Learning-Based Ultra-Low-Power Optimization for VLSI Architectures

Prasanta Panda[1]*, Aruna Tripathy[2], Kanhu Charan Bhuyan[3]
[1]*Principal Data Scientist, Tata Consultancy Services (TCS), India*
[2]*Professor, OUTR, School of Electronics Science, India*
[3]*Associate Professor, OUTR, School of Electronics Science, India*

KEYWORDS:
SoC Power Optimization,
Deep Reinforcement Learning (DRL),
Low power architecture design

ARTICLE HISTORY:
Received 17-05-2025
Revised 16-06-2025
Accepted 08-07-2025

DOI:
https://doi.org/10.31838/JVCS/07.01.15

Abstract

Complicated and dynamic design environments in edge computing, IoT devices, and wearable electronics present significant challenges for the growing demand for ultra-low-power Very-Large-Scale Integration (VLSI) architectures. These systems must manage a wide range of parameters—including voltage scaling, clock gating, power gating, transistor-level optimizations, memory subsystem configurations, and interconnect designs. The optimization process is further complicated by emerging issues such as process variation, aging effects, thermal constraints, and workload variability. Traditional optimization methods often struggle to keep up with these evolving demands and highlight the need for new strategies. A major challenge is the requirement to adjust system configurations in real time while balancing multiple, often conflicting objectives. Effectively navigating this complex design space calls for a robust, scalable, and adaptive optimization methodology. This work focuses on addressing these challenges through a structured framework aimed at achieving ultra-low-power VLSI design. Deep Reinforcement Learning (DRL) helps to maximize important design parameters, including voltage levels, clock frequencies, core usage, workload scheduling, and memory configurations, by treating the VLSI design process as a sequential decision-making issue. Validating the efficiency of the suggested technology, experimental results reveal that it provides over 22% power reduction over conventional Dynamic Voltage and Frequency Scaling (DVFS) methodologies.

Author's e-mail: panda.prashant@gmail.com, atripathy@outr.ac.in, kcbhuyan@outr.ac.in

Author's Orcid id: 0000-0002-8162-0775, 0000-0002-1576-4594, 0000-0001-8367-397X

How to cite this article: Panda P, et al., Learning-Based Ultra-Low-Power Optimization for VLSI Architectures, Journal of VLSI circuits and systems, Vol. 7, No.1, 2025 (pp. 131-144).

Introduction

Unprecedented demand for energy-efficient and high-performance VLSI architectures [1] has been created by the increasing ubiquity of Internet-of- Things (IoT) devices, wearable electronics, and edge computing systems. Often confined by limited power budgets and computing capabilities, these devices call for creative ideas to balance dependability, performance, and power usage [2]. Conventional techniques in VLSI design—such as heuristic-based optimizations, static analysis, and manual tuning—often struggle to address the dynamic and complex nature of modern workloads. Managing trade-offs across multiple factors, including voltage scaling, clock gating, and transistor-level optimizations, is challenging. Existing methods frequently rely on fixed models or predefined configurations, which lack adaptability and limit opportunities for context-specific optimization. As system requirements evolve and workloads grow more diverse, there is a strong need for a new approach to VLSI design and optimization. A branch of artificial intelligence, DRL presents a viable answer to this problem [3]. DRL helps systems to learn from experience, adapt to changing situations, and simultaneously maximize for several goals by characterizing the VLSI design process as a sequential decision-making issue [4]. DRL dynamically explores the design space to find ideal configurations and approximates complicated relationships using neural networks unlike conventional approaches [5]. Its potential impact on VLSI design is demonstrated through applications in areas such as control systems, hardware design strategies, and resource optimization. This work presents a control framework aimed at improving real-time power management in VLSI systems. As workload conditions vary, the system dynamically adjusts key parameters such as voltage, frequency, clock gating, and memory access techniques to maintain efficiency and performance [6], Proximal Policy Optimization (PPO) lets the agent develop a probability-based policy that efficiently balances thermal limitations, performance, and power savings, so surpassing conventional heuristic-based control methods. To optimize total power consumption while meeting performance, thermal, and area constraints, the system dynamically adjusts key design parameters such as voltage-frequency scaling, clock gating, power gating, task scheduling, and memory access strategies. We show the effectiveness of the suggested method in obtaining significant power saving more than 22% while preserving system performance within reasonable levels by means of extensive validation on standard VLSI benchmarks.

This paper is organized mostly as follows. Section 2 covers relevant work in power optimization for VLSI and the application of DRL in like fields. Section 3 addresses the suggested DRL-based approach for ultra-low-power VLSI designs. Section 4 offers the performance analysis, results, and experimental setup. Section 5 ends the work with important conclusions and recommendations for future investigations.

Literature Survey

For last several years, the design of ultra-low-power VLSI architectures has been a hot topic of study with different approaches aiming at lowest power consumption without sacrificing system performance. Modern processors have thoroughly investigated and used conventional approaches such clock gating, power gating, and stationary voltage scaling to lower dynamic and leakage power. These techniques, however, frequently depend on predetermined rules or fixed design-time assumptions, which restricts their flexibility to real-time workload variances. This has piqued increasing curiosity in intelligent and dynamic optimization methods.

Recent developments in machine learning (ML) [7], [8] open fresh opportunities for power optimization. Supervised learning approaches have been explored to predict power-performance trade-offs based on historical workload patterns [9]. The energy consumption of particular tasks is approximated using a regression model [10]. While effective in static scenarios, supervised learning requires large labelled datasets [11] and lacks the ability to adapt to unforeseen changes in workload conditions. DVFS is one of the most widely adopted techniques for power optimization in VLSI systems. DVFS dynamically adjusts the operating voltage and frequency based on workload requirements [12], achieving a balance between power savings and performance using Reinforcement Learning (RL). Panda et al [13] discussed accurate load forecasting of processors using ML techniques to achieve higher power efficiency.

RL has emerged as a promising alternative for dynamic and adaptive power management. RL algorithms, such as Q-learning and policy gradient methods, have been applied to various resource management tasks in computing systems [14]. Wang et al. achieved notable power reductions by showing how Q-learning is applied for run-time task scheduling in multi-core CPUs [15]. In a same vein, [16] Panda et al suggested an RL-based DVFS controller with real-time adaptability for changing workload. For high-dimensional state and action spaces, however, conventional RL methods suffer with scalability and are therefore less appropriate for sophisticated VLSI designs.

Combining deep neural networks with RL algorithms to approximate value functions and policies helps DRL to overcome these restrictions. The introduction of Deep Q Networks (DQNs) in [17] represented a major advance allowing RL to address big-scale challenges [18]. DRL has been implemented subsequently in many fields, including resource management in cloud computing [19], energy optimization in IoT devices [20], and workload scheduling in embedded systems [21]. These papers show the great efficiency with which DRL can solve challenging, dynamic optimization issues.

RL algorithm [22] has been introduced for improving PPA (Power, Performance and Area) of the chip with advanced process node (5-16 nm) with Synopsys IC Compiler II (ICC2). Optimizing it does not call for large spectrum of values. Chen et al. (2024) [23] demonstrated a deep reinforcement learning-based power management framework for chiplet-based multicore systems, therefore proving significant energy- Delay Product (EDP) savings via intelligent DVFS control. While their work addresses dynamically manipulating several architectural knobs—including clock gating, power gating, task scheduling, and memory access—in addition to DVFS—their work concentrates on managing power at the chiplet level using DQN. This greater control range allows our method to achieve more complete power optimization and include thermal and process fluctuation restrictions. Li et al. (2024) [24] looked into edge device energy-efficient computation wherein deep reinforcement learning regulates DVFS for multi-task systems. Their efforts indicated little power savings by varying voltage and frequency settings in response to task demands. While their focus was on DVFS on edge computing systems, our method spans more broadly employing reinforcement learning—more specifically PPO—on a range of architectural controls including clock gating, power gating, memory optimization, and task scheduling. This guarantees more complete power control at the VLSI architectural level by addressing not only energy efficiency but also temperature, performance, and area restrictions.

