GEM5 Extensions: Broadening Support to Microcontrollers with GUI

Ashutosh Vishwakarma (2022UCS0083)
Guide: Dr. Subhasis Bhattacharjee
Computer Science & Engineering, Indian Institute of Technology, Jammu

***Abstract*—The primary objective of this project is to extend the Gem5 simulator's capabilities by implementing support for the AVR architecture, thereby enabling accurate simulation and analysis of AVR-based embedded systems. As a preliminary step to understand Gem5's simulation environment and functionality, a comparative performance analysis was conducted across ARM, RISC-V, and x86 architectures using matrix multiplication algorithms with varying cache configurations. This initial study helped in gaining practical experience with Gem5's simulation framework and understanding its performance metrics collection mechanisms. The core focus remains on developing the AVR extension, where fundamental ISA descriptions for basic operations like ADD and SUB have been implemented, establishing the foundation for a complete AVR simulation environment. This extension will ultimately contribute to Gem5's versatility in embedded systems research and development.**

I. Introduction

The landscape of embedded systems development and research has been significantly shaped by the availability and capability of architectural simulators. Among these, the Gem5 simulator stands out as a powerful and flexible platform for computer architecture research. However, despite its extensive support for mainstream architectures like ARM, x86, and RISC-V, there remains a notable gap in its coverage of microcontroller architectures that are fundamental to embedded systems and IoT devices.

A. Background and Motivation

Microcontrollers, particularly those based on the AVR architecture, have been cornerstone components in embedded systems education and development for decades. The AVR architecture, known for its simplicity, efficiency, and widespread use in Arduino platforms, represents an ideal candidate for educational and research purposes. However, the current limitation of Gem5 in supporting such architectures creates a significant barrier in:

- Academic research requiring detailed microcontroller simulation
- Educational environments teaching embedded systems concepts
- Development workflows requiring accurate performance analysis
- Comparative studies across different microcontroller architectures

B. Project Objectives

This project aims to bridge this critical gap by extending the Gem5 simulator to support the AVR architecture, complemented by a comprehensive GUI-based debugging and visualization system. The specific objectives include:

1) Implementation of complete AVR instruction set support within Gem5
2) Development of accurate timing and pipeline models
3) Integration of peripheral functionality typical in microcontroller systems
4) Creation of an intuitive GUI interface for simulation control and analysis
5) Validation through comprehensive benchmarking and comparison studies

C. Scope and Significance

The scope of this project encompasses:

- Full implementation of the AVR instruction set architecture
- Accurate modeling of timing and pipeline behavior
- Development of peripheral models (UART, Timer, GPIO)
- Creation of a GUI-based debugging and visualization system
- Comprehensive validation and performance analysis framework

The significance of this work lies in its potential to:

- Enable detailed microcontroller architecture research
- Provide educational tools for embedded systems teaching
- Facilitate comparative studies between different architectures
- Support embedded software development and optimization

D. Technical Approach

Our approach combines systematic architecture implementation with modern software engineering practices:

- Modular development methodology
- Extensive testing and validation procedures
- Integration with existing Gem5 frameworks
- User-centric GUI design principles

E. Report Organization

The remainder of this report is organized as follows:

Section 2 presents a comprehensive literature review and related work.

Section 3 details the system architecture and design methodology.

Section 4 describes the implementation details and technical challenges.

Section 5 presents experimental results and performance analysis.

Section 6 discusses the challenges encountered and solutions developed.

Section 7 outlines future work and potential enhancements.

Section 8 concludes with a summary of achievements and contributions.

This extension of the Gem5 simulator represents a significant step forward in microcontroller architecture simulation capabilities, providing researchers, educators, and developers with a powerful tool for understanding and optimizing embedded systems.

II. Literature Review

The evolution of architectural simulators and their applications in computer architecture research has been extensively documented in the literature. This review synthesizes key findings across simulator development, architectural comparisons, and performance analysis methodologies.

A. Architectural Simulation Frameworks

The foundation of modern architectural simulation was established with the introduction of the gem5 simulator by Binkert et al. [1], which provided a flexible and extensible platform for computer architecture research. This framework has been continuously enhanced, as demonstrated by Power et al. [2] through their gem5-gpu extension, enabling heterogeneous CPU-GPU simulation capabilities.

The accuracy of architectural simulators has been rigorously evaluated, with Butko et al. [3] providing comprehensive validation of the GEM5 simulator system. Recent developments include Lee et al.'s [4] extension of GEM5 to support AVX instruction sets, demonstrating the simulator's adaptability to new architectural features.

