ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

Department of Information Technology and Electrical Engineering

Autumn Semester 2015

PULPino implementation in 65nm CMOS

Semester Project

Florian Zaruba
zarubaf@student.ethz.ch

December 2015

Supervisors: Frank K. Gürkaynak (DZ), kgf@ee.ethz.ch
Michael Gautschi (IIS), gautschi@iis.ee.ethz.ch

Professor: Prof. Dr. Luca Benini, lbenini@iis.ee.ethz.ch

Acknowledgements

Abstract

PULPino is an open-source microcontroller like system, based on a small 32-bit RISC-V core that was developed at ETH Zurich. The core has an IPC close to 1, full support for the base integer instruction set (RV32I), compressed instructions (RV32C) and partial support for the multiplication instruction set extension (RV32M). It implements our non-standard extensions for hardware loops, post-incrementing load and store instructions, ALU and MAC operations. To allow embedded operating systems such as FreeRTOS to run, a subset of the privileged specification is supported. When the core is idle, the platform can be put into a low power mode, where only a simple event unit is active and wakes up the core in case an event/interrupt arrives.

The PULPino platform is available for RTL simulation, FPGA and as an ASIC in UMC 65nm (Imperio). It has full debug support on all targets. In addition we support extended profiling with source code annotated execution times through KCacheGrind in RTL simulations.

PULPino is based on IP blocks from the PULP project, the Parallel Ultra-Low-Power Processor that is developed as a collaboration between multiple universities in Europe, including the Swiss Federal Institute of Technology Zurich (ETHZ), University of Bologna, Politecnico di Milano, Swiss Federal Institute of Technology Lausanne (EPFL) and the Laboratory for Electronics and Information Technology of Atomic Energy and Alternative Energies Commission (CEA-LETI).

Declaration of Originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor. For a detailed version of the declaration of originality, please refer to Appendix B

Florian Zaruba,
Zurich, December 2015

Contents

List of Figures

List of Tables

List of Acronyms

ADB . . . . Advanced Debug Unit
AES . . . . Advanced Encryption Standard
AMBA . . . . . Advanced Microcontroller Bus Architecture
APB . . . . Advanced Peripheral Bus
ASIC . . . . . . Application-Specific Integrated Circuit
AXI . . . . Advanced eXtensible Interface Bus

DES . . . . Data Encryption Standard
DVI . . . . Device Independent File Format

ECC . . . . Elliptic Curve Cryptography
ECDSA . . . . . Elliptic Curve Digital Signature Algorithm
EPS . . . . Encapsulated PostScript

FPGA . . . . . . Field Programmable Gate Array

IC . . . . Integrated Circuit
IIS . . . . Integrated Systems Laboratory
ISA . . . . Instruction Set Architecture

LED . . . . Light-Emitting Diode

NIST . . . . . . National Institute of Standards and Technology

PDF . . . . Portable Document Format

PULP . . . . . . Open Parallel Ultra-Low-Power Processing-Platform

RISC . . . . . . Reduced Instruction Set Computer

SoC . . . . System on Chip

WYSIWYG . . . What You See Is What You Get

Part I.

PULPino

Chapter 1

Introduction

PULPino is a micro-controller like system based on IPs mostly taken from its bigger brother the Open Parallel Ultra-Low-Power Processing-Platform (PULP) project.

This project origins from the idea to open-source the PULP project. Since PULP is a huge project PULPino is the first effort in doing so. The direct relation to the PULP project is even expressed in the name chosen for the project: In Italian, adding an "-ino" at the end of a word usually means that word corresponds to a smaller version.

Apart from the open source release, having a smaller platform has some tremendous advantages for the PULP project as well. The PULPino platform easily allows us to evaluate new features without considering the overhead of the whole PULP platform in the first place. This is true regarding simple RTL simulation as well as for Synthesis estimates.

In addition to the opportunity stated above there is still the educational aspect of the project. Due to its simplicity it can be of great value for students who want to gather deep understanding of the basic building blocks of a micro-controller like system. This relates to the SoC architecture as well as the idea and construction of a RISC core. It is often useful for ones understanding of a concept to have the possibility to observe a working implementation.

1.1. General Overview

PULPino pursues a simple Harvard architecture (e.g. it has physically separated instruction and data RAMs). At its heart it has a Reduced Instruction Set Computer (RISC) core operating. We currently support two different instruction sets (ISA) for two distinct cores. This can either be our OpenRISC core OR1ON or our RISC-V implementation RI5CY. The cores are pin compatible and can therefore be swapped at one's convenience. The core has debug support enabled through the Advanced Debug Unit (ADB) partially adapted from the OpenCores project. The debug unit provides outside world communication via standard JTAG TAP. The core region (including the core, the debug unit and the RAMs) is communicating over a standard Advanced eXtensible Interface Bus (AXI).

A dedicated AXI to APB bridge connects the internal AXI bus to the (slower) Advanced Peripheral Bus (APB). Both bus specifications are part of the Advanced Microcontroller Bus Architecture (AMBA) specification.

1.2. Document Structure

This document is separated into two different parts. Part I deals with the general concept behind PULPino and everything that is needed to start developing programs and/or specialized IPs easily. It starts with an introduction to the build framework and on how to get everything up and running so that the reader can easily follow along.

