世界知的所有権機関
国際事務局
PCT
特許協力条約に基づいて公開された国際出願

(51) 国際特許分類6
G06F 15/16

A1

(11) 国際公開番号 WO95/26530

(43) 国際公開日 1995年10月5日(05.10.95)

(21) 国際出願番号 PCT/JP95/00552
(22) 国際出願日 1995年3月24日(24.03.95)

(30) 優先権データ
特願平6/57141 1994年3月28日(28.03.94) JP
特願平6/57142 1994年3月28日(28.03.94) JP

(71) 出願人（米国を除くすべての指定国について）
ソニー株式会社(SONY CORPORATION)[JP/JP]
〒141 東京都品川区北品川6丁目7番35号 Tokyo, (JP)
(72) 発明者；および
(75) 発明者／出願人（米国についてのみ）
勝尾 聡(KATSUO, Satoshi)[JP/JP]
志潟太郎(SHIGATA, Taro)[JP/JP]
〒141 東京都品川区北品川6丁目7番35号
ソニー株式会社内 Tokyo, (JP)
(74) 代理人
弁理士 小池 晃, 外(KOIKE, Akira et al.)
〒105 東京都港区虎ノ門二丁目6番4号
第11森ビル Tokyo, (JP)

(81) 指定国
JP, US, 欧州特許(DE, FR, GB).

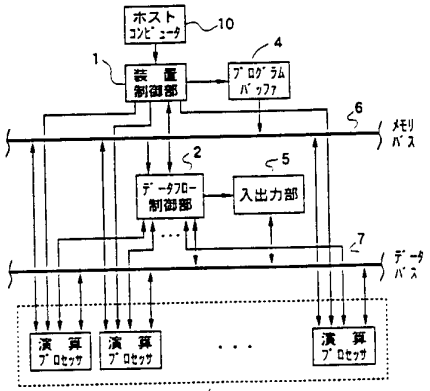
添付公開書類 国際調査報告書

(54) Title : METHOD OF COMPILING PARALLEL PROCESSING PROGRAM, IMAGE PROCESSOR AND IMAGE PROCESSING METHOD

(54) 発明の名称 並列処理プログラムのコンパイル方法、画像処理装置及び画像処理方法

(57) Abstract

A method of compiling a parallel processing program while avoiding the increase in size of an executable code without imparting a large burden to a program developer. In a step of analysis a new program is analyzed, while in a step of code generation are generated respective executable codes for arithmetic processor programs and other programs as the result of analysis by the analysis step. This method may be used to compile programs written in a programming language to allow a plurality of arithmetic processors for different parts of a process to perform parallel image processing.



1 ... apparatus control portion
2 ... data flow control portion
3 ... arithmetic processor
4 ... program buffer
5 ... input/output portion
6 ... memory bus
7 ... data bus
10 ... host computer

(57) 要約

プログラム開発者に大きな負担を強いることなく、生成される実行コードの増加を抑える並列処理プログラムのコンパイル方法の提供を目的として、解析工程では、作成したプログラムを解析し、実行コード生成工程では、解析工程による解析結果として演算プロセッサのプログラムとそれ以外のプログラムとを識別して実行コードを生成することにより、複数の演算プロセッサに対して画像処理を並列に行わせるプログラム言語を用いて作成したプログラムの実行コードの生成において、各々の役割を担当するプロセッサの実行コードを生成する並列処理プログラムのコンパイル方法である。

情報としての用途のみ
PCTに基づいて公開される国際出願をパンフレット第一頁にPCT加盟国を同定するために使用されるコード

AM アルメニア
AT オーストリア
AU オーストラリア
BB バルバドス
BE ベルギー
BF ブルギナ・ファソ
BG ブルガリア
BJ ベナン
BR ブラジル
BY ベラルーシ
CA カナダ
CF 中央アフリカ共和国
CG コンゴー
CH スイス
CI コート・ジボアール
CM カメルーン
CN 中国
CZ チェッコ共和国
DE ドイツ
DK デンマーク
EE エストニア
ES スペイン
FI フィンランド
FR フランス
GA ガボン
GB イギリス
GE グルジア
GN ギニア
GR ギリシャ
HU ハンガリー
IE アイルランド
IL イスラエル
IS アイスランド
IT イタリー
JP 日本
KE ケニア
KG キルギスタン
KP 朝鮮民主主義人民共和国
KR 大韓民国
KZ カザフスタン
LI リヒテンシュタイン
LK スリランカ
LR リベリア
LS レソト
LT リトアニア
LU ルクセンブルグ
LV ラトヴィア
MC モナコ
MD モルドバ
MG マダガスカル
ML マリ
MN モンゴル
MR モーリタニア
MW マラウイ
MX メキシコ
NE ニジェール
NL オランダ
NO ノルウェー
NZ ニュージーランド
PL ポーランド
PT ポルトガル
RO ルーマニア
RU ロシア連邦
SD スーダン
SE スウェーデン
SG シンガポール
SI スロヴェニア
SK スロヴァキア共和国
SN セネガル
SU ソビエト連邦
TD チャード
TG トーゴ
TJ タジキスタン
TM トルクメニスタン
TR トルコ
TT トリニダード・トバゴ
UA ウクライナ
UG ウガンダ
US 米国
UZ ウズベキスタン共和国
VN ヴィエトナム

明 細 書

並列処理プログラムのコンパイル方法、画像処理装置及び画像処理方法

技 術 分 野

本発明は、複数の演算プロセッサに対して画像処理を並列に行わせるプログラム言語を用いて作成されたプログラムの実行コードを生成する並列処理プログラムのコンパイル方法と、複数の演算の並列処理によって画像処理を行う画像処理装置と、この画像処理装置を動作させる画像処理方法に関する。

背 景 技 術

一般に、並列処理システムにおいて、ユーザ定義のプロシージャ（処理）を複数のプロセッサに分散して並列処理を行わせる際に、プログラム開発者は、各プロセッサ毎にこのプロシージャを行わせるようにプログラムの記述作成を行っている。

このユーザ定義のプロシージャを並列処理システムにおける負荷とすると、プログラム開発者は、予め上述したような並列処理システムにおいて各プロセッサに対する負荷の分散が適切に行われるようにソースプログラムを作成している。

従って、プログラム開発者は、処理の効率化を図るために意識的に個々のプロセッサへの負荷が分散されるようにソースプログラム

を記述しなければならない。

ところが、このプログラムの作成において、プログラム開発者には、負荷分散するプロセッサの数分だけ個々の負荷分散を意識させた作業が強いられることになり、大きな負担になってしまう。

このような並列処理システムを用いた画像処理において、画面分割処理を行った場合に、分割により得られる小領域の数が処理するプロセッサ数より多く、かつこのプロセッサ数で小領域の数が割り切れず画面に余りの領域、すなわち端数領域を生じることがある。このような端数領域が生じた際には、この端数領域に対する端数処理をいずれかのプロセッサに行わせなければならないことになる。

また、負荷分散した場合において、画面を分割した小領域での画像処理の結果を統合して演算する処理が一部のプロセッサに担わされると、この一部のプロセッサには、小領域に対する演算処理用の実行コードの他に、上述した統合して演算する統合処理用の実行コードの２種類が必要になる。このようなプログラムの作成は、プログラム開発者にこの統合処理をも意識したプログラム作成を行わせることになるから、上述した以上に負担が重くなる。

また、プロセッサに対して２種類の実行コードを供給することにより、このプロセッサに対する実行コード量が、小領域に対する演算処理用の実行コード量に比べて増加してしまい、また、この統合処理を追加することにより、使用するメモリをほぼ倍程度費やしてしまうことになる。

そこで、本発明は、上述したような実情に鑑みてなされたものであり、プログラム開発者に大きな負担を強いることなく、生成される実行コードの増加を抑えることができる並列処理プログラムのコ

ンパイル方法の提供を目的とする。

また、本発明は、例えば使用するアプリケーションプログラム等に応じて実行コードを生成して従来以上に効率のよい画像処理を行うことのできる画像処理装置及び画像処理方法の提供を目的としている。

発明の開示

上述した課題を解決するために、本発明に係る並列処理プログラムのコンパイル方法は、複数の演算プロセッサに対して画像処理を並列に行わせるプログラム言語を用いて作成されたプログラムをコンパイルして実行コードを生成する並列処理プログラムのコンパイル方法であって、上記作成されたプログラムから画像処理における並列処理を示す識別子が付された制御文の識別と解析を行う解析工程と、上記解析工程による解析結果と上記複数の演算プロセッサの数に応じた各演算プロセッサへの並列処理の実行コードを生成する実行コード生成工程とを有する。

