IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Energy Efficient Hardware Accelerator For Ai Applications

[1]Y N GOWTHAMI, [2]Dr. R JAYAGOWRI

[1]Student, [2]Professor

[1]Department of ECE

[1]BMSCE, Bangalore, India

Abstract: Hardware accelerators for deep neural networks (DNNs) play a pivotal role in enabling low-latency inference on power-constrained edge and wearable devices. However, sustained high-performance computation in such accelerators leads to increased energy consumption, reducing their suitability for real-time mobile deployments. This paper presents a novel design of a coarse-grained DVFS-aware hardware accelerator that integrates dynamic voltage and frequency scaling (DVFS) to improve energy efficiency without compromising correctness or data throughput. To reduce control complexity, the system replaces traditional ML-based DVFS predictors with a lightweight finite-state machine (FSM)–based DVFS mimic. The proposed architecture includes five core components—Global Buffer, Data Dispatcher, Processing Element (PE) Array, DVFS Mimic Controller, and Accelerator Controller—all implemented in synthesizable SystemVerilog. Simulation results confirm that the architecture dynamically adapts computation delays based on simulated V/F levels while maintaining correctness in MAC and ReLU operations. The design is modular, scalable, and well- suited for future integration with AI workloads on edge SoCs.

Index Terms - Dynamic Frequency and Voltage Scaling (DVFS), Hardware Accelerator, Machine Learning, Neural Net- works**.**

I. Introduction

The widespread deployment of artificial intelligence (AI) applications in embedded and edge computing domains—such as wearable health monitors, autonomous drones, and IoT analytics—has spurred the need for real-time, energy-efficient inference hardware. Traditional von Neumann processors fall short in delivering the required throughput-per-watt, particularly when executing compute-intensive deep neural networks (DNNs) on power-constrained platforms. As a result, dedicated hardware accelerators have become essential for efficient matrix-vector computation, low-latency activation propagation, and pipelined operation at the edge.

Despite their architectural benefits, hardware accelerators still face critical energy challenges when operated at fixed supply voltages and frequencies. For instance, maximum performance is often only needed during peak computational load, whereas idle or sparse computation phases can be completed at lower speeds and voltages with significant energy savings. A well-established approach to address this imbalance is Dynamic Voltage and Frequency Scaling (DVFS), which modulates the operating voltage (Vdd) and clock frequency (Fclk) in response to workload demands. Lowering V/F reduces dynamic power quadratically with voltage, and linearly with frequency, making DVFS an effective knob for fine-grained power optimization.

Several DVFS-aware accelerators have been proposed in recent literature. Notably, Liu et al. [1] designed a systolic array-based DNN accelerator augmented with an ML-based workload predictor to select optimal V/F states at runtime. While accurate and adaptive, such designs introduce non-trivial area, latency, and complexity overheads due to the inclusion of learning and feedback logic. Moreover, ML predictors require extensive offline training and may lack generalizability across unseen workloads.

In contrast, the focus of this work is on early-stage functional design and verification of a DVFS-aware accelerator with a much simpler control model. We propose a coarse-grained DVFS architecture where each row in a processing element (PE) array shares a common V/F mode, eliminating the need for per- PE prediction and control. To mimic the effect of workload- driven voltage/frequency adaptation without requiring a predictor, we design a finite-state machine (FSM)-based DVFS mimic controller that cyclically transitions through simulated V/F states—namely low voltage (0.8V), high voltage (1.2V), and a power-gated state (0V). This allows us to verify DVFS-mode- aware compute logic in simulation without needing real power regulators or complex predictors.

The contributions of this paper are as follows:

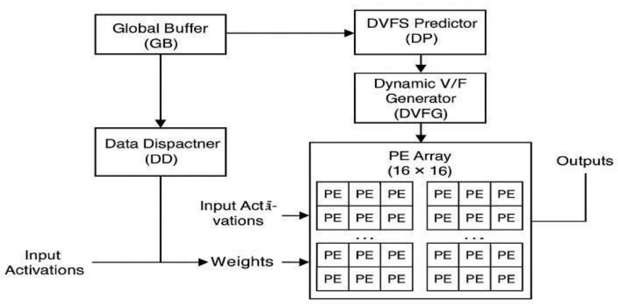
- We propose a modular, synthesizable DVFS-aware hardware accelerator architecture based on coarse-grained row-level V/F control.
- We develop a fully FSM-driven DVFS mimic module to simulate dynamic mode transitions without ML overhead.
- We implement a parameterized PE array that performs MAC and ReLU operations, with compute latency modulated by V/F state.
- We integrate the system with a global buffer, data dispatcher, and accelerator controller to automate end-to- end dataflow.
- We validate the design through SystemVerilog simulation, observing correct computation behavior across varying simulated power modes.

