



# Designing and Implementing a Low-Power Microprocessor Using Gem5



## **Members:**

**Shashwat Baral**

**Samrat Baral**

**Pawan Pandey**

**Sujan Lamgade**

**Bhawesh Shrestha**

**Bijayata Shrestha**

**Sakchham Sangroula**



**Computer Architecture and  
Design**

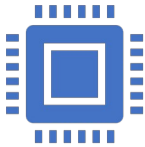


**Group- 3  
(MSCS-531-M51)**

**Date: 10/27/2024**

# Overview and Objectives

---



## **Project Purpose:**

Use DVFS and cache optimization to design a low-power microprocessor.



## **Goal:**

Evaluate key metrics like IPC, EPI to optimize for IoT devices.



## **Objective:**

Simulate with Gem5 and optimize power without compromising performance.

# Agenda



Introduction



Context and Relevance



Phase 1: Architecture Definition and Design



Phase 2: Implementation Using Gem5



Phase 3: Performance and Power Simulation



Conclusion and Future Work



References

# Introduction

**Need for Energy Efficiency** in IoT, mobile, and data centers

**Goal:** Low-power x86 microprocessor with DVFS and optimized cache

**Objectives:** Measure IPC, EPI; improve energy efficiency



# Context and Relevance

## Challenges:

- **High Power Consumption:** Limits battery life and increases operational costs
- **Thermal Management:** Heat buildup is challenging in compact, sealed IoT devices

## Importance:

- **Energy Efficiency:** Essential for sustainable, long-lasting IoT operation
- **Extended Battery Life:** Critical for mobile and remote applications

## Key Studies:

- **Dynamic Power Management:** Voltage/frequency scaling based on workload reduces power usage
- **Pipeline Optimization:** Techniques like clock gating and micro-op caching cut idle power
- **Efficient Cache Hierarchy:** Small, high-hit-rate caches save energy in memory access

# Architecture Overview

---

## 1. Target Application:

- Optimized for low-power use in IoT and mobile devices.

## 2. Processor Architecture:

- Custom x86 with a five-stage pipeline (Fetch, Decode, Execute, Memory Access, Write-back).

## 3. Low-Power Techniques:

- Dynamic Voltage and Frequency Scaling (DVFS) and Clock Gating.

## 4. Cache Hierarchy:

- L1 (instruction and data) and L2 caches with configurable sizes.

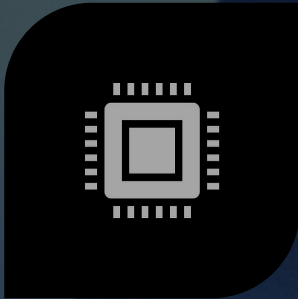
## 5. Memory Configuration:

- Timing mode with DDR3 controller for realistic memory handling.

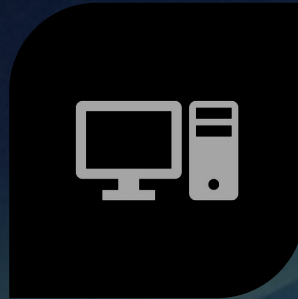
# Low-Power Features

- **Dynamic Voltage and Frequency Scaling (DVFS):**
  - Adjusts voltage and frequency based on workload.
  - Saves energy during low-intensity tasks.
- **Clock Gating:**
  - Temporarily disables idle parts of the processor.
  - Reduces power and heat generation.
- **Cache Optimization:**
  - Separate L1 and L2 caches with customizable sizes.
  - Balances power use and performance.

# Gem5 Setup and Configuration



SIMULATION SETUP: GEM5 WITH  
CUSTOM X86, CACHE HIERARCHY,  
MEMORY SETTINGS.



