5J−1

アプリケーションプログラムを基にしたプロセッサアーキテクチャの自動生成

−システムの実装と評価−*

門脇 一馬† 松田 和幸† 宮内 新† 石川 知雄†
武蔵工業大学‡

1 はじめに

近年のプロセッサ開発サイクルでは、汎用プロセッサの中で処理の短縮が必要とされる処理を、その処理専用のハードウェアを追加実装することにより高速化する試みが行われている. しかし, プロセッサ開発サイクルの短縮により, 専用ハードウェアによる高速化は短期間で行う必要が生じている.

我々は, 動画像処理を実時間で行うことを目標として, 画像処理を高速に行う専用プロセッサの設計方法についての提案を行った.[1] これは対象となるアプリケーションプログラムのアセンブラ記述を基にしてプロセッサのスペックを決定するものであったが, その作業は手作業で行われたため他の分野の処理には対応できていない. そこで, 本研究では画像処理等に用途を限定せず, 与えられたある特定のアプリケーションを高速化するために, 高級言語で記述されたプログラムから演算が最適に並列化されたマイクロプロセッサのHDL記述を生成することを目的とする.

2 生成されるプロセッサの構成

実行の最適化を進めるためには並列処理を行う必要がある. 本研究では,VLIW構成のプロセッサを提案することにより, 明示的に並列処理を実行する. まず, 与えられたアルゴリズム (以下, ターゲットアルゴリズム) に特化したプロセッサを生成するための基となるプロセッサ (以下, ベースプロセッサ) をVHDLで作成した. ベースプロセッサは並列度3であり,5段のパイプライン構成を持っている. ベースプロセッサにおいては, 基本命令セット (後述) のみを実装している. これに加えて, ターゲットアルゴリズムを最適に実装できるように命令を加えてく.

この他にベースプロセッサは次のような可変要素を持ち, これらをターゲットアルゴリズムに応じて変化させる.

- ALU ユニット内の演算器の種類と数
- レジスタの本数と幅
- メモリの容量
- アドレスバス・データバスの幅
- 並列度

*Automatic Generation of Processor Architecture Based on Application Program -Implementation and Evaluation of the System-

†Kazuma Kadowaki, Kazuyuki Matsuda, Arata Miyauchi, Tomo Ishikawa
‡Musashi Institute of Technology

3 システムの構成

本研究で提案するシステムの構成は図1のようになっている. 入力としては, 処理の対象となるアプリケーションのC言語によるプログラムソースに加え, 実装するデバイスの情報, 演算器のゲート数や遅延時間などのコンポーネント情報からなる. プロセッサを自動生成する流れは次のようになる.

1. プロプロセッサ：C言語のプログラムソースからプログラムの中で並列的に実行可能な箇所を抽出し, スケジューリングする.
2. プロセッサスペックの決定：演算の並列度や使用される演算の種類から, コンポーネント情報・デバイス情報を考慮し, プロセッサスペックを決定し, 最終的にVHDLソースを出力する.
3. 並列処理アルゴリズムの決定：演算の並列度とプロセッサのスペック情報からコードスケジューリングを行い, 実行可能なマシン語コードを出力する.

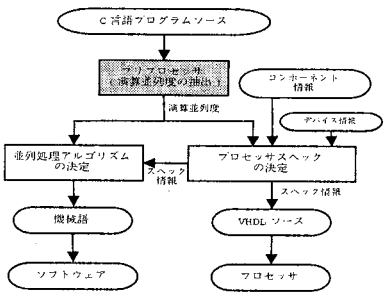

図1: システムの構成

4 プロセッサスペックの決定

本章では, プロセッサスペックの決定手法について述べる. スペックは以下の順序で行う.

1. 拡張命令の追加
2. 合成命令の定義と追加
3. レジスタ本数の決定
4. メモリサイズの決定
5. 命令フォーマットの決定
6. ゲートサイズの見積り
7. ゲートサイズに合わせて最終スペックの決定

4.1 命令セットの定義

命令セットは基本命令, 拡張命令, 合成命令の 3 つに大別される. 基本命令セットは, 既存の RISC アーキテクチャの命令セットの中で多く使用される命令で, 一般的な演算処理を行う上での最低限の命令セットであり, 前述のベースプロセッサに実装済みである.

拡張命令は, 基本命令とは逆に既存の RISC アーキテクチャの命令セットにおいてあまり使用されない命令と, それに加えて特定のアルゴリズムにおいて有用とされる命令をあげておく.

事前に定義しておいた拡張命令の中に有用な命令が存在しない場合, 複数の命令を 1 命令で実行可能な命令, すなわち合成命令を自動的に作成する. この操作を命令の合成と呼ぶ. 合成命令には, 水平型合成と垂直合成がある.

