UNIVERSITY OF CAPE TOWN

SUBJECT CODE

SUBJECT NAME

Title of Paper

Authors:
Author 1
Author 2
Author N

Student Numbers:
ATHNUM001
ATHNUM010
ATHNUM011

4 January 2020

This document is in draft format. It is known to be incomplet and incorrekt, and it has lots of bad formatting. Drafting styles are defined as follows:

Text that is old and must be re-worked or removed

Shorthand note that should be incorporated into the text later

Something that must be done

Something that must be rephrased

Internal link

Citation link

External link

```
// Comment
normal text
#pre-processor
class // Keyword
"String"
```

I know the meaning of plagiarism and declare that all of the work in the dissertation, save for that which is properly acknowledged, is my own.

.....

Author 1
Author 2
Author N

Abstract

The abstract should be a short summary of your dissertation. It is meant to 'sell' your thesis to interested 'buyers'.

This template includes advice relevant to a BSc final-year project (FYP), Masters dissertation or PhD thesis. While it has some relevance to a much smaller report, such as a course project report or even a lab report, these smaller types of reports will have less pieces and do not need nearly as much detail and depth – you are accordingly suggested to use the **Article** template in this repository; we will try to put together separate examples for a lab report and project report later.

Contents

List of Figures and Listings **iv**

List of Tables **v**

Nomenclature **vi**

1 Acronyms . . . vi

2 Terminology . . . x

1 Introduction **1**

1.1 Background . . . 3

1.1.1 Subsection . . . 4

1.2 Important Terminology . . . 4

1.3 Objectives . . . 5

1.3.1 Sub-objective / Motivational Discussions . . . 5

1.4 Problem Description or Problem Statement . . . 6

1.5 Terms of Reference – requirements and functions . . . 6

1.6 Hypotheses . . . 8

1.7 Scope and Limitations . . . 9

1.8 Dissemination Plan . . . 9

1.9 Document Outline . . . 10

2 Literature Review **12**

2.1 Suggested Approach . . . 12

3 Methodology **17**

3.1 Suggested Structure 19

4 Design **23**

4.1 System Design 24

4.2 Hardware Design 25

4.3 Software Design 26

4.4 Implementation 27

4.5 Integration or Test Rig 28

5 Experimentation **29**

6 Results **31**

6.1 Tips for Results Figures 32

6.2 Tables 33

6.3 Pictures and Screen-shots 34

6.4 Maths 36

7 Conclusion **37**

References **39**

A First Appendix **41**

A.1 Appendix Sections 41

B Second Appendix **42**

List of Figures and Listings

1 Introduction **1**

2 Literature Review **12**

Fig. 2.1 Literature review funnel structure 14

3 Methodology **17**

Fig. 3.1 Example methodology structure and example of how to visualize your methodology 21

4 Design **23**

Fig. 4.1 Example system level design illustration 24

Fig. 4.2 Example hardware level design illustration 26

Listing 4.1 OpenCL kernel to perform matrix multiplication 27

5 Experimentation **29**

6 Results **31**

Fig. 6.1 The correlation coefficient as a function of sample count. 32

Listing 6.1 Octave function to format a figure and save it to a high quality PDF graph 32

Listing 6.2 Example of how to use the FormatFig function 33

Fig. 6.2 Oscilloscope measurement showing physical line signals on both ends of a transmission line during master switch-over [1]. 33

Fig. 6.3 An example image with custom scaling 35

Fig. 6.4 Comparison of various image format qualities 36

7 Conclusion **37**

A First Appendix **41**

B Second Appendix **42**

List of Tables

1 Introduction **1**

TABLE 1.1 Breakdown of sub-tests to be performed in the acceptance testing. 8

2 Literature Review **12**

3 Methodology **17**

4 Design **23**

5 Experimentation **29**

6 Results **31**

TABLE 6.1 My Informative Table 34

7 Conclusion **37**

A First Appendix **41**

B Second Appendix **42**

Nomenclature

1 Acronyms

A ..... Amperes

AC ..... Alternating Current

ADC ..... Analogue to Digital Converter

API ..... Application Programmer's Interface

ARM ..... Advanced RISC Machine

ASIC ..... Application Specific Integrated Circuit

AXI ..... Advanced Extensible Interface

BAR ..... Base Address Register

BCD ..... Binary-Coded Decimal

Bd ..... Baud, in symbols per second

CFAR ..... Constant False Alarm Rate

CMOS ..... Complimentary Metal-Oxide Semiconductor

CPLD ..... Complex Programmable Logic Device

dBm ..... Deci-Bell, relative to 1 mW

DC ..... Direct Current

DDC ..... Digital Down Converter

DDS ..... Direct Digital Synthesis

DMA ..... Direct Memory Access

DSP ..... Digital Signal Processor (or processing)

EDA ..... Electronic Design Automation

FIFO ..... First-in, First-out (queue)

FIR ..... Finite Impulse Response

FMC ..... FPGA Mezzanine Card

FPGA ..... Field Programmable Gate Array

FSM ..... Finite State Machine

GUI ..... Graphical User Interface

HDL ..... Hardware Description Language

HPS ..... Hard Processor System

HSTL ..... High Speed Transfer Logic

I/O ..... Inputs/Outputs

I²C ..... Inter-IC

IC ..... Integrated Circuit

IDE ..... Integrated Development Environment

LE ..... Logic Element

LSb ..... Least Significant Bit

LSB ..... Least Significant Byte

LUT ..... Look-Up Table

LVCMOS .... Low Voltage Complementary Metal Oxide Semiconductor

LVDS ..... Low Voltage Differential Signalling

LVPECL .... Low Voltage Positive Emitter Coupled Logic

LVTTL ..... Low Voltage Transistor-Transistor Logic

MIMO ..... Multiple Input Multiple Output

MISO ..... Master Input / Slave Output

MOSI ..... Master Output / Slave Input

MSb ..... Most Significant Bit

MSB ..... Most Significant Byte

MSI ..... Message Signalled Interrupt

NCO ..... Numerically Controlled Oscillator

NTP ..... Network Time Protocol

PC ..... Personal Computer

PCB ..... Printed Circuit Board

PCI ..... Peripheral Component Interconnect

PCIe ..... PCI Express

PLL ..... Phase Locked Loop

PPDS ..... Point-to-Point Differential Signalling

PRF ..... Pulse Repetition Frequency

PRI ..... Pulse Repetition Interval

PSU ..... Power Supply Unit

RADAR ..... Radio-Assisted Direction and Ranging

REST ..... Representational State Transfer

RF ..... Radio Frequency

RISC ..... Reduced Instruction Set Computer

RMS ..... Root Mean Square

RPM ..... Revolutions per Minute

RSDS ..... Reduced Swing Differential Signalling

SI ..... Système International d'Unités

SoC ..... System On Chip

SPI ..... Serial Peripheral Interface

SSTL ..... Stub Series Terminated Logic

TCP ..... Transmission Control Protocol

TTL ..... Transistor-Transistor Logic

UART ..... Universal Asynchronous Receiver Transmitter

UDP ..... User Datagram Protocol

UFM ..... User Flash Memory

URL ..... Uniform Resource Locator

V ..... Voltage

VHDL ..... VLSI HDL

VLSI ..... Very Large-Scale Integration

XML ..... eXtensible Markup Language

2 Terminology

Developer	FPGA firmware developer, using any firmware development tool.
Device	The specific target FPGA.
Megafunction	A target-specific module, typically generated from within the vendor IDE, after the Altera nomenclature.
Module	Akin to a Verilog module; i.e. unit of digital circuit that has ports to the outside world and can exist at any level of the design hierarchy.
Object file	An intermediary file used in the ALCHA compilation process. It is the reult of compiling a single translation unit and describes a collection of objects.
Peripheral	Any device, external to the FPGA, that interfaces directly with the FPGA.
Platform	The platform that the ALCHA compiler runs on, including operating system and computer hardware.
Target	The platform ALCHA is compiling to, including FPGA, PCB, peripherals and vendor IDE.
Translation unit	Akin to a C language translation unit, i.e. a collection of source files and headers that translate to a single object file
User	The developer who is using ALCHA to develop FPGA firmware.
Vendor	The FPGA manufacturer.

