C-019

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
―タイミング制約からの面積見積り手法―
Automatic Generation of Processor Architecture Based on Application Program
― The Estimation Method of Circuit area and Delay time ―

河合 一慶† **宮内 新**† **荒井 秀一**†
Kazuyoshi Kawai Arata Miyauchi Syuichi Arai

1. はじめに

近年の組込み分野におけるプロセッサ開発では, システムに要求する様々な制約が存在する. 音声処理や画像処理等の処理時間, 消費電力, 回路規模である. このような厳しい設計制約下では, 汎用プロセッサとソフトウェアによる組み合わせでは, 十分な性能を得ることができない場合がある. そのような場合, 専用のハードウェアを追加実装する事で, 処理速度の向上を図り設計制約を満足してきた. しかし, プロセスの微細化やアルゴリズムの改良により, 追加実装する事でコストが上積みされていた専用ハードウェアが不要となる事がある. このような専用ハードウェアの開発は, 非常に短期間のサイクルで行う必要がある. また, 近年設計生産性危機が叫ばれている. これは, プロセスの微細化による LSI の集積度の向上率と, 設計生産性の向上率に大きな開きが存在し, 将来設計生産性が追いつかなくなるという問題である.

2. これまでの研究

我々は, 開発期間の短縮と設計生産性の向上を目標に,C 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 記述を自動的に生成する研究を行ってきた [1].

2.1 自動生成システム

C 言語で記述したアプリケーションプログラムを,SUIF コンパイラシステムを用いて中間言語に変換する. そして, デバイスリソースを考慮せずに並列スケジューリングを行う. 求めた演算の並列度, 使用する演算の種類から, プロセッサスペックを決定し VHDL ソースを出力する. また, 生成したプロセッサ上で動作するマシン語を出力する.

2.2 見積りの対象となるプロセッサ・モデル

自動生成するプロセッサは, 明示的な並列演算が可能な VLIW 型処理機構をもつパイプライン・プロセッサであり, 複数の ALU やメモリアクセスユニットを持つ.

2.3 従来研究における問題点

従来研究 [1] では, アーキテクチャの決定段階におけるハードウェアコストの見積りが正確に行えなかった為に, 生成したプロセッサにおけるプログラムの実行時間が, ベースプロセッサにおける実行時間よりも結果が悪くなるという問題が生じた. これは, アプリケーションに有効な演算器を追加する事によって, 実行クロック数を減少させるアプローチであったのだが, アプリケーションの実行時間は自動生成したプロセッサのクロックサイクル時間とクロック数との積であり, 演算器を追加する事によるクロックサイクル時間の増加が, 減少したクロック数よりも大きかった為である. つまり, 実行時間という観点からは最適なアーキテクチャを選択できたとは言えない.

2.4 研究目的

アプリケーションの実行時間が最適となるようなプロセッサアーキテクチャを自動生成する為に, アプリケーションの実行クロック数だけではなく, 生成したプロセッサの動作周波数を考慮できるような手法を提案する. また, 動作周波数の違いにより変化するプロセッサの回路面積を正確に見積もる事により, 与えられた面積制約の下で最適となるようなアーキテクチャを探索する.

3. プロセッサアーキテクチャ選択問題

3.1 問題の定式化

様々なアーキテクチャから, 最適なアーキテクチャ構成を選択する問題は, 最適化問題として以下のように定式化し, 分岐限定法を用いて解く.

$$T(X) = F(X) \times \left[\sum_{i=0}^{N} \{F_i \times t(B_i, X) + c_i\} - b \right] \quad (1)$$

$$X = (x_0, x_1, \ldots, x_n, P) \quad (2)$$

$T(X)$ を最小にするベクトル解 X を探索する. ここで,P はプロセッサの並列度を定義し,X はアーキテクチャ構成として考えられる実装方法の組合わせである.

”N” はアプリケーションプログラムの基本ブロックの総数を定義する.

”B_i” は基本ブロック $\#i$ を定義する.

”$t(B_i, X)$” はアーキテクチャ構成 X を用いて基本ブロック B_i を実行する為に必要なパイプラインの実行サイクル数を定義する.”$t(B_i, X)$” を求める為に, リターゲット可能な命令スケジューラを利用する.

”c_i” は基本ブロック B_i から別の基本ブロックへの制御に必要なクロックサイクルを定義する.

”b” は与えられたアプリケーションプログラムの実行において, 分岐不成立によって減少する実行サイクル数を定義する.

$F(X)$ はアーキテクチャ構成 X における, 生成したプロセッサのクロックサイクル時間を定義する. よって,$T(X)$ はアーキテクチャ構成 X におけるアプリケーションの実行時間を定義することになり, 与えられた面積制約下において $T(X)$ を最小にする X を探索する.

