Hybrid EA-DRL Optimisation Framework for Fast-Settling Multi-Stage PLL with Adaptive Locking Control

1st Aowen Yang
School of Information and Communication Engineering
Communication University of China
Beijing, China
awyang@mails.cuc.edu.cn

2nd Qin Zhang*
State Key Laboratory of Media Audio & Video
Communication University of China
Ministry of Education, China
zhangqin@cuc.edu.cn

***Abstract*—This paper presents a hybrid evolutionary algorithms and deep reinforcement learning (EA–DRL) optimisation framework for fast-settling multi-stage all-digital phase-locked loops (ADPLL). To address the coupled trade-offs among lock time, phase noise, overshoot, and robustness, a structure-aware NSGA-III algorithm (SA-NSGA-III) is developed to generate high-quality Pareto optimal static parameters that preserve the physical consistency of two-stage ADPLL dynamics. Based on this optimised baseline, a DRL controller based on twin delayed deep deterministic policy gradient (TD3) is used to fine-tuning the loop gain in real time to enhance transient convergence and steady-state accuracy. Experimental results demonstrate that the proposed framework achieves significant improvements in settling speed, jitter performance, and robustness to process, supply voltage, and temperature (PVT) variation compared to conventional static designs, reducing lock time by up to 98% during frequency transitions and substantially improving dynamic response performance.**

***Index Terms*—EA, DRL, ADPLL**

I. Introduction

High-performance phase-locked loops (PLLs) are essential in modern communication systems, clock generation, and radar platforms [1], which enable frequency synthesis and phase synchronisation by converting stable reference frequencies to tunable outputs with similar stability [2]. In dynamic environments such as 5G networks [3] and agile radars [4], the performance of PLL affects acquisition speed, frequency-tracking accuracy, and resistance to jamming and multipath. As systems advance to higher millimetre wave frequencies [5], stricter demands require optimising bandwidth, phase noise, power efficiency, and settle time.

Traditional PLL designs that employ static parameter configurations face an inherent trade-off among acquisition speed, locking accuracy, and environmental robustness [6]. To overcome these trade-offs, multistage or gear-shifted PLL topologies have been proposed [7], [8]. However, parameter selection in these architectures often relies on complex analytical derivations or empirical heuristics, which can be cumbersome and may not yield optimal results. Furthermore, real-world operating conditions introduce persistent disturbances, including fluctuations in process, supply voltage, and temperature (PVT) variation, and external noise, all of which continuously affect loop dynamics [9]. Static optimisation methods that lack adaptive mechanisms are inadequate for addressing these variations, leading to significant performance degradation and an inability to meet the reliability requirements of mission-critical applications.

Researchers have increasingly adopted various methods to improve the performance of PLL [10]. The evolution algorithm (EA) is effective in optimising non-convex PLL parameters in multimodal, high-dimensional spaces, as demonstrated in applications such as jitter-power trade-offs and non-linear dynamics tuning using genetic algorithms and particle swarm optimisation [11], [12]. However, EA provide only static solutions, which are insufficient for addressing runtime PVT drifts. Deep reinforcement learning (DRL) addresses this limitation by obtaining real-time policies through interaction with the environment [13]. During the past decade, DRL has improved response in power electronics [14] and adaptive phase locking for beam combining [15]. Despite these advances, standard DRL methods still face some challenges in learning PLL parameters, including slow convergence and sensitivity to initialisation.

This paper proposes a hybrid optimisation framework to address the parameter optimisation and adaptive control problems of multi-stage PLL under high dynamic conditions. The main contributions are as follows.

- A hybrid evolutionary algorithm and deep reinforcement learning (EA-DRL) framework addresses the high-dimensional multi-objective trade-off problem in high-performance PLL by integrating static evolutionary algorithm optimisation with dynamic DRL control, thereby alleviating the slow convergence and sensitivity to initial conditions in traditional DRL in practical applications.
- An improved NSGA-III algorithm with structure-awareness for PLL dynamics adjusts the exploration and convergence weights of each loop stage via adaptive crossover mutation, reference point selection, and embedded mode switching to achieve a better set of Pareto optimal parameters.

- The hybrid EA-DRL framework enables real-time PLL parameter tuning through a DRL agent for multi-stage control, outperforming static benchmarks under frequency hopping, PVT variations, and thereby enhancing system robustness and reliability.

The remainder of this paper is organised as follows. Section II introduces the multi-stage PLL architecture and constructs a 6 objective static optimisation model. Section III proposes the adaptive SA-NSGA-III algorithm and derives the optimal parameter set as the baseline. Section IV describes the hybrid EA-DRL architecture in detail. Section V presents the experimental results. Section VI gives the conclusion.

II. Multi-Stage PLL Optimization Model

A. PLL Basic Concepts

PLL is a typical negative feedback synchronisation system whose basic function is to keep the oscillator output phase consistent with the reference signal [2]. A conventional PLL comprises a phase detector (PD), a loop filter (LF), a voltage-controlled oscillator (VCO) and a feedback divider, as illustrated in Fig. 1.

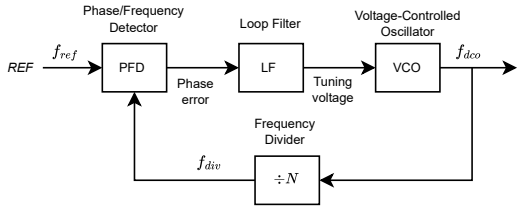

Fig. 1. Traditional PLL Block Diagram Consisting of PFD, LF, VCO, and Frequency Divider [2].

