情報工学実験 A（ハードウェア）報告書（の書き方）

（ここに適切な報告書のタイトルを記入のこと）

学生番号: 094305xx
提出者: 渡邊 誠也
E-mail: nobuya@cs.okayama-u.ac.jp

提出日: 2020 年 6 月 xx 日（？）
締切日: 2020 年 6 月 xx 日（金）

概 要

（ここに報告書の概要を簡潔にまとめる．本文中にある文章をピックアップしつなげてまとめることで作成する．特に「おわりに」にある文章が活用できる場合が多い.）

本稿では，情報工学実験 A（ハードウェア）プロセッサ設計実験の報告書の書き方について説明する．報告書執筆における基本的な注意点について説明する．さらに，報告書の作成と提出について簡単に述べる．この文書の LaTeX ソースを読むことで，LaTeX を使った報告書の執筆についてスキルを得ることを期待する．

1 はじめに

報告書の最初の節（ここでは「**はじめに**」というタイトルとしている）では，実験の背景と目的，および，本報告書の構成について述べる．報告書の構成は，各自の判断で分かりやすい構成にすればよい．このサンプルでは，**1** 〜 **4** 節の構成をとっているが，報告事項によって適宜変更する．各節では小節（subsection）を設けるなどして，わかり易い構成をとって欲しい．

以下に，かなり省略してはいるが,「**はじめに**」の記述例を示す．節（section）や小節（subsection）等の参照には，`\ref{ラベル名}`といった記述を用いる．ラベルの定義は，本ドキュメントのソースの見ればわかるだろう．

【記述例】

本実験の目的は，.... .... である．本報告書では，.... について報告する．情報工学実験テキスト [2] の第 1 章から第 3 章の.... .... を報告する．

本報告書の構成は次のとおりである．まず **2** にて.... .... について述べる．**3** では，.... .... する．最後に **4** で，本報告のまとめと.... .... を述べる．

2 報告書執筆における注意点

ここでは，執筆上の注意点について述べる．

2.1 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．テキストや参考にした文献やウェブページの文章を，一言一句をそのままコピー&ペーストしたのでは，著作権の問題（盗作）があるばかりか，自分のためにもまったく意味がない．参考にした文献で述べられていることを文中で引用する場合は，末尾の参考文献にその文献を挙げ，本文中では文献番号を付加し，他の文献からの引用であることを明確にすること．例えば，

   高速化を実現するために，イソテル社のプロセッサでもこの方式を採用している [3].

   というように，参考文献を引用する．

   自分自身が記述してある内容を**よく理解した上で自分の言葉で表現する**ように努めて欲しい．その際，事実を述べているのか，自分の意見や考えなのか，他人の考えや意見なのかを明確に区別して書くように心がけて欲しい．

   ただし，ここで言う「自分の言葉」というのは，日常，会話をする際に使っている言葉ではなく，報告書の文章として適切な言葉のことである．**口語体の文章は，報告書に用いる文章としては不適切**である点に注意して欲しい．

2. 他人が読んで理解できるようにすること.「報告書を読むのは担当教員だけだろうから，このことは担当教員も知っているだろうからこの部分だけ述べることにしよう」といった考えで報告書を書かないこと．第三者が報告書のみで内容を把握できるように書くように努めて欲しい．例えば,「課題 x を行なった」とある場合,「課題 x」はどういった内容なのかを**簡潔にまとめて**報告書に記載しておくべきである．

3. 結果報告だけの内容にならないこと．得られた結果に対しては，その結果が妥当な結果なのかを考察するべきである．また自分で見い出した問題点や，その問題点を解決するまでに考えたこと等を述べるべきである．

   例えば,「課題 x を行ない，正しく動いた.」といった記述だけでは，不十分である．上の 2. とも関連するが,「課題 x」は何なのか,「正しい」とは何をもって正しいと言えるのかをきちんと正確に記述する必要がある．

   また,「課題 y では，図 z に示す FSL 記述を作成した．(おわり)」といった記述も不十分である．課題の意図を理解し，その FSL 記述を作成するまでに考えたこと，FSL 記述の説明等を行なうべきである．逆に，やみくもにすべてを説明していたのでは，焦点が定まらず分かりにくくなることがあるので注意が必要である．その図や説明で主張したいことを伝えることができるように工夫して欲しい．