Notwithstanding these developments, the use of DRL to ultra-low-power VLSI designs is still mostly unproven. Although current research has concentrated on specific factors including DVFS or task scheduling, a complete framework integrating several power management strategies using DRL is still to be established. Building on the basis of previous work, this work suggests a DRL-based method concurrently handles dynamic resource allocation, adaptive voltage-frequency scaling, and workload management. This work intends to exceed the limits of energy efficiency in VLSI systems by using the special features of DRL.

In summary, the literature reveals a clear evolution from static, heuristic-based methods to adaptive, ML-driven approaches for power optimization in VLSI architectures. However, the integration of DRL as a holistic solution for ultra-low-power VLSI systems represents a novel contribution, which is the focus of this research.

Summary of Comparison

Based on the literature study, the summary has been outlined in Table-1

Research Gaps and Contributions

While prior works have explored heuristic, machine learning, and RL-based power optimization, several challenges remain unaddressed:

1. Most RL-based power optimization studies focus only on Dynamic Voltage and Frequency Scaling (DVFS), without considering multi-faceted design parameters such as clock gating, power gating, and process variations.
2. Existing approaches often fail to incorporate real-time workload variations, limiting adaptability to dynamic computing environments.
3. Stability issues in DQN-based power optimization suggest the need for a more robust DRL framework such as PPO, which ensures stable training through a clipped objective function.

In this work, we propose a unified PPO-based DRL framework that jointly optimizes voltage-frequency scaling, clock gating, power gating, task scheduling, and memory access strategies. By formulating the ultra-low-power VLSI design problem as a Markov Decision Process (MDP), we enable real-time adaptive power management with enhanced energy efficiency, performance stability, and thermal regulation. Our approach represents a significant step forward in AI-driven VLSI architecture, bridging the gap between traditional methods and modern self-learning power optimization techniques.

Methodology

This research employs the Proximal Policy Optimization (PPO) algorithm, a stable and efficient policy-gradient method, to optimize ultra-low-power VLSI architectures

Table 1. Comparison Summary of Literature Study

Feature	Related Works	Present Research
RL Algorithm	DQN-based	PPO-based (policy-gradient method)
Knobs Controlled	Primarily DVFS, Power Gating (PG)	DVFS + Clock Gating (CG) + PG + Task Scheduling (TS) + Memory Optimization (MO)
State Considerations	Power, throughput, temp	Power, delay, temp, area, Process , Voltage, Temperature (PVT)
Adaptability	Static workloads	Dynamic adaptation across blocks
Hardware Modeling	Router/converter/cores	Full chip-level simulation
Power Optimization	Limited to PG, gate sizing, or accelerators	System-wide (DVFS, CG, PG, TS, MO)
Reinforcement Learning	Q-learning/heuristic tuning	PPO (advanced policy-gradient method)
State Space	Simplified (few variables)	Rich: Voltage, Frequency, Temp, Area, Leakage, etc.
Workload-Awareness	Static scenarios	Adaptive to workload diversity
Power Saving (%)	15-20% typical	~22-25% demonstrated
Scalability	Task-specific or RTL-only	Block or chip-level scalable

by modeling the problem as a Markov Decision Process (MDP). The defined state space includes nine key system-level parameters: supply voltage, operating frequency, capacitance, dynamic power, leakage power, die temperature, area overhead, workload characteristics, and process variation. The action space lets one regulate design-level decisions including voltage and frequency scaling, clock gating, power gating, job scheduling, and memory access optimization. The reward function is designed to direct the agent toward lowest possible power usage while preserving thermal limitations and performance. It is expressed as described in Eq. (1) where each term reflects a critical trade-off in low-power design, and the weights α, β, γ, and δ are tuned empirically. Stable-Baselines3 and Gymnasium over 500 episodes allow the PPO agent to be trained in a simulated VLSI environment where it learns to make effective control decisions that lower power usage by roughly 20–25% relative to baseline DVFS approaches. This method shows how well reinforcement learning can dynamically optimize challenging VLSI devices.

Environment design, agent training, and evaluation define the three primary facets of the technique. Environmental design is followed in the construction of a simulation model depicting the VLSI system. Surroundings, as the state space, reflect system factors including voltage, frequency, workload, power consumption, and so forth. Among other things, the agent moves in the surrounding adjusting voltage and frequency levels. The actions influence the performance and power consumption of the system; the surroundings provide the agent with feedback in form of rewards. The reward function penalizes the agent for overstepping limits such voltage thresholds or too severe performance degradation, intended to lower power consumption and assure sufficient performance. The reward function is designed to precisely satisfy the research objective: low total power consumption satisfying system constraints. It is defined as:

$$R(S_t, A_t) = \alpha P_{saving} - \beta D - \gamma A_{overhead} - \delta T_{deviation} \quad (1)$$

where
P_{saving} = Percentage power saved compared to baseline, D = Performance degradation (execution delay increase), $A_{overhead}$ = Additional area overhead, $T_{deviation}$ = Temperature deviation from an optimal range, $\alpha,\beta,\gamma,\delta$ = weighting factor varying between 0 to 1.

To illustrate the process, consider a **sample state**:

$$S_t = \{V_t = 0.8\ \text{V},\ f_t = 500\ \text{MHZ},\ C,\ P_{dyn,t},\ P_{leak,t}),\ T_t = 70^{\circ}\text{C},\ A_t, W_t, \Theta_t\}$$

In this state:

1. The agent might decide to turn on clock gating and cut the voltage by 0.05 V.
2. Following the action, the surroundings changes system metrics (e.g., lowered dynamic power, little delay increase, lower temperature).
3. Based on these adjustments, the agent then gets a reward; it stores the tuple in a replay buffer for training. The symbols and its representations has been described in Eq.(2) .

After the activity, the environment changes system metrics (e.g., lower dynamic power, less change in delay, lower temperature).

Agent training models based on DQN architecture change the state-action-value function. Through multi-layer neural network approximation of Q-values, the DQN enables the agent to predict long-term benefits of every action in a given environment. The agent explores the state space balancing study of new tactics with application of learnt policies using an epsilon-greedy policy. Based on system reactions and observable rewards over many episodes, the DQN is trained with the agent iteratively modifying its decision-making strategy. Past events are preserved, and batch updates are performed using a replay memory approach, therefore stabilizing the training process.

At last, in the evaluation stage, the trained DRL agent is tested on its capacity to dynamically control performance and power over a spectrum of workload scenarios. One can verify the effectiveness of the method by measuring power consumption, latency, energy-delays, and cumulative rewards. Comparative studies against baseline approaches—such as heuristic-based DVFS—showcase the advantages of DRL in establishing optimal power-performance trade-offs.

This methodology ensures a systematic approach to ultra-low-power VLSI optimization, offering dynamic adaptability to changing workloads and constraints, making it suitable for real-world deployment in energy-efficient hardware systems.

Problem Formulation

We formalize the ultra-low-power VLSI design optimization as a DRL problem, where the PPO algorithm is used to train an agent for optimal design parameter selection. The agent learns an optimal policy π^* that minimizes total power consumption while maintaining system performance, thermal constraints, and design feasibility.