PLL operates as follows. PD measures the phase difference $e_\phi(t)$.

$$e_\phi(t) = \phi_{\text{ref}}(t) - \phi_{\text{div}}(t) \tag{1}$$

$$u(t) = K_p e_\phi(t) + K_i! \int e_\phi(\tau), \mathrm{d}\tau \tag{2}$$

The loop filter applies proportional-integral control (PI) to the error signal, generating a control signal $u(t)$ for the voltage-controlled oscillator. The phase output $f_{\text{out}}(t)$ of the oscillator is divided and returned to the PD, creating a closed loop.

$$f_{\text{out}}(t) = f_{\text{free}} + K_{\text{vco}} u(t) \tag{3}$$

The following conditions must be met for the system to remain locked.

$$f_{\text{out}} = \frac{R}{N} f_{\text{ref}}, \qquad \phi_{\text{ref}}(t) = \phi_{\text{div}}(t) \tag{4}$$

The performance of a PLL is commonly evaluated using three primary metrics [6]. The first metric, lock time t_{lock}, is the time it takes the system to regain stability at the target frequency after a frequency jump or an initial offset.

$$t_{lock} = min\left\{t : |f_{out}(t) - f_{target}| < \Delta f, |e_\phi(t)| < \Delta\phi\right\} \tag{5}$$

The second metric is phase noise, or timing jitter t_j, which is determined by the relationship between the root-mean-square (RMS) of the phase error and the carrier frequency.

$$t_j = \frac{\phi_{\text{RMS}}}{2\pi f_{\text{out}}} \tag{6}$$

The third set of metrics, Δf_{cap} and Δf_{lock}, describes the frequency acquisition range and the robustness of the locked state of PLL, respectively. Specifically, Δf_{cap} indicates the maximum initial frequency offset Δf_0 for which the PLL can achieve lock, while Δf_{lock} represents the maximum reference frequency step δf that the locked PLL can withstand without losing lock.

$$\Delta f_{\text{cap}} = \max\left\{\Delta f_0 : \text{the PLL can acquire lock}\right\} \tag{7}$$

$$\Delta f_{\text{lock}} = \max\left\{\delta f : \text{the PLL maintains lock}\right\} \tag{8}$$

Traditional PLL faces two primary limitations [16]. The first is the bandwidth-noise trade-off: increasing loop bandwidth accelerates lock-in and reduces settling time, but also amplifies in-loop noise transmission to the output. This degradation in phase noise performance complicates achieving both rapid acquisition and low-noise steady-state operation. The second limitation arises from hardware non-linearities, particularly the significant variation in VCO tuning gain across control voltages.

B. ADPLL Model and Optimization Objective

To address the identified limitations, a two-stage all-digital PLL (ADPLL) architecture is implemented to achieve performance decoupling. The structure diagram is as follows Fig. 2. This architecture employs two simultaneous control paths with distinct bandwidths. Stage-A provides rapid acquisition through higher gain and broader bandwidth, while Stage B uses lower bandwidth and a smaller digital oscillator gain K_{dco} to achieve steady-state, low-noise tracking.

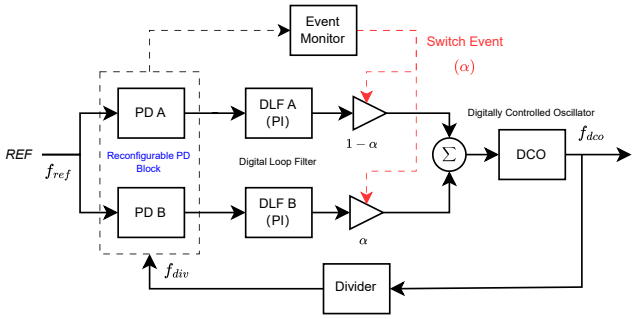

Fig. 2. Two-Stage ADPLL Architecture with Reconfigurable PD, Dual Loop Filters, and Event-Driven Switching.

The phase errors $e^{(i)}[n]$ of the two stages and the corresponding digital PI controller output $u^{(i)}[n]$ are given by

$$e^{(i)}[n] = \phi_{\text{ref}}[n] - \phi_{\text{div}}[n], \qquad i \in \{A, B\}, \tag{9}$$

and

$$u^{(i)}[n] = K_{p,i}\, e^{(i)}[n] + K_{i,i}\, T_u \sum_{k=n-W+1}^{n} e^{(i)}[k], \qquad (10)$$

where T_u denotes the control update period.

A natural transition between phases during the locking process is achieved on the basis of error statistics. The transition is triggered when either the phase error RMS within the sliding window of length W satisfies below the predefined threshold ε_ϕ:

$$\sqrt{\frac{1}{W}\sum_{k=n-W+1}^{n} e_\phi^2[k]} < \varepsilon_\phi, \qquad (11)$$

or the output frequency remains within a tolerance band around the target frequency for a continuous period:

$$\left| f_{\text{out}}[n] - f_{\text{target}} \right| < \rho_f f_{\text{target}}, \qquad (12)$$

where ρ_f denotes the normalised frequency-tracking tolerance.