また、本発明に係る画像処理方法は、複数の演算プロセッサに対して画像処理を並列に行わせるプログラム言語を用いて作成されたプログラムの実行コードに従って画像処理を行う画像処理方法において、上記作成されたプログラムから画像処理における並列処理を示す識別子が付された制御文の識別と解析を行う解析工程と、上記解析工程による解析結果と上記複数の演算プロセッサの数に応じた各演算プロセッサへの並列処理の実行コードを生成する実行コード生成工程と、上記生成された実行コードに基づいて、上記各演算プ

ロセッサがそれぞれ画像処理を行う画像処理工程とを有する。

また、本発明に係る画像処理装置は、複数の演算の並列処理によって画像処理を行う画像処理装置であって、入力された画像信号の画面分割された各小領域に対する画像処理の演算をそれぞれ並列に行う複数の演算手段と、該演算手段からの各演算結果を基に統合的な演算を行うデータ供給制御手段とを有する。

図面の簡単な説明

図１は、本発明に係る画像処理装置の構成を示す概略的なブロック図である。

図２は、上記画像処理装置の動作を説明する概略的なメインフローチャートである。

図３は、上記メインフローチャートにおけるサブルーチン１の処理手順を説明するフローチャートである。

図４は、上記画像処理装置の画面構成と小分割領域の関係及び小分割領域の動き方を説明するための模式図である。

図５は、上記メインフローチャートにおけるサブルーチン２の処理手順を説明するフローチャートである。

図６は、上記サブルーチン２で画像並列処理フラグを立てる場合の画像の一例を示す模式図である。

図７は、上記メインフローチャートにおけるサブルーチン３の処理手順を説明するフローチャートである。

発明を実施するための最良の形態

以下、本発明に係る並列処理プログラムのコンパイル方法、画像処理装置及び画像処理方法の一実施例について、図面を参照しながら説明する。

本発明の並列処理プログラムのコンパイル方法は、作成されたプログラムを解析工程で解析し、実行コード生成工程で解析工程による解析結果に基づいて演算プロセッサのプログラムとそれ以外のプログラムとを識別して実行コードを生成することにより、各々の役割を担当するプロセッサの実行コードを生成する方法である。

この実施例では、この方法を適用した画像処理装置の一例を説明する。

画像処理装置は、例えば図１に示すように、全体を制御する装置制御部１と、データの流れを制御するデータフロー制御部２と、複数の演算プロセッサを有する演算部３と、プログラムバッファ４と、入出力部５とを備える。

装置制御部１は、プログラムバッファ４を制御してプログラムバッファ４上のデータフロー制御部２及び演算部３に対応するプログラムをメモリバス６を介してデータフロー制御部２のプロセッサ及び演算部３の各演算プロセッサ内のプログラムメモリにロードさせる。また、装置制御部１は、データフロー制御部２及び演算部３の各演算プロセッサの実行制御、クロックの供給を行っている。

データフロー制御部２は、演算部３からの各演算結果を基に統合的な演算を行い、また、必要に応じてこの演算結果を演算部３に分配供給している。具体的に説明すると、データフロー制御部２は、入出力部５内の共有メモリ（図示せず）に制御信号を供給する。こ

れにより、データフロー制御部２は、共有メモリからデータバス７を介して各演算プロセッサ内の分散メモリ（図示せず）にデータを分配する制御を行う。また、データフロー制御部２は、演算部３の各演算プロセッサと制御信号の授受を行い、供給したデータに対する演算処理が終了した演算プロセッサからこの処理結果を回収する。そして、データフロー制御部２は、データバス７を介して処理結果のデータを入出力部５内の共有メモリに供給する制御を行う。データフロー制御部２は、演算処理が終了した演算プロセッサに対して共有メモリからデータバス７を介して次のデータを分配する制御を行う。このようにデータフロー制御部２は、データの分配と回収を繰り返し、このデータフロー制御部２のプログラムが終了するまで制御を行う。

このように、画像処理装置は、供給される画像信号に並列処理が施されることにより、画像処理を高速化させている。また、この画像処理装置は、画像データを複数に分割して得られる複数の小領域に対して、それぞれの演算プロセッサが同じプログラムを実行するようになっている。すなわち、画像処理装置は、この同じプログラムを実行する福数の演算プロセッサを有し、１画面を小領域に分割して並列処理を行うアーキテクチャにより、画像処理を高速に並列処理を行っている。

ところで、画像処理には、このアーキテクチャを満足するモデルに対応するように分けられた小領域内では閉じていないような画像処理がある。このような画像処理としては、例えば全画面の濃度平均値を求める等の場合がある。

そして、画像処理装置において、画像処理における負荷を複数の

演算プロセッサに分散させて、この負荷分散の各画像処理の結果をまとめる統合処理を一部の演算プロセッサに行わせると、演算部３には、分割領域処理用の演算プロセッサの実行コードと、統合処理用の演算プロセッサの実行コードとの２種類の実行コードが必要になる。このように生成された実行コードは、上述したアーキテクチャを満足するモデルとならず、かつ生成される実行コード量が増えることになる。

そこで、本発明を適用した画像処理装置は、このアーキテクチャを満足するモデルに基づき、共通のプログラムを演算部３の各演算プロセッサに実行させ、データフロー制御部２でそれぞれの演算結果を吸い上げて統合処理を行い、さらに、この統合処理結果を演算部３の各演算プロセッサに供給して演算させる等の処理を行っている。

またところで、処理する画面数が多い場合には、１画面ずつを各演算プロセッサに割り当てるという方法もある。処理する画面数が、演算プロセッサの数で割り切れる場合には、並列処理中に遊ぶ演算プロセッサもなく、並列処理度を最大に高めることができる。また、前述の画面分割処理では必要であった、データフロー制御部２と演算部３の演算プロセッサ間の通信も行う必要がなくなる。このため、処理する画面数が、演算プロセッサの数で割り切れる場合には、１画面ずつ各演算プロセッサに割り当てる画面並列処理の処理速度の方が速いことになる。

しかしながら、この画像処理装置の構成と画像信号の量的な関係においては、入力される画像信号の画面数を演算部３の演算プロセッサの数で割ったとき、上記画面数が上記演算プロセッサの数で割

り切れず、端数が生じる場合がある。

このような関係になると、画像処理装置では、極端に並列性が低くなり、画像処理の効率が著しく低下してしまう場合がある。

具体的な一例として、画像処理装置が、例えば演算部３に１６個の演算プロセッサを有している構成において１７個のフレームの画像を処理する場合を考える。演算部３は、１７番目のフレームを処理する際には、１つの演算プロセッサのみが動作して、１５個の演算プロセッサが動作しておらず、これらの演算プロセッサが遊ぶことになり、この結果、画像処理の効率を低下させてしまう。

そこで、この画像処理装置は、上述のアーキテクチャを満足するモデルに適合した１画面を小領域に分割してこれらの小領域毎に演算プロセッサを動作させる場合と、このアーキテクチャを満足するモデルに適合しない、１つの演算プロセッサに１画面ずつ割り当てて動作させる場合とを有する。

このような場合分けに対する処理は、例えば図１に示すように、コンパイルを行うホストコンピュータ１０で行う。また、ホストコンピュータ１０は、上述のアーキテクチャのモデルに基づいた統合処理を含めた画面分割処理を行う場合のコンパイルと、演算プロセッサに１画面毎の処理を割り当てるコンパイルとの場合分けも行うことにより、従来以上の処理効率を実現させている。

次に、この画像処理装置に対して並列処理による高速な画像処理を行わせるための画像処理方法について図２、図３及び図５のフローチャートと、画像処理装置の処理する画素及びブロック構成について図４を参照しながら説明する。上述したように、この画像処理装置にコンパイルして供給される並列処理プログラムの実行コード

は、この画像処理装置の外部に設けたホストコンピュータ１０で生成される。

ホストコンピュータ１０は、例えば図２に示すように、並列処理プログラムのコンパイルを開始してサブルーチンＳＵＢ１に進む。

このサブルーチンＳＵＢ１では、ホストコンピュータ１０は、並列処理プログラムの手順が記述された画像処理のソースプログラムにおける並列処理を示す識別子が付された制御文の識別と解析を行い、サブルーチンＳＵＢ２に進む。

次に、サブルーチンＳＵＢ２では、ホストコンピュータ１０は、画像処理される画面数と演算部３の複数の演算プロセッサの数との関係と、サブルーチンＳＵＢ１の解析結果とから、画像処理を画面分割処理で行うか画面並列処理で行うかの処理判別を行い、サブルーチンＳＵＢ３に進む。

