"Adaptive ML-Assisted DVFS Power Management Unit for Mini-SoC"

P. Anil Kumar

Assistant Professor

Department of Electrical and Electronics Engineering (EEE)

Matrusri Engineering College

Hyderabad, Telangana, India

Email: anilkumar.palarapu@matrusri.edu.in

B. Abhishek (1608-22-734-026)

Department of Electrical and Electronics Engineering (EEE)

Matrusri Engineering College

Hyderabad, Telangana, India

Email: abhishekbunga515@gmail.com

M. Sukumar (1608-22-734-307)

Department of Electrical and Electronics Engineering (EEE)

Matrusri Engineering College

Hyderabad, Telangana, India

Email: sukumarmalkajgiri2005@gmail.com

T. Vasanth Singh (1608-22-734-317)

Department of Electrical and Electronics Engineering (EEE)

Matrusri Engineering College

Hyderabad, Telangana, India

Email: vasanthsinghthakur26@gmail.com

CHAPTER 1 — INTRODUCTION

1.1 Background

The push for faster performance and longer battery life in today's embedded systems means power efficiency isn't optional—it's become a core challenge in designing SoC architectures. With more devices running on batteries and portability being key, there's real pressure to get power consumption down, but not at the cost of speed.

Dynamic Voltage and Frequency Scaling (DVFS) plays a big role here. The idea is simple: adjust the system's voltage and clock frequency on the fly, matching them to what the workload actually needs at any given moment. Because dynamic power goes up with the square of voltage and rises linearly with frequency, controlling both is pretty effective for saving energy.

But there's a catch. The usual DVFS techniques only react to changes—they don't see them coming. So, they end up making decisions based on what's happening now (or even a moment ago), which means if the workload suddenly jumps, the system is always a step behind. This leads to short periods of wasted power and even performance hiccups.

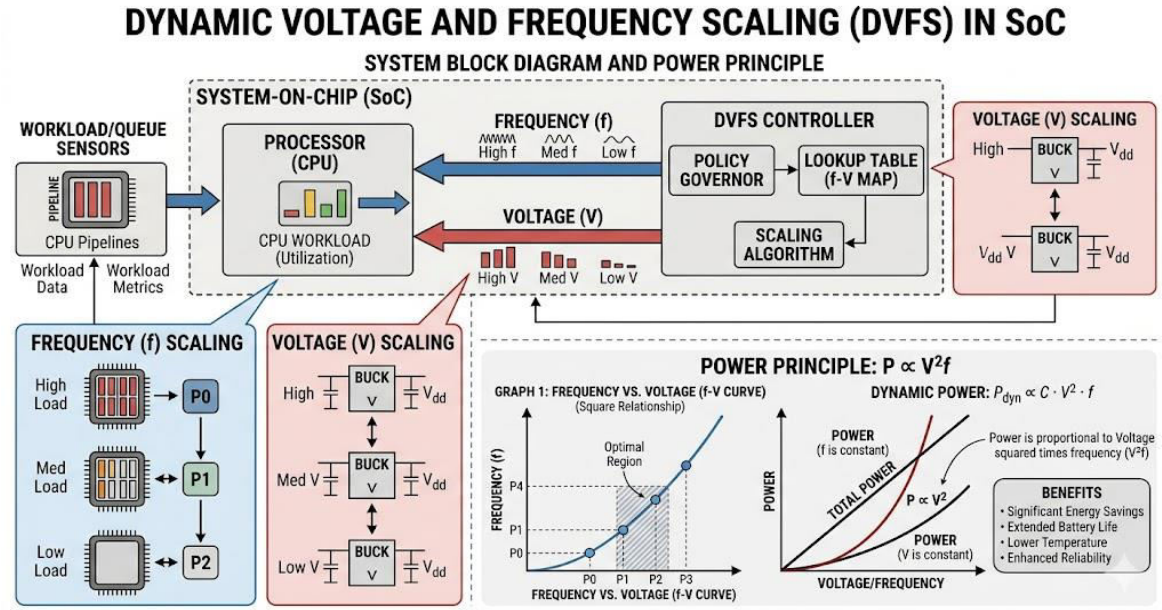

Fig. 1. Basic Concept of Dynamic Voltage and Frequency Scaling(DVFS)

1.2 Problem Statement

Current DVFS systems just don't look ahead—they only adjust frequency after spotting a change in workload. That causes delays and, frankly, wastes power. So we're left with three problems:

• Power is used inefficiently during workload shifts.

