C-7　HDLによるRISCプロセッサの設計経験(Ⅰ)

—命令セットアーキテクチャと設計—
Design Experiences of a RISC Processor with HDL
—Instruction Set Architecture and Design—

大八木 睦†　池田 修久†　山崎 勝弘†
Mutsumi Oyagi†　Nobuhisa Ikeda†　Katsuhiro Yamazaki†

1　はじめに

我々は、プロセッサの基本的なアーキテクチャを理解する目的で、独自に作成した16bit命令セットを対象として単一サイクル、マルチサイクル、パイプラインの各方式で16bit RISCプロセッサの設計をVerilog-HDLを用いて行った。また、動作検証には3つのテストパターンを与え、FPGAを想定したゲートレベルシミュレーションを行った。

2　16bitRISCプロセッサの設計

2.1　設計環境と背景

我々はHDLを用いたプロセッサの設計は初めてであるので、HDLによる簡単なプロセッサの設計と検証の一連の手順を理解することを目的として、本実験を行った。そこで、文献[1]にある単一サイクル、マルチサイクル、パイプラインの3方式で設計し、シミュレーションを行って検証した。CADツールとしてFoundation ISE（XILINX社）とModelSim（MTI社）を用いた。

2.2　命令セットアーキテクチャ

対象とする命令セットは16ビットの独自命令セット（以下MONI命令セット）であり、MIPS32ビット命令セットのサブセットに当たる。図1にMONI命令フォーマットを示す。

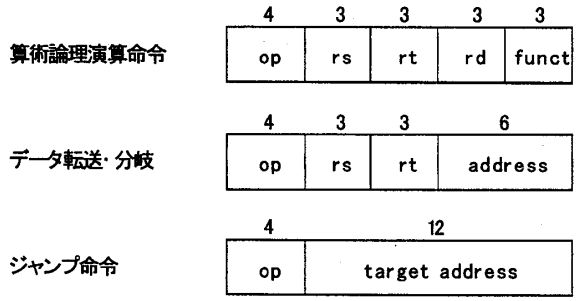

図1：MONI命令フォーマット

MONI命令セットは16ビットの固定命令語長であり、算術論理演算においては3オペランド方式を用いている。汎用レジスタを8つ用意し、演算はレジスタ間のみで行うLoad/Storeマシンである。現段階で用意されている命令は、算術論理演算命令であるADD、SUB、OR、AND、BIG（ソースレジスタの値の大きい方をセット）、SLT（第1ソースレジスタがより小ならば1をセット）、データ転送命令であるLW、SW、条件分岐命令としてBEQ、無条件分岐命令（JUMP）、及び停止命令（HALT）の全11命令である。

† 立命館大学大学院理工学研究科

3　プロセッサの設計

3.1　単一サイクル

単一サイクル方式のプロセッサは、1つの命令を1つのクロックサイクルで実行完了しようとするものである。クロックサイクルはマシン中の最長パスであるロード命令で決定される。短いクロックサイクルで処理可能な命令も最長パスにあわせなければならない。図2に単一サイクルプロセッサのデータパスを示す。

我々が一番悩んだ点は、プロセッサとメモリの関係である。メモリの使用方法がよく分からなかったので、命令メモリとデータメモリは16ビット×32本のレジスタファイルとして定義した。

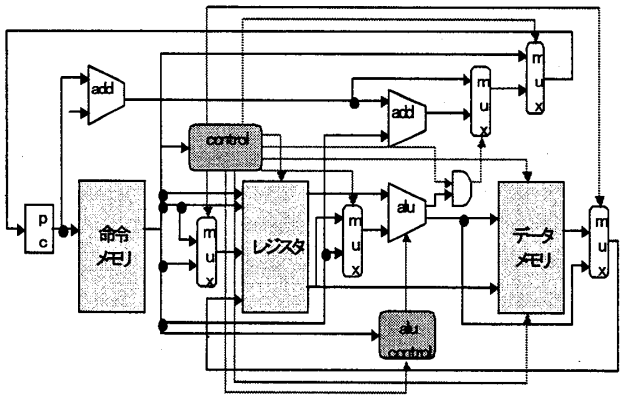

図2：単一サイクルプロセッサのデータパス

3.2　マルチサイクル

マルチサイクル方式のプロセッサは、1つの命令を複数のクロックサイクルに分けて実行する。単一サイクルに比べ、クロックサイクルを短くすることが可能であり、命令完了には3クロックから5クロックを要する。図3にマルチサイクルプロセッサのデータパスを示す。単一サイクル同様にメモリは16ビット×32本のレジスタファイルとして定義した。マルチサイクルは複数のクロックにわたって命令を実行するため、単一サイクルにはない一時レジスタ(命令レジスタ、ALUレジスタ)が必要になる。命令レジスタは現在実行中の命令を保持し、ALUレジスタはALUでの演算結果を次のクロックまで保持する。また、制御ユニットは命令フェッチ、命令デコードとレジスタフェッチ、

演算の実行、メモリアクセス、及びメモリ読み出しの各ステップの状態を、実行中の命令にあわせて保持するステートマシン記述をしているため、多少複雑になっている。設計に最も時間を要したのも制御ユニットである。

