TÉCNICO LISBOA

Linux capable RISC-V CPU for IOb-SoC

Pedro Nuno de Melo Antunes

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. José João Henriques Teixeira de Sousa

Examination Committee

Chairperson: ...
Supervisor: ...
Member of the Committee: ...

September 2022

Declaration

I declare that this document is an original work of my own authorship and that it fullfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

A few words about the university, financial support, research advisor, dissertation readers, faculty or other professors, lab mates, other friends and family...

Resumo

Com o avanço das tecnologias desenvolvidas em código aberto torna-se necessário estudar tanto o novo hardware desenvolvido como o software que tira partido deste novo hardware. Nesta dissertação de mestrado pretende-se conseguir correr um sistema operativo baseado em Linux numa variante do *IOb-SoC*. Ao longo deste trabalho à de-se realizar a implementação de um processador *RISC-V* de *32-bits* capaz de correr o Linux no *IOb-SoC*. No final desta tése pretende-se: em primeiro lugar, ser capaz de correr uma simulação do sistema criado, que mostre o seu correto funcionamento; e em segundo lugar, implementar a variante do IOb-SoC desenvolvida numa FPGA e a partir desta FPGA correr o Linux.

Palavras-chave: RISC-V, Linux, Sistema num Chip (SoC), Verilog

Abstract

With the advances in new open-source technologies, it's imperial that the new hardware solutions and software implementation on the new hardware is studied. The aim of this thesis is to successfully run a Linux based OS (Operative System) on an IOb-SoC variant. During this work, the implementation of a 32-bit RISC-V CPU capable of running Linux on the Iob-SoC is going to be developed. At the end of this thesis, it's expected to: firstly, be able to run a simulation of the SoC (System on Chip) used to run the Linux kernel and verify its correct functionality; and secondly, implement the IOb-SoC variant developed in an FPGA and successfully boot Linux.

Keywords: RISC-V, Linux, Systems on Chip (SoC), Verilog

Contents

List of Tables

List of Figures

Listings

List of Acronyms

AGU	Address Generator Unit
ALU	Arithmetic and Logic Unit
AP	Average Precision
API	Application Programming Interface
AXI	Advanced eXtensible Interface
BRAM	Block RAM
CISC	Complex Instruction Set Computer
CM	Configuration Module
CPU	Central Processing Unit
DMA	Direct Memory Access
DSP	Digital Signal Processing
FF	Flip-Flop
FM	Feature Map
FP	Fixed-Point
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
FPU	Floating-point unit
FU	Functional Unit
GPP	General-Purpose Processor
GPU	Graphical Processing Unit
IOb-SoC	IObundle SoC
IoU	Intersection over Union
IPv4	Internet Protocol
LRU	Least Recently Used
LUT	Look-Up Table
MMU	Memory Management Unit
PE	Processing Element
RAM	Random Memory Access
ReLU	Rectified Linear Unit
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
SFP	Static Fixed Point
SIMD	Single Instruction Multiple Data
SoC	System on a Chip
SRAM	Static Random Access Memory
UART	Universat Asynchronous Receiver-Transmitter

Chapter 1

Introduction

Introdução

1.1 Motivation

Motivação

1.2 Objectives

Objectivo

1.3 Thesis Outline

Estutura do documento

Chapter 2

Background

In this chapter some of the tecnologies used to develop this work will be described.

2.1 RISC-V

In order to understand the RISC-V architectures that are used by current processor families, first it is essential to understand RISC.

2.2 Linux

The Kernel source code can be obtainned through the github repository: https://github.com/torvalds/linux.

2.3 Verilog

Verilog is the HDL used on IOb-SoC to decribe its components.

