GEM5 Extensions: Broadening support to Microcontrollers, with GUI

BTP I
Mid Sem Report

Name: Ashutosh Vishwakarma
Student ID: 2022UCS0083
BTP Instructure: Dr. Subhasis Bhattacharjee

1. Motivation

Microcontrollers lie at the heart of numerous embedded applications, from simple sensor nodes to complex IoT devices. While gem5 is a well-established simulation framework for mainstream architectures (e.g., x86, ARM, RISC-V), its capabilities for microcontroller-focused ISAs have been limited. The scope of this project is to encompass multiple compact, embedded-oriented ISAs—AVR, ARM Thumb mode, Xtensa (as used by Espressif), and simplified RISC variants—to offer a unified simulation platform. Furthermore, to improve usability and educational value, a graphical user interface (GUI) will be developed to visualize run-time data such as pin status and inter-module communication. By extending gem5 in these ways, researchers, developers, and educators can thoroughly explore microcontroller architectures, gauge performance trade-offs, and validate firmware in a single environment. This helps unify studies in embedded domains with larger-scale computer architecture research, offering scalability and versatility under one simulation umbrella.

2. Literature Survey

gem5 remains a gold standard for architectural research, thanks to its modular design and support for multiple ISAs. Butko et al. [1] evaluated gem5's accuracy under diverse workloads, revealing how its componentized structure aids in exploring a wide range of systems. Bharadwaj and Vudadha [2] focused on performance metrics for x86 and ARM, demonstrating gem5's elasticity in simulating different ISAs.

The RISC vs. CISC debate continues to guide how researchers weigh simplicity, power efficiency, and compiler design. George [4] provided early insights into RISC vs. CISC distinctions, while Jamil [6] reasserted the theoretical underpinnings of instruction set design. Ling et al. [5] used simulation-based experimentation to question the extent to which ISA variants truly impact performance, emphasizing the importance of microarchitectural configurations, even within similar ISA families.

In gem5-specific studies, Abudaqa et al. [7] compared in-order vs. out-of-order cores for ARM and x86, highlighting microarchitectural variations in cache and pipeline design. Vikas and Talawar [8] examined cache behavior for different architectures using gem5, elucidating how differences in memory subsystems affect speed and energy consumption. Similarly, Saha et al. [3] illustrated how L1 cache size and latency can substantially influence overall throughput, reinforcing the significance of memory hierarchy.

Moreover, Lee et al. [9] showcased gem5's extensibility by integrating advanced SIMD instructions (e.g., AVX, AVX2, AVX-512) into its x86 model, illustrating that gem5 can be adapted for emerging or specialized instruction sets. Building on this adaptability, the current project expands gem5's scope even further by integrating microcontroller-oriented ISAs—AVR, ARM Thumb, Xtensa, and simplified RISC derivatives—to cover a wider range of embedded use cases.

3. Goal

The overarching goal is to develop a multi-ISA extension to gem5, focused on AVR, ARM Thumb mode, Xtensa (Espressif), and RISC. Additionally, the project seeks to build a GUI to present run-time simulation data in a visually intuitive manner. These goals can be summarized as follows:

- Provide accurate functional correctness for microcontroller ISA, respecting unique features such as Harvard architectures, shallow pipelines, limited memory, interrupt mechanisms, and specialized instruction sets.
- Extend gem5's pipeline and interrupt handling models for embedded-specific characteristics, maintaining modularity and consistency with the existing gem5 framework.
- Develop a user-friendly GUI to visualize simulation outputs, including pin monitoring, peripheral interaction, and communication logging, thus improving debugging and educational insights.

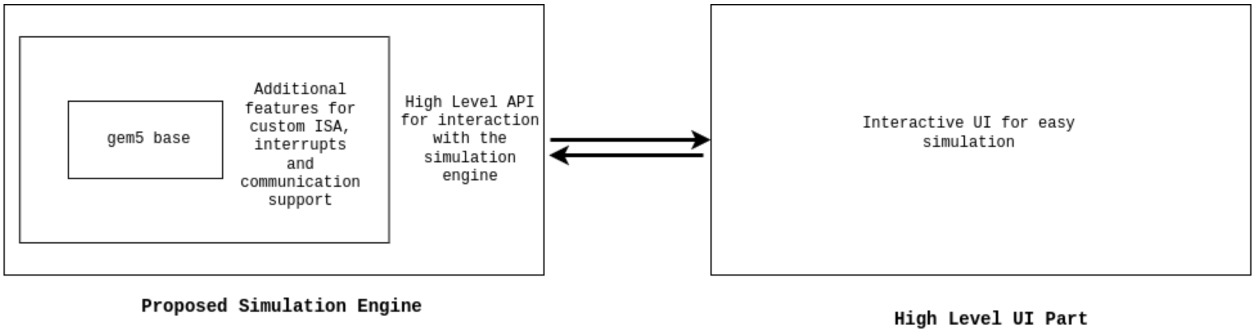
gem5 base
Additional features for custom ISA, interrupts and communication support
High Level API for interaction with the simulation engine
Interactive UI for easy simulation
Proposed Simulation Engine
High Level UI Part