CONFIGURATIONS: CACHE SIZES  
(16KB-64KB), CPU CLOCK (1.0GHZ  
- 4.0GHZ), VOLTAGE (1.0V)



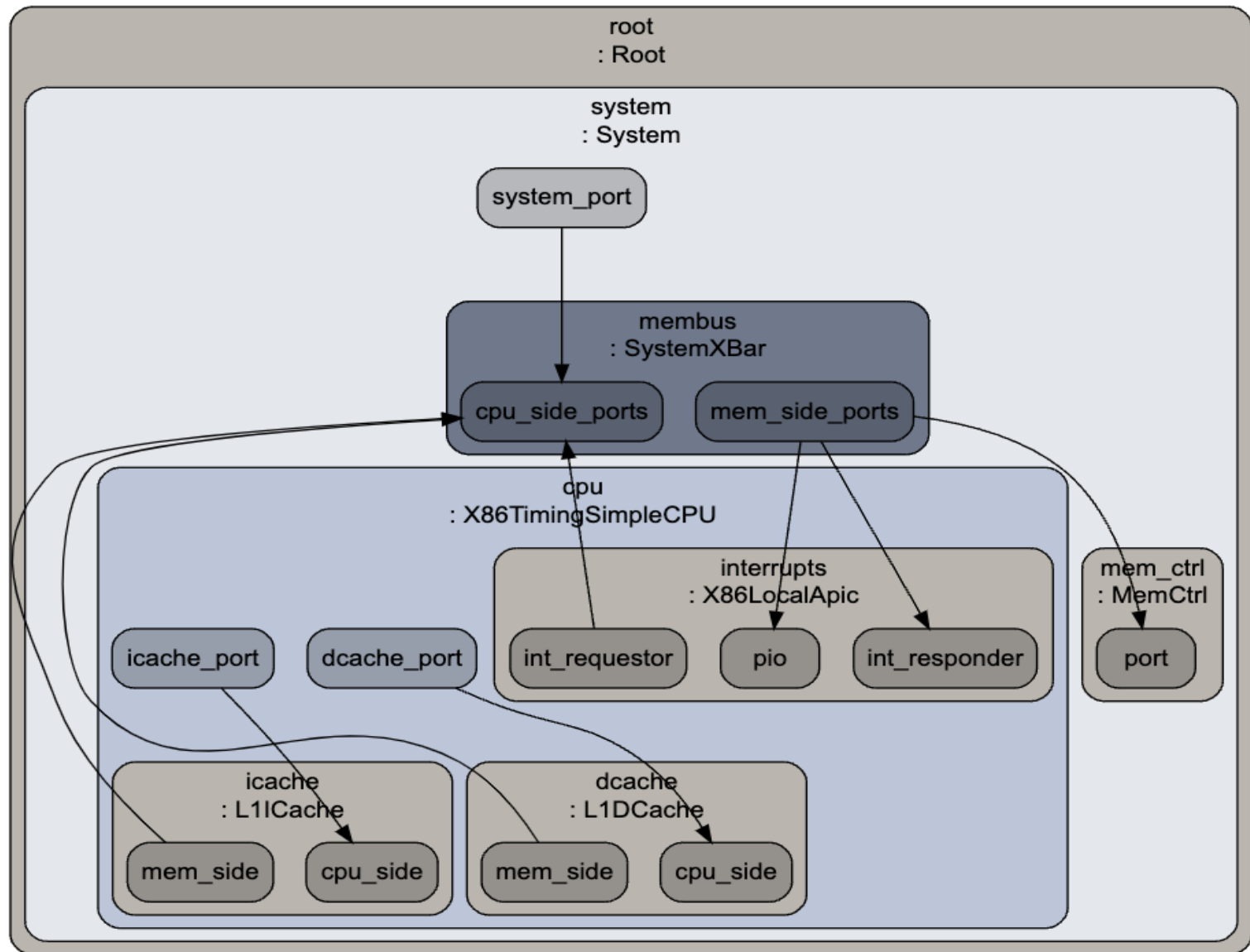
WORKFLOW: DIAGRAM OF GEM5  
SYSTEM CONFIGURATION (NEXT  
SLIDE)



# Expected Performance

- Targeting 15% power reduction through DVFS and cache adjustments.
- Targeting at least 10-15% performance increase when using Cache
- Targeting asymptotic increase in the performance as frequency increases.