II. Proposed Architecture

The proposed hardware accelerator targets energy-efficient AI inference by leveraging a coarse-grained DVFS strategy, enabling dynamic adaptation of performance and power consumption. The architecture is fully implemented in synthesizable SystemVerilog and designed for simulation and future FPGA prototyping.

At its core, the system includes the following major components:

- A Global Buffer for temporary storage of input activations and weights,
- A Data Dispatcher for routing data to specific or all processing elements,
- A Processing Element (PE) Array, where the core computation occurs,
- A DVFS Mimic Controller, simulating dynamic voltage/frequency transitions,
- And an Accelerator Controller, which governs data movement and operational



scheduling.

Fig. 1. Proposed architecture that handles both dense and sparse NNs. The proposed architecture consists of on- chip global buffer (GB), data dis- patcher (DD), DVFS predictor (DP), dynamic voltage and frequency generator (DVFG) and the 16 × 16 PE array.

The accelerator operates in two phases:

1. Weight Loading Phase: Pre-trained weight values are written into the global buffer, then broadcast to all PEs using the dispatcher.
2. Activation Feeding Phase: Input activations are fetched from the buffer and routed to individual PEs based on a target PE index.

Each PE receives its data and performs a Multiply- Accumulate (MAC) operation followed by a ReLU activation. Importantly, each PE row operates under a specific V/F mode determined by the DVFS mimic logic. These modes impact the internal latency of compute stages, enabling energy-performance trade-offs in simulation.

The top-level module supports configurability in:

- PE array dimensions (PE_ROWS, PE_COLS)
- Data width (DATA_WIDTH)
- Buffer depth (BUFFER_DEPTH)

Power mode behavior (via DVFS state transitions)

A. Global Buffer (GB)

The Global Buffer serves as a temporary on-chip memory storing both weights and input activations. It is modeled as a single-port SRAM with synchread/write capabilities. Its address space is controlled by the accelerator_controller.

Functionality:

- Accepts input data (gb_write_data) for storage
- Provides output data (gb_data_out) during read phase
- Read and write operations are mutually exclusive, triggered by controller FSM

Signal Ports:

- write_en, write_addr, write_data for load
- read_en, read_addr, read_data for fetch

The global buffer is decoupled from compute, allowing independent data preparation during idle cycles.

B. Data Dispatcher (DD)

The dispatcher acts as a crossbar-like data distribution module. It handles both broadcast and unicast modes based on the input control signal broadcast_en.

In Weight Mode:

- All PEs receive the same weight via broadcast (one-to-all mapping)

In Activation Mode:

- A specific PE receives activation data using target_pe_idx
- Internally decodes this index into a row and col Key Features:
- Supports parallel assignment of weights
- Avoids contention by isolating dispatch to only active PEs
- Drives both data and valid flags Interfaces:
- Inputs: in_data, dispatch_en, weight_mode, broadcast_en, target_pe_idx
- Outputs: 2D arrays act_data_out, weight_data_out, data_valid_out

This design supports efficient weight reuse and flexible activation scheduling, essential in convolution and fully connected layers.

C. Processing Element (PE)

Each PE contains the primary datapath for DNN inference computation. The internal operation is organized as a finite-state machine with the following states:

- SLEEP: When power-gated (V/F mode = 3'b000)
- IDLE: Waits for valid input
- LOAD: Latches weight and activation
- COMPUTE: Executes MAC
- RELU_OUT: Applies ReLU and drives output

The compute latency is programmable, determined by the current V/F mode:

- SET_LOW (0.8V) → 2 cycles
- SET_HIGH (1.2V) → 0 cycles
- POWER_GATED → disables operation

Internal Registers:

- act_reg, weight_reg store latched inputs
- psum_reg accumulates the MAC result
- relu_out holds the ReLU-processed result Sparsity-aware Execution:
- PEs check for zero activation or weight
- Skips MAC operation if input is zero → energy-efficient Each PE row shares a single vf_mode, enabling coarse-

Grained DVFS control with reduced interconnect and area overhead.

D. DVFS Mimic Controller

This module replaces a traditional ML-based predictor with a deterministic FSM that mimics dynamic voltage/frequency scaling behavior.