Once either criterion is met, the loop enters a dwell period of T_{dwell} samples to ensure the stability of the current stage before initiating the transition. After the dwell period, a smooth blend from Stage-A to Stage-B is performed. To prevent abrupt variations in the control word, a half-cosine window [17] is used to generate a time-varying blending factor $\alpha[n]$, defined as

$$\alpha[n] = \frac{1}{2}\left(1 - \cos\left(\pi \frac{n - n_s}{N_{\text{sw}}}\right)\right), \qquad \alpha[n] \in [0,1], \quad (13)$$

where n_s is the starting index of the switching window and N_{sw} denotes the total number of transition samples.

The effective loop control signal is then expressed as a weighted combination of the Stage-A and Stage-B control outputs:

$$u_{\text{eff}}[n] = (1 - \alpha[n])\, u_A[n] + \alpha[n]\, u_B[n]. \qquad (14)$$

Finally, the output frequency of the digitally controlled oscillator (DCO) is updated using the effective control word:

$$f_{\text{out}}[n] = f_{\text{free}} + K_{\text{dco,eff}}[n]\; u_{\text{eff}}[n]. \qquad (15)$$

$K_{\text{dco,eff}}[n]$ is obtained by dynamic interpolation of the two-stage parameters using the same hybrid weighting scheme. With this architecture, all tunable parameters can be uniformly expressed in vector form

$$\mathbf{P} = \left\{ \begin{array}{c} \text{PD}_A,\ K_{pA},\ K_{iA},\ K_{\text{dco}A} \\ \text{PD}_B,\ K_{pB},\ K_{iB},\ K_{\text{dco}B} \\ \text{Switch}\colon \varepsilon_\phi,\ \rho_f,\ T_{\text{sw}},\ \gamma_{\text{preset}} \end{array} \right\}.$$

The design incorporates several adjustable and tunable parameters, including the PD type and gain, control gain, digitally controlled oscillator gain, and switching mechanism. These parameters collectively influence acquisition speed, steady-state noise, robustness, and switching smoothness, thereby defining the decision space for subsequent multi-objective optimisation. To comprehensively characterise the performance of the multi-stage ADPLL, six static optimisation objectives were established as Table I.

TABLE I
STATIC PERFORMANCE OBJECTIVES OF THE MULTI-STAGE ADPLL

Objective Index	Definition		
	Expression	***Unit***	***Note***
F_1	t_{lock}	s	Lock time
F_2	$(f_{\max} - f_{\text{ß}})/f_{\text{ß}}$	–	Overshoot ratio
F_3	$\mathbb{E}[\lvert\phi_{\text{err}}\rvert]$	rad	Steady-state phase error
F_4	$\phi_{RMS}/(2\pi f_{\text{target}})$	s	Phase jitter
F_5^*	$\Delta f_{\text{cap}}/f_{BW,B}$	–	Normalized capture range
F_6^*	$\Delta f_{\text{lock}}/f_{BW,B}$	–	Normalized lock range

Two-stage ADPLL provides greater flexibility in balancing performance trade-offs than single-stage PLL. However, optimising all six performance metrics simultaneously still requires careful consideration of inherent trade-offs. A wider capture range generally requires a higher proportional gain K_{pA} in Stage-A together with a relaxed loop bandwidth, while extending the lock range favours a stronger integral gain K_{iA} and a lower noise floor. Similarly, reducing the Stage-B loop bandwidth effectively suppresses locked-state phase noise but compromises both capture and lock robustness. Conventional single-objective tuning approaches are insufficient for navigating this high-dimensional design space.

III. STRUCTURE-AWARE MULTI-OBJECTIVE OPTIMIZATION

In order to identify the optimal initial parameters in the six-objective space,this study develops a SA-NSGA-III algorithm based on the standard NSGA-III [18]. The proposed algorithm integrates the strengths of NSGA-III and adaptive sensing operators [19], [20] for multi-objective optimisation, while also incorporating the structural features of a two-stage ADPLL.

A. Standard NSGA-III

The widespread adoption of NSGA-III in multi-objective optimisation is attributed to its three-stage procedure:

- fast non-dominated ranking;
- construction of uniformly distributed reference points on a M-dimensional hyperplane;
- preservation of population diversity through extreme point identification, normalisation, and association of reference vectors.

However, as the number of objectives increases to $M = 6$, NSGA-III, despite outperforming NSGA-II in distribution control, still faces challenges. the Pareto front of the multi-stage ADPLL exhibits a highly non-convex geometry, leading to fixed reference points that do not align well with the underlying manifold. In this system, Stages A (fast acquisition) and B (steady-state optimisation) are connected using an event-triggered mechanism. As shown in Fig. 2, the two loops blend via α-weighted fusion, producing a distinct piecewise manifold structure in the performance landscape.

B. Adaptive Operator

SA-NSGA-III adaptively adjusts the crossover rate P_c and mutation rate P_m according to spread [21] and hypervolume

(HV) [22]. When hypervolume improvements stagnate or population spread collapses, potentially indicating convergence to a local optimum, the algorithm increases P_m to enhance exploration. In contrast, if the Pareto front expands consistently and the hypervolume growth remains stable, both P_c and P_m are reduced to facilitate convergence. Furthermore, the adaptive operator can also select strategies based on the structural characteristics of the two-stage ADPLL.