I then aim to give a more detailed description of the overall architecture, the different IP cores and their peculiarities. This section concludes in a explanation of the functional verification framework that is shipped alongside the PULPino project.

The second part contains ASIC (Imperio) specific information. It gives insight on the measures taken for the tape-out as well as chip related information and concludes with a chip data sheet. Since Imperio is the ASIC of PULPino everything explained in the first part of the document is directly applicable to Imperio as well.

In the appendix you can find a summary of details needed to start developing for PULPino. This includes a register description and a API description amongst others and it is supposed to act as a quick reference card for application developer.

Chapter 2

Preliminaries / Background

2.1. Getting Started

Chapter 3

Theory / Algorithms

Describe the algorithms you evaluated. The *algorithmic* flow of your work should be clear after this chapter. Do not talk much about the resulting hardware architecture as this is a different topic (next chapter)! If you performed any number precision evaluations put them in this chapter as well.

3.1. First Section

3.2. Second Section

Part II.

Imperio

Chapter 4

Hardware Architecture

Describe the architecture and the architectural decisions you took. Blockdiagrams, the description of control, data flow and interfaces go in here. Note that the architecture you present here usually is more general than what you actually implemented and can even be in a parameterized form.

4.1. First Section

4.2. Second Section

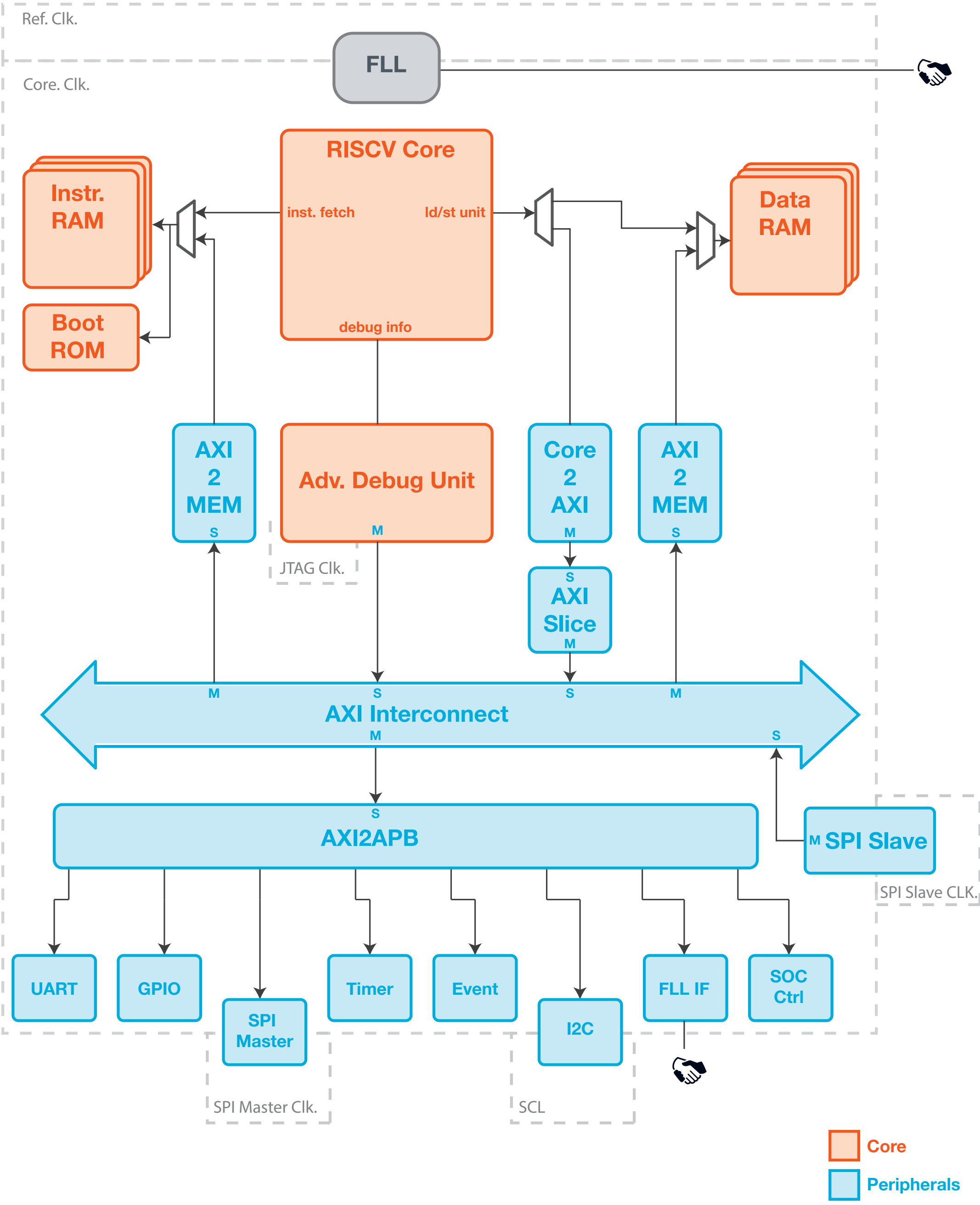

Figure 4.1.: Functional verification setup.

