This is the Project Title

Neil Weidinger

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2021

Abstract

This skeleton demonstrates how to use the `infthesis` style for undergraduate dissertations in the School of Informatics. It also emphasises the page limit, and that you must not deviate from the required style. The file `skeleton.tex` generates this document and can be used as a starting point for your thesis. The abstract should summarise your report and fit in the space on the first page.

Acknowledgements

Acknowledgements go here.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

1.2 Contributions

1.3 Report outline

Chapter 2

Background

Describe and outline background chapter.

2.1 Modern computer architecture

Ever since the advent of general-purpose microprocessor based computer systems, transistor density has been doubling roughly every two years. Famously known as Moore's law, for the first 30 years of the existence of the microprocessor the consequences of this graced the computing world with effortless biennial performance increases. Bestowed with this exponential growth of transistor density, chip designers could drastically increase core frequencies with each generation, and with ever increasing transistor budgets afford to design more complex architectural features like instruction pipelining, superscalar execution, and branch prediction. Without touching a line of code, software developers could expect programs to automatically double in performance every two years [2].

Accompanying Moore's law was another, related, effect: Dennard scaling. While Moore's law provided increased transistor counts, Dennard scaling allowed for this transistor doubling while ensuring power density was constant. The scaling law states roughly that as transistors get smaller, power density stays constant, meaning that power consumption with double the transistors stays the same. Additionally, as transistor sizes scale downward, the reduced physical distances enable reduced circuit delays, meaning an increase in clock frequency, boosting chip performance. When combined, with every technology generation transistor densities double, clock speeds increase by roughly 40%, and power consumption remains the same [1]. This remarkable scaling is what historically allowed for incredible performance gains year over year, all while keeping a reasonable energy envelope.

But starting around 2005, Dennard scaling has broken down: processors have reached the physical limits of power consumption in order to avoid thermal runaway effects that would require prohibitive cooling solutions. This is known as the power wall, and chip designers could no longer regularly rely on increasing clock frequencies to deliver performance gains [3]. The multicore era was born.

2.1.1 Rise of the multicore era

Thankfully, Moore's law is still going strong, but instead of being dedicated to more complex architectural features in order to extract sequential performance in single core processors, the extra transistors are largely used to build more cores. With diminishing returns on effort spent increasing single core performance, chip designers look to add multiple cores to be able to execute more instructions in parallel.

CPU performance can be described using the following equation [4], where performance is measured in terms of absolute execution time:

$$\text{CPU execution time for a program} = \frac{\text{Program instruction count} \cdot \text{CPI}}{\text{Clock rate (frequency)}}$$

where CPI is average clock cycles per instruction. No longer able to increase the clock frequency due to the power wall and increasing difficulty reducing CPI, efforts of hardware architects focused on simply reducing program instruction count per processor, by distributing instructions across multiple CPU cores to be executed in parallel.

Multicore processors are microprocessors containing multiple processors in a single integrated circuit, where each of these processors is known as a core. Almost all commodity multicore processors today are *symmetric multiprocessing (SMP)* systems, meaning all cores in a processor share the same physical address space. A single physical address space allows cores to operate on shared data. Armed with multiple cores, different programs or different parts of the same program can be run at the same time in parallel, reducing the time required to perform the same amount of work on a single processor, boosting performance.

Other variations of the multiprocessor philosophy (essentially running workloads on multiple processors in parallel) can be found in other approaches to parallel computing, for example network connected cluster based designs often found in the High Performance Computing (HPC) domain. These approaches will not be discussed in this report, although the ideas described could possibly be transferable.

As of 2021, it is difficult to find a processor that is not a multicore processor. The performance gains provided by having multiple cores have shown to be so profound that even the lowest end chips feature multiple cores. The recently released Raspberry Pi Zero 2, a £13.50 board in the Raspberry Pi family of low cost single-board computers, features a 64-bit quad-core Arm Cortex-A53 CPU.

Initially multicore processors may seem like a silver bullet to the question of what to do when faced with the power wall: for more performance simply scale the number of cores! In reality, as is typical, the situation is more nuanced. Many workloads cannot be trivially diced up and processed on multiple cores, and even if so, support for splitting up work and then computing this work in parallel must be explicitly supported and designed for.

2.1.2 Parallel computing and its difficulties

Even if the hardware a given piece of software is running on features multiple cores that can be run in parallel, the software must be carefully designed such that it actually takes advantage of the multiple processing units.

2.1.2.1 Amdahl's Law

2.2 Classical work stealing

2.3 Survey of related work

Chapter 3

Conclusions

3.1 Final Reminder

The body of your dissertation, before the references and any appendices, *must* finish by page 40. The introduction, after preliminary material, should have started on page 1.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default 1.5 spacing). Be careful if you copy-paste packages into your document preamble from elsewhere. Some LaTeX packages such as `geometry`, `fullpage`, or `savetrees` change the margins of your document. Do not include them!

Over length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

Bibliography

[1] Shekhar Borkar and Andrew A. Chien. The future of microprocessors. *Communications of the ACM*, 54(5):67–77, May 2011.

[2] John L. Hennessy and David A. Patterson. A new golden age for computer architecture. *Communications of the ACM*, 62(2):48–60, January 2019.

[3] Jeff Parkhurst, John Darringer, and Bill Grundmann. From Single Core to Multi-Core: Preparing for a new exponential. In *2006 IEEE/ACM International Conference on Computer Aided Design*, pages 67–72, November 2006. ISSN: 1558-2434.

[4] David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware Software Interface*. RISC-V edition. Elsevier, Cambridge, MA, second edition edition, 2021.