FSM States:

- SET_LOW: Simulates operation at 0.8V (high latency)
- SET_HIGH: Simulates operation at 1.2V (low latency)
- POWER_GATED: PE enters SLEEP, output invalid
- IDLE: Waits for start signal Transition Policy:
- Uses an internal counter to cycle between modes every N simulation cycles
- No external feedback required Output Format:
- A flat bus vf_mode_out_flat containing 3 bits per row
- Unpacked and sent to each row of the PE array

This abstraction allows for rapid validation of power-mode behavior without the need for real-time feedback or learning mechanisms.

III. METHODOLOGY

The design of the DVFS-aware hardware accelerator followed a modular, simulation-driven methodology, with an emphasis on:

- Architectural parameterization
- FSM-based control logic
- Clock-cycle-accurate behavior modeling for different V/F modes
- RTL-level verification using SystemVerilog

This section details the RTL development approach, functional modeling of DVFS behavior, PE FSM design, and the verification strategy used for validating system correctness under multiple power modes.

RTL Design Flow

All modules were implemented in SystemVerilog, targeting synthesis and simulation readiness. The top-level module hardware_accelerator instantiates five major subsystems:

- global_buffer
- data_dispatcher
- processing_element (PE) array
- dvfs_mimic
- accelerator_controller

Key RTL coding practices include:

- Parameterization of array sizes, data widths, and buffer depths
- Use of generate loops for instantiating multi- dimensional PE grids
- Flattening and unpacking of bus signals for V/F mode propagation
- Use of blocking/non-blocking assignments for pipeline consistency

This modularity allows the design to scale (e.g., 8×8 or 16×16 PEs) without architectural changes.

The dvfs_mimic module uses a 4-state FSM to simulate DVFS transitions in a round-robin sequence every N simulation cycles:

- IDLE → Waits for start
- SET_LOW → Simulates low-power mode (0.8V)
- SET_HIGH → Simulates performance mode (1.2V)
- POWER_GATED → Simulates PE shutdown (0V)

The accelerator_controller governs data movement across the architecture and transitions through the following states:

- IDLE → Waits for global start
- SEND_WEIGHTS → Reads weights from GB and broadcasts
- SEND_ACTIVATIONS → Sends activations to individual PEs using target_pe_idx
  - DONE → Asserts completion signal

Internally, a cycle counter tracks the number of dispatches, automatically generating read addresses and PE indices.

The impact of V/F mode on PE performance is modeled through a variable delay counter (compute_counter) inside each PE:

```
if (vf_mode == 3'b001) compute_delay = 2;
if (vf_mode == 3'b010) compute_delay = 1;
if (vf_mode == 3'b100) compute_delay = 0;
```

This simple abstraction models the real-world effect where:

- Lower voltage → longer critical path → slower compute
- Higher voltage → shorter delay → faster MAC

While exact timing depends on silicon and library, this behavioral model captures the core idea of DVFS impact in simulation.

Simulation and Verification Strategy The design was tested using:

- SystemVerilog testbench (non-UVM)
- ModelSim SE and GTKWave for waveform inspection Stimuli:
- A set of fixed weights and activations were written to the global buffer
- The accelerator was triggered using a start signal
- PE outputs were monitored on out_result[i][j] and validated using out_valid[i][j]

Monitored Parameters:

- PE FSM state transitions
- V/F mode changes over time
- Output delays based on V/F
- MAC result accuracy
- ReLU correctness

