMA-1 を対象とした言語仕様の設計と要求される記述能力を示す．

1. はじめに

並列処理による高速計算を目指して，様々な計算機アーキテクチャが提案されてきている．この中でもデータ駆動モデルに基づく命令レベルデータ駆動計算機は命令レベル，文レベル，関数レベルの並列性をアーキテクチャが自然に抽出して実行するという点で次世代の並列処理技術に不可欠なアーキテクチャの要素技術として注目されている．

電子技術総合研究所においては，次世代のスーパーコンピュータを目指した科学技術計算用データ駆動計算機 SIGMA-1 の開発を進めている[1,2]．SIGMA-1 は数多くの実用規模アプリケーション問題に適用して，性能を評価することにより，データ駆動計算機の実用性を検証することが目標である．

このようなデータ駆動計算機に対するユーザ用高級言語として，VAL[3]，Id[4]，SISAL[5]，Valid[6] などが提案，実装されてきた．これらに共通する性質として，高級言語レベルでの関数性の保持とそれにともなう単一代入規則があげられる．これらの性質により，高級言語により直接データ依存関係を記述した有向グラフ，すなわちデータフローグラフを記述することが可能となる．

SIGMA-1 においても，実用規模アプリケーションを記述するためのユーザ用高級言語として DFC (Data Flow C) の開発を行った[2]．DFC は一般のユーザが容易に SIGMA-1 を利用するための言語という設計の立場をとり，C言語[1] との互換性を維持しつつ，単一代入則の導入により副作用の排除を行ったものである．

しかし，並列処理において必要な同期記述能力に関して，DFC は変数の定義参照関係による同期能力しか実現されていない．したがって，式相互，文相互，文内部相互，関数相互の同期といった変数の依存関係が全くないものに関しては，疑似的な変数を工夫して用いる必要がある．

すなわち，DFC のように，並列処理が可能な部分の抽出を単一代入則に依存した言語では，並列性の抽出がコンパイラにとって容易となるものの，逆に並列性の制御や同期の記述能力に欠ける部分がある．

われわれは，C言語のシンタクスを利用して，並列処理可能な部分をコンパイラによりデータ依存関係として抽出するような言語 DFCII (Data Flow C II) の設計を行った．プログラムの字面から，逐次解釈を行うため，文相互，文内部相互，関数間の同期を記述することが可能となる．また，SIGMA-1 のようなデータ駆動計算機では，データ依存関係からデータ駆動計算機が並列性を自然に抽出して並列実行を行うことができる．

本論文ではこの DFCII について，SIGMA-1 を対象とした言語仕様の設計と要求される記述能力を示す．

† A Design of a Dataflow Language DFCII for New Generation Supercomputers by SATOSHI SEKIGUCHI, TOSHIO SHIMADA and KEI HIRAKI (Computer Architecture Section, Computer Science Division, Electrotechnical Laboratory).

†† 電子技術総合研究所情報アーキテクチャ部計算機方式研究室

2. DFCII の設計方針

DFCII の設計に当たっては，次の5点に特徴を与えた．

（1） C言語との互換性
（2） 並列処理記述言語としての汎用性
（3） 逐次解釈とデータ駆動原理に基づく並列実行
（4） ユーザ制御による副作用の扱い
（5） 多様な同期の記述能力

(1)に関しては，DFCII による記述対象がアプリケーションプログラム，システムプログラム，ハードウェア診断プログラムといった広範囲なものであることとユーザにとってある程度のなじみがあるという理由によりC言語を基にした．

(2)に関しては，言語を設計する上で，科学技術計算のいくつかの分野を代表するアプリケーションプログラムをとりあげ，解析を行った．

具体的には，格子ゲージ理論による色量子力学の数値シミュレーション QCD[8]，プラズマ粒子モデル等で粒子と場を交互に扱う PIC[9]，ボルツマン輸送方程式に基づく中性子輸送問題のモンテカルロ計算による解法 KENO[10]，科学技術計算用の数学ライブラリである LINPACK[11] などである．