Chapter 5

Design Implementation and Results

This chapter is about the architecture variant you actually implemented and its resulting performance; e.g., SNR, image quality, peak throughput, required bandwidth ... (whatever quality and performance metrics apply). In an ASIC or FPGA project you would also specify the key figures of your design; e.g., area/lut usage, timing figures, interface widths... In an ASIC project you would also talk about backend specific things such as the floorplan of your chip, design for test (and test coverage), power simulation, special clocking circuitry and pad/bonding diagrams.

5.1. First Section

5.2. Second Section

5.3. Verification

5.3.1. Functional

Do not forget to include information about how you managed to do the functional verification (golden model, testbench, etc.). Figure 5.1 illustrates an example setup.

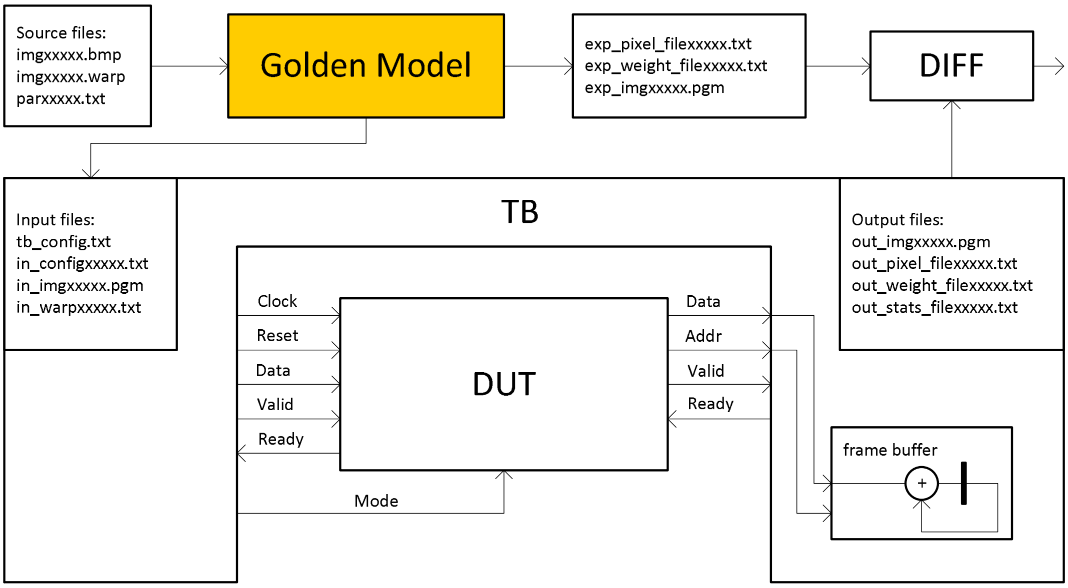

Figure 5.1.: Functional verification setup.

5.3.2. Design for Testability (DFT)

Automated Testpattern Generation

5.4. Results

If you only have very few results, it might be a better approach to insert them into this chapter (instead of putting the results into a separate one).

Chapter 6

Results

If you have a large amount of results you can move them to this separate chapter.

6.1. First Section

6.2. Second Section

Chapter 7

Conclusion and Future Work

Draw your conclusions from the results you achieved and summarize your contributions. Comparisons (e.g., of hardware figures) with related work are also appropriate here. Point out things that could or need to be investigated further.

7.1. First Section

7.2. Second Section

Appendix A

Task Description

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

Semester Project at the Department of
Information Technology and Electrical Engineering

Fall Semester 2015

Florian Zaruba

PULPino implementation in 65nm CMOS

September 15, 2015

Advisers:	Frank K. Gürkaynak (DZ), ETZ J60.1, Tel. +41 44 632 27 26, `kgf@ee.ethz.ch`
	Michael Gautschi (IIS), ETZ J69.2, Tel. +41 44 632 99 58, `gautschi@iis.ee.ethz.ch`
Handout:	September 14, 2015
Due:	December 18, 2015

1 Introduction

At the Integrated Systems Laboratory (IIS) we have been working on a Parallel Ultra-Low Power Processor (PULP) System for the past two years[1]. Throughout the project we have worked on different processor cores. For the first few designs, the original or1k code from the opensource project was used[2]. We then heavily modified this core and improved its IPC (instructions per clock) close to its optimum value of 1. In a next step we started a new implementation from scratch, and called this implementation Or10n[3]. This core has been the main processor core used in PULP projects.

The RISC-V[4] effort started also during this time and is being driven by the Computer Science Division of the EECS Department at the University of California, Berkeley. At the moment, this instruction set architecture (ISA) is very popular in academic circles, so much so that several members of the Open RISC projected have stated that they would stop working on the OpenRISC project and concentrate on RISC-V.

For this reason, we have also investigated the possibility to have a 32-bit version of the RISC-V architecture and during a recent Master Thesis we have developed RI5CY, our own implementation of the RISC-V32 architecture that is reaching a level of maturity close to that of the latest Or10n cores. This core has attracted much attention, also from the lowrisc project[5] of the Cambridge University with which we have a close collaboration.

