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

Supervisor(s): Prof. José João Henriques Teixeira de Sousa

Examination Committee

Supervisor: Prof. José João Henriques Teixeira de Sousa
Member of the Committee: Prof. Horácio Cláudio De Campos Neto

September 15, 2022

Abstract

Blablabla

Honor Declaration

“Declaro por minha honra de que ”

Local, Date Signature

Contents

List of Figures

List of Tables

Code Listing

1 Introduction

1.1 Motivation

1.2 Objectives

2 Must Have Concepts

During this chapter, we're going to discuss topics that help understand the technology developments made along the thesis project.

2.1 The IOb-SoC Template

2.1.1 Internal Buses

2.1.2 How peripherals work

2.2 Open Source Verification tools

For testing purposes, it is important to have a good hardware simulation environment. For that, we take advantage of already existing and well-developed tools. There exist a number of simulation tools, most of them are proprietary, as for example *'xcelium'* from *'Candence'*. Its utilization can increase the cost of a project significantly. In this Thesis we will make an effort of using open-source, free to use, verification tools. In specific, we will take advantage of *'Icarus Verilog'* and *'Verilator'*. Although both tools are for verification, they serve different purposes, due to their characteristics.

- ***'Icarus Verilog'*** is a Logic Simulator that uses verilog or system-verilog testbenchs to test the UUT (Unit Under Test). Unfortunately, its support for system-verilog is limited and some designs might not run in this simulator. *'Icarus Verilog'* is also known as *'IVerilog'*.

  After compiling the hardware design, *'IVerilog'* outputs a file which can be run line by line to simulate designed logic.

- ***'Verilator'***

The biggest differences are: *'Verilator'* only represents logic signal as 1's or 0's, contrary to *'IVerilog'* which also represents unknown values as X's; Since *'Verilator'* ends up being a C++ program it is much faster to run the simulation than with *'IVerilog'*; On another perspective *'Verilator'* is slower than *'IVerilog'* to interpret the hardware logic design. As such, it is easier to use *'IVerilog'* to detect errors on the design, but it is better to use *'Verilator'* for more complexed simulations.

2.3 RISC-V architecture

The RISC-V **CLINT** is described The RISC-V **PLIC** was first described in the privilege instructions documentation, but since version 1.10 it was moved to its own document.

2.4 What is the UART16550?

2.5 The Linux Boot Flow

2.5.1 Bootloader firmware

2.5.2 What is a device tree?

3 Existing Technologies

There already exists embedded microcontrollers capable of running Linux. Big companies as for example ARM, Qualcomm, MediaTek, Intel and AMD have created microcontroller capable of running Linux. But the processor architecture of those microcontrollers is not open-source, much less the microcontroller itself.

As an example, the *Raspberry Pi 4* is a very capable and cheap board where a developer can test and implement new software running in Linux. The Raspberry CPU is an *Cortex-A72* [1] witch is a System on Chip (SoC) developed by ARM on their ARMv8 64-bit CPU architecture. But if someone wanted to use the Raspberry as a base for his costume hardware design, that would be impossible. And thus appears the need for open-source hardware that allows creating something new without having to start from scratch every time. This led to the appearance of RISC-V the open-source CPU architecture.

3.1 Close source RISC-V Embedded Systems

Since then, a few companies using RISC-V have appeared. RISC-V CPUs are already present in the automotive and IoT markets, besides AI chips in data centers. Due to the RISC-V ISA royalty free license new StartUps tend to look at RISC-V CPUs as a solution for their cores. Even if the CPU Core isn't free to use it ends up being a cheaper solution.

While creating new products companies proved how advantageous the RISC-V architecture was. Furthermore, they have contributed to open-source software, hardware and documentation. Some companies with a big recognitions involved with RISC-V technology are:

- *Western Digital* who now uses RISC-V in its external storage disks.
- *Microchip* as launched the first RISC-V-Based System-on-Chip (SoC) FPGA, *PolarFire*.
- *Antmicro/Microsemi* [1] have built a software called Renode that is used to develop, debug and test multi-node RISC-V device systems.
- more???

These companies have all helped pave the way for a full-feature Operating System based on the Linux kernel to be compatible with the RISC-V architecture. However, there are two companies that have a bigger impact on RISC-V CPU design, those are Andes Technology and Sifive.

[1] Microchip has acquired Microsemi Corporation in May 2018.