Alternative simulation frameworks have also emerged, including:

1) ZSim by Sanchez and Kozyrakis [5], focusing on thousand-core system simulation
2) PTLsim by Yourst [6], offering cycle-accurate x86-64 simulation
3) Sniper by Carlson et al. [7], exploring scalable multi-core simulation
4) UNISIM by August et al. [8], providing an open environment for collaborative development

B. ISA Performance Analysis

Comparative analysis of different instruction set architectures has been a crucial area of research. Bharadwaj and Vudadha [9] conducted detailed evaluations of x86 and ARM architectures using compute-intensive workloads. This work was complemented by Ling et al. [10], who investigated the fundamental question of ISA impact on system performance.

The classical RISC versus CISC debate, initially explored by George [11] and later elaborated by Jamil [12], continues to influence modern architectural decisions. Abudaqa et al. [13] extended this comparison through detailed simulation studies of ARM and x86 processors using both in-order and out-of-order CPU models.

C. Cache and Memory Performance

Memory system optimization remains a critical aspect of architectural design:

1) Saha et al. [14] analyzed the impact of cache size and latency on system performance, providing crucial insights for memory hierarchy design
2) Vikas and Talawar [15] studied cache behavior using Splash-2 benchmarks on ARM and Alpha processors, demonstrating the importance of cache optimization across different architectures

D. Research Gaps and Opportunities

The literature review reveals several key areas requiring further investigation:

1) Limited support for microcontroller architectures in mainstream simulators
2) Need for comprehensive GUI-based debugging tools for architectural simulation
3) Lack of standardized performance metrics for microcontroller simulation
4) Gap in comparative studies involving emerging embedded architectures

E. Synthesis and Research Direction

This review demonstrates the maturity of architectural simulation tools while highlighting the need for expanded support for microcontroller architectures. Our work builds upon these foundations, particularly extending GEM5's capabilities to support AVR architecture, addressing a significant gap in current simulation frameworks. The integration of GUI-based debugging tools and comprehensive performance analysis capabilities represents a natural evolution in architectural simulation technology.

III. Methodology

In this project, we proceeded in two phases. First, we explored Gem5's capabilities and developed a deeper understanding of the simulator's internal workings. To do this, we studied the impact of different instruction set architectures (ISAs) on execution speed and cache miss rates under a given workload. We used the Gem5 simulator to run a set of benchmarks and collected data on execution time and cache performance. This data was then analyzed to evaluate how different ISAs influence performance. The insights gained helped us identify the strengths and limitations of each ISA in various scenarios.

In the second phase, we worked on extending Gem5 to support the AVR architecture. We began by familiarizing ourselves with the existing codebase and identifying the

components that required modification. We then implemented support for basic AVR instructions such as `add` and `sub`.

A. Testing the gem5 simulator

For the testing we used the algorithm of multiplying two matrices in row major format. The algorithm was written in C and compiled for different architectures.

1) Compilation: The used code was:

```c
// Multiplication of two matrices
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int** createRandomMatrix(int n) {
    int** matrix = malloc(n * sizeof(int*));
    for (int i = 0; i < n; i++) {
        matrix[i] = malloc(n * sizeof(int));
        for (int j = 0; j < n; j++) {
            matrix[i][j] = (rand()%2==1?-1:1)*rand()
     % MAX; // capping the values
        }
    }
    return matrix;
}

int** createZeroMatrix(int n) {
    int** matrix = malloc(n * sizeof(int*));
    for (int i = 0; i < n; i++) {
        matrix[i] = calloc(n, sizeof(int));
    }
    return matrix;
}

void printMatrix(int** matrix, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%3d ", matrix[i][j]);
        }
        printf("\n");
    }
}

void multiplyMatrices(int** A, int** B, int** C, int
     n) {
    for (int i = 0; i < n; i++) {           // Row
    of A
        for (int j = 0; j < n; j++) {       //
    Column of B
            for (int k = 0; k < n; k++) {   //
    Iterate over row-col
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

void freeMatrix(int** matrix, int n) {
    for (int i = 0; i < n; i++)
        free(matrix[i]);
    free(matrix);
}

int main() {
    int n = DIM;
    srand(time(NULL));

    int** A = createRandomMatrix(n);
    int** B = createRandomMatrix(n);
    int** C = createZeroMatrix(n);

    printf("Matrix A:\n");
    printMatrix(A, n);

    printf("\nMatrix B:\n");
    printMatrix(B, n);

    multiplyMatrices(A, B, C, n);

    printf("\nMatrix A x B:\n");
    printMatrix(C, n);

    freeMatrix(A, n);
    freeMatrix(B, n);
    freeMatrix(C, n);

    return 0;
}
```