• The system is slow to adjust frequencies, adding lag.

• Overall performance takes a hit—it never fully keeps up.

These issues make it clear: we need smarter DVFS. The system should predict what's coming and adjust in advance.

1.3 Proposed Approach

That's where this project steps in. We're bringing a lightweight neural network into the DVFS controller for Mini-SoCs, so the system becomes predictive instead of reactive. The neural net learns typical workload patterns and can raise or lower frequencies before they spike, letting it get ahead of the curve.

The whole setup is a full hardware-software co-design. We built the model in TensorFlow, turned it into hardware with NNgen, and checked its logic through Verilator simulations. To top it off, we measured power usage using SAIF-based estimation in Yosys.

1.4 Contributions of the Work

Here's what this work brings to the table:

• Predictive DVFS for Mini-SoCs: Developed a frequency scaling system that uses machine learning to anticipate system needs rather than just follow them.

• Efficient Model Integration: Packed a neural network into hardware with the NNgen tool, so it stays slim and fits into resource-limited designs.

• FSM-Based Control Logic: Built a finite state machine to handle DVFS control, keeping transitions smooth.

• Verilator Performance Checks: Verified the system's functions with high-speed, detailed hardware simulation.

• Real Power Analysis: Used SAIF data for precise power measurements in Yosys—no guesswork.

• Direct Comparisons: Ran side-by-side tests against traditional DVFS setups to show exactly where and how the new system outperforms them.

SECTION II — LITERATURE REVIEW

2.1 Overview of DVFS Techniques

DVFS has been the standard tool to trim power in digital systems for years. It tweaks voltage and frequency on the fly, using feedback from the workload, overall usage, or some simple rules and thresholds.

The main approaches people use include:

• Static threshold-based methods

• Feedback control loops

• Heuristic and rule-based algorithms

Easy to set up? Definitely. Adaptable? Not really. Because these systems only respond to what's happening now—or what's just happened—they're slow to react when things change suddenly.

2.2 Limitations of Conventional DVFS

The old methods all share one weakness: they're reactive. They only kick in after load has already changed. That creates a few headaches:

• Adjustment lag: When workload spikes, system frequency ramps up too late, so you get short-term performance drops.

• Wasted power: When things calm down, the system stays at high power too long—burning energy unnecessarily.

• Inefficient resource use: Since the system's always chasing the curve, it can't manage resources effectively for what's ahead.

In embedded SoCs and AI-driven systems, where activity is unpredictable, these delays are even harder to ignore.

2.3 Machine Learning-Based Power Optimization

Now, to overcome these sticking points, researchers are using machine learning to make smarter, more proactive power management decisions. Instead of always reacting, ML models look at trends in data and predict where the workload's going—so the system can get ready in advance.

There's momentum in a few key directions:

• Regression models for workload predictions

• Reinforcement learning that lets systems adapt through experience

• Neural networks that directly control dynamic scaling

Real-world tests prove these ML techniques work better: they cut out lag and lead to real power savings because they anticipate what's needed, not just what's already happened.

