EPFL Embedded Systems Laboratory (ESL)
Director: Prof. David Atienza Alonso http://esl.epfl.ch

Porting and Optimization of biomedical benchmark applications to the X-HEEP Open-Source RISC-V-based microcontroller platform

Contact Persons: Dr. Davide Schiavone (davide.schiavone@epfl.ch)
Mr. Dimitrios Samakovlis (dimitrios.samakovlis@epfl.ch)
Mr. Stefano Albini (stefano.albini@epfl.ch)
Prof. David Atienza (david.atienza@epfl.ch)

Thesis Description

Biomedical Applications for ultra-low-power wearable devices

Wearable devices promise to improve preventive medicine through continuous health monitoring of chronic diseases. The design of low-power wearables for the biomedical domain has received much attention in recent decades, as technological advancements in chip manufacturing have allowed real-time monitoring of patients within the µW range. To ensure continued progression in this domain, a co-design view that optimizes both hardware and software simultaneously, and standardized tools are necessary.

In response to the aforementioned needs, the Embedded Systems Laboratory(ESL) has developed BiomedBench. BiomedBench is a new benchmark suite composed of state-of-the-art (SoA) biomedical applications for real-time monitoring of patients using wearable devices. Each application presents different requirements during the typical signal acquisition and processing phases, including varying computational workloads and relations between active and idle times. Therefore, BiomedBench provides hardware developers with a tool to assess the efficiency of their ultra-low power (ULP) platform designs under varying requirements. Moreover, the open-sourcing nature of BiomedBench will serve as a baseline for future application developers aspiring to develop and deploy their biomedical applications in ULP devices.

Typically, biomedical applications for patient monitoring tasks include the modules depicted in Figure 1 below. Typically, the processing step consists of signal preprocessing, feature extraction, and inference based on these features. However, applications can exhibit a wide range of workloads and computational requirements. For example, feature extraction can be implemented explicitly (that is, manually engineered features) or implicitly (e.g., convolutional neural network (CNN)). Similarly, the inference step can use a lightweight machine learning method, such as a random forest or a complex deep neural network (DNN).

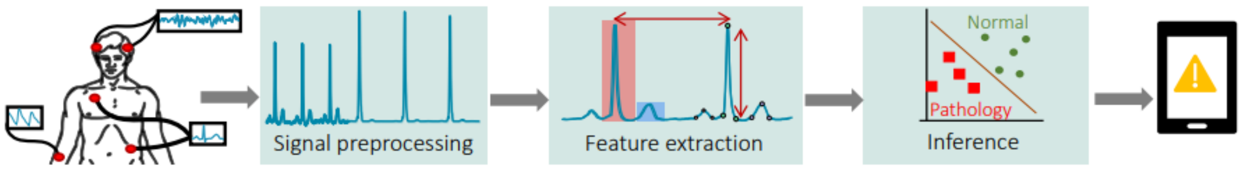

Figure 1: Typical computational pipeline of biomedical applications

From an implementation point of view, the system undergoes an always-on acquisition phase and an intermittent processing phase, as presented in Figure 2. A complete processing period consists of an idle period, during which the processing unit is in low-power mode, and a computation upon acquisition of the full input signal. The duration of the idle period varies significantly between applications and can dominate the system's energy consumption.

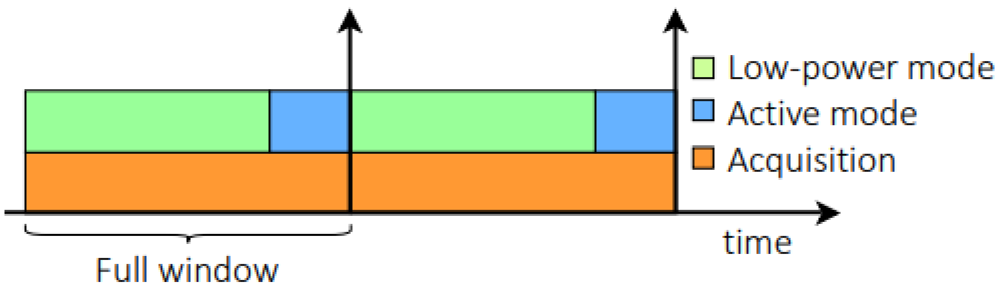

Figure 2: System operating modes during signal acquisition

BiomedBench includes eight applications representative of the biomedical domain that offer a variety of workloads and profiles for the processing, idle, and acquisition phases. All applications are coded in C or C++. Four applications are implemented in fixed-point arithmetic, targeting low-end MCUs. The rest are implemented in 32-bit floating point arithmetic. Four applications also include a multi-core implementation that enables significant acceleration in the presence of multiple cores.

