1C－2　専用プロセッサ設計支援システム(SYARDS)における パイプライン処理システムのための最適化

吉田 裕　上田 穣　樋渡 仁　白井 克彦

早稲田大学 理工学部

1　はじめに

近年のLSI製造技術の急速な進歩に伴い、様々なシステムのLSI化が実現されている。そしてLSIシステムの規模が増大し、その応用分野も急激に広がりつつある。今後、LSIの多様な分野への応用が進む中で、一層、専用プロセッサへの要求が増大していくと考えられる。

その反面、システムが大規模になるとシステムの設計は複雑化し、設計期間の長期化とコストの増加が生じる。今後の専用プロセッサの要求の増大に応えるためには、より容易に短期間でLSIを設計することのできる設計支援システムが必要不可欠となる。

そこで、我々は高位仕様記述(Pascal)を入力とした設計支援システムの研究、開発を行っている。[1] [2]

本報告では、パイプライン処理システムの設計におけるパイプラインステージ構成の変更と、その構成に応じたコードを生成することでハードウェア資源の効率的利用とパイプライン処理による高速なプロセッサの実現を目指すものであり、その方法と具体例を示す。

2　システムの概要

本システムは、中間情報生成系、解析系、合成系の3つの系からなる。Userは、ほぼPascalの仕様にしたがって、ハードウェアとして実現したいアルゴリズムの仕様記述を記述し、実行するサンプルデータと共に入力する。

中間情報生成系では、仕様記述に対して字句・構文解析、冗長なコードを削除する等の種々の解析を行い、中間情報として3番地文によるフロー情報を出力する。

解析系ではサンプルデータに基づいてシミュレーションを行なう。ブロック・命令ごとの頻度解析やパイプライン処理のためにデータ依存解析を行い、Userに結果を出力する。同時に高機能命令として合成可能な命令を抽出する。

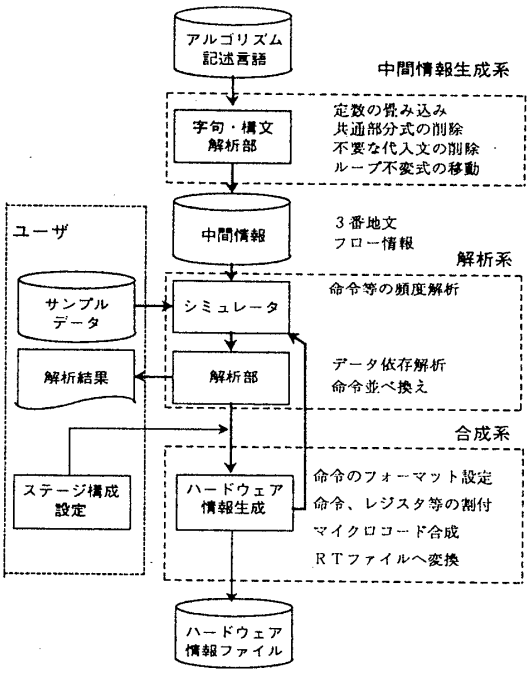

図1: システム構成図

そして合成系においてUserより与えられた機能回路、命令をレジスタ・メモリが最少となるよう割付を行い、システムの出力として使用した命令・レジスタの情報とそのフロー情報をハードウェア情報として出力する。

Userは、解析系での結果と合成系より得られたコードの再度のシミュレーションにより、速度と資源の制約を満たしたハードウェアを合成して行く。

3　パイプライン処理のための最適化

本研究ではパイプライン処理を導入し、速度性能を上げることとハードウェア資源の有効利用を目的としている。本システムでは生成されるプロセッサのデータパス・制御回路はある程度限定しており、各ステージを機能ごとに機能回路として実現する。各ステージは、命令の読みだし、デコード、オペランドの読み出

Optimization for Pipeline Processing in Special-Purpose Processor Design System (SYARDS)