- large fluctuations in Stage-A-dominated objectives (e.g., lock time) increase the mutation priority for Stage-A parameters (K_{pA}, K_{iA}, PD type);
- when Stage-B-dominated objectives (phase noise, steady-state error) dominate convergence, local search is intensified on Stage-B parameters;
- poor cross-stage behaviour (e.g., overshoot) triggers stronger mutations in transition-related parameters such as the α-curve, fade-in and fade-out window, and preset integrator state.

C. Reference Point Refinement

SA-NSGA-III implements a reference point refinement mechanism.

- reference points are adaptively redistributed around regions of high curvature on the current Pareto front, emphasising performance inflexion zones;
- independent reference point densities are maintained in the Stage-A-dominated and Stage-B-dominated regions to avoid structural imbalance;
- event-driven dimensions (e.g., ρ_f,T_{sw}) are weighted more heavily to improve sensitivity to switching mechanisms.

D. Static Optimization Procedure of SA-NSGA-III

The static optimisation process as shown in Algorithm 1 consists of three structure-aware phases: initialisation, adaptive evolution, and Pareto front construction.

1) Structure-Aware Population Initialization: The initial population is generated within a physically valid ADPLL parameter region. This includes setting a valid PD configuration, defining a suitable operating range K_{VCO}, selecting feasible two-stage PI gains, and applying constraints to event-driven parameters.

2) Adaptive and Structure-Guided Evolution: During the evolutionary process, the SA-NSGA-III algorithm jointly monitors population spread, hypervolume, and interactions between the two-stage parameters. Adaptive crossover and mutation operators are employed, together with a refined reference point. Optimisation is conducted for different objectives. Stagnation in lock-time improvement increases exploration of first-stage parameters, saturation of noise-related metrics prompts abrupt changes in Stage-B parameters, and anomalous overshoot amplifies perturbations in transition-related parameters.

3) Pareto Solutions with Consistent Structure: The final Pareto front balances all six objectives and maintains consistency with the two-stage ADPLL architecture. PI gains in stages A and B evolve naturally from capture to tracking, switching, and PD parameters, facilitating smooth event triggering, and K_{VCO} remains aligned with the steady-state error constraint.

Algorithm 1 Proposed SA-NSGA-III for Multi-Stage ADPLL

Input: Population size N, generations G, reference set $\mathcal{D}$, ADPLL simulator.
Output: Pareto-optimal parameter archive A.

Structure-aware initialization of P_0 (Stage A/B gains, K_{VCO}, PD type, events).
Evaluate P_0; initialize archive $A \leftarrow$ non-dominated members of P_0.
for $g = 0$ to $G-1$ **do**
  Adapt (P_c, P_m) based on HV/Spread.
  Generate offspring C by selection, SBX crossover, and structure-aware mutation.
  Evaluate C; merge $P_{\mathrm{all}} = P_g \cup C$.
  Perform fast non-dominated sorting; fill P_{g+1} with entire fronts.
  If needed, apply NSGA-III niching w.r.t. $\mathcal{D}$ to complete P_{g+1}.
  Update archive A and recompute HV/Spread.
  If HV stagnates, refine reference directions $\mathcal{D}$.
end for
Return archive A.

IV. Hybrid EA-DRL Optimization Framework

Multi-stage ADPLL require the global optimisation of static parameters and the real-time correction of dynamic behaviour. To address these requirements, a hybrid EA-DRL framework is introduced. Initially, SA-NSGA-III explores the global parameter space to establish a static physically consistent baseline. Subsequently, a twin delayed deep deterministic policy gradient (TD3) [23] DRL model performs real-time fine-tuning relative to this baseline to correct transient deviations during operation.

A. Framework Structure

1) Static Baseline Generation: SA-NSGA-III is used to investigate the six-dimensional objective landscape of the two-stage ADPLL, considering lock time, overshoot, steady-state error, noise, capture range, and lock range. The algorithm generates a Pareto set that maintains a consistent two-stage structure, allowing the identification of a knee solution or a weighted-best solution. $\mathbf{P}^*_{\mathrm{base}}$ is selected as the static baseline. This baseline ensures stable locking without DRL assistance and narrows the DRL search region to a physically valid domain, allowing DRL to focus on local refinements rather than global exploration.

2) Real-Time Dynamic Correction: During online step by step simulation, the DRL agent outputs incremental PI-gain corrections:

$$\Delta\mathbf{P}(t) = \big(\Delta K_{pA}, \Delta K_{iA}, \Delta K_{pB}, \Delta K_{iB}\big), \qquad (16)$$

These parameters are applied to the two-stage PI controller. The DRL policy operates near the structure-consistent optimum determined by SA-NSGA-III and avoids unsafe configurations. During training, $\mathbf{P}^*_{\mathrm{base}}$ is established in the environment as the initial parameter set and the corresponding metrics serve as a baseline for the comparison of rewards.

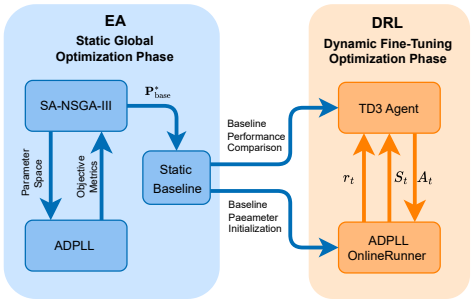

Fig. 3. Hybrid EA-DRL Optimization Framework: Static Global Search via SA-NSGA-III and Dynamic Fine-Tuning via TD3 Agent for Multi-Stage ADPLL.