これらのプログラムは科学技術計算を並列処理する場合に必要な基本的な手法[14]が含まれている．

(3)に関しては，従来の逐次処理言語がいずれも文法的に並列性を明示したり，単一代入則などでユーザに並列性の記述を要求していた．しかし，DFCII では単一代入則を用いることなく，データ依存関係をコンパイラが自動的に抽出する．すなわち，C言語のように $a=a+1$ といった再代入を記述した場合，これ以降の変数 a の参照は新たに代入された a に対して行う．このことは，DFCII で記述された文の並ぶ順序に意味があり，C言語と同様な逐次解釈が行われることを意味する．さらに，コンパイラによって抽出されたデータ依存関係は，データ駆動計算モデルにより並列に実行される．したがって，プログラム中で特にユーザが逐次化を指定しない限り並列性を明示しなくても，並列実行用のオブジェクトが自然に生成される．このように，DFCII 言語処理系は逐次解釈に基づいてデータ依存関係を抽出してコード生成を行う[15]．

また，for 文のようにループ構造を持つ文では，世代をまたがる変数に対して単一代入則を何らかの形で破らざるを得ない．new，old といったキーワードを用いた Id や SISAL，場所を限定して単一代入則を破った DFC のような工夫が必要であった[2]．このように限定的ではあるが，単一代入則を破る記述は極めて不自然である．DFCII では逐次解釈を導入することにより，世代をまたがる変数に関してもCと同じ記述が可能となる．

(4)と(5)は(3)とも関連する．DFC などの関数型言語では本来，副作用との親和性が悪い．したがって，コンパイラは副作用を含む部分を禁止するか，またはシステムが適当にサポートを行うという手法をとってきた．しかし，システムの介入には限界があり，副作用の制御を十分に行えなかった．そこで，逐次化による順序付けをユーザが同期として記述することにより，ユーザの裁量に任された副作用を扱えるようにする．(1)の方針ともあわせて，従来からのプログラム資産をそのまま DFCII でコンパイルすることが容易になる．

3. DFCII の配列構造と記憶クラス

3.1 I構造とB構造

科学技術計算で最も基本的なデータ構造である配列操作の並列処理を考える．

関数型言語に自然な形で配列を導入したのが Arvind らのI構造[7]である．これは，書き込みが一度，読み出しが一度という配列であり，単一代入則を配列に拡張した形式になっている．

また，島田らはB構造[2]を提案し SIGMA-1 への実装を行った．これは，書き込みが一度で，読み出しが複数回という配列である．I構造が読み出すことで配列の当該要素が未定義になるのに対して，B構造は一度書き込まれたデータは読み出しによって変化しない．しかし，I構造では読み出しによって記憶領域が解放されるが，B構造では明示的に記憶領域を解放する必要がある．

このいずれの配列も同一要素への書き込みが発生した場合，元の配列全体をコピーして新たな配列とした上で書き込みを行う．配列用の記憶領域が無限であれば，コピーによる資源の消費は問題とならないが，実際には使われなくなった古いコピーを回収する必要が生じる．DFC におけるB構造ではコピー回数を現実的に減少させる工夫を行っているが，一般に回収時期の検出は困難である．

3.2 I構造の制限的利用

I構造に関して，書き込み操作と読み出し操作の関

係から見直してみる．それぞれの操作は

1．書き込み操作によりデータの存在ビット（Pビット）を立てる，
2．Pビットが立っている要素に書き込み操作を行った場合はエラーとする，
3．読み出し操作によりPビットが立っている場合は，そのデータを読み出してPビットを降ろす，
4．読み出し操作の時点でPビットが立っていない場合，読み出し操作は保留される，

という関係を満たす．

Arvind らの原論文では2の状況が起こった場合に，エラーが発生しないよう，新たな配列をコピーで生成し，書き込みを行う実現法が提案されている．確かに関数型言語の中で配列を扱うにはI構造に優れた点が多いが，任意の順序による汎用な要素アクセスを許したために現実的にはコピー操作が必要となった．