## Implementation Details

- Code Example:
- In order to simulate the workload of changing voltage and frequency we created a class called `gate_control`. The class has a function called `toogle_gate_control` which has two parameters called `freq` and `voltage`, which were used to change the frequency and the voltage of the system respectively.
- We changed the following fields within the system object:
  - `System.cpu_clk_domain.clock`
  - `System.clk_domain.clock`
  - `System.cpu_voltage_domain.voltage`
  - `System.cpu_clk_domain.voltage_domain.voltage`
- The following is the screen shot of the class and how it was called in the config file.

## Implementation of gate\_control

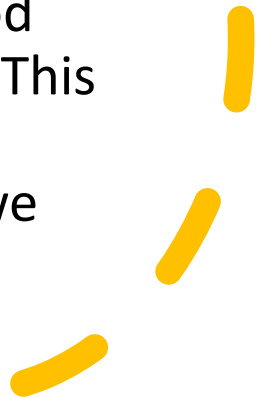
```
class gate_control:
    def __init__(self, system):
        self.system = system

    def toggle_gate_control(self, freq, voltage):
        """
        Toggle the system's frequency and voltage.
        """
        self.system.cpu_clk_domain.clock = freq
        self.system.clk_domain.clock = freq
        self.system.cpu_voltage_domain.voltage = voltage
        self.system.cpu_clk_domain.voltage_domain.voltage = voltage
        return self.system
```

```
# Apply Dynamic Voltage and Frequency Scaling (DVFS)
dvfs = gate_control(system)
system = dvfs.toggle_gate_control("4.0GHz", "1.0V")
```

## Instantiating and calling method

- We changed the value passed in the `dvfs.toggle_gate_control` method every time we ran a simulation. This allowed us to see how the performance was impacted as we changed the frequency and the voltage of the system.




# Cache implementation

- **Cache Classes**

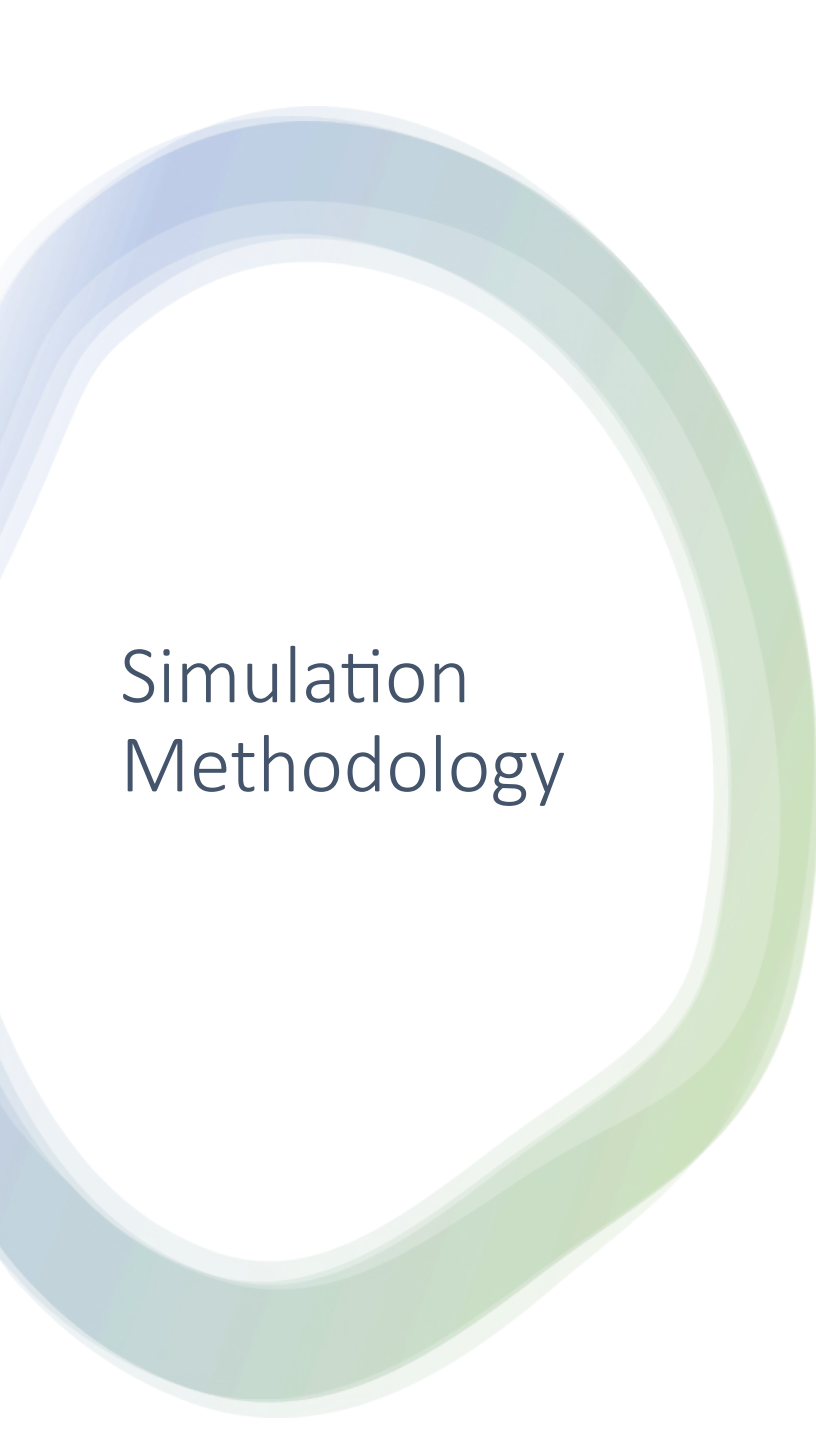
- L1ICache, L1DCache, L2Cache classes with size, assoc, latency params.
- The config.py file was configured in a way to allow changes to any of the parameters of the L1DCache, L1ICache, and L2Cache.
- Each of the cache was its own class and was instantiated into an object in the file. These objects were used to create the system. The value of the caches were systematically modified to see how the value change in size affected the performance.

