平成 29 年度　計算機科学実験及演習 3A
(3 回生前期学生実験　HW 中間報告)

アーキテクチャ検討報告書

提出期限：5 月 11 日
提出日：5 月 11 日

第 22 班

1029277526　白石　竜也

1029272870　谷　勇輝

目次

1 要求仕様、設計目標、方針、特長

1.1 要求仕様

SIMPLE アーキテクチャをベースとした独自のプロセッサを設計する。SIMPLE/B 設計に変更、拡張を加えた独自のハードウェア設計を用い、アーキテクチャの拡張、改良も行うことにより SIMPLE/B 設計よりも高機能・高速な設計を実現する。

1.1.1 命令セット仕様

1. SIMPLE アーキテクチャが提供する全ての基本命令セット
2. 即値 ADD 命令 ADDI
3. 関数呼出命令 JAL と復帰命令 JR

1.1.2 ハードウェア設計仕様

1. ５段パイプライン
   - データハザード対処
   - 分岐ハザード対処
   - 静的分岐予測
   - フォワーディング実装

2. ハーバード・アーキテクチャ

1.1.3 補助モジュール仕様

- 表示・入力用モジュール
  1. 外部出力の表示（全 7segLED、LED、ブザーの使用可）
  2. 外部入力の受付（全入力方法の利用可）
  3. クロックサイクル数のカウント機能

- （おまけ）簡易アセンブラソフト
  1. 命令語から機械語への翻訳機能
  2. メモリ初期化用.mif ファイルの自動生成機能

1.2 設計目標

目標最大動作周波数　　70MHz 以上
＜ソート速度コンテスト＞
　目標サイクル数　　300,000 以下
　目標時間　　5.000ms 以下

1.3 設計方針

1.3.1 基本の方針

５段パイプラインの各フェーズを以下のように定める。

- IF（命令フェッチ）
- ID（命令デコード）
- EX（演算）
- MA（メモリ操作）
- WB（レジスタ書込み）

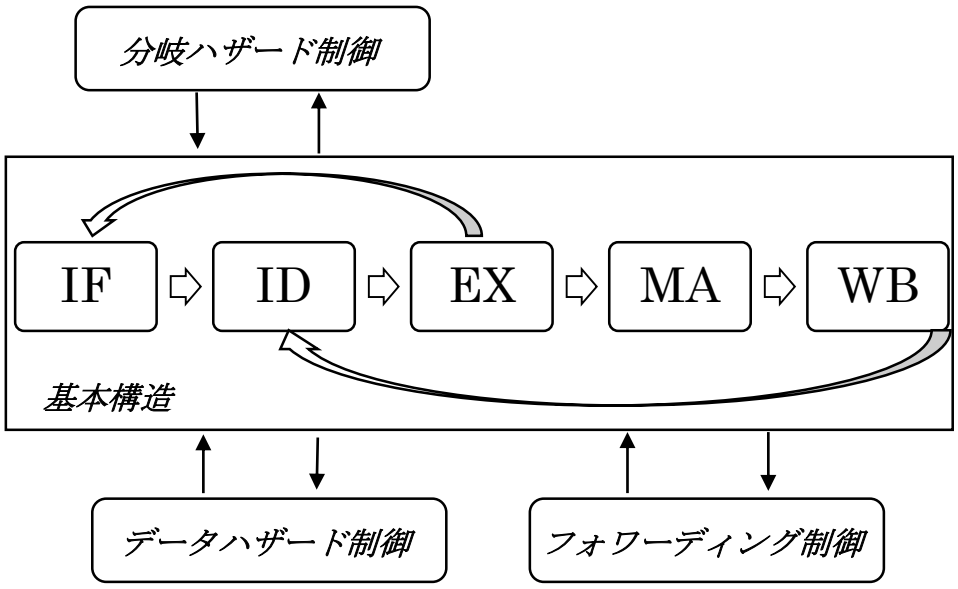

図 1　基本の設計方針

それぞれのフェーズを個別のモジュールに担当させ、計５個のモジュールが連結された回路を基本構造とした。個々の命令の処理は原則としてモジュ

ール内部で完結させ、それを並列処理させ５段パイプラインとして動かすための制御部はモジュール外部に別途追加モジュールを設置して行う方針とした。(図 1)