From the start the intention was to make the entire PULP project open, and provide everyone access to high-quality processor complete with all support (tool flow, debugging, FPGA and ASIC mapping scripts). Until now, we have been unable to rollout a release mainly due to lack of experience in large open projects, and the state of the documentation. As a first step we have decided to release PULPino, a single core processor that uses the same cores we have developed, together with a selected set of peripherals. The release is expected within 2015 and should also contain an FPGA mapping to the popular ZED platform which will allow many people to start using it.

This project has attracted considerable attention, and we have been asked whether or not the processor would support operating systems. In several instances a desire to run FreeRTOS[6] as well as sel4[7] was stated.

2 Project Description

The goal of this project is to implement the first ASIC version of PULPino in UMC65 CMOS process, and to make the necessary changes to the system to run FreeRTOS and/or Sel4. The group is already working on a PULPino release (for FPGA) and the goal is to work together to adapt and extend the PULPino so that it can also support the FreeRTOS and/or sel4. This will at the very least require a timer unit that needs to be added to the system, but may also require some additional changes such as support for supervisor mode commands etc. The tape-out is scheduled to coincide with the student tape-out early in January 2016.

We will join the Orconf conference in October in Geneva[8], to present the PULPino project to a wider audience, and we expect to get some more feedback from the community in general on this topic. Following this conference, and time permitting, we might wish to integrate the ideas from this meeting into the current project as well.

3 Goals

First and foremost, the goal of this project, is to learn and experience the design process for a digital ASIC. At the end of the project a chip in UMC65 will be taped out. We expect that the manufactured chip:

- Uses the latest PULP core either Or10n or RI5CY.
- Be able to boot from an external EEPROM like other PULP systems
- Run an operating system (FreeRTOS, sel4 or other) and a demo application on it.
- Have a basic set of interfaces (SPI, GPIO)

We would like that this work complements our current PULPino project. Furthermore, it would be good if the result of this project could be done in a way that will benefit future collaborations with our partners, specifically ACP which has an interest in running OsmocomBB on FreeRTOS and Simless (part of the LowRISC project) that wants to develop a security module based on a core running sel4.

4 Project Realization

4.1 Project Plan

Within the first month of the project you will be asked to prepare a project plan. This plan should identify the tasks to be performed during the project and sets deadlines for those tasks. The prepared plan will be a topic of discussion of the first week's meeting between you and your advisers. Note that the project plan should be updated constantly depending on the project's status.

4.2 Meetings

Weekly meetings will be held between the student and the assistants. The exact time and location of these meetings will be determined within the first week of the project in order to fit the students and the assistants schedule. These meetings will be used to evaluate the status and progress of the project. Beside these regular meetings, additional meetings can be organized to address urgent issues as well.

4.3 HDL Guidelines

Since most of the PULP project is written in System Verilog, it is strongly suggested to use System Verilog for this project as well. However, any other HDL can also be used if there are strong arguments to use them.

Adapting a consistent naming scheme is one of the most important steps in order to make your code easy to understand. If signals, processes, and entities are always named the same way, any inconsistency can be detected easier. Moreover, if a design group shares the same naming convention, all members would immediately *feel at home* with each others code. At the IIS we make use of the naming convention proposed by the Microelectronics Design Zentrum [9]. The PULP code uses a similar but slightly different style. Thus, try to maintain the PULP naming convention in order to create readable and maintainable HDL code. Note that there might still be some legacy code which may not be compatible to the naming convention. It is not the goal of this work to re-write and adapt all code to a common naming convention, but the newly developed code should be compatible, and the top-level interfaces to legacy code should be adapted to be compatible to the rest of the system.

4.4 Report

Documentation is an important and often overlooked aspect of engineering. One final report has to be completed within this project.

The common language of engineering is de facto English. Therefore, the final report of the work is preferred to be written in English. Any form of word processing software is allowed for writing the reports, nevertheless the use of LaTeX with Tgif[1] or any other vector drawing software (for block diagrams) is strongly encouraged by the IIS staff.

Final Report The final report has to be presented at the end of the project and a digital copy need to be handed in. Note that this task description is part of your report and has to be attached to your final report.

4.5 Presentation

There will be a presentation (15 min presentation and 5 min Q&A) at the end of this project (usually last Thursday of the semester) in order to present your results to a wider audience. The exact date will be determined towards the end of the work.

[1] Tgif is a simple vector drawing software, quite useful for drawing block diagrams. For further information about Tgif we refer to `http://bourbon.usc.edu:8001/tgif/index.html` and `http://www.dz.ee.ethz.ch/en/information/how-to/drawing-schematics.html`.

References

[1] PULP home page: http://pulp.ethz.ch

[2] Opencores website: http://opencores.org

[3] Or10n project website: http://iis-projects.ee.ethz.ch/index.php/Ultra-low_power_processor_design

[4] The RISC-V website: http://riscv.org

[5] The LowRISC website: http://www.lowrisc.org

[6] The FreeRTOS website: http://www.freertos.org

[7] The Sel4 website: https://sel4.systems

[8] The ORconf2015 website: openrisc.io/orconf

[9] The EDA www page (ETH Zurich internal) http://eda.ee.ethz.ch and VHDL naming conventions: http://eda.ee.ethz.ch/index.php/Naming_Conventions

[10] H. Kaeslin. "Top-Down Digital VLSI Design, 1st Edition From Architectures to Gate-Level Circuits and FPGAs". *Morgan Kaufmann*, 2014.