Fig1. Proposed model for the simulation of microcontrollers

4. Strategies Towards the Goal

The project will be completed in multiple parts as:

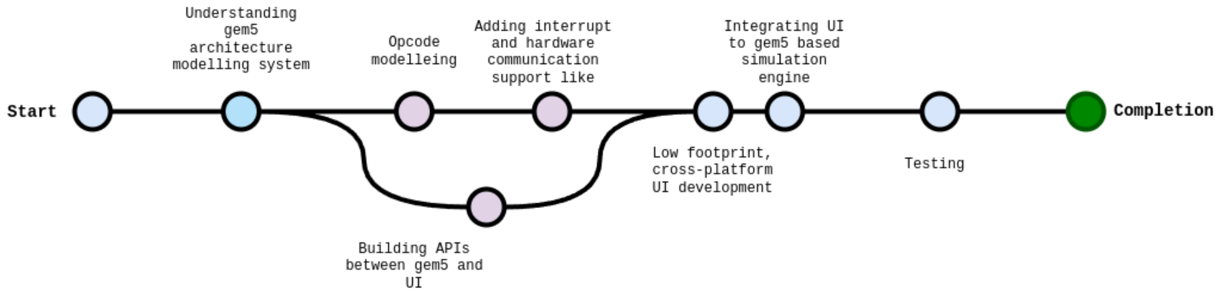
Understanding gem5 architecture modelling system
Opcode modelleing
Adding interrupt and hardware communication support like
Integrating UI to gem5 based simulation engine
Start
Completion
Low footprint, cross-platform UI development
Testing
Building APIs between gem5 and UI

Fig2. Strategy for implementing the project

4.1 Instruction Set Implementation

• AVR, ARM Thumb, Xtensa, and RISC Instruction Decoding: Define opcodes, decode logic, register sets, and addressing modes for each ISA, aligning with gem5's internal structure.

• Validation: Cross-check against official documentation, existing microcontroller simulators, and test suites (where available) to confirm correctness.

4.2 Pipeline and Microarchitectural Modeling

• Simplified Pipelines: Implement or adapt lightweight, in-order pipelines relevant for microcontrollers (e.g., single or two-stage fetch-execute).

• Specialized Features: Incorporate interrupt vectors, special I/O registers, or hardware loops (especially in Xtensa) to mirror real-world usage.

4.3 GUI Development

• Data Presentation Layer: Build a dedicated GUI that interfaces with gem5's simulation data, allowing users to see pin toggling, UART/I2C/SPI transfers, and real-time resource utilization.

• Debug and Monitoring: Develop interactive charts or timelines for performance metrics (e.g., cycles-per-instruction, memory accesses), enabling deep inspection of firmware behavior.

4.4 Validation and Benchmarking

• Assembly-Level Tests: Use known microcontroller test routines to verify instruction correctness, interrupt responses, and pipeline behavior.

• Firmware and Application Benchmarks: Run small-footprint applications (e.g., sensor polling, serial communication) to ensure the combined model (ISA + pipeline + memory + GUI) reflects realistic performance.

- Comparative Analysis: Compare gem5's simulation results for each ISA with real hardware (where available) or with other ISS (instruction set simulators) to benchmark accuracy and reliability.

5. References

[1] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy evaluation of GEM5 simulator system," in 7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2012.

[2] S. V. Bharadwaj and C. K. Vudadha, "Evaluation of x86 and ARM architectures using compute-intensive workloads," in 2022 IEEE International Symposium on Smart Electronic Systems (iSES), 2022.

[3] R. Saha, Y. P. Pundir, S. Yadav, and P. K. Pal, "Impact of size, latency of cache-L1 and workload over system performance," in 2020 International Conference on Advances in Computing, Communication Materials (ICACCM), 2020.

[4] A. D. George, "An overview of RISC vs. CISC," in 1990 Proceedings of the Twenty-Second Southeastern Symposium on System Theory, 1990.

[5] M. Ling, X. Xu, Y. Gu, and Z. Pan, "Does the ISA really matter? A simulation-based investigation," in 2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), 2019.

[6] T. Jamil, "RISC versus CISC," IEEE Potentials, vol. 14, no. 3, pp. 13–16, 1995.

[7] A. A. Abudaqa, T. M. Al-Kharoubi, M. F. Mudawar, and A. Kobilica, "Simulation of ARM and x86 microprocessors using in-order and out-of-order CPU models with gem5 simulator," in 2018 5th International Conference on Electrical and Electronic Engineering (ICEEE), 2018, pp. 317–322.

[8] B. Vikas and B. Talawar, "On the cache behavior of Splash-2 benchmarks on ARM and Alpha processors in gem5 full system simulator," in 2014 3rd International Conference on Eco-friendly Computing and Communication Systems, 2014, pp. 5–8.

[9] S. Lee, Y. Kim, D. Nam, and J. Kim, "Gem5-AVX: Extension of the Gem5 Simulator to Support AVX Instruction Sets," IEEE Access, 2024.