Chapter 1

Introduction

Concerning Use of Latex

If you are new to LaTeX, I would suggest reading [2]. The TA and tutors can provide LaTeX support.

Remember that, for bibliography citations to work, you have to include running BibTeX in the compile chain. You can, for instance, use TeXstudio [3] with a compile chain for "Build & View" set up as follows:

```
txs:///bibtex | txs:///pdflatex |
txs:///bibtex | txs:///pdflatex |
txs:///view-pdf-internal
```

Similar options exist for TeXmaker, VS Code and other IDEs.

Template Features

This template defines various macros aimed at assisting the author in writing consistently styled documents. You are encouraged to make use of these instead of the more low-level LaTeX commands.

Examples of how to use most of these macros are included throughout this example document. One not mentioned, however, is drafting markup. When the template is in draft mode, you can use various helper macros, as illustrated below:

This is old text that should be removed. *This is a note about something to remember, or comments from the proof-reader.* **This is something that still needs doing.** When compiled with `\Draftfalse`, the content of these macros are removed from the output, *except something that needs to be rephrased.*

You can also use cards, as follows:

TODO

This is a todo card.

It is a minipage environment, so you can have all sorts of stuff in it. It can be many paragraphs long, but don't make it too long, because LaTeX will force the whole card onto a single page.

NOTE

This is a nested note card. You can nest cards of arbitrary types as deep as you like.

Introduction Funnel – A General Guide

The introduction, and especially the first paragraph, is a key part of your thesis or dissertation. First impressions matter! This comment goes together with the abstract, but in a way the start of the introduction is setting an even more important first impression (as examiners know that an abstract is often a short piece that is highly polished and possibly not entirely representative of the quality that will appear later).

You want to start off with a "fantastic opening paragraph" and then lead-in to the topic (I call this the "once upon a time" part of the thesis, which is setting the scene for your 'story' ... I mean study). I suggest the use of the "funnel approach" that is illustrated below.

For introducing your dissertation, you essentially want to start broad and then focus in, getting to the main point of the focus. When you've got all the way to the scope, which explains some of the lower level details of e.g. specific context(s) of focus or aspects of exclusion, then one can transition (on something of a new tack) to explaining the outline of the dissertation, what chapters will follow and a brief summary of these, but without giving too many interesting points and findings away in these summaries of the chapters.

```
|  Background   |
|     Terms     | - Optional
| Problem Desc  | - General problem -- may want to jump into objectives
|   Objective   |
 \  Spec Probs /  - Specific problems to study
  \ Questions /   - This is the research question you need to answer
   \   ToR   /    - Terms of reference, requirements, function
    \ Scope /     - Discussing details of focus, things to leave out etc
     | DiP |      - Dissemination Plan; optional - but highly recommended
      -----
      |   |       You kind of close off with the disemination plan,
      -----       which is ulitimately the delivery of your work to
     /     \      the wider audience
    /       \
   / Thesis  \    Then you can go on with (a somewhat different theme
  / Overview  \   that is the structure of the thesis
 /             \
```

A suggested headings layout follows...

1.1 Background

Nice intro to background here...

1.1.1 Subsection

Some more detail follow in subsections...

1.1.1.1 Sub-subsection

Sub-subsections are rare in long style reports, and generally do not appear in the table of contents, but you might want to take it even further:

Note on numbering Note that usually this level does not need to be numbered, nor is it necessary to have it in the table of contents. If you really want to, you can modify the template to include numbering at this level and include it in the table of contents, but having more than four numbers in a row is too much and too distracting for most readers, so be sensible in choosing the number of levels used.

1.2 Important Terminology

If you have a few important terms, e.g. specially defined concepts or acronyms, it's good to introduce them early on, and this is where you could do it (i.e. possibly even before the objective if the objective depends on these terms). Note this is not the same as the nomenclature in the preamble – it is a body of text that might have an introductory passage introducing the terms and why you need to define them, then you might go on to make important definitions, for example in the following format:

> **Thesis:** In this document, 'thesis' refers to a proposition put forward, and then discussed and proved or demonstrated in a scientific manner.

The above style is good for a single term, but if you have a longer list of definitions, rather use a `definitions` environment, as follows:

Term 1: The definition of term 1. It may span multiple lines in the table, but not multiple paragraphs.

Term 2: The definition of term 2. It may span multiple lines in the table, but not multiple paragraphs.

Term 3: The definition of term 3. It may span multiple lines in the table, but not multiple paragraphs.

Term 4: The definition of term 4. It may span multiple lines in the table, but not multiple paragraphs.

1.3 Objectives

Start with a statement of your broad objective

Narrow down to the specific objective of your project (but see note below).

Note

There are essentially two slightly different approaches that you may choose to follow. You can decide which you prefer (my preference tends to be the 'reading in a hurry approach' but you can decide as its your thesis).

Reading in a hurry

1.3 Objectives → *1.4 Problem Description / Problem Statement* → *1.5 Sub-objective / Research Questions* → [*1.6 Terms of Reference*]

Identify board area then narrow down to specifics

1.3 Problem Description / Problem Statement → *1.4 Objectives* → *1.5 Sub-objective / Research Questions* → [*1.6 Terms of Reference*]

1.3.1 Sub-objective / Motivational Discussions

Explain how each sub-objective fits into the main objective, or is needed for some reason to accomplish the objective.

1.4 Problem Description or Problem Statement

(note: this is an alternate to Hypothesis)

If you have a broad problem that you intend to discover insights or useful facts about, and you can not apply a "yes it works" or "no it doesn't work" answer to your research, then use the problem description approach. This approach involves describing what you plan to observe, discover or get interesting – but as yet unknown – outputs from.

1.5 Terms of Reference – requirements and functions

This section is certainly a must-have for an engineering report, dissertation or thesis in which you are to build a prototype or system of some sort according to user requirements. If you are pursuing an entirely empirical study, this section might be irrelevant as in such a case you would probably not have anyone asking for requirements.

Start off by referring back to the objective(s) section. Then outline the specific requirements that you need to satisfy for your project. It is a good idea to briefly explain how these requirements were established and to reference the sources of this information (e.g. meetings, email correspondence, telephone calls, etc. See the APA or Harvard referencing guide for how to reference such things).

A good practice is to have a numbered list: one item for each main requirement (in the methodology you might break these top-level requirements into smaller parts; this would be an effective means to avoid having too much detail in the opening chapter).

Once you have listed the requirements, then list the functionality. It is good to have some text between the requirements and functionality explaining how you went from the one list (i.e. requirements) to the other (i.e. functionality). You may need to draw attention to particular portions of the literature review in which you investigated ways to provide functionality for a particular requirement. The functionality can also be given in a numbered list. For each function indicate which requirement it is linked to (i.e. this would essentially be following the good practice of a traceable design process).

