C-8 HDLによるRISCプロセッサの設計経験（II）

- 性能評価と考察 -
Design Experiences of a RISC Processor with HDL
- Performance Evaluation and Discussions -

池田修久† 大八木睦† 山崎勝弘†
Nobuhisa Ikeda Mutsumi Oyagi Katsuhiro Yamazaki

1. はじめに

本稿では、設計したプロセッサにテストプログラムを与え、シミュレーションにより得られた結果の評価を行う。ここではXilinx社のFPGAチップ（Vertex2 xc2v80000）を対象として、配置配線後のシミュレーションを行った。また、命令セットの改良点とメモリの扱い方について述べる。

2. 性能評価

シミュレーション結果を表1、表2に示す。

表1：各プロセッサの最短クロックサイクル

単位：ns

単一サイクル	マルチサイクル	パイプライン
130	54	30

表2：8通りのテストパターンの実行時間

単位：ns

	和	最大値	最大公約数
単一サイクル	52910	4940	9100
マルチサイクル	77058	7992	13230
パイプライン	15360	1770	4050

表1は各プロセッサの最短クロックサイクルである。記憶素子からの読み出し、書き込みを考慮した上で、これ以上クロックサイクルを短くすると、データ遅延により正しい値を得ることができなくなるので、表1で示す値を最短クロックサイクルとした。

表2は最短クロックサイクルを用いてテストプログラムを与えた場合の実行時間である。これにより、パイプラインが最も速く実行完了しており、単一サイクルをパイプライン化することでの高速化が確認できる。また、マルチサイクルの実行時間がどのテストパターンにおいても最大となっている。これは、マルチサイクルの最短クロックサイクルが単一サイクルの0.4倍程度に止まったことと、1命令完了までに3〜5クロック必要とするからである。

† 立命館大学大学院理工学研究科

表3：総クロックサイクル、CPI、ストール回数

書式：クロックサイクル数（CPI）

	和	最大値	最大公約数
単一サイクル	407(1)	38(1)	70(1)
マルチサイクル	1427(3.5)	148(3.5)	245(3.5)
パイプライン	512(1.3)	59(1.6)	135(1.9)
ストール回数	1	11	46

表3に総クロックサイクル数、CPI、パイプラインにおけるストール発生回数を示す。パイプラインのCPIには多少のばらつきがある。これはパイプラインストールの発生回数に依存している。ストール回数が1と少ない「和」ではCPIは1.3であるのに対し、ストール回数が46回と多い「最大公約数」ではCPIは1.9となっている。テストプログラムを作成するにあたり、レジスタ間の依存関係を考慮することにより、ストールの発生回数を減少させることができるが、今回はそのような最適化は行っていない。

3. 考察

単一サイクルのクロックサイクルと速度を1としたときの3つのプロセッサの性能比を表4に示す。

表4：各プロセッサの性能比

	最短クロックサイクル	速度向上
単一サイクル	1	1
マルチサイクル	0.42	0.6
パイプライン	0.23	3.4

命令実行時間は3つのテストパターンにおいてマルチサイクルが一番長くなっている。この理由として、命令セットの簡易性が関係していると考えられる。現在、命令語長が16ビットで命令数が11と限られているためクロック毎に変化する、各モジュールへの制御が容易である。命令数が増え、制御が複雑になると、異なる結果になると予想できる。また、マルチサイクルの命令フェッチステップでは、クロックの立ち上がりで制御ユニットから制御線が出力され、その出力に依存してメモリから実行する命令が読み出される。そして、同じクロックの立下りでその命令が命令レジスタに書き込まれる。これにより、制御ユニットの出力安定遅延と、メモリの読み出し遅延がマルチサイクルの最短クロックサイクルに大きく影響を与えていると考えられる。

一方、単一サイクルをパイプライン化することで最短クロックサイクル 0.23 倍に短縮でき、最大 3.4 倍の速度向上が得られており、パイプライン処理による高速化が確認できる。

4. 命令セットの改良

現在の命令セットは分岐命令、即値演算命令等において不十分であり、改良すべき点が多い。命令長、フォーマット、3 オペランド方式という条件は変えることなく、命令の追加を行う。

