情報工学実験 (ハードウェア実験) 報告書

PARTHENONによる32ビットマイクロプロセッサの作成

学生番号: 09425566
提出者: 佐藤 佑太

提出日: 2015 年 7 月 27 日
締切日: 2015 年 7 月 27 日

概 要

本稿では，情報工学実験（ハードウェア実験）において作成した、32 ビットマイクロプロセッサについてまとめる。

1 はじめに

本実験の目的は、ハードウェア記述言語と CAD ツールを利用したマイクロプロセッサの設計を通して、論理回路、コンピュータアーキテクチャ、およびコンピュータシステムに関する理解を深めることである。

本報告書では，ハードウェア記述言語と CAD ツールを用いた論理回路の設計と、32 ビットマイクロプロセッサの設計について報告する．

本報告書の構成は次のとおりである．まず **2** にて本実験で設計したプロセッサの概要について述べる．

3 にて本実験における実施内容の状況報告を行う。

4 では，アセンブリによって作成したプログラムの作成に関しての報告を行う。

5 では，作成したプロセッサの設計に関して報告を行う。

6 では，発展課題で取り組んだ課題について報告する。

7 では，諸事項について検討を行い、それについての考察を記述する。

8 では，設計の際に工夫した点や特に注力した点について報告する。

9 では，本実験を通して、どこれまでから一層理解が深まった部分などについて記述する。

10 では，本実験の進捗状況報告ファイルを掲載する。

11 では，本実験で作成したアセンブリ言語プログラム、設計した SFL 記述、テスト用スクリプト、テスト結果、論理合成時の出力等のファイルの一覧を掲載する。

最後に **12** で，本報告のまとめと今後の課題を述べる。

2　設計したプロセッサの概要

ここでは，設計したプロセッサの概要について述べる．

2.1　サポートする命令セット

まず、本実験でサポートする命令セットを以下の表に示す。

表 1: サポートする命令一覧

命令種別	サポートする命令	サポートしない命令
算術論理演算	add, addu, addi, addiu, sub, subu, and, andi, or, ori, xor, xori, nor	mult, multu, div, divu
比較	slt, sltu, slti, sltiu	
シフト	sll, srl, sra, sllv, srlv, srav	
ロードストア	lw, sw, lb, sb	
分岐	beq, bne	
ジャンプ	j, jr, jal, jalr	
データ転送	lui, mfhi, mflo	
例外，システムコール	syscall	

今回は掛け算、割り算の命令については時間が足りなかったため実装しないこととした。

3　実施状況の報告

ここでは，取り組んだ課題の実施状況について報告する。

3.1　報告内容に関する事項

私たちの班では、役割分担をせずにそれぞれがプログラムの作成を行った。そのため、各プログラムにおける個人の役割は 100 中全員がすべて 100 であると考えたため、表からは省略している。

表 2: 実施状況

設計課題番号	内容	実施
D-1	32 ビット加算器 add32 の設計	実施済み
D-2-1	カウンタの設計	実施済み
D-2-2	カウンタの設計	未実施
D-2-3	カウンタの設計	未実施
D-3	32 ビット ALU の設計	実施済み
D-4	32 ビットシフタの設計	実施済み
E-1	32 ビット Carry Lookahead Adder の設計	未実施
E-2	32 ビット整数乗算器の設計	未実施
E-3	32 ビット除算器の設計	未実施
5-1	5 ビット比較器	実施済み
5-2	レジスタファイル	実施済み
5-3	メモリユニット	実施済み
6-1	p32 プロセッサコア (マルチサイクル)	実施済み
E6-1	p32 プロセッサコアの改良	未実施
E6-2	乗算機能の実装	未実施
7-1	p32 プロセッサコア (マルチサイクル v2)	実施済み
E7-1	p32 プロセッサコアの改良	未実施
E7-2	乗算機能の実装 (2)	未実施
8-1	p32 プロセッサコア (パイプライン)	実施済み
E8-1	p32 プロセッサコアの改良	未実施

4 プログラミング課題に関する報告

ここでは，設計したプロセッサの概要について述べる．

4.1 報告内容に関する事項

実験レポートの報告内容に関する注意事項は次のとおりである．

1. 自分で文章を組み立てること．テキストや文献の文章を，一言一句をそのままコピー&ペーストしたのでは，モラル的に問題があるばかりか，自分のためにもまったく意味がない．

5　プロセッサ設計課題に関する報告

ここでは，設計したプロセッサの概要について述べる．

5.1　報告内容に関する事項

実験レポートの報告内容に関する注意事項は次のとおりである．

1. 自分で文章を組み立てること．テキストや文献の文章を，一言一句をそのままコピー&ペーストしたのでは，モラル的に問題があるばかりか，自分のためにもまったく意味がない．