You should also mention in this section about the testing procedures that will be used to check the performance (and/or adequate provision) of the functionality and the requirements (in some cases the requirements may be satisfied by assuring its dependent functionality is properly provided). You could have a table that relates each test to satisfying one or more functionality item (possibly clarifying the requirement that is checked as a result).

I usually like to have requirements numbered R1 to Rn (i.e. if you have three requirements you would number them R1, R2, and R3). In a similar manner, label the functionality F1, F2 and so on. In summary, you should end up with a structure something like this:

The main requirements:

R1. The first requirement described

R2. The second requirement described

R3. So on

R3.1 Although the `SpecList` environment does support nesting (up to 4 levels) you are encouraged not to make use of this feature.

The functionality needed to achieve these requirements:

F1. The first function described (derived from R1)

F2. The second function described (derived from R2)

F3. So on (derived from R3)

Then introduce the table for doing the acceptance testing, as shown in table 1.1. You can refer to a specific test like this: Test T2. Ideally you should only have one function and one requirement per test, otherwise the table becomes too wide. If you do want to add more to the list, reduce the `\TestTableDescriptionWidth` length. Feel free to modify the `TestTable` environment to suite your use-case.

TABLE 1.1

Breakdown of sub-tests to be performed in the acceptance testing.

Test	Description	Func	Req
T1	This is the test description of the first test. It can be a paragraph long.	F1	R1
T2	This is the test description of the second test. It can be a paragraph long.	F2	R2
T3	This is the test description of the last test. It can be a paragraph long.	F3	R3

1.6 Hypotheses

(note: this is an alternate to Problem Description)

A hypothesis is appropriate if you have a well focused project in which you want a binary result as in: "yes it works" or "no it doesn't work". Of course there is still likely to be some 'fuzziness' in the result such as "yes it works well in situation A, kind of in situation B and not at all in case C", but essentially you'd be trying to show a yes/no result (or multiple yes/no results) in the end.

Following the hypothesis approach, you would mention hypotheses somewhere in this introduction chapter, not necessarily at this particular point – maybe earlier, maybe later, depending on whatever makes it easier to read.

Generally, it's a good idea to start with a broad hypothesis, typically labelled H0.

You then refine H0 into a few sub-hypotheses, e.g. H1 to H2, which are often related to the sub-objectives described in 1.3.1. You would then clearly be deciding a series of yes or no answers to H1 to H2, which would then imply something about H0. If you wanted to be fancy you might go as far as using mathematical logic to explain these implications, but that's more commonly found in math / applied math theses.

Example of how the main hypothesis and sub-hypotheses could be presented:

Hypothesis H0: A 600 bps data transmission modulated upon a 1 s sonar ping of base frequency 3 kHz can sustaining data transmissions of 75 bytes per ping over a 2 km distance, with bit error rate below 2%.

Hypothesis H1: Data packets of length 75 bytes, overlaid on 3 kHz sonar pings, are sufficient to provide encrypted submarine identification codes.

Hypothesis H2: Data transmissions overlaid on 3 kHz sonar pings cause minimal degradation in the accuracy of echo location results.

1.7 Scope and Limitations

What were the restrictions on your project? Usually, they are related to time and budgetary constrains. Other scope limitations taking measure to compensate for, or eliminate, potential ethics conflicts, include avoiding invasive testing techniques, contention concerning IP rights, and taking human or animal test subjects out of the research design to circumvent potential injury, subjectivity or other factors.

A general structure for the scope could read as follows:

- short blurb making mention of major scope and reasons for this
- point form list of scope details
- rounding off blurb reflecting on the scope

1.8 Dissemination Plan

PhD must have this!

In this section you explain the way you plan to disseminate this project (i.e. share this work and it's findings with others). If you've completed the dissemination section of the proposal, then you know pretty much what goes here, but, unlike the proposal perhaps, don't make too many promises in this one, because the examiner will see it.

Ideally, if you have a paper published already you would have it in the list of planned outputs and be able to make a song and dance that it has been accepted / published and has gone through a thorough peer review process and been accepted by the scientific community.

That can be said for either a conference or journal – but only, of course, if it is one that has a peer review process, if there's just a review of the abstract, or even no review (i.e. just pay to attend the conference) then you don't want to really have a mention of what sort of low value output, if anything it would just reinforce difficulty of getting the topic accepted, so you only want to mention options and accomplishments that are peer reviewed, and, even better, accredited.

Aim for being accredited and indexed by Thomson / Clarivate Analytics Web of Science (this 'Web of Science' is top notch indexing), or Scopus Indexed (if it is Scopus indexed it is also Scopus accredited), which is prestigious also (all 'Web of Science' tend to be Scopus as well, but not all Scopus items are in 'Web of Science'). If it's not indexed by either of these, then the lower (but still somewhat OK) is the Directory of Open Access Journals (DOAJ). Copernicus is varied, from great to not great, somewhat OK but not great if just this one. INSPEC has increased greatly in quality but is not as prestigious as a 'Web of Science' or Scopus indexing.

If the journal raves about something like 'Google Scholar' or 'Citesear' indexing, it doesn't really mean much as these are automatic engines that are impartial to assessing the quality of the journal.

1.9 Document Outline

Suggestion: Discuss layout of thesis at the end of this chapter. Ideally, the preceding subsections should have given a sufficiently clear and concise explanation of what is to be

done without the reader having to delve into this part.

You could optionally have a figure in this section to augment your description. You could start each paragraph summarising the chapter with the chapter and its number in bold as below (but you could e.g. leave out the bold if you think that makes it more consistent with the rest of the manuscript style.)

Chapter 2 concerns the literature review. It presents technologies, theories and techniques on which this thesis builds.

Chapter 3 (usually) presents the research methodology for this project. The methodology should give a good amount of detail about the acceptance test. i.e. For the acceptance test, explain how you are going to meticulously prove that each item of functionality works, and by implication each requirement as satisfied (as was listed in the introduction).

Chapter 4 gives the design (often called Prototype Design if indeed you are developing a prototype to experimentally test a design concept). You may decide to include a specification prior to the design (usually this is not expected in a BSc or MSc project, but should be given in a PhD thesis). If you have a significant amount to write up about the implementation (i.e. to show pictures of your PCBs, development environment setup, various code and screenshots) then add a separate chapter for implementation. Note that it is preferred to have lengthy blocks of code in the appendix, keeping only snippets (or code fragments) to explain parts of the implementation within the main part of the write-up.

Chapter 5 presents the results, showing results of the design / system in action, and results of the tests (i.e. carrying out the stages of the acceptance tests).

Chapter 6 provides the conclusions. This may be preceded by a discussion of what was done, a discussion of results and testing (e.g. what limitations were found related to ways tests were done, good or bad things about the methodology, amongst other observations related to results or the way they were obtained), followed by a more generalised discussion and conclusions. Provide also future work plans and suggestions.

And don't forget the references at the end of the last chapter!

Chapter 2

Literature Review

2.1 Suggested Approach

Here you reference other, related work, as well as a summary relating to how you improve upon said work [4].

As with suggestions provided concerning the Introduction, in which a funnel type of approach was suggested, essentially going from broader issues to more specific ones, a similar approach can be highly effective for the literature review.

A literature review is one of the most difficult chapters to write, and particularly when it comes to deciding what should – and should not – be provided. Often students provide too much, and go into an excessive amount of detail, on things covered in the literature review. A guiding principle is demonstrating your awareness of the field and related works that will (or could) guide your work, are important to consider (e.g. standards) or that you are building upon.