まず、分岐命令、即値命令、レジスタ間演算命令を強化することを考えた。分岐命令は BEQ と JUMP の 2 種類だけであるため、作成したテストプログラムの分岐処理は非常に分かりにくい。また、即値命令を用意していなかったため、ループを実現する際のループカウンタの更新処理等も分かりにくい。今後、サブルーチン呼び出しが必要になることを考えて JAL（Jump and Link）、JR（Jump Register）命令を付加した。レジスタ間演算命令においては 3 ビットで表現できる８つの命令は何が最適かということを検討し、BIG（２つの大小を比較し、大きいほうを格納）をなくし、論理シフト（SR、SL）、排他的論理和（XOR）を付加して全８命令とした。以下に改良後の命令セットを示す。追加した命令には下線を付ける。

- レジスタ間演算命令
  ADD, SUB, AND, OR, XOR, SL, SR, SLT
- 即値演算命令
  ADDI, SUBI
- 転送命令
  LD, ST, LDI
- 条件分岐命令
  BEQ, BNE, BP, BN
- 無条件分岐命令
  JUMP, JAL, JR
- その他の命令
  HALT

JAL、JR を定義したので汎用レジスタのうち、2 つをそれぞれスタックポインタ、戻り先 PC の専用レジスタとして割り当てた。また、レジスタ０は常に０であると決めており、ユーザが自由に使用できる汎用レジスタは 5 つである。これにより命令セットの直行性は多少犠牲になるが、重要な処理であるのでこちらを優先した。

今回の命令セットでは入出力や割り込みに対する命令は用意していない。汎用プロセッサを目指すのであれば、16 ビット固定命令長、3 フォーマット、3 オペランド方式という条件のもとでは実現できる命令数が少なく、良い命令セットを構築するには限界があるが、今回の実験では、この条件のもと、全 21 命令の最低限の命令を選ぶことができたと考えている。新しい命令セットで 3 つのテストプログラムを記述すると、全体として 20%程行数が短くなり、簡潔で読みやすいプログラムとなった(和:10→8、最大値:16→11、公約数:15→14)。

5. メモリとレジスタ

我々が設計した 3 通りの実行方式によるプロセッサは、現段階では本当の意味でプロセッサと呼べるものではなく、命令を処理することのできる機能モジュールに過ぎない。その理由は、プロセッサとメモリの関係である。今回の実験では、外部メモリとして定義すべき命令メモリとデータメモリを、FPGA 内の小規模なレジスタとして定義することで設計、シミュレーションを進めた（図 1(a)）。本来は、図１(b)の様に命令メモリとデータメモリを FPGA の外部に配置すべきであった。これは、プロセッサを HDL で記述する際に、メモリの階層記述を実現できなかったためである。図 1(b)の様に修正し階層表現を実現することで、シミュレーション時に想定した FPGA チップ自体をより小さいものにすることができ、ハードウェア規模が小さくなる。また、FPGA 内のクリティカルパスの短縮につながり、結果的に最短クロックサイクルもより短くなると考えられる。そのためにはシミュレーションにおける HDL 記述の理解を深め、CAD ツールを使いこなす必要がある。

また、FPGA の高集積化にともない、Xilinx 社では RAM を含むデバイスが提供されているので、メモリを別に設計し、配置配線の段階で演算モジュールとマージして、1 つのプロセッサとする方法も考えられる（図 1(c)）。この方法を用いると、小規模ではあるがメモリを含めたプロセッサ全体を１つの FPGA 内に実現することができる。

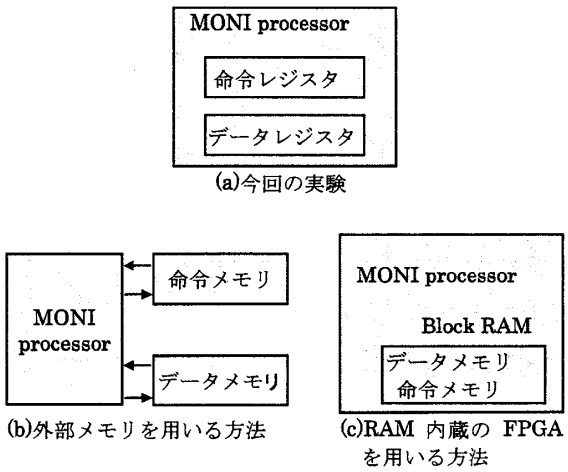

図 1：FPGA とメモリの関係

6. おわりに

HDL を用いて、3 通りの実行方式によるプロセッサを設計･検証して、コンピュータアーキテクチャと HDL による一連の設計･検証手順を理解することができた。今後の課題として、改良後の命令セットを対象として、RAM 内蔵の FPGA または外部メモリを用いたプロセッサの設計と検証が必要である。

参考文献

[1] John L. 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] 桜井至:HDL によるデジタル設計の基礎,テクノプレス,1998.