2.4 IOb-SoC

Building processor-based systems from scratch can be challenging. The IOb-SoC is a System-on-Chip (SoC) template that eases this task by providing a base Verilog SoC equipped with an open-source RISC-V processor (picorv32), an internal SRAM memory subsystem, a UART (iob-uart), and an optional interface to an external memory. If the external memory interface is selected, an instruction L1 cache, a data L1 cache and a shared L2 cache are added to the system. The L2 cache communicates with a 3rd party memory controller IP (typically a DDR controller) using an AXI4 master bus. To help getting started it also comes with an example firmware program. Finnaly, users can add IP cores and software to easily build their SoCs.

2.5 cocoTb

https://indico.cern.ch/event/860269/attachments/1955631/3256707/mdt$_{cocotb_{t}alk.pdf}$

Chapter 3

Proposed System

The aim of this projecct is to build a Linux capable RISC-V-based SoC using IOb-SoC. IOb-SoC already has a RISC-V CPU, picorv32, but this processor isn't capable of running Linux.

3.1 CPU selection

When chosing which CPU should be implemented on the IOb-SoC in order to be able to run Linux on it some specification need to be taken into consideration. Supported architectures are rv32i or rv64i plus standard extensions (a)tomics, (m)ultiplication and division, (f)loat, (d)ouble, or (g)eneral for MAFD. And C stands for compact.

3.1.1 BOOM

The Berkeley Out-of-Order Machine (BOOM) is a synthesizable and parameterizable open source RV64GC RISC-V core written in the Chisel hardware construction language. While BOOM is primarily ASIC optimized, it is also usable on FPGAs. It was created at the University of California, Berkeley in the Berkeley Architecture Research group, its focus is to create a high performance, synthesizable, and parameterizable core for architecture research.

Pros:

•

Cons:

- 64-bit arquitecture
- writen in Chisel

3.1.2 CVA6/Arianne

CVA6 is a 6-stage, single issue, in-order CPU which implements the 64-bit RISC-V instruction set. It fully implements I, M, A and C extensions as specified in Volume I: User-Level ISA V 2.3 as well as the