You do not really want to have a literature review that are essentially recreating lecture notes or summaries on the theory; certainly in some case you do need brief recaps of techniques or theories (especially with references to direct the reader to resources where they can touch up on their understanding of the issues covered). MSc student in particularly are likely going to need advice from their supervisor in making an effective

choice of what should be, and should not be, recapped in the literature review.

Fig. 2.1 provides an illustration of a suggested "funnel structure" around which you could build your literature review. This structure should work for either BSc, MSc or PhD. The difference between them (i.e. BSc towards PhD) is the scale, scope, complexity and novelty of these. For example, a BSc will be small scale (often just a very specific gap in knowledge, if any; it might be more a build and test rather than a finding something to fill a knowledge gap) and not so complex and might not even have any novelty.

A MSc would be a bit larger in scale (but not too much), more complex (likely more things happening and more complicated connections between these) and hopefully (but not necessarily) having some novelty (e.g. using a different platform – a Raspberry Pi instead of an Arduino to trial an embedded solution).

A PhD is of course quite a step up from a MSc in terms of scope, complexity. It is required to prove that you've made a novel contribution in that it has added new knowledge (i.e. filled a non-trivial gap in the research literature).

The parts of the literature review funnel structure are explained briefly in the points below. Note that you do not really want to just use these bullet points for the names of the subheadings of you lit review: you need to think a bit more about what would be suitable heading names and sequencing of these. These tips are only suggestions to help you think about how to construct you literature review and should not be considered a prescribed structure or method to use (theses do tend to be quite unique, there is not really one method that would works for all cases, generally it is built through a process of reviewing what others have done, writing and reworking the document).

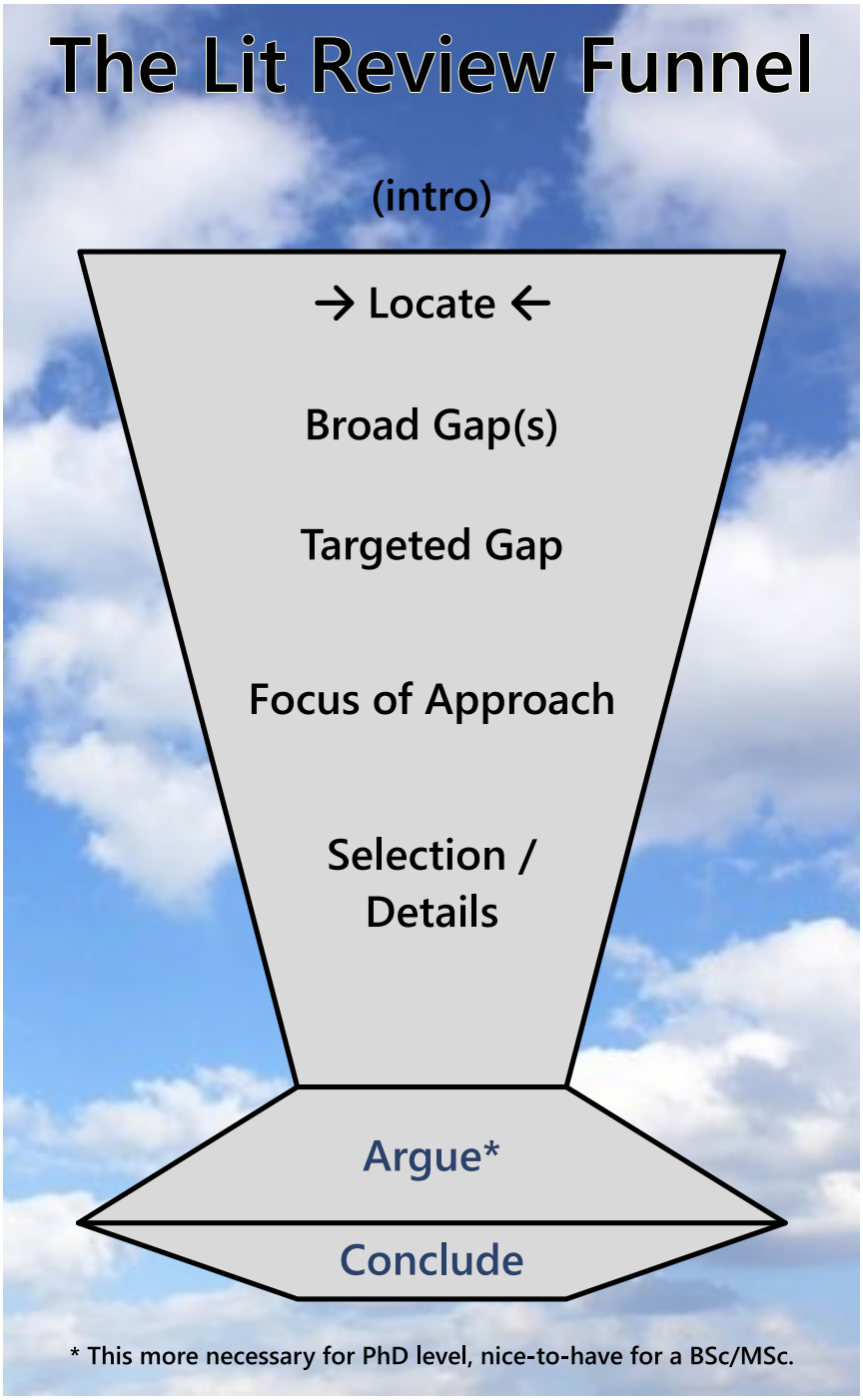

Fig. 2.1. Literature review funnel structure

- **Intro:** Start with explaining the approach to the literature review and its structure. You may want to include a figure (e.g. hierarchical tree view) here that shows the structure of the literature review and which you can connect with in the text for this part.

- **Locate:** Entry to the top funnel, outlining what you will cover. Start drilling down into the field and broad theories. Cover significant aspects of main thinking / trends you are plugging in to.

- **Gap(s):** Ideally, bring the discussion towards a point where you identify gaps in research or areas that need further investigation.

- **Targeted gap:** Of the various gaps identified, indicate which one(s) you will focus on in this project (for BSc / MSc probably just one gap; PhD may have more but not too many).

- **Focus of approach:** Get into the more specific issues, i.e. techniques that will be used to solve the problem and review these theories / methods / tools. At the same time you may cover a few 'alternate approaches' that could inspire your approach or what you are learning from to refine the approach you have chosen, learning from past attempts or similar types of investigation (they obviously don't need to be trying to do the same thing, but have some relation / relevance to your project).

- **Selection / Details:** At this part you get into more specifics of potential development tools or libraries you are planning to use and why. This can be handled fairly easily, e.g. doing brief literature surveys, e.g. web searches, on what are the most popular tools. Could expanded by describing a methodology to use for selecting these tools, the result of this could be tables showing the tool / library options, as well as their pros and cons*. The tool rated best could be the one to choose in the project.

- **Argue:** Nice-to-haves (emphasising 'the researcher' and research benefits); broaden out with argument; identifying new knowledge needed; how this research investigation will contribute new understanding. This part depends on space availability*.

- **Conclude / Summary:** Try to close in some elegant way. This could be done with a short summary of key observations / gaps and useful theories / methods you'll build

on. For e.g. PhD proposal in particular you could end it by justifying (reaffirming the need for) this research.

- *: Items marked by asterisk are more necessary for PhD level, but still a nice-to-have for a BSc/MSc dissertation.

Chapter 3

Methodology

Your methodology tends to fit in well immediately after your literature review, and should flow from it, e.g. the methodology pulling on thing in the lit review.

At this point you should have already defined your research question (in the introduction) and conducted a review of what other scholars have done on the topic (as part of the lit review – but note, as per the guidelines provided for the lit review, that chapter isn't all about what others have done in the focus-area you chose, but may include other things such as important theories and technologies that you are building on).