水平型合成は, 同一クロックサイクル中で並列に実行可能な命令の組合せについて, 実行の頻度の高いものについて,1 つのオペコードで複数の命令を同時に実行可能とするものである. これにより, 命令コード領域を縮小することが可能となる. 垂直型合成は, 連続した演算の組合せについて, その実行の頻度の高いものについて, 一つの命令発行で複数の演算を続けて行う命令を定義するものである. これらの操作については, コンポーネント情報を参照し, 処理に必要なビット幅の演算器が 1 クロックの時間の中で実行が可能なものについてのみ, 合成命令の実装を行う. 以上のような操作により, 命令セットの定義を行う.

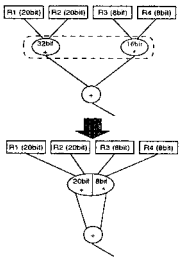

図 2: 水平型合成命令

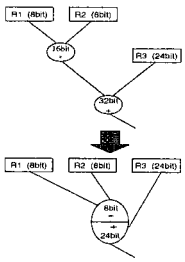

図 3: 垂直型合成命令

4.2 レジスタ本数, メモリサイズの決定

次に, プログラム実行に必要なデータ格納領域についてのスペックの検討を行う. アルゴリズム実行に必要なレジスタの本数は, 変数のライフタイム解析を行うことで決定する. また, 変数の最大サイズにあわせて, レジスタのビット幅, データバス幅も合わせて決定する. さらに, 実行に必要なプログラムサイズ, データサイズからメモリサイズ, アドレス幅を決定する.

4.3 スペック決定

これまでに決定された, 実装する命令セット・演算器構成に基づいて命令フォーマットを決定する. また, レジスタ数に応じてオペランドサイズを決める. そして, 並列度に応じて命令を複数並べ, ユニットコントロールフィールドを付加して命令フォーマットを決定する.

また, ここまではハードウェアの制限無しにスペック決定を行ってきたが, デバイスのクロックサイクル, ゲート数の制限により実装が不可能と判断される場合には, 追加された命令, レジスタ数の削減などを行い, 再び見積りを行う. このようにして, 最終的なスペックを決定する.

5 演算並列度の抽出

プリプロセッサにおいては, プロセッサの並列度の決定, 先に述べた合成命令を定義するための情報の抽出を行う. この操作は, コンパイラシステムを用いて得られる中間言語を用いて行われ, その基本ブロックを処理の単位とする. プロセッサの演算器構成はアプリケーションプログラム中で最も多く実行される部分について最適化されることが望ましいため, ループの内部に存在する基本ブロックに特に着目して行われる.

操作の流れとしては, まず得られた中間言語を基にして, ソースプログラム中の一つのステートメントを, 演算をノード, データパスをエッジとする木構造として表現する. この木構造の中で, 同時に実行できる命令 (ノード) の集合をつくり, その後の文中の命令については, 以前の文との依存関係が解消されるように同時に実行できる命令の集合に加えて行く. その際に, ループの連続する実行などにより不要となる命令は削除する.

図 2 のサンプルプログラムに対して, このような操作を行った例を図 3 に示す. これに同時に実行可能な命令の集合を抽出することができる. 以上のような操作により得られたデータの中で, 同時に実行可能な命令の集合の中での任意の命令の組合せが水平型命令合成の候補に, また木構造のエッジが垂直型命令合成の候補となる. またこれらの情報からコードスケジューリングを行うことも可能となる.

```
for(i=1;i<MAX;i++){
    x[i] = a[i-1] + a[i] * b + a[i+1];
    sum = sum + x[i];
}
```

図 4: サンプルプログラム

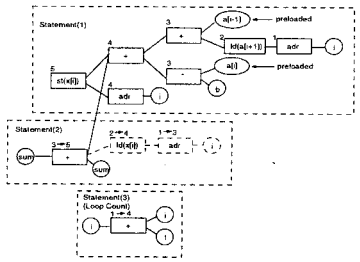

図 5: 再構成された木構造

6 おわりに

本稿においては, ターゲットアルゴンリズムから並列度を得て, プロセッサのスペックを決定する本システムの全体の流れについて報告した. プロセッサの外部にあるメモリのアクセスなどを考慮したスペック決定や, スケジューリングアルゴリズムの提案が今後の課題である.

参考文献

[1] 須藤, 宮内, 石川 :『画像処理プロセッサのアーキテクチャ決定法に関する検討 (1) ～アプリケーションプログラムからの命令セットの選定法～』情報処理学会, 第 59 回全国大会, 1999

[2] 松田, 門脇, 宮内, 石川 :『アプリケーションプログラムを基にしたプロセッサアーキテクチャの自動生成』第 3 回組込みシステム技術に関するサマーワークショップ,2001