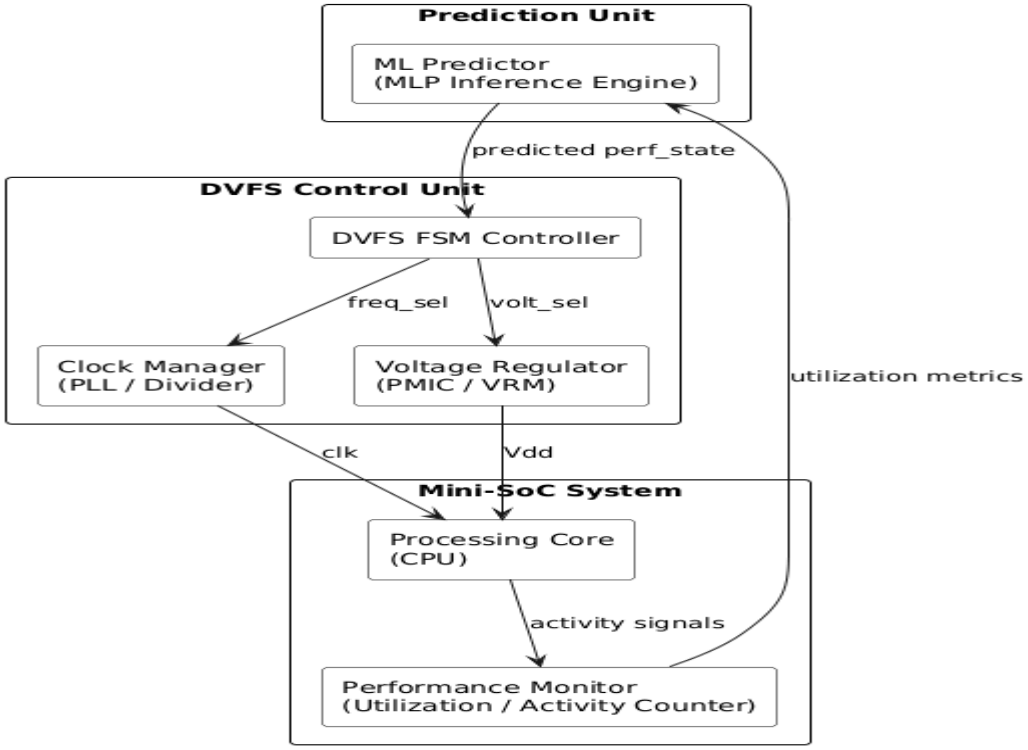

Fig. 2. Architecture of machine learning-based DVFS system

2.4 Existing ML-Based DVFS Approaches

A bunch of studies tackle ML-powered DVFS, usually by training a model to predict system demands, which then guides DVFS adjustments. But there are still obstacles:

• Too heavy for small SoCs: Many models eat up too much power or are just too large for lightweight chips.

• Stuck in simulation: Most projects run only in software and never make it to hardware, so real performance goes untested.

• Power analysis lacks detail: Evaluations often depend on high-level estimates, not real hardware data with accurate switching activity.

2.5 Research Gap

Despite the promise of ML-based DVFS, there's still a shortage of true hardware-ready, lightweight models for resource-constrained SoCs.

Here's what's missing:

• Simple, deployable hardware models that won't overwhelm tiny chips

• RTL-level integration for rigorous testing in real hardware contexts

• Detailed, low-level power analysis using actual switching data (not broad estimates)

• Apples-to-apples benchmarking between ML-powered and traditional DVFS methods under the same conditions

2.6 Motivation for Proposed Work

This project tackles those gaps directly. We've built an ML-powered DVFS that's both efficient and hardware-friendly—a framework that bakes a neural net straight into the DVFS controller for Mini-SoCs, making use of NNgen to do it.

What makes this work different? Four things:

• It's designed for actual hardware, not just simulations.

• The neural network is light on overhead, keeping the overall system energy-efficient.

• We use professional, industry-standard tools for power analysis, not rough estimates.

• We put our system head-to-head with classic DVFS so we get clear, fair results on what really works better.

SECTION III — PROPOSED METHODOLOGY

3.1 System Overview

Here's the big picture: This system brings together Adaptive Machine Learning and Dynamic Voltage and Frequency Scaling (DVFS) inside a Mini-SoC setup. The main goal? Change the operating frequency ahead of time based on what the workload looks like it'll be, so we get better power use and smoother performance.

The whole architecture breaks down into three parts: a unit that watches system activity, a machine learning model that makes predictions, and a DVFS control unit. First, the workload monitoring unit gathers system activity numbers (things like switching behavior or utilization). Those go into the ML model, which predicts what's coming. The DVFS controller then takes this output and adjusts the frequency to match.