In planning your methodology, you will likely use insights from the articles and books you read in your literature review – together with advice from your supervisor and other project stakeholders – to decide how you will address your research objective (or research question or hypothesis) that you set in Chapter 1. These plans for an engineering project are likely about how you will choose components and tools (both software and hardware tools), what to prototype and how to build the prototype, how to gather data, how to do testing, etc. Remember that you need to show some logical reasoning for why you chose the methods and approach you are using (possibly referring back to supportive items of lit review or even meeting minutes, which can also be referenced).

The methodology is probably the piece that will be the most critically interrogated by your examiners – because it is demonstrating your understanding of the research process and also a deep understanding of your discipline and your ability to choose effective practices.

Thus, to emphasise what the methodology is: it is describing the methods you followed in your research project for the development and experimentation of a solution to solve (or mitigate) a problem, together with how testing was carried out to show how that problem was solved or reduced using your approach.

Note: Not all engineering projects are a build and test undertaking; please ensure to choose an appropriate methodology and writing structure for your specific project and to do so in collaboration with your supervisor[1].

Methodology vs Design: A vitally important terminology consideration is to realize that 'methodology' is not the same as 'design'. This is a common confusion amongst students who have not done a research project or written a thesis before. This short explanation should suffice to ensure you understand the difference:

- **Methodology:** This describes how you are going to go about doing the research project. It describes the main high-level steps (or phases) of the project. It does not detail design or pieces of your design. You should not need any block diagrams in the methodology.
- **Design:** This is what you have built (or are going to build). It includes diagrams, system block diagram, flow charts, pseudocode / algorithms, schematics, etc. Generally, in a computing-related project, it will end off with implementation details (actual code snippets). These implementation details, depending on how much there are of them, may be better placed in a separate Implementation chapter. Similarly, you might decide to have a separate Integration chapter if there are a lot of pieces that takes quite a bit of explaining and diagrams to describe how they connect and work together.

Brief points about methodology structure

- The start of your methodology should give a brief outline of what is involved (just a paragraph or two).

[1] Note that, yes, unfortunately it is sometimes not possible to provide an effective solution to the chosen problem – it may be too expensive or take too long to solve properly – but even then it can still be considered a contribution of knowledge and still have some value as a research investigation.

- Part of the intro blurb to this chapter should also indicate how you build upon some literature; some considerations of how your methodology was inspired / influenced by other work that you have read.
- Typically an easy and clear approach is presenting your methodology as a series of phases, i.e. a bit inspired by the Waterfall Model [5]. While the Waterfall Model may suggest the big pieces, Boehm's Spiral Model [6] may be more the reality of how the project was carried out (i.e. in an interactive process in which progress and potential risks were assessed and reassessed at various stages during the project.)
- Do not cover the details of the specific design: that is for the design chapter. Do cover the stages surrounding the design, as well as the big pieces of the design activities.
- Explain your experiment approach, important / specialized metrics used, etc.

3.1 Suggested Structure

- Overview of Approach / Plan of action (possibly including an overview or flow diagram showing a visualization of the methodology process)
- Answering the 'How it is done question' (possibly including discussion of metrics used)
- Phases of the research project, or steps taken to perform the research
- Experiment design (experiment design *is not* prototype design, it is the design of experiments to test your prototype or system that you build.)
- Data collection methods
- Data analysis and interpretation methods

Please see Fig. 3.1 for illustration on the structuring of a research methodology. There may be some back-and-forth, at least in the initial stage of the project, in deciding the objectives or research questions to choose for the study. You may, or may not, decide to start you methodology description from such a point (i.e. you might want to rather start with discussing the initial lit review in the understanding that there was a prior process in

deciding the actual focus for the study which you and your supervisor may decide is out of the scope for the write-up.)

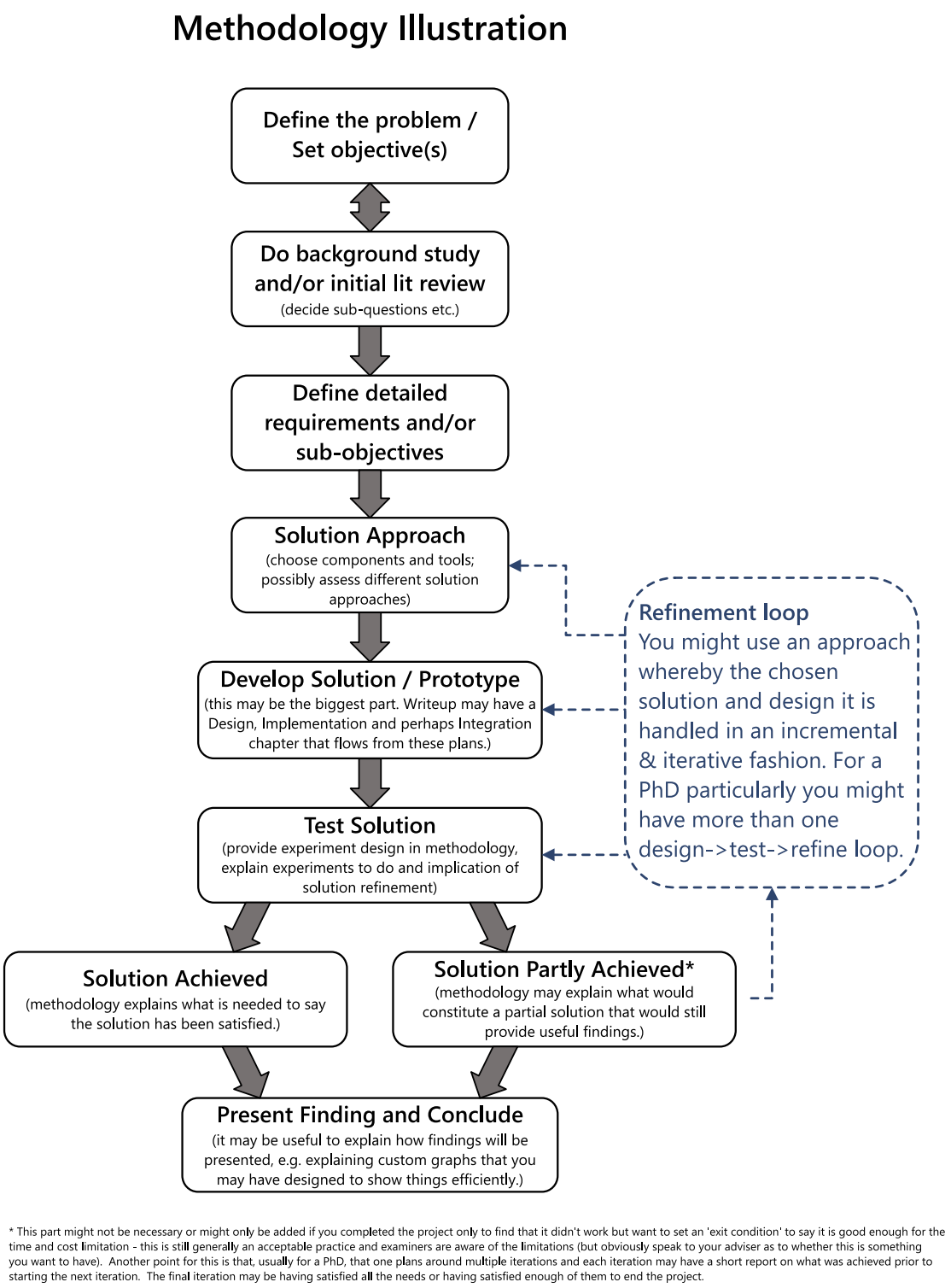

Fig. 3.1. Example methodology structure and example of how to visualize your methodology

Thoughts on using an Appendix: You want to keep your methodology chapter focussed and easy-to-read (but obviously not simplistic). An appendix to the methodology can help keep it focused and uncluttered. For material that is useful in supporting or explaining your method, but is perhaps a bit indirectly connected or at a rather low-level that distracts from explaining the overall method clearly, you could put such details in an appendix (e.g. as supporting material that might clarify, at a lower level, the 'how' and 'why' aspects of you approach). Appendix items for a methodology typically include: copies of questionnaires, detailed interview questions or plans, interview or meeting schedules, minutes from meetings or correspondence (that you are permitted to make public and feel would support decisions in your project), design sketches, mind-maps or concept drawings.