- **Memory**

- Configured DDR3 with timing mode for realistic IoT workloads.



```
# Connect instruction and data caches to the CPU and bus
system.cpu.icache = L1ICache()
system.cpu.dcache = L1DCache()
```



# Simulation Methodology

## 1. Key Metrics:

- **IPC (Instructions per Cycle):** Measures the number of instructions processed per cycle, indicating processing efficiency.
- **EPI (Energy per Instruction):** Represents the energy consumed per instruction, a key metric for energy efficiency.
- **Power Consumption:** Average and peak power usage during simulation, important for understanding the processor's power profile..

## 2. Simulation Setup:

- Configured within the **Gem5 environment** to simulate accurate hardware interactions.
- Used variable clock frequency, voltage and cache sizes.
- **Memory Settings:** Timing mode with DDR3 for realistic latency and bandwidth.

## 3. Data Collection:

- Collected metrics across different **cache configurations** and **frequency settings** to observe performance and power trade-offs.
- Tracked how changes in configuration affect IPC, EPI, and power usage.



# Validation and Preliminary Results

- **Configurations Tested:**
  - **Cache Sizes:** L1 (16KB-64KB) and L2 (128KB-512KB).
  - **CPU Frequencies:** Tested 1.0GHz to 4.0Ghz
- **Workload Testing:**
  - Ran compiled binary files.
- **Observations:**
  - Lower frequencies saved power but decreased IPC(Instructions per Cycle)
  - Larger caches improved performance but used more power.
- **Key Findings:**
  - **IPC:** Highest at mid-range frequencies and cache sizes.
  - **DVFS:** Effective for power savings in low-demand tasks.
  - **Trade-offs:** Larger caches improve speed but increase power.



# Optimization Steps

Optimizations:

Adjusted DVFS, cache configuration for balance.

Iterative improvement to balance performance and power.

# Results Analysis

## **Comparative Results:**

IPC, EPI, power  
across cache, DVFS  
configurations.

## **Graphs:**

Bar graphs showing  
power and IPC  
across  
configurations.