6　追加課題や発展課題に関する報告

ここでは，設計したプロセッサの概要について述べる．

6.1　報告内容に関する事項

実験レポートの報告内容に関する注意事項は次のとおりである．

1. 自分で文章を組み立てること．テキストや文献の文章を，一言一句をそのままコピー&ペーストしたのでは，モラル的に問題があるばかりか，自分のためにもまったく意味がない．

7　検討・考察

ここでは，設計したプロセッサの概要について述べる．

7.1　報告内容に関する事項

実験レポートの報告内容に関する注意事項は次のとおりである．

1. 自分で文章を組み立てること．テキストや文献の文章を，一言一句をそのままコピー&ペーストしたのでは，モラル的に問題があるばかりか，自分のためにもまったく意味がない．

8　工夫した点や特に力を注いだ点

ここでは，設計したプロセッサの概要について述べる．

8.1　報告内容に関する事項

実験レポートの報告内容に関する注意事項は次のとおりである．

1. 自分で文章を組み立てること．テキストや文献の文章を，一言一句をそのままコピー&ペーストしたのでは，モラル的に問題があるばかりか，自分のためにもまったく意味がない．

9　本実験を実施して得られたこと

ここでは，設計したプロセッサの概要について述べる．

9.1　報告内容に関する事項

実験レポートの報告内容に関する注意事項は次のとおりである．

1. 自分で文章を組み立てること．テキストや文献の文章を，一言一句をそのままコピー&ペーストしたのでは，モラル的に問題があるばかりか，自分のためにもまったく意味がない．

10　進捗状況報告

ここでは，設計したプロセッサの概要について述べる．

10.1　報告内容に関する事項

実験レポートの報告内容に関する注意事項は次のとおりである．

1. 自分で文章を組み立てること．テキストや文献の文章を，一言一句をそのままコピー&ペーストしたのでは，モラル的に問題があるばかりか，自分のためにもまったく意味がない．

11　作成した設計記述、プログラム等のリポジトリ名、ファイル名の一覧

ここでは，設計したプロセッサの概要について述べる．

11.1　報告内容に関する事項

実験レポートの報告内容に関する注意事項は次のとおりである．

1. 自分で文章を組み立てること．テキストや文献の文章を，一言一句をそのままコピー&ペーストしたのでは，モラル的に問題があるばかりか，自分のためにもまったく意味がない．

12　おわりに

本報告書では，.... .... について報告した．本実験テーマの目的である (1) ....，(2) ... ··· (n) ... は，それぞれ達成できた．また，.... .... により， .... であることがわかった．今後の課題としては .... がある．

参考文献

[1] 著者名，文献のタイトル，(もしあればページ，)，出版社，出版年．

[2] 渡邊誠也，ハードウェア記述言語を用いたマイクロプロセッサの設計，情報工学実験テキスト，岡山大学工学部情報工学科 (2003)

付録

報告書本体部分に掲載するとわかりにくくなるものは，付録に掲載する．例えば，出力結果全体やSFL 記述のリストなどである．本文中に掲載したほうが，分かりやすい場合は本文中に掲載すべきであり，その場合でも掲載は必要最小限になるように努めるべきである．

A　4ビット ALU の SFL 記述

```
1  /* (alu4.sfl) */
2  %i ‘‘add4.h’’
3  %i ‘‘alu4_func.def’’
4
5  module alu4 {
6      input  a<4>, b<4>;  /* input date */
7      input  func<3>;     /* function */
8      output out<4>;      /* output data */
9      instrin enable;
10
11     add4 adder;
12
13     instruct enable alt {
14         func == THAFUNC: out = a;
15         func == THBFUNC: out = b;
16         func == ANDFUNC: out = a & b;
17         func == ORFUNC:  out = a | b;
18         func == XORFUNC: out = a @ b;
19         func == NOTFUNC: out = ^a;
20         func == ADDFUNC: out = adder.enable(a, b, 0b0).sum;
21         func == SUBFUNC: out = adder.enable(a, ^b, 0b1).sum;
22     }
23 }
24 /* End of file (alu4.sfl) */
```

図 1: 4 ビット ALU の SFL 記述

B　表の例

表の例を表 3 と表 4 に示す[1]．

表を参照する際には，単に表を掲載するだけではなく,「表 3 に論理合成で得られた諸量をまとめた結果を示す」といった文章にて，参照している「表」が何を示しているのかを説明した後に，その詳細に関する説明が必要である．