サブルーチンＳＵＢ３で、ホストコンピュータ１０は、サブルーチンＳＵＢ２の結果を基に実行コードの生成を行う。

ホストコンピュータ１０は、これらの３つのサブルーチンＳＵＢ１～ＳＵＢ３の順に行うことにより、並列処理プログラムのコンパイルを終了し、ステップＳ１に進む。

ステップＳ１では、ホストコンピュータ１０は、サブルーチンＳＵＢ１～ＳＵＢ３においてコンパイルされた実行コードを画像処理装置の装置制御部１に供給する。そして、装置制御部１は、ホストコンピュータ１０からの実行コードをプログラムバッファ４に一旦記憶した後、プログラムバッファ４上のデータフロー制御部２及び演算部３に対応するプログラムの実行コードを、メモリバス６を介してデータフロー制御部２のプロセッサ及び演算部３の各演算プロ

セッサ内のプログラムメモリにロードさせる制御を行う。

ステップＳ２では、演算部３の各演算プロセッサは、入出力部５を介して供給される２次元データを、実行コードに応じて１画面を分割した小領域あるいは１フレーム単位に処理し、データフロー制御部２の演算プロセッサは、実行コードに応じてデータの供給／回収や演算部３の演算結果の統合処理を行う。このような一連の演算処理により、画像処理が高速に実行される。

さらに、上述したサブルーチンＳＵＢ１～ＳＵＢ３の手順について説明する。

ここで、この並列処理プログラムのソースプログラムには、高級言語の一つである、いわゆるＣ言語に準拠した言語が用いられている。

サブルーチンＳＵＢ１では、作成されたソースプログラムにより、１つの演算処理である、「関数」内に、２次元データを分割して複数の小領域で行われた演算結果を用いて、統合的な演算を行うといった統合的な演算処理があるかないか、また、この統合的な演算結果を各演算プロセッサ等に分配するといった分配処理があるかないかの解析を行う。

具体的には、図３に示すフローチャートのステップＳ１０において、ホストコンピュータ１０は、画像処理装置用の例えばコンフィグレーション・ファイル（CONFIG.FILE:以下、コンフィグ・ファイルという）の解析を行う。このコンフィグ・ファイルには、画像処理装置における例えば演算部３に組み込まれている演算プロセッサの数等の情報がヘッダ情報として記載されている。そして、この情報を用いて適用する画像処理装置が有する演算プロセッサの総数が

調べられる。

次に、ステップＳ１１においては、ホストコンピュータ１０は、並列処理プログラムのソースプログラムの解析を行う。

並列処理プログラムのソースコードを記述するためのこの言語では、例えば繰り返し処理を示すＣ言語の for文の前に画像処理における並列処理を行わせるためのきっかけとなるキーワードである識別子であって、かつそれぞれ垂直方向の繰り返しを意味する“v”と水平方向の繰り返しを意味する“h”とを付加して新たな並列処理用のvfor、hforが用いられている。このvfor文とhfor文の括弧内のパラメータは、Ｃ言語のfor 文と同じである。

同様に、画面を分割した小領域に関する新たな並列処理用の変数Vinit 及びHinit は、それぞれ上記小領域の左上位置における垂直方向と水平方向の座標を示す変数である。

そして、ソースプログラムの一例として例えば

```
.....
 vfor(Vinit=0;Vinit<512;Vinit+=8){          ・・・ (1-1-1)
  hfor(Hinit=0;Hinit<768;Hinit+=8){         ・・・ (1-1-2)
      read(x, 8, 8);                        ・・・ (1-1-3)
      ..........
      tmp=x[0]+·····                        ・・・ (1-1-4)
      ..........
      write(y, 8, 8);                       ・・・ (1-1-5)
      ..........
      total=total+tmp;                      ・・・ (1-1-6)
  }                                         ・・・ (1-1-7)
```

```
  }                                                    ・・・ (1-1-8)
 ..........
  vfor(Vinit=0;Vinit<512;Vinit+=8){                    ・・・ (1-2-1)
   hfor(Hinit=0;Hinit<768;Hinit+=8){                   ・・・ (1-2-2)
       read(x, 8, 8);                                  ・・・ (1-2-3)
       ...........
       tmp=total+·····                                 ・・・ (1-2-4)
       ...........
       write(y, 8, 8);                                 ・・・ (1-2-5)
       ...........
    }                                                  ・・・ (1-2-6)
  }                                                    ・・・ (1-2-7)
```

という書式例を検討する。

このソースプログラムのvfor文とhfor文の括弧内のパラメータ、すなわちVinit=0;Vinit<512とHinit=0;Hinit<768は、図４（ａ）に示すように、画面が、512×768画素で構成されていることを表している。Vinit+=8とHinit+=8は、画面を分割した小領域が、8×8画素のサイズのブロックから構成されていることを表している。また、Vinit=0とHinit=0は、図４（ｂ）に示すように、この小領域の読出し処理や書込み処理が、先ずhfor文によって画面の左上から右端に向かって行われた後にvfor文によって次のラインに相当する左端位置の小領域の読出し処理や書込み処理が行われることを表している。

この小領域の読出しや書込み処理には、read文、write文が用いられる。read文及びwrite文の第１の引数xは、小領域を格納するためのバッファ（配列）名であり、第２の引数は、水平方向のサイ

ズ指定であり、第３の引数は、垂直方向のサイズ指定である。

文（1-1-1）、（1-2-1）のvfor文、文（1-1-2）、（1-2-2）のhfor文、文（1-1-3）、（1-2-3）のread文、文（1-1-5）、（1-2-5）のwrite文に着目すると、ステップＳ１１では、ホストコンピュータ１０は、ソースプログラムの解析により、データフロー制御部２の実行コード生成に必要な小領域のパラメータを知ることができる。すなわち、このソースプログラムの解析により、ホストコンピュータ１０は、１つの小領域の水平・垂直方向のサイズ、最初の位置、動かし方を知ることができる。すなわち、画面全体の処理すべき小領域の総数は、vfor文及びhfor文のパラメータから判る。

次に、ステップＳ１２では、ホストコンピュータ１０は、ステップＳ１１における解析結果を基にデータフロー制御部２の情報テーブルを作成してステップＳ１３に進む。

ステップＳ１３では、ホストコンピュータ１０は、このソースプログラムの中での演算プロセッサの処理に関する情報テーブルを作成する。この演算プロセッサの処理に関する情報テーブルとは、各小領域に対する処理について、例えばソースプログラムにおける文（1-2-1）のvfor や、文（1-2-1）のhfor等の構文のキーワードで囲まれたソースプログラムを解析して得られた情報がまとめれらたものである。

次に、ステップＳ１４では、ホストコンピュータ１０は、最初の演算ブロックの情報を取り出す。すなわち、ホストコンピュータ１０は、例えば文（1-1-3）のread文と文（1-1-5）のwrite文との間の文（1-1-4）や、文（1-2-3）のread文と文（1-2-5）のwrite文との間の文（1-2-4）等の演算プロセッサを小領域に対してどのような

演算処理を行わせるかの情報を読み込む。ここで、演算ブロックとは、vfor、 hforで囲まれ、演算プロセッサが行う演算内容を記述したものである。

次に、ステップＳ１５では、ホストコンピュータ１０は、統合処理を行う計算式がステップＳ１４で読み込んだ演算ブロックの中に含まれているかを調べる。小領域の画像処理に対して統合処理を行う例えば文（1-1-6）のような計算式がない場合（Ｎｏ）、ステップＳ１７に進む。一方、例えば 文（1-1-6）のような計算式がある場合（Ｙｅｓ）、ステップＳ１６に進む。

ステップＳ１６では、ホストコンピュータ１０は、この計算式に関する演算をデータフロー制御部２内の演算プロセッサに行わせるように、例えば統合フラグを立てて（“１”をセットして）ステップＳ１７に進む。

なお、ステップＳ１５では、ホストコンピュータ１０は、統合的な演算の結果の分配の有無も調べることができる。すなわち、ホストコンピュータ１０は、例えば 文（1-2-4）のような計算式がない場合（Ｎｏ）、ステップＳ１７に進む。一方、ホストコンピュータ１０は、文（1-2-4）のような計算式がある場合（Ｙｅｓ）、ステップＳ１６に進み、ステップＳ１６で分配フラグを立ててステップＳ１７に進むようにしてもよい。

ステップＳ１７では、ホストコンピュータ１０は、各演算プロセッサを分割した画面の小領域に対応させる際に端数ができるかどうかを調べる。すなわちこの端数とは、画面を小領域に分割した際の小領域の総数が演算プロセッサ数の総数で割り切れなかった剰余分を示す。当然、小領域の端数の数は、演算プロセッサ数より少ない

ことを意味する。

そして、ホストコンピュータ１０は、この端数が存在するとき（Ｙｅｓ）、ステップＳ１８に進み、端数が存在しないとき（Ｎｏ）、ステップＳ１９に進む。

ステップＳ１８では、ホストコンピュータ１０は、端数フラグをセットしてステップＳ１９に進む。

ステップＳ１９では、ホストコンピュータ１０は、全演算ブロックに対する解析処理が終了したかどうかの判断を行う。全演算ブロックに対する解析処理が終了していないとき（Ｎｏ）、ステップＳ１９Ｎに進む。