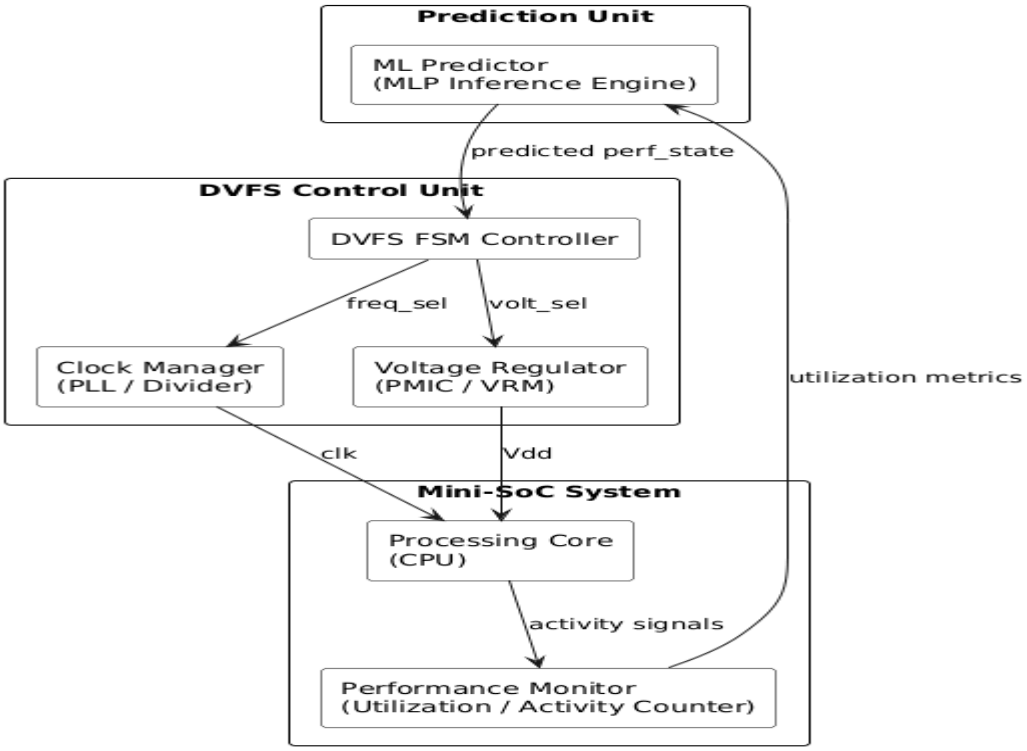

Fig. 3. Proposed ML-assisted DVFS system architecture

3.2 Workload Monitoring and Dataset Generation

To monitor system workload, we track things like utilization and switching patterns during simulation. We record these metrics under a variety of operating conditions, so the dataset covers lots of different workload types.

Each sample in the dataset pairs a set of activity features with the ideal frequency level for those conditions. We build this dataset using multiple simulation runs to capture plenty of variability.