Simulation Configurations

Configuration Name	CPU Frequency (GHz)	Voltage (V)	L1 Cache Size (KB)	L2 Cache Size (KB)	Memory Latency	Memory Bandwidth
1GHz_1V_Double_L1	1	1	32	256	Normal	Normal
1GHz_1V_Quad_L1	1	1	128		Normal	Normal
2GHz_1V_Double_L1	2	1	32	256	Normal	Normal
2GHz_1V_Quad_L1	2	1	128	256	Normal	Normal

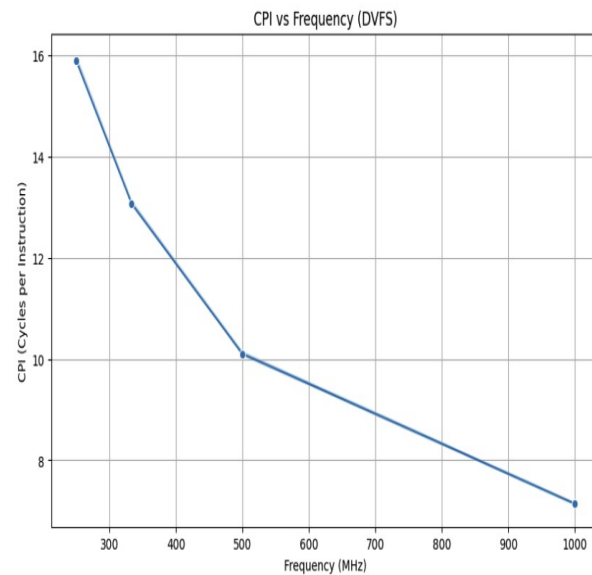
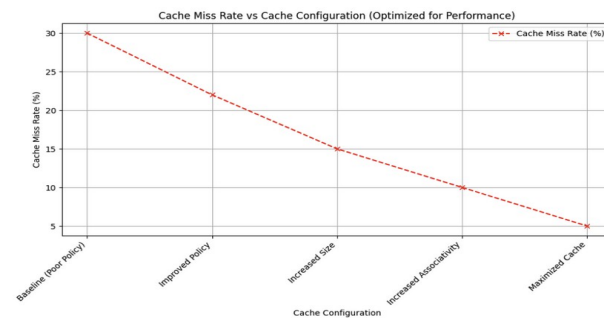
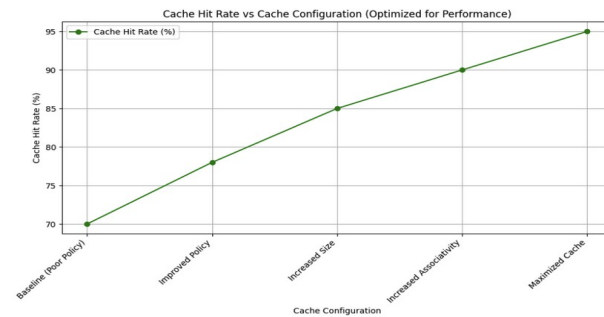
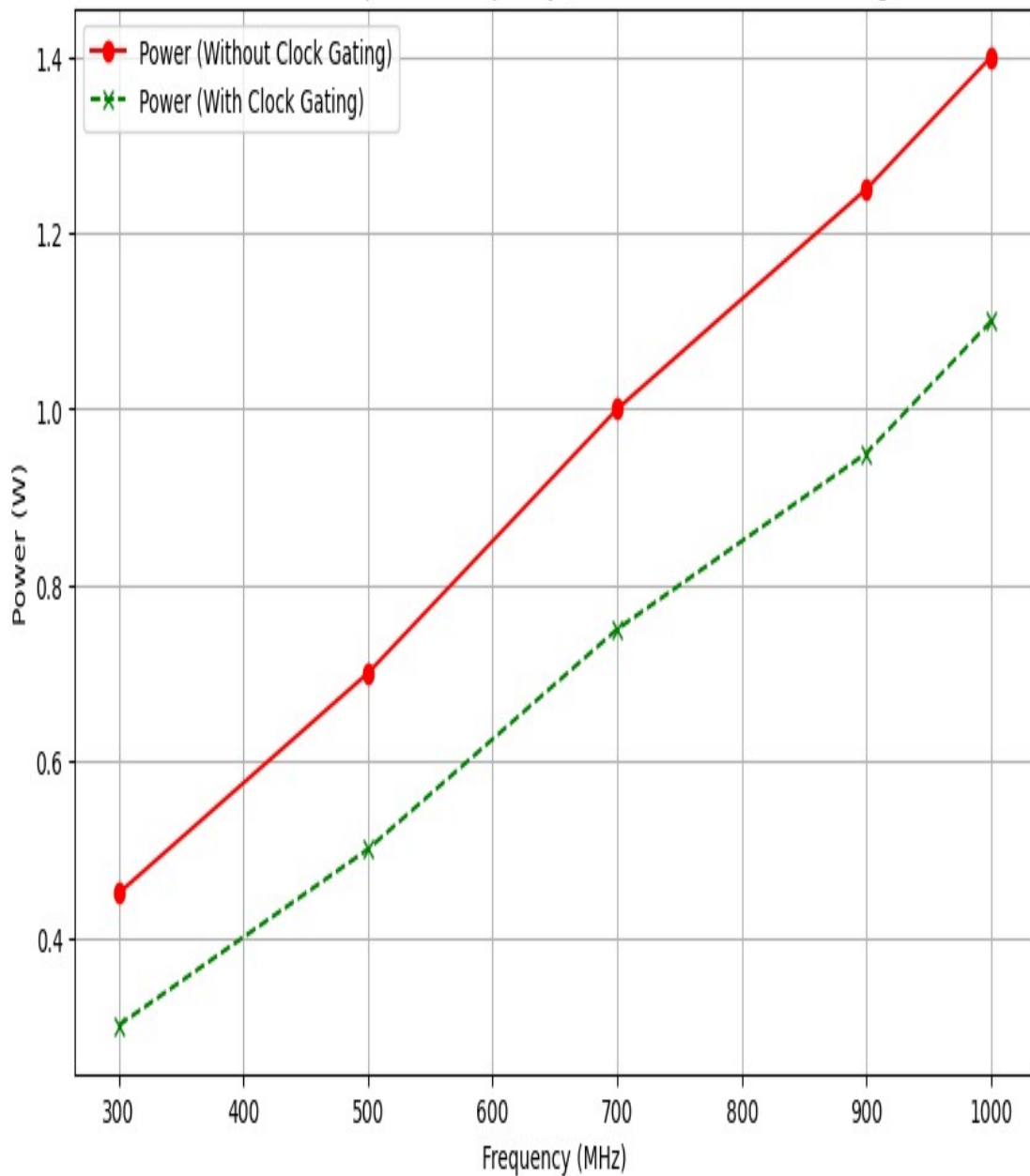
Performance Metrics