各フェーズ間でデータを保持するレジスタは、前のフェーズを担当しているモジュールがアウトプットとしてモジュール内部に保持する設計とした。従って外部から見れば各モジュールは、入力信号に応じて同期的に値を変更する巨大なレジスタと見なすことができる。

1.3.2 制御の方針

命令個別の挙動を実現するための基本制御信号は **ID 内部で生成**し、それ以降のフェーズで使用する方針とした。以下表１に、生成する 9 bit の基本制御信号とそれが使用され制御が行われるフェーズを示す。

表 1　基本制御信号と制御を担当するフェーズ

信号名	用途	使用フェーズ
RegDst	レジスタ書き込み用アドレスの選択	ID
ALUOp1	演算用制御信号の生成	ID
ALUOp0		
ALUSrc	演算に即値を使用するかの選択	EX
Branch	分岐を行うかの選択	EX
MemRead	メモリ読み出しを行うかの選択	MA
MemWrite	メモリ書き込みを行うかの選択	MA
RegWrite	レジスタ書き込みを行うかの選択	WB
MemtoReg	演算結果かメモリ読み出し結果かの選択	WB

1.3.3 メモリ・レジスタの方針

メモリの操作による動作速度の低下を抑えるため、命令が記憶されるメモリとデータが記憶されるメモリを切り離し独立動作させる**ハーバード・アーキテクチャ**を採用した。命令を記憶するメモリは IF モジュールが、データを記憶するメモリは MA モジュールがそれぞれ保持し、管理、操作を行う。

レジスタは ID モジュールが保持する。WB フェーズでもレジスタ操作を

行うが、そのクロック管理は ID モジュールに委託する方針とした。すなわち、WB モジュールには見かけ上 clock が入力されない。

なお、データハザードの対応範囲を狭めるため、WB フェーズのレジスタ書き込みはクロックステップの前半に終了させ、ID フェーズのレジスタ読み出しは後半に行うこととした。

1.3.4 分岐処理の方針

分岐判断は EX モジュールにて行う。SIMPLE/B 設計では分岐アドレスは最終フェーズから PC に送られるが、今回の設計では分岐が確定した時点で EX モジュールから IF モジュールに通告する方針にした。これは、分岐ハザードによる損失を最小限に抑えるためである。

また、SIMPLE/B では ALU で分岐アドレスの計算を行っていたが、今回の設計ではアドレス計算に使用する加算器はデータの演算に使用する ALU とは別に、独立して用意することとした。

1.3.5 環境の構築方針

プロセッサの設計に当たって、プロセッサ本体だけでなくテスト・デモンストレーション用のモジュール設計も重視した。方針として、FPGA ボードの入出力をいつでも自由に使用できるようにし、スムーズな単体・結合テストが行えるよう環境を構築することとした。

また同様の目的で補助的にアセンブラソフトを作成し使用することにした。

1.4 特長

今回作成するプロセッサの特長は、①高速動作、②高実用性の 2 点である。

1.4.1 高速動作

今回の設計では、フォワーディングや分岐予測等、5 段パイプライン方式の基本的な機能は全て実装している。また設計の随所で高速化に気を回し、大小様々な工夫を盛り込むようにしている。結果としてスムーズで高速な動作が特長として期待できる。

1.4.2 高実用性

SIMPLE アーキテクチャからの命令の拡張として、即値 ADD 演算と関数機能を採用している。これらの命令によってより直感的で実用的なプログラミングが可能になる。また、ボードの入力と出力等、環境周辺も丁寧に構築し実用性を高めている。プロセッサが 1 つの装置として完成した状態に、高実用性という特長を期待できる。