われわれは，エラーを起こさず，また，コピー操作を不要とする代償として要素アクセスに『書き込みの実行回数 w (≥ 0) と読み出しの実行回数 r (≥ 0) について常に $w-1 \leq r \leq w$ である』という制限を与える．これにより読み出しが実行される前は $r=w-1$ で，読み出しが実行された後は $r=w$ となることが保証される．すなわち，書き込みが実行されないうちに読み出しは実行されず，読み出しが実行されないうちに次の書き込みが実行されないことを保証している．ここで，仮に書き込みの要求や，読み出しの要求が制限を破るように先行した場合は，制限が満たされるまでそれぞれの要求は保留されるものとする．これにより，I構造の要素ごとに書き込みと読み出しの1対1対応が付き，同期の実現ができる．このようなI構造に制限を与えたものを便宜的に制限的I構造と呼ぶ．

この制限的I構造を用いて統計処理が並列に実行できることを示す．簡単な例として度数分布表を作成する場合を考える．それぞれの統計区間ごとの度数に配列の要素を割り当てて考える．これは度数分布表以外にも，PIC の中で重要な『粒子の情報を格子点で代表させる』ような時にも同様な処理が必要となる．

度数分布表の特性として，配列要素が更新される回数が要素ごとに異なり，あらかじめ予測することも不可能である．このため，コピー操作を必要とするI構造やB構造では読み出しと書き込みを不可分領域として処理するため全サンプルの処理を逐次化して配列要素を更新する必要があった．

ところが，制限的I構造を用いると，全サンプルの処理を並列実行できる．並列処理で問題となるのは，同時にある統計区間の更新が発生する場合に不可分領域を扱うことである．I構造，B構造では逐次化することで回避した．制限的I構造では，ある要素があるプロセッサで更新操作をされている時には，更新が終了して再び書き込まれるまで次の読み出し要求は保留されて，不可分領域が実現されている．これが，制限的I構造の性質として自然に得られる．

3.3 非同期アクセスのM構造

制限的I構造では読み出しの実行回数と書き込みの実行回数に関して制限を設けることにより，構造体上で同期を実現していた．この制限的I構造は排他的制御が必要な変数に対して有効である．しかし，ある条件下で読み出し専用となる変数の場合には制限的I構造の排他的制御で読み出しが逐次化されるためオーバヘッドとなる．また，制限的I構造では連立一次方程式の非同期反復解法[14]で必要な非同期に読み書きを可能とする配列構造を実現できない．

このように制限的I構造では要素アクセスの条件が厳格であるため，多様なアクセスを必要とする科学技術計算の配列構造としては不十分である．ここでは，フォンノイマン型のメモリと同様に非同期なアクセスを許すM構造を提案する．

書き込みと読み出しの関係について，

1．書き込み操作によりデータの存在ビット（Pビット）を立てる，
2．Pビットが立っている要素に書き込み操作を行った場合は新しいデータを書き込む，
3．読み出し操作によりPビットが立っている場合はそのデータを読み出す，
4．読み出し操作の時点でPビットが立っていない場合，読み出し操作は保留される，

という関係を満たす．

M構造では，データを上書きすることを許し，最初の1回を除けば書き込みと読み出しは非同期に行われる．すなわち，データの読み書きの管理はすべてプログラマに開放されている．

QCD 計算でのリンクの更新や，Red-Black SOR 法による連立一次方程式の解法など，配列データの全要素に対して，定数回の読み出しが行われ，これが繰り返される場合がある．全要素に対して，同一回数の読み出しが発生する場合には，このM構造によりコピーの作成や不要領域の回収などのオーバヘッドを避け得る．ただし，その代償として，プログラマが同期

によりデータ値を管理操作する必要がある．このため，文相互，関数相互，文内部での同期を記述する能力が不可欠なものとなる．

3.4 DFCII の記憶クラス

構造体の性質として同期を内包した制限的I構造と，構造体としては非同期性を持ち，ユーザに同期の記述を開放したM構造という2種類の配列構造により，科学技術計算で現れる配列操作を記述することが可能となる．DFCII における制限的I構造とM構造を表現する言語の形式として，Cの記憶クラス指定子を用いる．

スカラ変数（foo）と配列変数（foo[ ]）について分類を行う．スカラ変数における配列構造とは要素が1の配列と考える．もちろんこの場合，制限的I構造は通常のデータ駆動原理による待ち合わせ機構と同値である．