2.2 体裁に関すること

その他の注意事項（体裁に関すること）としては，次の事項が挙げられる．

1. 「〜です.」「〜します.」といった **ですます調** で書かない．

2. 句読点は,「.」「,」を用いる[1]．`~/.emacs` ファイルに次の式を記述をしておけば，デフォルトの句読点が「.」および「,」となる．

```
;; 2バイト文字の句読点設定
(setq its-hira-period ". ") ;;  「.」は全角
(setq its-hira-comma  ", ") ;;  「,」は全角
```

[1]例えば，情報処理学会，電子情報通信学会の和文の論文誌では，句読点は「.」と「,」を用いている．

```
\begin{verbatim}
・ 最初の項目
・ 次の項目
・ ３番目の項目
・ 最後の項目
\end{verbatim}
```

図 1: verbatim 環境による列挙の記述例

上記設定をしていない場合に「.」および「,」をタイプする場合，日本語モードでそれぞれ z-.（大文字の Z をタイプ後，ピリオド (.) をタイプ），z-,（大文字の Z をタイプ後，カンマ (,) をタイプ）を入力する．

3. TEX の verbatim 環境の乱用は避ける．しばしば，報告書全体を verbatim 環境にて書いている学生が見受けられる．そのような書き方では TEX で美しい文書が作成できる機能を十分に利用できていないことになる（すなわち，レポート点の減点になりうる）．また，次以降に挙げる箇条書きやフォントの切替え等もできない．

4. 箇条書きや列挙を活用する．実験報告書は基本的に，文章主体で書くべきものであるが，文章だけでだらだらと書いていたのでは分かりにくい場合もある．箇条書き（itemize）や列挙（enumerate）等を用いることでわかり易くなる場合には積極的に利用することを推奨する．

5. フォントを適切に選択する．例えば，強調したい単語は，**ゴシック体**や **bold 体**にするとか，プログラムリストや結果出力は，タイプライタフェイス (typewriter face) のフォントで書くといったようにフォントを適切に使用するとわかり易い．プログラムコードなどには，タイプライタフェイスのフォント（this uses typewriter face fonts）を用いる．

   数式は，LATEX の数式モードを利用することで，適切な数式用のフォントを使う．このサンプルでは，適切なフォントを利用している．

6. 図や表には，図番号あるいは表番号とタイトルをつける．また図や表を掲載する場合には，本文中でそれらの図番号あるいは表番号を参照して説明をすること．本文にて言及されない図や表は掲載しない．

7. 長いソースリストや結果の記載は，付録を活用する．(なお，本当にそのソースリストや結果の掲載が必要なのかよく考えること)．その際，図や表と同様，本文中から参照して簡単に説明すること．

8. 不要な改行や，改ページを行わないこと．パラグラフ（段落）は，TeX のソースで空行を入れると適切に，別の段落として整形される．本文中では，強制改行の \\ を使う必要なほとんどないはずである．(表などでは必要である)．

ここでは，例として itemize 環境の使い方を知らない人がよく書く verbatim 環境を用いた列挙の記述例を図 1 に示す． 図 1 に示すコードからは，LATEX により次のように整形される．

・ 最初の項目
・ 次の項目
・ ３番目の項目
・ 最後の項目

図 1 に示した記述は，LATEX では図 2 に示す記述のように itemize 環境を用いた記述にするべきである．図 2 に示すソースコードからは，LATEX により次のように整形される．

```
\begin{itemize}
  \item 最初の項目
  \item 次の項目
  \item ３番目の項目
  \item 最後の項目
\end{itemize}
```

図 2: itemize 環境による列挙の記述例

- 最初の項目
- 次の項目
- ３番目の項目
- 最後の項目

各項目に番号を振りたい場合，itemize 環境の代わりに enumerate 環境を利用するとよい．図 1 に示した記述を enumerate 環境で記述した際には次のように整形される．

1. 最初の項目
2. 次の項目
3. ３番目の項目
4. 最後の項目

最初から完全な報告書は要求しないが，徐々に改善していくように努力してもらいたい．4 年生で研究室配属されると所属研究室の教員から特別研究の指導を受け，最終的に特別研究報告書にまとめる際に事細かい指導を受けることになるが，日頃のレポートや報告書作成においても自分自身でよりよい報告書となるように取り組んでいくことを推奨する．

なお，本ドキュメントの LaTeX ソースを公開するので，報告書を作成する際の参考にして頂きたい．この文書自体にはあらわれていないが，ソース内にコメント等を追加してあるので参照されたい．

