Project Proposal

"Embedded software application for a RISC-V based system-on-chip (SoC) for LHCb Velo detector"

Tarasenko Viktoriia

May 8, 2024

1. Project Summary

1. 1 Problem statement

The increase in complexity and size of modern ASIC designs in the HEP community raises the need for a shift toward an abstract design methodology that takes advantage of modularity and programmability to achieve a faster turnaround time both for design and verification. Integrating System-on-Chip (SoC) design techniques can help provide a structured approach to manage and optimise the integration of multiple functions onto a single chip, and allow programmability in the on-detector ASICs.

Possible applications within the HEP community could range from simple control and monitoring tasks, like calibration and configuration, to performing on-chip complex data processing.

A platform to automatize the generation of the SoC hardware description and software stack is currently under development. This will strongly facilitate the design of complex on-detector front-end electronics.

As part of this effort, to expand programmability options of the specified platform, it's necessary to develop a technological concept of how high-level C++ coding can be used for such interactions, as well as the development of code for simulating and testing the generated SoC hardware.

To optimize the hardware architecture to the target application (as the readout ASICs of the LHCb Velo detector), is necessary to study and simulate the SoC already executing the expected functionalities This may introduce a shift toward abstract design methodology for faster turnaround time of effective system implementation.

1.2 Solution

It's supposed to develop a conceptual prototype for integration of firmware applications written in high-level C++ with a simulated microcontroller based on RISC-V architecture. This will help identify problems at the design stage and ultimately make improvements. Furthermore, it's crucial to understand the software and effectively use the libraries provided for future application development.

1.3 Goals

• Understand how to run a software developed in C++ code on the simulated hardware of a custom RISC-V processor.
• Design a general concept of software interoperability between different abstractions applied in the project.
• Create application prototype to demonstrate scoped functionality of the project for further evaluation.
• Develop different C++ short routines for testing the peripherals and functionalities of the SoC, and execute them on the simulated hardware
• Develop a C++ code for a specific application of the SoC (calibration routine of the PicoPix chip, a prototype readout ASIC for LHCb Velo Detector.).

2 Project plan

Weeks 1 - 3

• Assess details about the SocMake platform, RISC-V instructions set and general project objectives.
• Learn and understand the hardware architecture used in the project.
• Understand how to run a software developed in high-level C++ code on the simulated hardware.
• Setup and configure the development environment for C++ coding with the help of the team colleagues.

Weeks 4 - 6

• Design a concept of general software implementation applied in the project.
• Integrate the environment with low-level project peripherals through the HAL.

Weeks 7 - 9

• Fine-tune general system operation ability targeting scoped scenarios.
• Develop a prototype of conceptual project functionality and testing routines for the SoC peripherals.

Weeks 10 - 12

• Write a software routine for the specific application of the SoC (calibration of the PicoPix chip, a prototype readout ASIC for LHCb Velo Detector).
• Create a demonstration of the project prototype with test data.