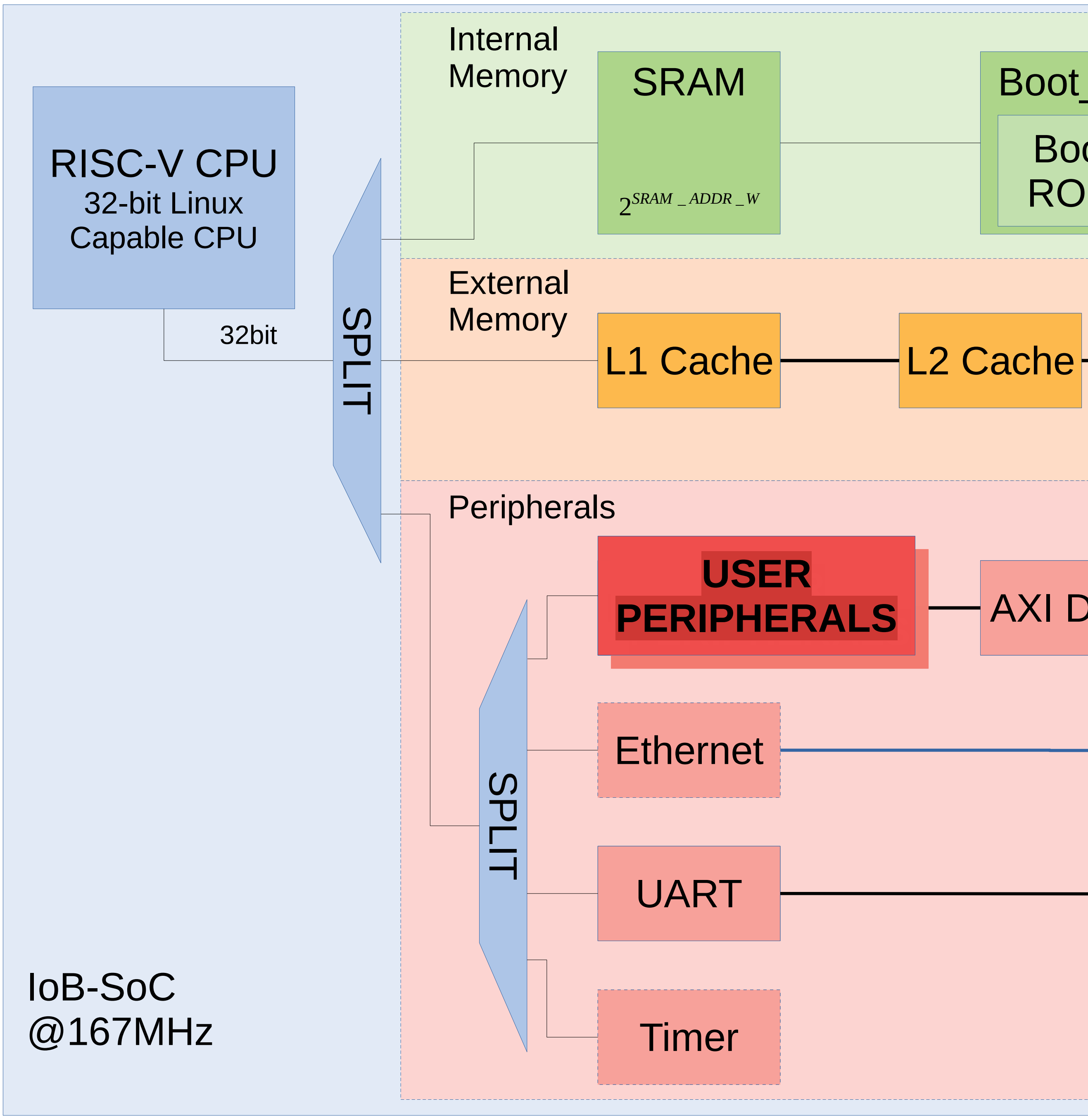

Figure 3.1: Modular hardware arquitecture of the proposed system

draft privilege extension 1.10. It implements three privilege levels M, S, U to fully support a Unix-like operating system.

Pros:

-

Cons:

- 64-bit arquitecture

3.1.3 VexRiscv

VexRiscv CPU, a 32-bits Linux Capable RISC-V CPU written in Spinal HDL

Pros:

- 32-bit arquitecture

Cons:

-

3.2 Creation of an embedded Linux image

Linux repository to create kernel, Busybox to create basic application layer, emulating using Qemu

3.3 Expected system requirements

Silicon Area (ASIC mm2; FPGA LUTs, BRAM, DSPs), Max Freq (ASIC/FPGA)

3.3.1 LitexSoC

Chapter 4

Extra Activities

Translate console to python in the hope to integrate with eth controller (make it easy to use files, sockets, and so on)   Compiling IOb-SOC with Verilator to use a fast and free of charge simulator (show it compiles to competition without errors)   Exploration of Coco$_t b : testbenchthatcommunicateswithconsole(python)impler$

Chapter 5

Conclusions

In conclusion, during this semester some progress was already made for the final objective of my thesis.

5.1 Achievements

During this semester there were both teorical and pratical achievements. A pratical aplication achivement was the development of the IOb-SoC simulation with cocotb (based on python) and the rewriting of the console to python. A teorical achivement was understanding the hardware requirements in a CPU in order to be able to boot Linux.

5.2 Future Work

During the next semester it's when the rest and most of my thesis work will be developed. And during the thesis development a Linux distribution is to be compiled and booted on to the IOb-SoC with a RISC-V processor. The integration of a RISC-V processor is going to be completed on the next semester. There are two possibile solution, either a costum CPU is developed and implemented on the IOb-SoC or a already existing CPU is adapted in order to work with the other IOb-SoC components and implement in it. After that .... boot linux, find linux distribution,

Bibliography