ステップＳ１９Ｎでは、次の演算ブロックの情報を読み込んで解析を続行する。すなわち、ホストコンピュータ１０は、次の演算ブロックの情報を読み込み後、ステップＳ１５に進む。一方、ホストコンピュータ１０は、ステップＳ１９において、全演算ブロックに対する解析処理が終了しているとき（Ｙｅｓ）、リターンに進んでこのサブルーチンＳＵＢ１を終了する。

ここで、以下に示す例えば４枚の画像に対して統合処理を含む処理プログラムに関する並列処理について説明する。この処理プログラムは、例えば、

```
.........
 vfor(Vinit=0;Vinit<512*2;Vinit+=512){        ・・・(2-1)
  hfor(Hinit=0;Hinit<768*2;Hinit+=768){       ・・・(2-2)
   vfor(Vinit=0;Vinit<512;Vinit+=8){          ・・・(2-3)
    hfor(Hinit=0;Hinit<768;Hinit+=8){         ・・・(2-4)
      read(x, 8, 8);                          ・・・(2-5)
```

```
        ..........
        tmp=x[0]+·····                          ・・・(2-6)
        ..........
        write(y, 8, 8);                         ・・・(2-7)
        ..........
        total=total+tmp;                        ・・・(2-8)
         }                                      ・・・(2-9)
       }                                        ・・・(2-10)
     }                                          ・・・(2-11)
  }                                             ・・・(2-12)
 ..........
```

という書式例である。

この処理プログラムでは、演算部３に例えば４個の演算プロセッサがあるとき、例えば図６に示すように、512×768画素を１枚の画面とする４枚の画面に対して画像処理する場合を示している。このような全画面を単位とする演算では、例えば全画面の濃度平均等を求める画像処理を行うことがある。この画像処理を前述したアーキテクチャに応じたモデルに基づいて行うために、サブルーチンＳＵＢ２では、ホストコンピュータ１０は、サブルーチンＳＵＢ１で解析した統合処理を含む画面分割処理と、１フレーム単位毎に対応する演算プロセッサで画像の演算を並列に行う画面並列処理との判別を行う。

例えば図５に示すように、サブルーチンＳＵＢ２のフローチャートのステップＳ２０において、ホストコンピュータ１０は、最初の演算ブロックの情報を取り出してステップＳ２１に進む。

ステップＳ２１では、ホストコンピュータ１０は、統合処理フラグの有無により、統合処理があるかどうかの判別を行う。ホストコンピュータ１０は、例えば 文（2-8）で示すような統合処理があるとき、統合処理フラグが立っているはずなので（Ｙｅｓ）、ステップＳ２２に進み、統合処理がないとき（Ｎｏ）、ステップＳ２５に進む。

ステップＳ２２で、ホストコンピュータ１０は、処理画面数が１枚かどうかの判別を行う。ホストコンピュータ１０は、処理画面数が１枚のとき（Ｙｅｓ）、ステップＳ２５に進み、処理画面数が複数の枚数のとき（Ｎｏ）、すなわち例えば文（2-2）～文（2-5）に示すようなプログラムが書かれているとき、ステップＳ２３に進む。

ステップＳ２３では、処理画面数が演算部３の演算プロセッサ数で割り切れるかどうかの判別を行う。ホストコンピュータ１０は、処理画面数が演算プロセッサ数で割り切れるとき（Ｙｅｓ）、ステップＳ２４に進み、処理画面数が演算部３の演算プロセッサ数で割り切れないとき（Ｎｏ）、ステップＳ２５に進む。

ステップＳ２４では、ホストコンピュータ１０は、画面並列処理フラグをセットしてステップＳ２５に進む。

ステップＳ２５では、ホストコンピュータ１０は、全演算ブロックに対する解析が終了したかどうかの判断を行う。全演算ブロックに対して演算処理方法の解析が終了していないとき（Ｎｏ）、ステップＳ２６に進む。

ステップＳ２６では、ホストコンピュータ１０は、次の演算ブロックの情報を読み込んで解析を続行する。すなわち、ホストコンピュータ１０は、次の演算ブロックの情報を読み込み後、ステップＳ

２２に進む。一方、ホストコンピュータ１０は、ステップＳ２５において、全演算ブロックに対する解析処理が終了しているとき（Ｙｅｓ）、リターンに進んでこのサブルーチンＳＵＢ２を終了する。

このようにして画面並列処理と画面分割処理の判別を行うことにより、プログラムに対する以後サブルーチンＳＵＢ３での実行コード生成が行われる際に、例えば前述したアーキチャクチャに基づくモデルが崩されないようにしている。

このように行われたソースプログラムの解析結果及び処理制御に応じたコンパイルが、図７に示すサブルーチンＳＵＢ３のフローチャートに従って行われる。

すなわち、ステップＳ３０では、ホストコンピュータ１０は、コンパイルするために供給されるソースプログラムがデータフロー制御部２用のプログラムか否かの判別を行う。ホストコンピュータ１０は、ソースプログラムが、データフロー制御部２用のプログラムの場合（Ｙｅｓ）、ステップＳ３１に進み、ソースプログラムが、データフロー制御部２のプログラムでない場合（Ｎｏ）、演算プロセッサ用のプログラムとしてステップＳ４０に進む。

ステップＳ３１では、ホストコンピュータ１０は、画面並列処理フラグが立っているかどうかを判別する。ホストコンピュータ１０は、画面並列処理フラグが立っているとき（Ｙｅｓ）、画面数が演算プロセッサ数で割り切れていることを示すものと判断してステップＳ３２に進み、画面並列処理フラグが立っていないとき（Ｎｏ）、ステップＳ３３に進む。

ステップＳ３２では、ホストコンピュータ１０は、１画面すべてのデータを対応する１つの演算プロセッサに供給して複数の画面に

対して並列に画像処理を行うデータフロー制御部２の実行コードを生成する。この後、ステップＳ４３に進む。具体的な画像処理として、例えば１画面の濃度平均値をとる等の処理である。これにより、データフロー制御部２に対して生成される実行コードとして、例えば演算部３の１つの演算プロセッサに対して演算に必要な１画面のデータを送るような実行コードが生成される。

ステップＳ３３では、ホストコンピュータ１０は、統合フラグが立っているか否かの判定を行う。この統合フラグが立っていないとき（Ｎｏ）、ステップＳ３７に進み、この統合フラグが立っているとき（Ｙｅｓ）、ステップＳ３４に進む。

ステップＳ３４では、ホストコンピュータ１０は、端数フラグが立っているかどうかの判定を行う。端数フラグが立っているとき（Ｙｅｓ）、ステップＳ３５に進み、端数フラグが立っていないとき（Ｎｏ）、ステップＳ３６に進む。

ステップＳ３５では、ホストコンピュータ１０は、統合処理及び端数処理を含む画面分割処理を行うデータフロー制御部２の実行コードを生成してステップＳ４３に進む。

この画面分割処理は、例えば必要に応じて、統合処理的な演算を行った結果等のデータを演算部３の各演算プロセッサに数値として送り、各演算プロセッサがこの供給された数値を基に演算を行う処理を含む。

すなわち、画面分割処理の実行コードは、データフロー制御部２のプロセッサが、演算部３の演算プロセッサの数分の小領域のデータを入出力部５内の共有メモリから読み出して、各演算プロセッサ内の分散メモリに配給し、各演算プロセッサの演算終了後に各処理

済みのデータを入出力部５内の共有メモリに回収するという制御を繰り返し行うようなソースプログラムから生成されたものである。

また、端数処理の実行コードは、剰余領域の画像データに対して繰り返し行う処理を１回多くし、小領域の総数を越えた部分からのデータ処理結果の入出力部５の共有メモリに対する書込みアクセスを禁止する条件文を含む実行コードである。すなわち、端数処理とは、データフロー制御部２が、全小領域に対する演算結果のみを入出力部５の共有メモリに書き込み、それ以外の演算結果を入出力部５の共有メモリに書き込まれないように処理することを意味している。

次に、ステップＳ３６では、ホストコンピュータ１０は、統合処理を含む画面分割処理を行うデータフロー制御部２の実行コードを生成してステップＳ４３に進む。

一方、ステップＳ３３の判断（Ｎｏ）によりステップＳ３７に進むと、このステップＳ３７でもホストコンピュータ１０は、ステップＳ３４と同様に端数フラグが立っているかどうかの判定を行う。そして、ホストコンピュータ１０は、端数フラグが立っているとき（Ｙｅｓ）、ステップＳ３８に進み、端数フラグが立っていないとき（Ｎｏ）、ステップＳ３９に進む。