2 高速化/並列処理の方式

2.1 基本設計の改良による高速化

2.1.1 ハーバード・アーキテクチャによる高速化

ハーバード・アーキテクチャは、メモリに同時アクセスするための構築である。メモリの読み書きは SIMPLE/B 設計の中でもボトルネックになっている箇所であるが、この設計のままではメモリに同時アクセスできず、パイプライン化した際に待ち時間が発生してしまう。プログラム用メモリを IF モジュール内に、データ用メモリ MA モジュール内に独立に用意して、メモリへの同時アクセスを実現することで高速化を行う。

2.1.2 分岐処理の繰上げによる高速化

1.3.4 項でも取り上げたように、分岐処理を EX フェーズまで繰り上げて設計している。これにより、分岐をより早い段階で行うことができ、分岐命令によるストール時間が削減される。

2.2 命令の拡張による高速化

2.2.1 即値 ADD 演算による高速化

SIMPLE アーキテクチャには即値演算が用意されていない。例えば任意の整数をあるレジスタに加えるためには、LI 命令で他のレジスタに整数を格納してから、ADD 命令で加算を行う必要がある。

あるレジスタに１を足す命令は、繰り返し構文の内部などで頻繁に現れる。２つの命令を１つにまとめることで高速化を目指す。

2.2　５段パイプライン化による高速化

今回の設計の高速化の主要要素は５段パイプライン化である。IF～WB フェーズを並列して動作させることにより大幅な高速化を実現する。

パイプラインの性能を最大限に発揮するためには、できるだけハザードによるストールを回避することが重要である。今回は以下の２つのシステムでストール回数削減を図る。

- フォワーディング
- 分岐予測

3 性能/コストの予測

過去のソート速度コンテストの類似する機能を持つプロセッサの結果を参考に、このプロセッサの性能を予測すると以下のようになる。

最大動作周波数　約 70MHz
＜ソート速度コンテスト＞
　サイクル数　　約 300,000
　時間　　約 5.000ms

以下は各改良方式ごとの性能向上/コスト増加の予測である。

3.1　基本設計の改良の性能/コスト予測

3.1.1 ハーバード・アーキテクチャの性能/コスト予測

仮に SIMPLE/B 設計のまま 5 パイプライン化したと仮定すると、IF(p1)フェーズと MA(p4)フェーズで順番にメモリアクセスが行われることになってしまう。その場合、１クロックサイクルに必要な時間はメモリアクセス２回分の時間となることが予想される。一方ハーバード・アーキテクチャを採用して２つのメモリを独立させていると、同時に２回分のメモリアクセスが可

能であるから、単純計算で 1/2 のクロックサイクル時間を実現できることが期待できる。

ハーバード・アーキテクチャにより、動作周波数は２倍近く高速になると予測できる。

コストの面を考えると、メモリの分割を行うことによって読み書きの機構などメモリの基本機構を２つ用意することになる。従って、素子数等コストはメモリの基本機構１つ分増加することが予測される。

3.1.2 分岐処理の繰上げの性能/コスト予測

EX フェーズへの分岐処理繰上げによって、１回の分岐ハザードによる発生バブル数が４から２に減少する。分岐命令数がプログラム全体の 5%であり、分岐ハザード以外のハザードを無視できると仮定すると、削減されるクロックサイクルの割合は