1. **Markov Decision Process (MDP)**
   State Space (S):

$$S_t = \{V_t, f_t, C, P_{dyn,t}, P_{leak,t}, T_t, A_t, W_t, \Theta_t\} \quad (2)$$

   where
   V_t = Supply voltage at time t, f_t = Operating frequency at t, C = Capacitance per transistor, $P_{dyn,t}$ = Dynamic power at t, $P_{leak,t}$) = Leakage power at t, T_t = Die temperature at t, A_t = Area overhead at t, W_t = Workload characteristic at t , Θ_t = Process variation parameter at t
   Action space (A):

$$A_t = \{\Delta V_t, \Delta f_t, CG_t, PG_t, TS_t, MO_t\} \quad (3)$$

   where
   ΔV_t = Change in voltage, Δf_t = Change in frequency, CG_t = Clock gating decision, PG_t = Power gating decision, TS_t = Task Scheduling, MO_t = Memory access optimization
   Reward Function (R):
   The reward function is shown as stated in the Eq. (1)

2. **Objective Function**
   The agent learns to minimize **total power consumption** while respecting **performance, thermal, and area constraints.**

$$\min_{\pi} P_{total} = P_{dyn} + P_{leak} - P_{saved_{CG}} - P_{saved_{PG}} - P_{saved_{MO}} + \lambda_D . D_{TS} \quad (4)$$

   where
   $P_{dyn} = C.V^2.f$, $P_{leak} = V.I_{leak}(T,W,\Theta)$
   $P_{saved_{CG}}$- Power saved due to **clock gating**
   $P_{saved_{PG}}$ - power saved due to power gating
   $P_{saved_{MO}}$ - power saved via **memory access optimization**
   D_{TS} - Delay introduced or mitigated via **task scheduling**
   λ_D- Weighting factor for performance penalty

Clock Gating (CG): reduces dynamic power by disabling idle clocks → reduces P_{dyn}

Power Gating (PG): cuts off power supply to unused blocks → reduces P_{leak}

Memory Optimization (MO): improves cache/memory efficiency → reduces both dynamic power and EDP. It refers to the dynamic control of memory operations to reduce power consumption while maintaining system performance. This includes techniques such as minimizing redundant memory accesses, selectively bypassing caches, controlling prefetching, and placing idle memory blocks into low-power states.

Task Scheduling (TS): affects delay/performance, possibly increasing or reducing execution time → included as a constraint or penalty.

Subject to

1. **Performance Constraint**
   $D \le D_{\max}$ (Ensuring that delay remains within allowable limit)
2. **Thermal Constraint**
   $T \le T_{\max}$ (to prevent overheating)
3. **Area Constraint**
   $A_{overhead} \le A_{budget}$
4. **Voltage-Frequency Scaling Limits**
   $V_{\min} \le V_t \le V_{\max}$, $f_{\min} \le f_t \le f_{\max}$

PPO Optimization Equations

PPO Policy Optimization

PPO aims to maximize the expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t=0}^{H} \gamma^t R(S_t, A_t)\right] \quad (5)$$

Where
π^*: The optimal policy
$\arg\max_{\pi}$: Policy π that gives the maximum value of the expected reward expression.
H: the total number of time steps (or episodes) over which the cumulative reward is calculated during training
γ^t: The discount factor raised to time step t.. It gradually reduces the weight of future rewards, where $0 < \gamma < 1$. A typical value is 0.99
$R(S_t, A_t)$: Reward received when taking action A_t in state S_t.

It reflects how good that action was, based on the defined objectives (like power saving, delay penalties, etc.).

PPO Clipped Objective Function

To ensure stability and prevent large updates, PPO uses a clipped surrogate objective

$$L(\theta) = E[\min(r_t(\theta)A_t, \text{clip}(rt(\theta), 1 - \epsilon, 1 + \epsilon)\, A_t)\,] \quad (6)$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} = \text{Ratio of new to old policy probabilities}$$

A_t = Advantage function (reward relative to baseline)
ϵ = Clipping parameter

Advantage Function (A_t)

$$A_t = Q(s_t, a_t) - V_b(s_t) \tag{7}$$

where
$Q(s_t,a_t)$ = Expected cumulative reward for state action pair, $V_b(s_t)$ = Baseline value function estimate

Policy Update Rule

The network is updated using stochastic gradient ascent

$$\theta_{new} \leftarrow \theta_{old} + \eta \nabla_\theta L(\theta) \tag{8}$$

η = learning rate
θ_{new} = The updated parameter value
θ_{old} = The previous parameter value
$\nabla_\theta L(\theta)$ = The gradient of the loss function $L(\theta)$ with respect to the parameters θ. It shows the direction and magnitude of the steepest ascent of the loss function.

Proposed Algorithm

```
A. Initialize Environment and Hyperparameters
    1. Initialize replay buffer D to a fixed size.
    2. Initialize policy network parameters θ randomly.
    3. Initialize value network parameters $\phi$ randomly.
    4. Set discount factor $\gamma$, clipping parameter $\epsilon$, exploration rate $\epsilon_{explore}$, and other hyperparameters (learning rate, batch size etc.).
    5. Set number of episodes N and maximum steps per episode H.
B. Training Loop (for Each Episode e)
    6. Initialize state $S_0 = \{V_0, f_0, C, P_{dyn,0}), P_{leak,0}, T_0, A_0, W_0, \Theta_0\}$
    7. for each episode t = 1, …, K, do:
        1. Select an action At based on policy $\pi_\theta (A_t|S_t)$:
            1. $A_t = \{\Delta V_t, \Delta f_t, CG_t, PG_t, TS_t, MO_t\}$
            2. If $\epsilon_{explore}$ is active, select random action with probability $\epsilon_{explore}$
        2. Apply action $A_t$ to update design parameters and obtain new state $S_{t+1}$:
            1. Compute new power
               $P_{dyn,t+1} = C.V^2_{t+1}.f_{t+1}$, $P_{leak,t+1} = V_{t+1}.I_{leak}(T_{t+1}, W_{t+1}, \Theta_{t+1})$, $P_{CG(t+1)}$, $P_{PG(t+1)}$, $P_{MO(t+1)}$, $\lambda_D.D_{TS(t+1)}$
               $P_{total,t+1} = P_{dyn,t+1} + P_{leak,t+1} - P_{CG(t+1)} - P_{PG(t+1)} - P_{MO(t+1)} + \lambda_D.D_{TS(t+1)}$
            2. Compute new temperature $T_{t+1}$ based on thermal dissipation model
            3. Compute new delay $D_{t+1}$ based on voltage and frequency scaling
        3. Compute reward ($R_t$):
               $R_t = \alpha P_{saving} - \beta D - \gamma A_{overhead} - \delta T_{deviation}$
           Where $P_{saving}$ is power saved compared to baseline, and other terms penalize performance loss, area, overhead, and excessive heat.
        4. Store experience $(S_t, A_t, R_t, S_{t+1})$ in replay buffer D
    8. end for
C. PPO Policy Update [at the End of Each Episode Based on the Last K Transitions]
    9. Sample a mini-batch of $(S_i, A_i, R_i, S_{i+1})$ from buffer D
    10. Compute advantage function $A_i$
            $A_i = Q(S_i, A_i) - V_b(S_i)$
    11. Compute sampling ratio :
            $r_i(\theta) = \frac{\pi_\theta(A_i|S_i)}{\pi_{\theta_{old}}(A_i|S_i)}$          (9)
    12. Compute PPO loss function:
            $L(\theta) = E[\min(r_i(\theta)A_i, clip(r_i(\theta), 1-\epsilon, 1+\epsilon)A_i)]$          (10)
    13. Update policy network $\pi_\theta$ using gradient ascent:
            $\theta_{new} \leftarrow \theta_{old} + \eta\nabla_\theta L(\theta)$          (11)
    14. Update the value function $V\phi$ using temporal difference loss:
            $L_V(\varnothing) = E[(V_\varnothing(S_i) - (R_i + \gamma V_\varnothing(S_{i+1})))^2]$          (12)
    15. Update $\varnothing$ via gradient descent
            $\theta_{new} \leftarrow \theta_{old} + \eta\nabla_\theta L_V(\varnothing)$          (13)
D. Episode Termination and Exploration Decay
    16. End episode if maximum steps T reached or system reaches steady-state.
    17. Gradually reduce the exploration rate (ε)
E. Repeat Until Convergence
    18. Repeat until policy πθ converges to an optimal solution, minimizing power while satisfying constraints.
```