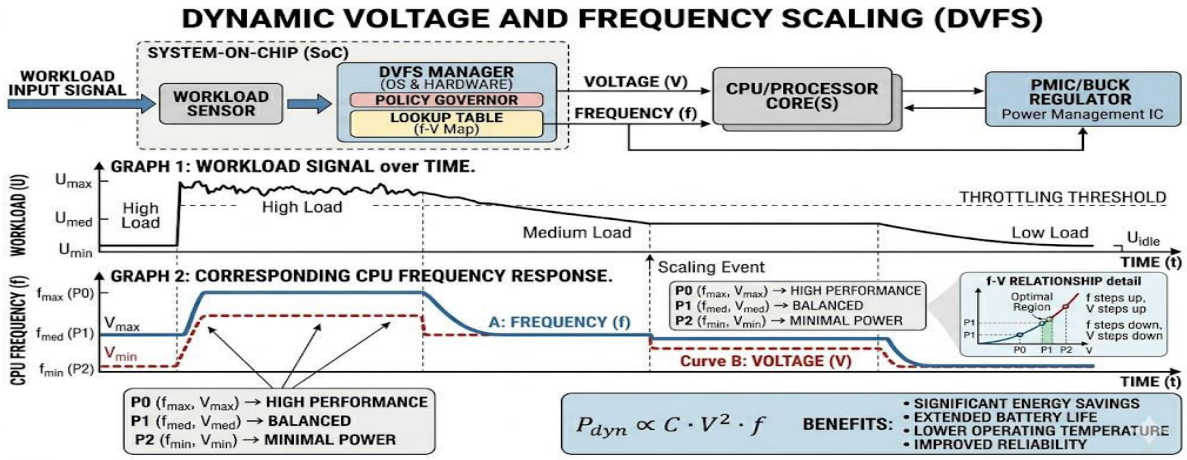

Fig. 4. Dynamic behavior of DVFS based on workload variations

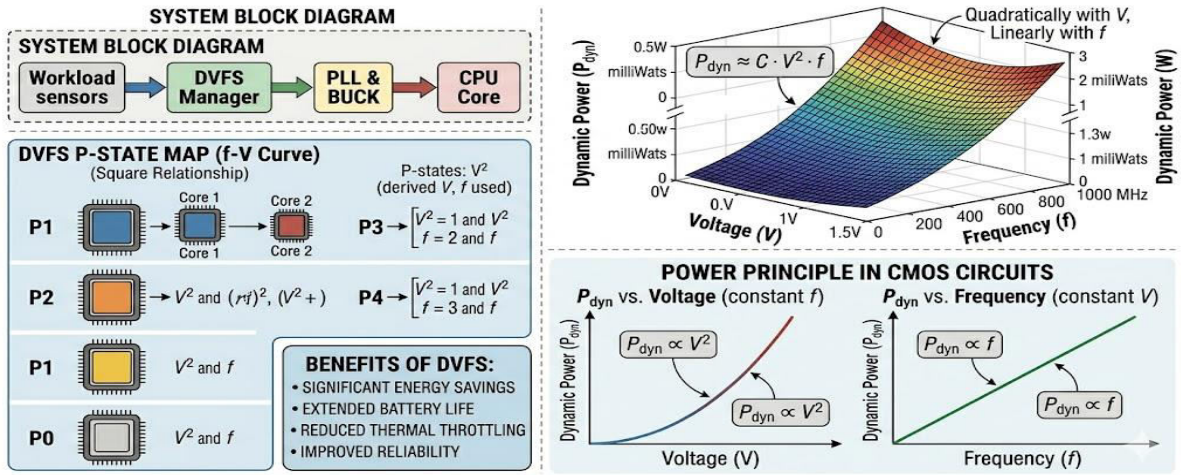

Fig. 5. Power consumption variation with voltage and frequency

3.3 Machine Learning Model Design

For predictions, we use a lightweight Multi-Layer Perceptron (MLP). It's a simple neural network that fits well into hardware. The model starts with an

input layer that grabs workload features, has a hidden layer with 8 to 16 neurons, and ends with an output layer that gives the best frequency level. We train it in TensorFlow using supervised learning, then convert it to hardware using NNgen. This way, the ML predictor fits right into the Mini-SoC's hardware.

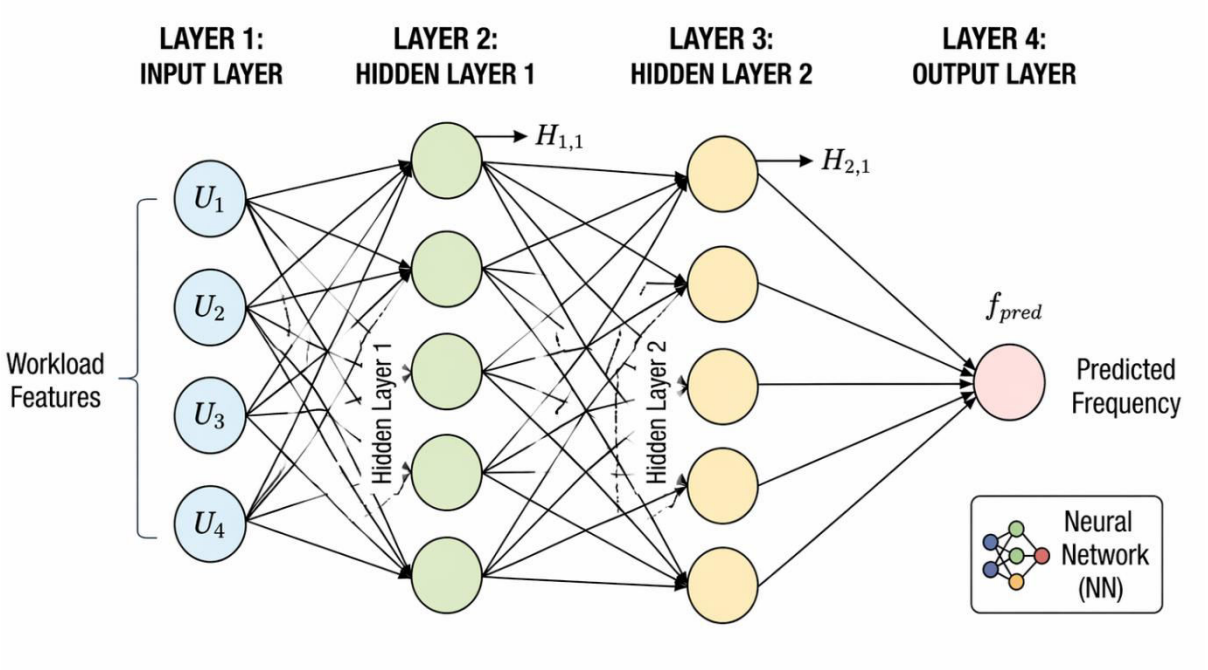

Fig. 10. Architecture of the multilayer perceptron (MLP) model

3.4 DVFS Control Unit

The DVFS controller uses a straightforward Finite State Machine (FSM). It manages a set of fixed frequency levels and moves between them based on the ML model's predictions. The FSM makes sure frequency changes happen smoothly — no wild swings that could harm performance.