$$\frac{5 \times 4 - 5 \times 2}{100 + 5 \times 4} = \frac{1}{12} \cong 8.3\%$$

静的分岐予測が行われている条件下であっても、予測精度が 50%であったと仮定すれば、この半分の割合の削減を期待できる。

分岐処理の繰上げにより、プログラム実行の総クロックサイクルは約４～８%削減できると予測できる。

フェーズの繰上げによって EX/MA 間と MA/WB 間のレジスタが削減されるため、レジスタ 16 本分程度のコスト削減が見込める。

3.2 命令の拡張の性能/コスト予測

3.2.1 即値 ADD 演算の性能/コスト予測

値が増加するポインタを持つようなループ内の処理が全実行命令の 20%を占め、内部処理が平均して 15 命令であると仮定すると、削減されるクロックサイクル数は

$$\frac{15 - 14}{15} \times \frac{20}{100} = \frac{1}{75} \cong 1.3\%$$

即値 ADD 演算はその他にも多様な用途があるため、削減数はプログラム

の内容によって大きく変動する。全体としては、5%程度の速度改善を見込んでいる。

追加でかかるコストは、命令の増加に伴う制御部の拡大分となる。基本的にはセレクタの増加であるので、増加コストは全体の 1%未満に収まると予測できる。

3.2.2 その他の命令拡張による性能/コスト予測

関数命令の拡張では、同じ関数を使い回すことによるプログラム行数の削減は見込めるが、クロックサイクル数としては JAL 命令と JR 命令の処理の分だけ多くなってしまうことが見込まれる。しかし、実用的なアルゴリズムの実装等では必須といえるほどの機能であり、非機能要件の面では重要である。

関数命令の拡張によるコストの増加は、ADDI 命令の拡張の時と同じく命令の増加に伴う制御部の拡大分となる。基本的にはセレクタの増加であるので、増加コストは全体の 1%未満に収まると予測できる。

3.3 5 段パイプラインの性能/コスト予測

5 段パイプラインは、理想的には通常の SIMPLE/B と比べて 5 倍近い性能を発揮できる。しかし実際には各フェーズの処理時間の差やハザードによるストールによって性能は落ちてしまう。これらへの対処を行わない場合、パイプライン化による速度改善はほとんど見込めない。

フォワーディングは LD 命令の次の命令のデータハザードは回避できないが、その他のデータハザードは回避できる。(フォワーディングを行わない場合と比べて 50%近い改善が見込めると予測できる。)

分岐予測は分岐ハザードの発生確率を統計的に落とし、バブル発生を低減させる。分岐命令の実行回数は平均的に多く、プログラムの例等から考察して 20%近い改善が見込めると予測できる。

全体として、パイプライン化によって 100%−(50%×80%) = 60%程度のサイクル数削減が見込める。

パイプライン化によって増加するコストはデータハザード制御部、分岐ハザード制御部、フォワーディング制御部のコスト総計となる。内部的には論理演算による判定とセレクタによる選択になるので、それぞれ基本構造のフ

ェーズ１つ分程度のコストとなると予想した。従ってパイプライン化によるコスト増加は 3/5 ≒ 60%程度になると予測できる。

4 考察等

採用した仕様を決定した要因と経緯を示す。

今回の実験は、SIMPLE アーキテクチャが動作するプロセッサを作成し独自の拡張を行うことが課されているが、それに加えて、ソート速度コンテストで１桁ミリ秒以内の結果を出せるプロセッサを作成することを目標とすることにした。そこで重要となるのが、①高速で動作し、②柔軟に改良・使用ができる、という２つの性質を満たすことである。

①の対策として、まず、実装は重いが大幅な高速化が見込める**５段パイプライン方式**を改良の根底に据えることにした。ハーバードアーキテクチャの採用、各種ハザード対処、フォワーディング制御はこの時点でほぼ採用必須となる。また、分岐予測は簡単なものであれば自然に採用でき、効果も期待できるため仕様に追加することに決めた。

命令数の単純な減少という面から、即値 ADD 演算を加えることにした。その他の追加命令も魅力的ではあるが、工数の関係から最も使用頻度が高くなると予想できた ADDI 命令のみを仕様としては採用するに至った。時間が許すのであればその他の追加命令も検討してみたい。

次に②の観点から、関数呼び出し命令を追加実装することに決めた。柔軟な使用性を確保するためには最も重要な機能であると考えたためである。

このプロセッサの要であり、第 22 班の最大の目標は 5 段階パイプラインを動作させることである。その為には柔軟な改良とスムーズな開発を行う必要があるだろう。分かりやすくシンプルな基本設計を始め、豊富な入出力を提供するモジュールやプログラム作成用ソフト等、環境の充実にも力を入れ、本実験を進めていくことにした。

以上