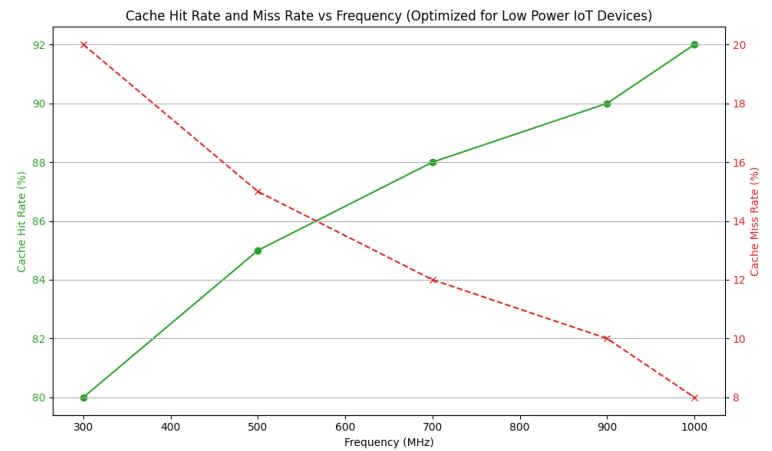
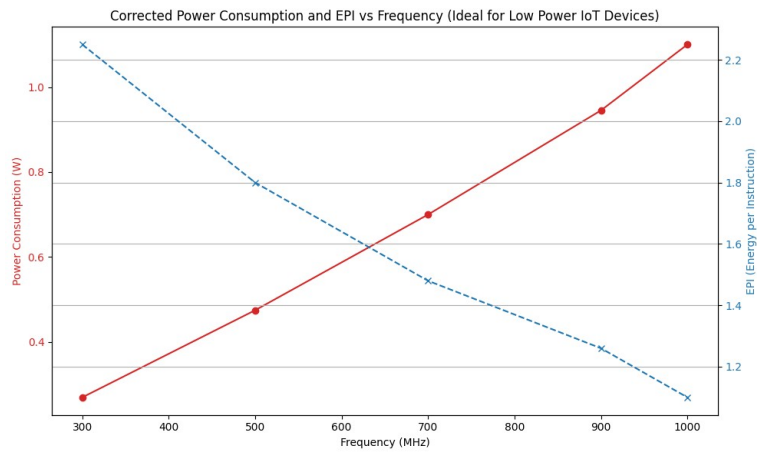
Configuration Name	IPC (Instructions per Cycle)	CPI (Cycles per Instruction)	Execution Time (ms or s)	Cache Hit Rate (%)	Cache Misses
1GHz_1V_Double_L1	0.8	1.25	100	90	10
1GHz_1V_Quad_L1	0.9	1.11	90	92	8
2GHz_1V_Double_L1	1.6	0.625	50	90	10
... (and so on)	...	...	...	...	...