[1]表の見出し（表題）は，表の上に置く．一方，図の見出しは図の下に置く．

表 3: 論理合成で得られた諸量のまとめ

モジュール	最大遅延 (ns)	最大動作周波数 (MHz)	ゲート数	実装面積 ($1000\mu m^2$)	消費電力 ($\mu W/MHz$)	最大動作周波数で動作時の消費電力
4 ビット加算器						
4 ビット桁上げ先見加算器						
8 ビット桁上げ先見加算器						
16 ビット桁上げ先見加算器						
4 ビット ALU						
8 ビットシフタ						
4 ビットカウンタ						
10 進カウンタ						
4 ビットアップダウンカウンタ						

表 4: マイクロプロセッサ p16 の設計状況

	モジュール		設計状況
1.	16 ビット加算器	`add16`	(0) 未着手
2.	16 ビット実行ユニット	`exec16`	(1) 設計中
3.	レジスタファイル	`reg16x8`	(2) 動作確認済み
4.	増加器	`inc16`	(2) 動作確認済み
5.	メモリユニット	`memunit`	(2) 動作確認済み
6.	p16 トップモジュール	`p16`	(3) 設計完了
7.	p16 制御ユニット	`p16_controller`	(3) 設計完了

C　画像の取り込み

画像の取り込みの例を図 2 に示す．図 2 に示す画像は，画面に表示されるウィンドウを取り込んで，EPS ファイルとして保存したものを TeX に貼りつけたものである．画像の取り込み方法は C.1 にて具体的に説明する．

C.1　画像の取り込み方法

以下では，画面に表示されている画像（ウィンドウ）を EPS として保存する方法を説明する．

1. 取り込みたい図を画面に表示させ，ターミナルから `gimp` とタイプし，gimp を起動させる．
2. 「The GIMP」というタイトルのついたウィンドウの「ファイル」メニューから「取り込み」「画面取り込み...」を選択する．
3. 「画面取り込み」といタイトルのついたウィンドウが表示されるので，そのウィンドウにて適切に設定を行なった後に「了解」ボタンをクリックする．
4. カーソルが十字にかわるので，取り込みたいウィンドウの上でクリックする．すると，取り込まれたウィンドウの画像が表示される．
5. 取り込まれた画像ウィンドウの上にカーソルをあわせ，右クリックを押し,「ファイル」メニューの「別名で保存」を選択する．

6. 保存先とファイル名を指定して,「了解」ボタンをクリックする．ファイル名の最後を.eps としておくと，自動的に EPS ファイルとして保存される．

7. 保存された EPS ファイルを報告書の TeX ソースから読み込むことで，報告書に画像（画面）を張り付けることができる．

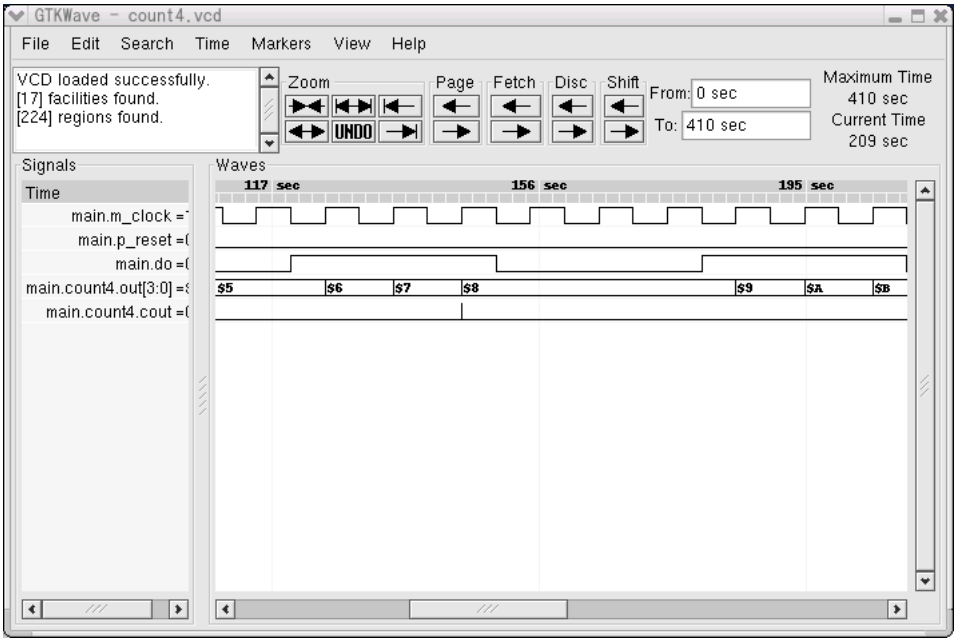

図 2: 4 ビットカウンタをシミュレーションした際の信号波形