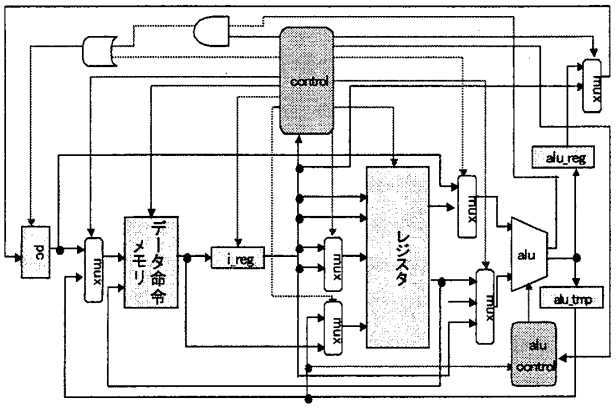

図 3：マルチサイクルプロセッサのデータパス

3.3 パイプライン

設計したパイプラインプロセッサは命令フェッチ（IF）、命令解読（ID）、命令実行（EX）、メモリアクセス（MEM）、ライトバック（WB）の 5 段パイプラインである。図 4 にパイプラインプロセッサのデータパスを示す。命令メモリとデータメモリは 16 ビット×32 本のレジスタファイルとして定義した。単一サイクルプロセッサのデータパスを基に、パイプラインレジスタ、フォワーディングユニット、ハザード検出ユニット等を付加することによりパイプライン処理を実現した。フォワーディングユニットはデータのフォワーディングを行うかどうかの判定を行いALU 出力を制御する。また、ハザード検出ユニットはハザードを検出しパイプラインをストールさせる機能を持つ。その他、分岐判定専用ユニットを設け、分岐判定を元のEX ステージから ID ステージに前倒しすることにより、分岐によるオーバーヘッドを軽減した。

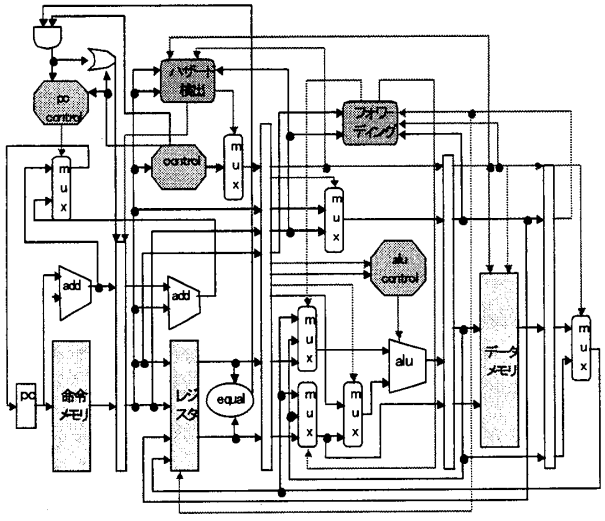

図 4：パイプラインプロセッサのデータパス

フォワーディングユニットにおいて、注意した点はデータの依存関係の判断とフォワーディングを行う際のタイミングである。正しいデータがフォワーディングによって書き変わってしまうことがあり、その制御に苦労した。また、分岐ハザード検出ユニットを設計する際、ハザードを正しく検出することに大変苦労した。正しくハザードが検出されるまでに様々な条件を付け加えていかなければならず、最も時間を要した。

4 HDL による設計と検証

4.1 HDL による設計

プロセッサの設計は Verilog-HDL を用いた。仕様の作成から HDL による記述、動作レベルシミュレーション、論理合成を経て、RTL シミュレーション、及びゲートレベルシミュレーションを行うという、トップダウン設計を行った。これにより、動作シミュレーションにおいて、早期に機能モジュールのバグを取り除くことができた。ここで、3 つの方式によるプロセッサの HDL のプログラムの規模とハードウェア規模（フリップフロップと LUT）、及び開発期間を表 1 に示す。パイプラインは単一サイクルのパイプライン化を行ったため、開発期間の多少の短縮は図れたが、制御が複雑でハードウェア規模が大きく、プログラムの行数も多いため、単一サイクルやマルチサイクルの 2 倍の期間を要した。

表 1：プログラムの規模と開発期間

	単一サイクル	マルチサイクル	パイプライン
HDL 行数	402	562	892
FF 数	1157	1223	1337
LUT 数	3228	2393	3549
開発期間	1 人月	1 人月	2 人月

4.2 シミュレーション

MTI 社の ModelSimXE を用いてゲートレベルシミュレーションを行った。シミュレーションに用いるテストパターンとして、1 から N までの総和（N = 100、静的命令数：10、動的命令数：407）、N 個の要素の中の最大値（N = 5、静的命令数：16、動的命令数：38）、ユークリッド互除法による 2 数（152、36）の最大公約数（静的命令数：15、動的命令数：70）の 3 種類を用意した。それぞれのテストパターンにおいて、ゲートレベルシミュレーションでの正確な値が出力されたので、設計した各プロセッサは、MONI 命令セットにより正しく動作していることが確認できた。

5 おわりに

単一サイクル、マルチサイクル、パイプラインの 3 方式で単純な 16bitRISC プロセッサを設計した。今回、メモリをレジスタファイルとして定義したので、我々が設計したプロセッサは限られたプログラムを実行できる機能モジュールに過ぎないが、当初の目的は達成することができた。今後、MONI 命令セットの更なる改良や記憶素子の再検討、さらには体系的に動作するプロセッサの設計が課題となる。

参考文献

[1]JohnL.Hennessy,David A.Patterson 著,成田光章訳:コンピュータの構成と設計(上)(下),日経 BP 社,1999.
[2]小林優:入門 VerilogHDL 設計入門,CQ 出版社,2001.