3　報告書の作成と提出

LaTeX で作成した実験報告書は，PDF ファイルに変換して所定の場所へ提出する．提出方法等に関しては，別途，指示があるのでそちらにしたがうこと．

3.1　報告書の作成

報告書は，実験用サーバの LaTeX あるいは，自宅 PC の環境で作成する．報告内容や執筆上の注意点は，**2** で述べたとおりである．エディタ（Emacs 等）で報告書ファイルを記述し，LaTeX で整形する．LaTeX の使い方については，情報工学実験の範囲ではないのでここでは説明しない．適宜，プログラミング演習で習ったことを復習しておくこと．

DVI ファイルを PDF ファイルへ変換する際には，dvipdfmx コマンドを用いる．LaTeX で生成した DVI ファイル report.dvi を report.pdf というファイル名の PDF ファイルに変換する例を図 3 に示す．作成した PDF ファイルがプレビューア （xpdf，acroread 等）で読めるかを提出する前に確認することを忘れないで欲しい．2020 年度は，演習室 PC を利用しないため，生成した PDF ファイルを手元の PC にコピーしてプレビューアは自分の手元の PC で行う必要がある．

```
nobuya@edu006[304]% platex report.tex
This is pTeX, Version p2.1.11, based on TeX, Version 3.14159 (EUC) (Web2C 7.3.1)(report.tex
pLaTeX2e <2000/11/03>+0 (based on LaTeX2e <2001/06/01> patch level 0)
(/usr/share/texmf/ptex/platex/base/jarticle.cls
Document Class: jarticle 1999/05/18 v1.1q Standard pLaTeX class
(/usr/share/texmf/ptex/platex/base/jsize10.clo))
(/usr/share/texmf/ptex/platex/base/ascmac.sty
(/usr/share/texmf/ptex/platex/base/tascmac.sty))
(/usr/share/texmf/tex/latex/graphics/graphicx.sty
(/usr/share/texmf/tex/latex/graphics/keyval.sty)
(/usr/share/texmf/tex/latex/graphics/graphics.sty
(/usr/share/texmf/tex/latex/graphics/trig.sty)
(/usr/share/texmf/tex/latex/config/graphics.cfg)
(/usr/share/texmf/tex/latex/graphics/dvips.def))) (report.aux) [1] [2] [3]
[4] (alu4.sfl.tex) [5] [6] <count4_gtkwave.eps> [7] (report.aux) )
Output written on report.dvi (7 pages, 23196 bytes).
Transcript written on report.log.
nobuya@edu006[305]%
```

図 3: DVI ファイルの作成例

4 おわりに

最後の節では，本報告書のまとめを述べる．どういったことを報告したのか，目的は達成できたか，どういう成果があったか，どういうことが分かったのか，不十分な点や課題事項は何かを簡潔にまとめる．

なお，この「おわりに」に書かれていることは，基本的に「おわりに」に至るまでの本文中に書かれていることであるべきである．つまり，ここには本文中で書かれていない新たなことを書くことは避けるべきである．

【記述例】

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) ... $\cdots$ (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．

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．

[3] イソテル社の超高速プロセッサ, http://www.isotel.co.jp/high-speed-processor/．

表 1: FPGA への論理合成等で得られた諸量のまとめ

モジュール	最大動作周波数 Fmax 85 °C Model	0 °C Model	LE 数 (使用率)	CF 数 (使用率)	レジスタ数 (使用率)
プロセッサ `p16m1`	12.34	23.45	12,345 (2%)	1,111 (1%)	2,222 (2 %)
プロセッサ `p16m2`					
プロセッサ `p16p1`					

（Fmax の単位は MHz）

表 2: ターゲット FPGA デバイスとそのリソース量

ターゲットデバイス	Intel Cyclone IV E (EP4CE115F29C7)
Logic Element (LE) 数	114,480
レジスタ数	114,480
メモリ容量 (ビット)	4,981,312
9 ビット乗算器	432

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 FSL 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説明に必要な最小限になるように努めるべきである．

A　8 ビット ALU の FSL 記述

図 4 に 8 ビット ALU の FSL 記述を示す．図 4 の 11 行目から 13 行目で内部関数 `zero_check` を定義している．このモジュールで定義されている入力関数では，出力 `out` を求め，内部関数 `zero_check` を起動し，出力 `out` の値がゼロかゼロでないかを判定し，出力 `zero` の値を求めている．