ステップＳ３８では、ホストコンピュータ１０は、端数処理を含む画面分割処理を行うデータフロー制御部２の実行コードを生成してステップＳ４３に進む。

ステップＳ３９では、ホストコンピュータ１０は、通常通りにデータフロー制御部２の実行コードを生成してステップＳ４３に進む。

また一方、ステップＳ４０では、ホストコンピュータ１０は、統

合処理フラグが立っているかどうかの判定を行う。ホストコンピュータ１０は、統合処理フラグが立っているとき（Ｙｅｓ）、ステップＳ４１に進み、統合処理フラグが立っていないとき（Ｎｏ）、ステップＳ４２に進む。

ステップＳ４１では、ホストコンピュータ１０は、統合処理を含むデータフロー制御部２と演算部３の演算プロセッサの実行コードを生成してステップＳ４３に進む。

ステップＳ４２では、ホストコンピュータ１０は、通常通りに演算部３の演算プロセッサの実行コードを生成してステップＳ４３に進む。

ステップＳ４３では、ホストコンピュータ１０は、全演算ブロック分のデータフロー制御部２と演算部３の演算プロセッサに対する実行コードの生成が終了したかどうかの判定を行う。そして、ホストコンピュータ１０は、まだコンパイルが終了していないとき（Ｎｏ）、ステップＳ３０に戻って上述した手順を繰り返す。一方、ホストコンピュータ１０は、コンパイルが終了したとき（Ｙｅｓ）、リターンに進み、この実行コード生成処理を行うサブルーチンＳＵＢ３を終了する。

このように供給する画面数と演算プロセッサの数に応じて処理を切り換えて、演算部３の各演算プロセッサにおいて小領域、あるいは１フレーム単位での演算処理を行い、データフロー制御部２の演算プロセッサで統合的な演算処理を行い、必要に応じてこの統合的な演算処理結果を演算部３の各演算プロセッサに供給することにより、各演算プロセッサを与えられた領域に対する画像処理に専念させることができ、上述のアーキテクチャを満足するモデルを崩さず

に自動的に効率のよい方を選択して生成される実行コードを１種類だけに抑えることができる。これにより、生成する実行コードの生成量を極力抑え、プログラムの最適化も行うことができる。

ソースプログラム中の並列処理を示す識別子を検出し、この識別子に基づいてデータフロー制御部２と演算部３との処理に区分すると共に、演算部３の複数の演算プロセッサに対する実行コードを生成することにより、プログラム開発者には、各演算プロセッサへの役割分担を意識させることなくプログラムを書くことができ、プログラム開発の時間を短縮させることができる。これにより、開発コストを抑えることができるようになる。

また、分割した小領域の数の演算プロセッサ数に対して定まる端数（剰余分）に対応して上述した端数処理を行うことにより、従来の実行コード量に比べて大幅に抑えて効率的な処理を行わせることができる。

画像処理方法においては、分割した画面数に応じて役割分担された演算プロセッサでの演算結果を集めて統合処理する手順を経ることにより、画像処理方法における実行コードの生成量を大幅に軽減させて画像処理の効率のアップを図ることができ、短時間の開発を可能にして開発コストを抑えることができるようになる。

また、画像処理方法では、例えば濃度平均等の処理のように行わせる処理に応じて処理する画面数と演算プロセッサの数を調べて処理方法を選択して実行コード生成することにより、従来の画像処理以上に処理を高速化させることができる。これにより、例えばアプリケーションプログラムにおいても並列処理の一つとして要求されるアーキテクチャを満足するモデルを崩さずに画像処理を行って従

来の画像処理以上に処理を高速化させることができる。

画像処理装置では、データフロー制御部が供給したデータを演算部の各演算プロセッサで演算し、データフロー制御部で回収した各演算プロセッサの演算結果を用いて統合的な演算を行い、必要に応じて再び各演算プロセッサに分配することにより、演算部の各演算プロセッサの処理を与えられた小領域の計算に専念させ、供給される画像信号に並列処理による画像処理を施して従来より高速な画像処理を行わせることができる。

産業上の利用可能性

本発明では、複数の演算プロセッサに対して画像処理を並列に行わせるプログラム言語を用いて作成されたプログラムをコンパイルして実行コードを生成する際に、作成されたプログラムから画像処理における並列処理を示す識別子が付された制御文の識別と解析を行い、解析結果と複数の演算プロセッサの数に応じた各演算プロセッサへの並列処理の実行コードを生成する。そして、生成された実行コードに基づいて、各演算プロセッサがそれぞれ画像処理を行い、各演算プロセッサの演算結果を統合処理することにより、プログラム開発者は、各プロセッサへの役割分担を意識せずにプログラムを書くことができ、プログラムの開発時間を短縮させることができる。これにより、プログラム開発者の負担を軽減させることができ、開発コストを抑えることができるようになる。

請 求 の 範 囲

1. 複数の演算プロセッサに対して画像処理を並列に行わせるプログラム言語を用いて作成されたプログラムをコンパイルして実行コードを生成する並列処理プログラムのコンパイル方法であって、

上記作成されたプログラムから画像処理における並列処理を示す識別子が付された制御文の識別と解析を行う解析工程と、

上記解析工程による解析結果と上記複数の演算プロセッサの数に応じた各演算プロセッサへの並列処理の実行コードを生成する実行コード生成工程と

を有することを特徴とする並列処理プログラムのコンパイル方法。

2. 上記複数の演算プロセッサは、画面を分割して得られる小領域毎の画像データが順次割り振られて、上記画像データ並列演算し、この処理を繰り返すことで画面全体に対する画像処理が行われる

ことを特徴とする請求項1記載の並列処理プログラムのコンパイル方法。

3. 上記画面を分割して得られる小領域の総数が上記複数の演算プロセッサの数で割り切れないときに生じる剰余領域の画像データに対して繰り返し行う処理を1回多くし、この1回多くしたことにより生じる上記小領域の総数を越えた部分のデータ処理結果の入出力メモリへの書き込みを禁止することを特徴とする請求項2記載の並列処理プログラムのコンパイル方法。

4. 複数の演算プロセッサに対して画像処理を並列に行わせるプログラム言語を用いて作成されたプログラムの実行コードに従っ

て画像処理を行う画像処理方法において、

上記作成されたプログラムから画像処理における並列処理を示す識別子が付された制御文の識別と解析を行う解析工程と、

上記解析工程による解析結果と上記複数の演算プロセッサの数に応じた各演算プロセッサへの並列処理の実行コードを生成する実行コード生成工程と、

上記生成された実行コードに基づいて、上記各演算プロセッサがそれぞれ画像処理を行う画像処理工程と

を有することを特徴とする画像処理方法。

5. 上記解析工程は、上記プログラムの各演算処理内における、上記画像処理における画像データが複数に分割されて生成された小領域に対する演算結果に基づく統合的な演算の有無と、この統合的な演算の結果の分配の有無との少なくとも一方を検出する工程を含む

ことを特徴とする請求項4記載の画像処理方法。

6. 上記解析工程は、上記画像処理される画面の数が上記演算プロセッサの数で割り切れるか否かを判断する工程を含み、

上記実行コード生成工程は、上記解析工程で上記画像処理される画面数が上記演算プロセッサの数で割り切れると判断された場合には、上記各画面を構成する画像データ毎に上記各演算プロセッサで並列処理されるような実行コードを生成する

ことを特徴とする請求項4記載の画像処理方法。

7. 上記解析工程は、上記画像処理される画像データが複数に分割されて生成された小領域の数が上記演算プロセッサの数で割り切れるか否かを判断する工程を含み、

上記実行コード生成工程は、上記解析工程で上記小領域の数が上記演算プロセッサの数で割り切れないと判断された場合には、上記小領域のうち剰余の部分が上記演算プロセッサで処理されている際に上記剰余の部分が割り当てられていない演算プロセッサの処理結果をメモリへ書き込むことを禁止するように実行コードを生成することを特徴とする請求項４記載の画像処理方法。

８．　複数の演算の並列処理によって画像処理を行う画像処理装置であって、

入力された画像信号の画面分割された各小領域に対する画像処理の演算をそれぞれ並列に行う複数の演算手段と、

該演算手段からの各演算結果を基に統合的な演算を行うデータ供給制御手段とを

有することを特徴とする画像処理装置。

９．　上記複数の演算手段は、上記入力された画像信号の画面数と上記複数の演算手段に応じて上記供給される画像信号を１画面の分割された各小領域に対する画像処理の演算をそれぞれ並列に複数の演算手段で行いこれらの演算手段からの各演算結果を基に統合的な演算を行う第１の画像処理と、上記入力された画像信号の１画面単位で画像処理の演算を上記演算手段毎に行い複数の画面に対する画像処理をそれぞれ並列に行う第２の画像処理とのいずれか一方を処理する画面数と上記演算手段の個数に応じて切換選択する機能を有する制御手段の制御に応じて動作する