Zurich, September 15, 2015 Prof. Dr. Luca Benini

Appendix B

Declaration of Originality

Include the declaration of authorship with the `\includepdf` command (sign it and scan it). For more information about plagiarism, please visit https://www.ethz.ch/students/en/studies/performance-assessments/plagiarism.html

- **English version:** https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/declaration-originality.pdf
- **German version:** https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiat-eigenstaendigkeitserklaerung.pdf

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

This is a sample title

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	First name(s):
First	Student
Second	Student

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	Signature(s)
Zurich, 01.01.2000	First Student Signature
	Second Student Signature

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Appendix C

File Structure

Describe how the project directories/files are organized, e.g.:

```
/
├── README ......... A README with some general information about the project.
├── 01_report ............................. The source files of the project report.
├── 02_presentation ......................... The source files of the presentation.
└── 03_designflow ................................ Some designflow-specific files.
```

What needs to be done to run an RTL simulation (stimuli generation, compilation...)?

Appendix D

Datasets

If you have a data set comprising several test images, you could depict and describe them here. Use a simple naming scheme such that you can easily refer to certain elements of this data set in the text.

Appendix E

More Evaluation Results

If you conducted an extensive evaluation you could move surplus graphs/results to the appendix.

Appendix F

Algorithms / Tables

Large algorithm boxes and tables may clutter your chapters and impair the readability. If they are not very important, consider moving them to the appendix as well.

Appendix G

ASIC Datasheet (<Chipname>)

If you have designed an Application-Specific Integrated Circuit (ASIC) during your work, you should include a datasheet for your chip into the report. As soon as you start testing your fabricated chip, you will be glad to have such a datasheet. An example structure of such a datasheet is given in the following. For more inspirations on what you may include in your datasheet, have a look at the datasheet of a commercial Integrated Circuit (IC).

G.1. Features

- Lorem ipsum dolor sit amet, ...
- Lorem ipsum dolor sit amet, ...
- Lorem ipsum dolor sit amet, ...
- Lorem ipsum dolor sit amet, ...

G.2. Applications

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

G.3. Description

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

G.4. Packaging

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

G.5. Bonding Diagram

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

G.6. Pin Map

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

UMC 180nm mini@sic QFN56
standard bonding diagram

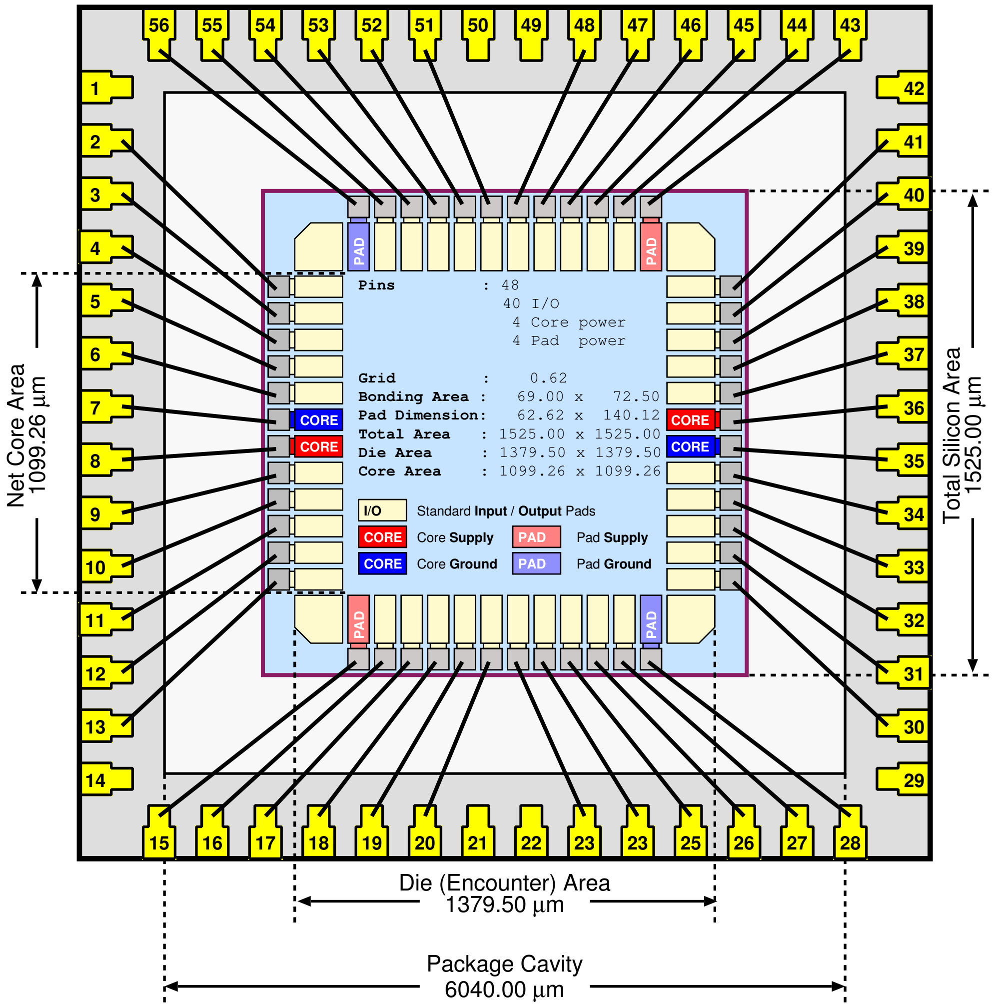

Figure G.1.: Bonding diagram.