Considered ULP hardware platform - X-HEEP / HEEPocrates

On the hardware side, ESL has devoted a lot of research efforts to developing a new open-source hardware platform, called X-HEEP (eXtendable Heterogeneous Energy-Efficient Platform), to support the monitoring of participants in clinical studies with low energy footprint. X-HEEP is an open-source, configurable, and extensible single-core RISC-V 32b MCU, sponsored by the EcoCloud Sustainable Computing Center of EPFL. It is based on many third-party open-source IPs and in-house IPs developed at the Embedded Systems Laboratory (ESL) jointly with other EPFL laboratories. X-HEEP provides a framework to run applications compiled for RISC-V on a simulator (Verilator, Questasim, or VCS), on a Xilinx FPGA, and can be implemented in silicon as well.

In 2023, ESL fabricated HEEPocrates, the first ASIC implementation (in TSMC 65nm) deploying X-HEEP configured with the cv32e2 core and with 256kB of memory. HEEPocrates instantiates X-HEEP as the main microcontroller driving a CGRA, an In-Memory Computing macro. HEEPocrates belongs to the category of ULP platforms featuring a 6mm2 X-HEEP chip, a maximum frequency of 470 MHz consuming up to 48mW. Hence, HEEPocrates is a suitable platform to deploy the BiomedBench applications on and conduct a performance and energy analysis.

Thesis summary

The goal of this thesis is to **utilize BiomedBench to evaluate the X-HEEP and HEEPocrates platforms**. To achieve this, the student will have to:

1. **Learn to deploy the BiomedBench applications on X-HEEP / HEEPocrates** by efficiently utilizing the capabilities of the platform for the sleep and acquisition phases
2. **Perform timing and energy measurements** of each application running on X-HEEP /HEEPocrates, analyze and compare with SoA results.
3. **Deploy the BiomedBench applications on X-HEEP FPGA** changing the memory size and CPU to find the optimal X-HEEP configuration for BiomedBench, including data transfers from the FLASH to the on-chip SRAM when data overfit the internal capacity.
4. (OPTIONAL) **Apply algorithmic and software optimizations** on each application to speed up computations or/and reduce memory footprint in HEEPocrates without degradation of the final application-level accuracy result.

Thesis outcome

The outcome of the M.Sc. thesis will be published open-source in the BiomedBench and X-HEEP (link) repositories: The expected outcomes of this thesis are:

- [Deployment of complete applications to X-HEEP and/or HEEPocrates] Development of software to run all the applications on X-HEEP / HEEPocrates. The basic C/C++ implementation of each application is given, but the porting to the platforms requires some extra code and smart deployment decisions to respect the memory constraints of the platform. Complementary to the processing part of each application, the acquisition and sleep mode should be programmed efficiently.

- [Performance and energy results] Measuring performance and energy consumption running each complete application on X-HEEP / HEEPocrate and comparing with other state-of-the-art platforms.
- [Finding the optimal] Find the optimal configuration of X-HEEP FPGA implementation for the benchmark by varying the CPU and memory capacity.
- [Application optimization] Optimize the C/C++ implementation and/or the algorithms involved in each application. The target of the optimizations is to improve the energy efficiency of the complete application which can be achieved through decreasing the execution time or the memory footprint, provided there is no accuracy degradation in the final result.

Learning outcome
Throughout the thesis, the student will learn:
- How different real-time patient monitoring applications are structured and what is the state-of-the-art in the domain
- How to deploy such applications in resource-constrained devices
- How to orchestrate the processing, acquisition, and sleeping phases of such applications
- How to identify application bottlenecks and apply algorithmic or software optimizations
- How to study the critical architectural features of a platform, such as X-HEEP, to achieve maximum energy efficiency for each application deployment
- How to use Git to manage projects with multiple developers
- How to collaborate with the team and to analyze and present the results

The thesis will be carried out at the ESL at EPFL, one of the world's top-class universities. ESL is an active group (24 Ph.D. students among 45 members) involved in many research aspects. The student will be under the supervision of Prof. David Atienza, Dr. Davide Schiavone, and two Ph.D. students (Dimitrios Samakovlis and Stefano Albini).

Required knowledge and skills:
- Low-level software design (C and/or C++ is going to be used throughout the thesis)
- Good understanding of memory architectures and microcontrollers
- Good analytical skills
- Makefiles for complex project structures
- Teamwork and git

- Good background in algorithms and common ML models (required for the optimization part at the end of the project, which is optional)

Appreciated skills:

- Scientific curiosity
- Good communication skills
- Advanced English
- Assembly knowledge (useful for in-depth performance analysis)

Type of work: 10% theory analysis, 90% coding and experimenting