B. State Space Design

The state vector is defined as

$$S_t = \left[\phi_e(t),\ \dot{\phi}_e(t),\ Switched,\ \widehat{K}_{pA}, \widehat{K}_{iA}, \widehat{K}_{pB}, \widehat{K}_{iB}\right], \tag{17}$$

where $\phi_e(t)$ is instantaneous phase error, $\dot{\phi}_e(t)$ its rate of change, $Switched$ indicates a Stage-A/B transition, and the last four elements are normalised PI gains. This design provides both transient observability and controller-awareness.

C. Action Space Design

The agent outputs a four-dimensional continuous action:

$$A_t = [a_{pA}, a_{iA}, a_{pB}, a_{iB}] \in [-1, 1]^4, \tag{18}$$

mapped to PI gains through log-domain scaling:

$$K(t{+}1) = K(t) \cdot \exp(\eta a), \tag{19}$$

where η is a learning-rate factor. Parameter updates are further bounded around $\mathbf{P}^*_{\text{base}}$ to ensure physical validity and prevent instability.

D. Reward Function Design

The reward consists of immediate (stage-dependent) rewards and a final trajectory-based reward.

1) Stage-A and Stage-B Immediate Rewards: Stage-A (fast acquisition, $Switched = 0$) emphasises rapid error reduction:

$$r_t^{(A)} = w_{\text{drop}} \cdot \max(|\phi_e(t{-}1)| - |\phi_e(t)|, 0) - w_{\text{abs}}|\phi_e(t)|. \tag{20}$$

Stage-B (steady-state refinement, $Switched = 1$) emphasises purity and stability:

$$r_t^{(B)} = -w_{\text{abs}}|\phi_e(t)| - w_{\text{rate}}|\dot{\phi}_e(t)| - w_{\text{ovr}} \cdot \text{Overshoot}. \tag{21}$$

2) Final Reward: The terminal reward considers lock time, overshoot, mean error, and RMS noise:

$$R_{\text{terminal}} = \frac{w_T}{1 + t_{\text{lock}}} - w_{\text{ovr}} \cdot \text{Overshoot} - w_{\text{mean}} \cdot \text{Mean} - w_{\text{rms}} \cdot \text{RMS}. \tag{22}$$

If the SA-NSGA-III baseline is provided, a differential bonus is added:

$$R_{\text{baseline}} = w_{\text{lock}}\left(t_{\text{lock}}^{\text{base}} - t_{\text{lock}}\right), \tag{23}$$

$$R_{\text{final}} = R_{\text{terminal}} + R_{\text{baseline}}. \tag{24}$$

3) Total Reward:

$$R_{\text{total}} = \sum_{t=0}^{T} r_t + R_{\text{final}}. \tag{25}$$

This reward formulation allows the DRL policy to enhance both transient dynamics and overall $\mathbf{P}^*_{\text{base}}$ final performance .

V. Experimental Studies

The experiment included two phases: static multi-objective optimisation and dynamic online control, to comprehensively evaluate the SA-NSGA-III and Hybrid EA-DRL frameworks. All experiments were run on a Personal Machine: 13th Gen Intel(R) Core(TM) i9-13900KF 3.00 GHz, 32.0 GB RAM, and Microsoft Windows 11 Professional Edition. The simulation code was compiled in Python 3.13.2 using Visual Studio Code 1.106.2.

A. Static Performance Analysis

To assess algorithmic performance, we compared NSGA-II [24], NSGA-III [18], and SA-NSGA-III on the ADPLL six-objective optimisation problem. Spread [21], hypervolume [22], and the proposed Intra-Population Quality Score (IPQS) are used to comprehensively evaluate the performance of multi-objective optimisation methods. Spread measures the distribution density of Pareto solutions, where a lower value indicates a more uniformly spread front. Hypervolume quantifies both convergence and diversity by measuring the volume of the objective space dominated by the Pareto front, with larger values indicating better overall multi-objective performance. In contrast, IPQS assesses the internal quality of each algorithm's Pareto set by evaluating the consistency of the solution's approximation to the algorithm's optimal performance region across all objectives.

The two-stage ADPLL was simulated with a 50 MHz loop clock and reference, a 1.9 GHz free-running DCO, and a 40× multiplier. The NSGA-III optimiser explored a 14-dimensional decision space comprising Stage-A/B phase detector types, loop gains, proportional–integral coefficients, DCO gains, and switching-control parameters, including frequency tolerance ratio, phase-error threshold, switching time, and preset scale. A population size of 180 was maintained over 120 generations and six objective functions were evaluated for each solution. A bounded external archive was used to maintain the non-dominated set. The complete configuration is provided in Table II.