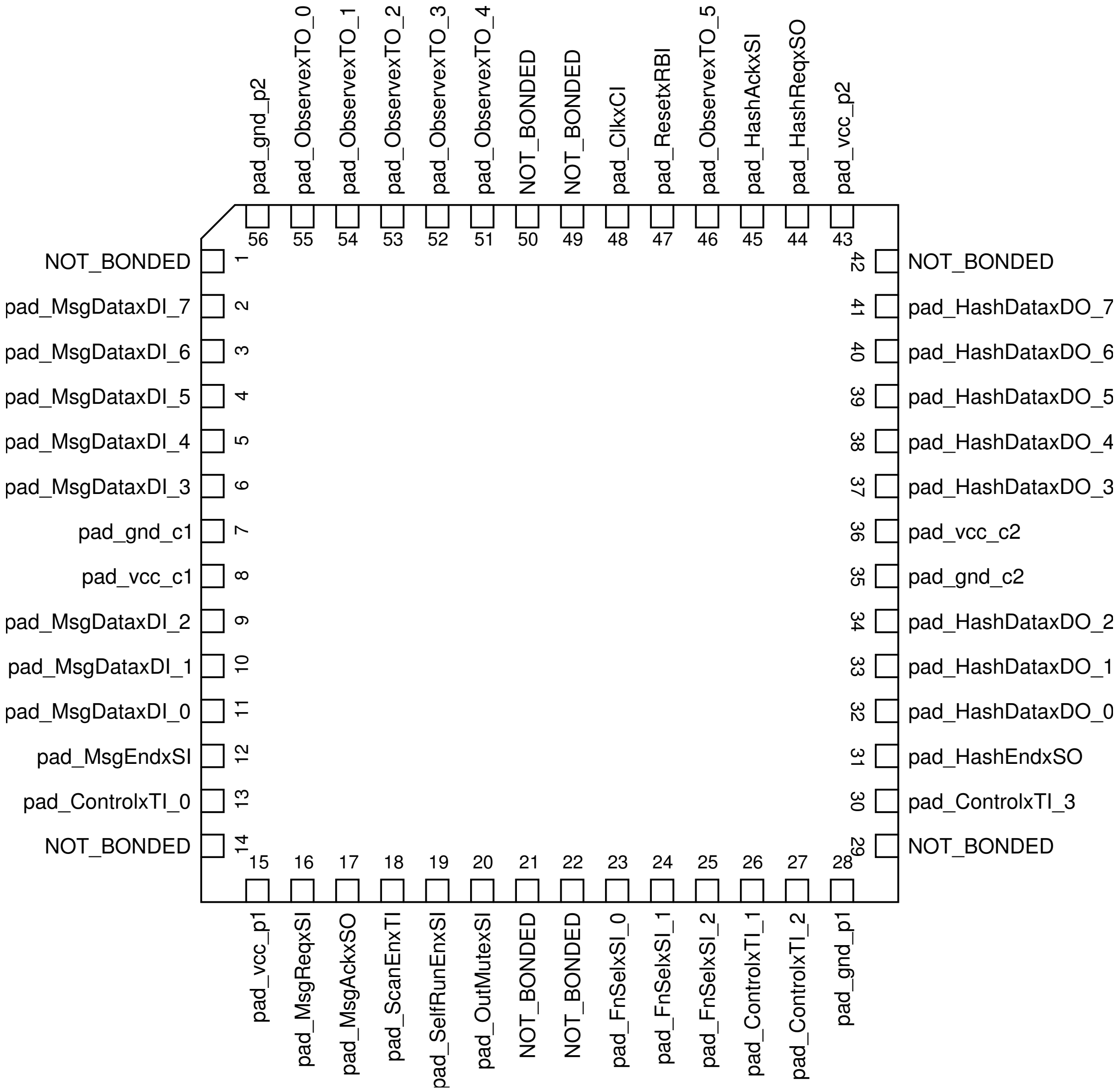

Figure G.2.: <Chipname> pinout.