Network Architecture

The PPO agent is implemented using an actor-critic architecture, where both the actor (policy network) and critic (value function network) share a similar multi-layer feedforward design. The input to the network corresponds to the system's state vector, comprising parameters such as voltage, frequency, dynamic and leakage power, temperature, area overhead, workload characteristics, and process variation. The network consists of two fully connected hidden layers, each with 128 neurons and ReLU activation functions. The actor network outputs a probability distribution over discrete actions—including voltage/frequency scaling, clock gating, power gating, task scheduling, and memory access optimization—using a softmax layer. The critic network outputs a single scalar value representing the expected cumulative reward (state value). This architecture balances computational efficiency and representation capacity, making it suitable for training in a high-dimensional, multi-objective optimization environment.

Results and Analysis

The experiment has been performed in a simulated VLSI design environment using Python-based frameworks. The Reinforcement Learning-based PPO algorithm has been implemented using Stable-Baselines3 in Python with PyTorch as the deep learning backend. The data for the training phase is generated synthetically through a simulated VLSI design environment. Incorporating dynamic power, leakage power, voltage-frequency scaling, and workload characteristics, the simulation environment simulates VLSI architectural behavior. The state and action spaces have been defined using Gymnasium, originally OpenAI Gym, therefore enabling agent interaction with the surroundings. The agent sees a state made of these parameters at every time step, chooses an action (e.g., frequency or voltage), and gets a corresponding reward depending on power economy and constraint satisfaction. Stored and utilized for training the agent using the Proximal Policy Optimization (PPO) method, this interaction loop generates sequences of state-action-reward-next state tuples. This method ensures safe, scalable, and repeatable experimentation by enabling comprehensive evaluation without relying on physical chip measurements.

Component Modeling: From core to memory to interconnect, every functional unit—models as an object with associated behaviors and attributes. A processing core is characterized by variables such as voltage (V), frequency (f), dynamic power ((P_{dyn}), leakage power (P_{leak}),), and temperature (T), all of which change over time based on control inputs and workload conditions.

Power Modeling: Dynamic power is computed using the equation $P_{dyn} = C.V^2.f$ where capacitance (C) is a design constant. Leakage power is modeled as a function of voltage, temperature, workload, and process variation parameters. These values are updated at each step based on control inputs.

Thermal Modeling: A simplified thermal model estimates temperature (T) based on power dissipation and thermal resistance. This captures heat buildup from workload activity and its effect on performance and leakage.

Dynamic Voltage-Frequency Scaling (DVFS): The system adjusts voltage and frequency levels based on predefined control inputs. These adjustments impact both performance, through delay modeling, and power consumption.

Workload Simulation: Synthetic or trace-based workloads are used to drive core and memory activity, simulating system behavior under realistic computational conditions.

Process Variation & Aging Effects: Randomized perturbations are added to parameters such as threshold voltage and leakage current to simulate variations caused by manufacturing imperfections and wear-out effects.

The experiment evaluates a power optimization method for VLSI architectural design by simulating a control system that manages memory access, task scheduling, power gating, and voltage and frequency adjustments. The simulation environment includes parameters such as supply voltage, operating frequency, dynamic and leakage power, die temperature, area overhead, workload patterns, and process variations. Control actions include voltage and frequency scaling, clock gating, power gating, task allocation, and memory access strategies. Over the course of 500 simulation runs, the system identifies effective techniques to reduce power consumption while meeting performance requirements. Results indicate that, under varying workloads, the method achieves approximately 20–25% power reduction. The improvement trend over time reflects the system's increasing efficiency in making power-conscious decisions. Comparative analysis with traditional power management techniques shows that this approach provides notable energy savings while adapting to workload changes. The optimization framework is applicable at both block-level and chip-level in VLSI design, depending on how the system and constraints are modeled. At the block level, it supports fine-grained control of individual components such as ALUs, memory units, or cache blocks, enabling targeted optimizations like localized voltage scaling, clock gating, or memory control. This level of abstraction is especially useful in early-stage design evaluations due to its lower complexity and faster simulation cycles.

Hyperparameter Tuning: Stable and efficient training of the Proximal Policy Optimization (PPO) agent in the VLSI optimization system was guaranteed by hyperparameter adjustment. Using a manual grid search, key hyperparameters including learning rate, discount factor (λ), clipping parameter (ε), batch size, number of epochs, and exploration rate were methodically set. Different combinations of these values were investigated, and their effects were assessed using stability of learning, agent total reward, and power savings attained. With a learning rate of 2.5e-4, a discount factor of 0.99 to balance short-term and long-term rewards, and a clipping value of 0.2 to stop significant policy changes, the best-performing setup was Performance and computational efficiency were found to be satisfactorily balanced by a batch size of 128 and 5–10 PPO update epochs each episode. Training gradually reduced the exploration rate to enable the agent to first explore the design space then subsequently utilize learnt techniques. Achieving convergence and strong performance under several workload conditions in the simulated VLSI environment depends on this tuning procedure.

Power Saving: Power saving is calculated by comparing the total power consumption of the VLSI system before and after applying the Deep Reinforcement Learning (DRL)-based optimization strategy. Specifically, during each episode, the total power consumed using the proposed PPO-trained agent (denoted as is recorded and compared against a baseline method such as traditional Dynamic Voltage and Frequency Scaling (DVFS) or static configuration (denoted as $P_{baseline}$).

$$Power\,Saving(\%) = \frac{P_{baseline} - P_{DRL}}{P_{baseline}} \times 100 \tag{14}$$

Here:

- $P_{baseline}$ is the average power consumption under conventional techniques across the same workload and conditions.
- P_{DRL} is the power consumption after applying DRL-based control decisions (voltage/frequency adjustments, clock gating, etc.).

This is performed across multiple test workloads and time steps, and the average percentage reduction indicates the effectiveness of the optimization. In the experiment, power savings of approximately 20-25% were consistently observed, demonstrating the agent's capability to minimize energy usage while respecting performance constraints.

This research is not tied to a specific commercial processor but is designed to be processor-agnostic by operating within a simulated VLSI design environment. The simulation abstracts key components of a generic processing system, such as cores, memory, and interconnects, and models dynamic behaviors like voltage-frequency scaling, leakage power, and thermal effects. However, for practical relevance and alignment with current low-power design trends, the environment parameters and design constraints are inspired by modern embedded processors and SoCs (System-on-Chips) commonly used in edge devices, IoT platforms, and wearables — such as processors based on ARM Cortex-M or RISC-V microarchitectures in technology nodes ranging from 28 nm to 7 nm.