A comparative analysis of NSGA-II [24], NSGA-III [18], and the proposed SA-NSGA-III highlights the advantages of the structure-aware design. As shown in Fig.4 Left, SA-NSGA-III exhibits the fastest hypervolume growth and maintains the highest hypervolume throughout evolution, while Fig.4 Right shows that its spread value remains the lowest and most stable, indicating a more uniformly distributed and better-converged Pareto front. The quantitative statistics in TableIII, where SA-NSGA-III achieves a final HV of 644.7 (enhancing NSGA-II by 73.5% and NSGA-III by 58.7%, an HV-AUC of 628 (improving NSGA-II by 52.8% and NSGA-III by 10.2%),

TABLE II
SUMMARY OF SIMULATION AND OPTIMIZATION SETTINGS

Global System Parameters (ADPLL Simulation)	
Loop update frequency ($f_{\mathrm{s,loop}}$)	50 MHz
Reference frequency (f_{ref})	50 MHz
Simulation duration (T_{sim})	100 μs
Free-running DCO frequency ($f_{\mathrm{dco,free}}$)	1.9 GHz
Divider ratio (N_{div})	40
Target output frequency (f_{out})	2.0 GHz
General Evolution Settings (NSGA Optimization)	
Population size	180
Number of generations	500
Number of objectives	6
Decision space dimension	14
Archive size ($\lvert A_{\max}\rvert$)	540 (3× population)

and the smallest spread (0.006956), representing a reduction of 64.2% and 63.1%, respectively. To explain, AUC (Area Under the Curve) represents the integral of the metric across all generations, measuring the overall performance throughout the optimisation process.

In addition to convergence behaviour, Fig.5 presents the IPQS distribution, calculated using min-max normalisation per-population with equal sample sizes across all algorithms. SA-NSGA-III yields a distribution that is both more concentrated and shifted toward higher values, indicating superior internal solution quality and more balanced multi-objective trade-offs. Statistical analysis supports this finding: SA-NSGA-III achieves a mean score of 0.7687 and a median of 0.7817, surpassing NSGA-II by 32.21% and NSGA-III by 13.09%. These results demonstrate that even under relative normalisation, SA-NSGA-III consistently produces a higher proportion of high-quality Pareto solutions.

Together, the evidence from hypervolume, spread, and IPQS shows that SA-NSGA-III achieves faster convergence, higher solution quality, and substantially improved distribution uniformity compared to NSGA-II and NSGA-III, demonstrating the effectiveness of the proposed structure-aware enhancements.

TABLE III
COMPARISON OF HYPERVOLUME, SPACING (STD), AND CONVERGENCE ITERATIONS

Algorithm	**HV (final) ↑**	**HV (AUC) ↑**	**Spread ↓**
NSGA-II	371.5	411.1	0.01954
NSGA-III	406.3	569.9	0.01847
SA-NSGA-III	**644.7**	**628**	**0.006956**

Note: AUC refers to the Area Under the Curve

The Pareto front generated by the proposed SA-NSGA-III algorithm shows the trade-offs between metrics such as lock-in time, phase noise, overshoot, and frequency range. The pairwise scatter plot reveals three main solution regions: the fastest lock-in, the lowest-noise, and the knee point. The knee-point solution offers the most balanced performance: its lock-in time is substantially shorter than the noise-optimal

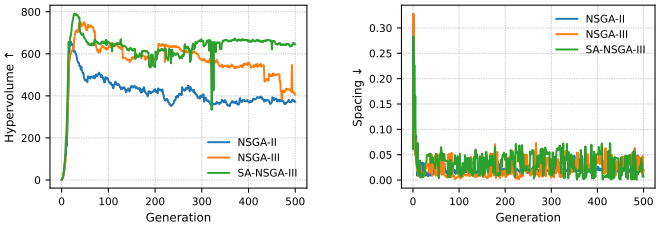

Fig. 4. Evolution of optimization metrics across generations: Left: Hypervolume convergence curves; Right: Spacing metric evolution for NSGA-II, NSGA-III, and SA-NSGA-III.

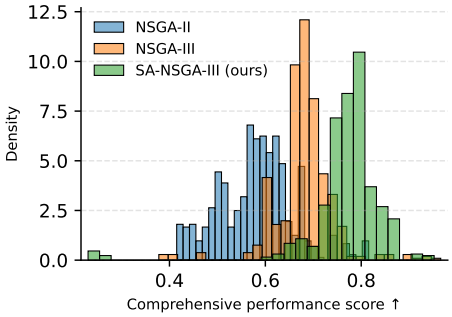

Fig. 5. Comparison of Comprehensive Performance Score Distributions for NSGA-II, NSGA-III, and SA-NSGA-III Algorithms.

solution, its phase noise is much lower than the lock-in-time-optimal solution, and it also provides improved acquisition and lock-in ranges. Parallel coordinate visualisation confirms that the knee-point solution balances all six objectives. This solution will serve as the basis $\mathbf{P}^{*}_{\mathrm{base}}$ for upcoming dynamic-optimisation experiments.

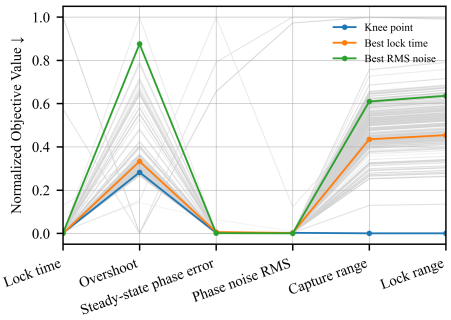

Fig. 6. Comparison of Normalized Objective Values for Knee Point and Best Individual Objectives.