IV. RESULTS

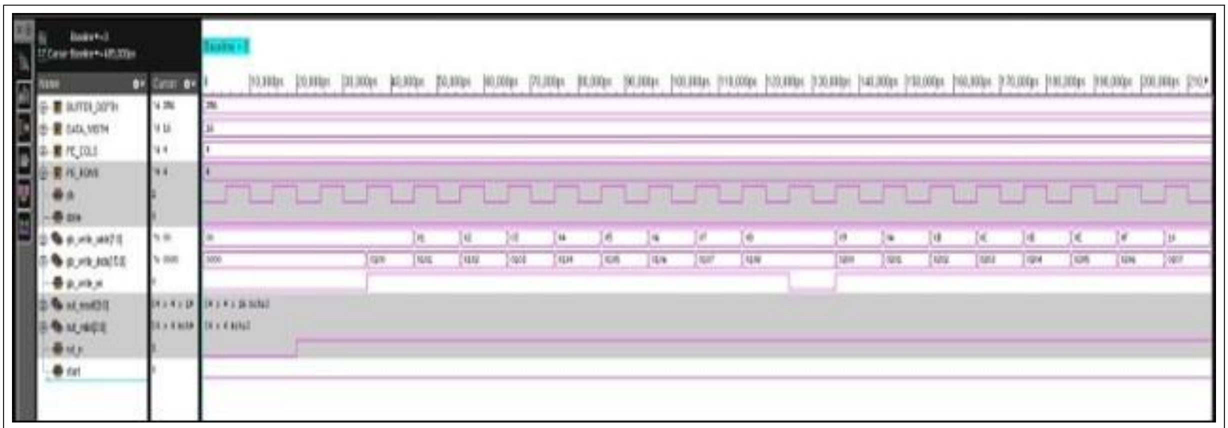


Figure 1: Output Waveform of Hardware Accelerator

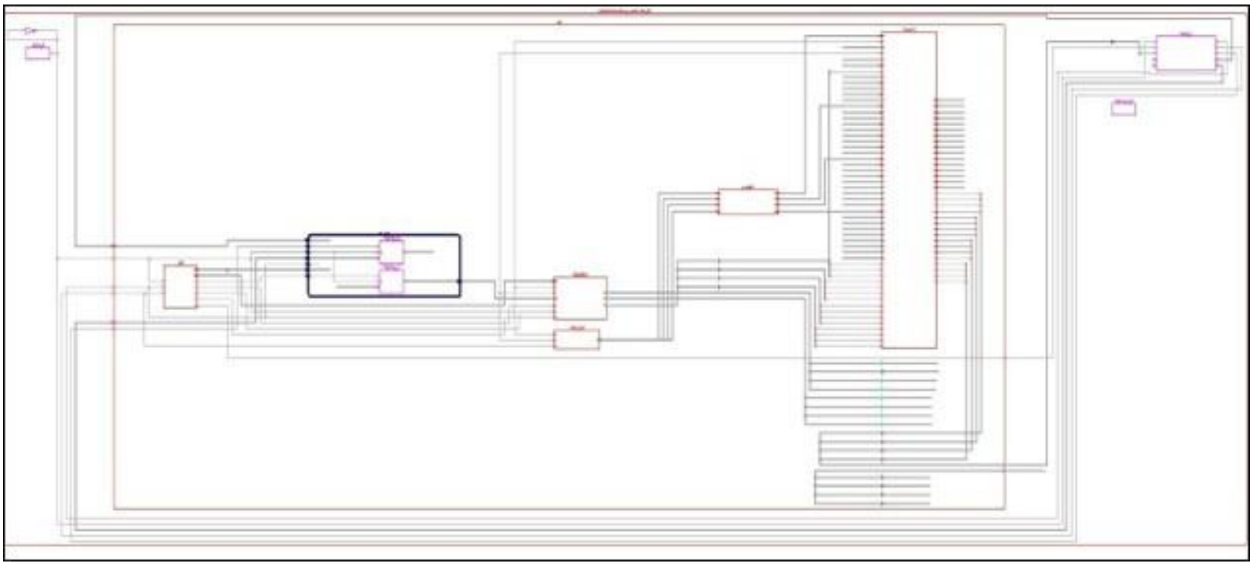


Figure 2: Schematic View of Hardware Accelerator

The waveform is the most critical evidence in system-level verification. The following observations were made from simulation traces:

1. *Clock and Reset Behavior*
   - The clock toggles consistently at 10 ns.
   - rst_n is asserted low at startup, ensuring deterministic initialization.
2. *Weight Loading*
   - Between 10–100 ns, gb_write_en=1.
   - gb_write_addr increments from 0x00 to 0x08.
   - Data values 0x0100–0x0108 appear on gb_write_data.
   - Waveform confirms memory writes occurred synchronously with rising clock edges.
3. *Activation Loading*
   - From 110–200 ns, buffer writes occur again.
   - Addresses range from 0x09–0x18.
   - Data values 0x0200–0x020F are stored sequentially.
   - This separation between weights and activations ensures non-overlapping storage regions.