Chapter 4

Design

Note

The Design is, as the name suggests, about the prototype or system you designed in order to achieve investigation or development goals of your research objective. The Design chapter is something that is typically found in engineering theses, hence our inclusion of that chapter. The scope and complexity of this chapter (or associated design chapters) depends on the level of the project: obviously a BSc final year project is going to be smaller scale and less complicated than a MSc project.

Commonly, systems that are built nowadays, and this relates especially to computer engineering or mechatronics types projects (but is relevant to many electrical engineering project too), involve multiple aspects. Typically: (a) the System Level; (b) the Hardware Level and (c) the Software Level. In addition, there may be considerations for the environment and/or 'test rigs' (i.e. the infrastructure that may need to be set up around the system under test in order to perform the testing, and the test rig may in itself be a complicated system that needs thorough design and/or explanation.)

For a PhD, or if your Design chapter starts getting too long, you might decide to rather divide it into separate chapters e.g. according to the subsections provided here.

4.1 System Design

As mentioned, the system you are designing may have multiple parts, both of the prototype and its surrounding test rig (you could call this 'System Level Design' if you prefer, or something more accurate for your particular project). The system level section of the design aims to explain what these big pieces or subsystems are that you are going to develop. Often, for embedded systems particularly, the design is divided into a front-end and a back-end.

The front-end provides the point of interaction with other systems and/or the user. A graphic user interface (GUI) may be part of the front-end (depending on the design) ... or the front-end might be signal conditioning and sampling electronics that then feeds into a front-end processor (e.g. FPGA) and further on into the system (e.g. towards back-end processing stages and storage). The user interface and GUI might be more in the back-end in some designs, e.g. a website services which the user or other programs connect to.

Note: It is usually imperative to have a clear and easy-to-follow diagram (e.g. Fig. 4.1) to illustrate the system level and to refer to in your explanations for this section.

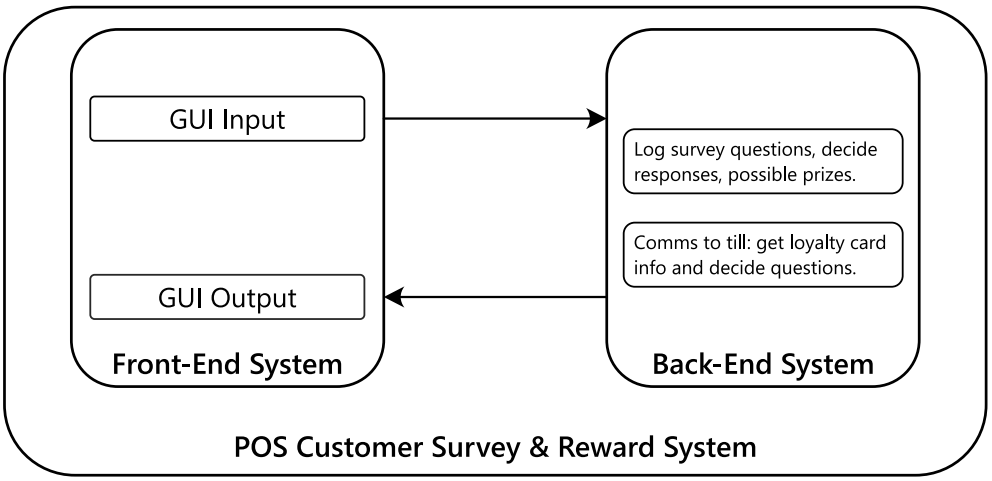

Fig. 4.1. Example system level design illustration

The sections that follow the System Level depends on what your system involves. We have provided an example here of a system that has some Hardware Level aspects, some Software Level aspects and considerations for Integration (in this case setting up a test rig).

4.2 Hardware Design

The Hardware Design sections include significant details on your hardware design, PCB considerations, hardware interfacing and connections and power considerations, among other aspects that are specific to the hardware concerns of your prototype or system being built. By 'significant details' it is suggested that you do not need to go into excessive minutae of the design – if you are building a significant piece of hardware largely from scratch, then you probably need a good amount of details to explain your choices etc. You can also use an appendix in which to park information that may be providing extraneous detail that you think is nevertheless needed but is causing the write-up to become too bulky.

Note: Even if your project is entirely software, you may still want to have a Hardware section to explain what platform and related components you were using; such information can help others to recreate your experiments, which are a desirable property of a good thesis. If you are doing software performance tests you would need to provide characteristics of the platform, thus another reason for having a hardware section (but if the hardware section is just there for platform specs, then you can keep the section pretty short, likely not needing more than a page).

It's generally a good idea to include a block diagram and or schematics at this point. Do not simply have a text-heavy discussion of what parts were used with a detailed schematic and photo of the hardware device that was built (doing so would offload the explanations and logical progression from design to PCB to the examiner to figure out, which is certainly not an advisable approach even if it saves much space).

A representative block diagram, which provides a clear explanation of a specific piece of a design, is presented in Fig. 4.2. This figure was drawn in InkScape [7].

> *NOTE*
>
> When you want to import an InkScape figure (SVG format) into LaTeX, simply save it to PDF (use the drawing extents as the media box area) and include the figure. This template includes a `'make figures'` target meant for this purpose.

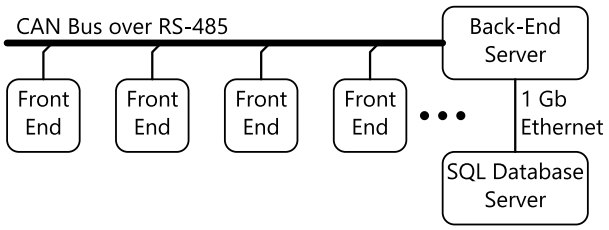

Fig. 4.2. Example hardware level design illustration

4.3 Software Design

The software design section should go from the high-level design aspects, using for example block diagrams or UML class diagrams to explain the main parts of the system, and then going into details of specific operations or algorithms using some or a combination of, for example, pseudocode, UML state charts, UML activity diagrams, flow charts, or other appropriate figures to help the explanations.

You might decide to have some actual code (usually not more than code snippets, i.e. not whole programs) in the Software section, or you might decide to put such details into an implementation section (since the code is something that carries the design into an instantiated implementation).

Things you may have in the Software section include the following:

- Software designs drawings (e.g. block diagrams, UML diagrams, etc.)
- Algorithms (maybe in pseudocode, or actual code such as MATLAB)
- Code snippets (where relevant, used to illustrate how you went from algorithm, or an element of the software design, to executable implemented code)
- Implementation and development methods (for example specific software tools that had to be installed, scripts to run, parameters to use; but remember that some of this particular item may be better placed in the methodology – particularly if it relates to choices that were made earlier on or even before development started.)