G.7. Pin Description

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

G.8. Interface Description

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

G.9. Register Map

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

G.10. Operation Modes

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

G.10.1. Functional Modes

G.10.2. Test Modes

G.11. Electrical Specifications

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

G.11.1. Recommended Operating Regions

G.11.2. Absolute Maximum Ratings

Appendix H

The Template Directory Structure

This LaTeX framework suitable for creating reports spreads over various directories and files. In order to give you a short overview of this structure, the respective directories and the contained files are described in the following:

`/`

- `README` .... README file with a quick start guide.
- `Makefile` .... Makefile with some LaTeX related build targets.
- `report_template.tex` ...... The main LaTeX file of the report document, which further loads other (content) files.
- `bib` .... Contains bibliography related files.
  - `main.bib` .... Bibliography file.
- `content` .... Contains the actual source files of your report.
  - `*.tex` .... Here, multiple content files are provided.
- `figures` .... Contains the images which are loaded during your report.
  - `eth_logo.*` ...... ETH logo in Encapsulated PostScript (EPS) and Portable Document Format (PDF) format.
  - `titlepage_logo.*` .... Titlepage logo in EPS and PDF format.
  - `asic_pinout.*` .... Sample pinout of an ASIC in EPS and PDF format.
- `figures_raw` .... Contains the raw sources of your figures.
  - `titlepage_logo.obj` .... Tgif titlepage logo source.
- `glossaries` .... Contains glossaries.
  - `glossaries.tex` . The glossaries file containing both the entries of the list of acronym entries and the entries of the main glossary.
- `preamble` .... Contains preamble information of the document.
  - `preamble.tex` .... Preamble of the report document.

Appendix I

LaTeX Tips

Writing a report with LaTeX may not be as intuitive as it is the case with What You See Is What You Get (WYSIWYG) editors. Especially if you are using LaTeX (more or less) the first time, some problems with the syntax will occur. In general, the present document should already serve as a good starting point for your report and in the best case you only have to insert the content of your project based on this framework.

Nevertheless, I will try to give some useful tips with regard to LaTeX throughout the next sections, which may help you to increase the quality of your documents even further. If you want to use any of the presented ideas, simply copy the LaTeX source code of the appropriate section to your on document and adapt it accordingly.

I.1. Compiling a LaTeX Document

Basically, either `latex` or `pdflatex` can be used in order to generate the document output in Device Independent File Format (DVI) or PDF format, respectively. Throughout this section I will solely use the `pdflatex` command for demonstration purposes (if you prefer a DVI document, just replace the `pdflatex` command by `latex`0).

Compiling a latex document at the Integrated Systems Laboratory (IIS) computers is, in general, as simple as executing the following command in a UNIX terminal window:

```
pdflatex <document_name>
```

Currently[1] a TeX Live version from the year 2008 is the default distribution at the IIS. In order to use the present LaTeX framework for your report, you have to use a more up-to-date version of TeX Live, because the framework uses some LaTeX packages which are

[1]State: July 2012

not part of the 2008 version. I suggest using the 2011 version of TeX Live. The simplest way to check that you can build the report template successfully, is by executing:

```
pdflatex-2011 report_template.tex
```

This should (re)generate the PDF output of the report template, i.e., the file you are currently reading through. If typing in the `-2011` postfix becomes annoying for you, you may add aliases into your `.cshrc` as follows:

```
alias latex 'latex-2011'
alias pdflatex 'pdflatex-2011'
```

If you also want to (re)build the glossaries (maybe you have added some acronyms or the like), you have to compile your report together with the glossaries as follows:

```
pdflatex-2011 your_report.tex
makeglossaries-2011 your_report
pdflatex-2011 your_report.tex
```

Furthermore, when you modify the references of your report (within the bibliography file), you also have to (re)run BibTeX in order to update your bibliography, i.e.:

```
pdflatex-2011 your_report.tex
bibtex-2011 your_report
pdflatex-2011 your_report.tex
pdflatex-2011 your_report.tex
```

I.2. Figures

In order to include an image into your report (as it has been done within in the previous sample chapters), you may use the `figure` floating environment. With that, LaTeX will take care of placing them nicely and you can focus on the actual content of your document. Figure I.1 shows an example of how to insert a single figure.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Figure I.1.: Standard ETH logo.

If you want to place multiple figures side-by-side, you can do this with the use of `minipages`. Figure I.2 and I.3 illustrates an example.

Figure I.2.: Left ETH logo.

Figure I.3.: Right ETH logo.

In order to create a single figure with multiple subfigures, you can do this as presented in Figure I.4

(a) Left ETH logo.

(b) Center ETH logo.

(c) Right ETH logo.

Figure I.4.: Multiple ETH logos as subfigures.

I.3. Tables

Tables in LaTeX allow you to present your results quite nicely. Table I.1 shows a standard table.

Table I.1.: Standard table.

Row 1 - Column 1	**Row 1 - Column 2**	**Row 1 - Column 3**
Row 2 - Column 1	Row 2 - Column 2	Row 2 - Column 3
Row 3 - Column 1	Row 3 - Column 2	Row 3 - Column 3
Row 4 - Column 1	Row 4 - Column 2	Row 4 - Column 3

Sometimes you may want to add a table which stretches one of its columns in order to reach the full width of the document. Such an example is shown in Table I.2.

If you need to place two tables next to each other, you may use an approach based on `minipages` as shown in Table I.3 and Table I.4.

I.4. Creating Glossaries

In order to generate a glossary within your report (e.g., a list of acronyms or an actual glossary), take a look into the file `glossaries.tex`. There, you will find some examples

Table I.2.: Stretched table.

Row 1 - Column 1	**Row 1 - Column 2**	**Row 1 - Column 3**
Row 2 - Column 1	Row 2 - Column 2	Row 2 - Column 3
Row 3 - Column 1	Row 3 - Column 2	Row 3 - Column 3
Row 4 - Column 1	Row 4 - Column 2	Row 4 - Column 3

Table I.3.: Left table.