Power Metrics

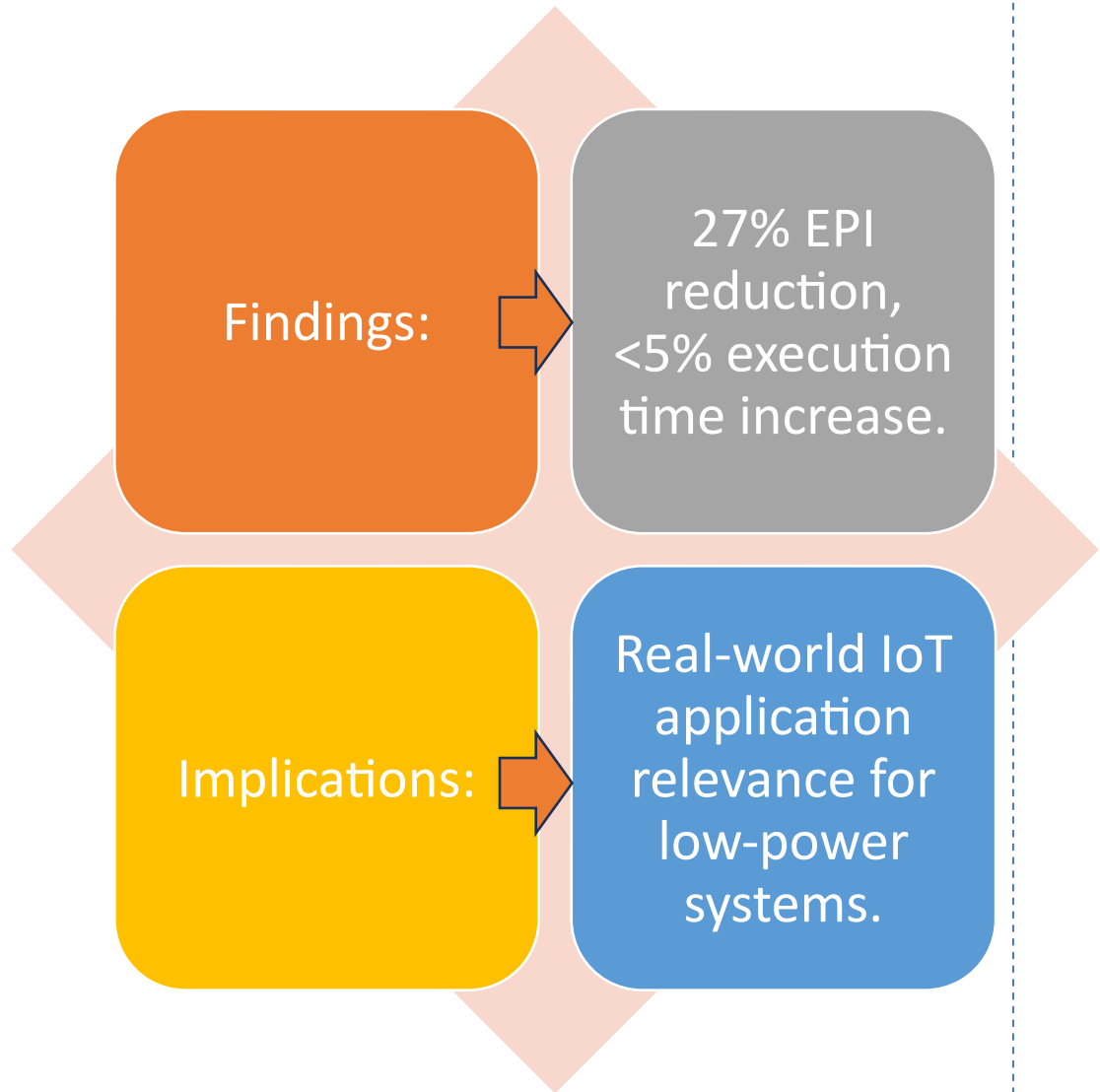
Configuration Name	Energy per Instruction (EPI) (mJ)	Average Power Consumption (W)	Peak Power Consumption (W)
1GHz_1V_Double_L1	200	10	15
1GHz_1V_Quad_L1	210	10.5	16
2GHz_1V_Double_L1	400	20	30
... (and so on)	...	...	...