4.4 Implementation

For a project that is largely hardware based, the implementation section is sometimes rather short, providing photos of the system and explaining some tips and methods on how it was put together (e.g. solutions that were learned for how to solder on parts effectively, or through the implementation experience realising parts that need to be handled with special care etc.)

For a project that involves hardware and software, this section could include both tips on getting the hardware together, as well as details about implementing the software. The snippets below provide suggestions on how to provide code snippets, as well as providing examples of what might be included in terms of code snippets explaining how part of a design was implemented.

```
# You can include inline Matlab / Octave code
x = linspace(0, 2*pi, 1000);
y = sin(x);
plot(x, y); grid on;
```

The above shows a snippet without caption and placed inline (i.e. it does not float). You could alternatively make it a 'float', as shown in listing 4.1. Floats are tables, figures and listings that appear at a different place than in the source code. This template handles floats in various different ways, depending on the state of the `\Float` and `Draft` flags in `Template/Flags.tex`.

```
__kernel void Multiply(
  __global float* A, // Global input buffer
  __global float* B, // Global input buffer
  __global float* Y, // Global output buffer
  const    int    N  // Global uniform
){
  const int i = get_global_id(0); // 1st dimension index
  const int j = get_global_id(1); // 2nd dimension index

  // Private variables
  int   k;
  float f = 0.0;

  // Kernel body
  for(k = 0; k < N; k++) f += A[i*N + k] * B[k*N + j];
  Y[i*N + j] = f;
}
```

Listing 4.1. OpenCL kernel to perform matrix multiplication

Only list what is relevant. Do not give too much detail – just enough to show what you've done. This template supports the following languages:

- Matlab (Octave)
- GLSL
- OpenCL
- Verilog
- VHDL
- TCL
- Python
- C++ (use the name 'Cpp')
- Scala

4.5 Integration or Test Rig

Some research projects require the development of surrounding infrastructure or a suitably conditioned or prepared environment in order to carry out the testing. This may involve developing some sort of test rig into which the prototype is placed or coupled so that testing can be performed on it. As a simple example, consider a vibration measuring device. If you want to test it in the lab, which has concrete floors but you want to test it for a range of flooring types, it may be necessary to build one or more test rigs that will provide the needed characteristics in order to test the product in a sufficiently authentic situation.

The integration section may alternatively, or in addition to the above point, explain how different subsystems of the system constructed are connected up. For example, this section might be used to explain the different ways to connect up a system that combines some software on a PC, a complicated DSP platform, and perhaps separate front-end conditioning circuitry, in order to complete experiments to test the achievement of different sub-objectives of the project.

Chapter 5

Experimentation

This chapter provides explanation of the software and/or hardware tests to be done. Often, an accumulative sequence of testing is performed, starting with smaller focused tests (e.g. testing algorithms), progressing towards increasingly more comprehensive tests. A typical sequence would be:

1. Algorithmic testing (e.g. demonstrating cases that the algorithm works effectively either mathematically / theoretically or using rapid scripting such as MATLAB).

2. Modular testing (for software) or component testing or component integration / performance testing (for hardware) – note that these are commonly testing pieces in isolation, or testing connected pieces that are not fully assembled into the larger system.

3. Sub-system testing – this is often just testing collections of modules or components to see if they work together. (Might skip this or put it in an appendix as it can lead to the manuscript being too long.)

4. Integration testing – this may be integration of sub-systems or testing of the system as a whole with its various parts working together.

5. System testing or acceptance testing – this may be thorough testing of the specified requirements, both functional and non-functional requirements and/or determining the extent to which the research objectives are met. This may be done in addition, or

instead of, integration testing (integration testing, in contrast to acceptance testing, tend to focus more on seeing if the parts work together and that the system is ready for the more structured acceptance testing to be applied).

You might utilise the V-Model [8] in planning your experiments, particularly for large systems (e.g. PhD level), where you want to show how each aspect is thoroughly tested. However, this is not a general case. More commonly, the experimentation process would follow the above points, which are relevant to BSc through to PhD level projects.

Chapter 6

Results

The results chapter is for presenting and discussing your findings, which can split into sections if the experiment has multiple parts, or stages. You might also want to have a 'Discussion' or 'Discussion of Results' chapter, which may focus on either more detailed discussion of particular results, or more comprehensive discussion of the results and system performance as a whole. If you have a Discussion of Results section, then you do not want to discuss too much about the specific results in this chapter and rather move the main discussion or reflective considerations of these to the Discussion of Results chapter.

However, and this is an important reminder, ensure that you do have some text of discussion for your results, to take the reader through your results and the figures you may provide. Make sure not to just put one figure after another without any attempt to explain the sequencing and what is being shown and perhaps some key issues in the figures presented (a monolithic sequence of figure after figure without any attempt at explanation will get undesirable responses from examiners).

If you have an experiments section, it is often useful to have a clear connection of experiment section corresponding to a result section showing results for that test. Examiners often appreciate that sort of clear and consistent structure that is easy to follow. For example, if you have section 5.1 as Modular Testing, then you could correspond to 6.1 Modular Testing Results (the two sections should be cross-linked with `\label` and `\ref`, in both directions)

6.1 Tips for Results Figures

Include good quality graphs (see Fig. 6.1 for an example). These were produced by the Octave code presented in listings 6.1 and 6.2. You can play around with the `PaperSize` and `PaperPosition` variables to change the aspect ratio. An easy way to obtain more space for an article is to use wide, flat figures, such as Fig. 6.2.

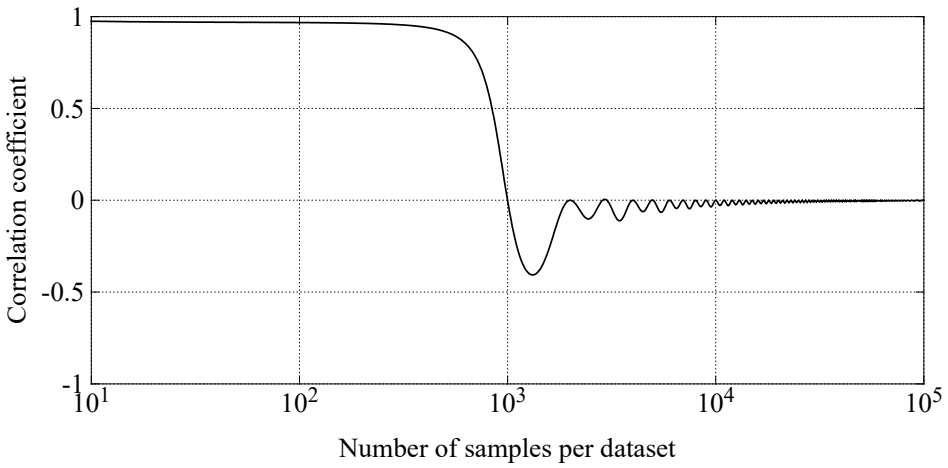

Fig. 6.1. The correlation coefficient as a function of sample count.

```
function FormatFig(X, Y, File);
  set(gcf, 'PaperUnits'      , 'inches');
  set(gcf, 'PaperOrientation', 'landscape');
  set(gcf, 'PaperSize'       ,       [8, 4]);
  set(gcf, 'PaperPosition'   , [0, 0, 8, 4]);

  set(gca, 'FontName', 'Times New Roman');
  set(gca, 'Position', [0.1 0.2 0.85 0.75]);

  xlabel(["\n" X]);
  ylabel([Y "\n\n"]);

  setenv("GSC", "GSC"); # Eliminates stupid warning
  print(...
    [File '.pdf'],...
    '-dpdf'...
  );
end
```