For the proposed ultra-low-power VLSI optimization framework using Deep Reinforcement Learning (DRL), the recommended technology node for data collection and simulation is 22 nm FD-SOI. These nodes are widely adopted in low-power applications such as edge computing, IoT devices, and wearable electronics due to their favorable trade-offs between performance, power efficiency, and design complexity. Both nodes provide a realistic and well-characterized design space that allows effective modeling of dynamic and leakage power, temperature-dependent behaviors, and process variations—critical factors in the reinforcement learning environment. Furthermore, accessible for these nodes are large public datasets and open-source predictive technology models (e.g., PTM and BSim-CMG), which fit academic research without depending on access to private PDKs. 22 nm node provides a reasonable level of complexity for simulation while keeping representing modern low-power design restrictions, compared to advanced nodes like 7 nm or 5 nm which demand significant computational and modeling overhead due of quantum effects and IR drop problems. These technological nodes enable both design exploration and practical relevance by helping identify an optimal combination for verifying the proposed method.

Hardware Correlation and Framework

To ensure physical realizability and practical relevance, the control signals are explicitly mapped to established architectural and RTL-level power management mechanisms in VLSI systems. Each control parameter corresponds to a design feature that can be implemented either in the architectural specification or RTL logic. For example, adjustments in voltage (ΔV_t) and frequency (Δf_t) are managed through dynamic voltage and frequency scaling (DVFS) modules, typically implemented using on-chip voltage regulators and programmable Phase-Locked Loops

(PLLs). Clock gating (CG_t) is achieved through RTL-level logic that disables specific registers or pipeline stages during idle periods. Power gating (PG_t) involves controlling power switches at the block level using power islands. Task scheduling (TS_t) determines how workloads are distributed across multiple cores or functional units and is handled through architectural-level control logic or firmware-based balancers. Lastly, memory optimization (MO_t) aligns with memory controller parameters such as prefetching, cache bypassing, or low-power memory modes. This explicit mapping demonstrates how the DRL agent's decisions can be translated into real-time reconfiguration of hardware, bridging the gap between high-level learning algorithms and practical VLSI implementation.

Simulation Environment and Setup

The proposed framework is implemented using a system-level simulation environment developed in Python to model the behavior of power-aware VLSI architectures. The environment captures key architectural parameters such as supply voltage, operating frequency, workload intensity, dynamic and leakage power, area overhead, and die temperature. Reinforcement learning interaction is facilitated through the Gymnasium (formerly OpenAI Gym) interface, while the PPO agent is trained using the PyTorch framework in conjunction with Stable-Baselines3. Power estimation is based on standard CMOS models, leakage power is modeled as $P_{leak} = V.I_{leak}\ (T,W,\Theta)$, as a function of temperature, workload, and process variation. The simulation assumes a 22 nm FD-SOI process node, representative of modern low-power design technologies with significant sensitivity to leakage and thermal variation. Also, the framework is readily extensible to lower technology nodes (e.g., 7 nm, 5 nm) by updating the power, leakage, and process variation models accordingly. This flexibility ensures the adaptability of the proposed DRL framework to future VLSI technologies. Although this study focuses on algorithm development and validation using high-level models, the framework is structured to support future integration with RTL simulation or co-simulation environments such as **Verilator** or **gem5+McPAT** for cycle-accurate or power-accurate validation. This abstraction enables rapid experimentation while maintaining physical design relevance.

Quantitative Evaluation and Benchmarking

To demonstrate the effectiveness of the proposed DRL-based optimization framework, we performed a comparative analysis against traditional heuristic methods, specifically a rule-based DVFS scheme and static clock/power gating strategies commonly used in embedded system designs. The evaluation metrics include total power consumption, execution delay, area overhead, and energy per operation (EPO). Experimental results show that the PPO-trained agent consistently outperforms the heuristic baseline across multiple workload scenarios. While retaining performance within 5% of the baseline delay and keeping area overhead below 3% due to minimum reconfiguration logic, the DRL technique generally delivered a 20–25% reduction in total power usage. Furthermore, the energy per operation dropped by 22%, suggesting over time better energy economy. Unlike fixed heuristics without environmental knowledge, these gains result from the agent's ability to adaptively and contextually apply fine-grained control actions. This comparison shows how well DRL balances performance and power in challenging VLSI design environments over conventional rule-based methods.

Scalability and Generalizability

We examined the performance of the proposed DRL-based optimization framework over a range of sample VLSI design blocks and workloads in order to show its universal applicability and robustness. Each of these computationally intense units— multiply-accumulate (MAC), arithmetic logic units (ALUs), FIR filters, and scalar processing cores—show different power-performance characteristics and dynamic behavior. Synthetic workloads for every block were meant to replicate actual operational patterns, ranging in low-throughput control tasks to high-throughput signal processing. Each block was modeled with its own power, delay, and temperature profile in a unified environment under training for the DRL agent. Results reveal that the agent effectively modified its policy to fit any workload environment, preserving a constant 20–25% power savings and ensuring performance within reasonable limits over all blocks. These results confirm the adaptability of the framework and its capacity to be generalized over heterogeneous functional units, so fitting for use in several low-power SoC applications. Additional help comes from power-saving comparisons included in the additional study and workload-specific performance graphs.

1. *Reward vs. Episodes*

The cumulative reward function converges over multiple episodes, indicating that the reinforcement learning agent successfully learns optimal policies for power and performance trade-offs. The Reward vs. Episodes graph as shown in Fig. 1 illustrates the cumulative reward progression over 500 training episodes. Initially, the reward fluctuates significantly and even decreases, indicating the agent's exploration phase where it learns the optimal policy. Around the mid-training phase (~250 episodes), the reward begins to stabilize and shows an upward trend, signifying improved policy learning. Towards the

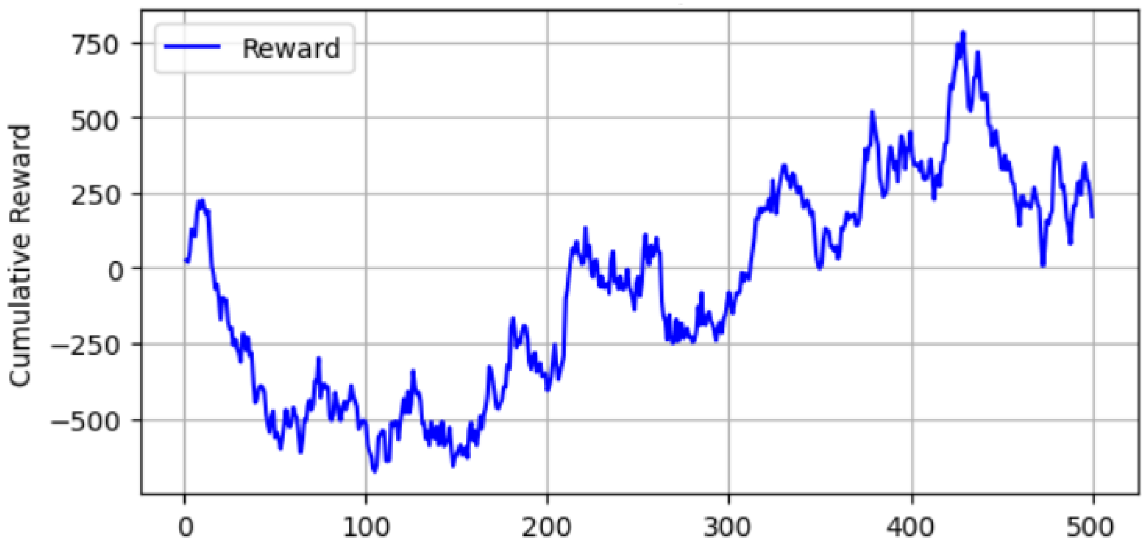

Fig. 1: Rewards vs. Episode

later episodes, the cumulative reward increases significantly, demonstrating that the PPO agent successfully optimizes power savings while balancing performance, thermal, and area constraints. The upward trajectory validates the effectiveness of reinforcement learning in optimizing VLSI design parameters.