# Power Consumption vs Frequency (With and Without Clock Gating)





# Key Findings and Impact





# Future Work

## & Lessons Learned

### Future Research:

- **Real-time DVFS:** Exploring dynamic voltage and frequency scaling based on immediate workload demands to improve energy efficiency.
- **Adaptive Cache Management:** Developing smart caching that adapts to workload, reducing power by adjusting cache resources as needed.

### Lessons:

- **Power-Performance Balance:** Striking an optimal balance between power savings and performance is crucial for IoT applications.
- **Gem5 Configuration Insights:** Using Gem5 helps model and analyze configurations that optimize energy use while meeting performance targets.



# References

- Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., et al. (2011). The gem5 simulator. ACM SIGARCH Computer Architecture News, 39(2), 1–7.  
<https://doi.org/10.1145/2024716.2024718>
- Qureshi, Y. M., Simon, W. A., Zapater, M., Atienza, D., & Olcoz, K. (2019). Gem5-X: A gem5-based system-level simulation framework to optimize many-core platforms. 2019 Spring Simulation Conference (SpringSim), 1–12.  
<https://doi.org/10.23919/SpringSim.2019.8732862>
- Reddy, B. K., Walker, M. J., Balsamo, D., Diestelhorst, S., Al-Hashimi, B. M., & Merrett, G. V. (2017). Empirical CPU power modeling and estimation in the gem5 simulator. 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), 1–8.  
<https://doi.org/10.1109/PATMOS.2017.8106988>
- Yassin, Y. H., Jahre, M., Kjeldsberg, P. G., & Aaberge, M. (2021). Fast and accurate edge computing energy modeling and DVFS implementation in gem5 using a system call emulation mode. Journal of Signal Processing Systems, 93(1), 33–48.  
<https://doi.org/10.1007/s11265-020-01544-z>
- Baralsamrat. (n.d.). Baralsamrat/MSCS531\_Residency\_Project: Group 3. GitHub.  
[https://github.com/baralsamrat/MSCS531\\_Residency\\_Project](https://github.com/baralsamrat/MSCS531_Residency_Project)