さらに，関数をまたがって参照可能な大域変数と，関数内部からの参照しかできない局所変数に分類する．大域変数は親子関係にない関数間でデータ交換や同期を簡単に実現するために必要である．この大域変数を用いると，物質，物理定数をすべての関数から参照することも容易に記述できる．

記憶クラス指定子を**表1**に示す．**extern** で宣言される変数はすべてその定義が外部にあることを示し，実行時にアドレスが与えられる．**static, auto** で宣言された変数は静的変数でM構造として実現される．**register** で宣言される変数は制限的I構造とする．**vector** で宣言される変数は，高機能構造体（HLS）[13]としてベクトル処理用の変数として，ベクトル演算の対象となる．

制限的I構造とM構造を用いることにより，コピー操作が排除されるため，任意の配列要素のアドレスが決定する．したがって，ポインタを取った操作が可能となる．

表1 記憶クラスの宣言と実体
Table 1 Declaration of storage class and its structure.

宣言	大域変数		局所変数	
	foo	foo[ ]	foo	foo[ ]
extern	実行時	実行時	実行時	実行時
static	M構造	M構造	M構造	M構造
auto	M構造	M構造	M構造	M構造
register	制限的I構造	制限的I構造	変数	×
vector	×	HLS	×	HLS
宣言なし	M構造	M構造	変数	M構造

4. DFCII による並列処理記述

ここでは，C言語から DFCII が拡張した点を述べる．

4.1 強制的な同期の記述

DFC における同期は変数相互の依存関係による記述だけであった．したがって，次の文，

$f(z)$;
$x=y$;

相互で同期をとろうとすると，

$dum=f(z)$;
$x=y+dum-dum$;

として疑似的な変数 dum を用いるのが一つの解決法である．このとき，dum が値を持つので，$(dum-dum)$ のように，同期による副作用をキャンセルする必要が生じる．さらに，強制的に同期を記述したい場合でも，同期用の変数が通常の変数と区別がつかないという欠点がある．

これを解決するために DFCII では，ラベル変数に文の実行終了信号が代入されることを可能とした．すなわち，ある文の実行が終了すると，そのラベルの値が確定するという実行制御を，データフローモデルで解釈する．

ラベルを用いて文相互，文と変数相互の同期をとるには **sync** 文を用いる．sync 文は

sync（ラベルまたは変数並び）文；

という形式をとる．その意味は『ラベルまたは変数並び』に記述された識別子（同期変数）の値がすべて確定することと，『文』で参照する変数（被同期変数）の値の確定とを同期させるものである．すなわち，同期変数がすべて確定しない限り『文』の実行は開始されない．

この sync 文を用いることで先の例は，

$L1$: $f(z)$;
sync($L1$) $x=y$;

と記述できる．実行の意味は，$f(z)$ の関数が終了してから $x=y$ を実行する，すなわち，x の値が確定する．

ラベル文は複文に対しても付加できるので，構造化した同期を記述することができる．また，ラベルの変数としてのスコープは関数内部に局所的である．

これにより，Cでは値を返していない文に関する実行終了をラベルという疑似変数を明示的に用いることでデータ駆動制御による文の並列制御，同期実行が可

能となった.

4.2 並列性を制御するループ構造文

データフローモデルには動的モデルと静的モデルがある. SIGMA-1 では動的モデルに基づくアーキテクチャを採用しているため, ループのような繰り返し構造の場合, ループカウンタというタグで計算の論理的空間をコピーして可能な限り並列性を獲得して計算を行う. しかし, この方法は並列性の爆発という危険を秘めている. すなわち, 計算資源としてのループカウンタが枯渇する可能性がある.

このため, ループ構造文に対しては, 並列性の制御を行う. ここでは, ループの本体は同時に一つしか実行しない, という静的なデータフローモデルを実現することで最も確実に計算の爆発を制御する.

静的なデータフローモデルを記述するために syncfor 文と syncdo-while 文がある.

syncfor 文は次の形をしている.

syncfor (式 1_{opt}; 式 2_{opt}; 式 3_{opt}) 文;

ループを繰り返す終了条件判定の前で, 次のループ実行に必要な変数の同期がとられる.

do-while 文に対応する静的なループ記述として, syncdo-while 文がある. 形式は以下のようである.

syncdo 文 **while** (式);

このようなループ構造により, M構造を持つ配列へのアクセス順を制御することができる.