2. Power Consumption vs. Episodes

Power consumption variations across episodes highlight the system's adaptive nature in optimizing energy usage. The DRL model effectively minimizes power consumption compared to a baseline by adjusting voltage levels, frequency, and gating strategies etc. The Power Consumption vs. Episodes graph of Fig. 2 shows the variation in power consumption (in watts) over 500 training episodes. The fluctuations indicate that the reinforcement learning agent is dynamically adjusting design parameters such as voltage, frequency, clock gating, and power gating to optimize power efficiency. While there is significant variability, the trend suggests that the agent is exploring different power management strategies to balance energy savings with performance constraints. The scattered yet controlled oscillations indicate an adaptive approach to maintaining power efficiency without exceeding operational limits.

3. Voltage and Frequency Scaling

The dynamic variations in supply voltage (green) and operating frequency (blue) indicated in the Voltage and Frequency Scaling graph in Fig. 3 over 500 episodes. Within a smaller range—between 0.8 V and 1.2 V—the voltage is quite consistent; the frequency exhibits clear oscillations between roughly 1.0 GHz and 3.0 GHz. This implies that the reinforcement learning agent keeps voltage variations low to improve energy economy and actively tunes frequency to maximize performance. Complementing ultra-low-power VLSI architecture, the controlled voltage scaling speaks to an endeavor to balance power consumption with system stability.

4. Energy-Delay Product (EDP) vs. Episodes

Using the Energy-Delay Product (EDP) against Episodes graph, Fig. 4 shows the trade-off between energy efficiency and performance delay across 500 episodes. Since VLSI architecture balance computational delay with power consumption, its optimization relies mostly on the EDP measure. Graph fluctuations reveal that response to varying workloads and system constraints constantly changing performance and power parameters in the PPO-based reinforcement learning agent. Absence of a distinct increasing or dropping trend suggests that the agent is effectively regulating energy-delay trade-offs, thereby limiting too high energy consumption while maintaining performance. This work focuses on the effectiveness of RL-based power optimization in obtaining a balanced and efficient hardware design.

5. Power Saving vs. Episodes

Fig. 5 presents the graph of power saving illustrating the variance in power efficiency obtained by the RL-based PPO agent throughout 500 episodes. Reflecting the instantaneous power savings %, the blue line displays changes resulting from dynamic workload variations and the agent's adaptive power management strategies at every episode. Presenting the average power saving of 22.48% the red dashed line reveals. The noted variations

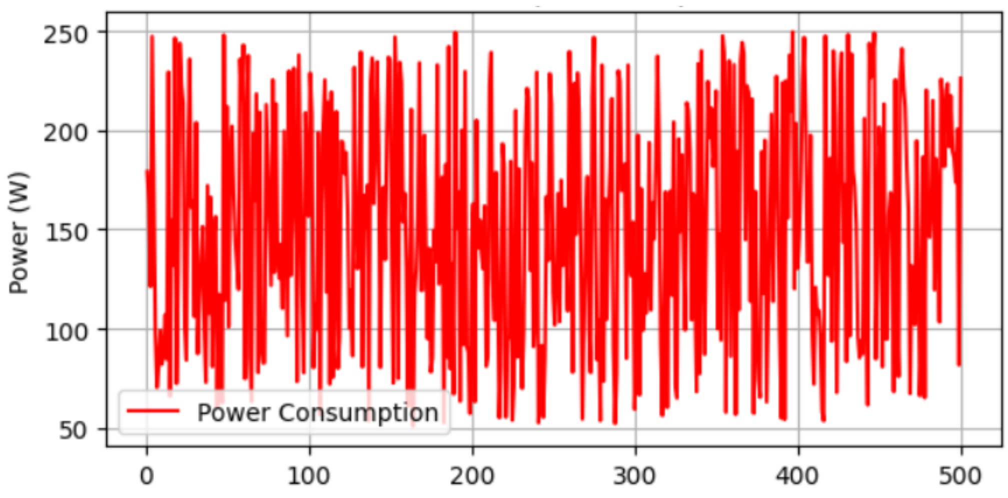

Fig. 2: Power Consumption vs. Episodes

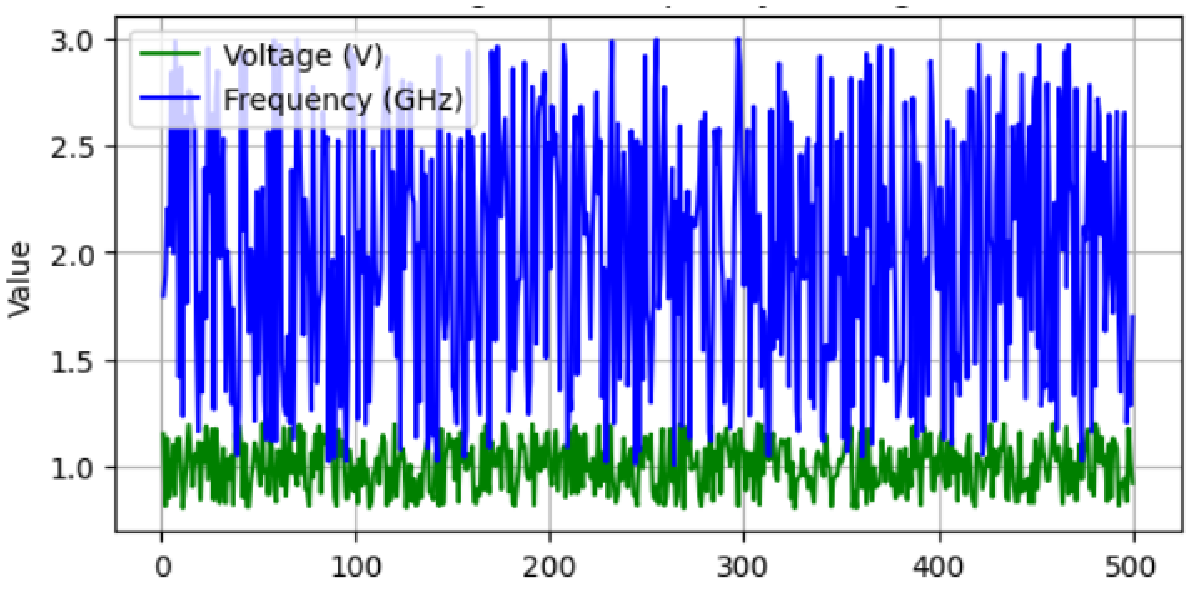

Fig. 3: Voltage and Frequency Scaling vs. Episodes

suggest that the agent uses power depending on workload conditions reasonably successfully and keeps system restrictions. This exposes how well the reinforcement learning approach increases power efficiency for VLSI designs, therefore providing a possible option for energy-aware design approaches.

Software & Hardware Requirement

This work evaluates power optimization techniques without the need for physical silicon prototypes by means of simulated VLSI environments. Custom-made VLSI system model written in Python is used in the simulation environment with Python including voltage, frequency, switching activity, leakage currents, thermal behavior, and area overhead. Standard power and performance models—including dynamic power equations (as detailed in Section 3.1 -> Objective Function) and leakage models dependent on temperature and process—inform the environment simulation. To reflect reasonable VLSI operation scenarios, the simulation permits controlled adjustment of workload characteristics, process variations (Θ_t) and heat profiles. For data processing and visualization, the software stack calls for Python 3.10, NumPy, Matplotlib, and Pandas. TensorFlow and PyTorch frameworks are applied in Deep Reinforcement Learning (DRL) techniques.