B. Dynamic Performance Analysis

In the dynamic experiment, the TD3 agent was trained using the Stable Baselines3 library [25], which offers robust PyTorch implementations of DRL algorithms. The policy network comprises two hidden layers with 256 units each and employs a ReLU activation function. The learning rate decays linearly from 3×10^{-4} to 10%. The replay buffer size is set to 8×10^{5}, the training begins after 2,000 steps, and the batch

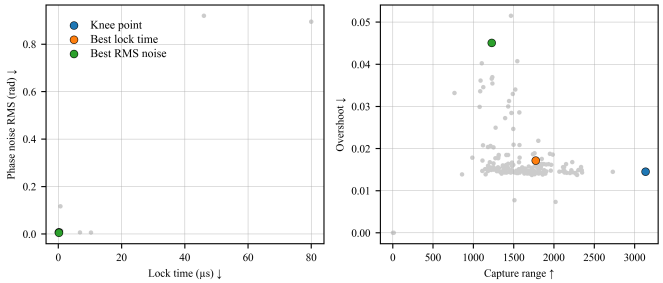

Fig. 7. Pareto Front Distribution and Optimal Trade-off Solutions(Left: Lock Time vs Steady-State Phase Noise RMS; Right: Capture Range vs Overshoot).

size is 512. The target network is updated with $\tau = 0.005$ and the policy delay is set to 2. Gaussian exploration noise decays from 0.20 to 0.05 during training. The training environment is a two-stage stepwise ADPLL simulation that replicates the behaviour of the baseline PLL model, including TDC, PI, and DCO updates, stage switching logic, and lock detection.

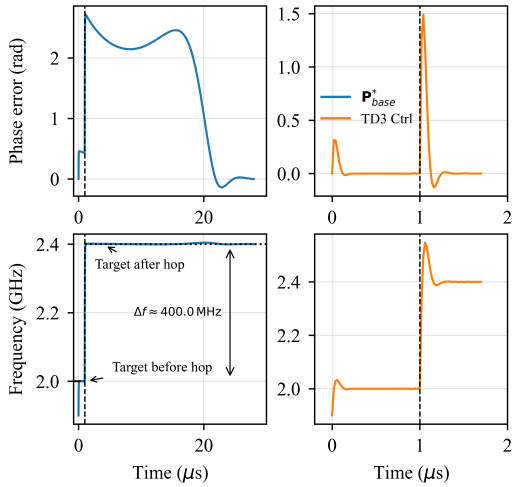

Fig. 8. Frequency-hop response of the two-stage ADPLL: baseline static control vs. RL-enhanced TD3 control.

TABLE IV
POST-FREQUENCY-HOP PERFORMANCE COMPARISON

Objective	$\mathbf{P}^*_{\text{base}}$	***DRL***
Lock time (μs)	24.04	**0.52**
Overshoot ratio	0.00177	0.06112
Phase error (rad)	1.3487	**0.1327**
Phase jitter (ps)	114.06	**27.266**

As shown in Fig. 8 and Table IV, when the reference frequency rises from 50 MHz to 60 MHz (corresponding to a 400 MHz output shift with $N = 40$), the TD3-controlled ADPLL achieves substantially faster dynamic recovery and superior steady-state precision compared to the static two-stage baseline $\mathbf{P}^*_{\text{base}}$. The reset time is reduced from 24.04 μs to 0.52 μs (a 97.8% improvement), and the steady-state phase error decreases from 1.3487 rad to 0.1327 rad (a reduction of 90.2%). Although the reinforcement-learning controller introduces a higher transient overshoot (0.06112 vs. 0.00177), the loop remains well anchored and rapidly convergent. Furthermore, the phase-jitter level improves from 114.06 ps to 27.266 ps (a reduction of 76.1%, reflecting enhanced loop stability and noise suppression in the fine-lock regime.

TABLE V
PVT CORNER COMPARISON OF STATIC PLL AND RL-CONTROLLED ADPLL

PVT	**Method**	t_{lock} (μs)	Overshoot	$\mathbb{E}[\phi_{\text{err}}]$ (rad)	RMS (ps)
Nominal	Static	0.212	0.015	0.000	0.0796
Nominal	RL	0.420	0.065	0.193	41.43
Fast	Static	13.816	0.845	-0.015	1.67
Fast	RL	**0.816**	**0.000**	0.050	16.23
Low-V_{DD}	Static	0.432	0.016	-0.000	0.0796
Low-V_{DD}	RL	0.928	0.025	0.378	66.13
High-Temp	Static	1.892	0.015	-0.000	0.0796
High-Temp	RL	**0.560**	0.079	0.230	55.53

To assess the robustness of PVT variations, we conducted simulations applying proportional scaling to stage-A/B DCO gains and the loop time-constant parameter τ_{res} to construct Nominal (1.0,1.0,1.0), Fast (1.4,1.4,0.6), Low-V_{DD} (0.6,0.6,1.0) and High-Temperature $(0.95, 0.95, 1.4)$ corners. This introduces roughly $\pm 40\%$ variations in the dominant loop parameters while ensuring identical perturbations for static and RL-controlled loops. As shown in Table V, the TD3-controlled ADPLL maintains strong robustness under non-nominal conditions, achieving substantial reductions in relock time 94.1% in the fast corner (13.816 μs to 0.816 μs) and 70.4% in the high-temperature corner (1.892 μs to 0.560 μs) while preserving stable steady-state behaviour. These gains demonstrate the ability of the learnt policy to generalise to unseen loop-gains and delay variations. Although the Low corner V_{DD} yields an elevated steady-state error due to severely reduced DCO sensitivity not present in training, the RL loop remains stable and within lock. In general, TD3-controlled ADPLL offers improved dynamic resilience across the most challenging PVT variations compared to static baseline.