ことを特徴とする請求項８記載の画像処理装置。

１０．　上記制御手段は、処理する画面数が１あるいは上記画面数を演算手段の数で割った値が剰余を有する場合、上記第１の画像

処理を行い、

上記画面数が上記演算手段の数で割り切れる場合、上記第２の画像処理を行って制御をする

ことを特徴とする請求項９記載の画像処理装置。

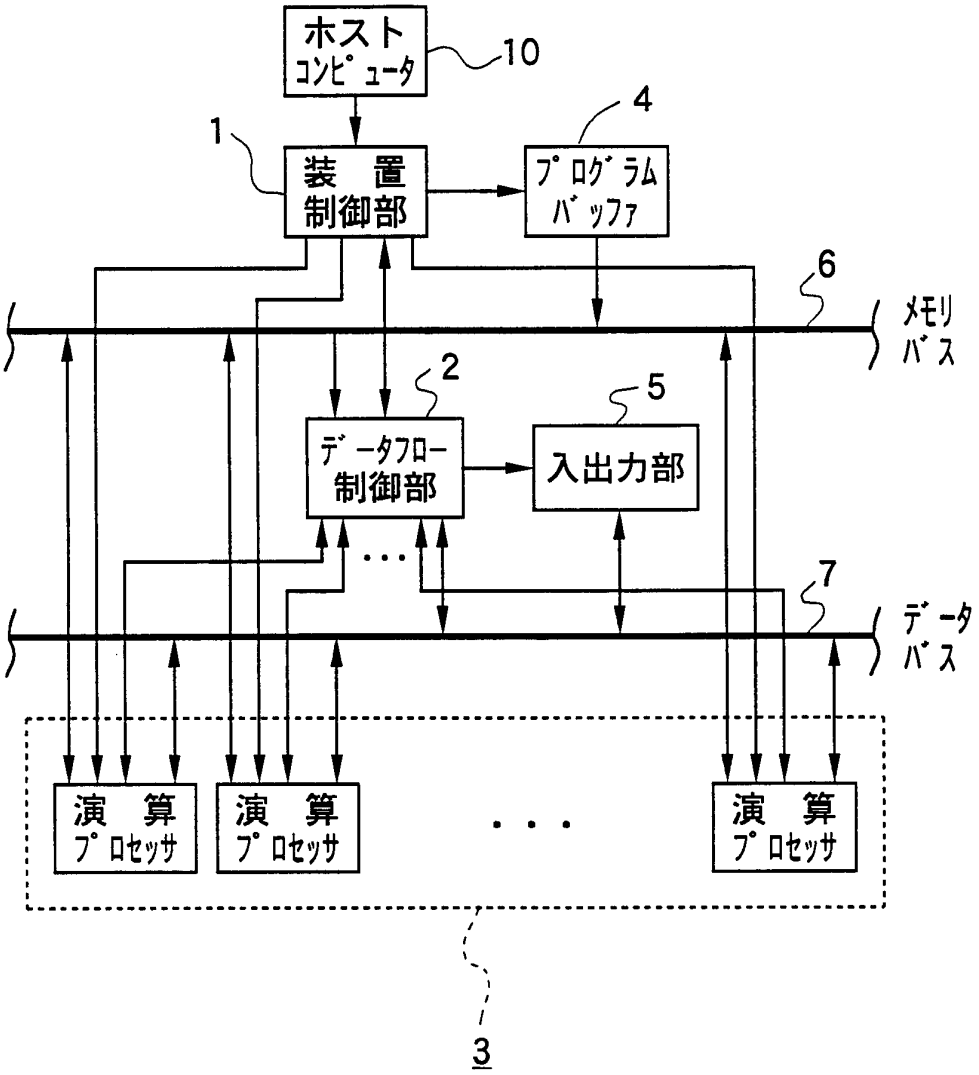
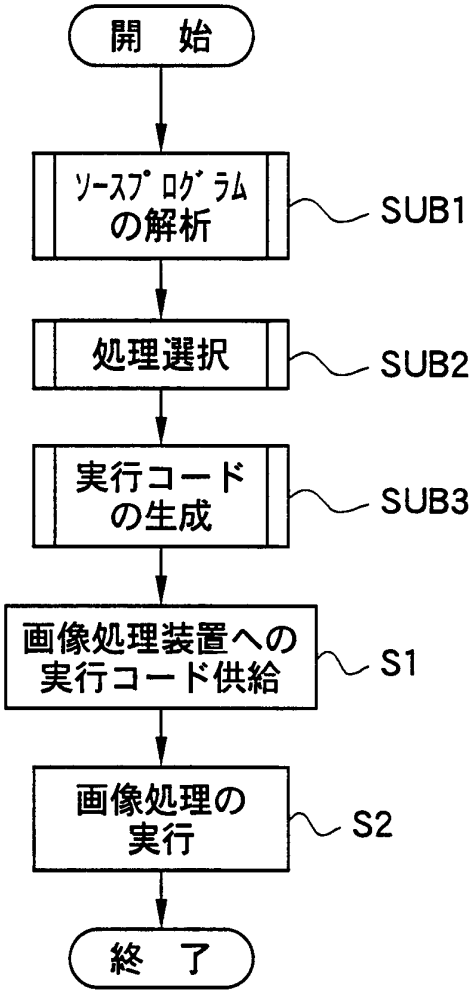

ホスト
コンピュータ
10
1
装置
制御部
4
プログラム
バッファ
6
メモリ
バス
2
データフロー
制御部
5
入出力部
7
データ
バス
演算
プロセッサ
演算
プロセッサ
演算
プロセッサ
3

Fig. 1


開　始
ｿｰｽﾌﾟﾛｸﾞﾗﾑ
の解析
SUB1
処理選択
SUB2
実行コード
の生成
SUB3
画像処理装置への
実行コード供給
S1
画像処理の
実行
S2
終　了

Ｆｉｇ．２

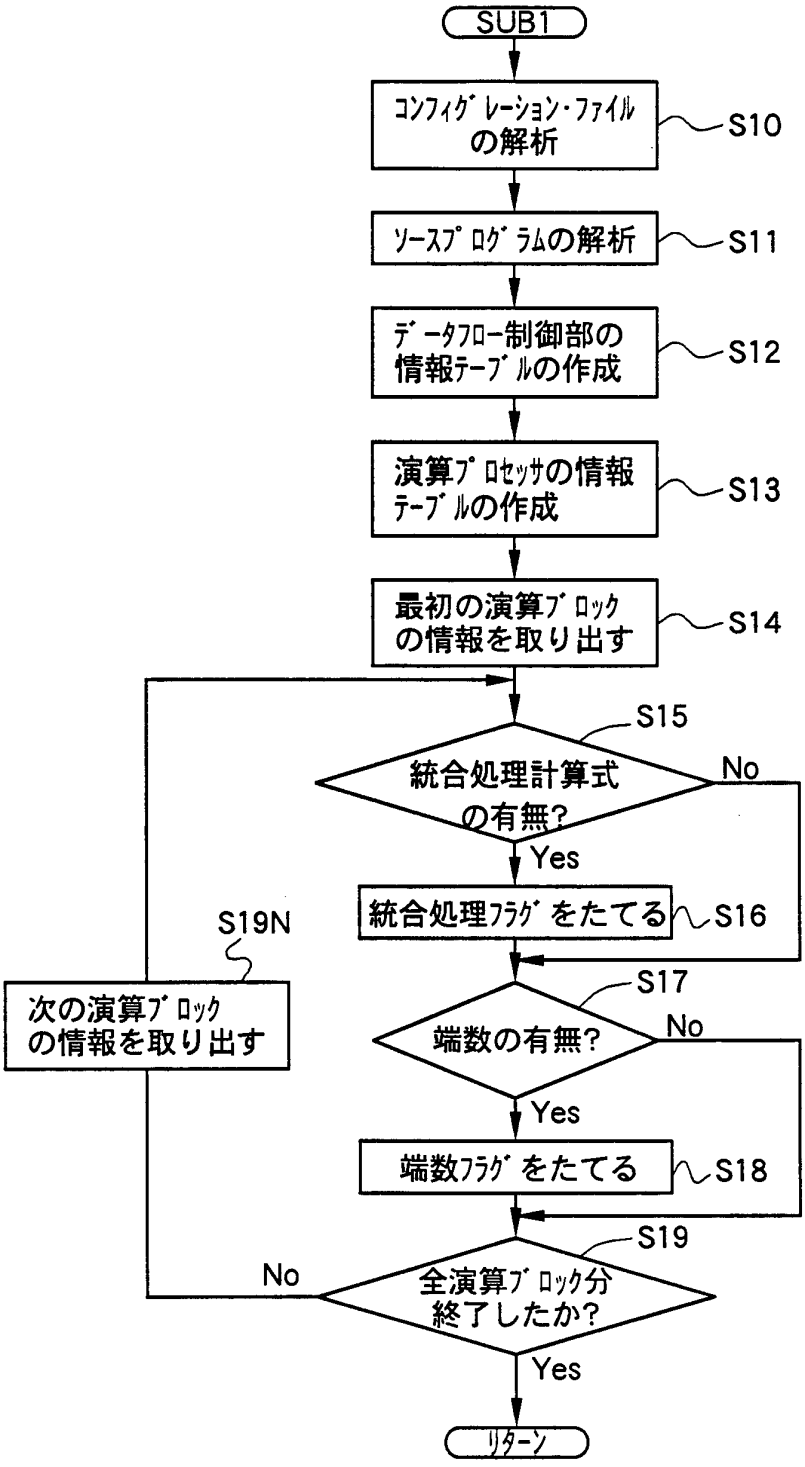
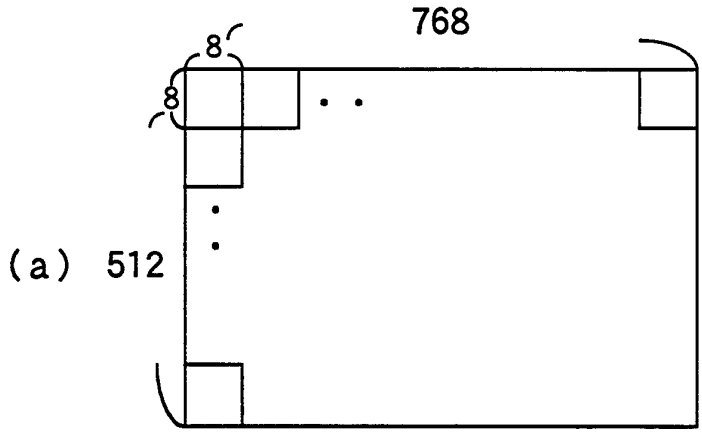
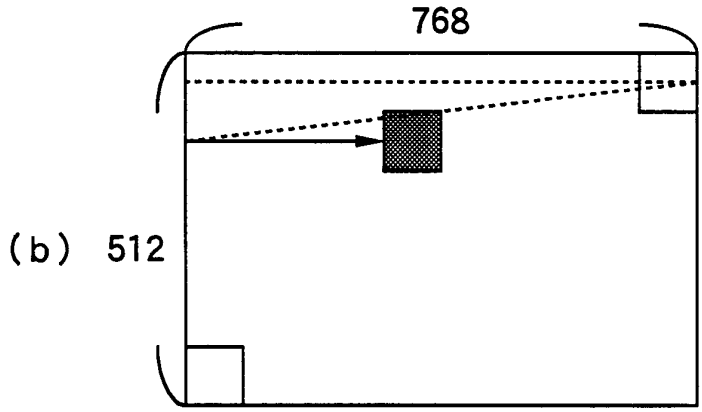