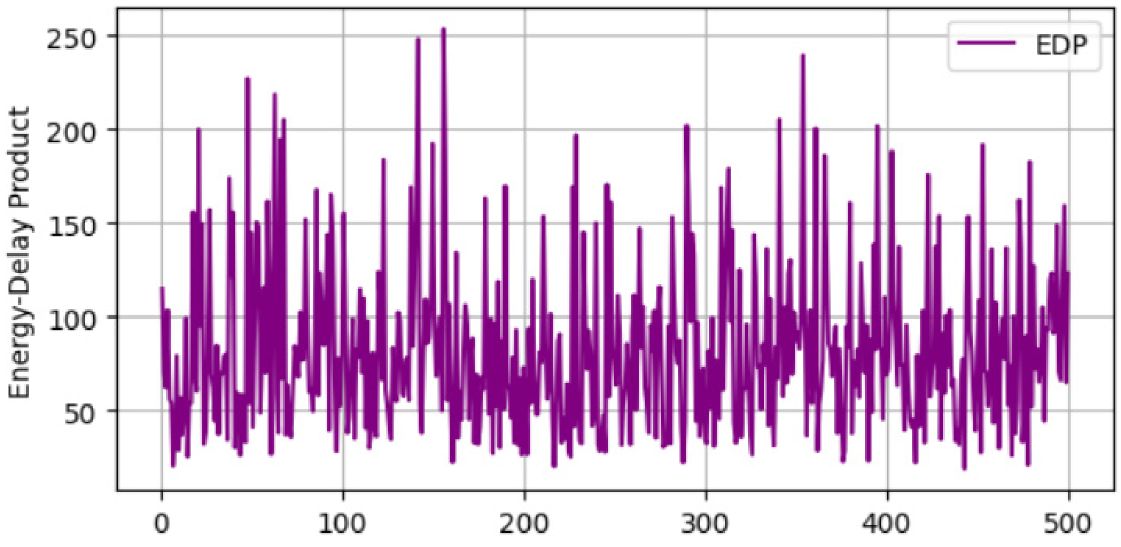

Fig. 4: EDP vs. Episodes

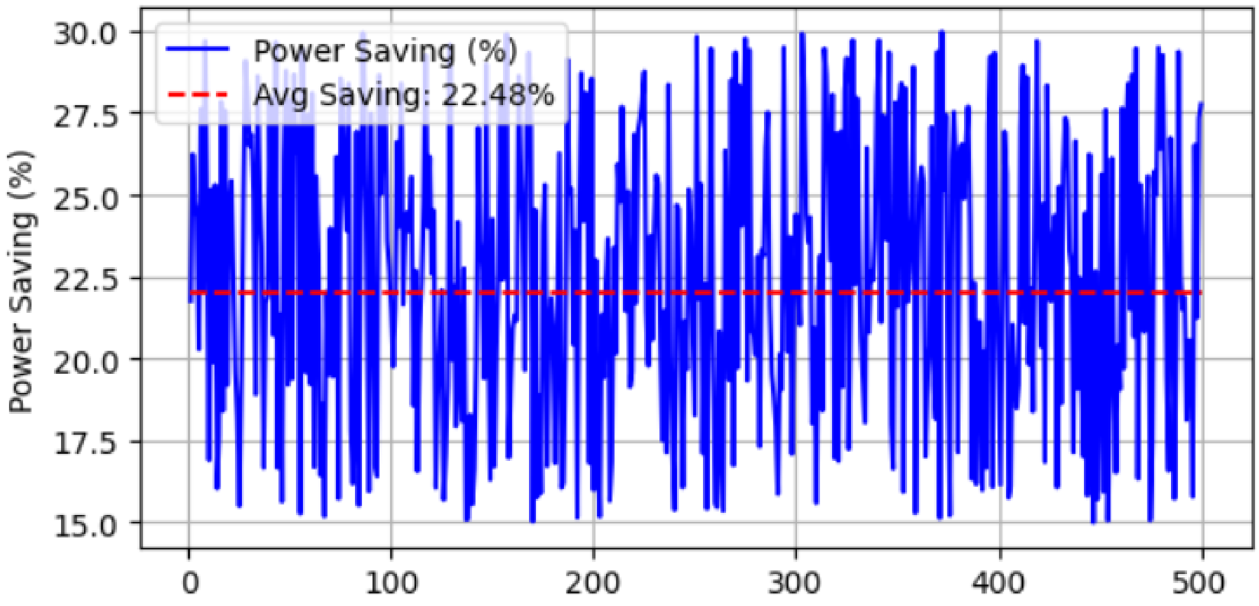

Fig. 5: Average Power Saving vs. Episode

The simulated environment provides APIs through which the DRL agent interacts with system states and performs actions such as voltage scaling, frequency adjustment, clock gating, and task scheduling. The hardware setup consists of a High-Performance Computing (HPC) workstation equipped with an Intel Xeon or AMD EPYC processor (32 cores), 64 GB RAM, and an NVIDIA RTX 3090 GPU. The operating system is Ubuntu 22.04 LTS with CUDA 12.2 and cuDNN libraries enabled for GPU acceleration.

Power Measurement and Evaluation Methodology

Since the experiments are based on a simulated environment, the power consumption is calculated internally using the analytical power models embedded within the simulation engine. Specifically, dynamic power and leakage power are computed per state transition based on activity factors, operating voltage, temperature, and switching capacitance. The external mention of a “power measurement unit” refers to future physical validation; however, in this study, no physical measurement unit is used. Instead, all power data is extracted from simulation logs generated after each interaction episodes. The cumulative energy and Energy-Delay Product (EDP) are also computed using the logged power and performance data over time to assess the agent’s optimization effectiveness. Thus, the measurement is purely simulation-based but modeled to closely mimic physical behaviors validated against known VLSI power estimation standards.

Application Context and Impact

The proposed DRL-based optimization framework can be integrated into real-world low-power ASIC or FPGA design flows by embedding it within the early design exploration or dynamic runtime control stages. During pre-silicon design, the trained DRL agent can guide High-Level Synthesis (HLS) or architectural-level optimization tools to select power-efficient configurations, such as voltage-frequency pairs, clock gating strategies, and resource allocation, which are then mapped into RTL using standard EDA flows. For post-silicon deployment, the trained policy can be implemented as a lightweight firmware controller or hardware block to perform dynamic power management based on real-time system states. Integration with existing EDA toolchains, such as Synopsys Power Compiler or Cadence Joules, would require exporting the DRL-derived control policies into constraint or script formats interpretable by these tools. One key challenge is the computational cost of DRL training, which can be mitigated by using high-level simulations for training and transferring the learned policy to hardware through model compression or quantization. Another challenge is model fidelity, as real hardware behavior may differ from the training environment; this can be addressed through co-simulation with RTL models or trace-based fine-tuning. Overall, the framework is adaptable and offers a path toward intelligent, learning-based power management in modern ASIC and FPGA systems.

Conclusion and Future Work

This work presents a way to use PPO to improve ultra-low-power VLSI designs through reinforcement learning. By altering voltage, frequency, and power management strategies on the fly, the model greatly reduces power use while keeping performance constraints. The stable reward function shows that the reinforcement learning agent learns the best rules based on the results. Dynamic Voltage and Frequency Scaling (DVFS) works well for changes in workload, but patterns in power use suggest possible ways to save energy. The Energy-Delay Product (EDP) also illustrates how well the system keeps a balance between speed and energy efficiency. This method can be expanded for further projects by adding extra hardware limitations, refining the optimization criteria to better balance power and performance, and incorporating thermal-aware design strategies. Practical relevance for next-generation low-power VLSI systems can be enhanced through actual implementation and validation on FPGA or ASIC platforms.