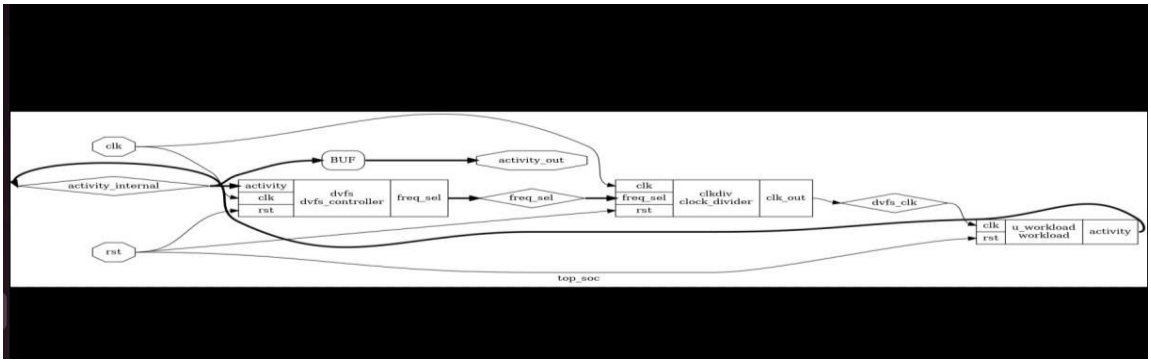

Fig. 6. RTL block diagram of conventional DVFS system

3.5 Hardware Implementation Flow

We use a classic hardware-software co-design flow. First, we generate the dataset, then train the ML model. Once the model's ready, we use NNgen to turn it into register-transfer level (RTL) hardware. The DVFS controller and ML module are combined, and then we run the whole system with Verilator to test things out. We rely on GTKWave for waveform analysis to really see what's happening inside.

3.6 Power Analysis Methodology

For power analysis, we look at real switching activity. Simulations create VCD files showing signal transitions during operation. We convert those to SAIF format and analyze them in Yosys. This approach gives accurate numbers for dynamic power use, letting us compare a standard DVFS setup against our ML-assisted method.

SECTION IV — RESULTS AND ANALYSIS

4.1 Simulation Setup

We put the ML-based DVFS system through its paces using a variety of workload scenarios, simulating everything in Verilator. We also capture all the switching activity for later power checks. The tests pit the proposed approach against a conventional reactive DVFS system, both running under the same conditions.

