2J−9

特定用途向きマイクロプロセッサ開発システム (1) −システムの概要−

佐藤 淳†, 博多 哲也†, Alaudain Y. Alomary†, 今井 正治†, 引地信之††
†) 豊橋技術科学大学 情報工学系 ††) SRA

1. はじめに

半導体回路の集積度の向上およびASIC(特定用途向き集積回路)設計ツールの発達に伴い, 専用のCPUコアを内蔵した大規模なASICマイクロプロセッサ (ASIP : Application Specific Integrated Processor) の実現が可能になった.

本稿ではまず, ASIPの開発手法について考察する. 次に, VLSIの設計およびCPUアーキテクチャの設計に関する専門知識を持たない応用システム技術者が, 比較的容易にASIPの設計・開発を行なえることを目的とする, ASIP開発環境を提案する.

本システムは, CPUの動作記述などを入力とする従来のCPUコア自動生成システムとは異なり, 応用プログラムの特徴 (演算子の実行頻度など) を利用してCPUコアの生成を行なう. したがって, ユーザーはHDL(Hardware Description Language) などによるCPUのハードウェア記述を行なう必要がない.

また, CPUコアの生成と同時にソフトウェア開発環境 (コンパイラ, シミュレータなど) の生成を行なうことが可能であるので, ソフトウェアの開発を含めたシステム全体の開発期間の短縮が可能になると期待される.

2. システムASICの開発手法

ASIPのCPUコアを開発する場合, 以下の2通りの方法が考えられる.

(a) 既存のCPUを流用する,

(b) 専用CPUを自動生成する.

(a) の手法は, CPUコアのサイズが小さく, 既存のソフトウェア開発環境の流用が可能であるが, 命令の追加や削除が困難であり, 使用するCPUコアによって全体の性能が制約される. また, CPUの版権や使用料に関する問題も一般的には存在する.

CPUコアの開発に (b) の手法を用いた場合, 以下のような利点が考えられる.

(1) 特定の応用に特化した命令などの実現が容易である.

(2) ペリフェラルなどを含めたシステム全体の性能を高めるためのバランスのとれた設計が可能である.

(3) 特定のプロセス技術に依存しないので, 現在使用可能な最新のプロセス技術の使用が可能である.

一方, (b) の手法を実用化するためには, 以下の問題点を解決する必要がある.

(1) 自動生成が容易であり, かつ応用分野に適した高性能な専用アーキテクチャの採用.

(2) 既存の論理合成システムとのインタフェースの開発.

(3) 応用プログラム開発環境の整備.

これらの問題点を解決することができれば, 高性能なASIPの開発環境の実現が可能となると考えられる. 次節では, これらの問題点に対する解決法を提案する.

3. ASIP開発システム

3.1 システムの概要

ASIP開発システムの構成を図1に示す. 本システムは以下の4つのサブシステムから構成される.

(1) 応用プログラム解析部
C言語で記述された応用プログラムとデータの集合を入力し, 応用プログラムの静的および動的解析を行なう. 解析はC言語の演算子, データ構造, アクセス方法などの項目に関して行ない, 各項目について使用形態や使用頻度などの統計量を出力する[(1)].

(2) アーキテクチャ生成部
応用プログラム解析部で解析した結果にもとづき, CPUができるだけ高性能になるように命令セットと各命令の実現方法などのアーキテクチャ情報を決定する. 命令セットおよび各命令の実現方法を決定する場合の制約条件としては, 消費電力, チップ面積, 動作速度を用いる[(2)].

(3) CPUコア生成部
アーキテクチャ生成部で得られたアーキテクチャ情報にもとづいてCPUコアの生成を行なう. 生成するCPUコアのアーキテクチャを (4) で述べるコンパイラを考慮して生成することにより, コンパイラが想定しているハードウェアモデルと実際に生成するハードウェアの間のセマンティックギャップが小さくなり高性能化が期待できる. 生成されたCPUコアの設計結果は, 既存のCADシステムとのインタフェースが可能なHDLの形式で出力する.

Overview of the Application Specific Integrated Processor Design Environment
Jun SATO†, Tetsuya HAKATA†, Alaudain Y. Alomary†, Masaharu IMAI† and Nobuyuki HIKICHI††
†) Department of Information and Computer Sciences, Toyohashi University of Technology and ††) Software Research Associates

(4) 応用プログラム開発環境生成部

CPU コアの設計と同時に, コンパイラ, シミュレータなどのソフトウェア開発ツールの生成を行なう. コンパイラとシミュレータは, GNU ソフトウェアである GNU C Compiler(GCC) と GNU source-level debugger(GDB) を利用して自動生成する[3].