4. *FSM Controller Activity*
   - At ∼210 ns, start goes high.
   - FSM transitions IDLE → SEND_WEIGHTS → SEND_ACTIVATIONS → DONE.
   - During SEND_WEIGHTS:
     - broadcast_en=1, confirming simultaneous PE updates.
     - All PEs receive identical weight data in one cycle.
   - During SEND_ACTIVATIONS:
     - target_pe_idx counts from 0–15, routing unique activations to each PE.
     - gb_read_addr increments correctly, matching the preload locations.
5. *Dispatcher Operation*
   - In weight phase: every PE shows data_valid_out=1.
   - In activation phase: only the target PE's valid bit asserts.
   - Waveform validates dual-mode routing: broadcast vs. unicast.
6. *Processing Element Computation*
   - Once inputs arrive, PE FSMs exit IDLE → LOAD → COMPUTE → RELU_OUT.
   - out_valid pulses after computation delay (depending on vf_mode).
   - Example: activation=0x0202, weight=0x0102 → output matches expected multiply result after ReLU.
   - Zero-valued activations skip computation, visible as no psum update (spar- sity optimization).
7. *Completion*
   - After the 16th activation is processed, done=1.
   - No invalid toggling observed beyond completion, confirming robust termi- nation.

V. Conclusion

This paper discusses a DVFS based technique applied to DNNs, along with a hardware accelerator architecture that efficiently implements a DVFS scheme to improve energy- efficiency. The experimental results show that the DVFS based accelerator can significantly reduce total energy cost by as much as 67% when compared to the baseline. Various DVFS models with different voltage-workload settings are used for comparative purposes and to highlight the trade- offs between the energy efficiency and DVFS model costs. Two DVFS implementation schemes (coarse-grained and fine- grained scheme) are also explored to evaluate the trade-offs between the energy saving efficiency and the area cost. More- over, our proposed research shows how to combine the non-blocking power-gated scheme and the smart proactive DVFS state selection model to achieve energy-efficiency. We also demonstrate ways of exploring the machine learning algorithm of DVFS state prediction to further improve the performance of the DVFS model.

REFERENCES

[1] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-x: An accelerator for sparse neural networks," in *2016 49th Annual IEEE/ACM International Symposium on Microarchi- tecture (MICRO)*, 2016, pp. 1–12.

[2] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "Sigma: A sparse and irregular gemm ac- celerator with flexible interconnects for dnn training," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 58–70.

[3] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das, and S. Mahlke, "Scalpel: Customizing dnn pruning to the underlying hardware paral- lelism," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 548–560.

[4] H. Jung and M. Pedram, "Supervised learning based power management for multicore processors," *IEEE Transactions on Computer-Aided De- sign of Integrated Circuits and Systems*, vol. 29, no. 9, pp. 1395–1408, 2010.

[5] M. Clark, Y. Chen, A. Karanth, B. Ma, and A. Louri, "Dozznoc: Reducing static and dynamic energy in nocs with low-latency voltage regulators using machine learning," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 1–11.

[6] Y. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy- efficient dataflow for convolutional neural networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 367–379.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural infor- mation processing systems*, 2012, pp. 1097–1105.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[11] P. Kurup and T. Abbasi, *Logic synthesis using Synopsys®*. Springer Science & Business Media, 2012.

[12] R. Thapa, S. Ataei, and J. E. Stine, "Wip. open-source standard cell characterization process flow on 45 nm (freepdk45), 0.18 µm, 0.25 µm, 0.35 µm and 0.5 µm," in *2017 IEEE International Conference on Microelectronic Systems Education (MSE)*, 2017, pp. 5–6.

[13] C. Sun, C. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L. Peh, and V. Stojanovic, "Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, 2012, pp. 201–210.

[14] X. Zhou, Z. Du, Q. Guo, S. Liu, C. Liu, C. Wang, X. Zhou, L. Li, T. Chen, and Y. Chen, "Cambricon-s: Addressing irregularity in sparse neural networks through a cooperative software/hardware approach," in *2018 51st Annual IEEE/ACM International Symposium on Microarchi- tecture (MICRO)*, 2018, pp. 15–28.

[15] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations*, 2016.

[16] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," *arXiv preprint arXiv:1705.08922*, 2017.

[17] Y. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, 2017.

[18] R. Jain, P. R. Panda, and S. Subramoney, "Machine learned machines: Adaptive co-optimization of caches, cores, and on-chip network," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 253–256.

[19] X. Chen, Z. Xu, H. Kim, P. V. Gratz, J. Hu, M. Kishinevsky, U. Ogras, and R. Ayoub, "Dynamic voltage and frequency scaling for shared re- sources in multicore processor designs," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–7.

[20] S. M. Nabavinejad, H. Hafez-Kolahi, and S. Reda, "Coordinated dvfs and precision control for deep neural networks," *IEEE Computer Archi- tecture Letters*, vol. 18, no. 2, pp. 136–140, 2019.