†武蔵工業大学,Musashi Institute of Technology

3.2　$F(X)$ の求め方

最適化問題を解くにあたって,$F(X)$ を短時間でかつ正確な値を求めなければ, 最適解 X を現実的な時間で解く事ができない. そこで,$F(X)$ を以下のように定義する.

$$F(X) = D_{base} + \max_i(D_{min}(x_i)) \qquad (3)$$

D_{base} はプロセッサの最大遅延時間から ALU の遅延時間を引いた値であり, ベースプロセッサ固有の値である.D_{min} は演算器を速度優先で合成した場合の遅延時間であり, 演算器毎に速度優先で合成した値をあらかじめ求めておく. アーキテクチャ構成 X における, 最も遅延時間が大きい演算器の遅延時間とベースプロセッサの遅延時間の和によって $F(X)$ は短時間で求めることができる.

4.　タイミング制約からの回路合成

$F(X)$ はデータベースから求めた仮の最大遅延時間であるので, 実際に論理合成ツールによって $F(X)$ のクロックサイクル時間で動作するプロセッサが合成可能であることが保証されなければならない.

4.1　最適化による値のばらつき

各演算器は, $F(X)$ で与えられたタイミング制約によって様々な値をとり得る. 例として,$4bit \times 4bit$ 乗算器を考える. この回路の回路面積, 最大遅延時間は論理合成ツールの最適化によってどれくらいの範囲でばらつくのかを調べる. 使用した論理合成ツールは Synopsys 社の Design Compiler, ライブラリはローム株式会社 0.35μm CMOS デザインルールを使用して, 以下の条件で合成した.

1. 面積の制約を 0 にする
2. 遅延時間の制約を 0ns から 5ns まで増加させる
3. 回路面積, 遅延時間を最適化オプションをつけて論理合成する

横軸を最大遅延時間, 縦軸を回路面積としてグラフを描くと図 1 のようになる. タイミング制約によって様々な値をとり得ることが解る. 従来法では, 面積優先で回路を合成する事で得られた値（図中点 A）をデータベースとして保持しておけば良い. 従来法では, タイミング制約は与えられない, 又は与えられても十分大きい値であり, 回路面積は点 A に十分収束しているからである. しかし, $F(X)$ が点 B で与えられた場合,CAD により合成される回路面積は図中点 C となり, データベースの値 A と実際に合成される C との間に誤差が生じることになる.

4.2　タイミング制約からの回路面積見積り手法

制約を与える事で, 論理合成ツールは遅延時間と回路規模のトレードオフによって最適化を行い, その結果は図 1 で示すように回帰分析により二次の多項式で近似可能である事が解る.

また, 横軸をタイミング制約, 縦軸を合成された遅延時間としてグラフにしたものが図 2 である. グラフを見るとコンポーネントに対して与えたタイミング制約どおりに合成できる領域と, タイミング制約を与えても最大遅延時間が変化しない領域に分けることができる.

以上から, タイミング制約を与えることで, そのタイミング制約どおりの回路を合成可能な領域が存在し, その時の回路面積は見積もり近似式から求めることが可能である事が解った.

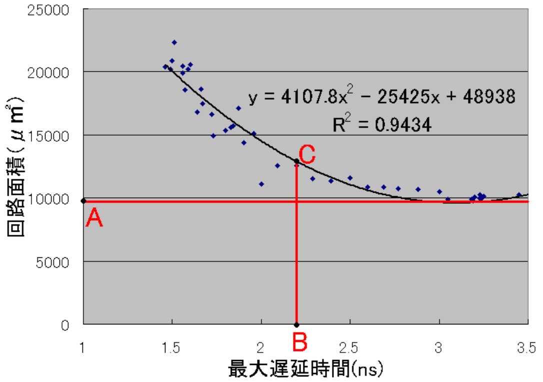

図 1: タイミング制約による値のばらつき

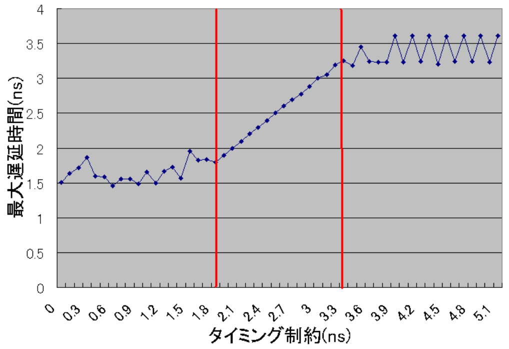

図 2: タイミング制約による最大遅延時間の変化

5.　むすび

生成するプロセッサの動作周波数をアーキテクチャ探索時に高速に見積もることで, 実行時間が最適になるアーキテクチャを探索する手法について述べた.

アーキテクチャ構成を現実的な時間で求める為に, プロセッサのクロックサイクル時間はデータベースから求める. その値をタイミング制約として利用し, タイミング制約からの回路合成, タイミング制約からの回路面積見積りを行う事で, 最適なアーキテクチャの探索が可能である.

参考文献

[1] 松田 和幸, 門脇 一馬, 宮内 新, 石川 知雄：”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”,SWEST,2001.7.25

[2] 塩見 彰睦, 久須 裕之, 伊藤 真紀子, 今井 正治：”パイプライン・プロセッサのためのアーキテクチャレベル面積見積り手法”, 情報処理学会論文誌,Vol.43 No.5,pp1171-1180(2002)