B　表の例

表の例を表 1〜表 4 に示す．なお，表の見出し（表題）は表の上に配置する．一方，図の見出しは図の下に配置することに注意のこと．FSL 記述等のソースリストは図扱いとする．

表を参照する際には，単に表を掲載するだけではなく,「表 1 に FPGA への論理合成，配置配線，および静的タイミング解析で得られた諸量をまとめた結果を示す．ターゲットとした FPGA デバイスとそのリソース量を表 2 に示す.」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本文中で説明した後に，その詳細に関する説明が必要である．本文中から参照されない図表は，載せてはいけない．

また，表に無駄に線（縦線，横線）を入れないこと．項目の区切りの部分など必要最低限に留めるようにすること．

C　画像の取り込み

画像の取り込みの例を図 5 に示す．図 5 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．

```
/* (alu8.fsl) */
module alu8 {
  input         a: Bit(8)
  input         b: Bit(8)
  output      out: Bit(8)
  output overflow: Bit(1)
  output     zero: Bit(1)

  val adder = new add8 // module instatiation

  private def zero_check(): Unit = {
    zero = if (out == 0x00) 0b1 else 0b0
  }
  def op_add(a, b): Unit = {
    out = adder.add(a, b, 0b0).sum
    overflow = ((~a(7) & ~b(7) &  out(7)) |
                ( a(7) &  b(7) & ~out(7)))
    zero_check()
  }
  def op_sub(a, b): Unit = {
    out = adder.add(a, ~b, 0b1).sum
    overflow = ((~a(7) &  b(7) &  out(7)) |
                ( a(7) & ~b(7) & ~out(7)))
    zero_check()
  }
  def op_and(a, b): Unit = {
    out = a & b
    overflow = 0b0
    zero_check()
  }
  def op_or(a, b): Unit = {
    out = a | b
    overflow = 0b0
    zero_check()
  }
  def op_xor(a, b): Unit = {
    out = a ^ b
    overflow = 0b0
    zero_check()
  }
  def op_nor(a, b): Unit = {
    out = ~(a | b)
    overflow = 0b0
    zero_check()
  }
} // module alu8
/* End of file (alu8.fsl) */
```

図 4: 8 ビット ALU の FSL 記述

表 3: プログラミング課題，設計課題および発展課題の実施状況

課題		状況
(プログラミング課題)		
1. 【プログラミング課題 1】N 個の語の加算		(2) 完了
2. 【プログラミング課題 2】N 語のメモリコピー		(0) 未実施
3. 【プログラミング課題 3】乗算		(2) 完了
(設計課題 2)		
4. 【設計課題 2-1】32 ビット加算器	`add32`	(0) 未実施
5. 【設計課題 2-2】32 ビット ALU	`alu32`	(1) 設計中
6. 【設計課題 2-3】32 ビットシフタ	`shift32`	(2) 設計完了
(発展課題 2)		
7. 【発展課題 2-1】32 ビット整数乗算器	`mult32`	(0) 未実施
8. 【発展課題 2-1】32 ビット整数除算器	`div32`	(0) 未実施
(設計課題 3)		
9. 【設計課題 3-1】レジスタファイル	`regs32x32`	(0) 未実施
10. 【設計課題 3-2】実行ユニット	`p32ExecUnit`	(0) 未実施
11. 【設計課題 3-3】デコードユニット	`p32DecodeUnit`	(0) 未実施
(設計課題 4)		
12. 【設計課題 4-1】プロセッサ	`p32m1`	(0) 未実施
13. 【設計課題 4-2】プロセッサ	`p32m2`	(0) 未実施
14. 【設計課題 4-3】プロセッサ	`p32p1`	(0) 未実施
(発展課題 4)		
15. 【発展課題 4-1】改良		(0) 未実施
16. 【発展課題 4-2】乗算機能の実装		(0) 未実施

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．

表 4: 目標達成度と自己評価

項目	達成度 (自己評価)
1. ハードウェア設計処理全般 (処理の概要と流れ)	1〜6
2. ハードウェア記述言語 (FSL)	1〜6
3. ハードウェア設計ツール類の使用方法	1〜6
4. プロセッサの命令セットアーキテクチャ	1〜6
5. アセンブリ言語とそれを用いたプログラミング	1〜6
6. プロセッサの動作原理	1〜6
7. プロセッサの設計	1〜6
8. 実験報告書	1〜6
9. その他	1〜6
10. 総合 (項目 1.〜9. の合計)	9〜54

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 5: 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