3.1.1 Andes Technology

Andes Technology is one of the founder members of the RISC-V International. Since it is highly involved with RISC-V it ended up being one of the major contributor (and maintainer) of the RISC-V tool-chain. This is important because the RISC-V ISA is merely an instruction set architecture, there needs to exist complementing software, such as compiler and development tools.

Nowadays Andes CPU's are applied nearly everywhere, from telecommunications, storage controllers, touch screen sensors to data centers, etc. Andes Technologies has had incredible success using RISC-V technology, as prove they have shipped billions of embedded SoC with RISC-V processors based on their RISC-V ISA variant, AndeStar™ V5.

Andes CPUs witch are capable of running Linux are the *A25* [4] and *AX25* [5]. Both support single and double precision floating point, the RISC-V P-extension (draft) DSP/SIMD ISA and an MMU (Memory Management Unit) for Linux applications. Besides that both enable the use of Machine (M), User (U) and Supervisor (S) Privilege levels that allow running Linux and other advanced operating systems with protection between kernel and user programs. Furthermore, both have L1 instruction and data cache. The difference between them is that *A25* is based on 32-bit architecture and the *AX25* is 64-bit. This leads to the *AX25* being ideal for embedded applications that need to access address space over 4GB, and the *A25* being smaller in gate count. Both CPUs can be implemented on the AE350 [3] SoC allowing to use these CPUs on developer boards, for example in the *ADP-XC7K160/410* [2].

3.1.2 Sifive

Sifive Founder, ..., was one of the bercley students who worked on RISC-V. In 2017 Sifive launched *'U54-MC'* witch was the first RISC-V CPU capable of running a full fledge Operating System like Linux. The SoC had a 64-bit architecture and was quad-core. PLIC, CLINT, MMU

3.2 Open Source Solutions

Built upon the RISC-V open-source CPU architecture, various CPU designs have emerged. RISC-V CPUs are most popular in ... . Some well known CPU are ... . Those will not be discussed here sice they do not meet the requirements to run the Linux Kernel. To run a Linux based Operating System, for example, the ... would have to ... .

3.2.1 PULP Platform

CVA6/OpenPiton

3.2.2 CHIPS Alliance

rocket-chip

3.2.3 SpinalHDL

VexRiscv and NaxRiscv Talk about litex-vexriscv!

4 Hardware Developed

4.1 Simulation Unit Under Test Wrapper

4.2 VexRiscv Wrapper

4.3 UART16550 Wrapper

4.4 CLINT Unit

4.5 PLIC Unit Wrapper

5 Software Developed

5.1 Python Console

5.2 Verilator Testbench

5.3 Barebones Interrupt Routine

5.4 IOb-SoC Linux Stage 0 Bootloader

5.5 IOb-SoC on OpenSBI

5.6 IOb-SoC Device Tree

5.7 IOb-SoC Linux *'rootfs'*

5.8 Makefiles

6 Products of the expedition, AKA Project Results

6.1 FPGA Resources Consumption

6.2 Run/Boot Linux Performance

7 Contributed Repositories

- **iob-soc**
- **iob-soc-vexriscv**
- **iob-lib**
- **iob-vexriscv**
- **iob-uart16550**
- **iob-clint**
- **iob-plic**

8 Conclusions

8.1 Achievements

8.2 Future Work

A Annex

A.1 Annex 1

Annex 1

A.2 Annex 2

Annex 2

Bibliography

[1] Arm Holdings (arm). *The Cortex-A72 processor specifications*. <https://developer.arm.com/Processors/Cortex-A72>.

[2] Andes Technology. *ADP-XC7K160/410, FPGA Based Development Platform*. <http://www.andestech.com/en/products-solutions/andeshape-platforms/adp-xc7k160-410/>.

[3] Andes Technology. *AE350 Platform*. <http://www.andestech.com/en/products-solutions/andeshape-platforms/ae350-axi-based-platform-pre-integrated-with-n25f-nx25f-a25-ax25/>.

[4] Andes Technology. *AndesCore™ A25, Compact High-Speed 32-bit CPU for Real-time and Linux Applications*. <http://www.andestech.com/tw/%E7%94%A2%E5%93%81%E8%88%87%E8%A7%A3%E6%B1%BA%E6%96%B9%E6%A1%88/andescore-processors/riscv-a25/>.

[5] Andes Technology. *AndesCore™ AX25, Compact High-Speed 64-bit CPU for Real-time and Linux Applications*. <http://www.andestech.com/tw/%E7%94%A2%E5%93%81%E8%88%87%E8%A7%A3%E6%B1%BA%E6%96%B9%E6%A1%88/andescore-processors/riscv-ax25/>.