Yutaka YOSHIDA, Yutaka UEDA, Jin HIWATASHI and Katsuhiko SHIRAI

Department of Electrical Engineering, Waseda University

し、演算の実行、結果の書き込み、を主な機能とする。機能回路としてはALU,ROM,RAM,デコーダ,レジスタファイルなどがあげられる。この時、使用する機能回路間で競合が起こらないよう構成する必要がある。

3.1 パイプライン処理における障害

アルゴリズムにおいてパイプラインの障害となるのは、一般に分岐命令を実行した場合及び連続する命令間にデータの依存関係がある場合に発生する。この対処方法として遅延実行方式を採用する。

分岐命令については、分岐命令と分岐先命令の間にステージ構成に応じた数の挿入命令を挿入する。またデータ依存のある命令が連続するような場合には、プログラムの実行に影響を与えない範囲で命令の順序を変更し、データ依存関係のある命令が連続しないようにする。適当な挿入命令が存在しない時はNOP命令の挿入を行なう。

3.2 命令の高機能化

命令の高機能化は前の命令の結果レジスタの値を後の命令のオペランドとして使用する命令列を1つの高機能命令として合成するものである。命令の高機能化を行なうことでデータ依存関係によるNOP命令の挿入を少なくすることができる。高機能化可能な命令は基本ブロック内で結果レジスタが1つの場合で、かつ前命令がALU命令の時のみ抽出する。

4 システムの適用例と評価

例としてParcor格子型フィルターのアルゴリズムについて評価を行った結果を示す。

パイプライン構成としては3段から5段のステージを構成した。次に各ステージに応じてアルゴリズムコードを生成し、シミュレートした結果を示す。ステージ数が少ないとデータに依存関係が少なくなることから障害が少なくなり、NOP命令の挿入が少ない。データ依存がある命令を高機能化可能命令として抽出する。この抽出された命令を実際に合成するかどうかはハードウェア資源、頻度解析等を考慮して選択する。命令の高機能化により、所要クロック数が減少しているのが分かる。

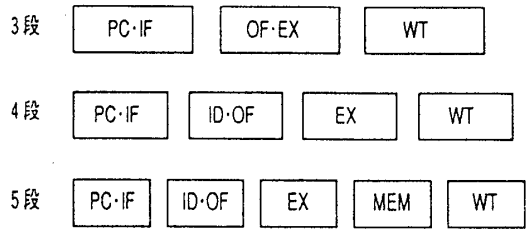

図2: ステージ構成

表1: 実行ステージ状況

ステージ数	生成コード数	所要クロック数	利用効率 [%]
3段	143	47243	94.35
3段 +OP1	135	43403	95.19
4段	201	66018	79.15
4段 +OP1	181	62182	78.98
+OP2	181	56418	79.42
5段	249	84069	65.53
5段 +OP1	215	70245	69.96
+OP3	189	60258	78.34

5 むすび

高位の仕様記述から専用プロセッサを設計するシステム(SYARDS)におけるパイプライン処理システムの導入について報告した。本システムにより、ユーザはパイプラインステージを機能回路の制限を満たしながら自由に設定し、ハードウェア資源の利用効率とスピードの要求のトレードオフから容易に最適な専用プロセッサの設計が可能となる。

今後は、ハードウェア機能回路の構成をライブラリ化するなどしてより多くの情報を収集することを可能とし、命令の高機能化の選択を幅広く行なうことにより現実的なシステムの実現と評価を試みたい。

参考文献

[1] 池永 剛, 白井 克彦:「高級言語によるアルゴリズム記述を入力とする専用プロセッサ設計支援システム」, 情報処理学会論文誌,Vol.32,No.11,(1991)

[2] Hironobu KITABATAKE, Katsuhiko SHIRAI:「Functional Design of a SpecialPurpose Processor Based on High Level Specification Description」, IEICETRANS. FUNDAMENTALS, VOL.E75-A,NO 10,(1992)