SUB1
ｺﾝﾌｨｸﾞﾚｰｼｮﾝ･ﾌｧｲﾙの解析
S10
ｿｰｽﾌﾟﾛｸﾞﾗﾑの解析
S11
ﾃﾞｰﾀﾌﾛｰ制御部の情報ﾃｰﾌﾞﾙの作成
S12
演算ﾌﾟﾛｾｯｻの情報ﾃｰﾌﾞﾙの作成
S13
最初の演算ﾌﾞﾛｯｸの情報を取り出す
S14
S15
統合処理計算式の有無?
No
Yes
統合処理ﾌﾗｸﾞをたてる
S16
S19N
次の演算ﾌﾞﾛｯｸの情報を取り出す
S17
端数の有無?
No
Yes
端数ﾌﾗｸﾞをたてる
S18
S19
No
全演算ﾌﾞﾛｯｸ分終了したか?
Yes
ﾘﾀｰﾝ

Fig. 3



1画面の画素数と画素ブロックの関係



画素ブロックのデータ供給の順序を説明する図

Fig. 4

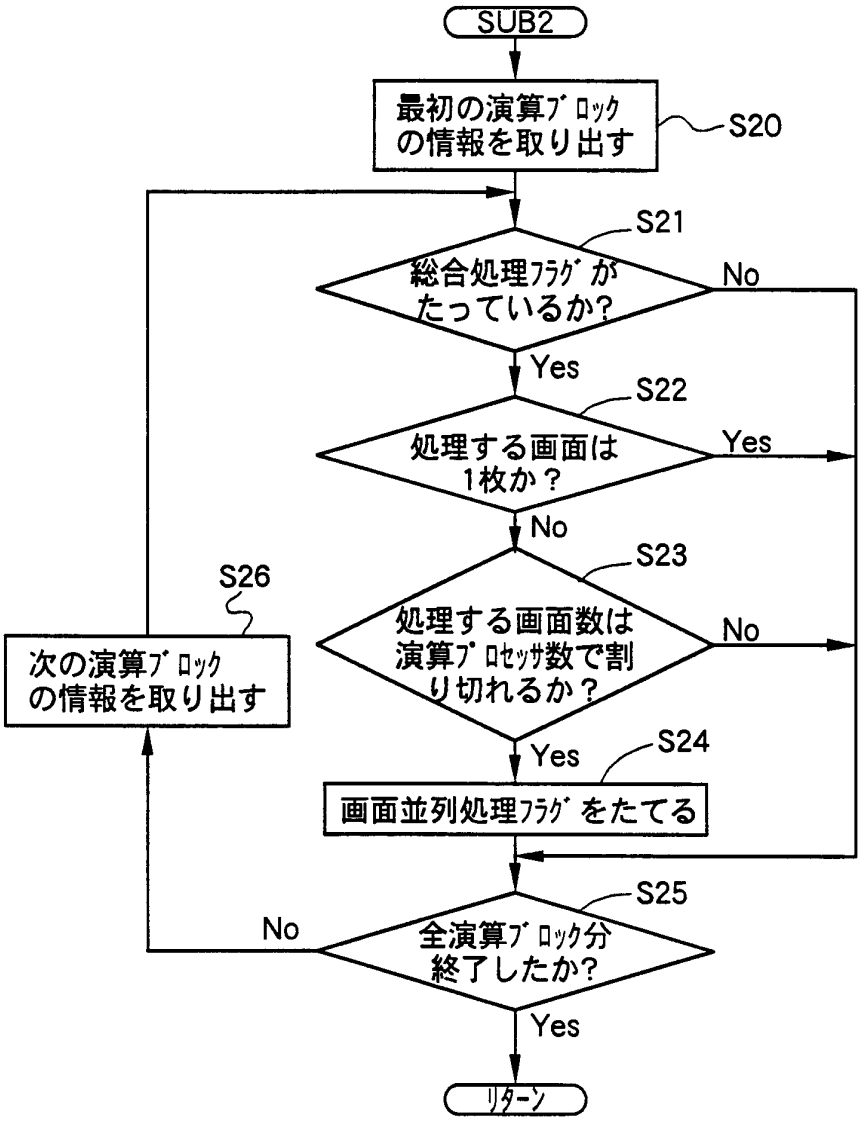

SUB2
最初の演算ﾌﾞﾛｯｸの情報を取り出す
S20
S21
総合処理ﾌﾗｸﾞがたっているか?
No
Yes
S22
処理する画面は1枚か？
Yes
No
S23
処理する画面数は演算ﾌﾟﾛｾｯｻ数で割り切れるか？
No
Yes
S24
画面並列処理ﾌﾗｸﾞをたてる
S25
全演算ﾌﾞﾛｯｸ分終了したか?
No
Yes
S26
次の演算ﾌﾞﾛｯｸの情報を取り出す
ﾘﾀｰﾝ