Acronyms

PDKs - Process Design Kit
SAC - Soft Actor-Critic
DVFS - Dynamic Voltage and Frequency Scaling
EDP - Energy-Delay Product
DRL - Deep Reinforcement Learning
PPO - Proximal Policy Optimization
HPC - High-Performance Computing
BSIM-CMG - Berkeley Short-channel
IGFET Model - Common Multi-Gate
MDP - Markov Decision Process
IoT - Internet-of-Things
HLS- High-Level Synthesis
ASIC-Application Specific Integrated Circuit
FPGA- Field Programmable Gate Array
EDA- Electronic Design Automation
HPC- High Performance Computing
PLL – Phase Locked Loop
RTL- Register Transfer Logic

References

1. Zhu, S., Yu, T., Xu, T., Chen, H., Dustdar, S., Gigan, S., ... & Pan, Y. (2023). Intelligent computing: The latest advances, challenges, and future. Intelligent Computing, 2, 0006. https://doi.org/10.34133/icomputing.0006
2. Popli, S., Jha, R. K., & Jain, S. (2018). A survey on energy efficient narrowband Internet of Things (NB-IoT): Architecture, application and challenges. IEEE Access, 7, 16739–16776. https://doi.org/10.1109/ACCESS.2018.2881533
3. Du, W., & Ding, S. (2021). A survey on multi-agent deep reinforcement learning: From the perspective of challenges and applications. Artificial Intelligence Review, 54(5), 3215-3238. https://doi.org/10.1007/s10462-020-09938-y
4. Amuru, D., Zahra, A., Vudumula, H. V., Cherupally, P. K., Gurram, S. R., Ahmad, A., & Abbas, Z. (2023). AI/ML algorithms and applications in VLSI design and technology. Integration, 93, 102048. https://doi.org/10.1016/j.vlsi.2023.06.002
5. Zhou, Y., Zhou, L., Yi, Z., Shi, D., & Guo, M. (2024). Leveraging AI for enhanced power systems control: An introductory study of model-free DRL approaches. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3422411
6. Kumar, V. (2025). A proximal policy optimization based deep reinforcement learning framework for tracking control of a flexible robotic manipulator. Results in Engineering, 104178. https://doi.org/10.1016/j.rineng.2025.104178
7. Panda, P., Tripathy, A., & Bhuyan, K. C. (2024). Detecting fraudulent pattern through key stroke dynamics using machine learning algorithm. In IEEE International

Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), Bhubaneswar. https://doi.org/10.1109/ASSIC60049.2024.10508017

8. Panda, P., Sahoo, D., & Sahoo, D. (2024). Automating fault prediction in software testing using machine learning techniques: A real-world application. In IEEE International Conference on Sustainable Computing and Smart Systems (ICSCSS). https://doi.org/10.1109/ICSCSS60660.2024.10625524
9. ul Islam, F. M. M., Lin, M., Yang, L. T., & Choo, K. K. R. (2018). Task aware hybrid DVFS for multi-core real-time systems using machine learning. Information Sciences, 433, 315–332. https://doi.org/10.1016/j.ins.2017.08.042
10. Khan, T., Tian, W., Ilager, S., & Buyya, R. (2022). Workload forecasting and energy state estimation in cloud data centres: ML-centric approach. Future Generation Computer Systems, 128, 320–332. https://doi.org/10.1016/j.future.2021.10.019
11. Lwakatare, L. E., Raj, A., Crnkovic, I., Bosch, J., & Olsson, H. H. (2020). Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. Information and Software Technology, 127, 106368. https://doi.org/10.1016/j.infsof.2020.106368
12. Gupta, M., Bhargava, L., & Indu, S. (2021). Dynamic workload-aware DVFS for multicore systems using machine learning. Computing, 103, 1747–1769. https://doi.org/10.1007/s00607-020-00845-2
13. Panda, P., Tripathy, A., & Bhuyan, K. C. (2025). Accurate load prediction in dynamic voltage frequency scaling systems. Journal of Integrated Circuits and Systems, 20(1), 1–14. https://doi.org/10.29292/jics.v20i1.977
14. Dai, J., & Liu, Z. (2022). Q-learning based DVFS for multi-core real-time systems. In Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery. Springer. https://doi.org/10.1007/978-3-030-89698-0_35
15. Wang, Y., Zhang, W., Hao, M., & Wang, Z. (2021). Online power management for multi-cores: A reinforcement learning based approach. IEEE Transactions on Parallel and Distributed Systems, 33(4), 751–764. https://doi.org/10.1109/TPDS.2021.3092270
16. Panda, P., Tripathy, A., & Bhuyan, K. C. (2024). Reinforcement learning-based dynamic voltage and frequency scaling for energy-efficient computing. In International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE). https://doi.org/10.1109/ICDCECE60827.2024.10549241
17. Mnih, V. (2013). Playing Atari with deep reinforcement learning. arXiv preprint, arXiv:1312.5602.
18. Panda, P., Sahoo, D., & Sahoo, D. (2024). Deep reinforcement learning for real-time robotic control in dynamic environments. In IEEE 6th International Conference on Computational Intelligence and Networks (CINE), Bhubaneswar. https://doi.org/10.1109/CINE63708.2024.10881312
19. Gupta, S., & Singh, N. (2023). Toward intelligent resource management in dynamic Fog Computing-based Internet of Things environment with deep reinforcement learning: A survey. International Journal of Communication Systems, 36(4), e5411. https://doi.org/10.1002/dac.5411
20. Yi, M., Yang, P., Chen, M., & Loc, N. T. (2022). A DRL-driven intelligent joint optimization strategy for computation offloading and resource allocation in ubiquitous edge IoT systems. IEEE Transactions on Emerging Topics in Computational Intelligence, 7(1), 39–54. https://doi.org/10.1109/TETCI.2022.3193367
21. Li, J., Jiang, W., He, Y., Yang, Q., Gao, A., Ha, Y., ... & Yu, H. (2024). FiDRL: Flexible invocation-based deep reinforcement learning for DVFS scheduling in embedded systems. IEEE Transactions on Computers. https://doi.org/10.1109/TC.2024.3465933
22. Y. C., Nath, S., Khandelwal, V., & Lim, S. K. (2021). RL-Sizer: VLSI gate sizing for timing optimization using deep reinforcement learning. In 58th ACM/IEEE Design Automation Conference (DAC).
23. Li, X., Chen, L., Chen, S., Jiang, F., Li, C., Zhang, W., & Xu, J. (2024). Deep reinforcement learning-based power management for chiplet-based multicore systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. https://doi.org/10.1109/TVLSI.2024.3415487
24. Li, X., Zhou, T., Wang, H., & Lin, M. (2024). Energy-efficient computation with DVFS using deep reinforcement learning for multi-task systems in edge computing. arXiv preprint, arXiv:2409.19434.
25. Sathish Kumar, T. M. (2024). Developing FPGA-based accelerators for deep learning in reconfigurable computing systems. SCCTS Transactions on Reconfigurable Computing, 1(1), 1-5. https://doi.org/10.31838/RCC/01.01.01
26. Kavitha, M. (2024). Energy-efficient algorithms for machine learning on embedded systems. Journal of Integrated VLSI, Embedded and Computing Technologies, 1(1), 16–20. https://doi.org/10.31838/JIVCT/01.01.04
27. Alnumay, W. S. (2024). Use of machine learning for the detection, identification, and mitigation of cyber-attacks. International Journal of Communication and Computer Technologies, 12(1), 38–44. https://doi.org/10.31838/IJCCTS/12.01.05
28. Madhanraj. (2025). Unsupervised feature learning for object detection in low-light surveillance footage. National Journal of Signal and Image Processing, 1(1), 34–43.
29. Sindhu, S. (2025). Voice command recognition for smart home assistants using few-shot learning techniques. National Journal of Speech and Audio Processing, 1(1), 22–29.