3.2 命令セットアーキテクチャ

GCC によるコード生成とハードウェアの生成を容易にするために, GCC の中間言語 RTL(Register Transfer Language) を基礎として, 命令セットを 3 つのクラスに分ける[2],[3].

(1) Primitive Simple RTL (PRTL)

(2) Simple RTL (SRTL)

(3) Extended RTL (XRTL)

3.3 CPU アーキテクチャ

GCC に与えるパラメータの変更を最小限にとどめ, コンパイラと CPU コアの自動生成の容易さを考慮して決定したアーキテクチャのモデルは, 次のような諸元を持つ.

(1) 命令形式
- 命令フォーマットは 1 ワード (32 ビット) 長のみに固定する.
- 3 アドレス形式を採用する.

(2) 記憶空間の構成
- バイトをアクセス可能にする.
- 線形なアドレス空間を持つ.
- 最大アドレス空間を 4GB として, この範囲内でアドレス空間を可変とする.

(3) レジスタの構成
- 汎用レジスタ方式とし, 一様なレジスタアクセスを可能にする.
- 用途別のレジスタ (整数, 浮動小数点など) を持つ.

(4) 処理方式
- シングルプロセッサを対象とする.
- 例外処理 (ゼロ割算など) は固定とする.
- 記憶管理と記憶保護のための専用ハードウェアを持たない.

パイプラインのステージ管理などは現時点の GCC では対応することが困難である[4]. これらは今後の GCC の改良によってサポート可能であると考えられる. 高性能な CPU コアを生成するためには, キャッシュの有効な利用, 記憶管理, 記憶保護, 例外処理などを考慮する必要がある.

4. おわりに

本稿では, 特定用途向きマイクロプロセッサ開発システムの提案を行ない, システムの概要について述べた.

これまでに, 応用プログラム解析部, および応用プログラム開発環境生成部の一部であるコンパイラ生成部のプロトタイプが実現されてきた[1],[3]. 現在, アーキテクチャ生成部と CPU コア生成部のプロトタイプを実現するために, 変更可能なアーキテクチャパラメータの範囲の決定, 命令の実現方法の決定方法, CPU コアの生成方法を検討している. アーキテクチャ生成部はエキスパートシステムを用いて実現する予定である. また, シミュレータ生成部の実現は今後の課題である.

謝辞 本研究に御協力頂くメンター・グラフィックス・ジャパン(株), 富士通 VLSI(株), および御討論して頂いた豊橋技術科学大学 VLSI 設計研究室の諸兄に深謝します.

参考文献

(1) 佐藤 淳, 福田孝一, 今井正治 : “特定用途向き CPU コアの自動合成法に関する一考察,” 信学技報, **VLD89-70**, 1989.

(2) 佐藤 淳, 福田孝一, 市田真琴, 今井正治 : “特定用途向き CPU コアの命令セットの実現方法に関する一考察,” 信学技法, **VLD89-110**, 1990.

(3) 博多哲也, 佐藤 淳, 今井正治, 引地信之 : “特定用途向きマイクロプロセッサ開発システム (2),” 情報処理学会第 42 回全国大会, 1991.

(4) Richard M. Stallman : “Using and Porting GNU CC,” 1990.

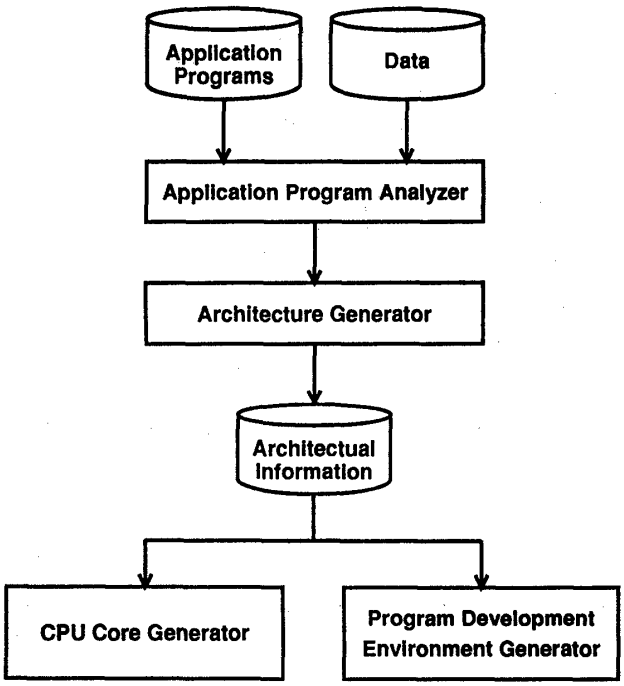

図 1 ASIP 開発システムの構成