Fig. 5

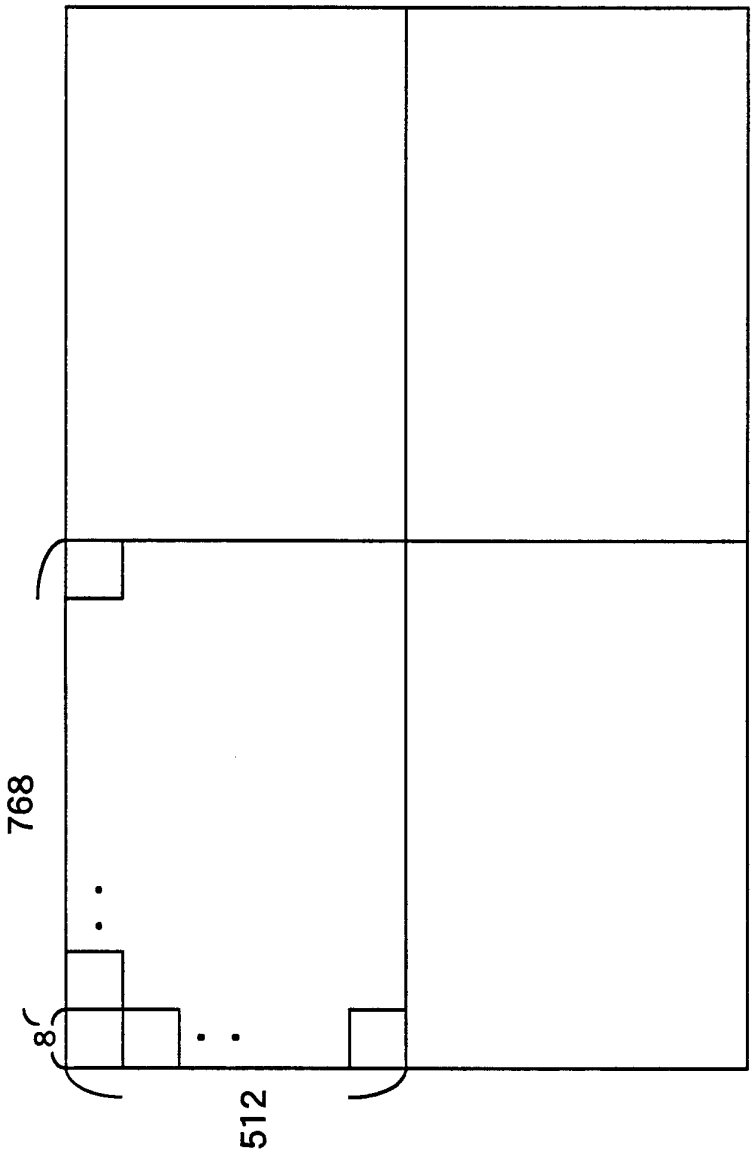


Fig. 6

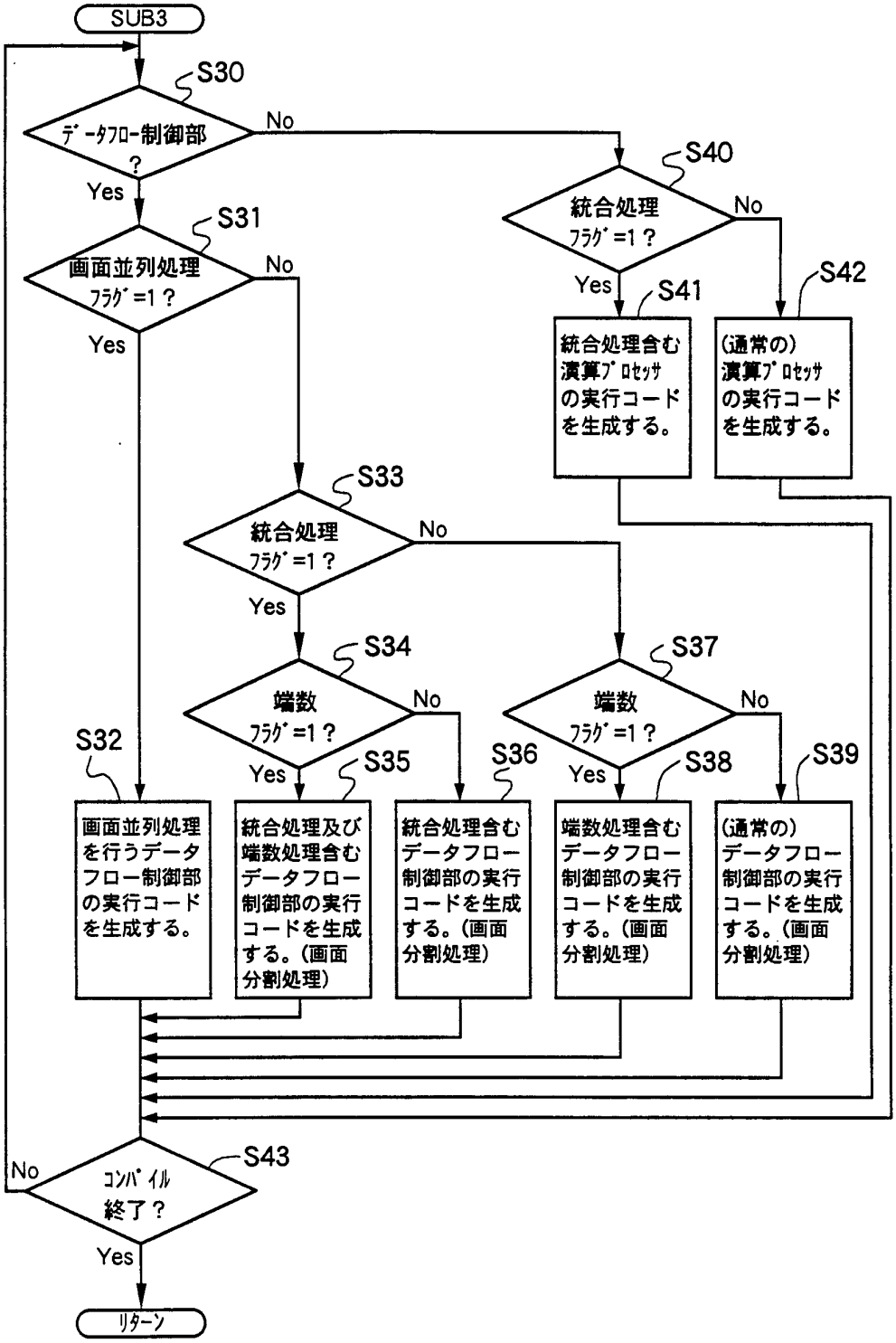

SUB3
S30
データフロー制御部？
No
Yes
S31
画面並列処理フラグ=1？
No
Yes
S40
統合処理フラグ=1？
No
Yes
S41
統合処理含む演算プロセッサの実行コードを生成する。
S42
(通常の)演算プロセッサの実行コードを生成する。
S33
統合処理フラグ=1？
No
Yes
S34
端数フラグ=1？
No
Yes
S37
端数フラグ=1？
No
Yes
S32
画面並列処理を行うデータフロー制御部の実行コードを生成する。
S35
統合処理及び端数処理含むデータフロー制御部の実行コードを生成する。(画面分割処理)
S36
統合処理含むデータフロー制御部の実行コードを生成する。(画面分割処理)
S38
端数処理含むデータフロー制御部の実行コードを生成する。(画面分割処理)
S39
(通常の)データフロー制御部の実行コードを生成する。(画面分割処理)
S43
コンパイル終了？
No
Yes
リターン

Fig. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP95/00552

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl^6 G06F15/16

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int. Cl^6 G06F15/16

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho 1925 - 1995
Kokai Jitsuyo Shinan Koho 1971 - 1995

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JP, 5-12033, A (Hitachi, Ltd.), January 22, 1993 (22. 01. 93) (Family: none)	1, 2, 4
X Y	Jun Hasebe, Ryohei Kato, Tokuichi Ito, Hiroyuki Fujita, Jun Yonemitsu, Atsushi Kikuchi "Real Time Image Processing System SIPS -High-Speed Processing Method-" IPSJ General Convention Lecture Transaction VOL. 35th, NO. 3, 1987, p.2017-p.2018	8 1, 2, 4
Y	JP, 60-140456, A (Hitachi, Ltd.), July 25, 1985 (25. 07. 85) (Family: none)	2, 8

☑ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
June 12, 1995 (12. 06. 95)	July 4, 1995 (04. 07. 95)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (July 1992)

国際調査報告

国際出願番号 PCT/JP 95/00552

A. 発明の属する分野の分類（国際特許分類（IPC））

Int. Cl⁶ G06F15/16

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（IPC））

Int. Cl⁶ G06F15/16

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報 1925－1995年
日本国公開実用新案公報 1971－1995年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献のカテゴリー＊	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Y	JP，5－12033，A（株式会社 日立製作所），22. 1月. 1993（22. 01. 93）（ファミリーなし）	1，2，4
X Y	長谷部 淳，加藤良平，伊藤徳一，藤田裕之，米満 潤，菊池 教「リアルタイム画像処理システム SIPS －高速処理方法－」情報処理学会全国大会講演論文集 VOL. 35th，NO. 3，1987，p. 2017－p. 2018	8 1，2，4

☑ C欄の続きにも文献が列挙されている。

☐ パテントファミリーに関する別紙を参照。

＊ 引用文献のカテゴリー
「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」先行文献ではあるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日
12. 06. 95

国際調査報告の発送日
04.07.95

名称及びあて先
日本国特許庁（ISA/JP）
郵便番号100
東京都千代田区霞が関三丁目4番3号

特許庁審査官（権限のある職員）
酒井恭信 5L 9190

電話番号 03－3581－1101 内線 3561

C（続き）．　関連すると認められる文献

引用文献のカテゴリー*	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Y	JP，60－140456，A（株式会社　日立製作所），25．7月．1985（25．07．85）（ファミリーなし）	2，8