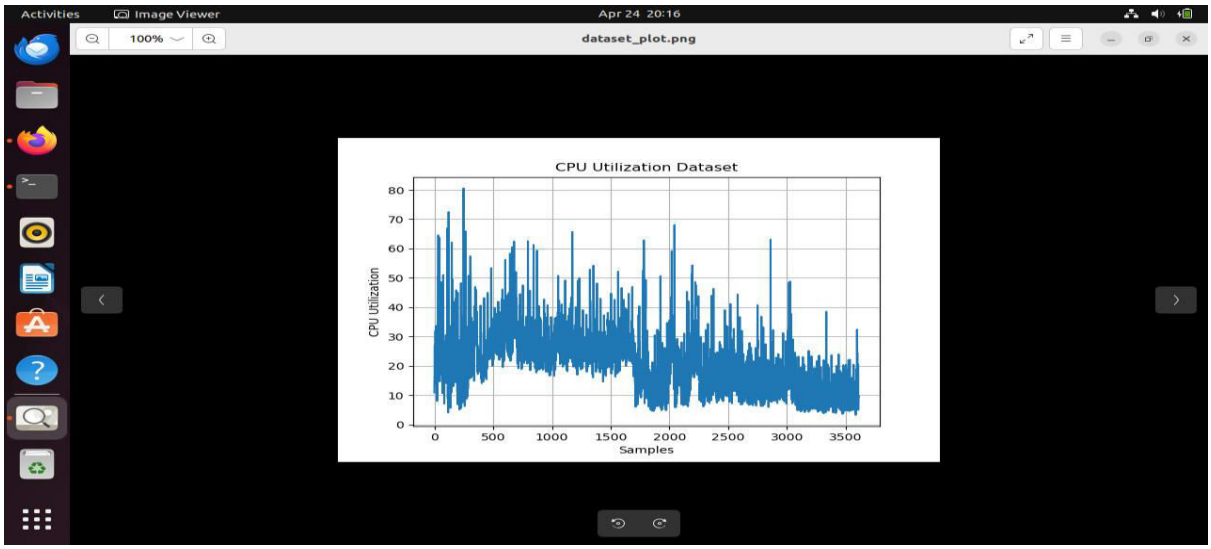

Fig. 11. Dataset representation used for training the ML model

4.2 Functional Verification

Waveform analysis tells the story: the ML model predicts workload changes from activity inputs, and the DVFS controller reacts to adjust the frequency. Overall, the ML-assisted system reacts faster than the conventional approach, cutting down on delays when scaling frequency.

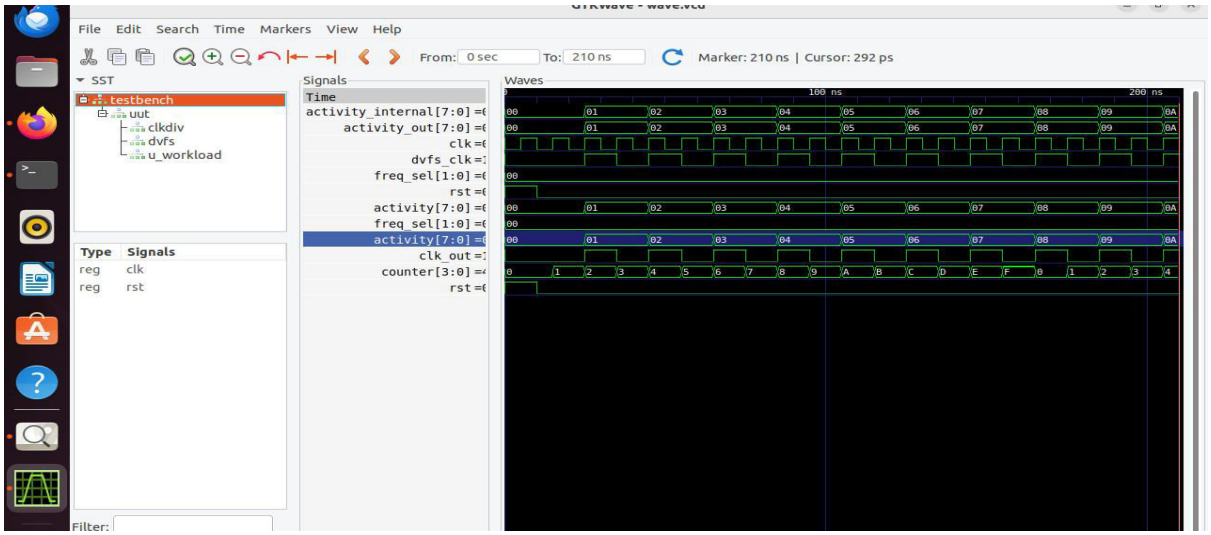

Fig. 12. GTKWave waveform showing workload prediction and frequency scaling

4.3 Power Analysis

With SAIF-based power estimation through Yosys, we see that the ML-assisted system uses less dynamic power than the standard DVFS setup. This drop in power comes from predicting workload surges early, so the system doesn't waste energy running at top speeds when it doesn't have to.