4.3 DFCII と SIGMA-1 のデータ型

SIGMA-1 には, データ型として整数である **sint** 型, 浮動小数点数である **sflt** 型, 文字を表現する **char** 型, 論理値を表す **bool** 型, メモリアドレスを表す **addr** 型, 配列の領域を管理する **desc** 型, 関数のエントリポイント, すなわち命令メモリに格納された関数の先頭アドレスを示す **ides** 型がある.

DFCII では SIGMA-1 で実行可能な型をできる限り言語として表現することを設計に採り入れた. これにより, **表 2** の表現を得る.

表 2 DFCII での表現と SIGMA-1 のデータ型
Table 2 Declaration of data types.

DFCII での表現	SIGMA-1 のデータ型
int 識別子	**sint**
float 識別子	**sflt**
char 識別子	**char**
bool 識別子	**bool**
型 *識別子	**addr**
型 識別子 [ ]	**desc**
型 識別子 ( )	**ides**
complex 識別子	(**sflt**, **sflt**)

また, 科学技術計算で必須な複素数を扱うデータ型として複素数型を定義する. SIGMA-1 のアーキテクチャにはこれに対応するデータ型がないが, コンパイラが内部処理を行って, 実部と虚部を **float** として扱う.

4.4 複数戻り値の関数

C言語では, 関数からの戻り値を複数としたい場合, 一般に副作用を用いて配列や構造体へのポインタとして値を返すことがある. しかし, データ駆動計算機ではポインタよりもデータ値そのものが返ってきた方が効率よく実行できる. また, ポインタでなく値として戻り値がある場合の方が副作用を局所化できる. そこで, 単一の戻り値を許す関数と複数の戻り値を許す関数を設けた.

また, 文法上他の文や式との混乱を招かないように [ ] を用いることとした.

後者の呼び出し形式は

[ret1, ..., ret$n$] = f(arg1, ..., arg2);

しか許されない. 関数本体では一箇所に限られた return 文により, **return** (ret1, ret2); と記述する.

複数戻り値の関数を次のように宣言する.

[**int**, **char***, **float**] f( );

これは第1関数値として int, 第2関数値として char へのポインタ, 第3関数値として float を返す関数を宣言している. リターン値の受け取り方は,

[識別子$_1$, ..., 識別子$_n$] = f( );

であり, 任意の識別子を省略することができる.

関数定義側では次の形で複数の値を関数の呼び出し元へ返す.

return 式並び;

これは, 関数の一番最後に一つだけ書かなくてはならないとした.

5. ま と め

逐次解釈とデータ駆動原理に基づく言語として DFCII の設計を行った. この DFCII は, SIGMA-1 を対象としながらC言語との互換性を保った汎用的な並列処理記述言語を目指した. DFCII はC言語から goto 文を排除し, 同期記述の文, 複素数型などのデータ型, 複数戻り値の関数を許したなどの点で拡張されている. また既に開発を行った DFC とは単一代入則を排除し, 逐次解釈に基づくデータ駆動実行をセマンティクスとした点で異なっている.

科学技術計算で必要な配列の扱いに関して従来のI構造やB構造に対して，制限的I構造とM構造を記憶クラスの違いで記述した．制限的I構造によりこれまで記述できなかった並列処理による度数分布表の作成や，M構造によりユーザ制御の下で配列への非同期，同期アクセスを可能とした点で大幅な記述力の向上を達成した．

また，ラベルによる文の同期，syncfor，syncdo-while によるループ文内部の同期，複素数型を含めた各種のデータ型，さらに複数戻り値を許す関数でC言語にない表現力を拡張した．

今後の課題としてデータ駆動計算機以外の並列処理計算機への適応性に関してさらに考察を深めていきたい．

謝辞 本研究は通産省大型プロジェクト「科学技術用高速計算システムの研究」の一環である．日頃より御指導いただく棟上昭男電子技術総合研究所情報アーキテクチャ部長，田村浩一郎情報科学部長，弓場敏嗣知能システム部長および計算機方式研究室同僚諸氏，また実現に御協力をいただいた(株)富士通ソーシャルサイエンスラボラトリの浅野利則氏に感謝いたします．