Row 1 - Column 1	Row 1 - Column 2
Row 2 - Column 1	Row 2 - Column 2
Row 3 - Column 1	Row 3 - Column 2
Row 4 - Column 1	Row 4 - Column 2

Table I.4.: Right table.

Row 1 - Column 1	Row 1 - Column 2
Row 2 - Column 1	Row 2 - Column 2
Row 3 - Column 1	Row 3 - Column 2
Row 4 - Column 1	Row 4 - Column 2

on how to define an acronym as well as a glossary entry. If you want to reference one of the acronyms within your report, you can do it the same way as I did it with the Light-Emitting Diode (LED) right here (just take a look into the source code).

As already mentioned in Section I.1, you have to rebuild your glossaries in order to display changes. For that, you first have to build your document using `latex-2011` or `pdflatex-2011` in a shell window, or the *build-button* in your preferred LaTeX editor GUI. Next, you have to call `makeglossaries-2011 <file_name>` in a shell window[2], followed by another build process of your main source file, i.e.:

```
pdflatex-2011 your_report.tex
makeglossaries-2011 your_report
pdflatex-2011 your_report.tex
```

I.5. Creating Algorithm Boxes

Algorithm boxes in LaTeX allow you to present your algorithms in pseudo code as shown in the following example:

[2]The `makeglossaries` script is a Perl script available at the IIS computer system and should also be part of most TeX distributions.

Algorithm 1: disjoint decomposition

input : A bitmap Im of size $w \times l$
output: A partition of the bitmap

```
special treatment of the first line;
for i ← 2 to l do
    special treatment of the first element of line i;
    for j ← 2 to w do
        left ← FindCompress(Im[i, j − 1]);
        up ← FindCompress(Im[i − 1, ]);
        this ← FindCompress(Im[i, j]);
        if left compatible with this then // O(left,this)==1
            if left < this then Union(left,this);
            else Union(this,left);
        end
        if up compatible with this then                    // O(up,this)==1
            if up < this then Union(up,this);
            // this is put under up to keep tree as flat as possible
            else Union(this,up);                           // this linked to up
        end
    end
    foreach element e of the line i do FindCompress(p)
end
```

Appendix J

General Writing Guidelines

As soon as you get familiar with the syntax of LaTeX (and I can promise you, you will get familiar with it quite quickly as soon as you start writing your reports with LaTeX), some more general writing tips might become of interest for your. Therefore, I collected a few general writing guidlines in the following sections, some of them with regard to LaTeX, some of them not.

Placement of Floating Environments Figures and tables are the two most prominent examples for floating environments. Although the figure examples presented in Section I.2 use `[htbp]` to tell LaTeX how to place them, you should normally only use the `h` parameter if you really require it. Since LaTeX then at first tries to place the figure at the same position as its source code, this somehow contradicts with the actual purpose of the `figure` environment. So, in general, try to place floating environments using one of the following parameters:

t Place the floating environment on **t**op of a page.

b Place the floating environment on the **b**ottom of a page.

p Puts the floating environment on a single *floating **p**age* with other floating environments.

Positioning of Figure and Table Captions Captions of figures are, in general, placed below the actual figure, whereas captions of tables should be placed on top of them. Section I.2 and I.3 contain some examples for figures and tables, including correct placement of captions.

Avoid Unneccessary LaTeX Packages Although there are so many "cool" LaTeX packages available everywhere on the Internet, try to use only those, which you really require. The main problem with loading too many, more or less unknown, packages is that some of them might redefine some commands, etc., which are used by another package which asumes that command to be the original one. Keeping track of these changes and the relations between different packages, is quite annoying and takes quite a lot of time. Hence, keep your preamble simple with regard to packages.

Make Use of Vector Drawings Since LaTeX handles vector drawings pretty good and their scalability allows you to print them in any resolution, prefer them compared to their pixel counterparts and use them whenever possible.

Glossary

Apple Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Candle Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Guitar Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Monkey Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Snake Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Bibliography

[1] D. E. Knuth, *The TeXbook*. Addison-Wesley, 1984.

[2] H. Kaeslin, *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication*. Cambridge University Press, Apr. 2008.

[3] D. Hankerson, S. Vanstone, and A. Menezes, *Guide to Elliptic Curve Cryptography*, ser. Springer Professional Computing. Springer, 2004.

[4] NIST, *Advanced Encryption Standard (AES) (FIPS PUB 197)*, National Institute of Standards and Technology, Nov. 2001.

[5] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption," in *ACM Conference on Computer and Communications Security*, 2001, pp. 196–205.

[6] Xilinx. (2011, Nov.) Virtex-6 FPGA Configuration - User Guide. UG360 (v3.4) November 18, 2011. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug360.pdf

[7] ——. (2011, Oct.) 7 Series FPGAs Configuration - User Guide. UG470 (v1.2) October 26, 2011. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf

[8] Wikipedia, "Isaac newton — wikipedia, the free encyclopedia," 2012, [Online; accessed 1-October-2012]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Isaac_Newton&oldid=514997436