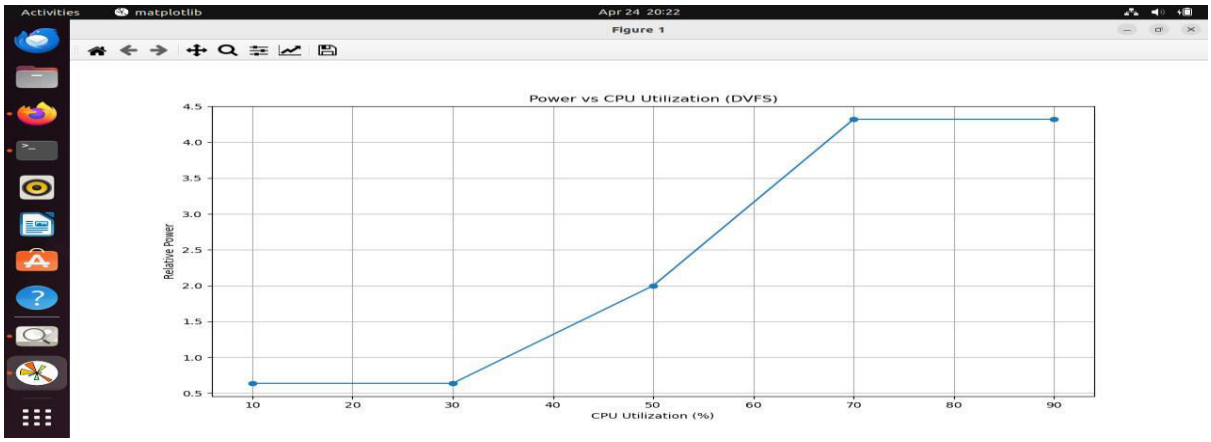

Fig. 13. Power consumption versus CPU utilization for DVFS operation

4.4 Performance Comparison

Parameter	Traditional DVFS	ML-Assisted DVFS
Control Type	Reactive	Predictive
Response Time	Delayed	Faster
Power Efficiency	Moderate	Improved
Frequency Scaling	After workload change	Before workload change
System Adaptability	Limited	High

4.5 Discussion

There's a clear win here — predictive control via ML pushes DVFS ahead of old-school reactive methods. Using a lightweight neural model keeps things lean, so the system doesn't get bogged down by heavy hardware. Still, the added complexity from including a neural network isn't nothing, especially if you're tight on resources.

SECTION V — CONCLUSION AND FUTURE WORK

5.1 Conclusion

To sum it up, this work introduces an Adaptive ML-assisted DVFS framework for making Mini-SoCs more power-efficient. By adding a simple neural network to the DVFS controller, the system predicts and adapts to workload changes ahead of time. Using NNgen, we brought the model into hardware and checked everything out with simulation and SAIF-based power estimates. Results prove it: we get lower power use and faster system response compared to traditional DVFS.

5.2 Future Work

Looking forward, there's room to sharpen the ML model even more, maybe by moving to something like reinforcement learning. Real-world validation on FPGA or ASIC platforms is another logical step. Plus, further fine-tuning could help cut down hardware overhead, making the approach scalable for bigger SoCs.

REFERENCES

[1] T. D. Burd and R. W. Brodersen, “Energy efficient CMOS microprocessor design,” Proc. IEEE Hawaii Int. Conf. System Sciences (HICSS), 1995, pp. 288–297.

[2] K. Govil, E. Chan, and H. Wasserman, “Comparing algorithms for dynamic speed-setting of a low-power CPU,” Proc. ACM MobiCom, 1995, pp. 13–25.

[3] J. Pouwelse, K. Langendoen, and H. Sips, “Dynamic voltage scaling on a low-power microprocessor,” Proc. ACM MobiCom, 2001, pp. 251–259.

[4] V. Pallipadi and A. Starikovskiy, “The ondemand governor,” Proc. Linux Symposium, 2006, pp. 215–230.

[5] H. Jung and M. Pedram, “Supervised learning based power management for multicore processors,” IEEE Trans. Computer-Aided Design Integr. Circuits Syst., vol. 29, no. 9, pp. 1395–1408, Sep. 2010.

[6] S. Herbert and D. Marculescu, “Analysis of dynamic voltage/frequency scaling in chip-multiprocessors,” Proc. Int. Symp. Low Power Electronics and Design (ISLPED), 2007, pp. 38–43.

[7] X. Wang and Y. Wang, “Machine learning based workload prediction for DVFS systems,” IEEE Trans. Very Large Scale Integration (VLSI) Syst., vol. 25, no. 1, pp. 123–135, Jan. 2017.

[8] NNgen: Neural Network Accelerator Generator. GitHub. https://github.com/NNgen/nngen

[9] C. Lattner and V. Adve, “LLVM: A compilation framework for lifelong program analysis and transformation,” Proc. Int. Symp. Code Generation and Optimization (CGO), 2004, pp. 75–86.

[10] C. Wolf, “Yosys Open SYnthesis Suite,” 2013. http://www.clifford.at/yosys/

[11] A. Carroll and G. Heiser, “An analysis of power consumption in a smartphone,” Proc. USENIX Annual Technical Conf., 2010.

[12] D. Meisner, B. T. Gold, and T. F. Wenisch, “PowerNap: Eliminating server idle power,” Proc. ACM ASPLOS, 2009, pp. 205–216.