Listing 1. Row major matrix multiplication algorithm

The used code was compiled with two constants `DIM` and `MAX`. The constant `DIM` is the size of the matrix and the constant `MAX` is the maximum absolute value of the elements in the matrix. The code was compiled with the following command:

```
    <compiler> -static -o matrix_mult matrix_mult.c
    -DDIM=<dimension> -DMAX=<max_value>
```

Listing 2. Compilation Command

The code was compiled with the `-static` flag to ensure that the code is statically linked. This is important because the gem5 simulator does not support dynamic linking. The code was compiled with the `-D` flag to define the constants `DIM` and `MAX`.

The code generates two random matrices of size `DIM` and multiplies them. The result is printed to the standard output. The code was compiled for the following architectures:

Architecture	Compiler
x86	gcc
ARM	aarch64-linux-gnu-gcc
RISCV	riscv-linux-gnu-gcc

TABLE I
LIST OF ARCHITECTURES AND THEIR COMPILERS

2) Gem5 System Architecture Design: The gem5 simulation has a modular design. It closely resembles a real system. The main part on which simulation run is a board. There are multiple types of boards in gem5. A board holds different components of a system namely clock, processor and memory. The binary resource is also loaded to this board. The Python API for gem5 has the following definition of a board:

```
processor = SimpleProcessor(
    cpu_type=CPUTypes.TIMING,
    num_cores=1,
    isa=isa
)
```

Listing 3. Gem5 Processor Definition

```
memory = SingleChannelDDR3_1600("1GiB")
```

Listing 4. Gem5 Memory Definition

```
cache_hierarchy = NoCache()
```

Listing 5. Gem5 Memory Definition

```
cache_hierarchy = PrivateL1PrivateL2CacheHierarchy(
    l1d_size="32kB",
    l1i_size="32kB",
    l2_size="128kB"
)
```

Listing 6. Gem5 Memory Definition

```
board = SimpleBoard(
    clk_freq="1GHz",
    processor=processor,
    memory=memory,
    cache_hierarchy=cache_hierarchy
)
```

Listing 7. Gem5 Board Definition

```
binary = BinaryResource("<path_to_binary>")
board.set_se_binary_workload(binary)
```

Listing 8. Gem5 Memory Definition

```
simulator = Simulator(board=board)
simulator.run()
```

Listing 9. Gem5 Memory Definition

REFERENCES

[1] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[2] J. Power, J. Hestness, M. S. Orr, M. D. Hill, and D. A. Wood, "gem5-gpu: A heterogeneous CPU-GPU simulator," *IEEE Computer Architecture Letters*, vol. 14, no. 1, pp. 34–36, 2015.

[3] 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, pp. 1–7.

[4] S. Lee, Y. Kim, D. Nam, and J. Kim, "Gem5-AVX: Extension of the Gem5 simulator to support AVX instruction sets," *IEEE Access*, vol. 12, pp. 1234–1245, 2024.

[5] D. Sanchez and C. Kozyrakis, "ZSim: Fast and accurate microarchitectural simulation of thousand-core systems," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, 2013, pp. 475–486.

[6] M. T. Yourst, "PTLsim: A cycle accurate full system x86-64 microarchitectural simulator," in *2007 IEEE International Symposium on Performance Analysis of Systems & Software*, 2007, pp. 23–34.

[7] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–12.

[8] D. August, J. Chang, S. Girbal, D. Gracia-Perez, G. Mouchard, D. A. Penry, O. Temam, and N. Vachharajani, "UNISIM: An open simulation environment and library for complex architecture design and collaborative development," *IEEE Computer Architecture Letters*, vol. 6, no. 2, pp. 45–48, 2007.

[9] 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, pp. 234–239.

[10] 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, pp. 1–6.

[11] A. D. George, "An overview of RISC vs. CISC," in *1990 Proceedings of the Twenty-Second Southeastern Symposium on System Theory*, 1990, pp. 447–451.

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

[13] 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.

[14] 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, pp. 45–50.

[15] 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.