Listing 6.1. Octave function to format a figure and save it to a high quality PDF graph

```
figure;                                 # Create a new figure
# Some code to calculate the various variables to plot...
plot(N, r, 'k', 'linewidth', 4); grid on; # Plot the data
xlim([0 360]);                          # Limit the x range
ylim([-1 1]);                           # Limit the y range
set(gca, 'xtick', [0 90 180 270 360]);  # Set the x labels

FormatFig(...                           # Call the function with:
  'Phase shift [\circ]',...                    # The x title
  'Correlation coefficient',...                # The y title
  ['r_vs_N;_f=' num2str(f) ';_P=' num2str(P)]... # Format the file name
);
close all;                              # Close all open figures
```

Listing 6.2. Example of how to use the FormatFig function

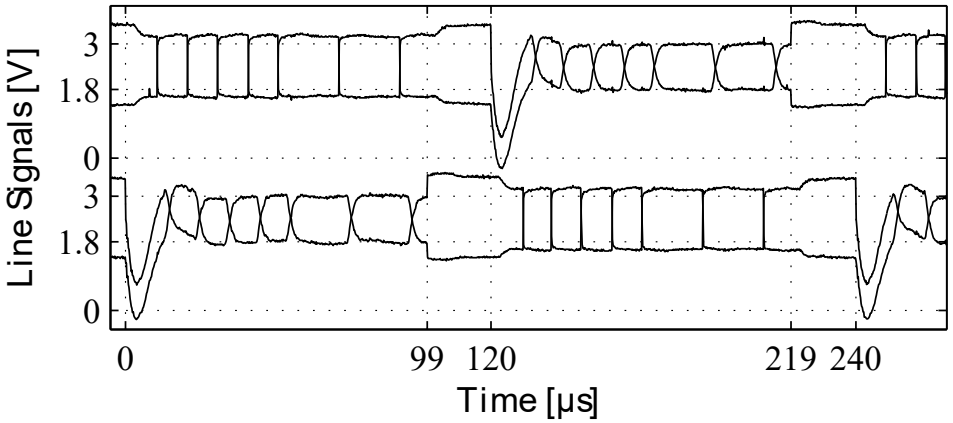

Fig. 6.2. Oscilloscope measurement showing physical line signals on both ends of a transmission line during master switch-over [1].

Always remember to include axes text, units and a meaningful caption in your graphs. When typing units, a µ sign has a tail! The letter "u" is not a valid unit prefix. When typing resistor values, use the Ω symbol.

6.2 Tables

Tables are often a convenient means by which to specify lists of parameters. An example table is presented in table 6.1. You can use Tablesgenerator to make your LaTeX tables.

TABLE 6.1

My Informative Table

Heading 1	Heading 2	Heading 3
Data	123	321
Data	456	654
Data	789	987

6.3 Pictures and Screen-shots

When you include screen-shots, pdfLaTeX supports JPG and PNG file formats. PNG is preferred for screen-shots, as it is a loss-less format. JPG is preferred for photos, as it results in a smaller file size. It's generally a good idea to resize photos (not screen-shots) to be no more that 300 dpi, in order to reduce file size. For 2-column article format papers, this translates to a maximum width of 1024 pixels. **Never change the aspect ratio of screen-shots and pictures!**

It is highly recommended to make use of the `\Figure` macro for figures. It puts all the formatting tweaks in one place, so that you don't need to update all the individual figure inclusion points when you want to do a styling update. The file name is used for the LaTeX label, such as "Fig. 6.3".

Fig. 6.3. An example image with custom scaling

Make sure to always use the best quality image possible. Use JPEG for photos, PNG for screen-shots and PDF (scalable vector graphics) for everything else. JPEG is lossy, but good for photos, whereas PNG is lossless and good for images with large areas of solid colour, as can be seen in Fig. 6.4.

(a) JPEG (b) PNG (c) SVG

Fig. 6.4. Comparison of various image format qualities

6.4 Maths

LaTeX has a very sophisticated maths rendering engine, as illustrated by equation 6.1. When talking about approximate answers, never use ±54 V, as this implies "positive or negative 54 V". Use ≈ 54 V or ∼ 54 V instead.

$$y = \int_0^\infty e^{x^2} \mathrm{dx} \tag{6.1}$$

Chapter 7

Conclusion

The conclusion should provide a summary of your findings. Keep in mind that people sometimes only read the introduction and conclusion of an academic work, which are the most interesting parts. They sometimes scan the tables and figures in-between to get a quick overview of specific things done. If the conclusion hints at interesting findings, only then will the reader be more likely go back and read the whole thesis or paper.

You can also include work that you intend to do in future or would like to recommend others to do, specifically ideas for further improvements, or to make the solution more accessible to the general user-base.

These are some tips for what a conclusion should provide:

- Reflect on what was done, this may be a recap of the main tasks carried out.
- Your conclusion should refer back to the original objectives / hypothesis or research questions and explain how they were achieved or why they weren't (or why the original objectives were ineffectively posed and better ones were developed).
- Discuss hypotheses and how they were proven or disproved. This is optional – only applicable if hypothesis approach was followed (and isn't redundant in terms of the above point).
- 'Last word(s)' on the project, e.g. the most important conclusion or something sweeping (and preferably ending on a positive issue) as a final point.

- Future work – usually you don't need to have a whole chapter dedicated to this. A couple paragraphs (say a half-page for MSc and whole page for PhD) are sufficient.
- You might (if you are one to strive for perfection) want to make the very last sentence/paragraph something elegant and memorable so that your manuscript ends on a high point, e.g. "Based on the unexpected results of X it is clearly a need for Y that may lead, in the future, to Z", or something along those lines.

References

[1] J. Taylor and J. G. Hoole, "Robust Protocol for Sending Synchronisation Pulse and RS-232 Communication over Single Low Quality Twisted Pair Cable," in *Proceeding of ICIT*. Taiwan: IEEE, Mar. 2016.

[2] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl, "The Not So Short Introduction to LaTeX 2_ε," https://tobi.oetiker.ch/lshort/lshort.pdf, Jul. 2015, version 5.05.

[3] B. van der Zander, J. Sundermeyer, and T. Hoffmann, "TeXstudio – A LaTeX Editor," https://www.texstudio.org/.

[4] A. Baboon, B. Charles, D. Ester, and F. Generalson, "An Amazing Title," Their Not-so-awesome University, Technical Report, Apr. 1492.

[5] W. Royce, "The Software Lifecycle Model (Waterfall Model)," in *Proceedings of WESTCON*, Aug. 1070.

[6] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, May 1988.

[7] "InkScape Website," http://www.inkscape.org/.

[8] I. Graessler, J. Hentze, and T. Bruckmann, "V-models for interdisciplinary systems engineering," in *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, 2018, pp. 747–756.

Appendix A

First Appendix

TODO

You might want to have some overview text here explaining what is provided by this appendix and possibly how it is broken down into pieces. The appendix is ultimately supplementary material and additional supporting evidence for aspects of the main text body. Typically you don't need to be overly fussed about explaining its structure and layout, but the quality of the document can be enhanced by a short outline explaining the appendix at its beginning.

A.1 Appendix Sections

Using sections and subsections is highly recommended to help improve the structure and navigation of your appendix, and for providing links from the main body to specific entries in the appendix.

Appendix B

Second Appendix

TODO

It is a good idea to name your appendices appropriately, but not going overboard otherwise it can cause your table of contents to become cluttered.