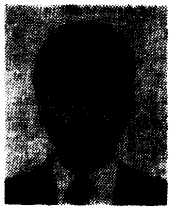
参考文献

1) Kernighan, B. W. and Ritchie, D. M.: *The C Programming Language*, Prentice Hall (1978).
2) 島田，関口，平木：データフロー言語 DFC の設計と実現，電子情報通信学会論文誌，Vol. J 71-D, pp. 501-508 (1987).
3) Ackerman, W. B. and Dennis, J. B.: VAL—A Value Oriented Algorithmic Language: Preliminary Reference Manual, TR 218, LCS, MIT (1979).
4) Arvind, Gostelow, K. P. and Ploufe, W.: An Asynchronous Programming Language and Computing Machine, TR 114 a, Dept. Comp. Sci., Univ. Calif., Irvine (1978).
5) McGraw, J.: SISAL: Streams and Iteration in a Single Assignment Language, Language Reference Manual, LLNL (1985).
6) 長谷川，雨宮：データフローマシン用関数型高級言語 Valid，電子情報通信学会論文誌，Vol. J 71-D, pp. 1532-1539 (1988).
7) Arvind and Thomas, R. E.: I-structures: An efficient Data Type for Functional Languages, TM 128, LCS, MIT (1979).
8) Wilson, K.: New Phenomena in Subnuclear Physics, Zichini, A. (ed.), Plenum Press (1977).
9) Kruer, W. L. et al.: The Dipole Expansion Methods for Plasma Simulation, *J Comput. Phys.*, Vol. 13, pp. 114-129 (1973).
10) Petrie, L. M. et al.: KENO VI an Improved Monte Carlo Criticality Program, ORNL-4938 W-7405-eng-26 (1975).
11) Dongarra, J. J. et al.: LINPACK User's Guide, SIAM (1979).
12) Hiraki, K. et al.: The SIGMA-1 Dataflow Supercomputer: A Challenge for New Generation Supercomputing Systems, *J. Inf. Process.*, Vol. 10, No. 4, pp. 219-226 (1987).
13) Hiraki, K. et al.: Efficient Vector Processing on a Dataflow Supercomputer, *Proc. SUPERCOMPUTING '88*, pp. 374-381 (1987).
14) 関口，小柳：科学技術計算における並列化技術，情報処理，Vol. 27, No. 9, pp. 985-994 (1986).
15) 関口，平木，島田：抽象命令セットを用いたデータ駆動計算機用中間言語 SASGA，電子情報通信学会技術研究報告，CPSY 89-36, pp. 31-36 (1989).

(平成元年6月20日受付)
(平成元年9月12日採録)

関口 智嗣（正会員）

1959年生．1982年東京大学理学部情報科学科卒業．1984年筑波大学大学院修士課程理工学研究科修了．同年電子技術総合研究所入所．計算機アーキテクチャと数値解析の研究に従事．特に科学技術計算用並列アルゴリズムに興味を持つ．市村賞受賞．現在，通産省大型プロジェクト「科学技術用高速計算システムの研究」の一環としてデータフロースーパーコンピュータの開発を進めている．



島田 俊夫（正会員）

昭和20年生．昭和43年東京大学工学部計数工学科卒業．昭和45年東京大学大学院修士課程修了．同年電気試験所(現電子技術総合研究所)入所．現在，同所計算機方式研究室室長．コンピュータグラフィックス，人工知能向き言語，LISP マシン，データフロー計算機の研究に従事．高度な並列処理方式に興味がある．電子情報通信学会，IEEE 各会員．

平木 敬（正会員）

昭和51年東京大学理学部物理学科卒業．昭和57年同大学院理学系研究科博士課程修了．同年電子技術総合研究所入所．理学博士．計算機アーキテクチャ全般，特にリスト処理計算機，データフローマシン，スケジューリングなどの研究に従事．元岡賞，市村賞各受賞．情報アーキテクチャ部計算機方式研究室主任研究官．現在 IBM ワトソン研究センター招聘研究員．