VI. CONCLUSION

This study introduces a hybrid evolutionary algorithm–deep reinforcement learning (EA–DRL) optimisation framework that integrates global static parameter exploration with real-time adaptive control for multi-stage all-digital phase-locked loop (ADPLL) systems. Incorporating a structure-sensitive NSGA-III variant enables efficient identification of high-quality, physically consistent Pareto solutions, providing robust baselines for subsequent dynamic learning. Subsequently, a twin-delayed deep deterministic policy gradient (TD3) agent further refines the transient response and steady-state accuracy through online fine-tuning. Experimental results indicate that the proposed approach achieves faster settling times, reduced jitter, and substantially improved resilience to PVT variations and large frequency steps compared to static two-stage PLL configurations. Future efforts will train the RL controller under richer stochastic and non-linear environments to improve generalisation, and will validate the proposed framework through

hardware-in-the-loop experiments and FPGA prototypes for practical deployment.

ACKNOWLEDGMENT

The author gratefully acknowledges Prof. Qin Zhang for her guidance, Xinyuan Zhong for providing computational resources, and Zhenwen Liao for his assistance with the early work.

REFERENCES

[1] Z. Ali, P. Paliwal, M. Ahmad, H. Heidari, and S. Gupta, "Fast settling phase-locked loops: A comprehensive survey of applications and techniques [feature]," *IEEE Circuits and Systems Magazine*, vol. 24, no. 2, pp. 62–79, 2024.

[2] R. B. Staszewski and P. T. Balsara, *All-Digital Frequency Synthesizer in Deep-Submicron CMOS*. Hoboken, NJ, USA: John Wiley & Sons, 2006.

[3] W. El-Halwagy, A. Nag, P. Hisayasu, F. Aryanfar, P. Mousavi, and M. Hossain, "A 28-ghz quadrature fractional-n frequency synthesizer for 5g transceivers with less than 100-fs jitter based on cascaded pll architecture," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 2, pp. 396–413, 2017.

[4] Y. Zhu, H. Zhang, and W. Hong, "A frequency agile synthesizer using dds and pll techniques for fmcw radar," in *2015 Asia-Pacific Microwave Conference (APMC)*, vol. 2. IEEE, 2015, pp. 1–3.

[5] F. Herzel, C. Carta, and G. Fischer, "Phase jitter analysis of cascaded plls for millimeter wave applications in silicon-based radar and communication systems," in *2025 IEEE Radar Conference (RadarConf25)*. IEEE, 2025, pp. 658–663.

[6] D. Banerjee, *PLL performance, simulation and design*. Dog Ear Publishing, 2006.

[7] S. M. Dartizio, F. Buccoleri, F. Tesolin, L. Avallone, A. Santiccioli, A. Iesurum, G. Steffan, D. Cherniak, L. Bertulessi, A. Bevilacqua *et al.*, "A fractional-n bang-bang pll based on type-ii gear shifting and adaptive frequency switching achieving 68.6 fs-rms-total-integrated-jitter and 1.56 μs-locking-time," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 12, pp. 3538–3551, 2022.

[8] S. M. Dartizio, F. Buccoleri, F. Tesolin, L. Avallone, G. Santiccioli, D. Cherniak, L. Bertulessi, A. Bevilacqua *et al.*, "A 68.6 fs rms-total-integrated-jitter and 1.56 μs-locking-time fractional-n bang-bang pll based on type-ii gear shifting and adaptive frequency switching," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65. IEEE, 2022, pp. 1–3.

[9] M. A. Scarpato, "Digital circuit performance estimation under pvt and aging effects," Ph.D. dissertation, Université Grenoble Alpes, 2017.

[10] D. Dutta, S. P. Tumukunta, N. Sivaraaj, and K. A. Majeed, "Exploring the landscape of phase-locked loop architectures: A comprehensive review," *IEEE Access*, 2024.

[11] T.-C. Wang, S. Lall, and T.-Y. Chiou, "Polynomial method for pll controller optimization," *Sensors*, vol. 11, no. 7, pp. 6575–6592, 2011.

[12] M. K. Rajak and R. Pudur, "Mathematical modelling and dynamic optimization of phase-locked loop systems using hybrid pso-gradient descent approach," *Systems Science & Control Engineering*, vol. 13, no. 1, p. 2448636, 2025.

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE signal processing magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[14] M. Gautam, "Deep reinforcement learning for resilient power and energy systems: Progress, prospects, and future avenues," *Electricity*, vol. 4, no. 4, pp. 336–380, 2023.

[15] G. Tan, W. Jiang, J. Gao, J. Dou, L. Zhong, J. Di, and Y. Qin, "Phase-locked control of the coherent beam combining system using dual-stream network and reinforcement learning," *Optics and Lasers in Engineering*, vol. 186, p. 108830, 2025.

[16] T. V. H. Nguyen and C.-K. Pham, "An overview of phase-locked loop: From fundamentals to the frontier," *Sensors (Basel, Switzerland)*, vol. 25, no. 18, p. 5623, 2025.

[17] T. Yamaoka and T. Oshima, "Sidelobe reduction for window functions by multiplication of cosine-sum windows," *IEEE Access*, vol. 13, pp. 35 588–35 596, 2025.

[18] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints," *IEEE transactions on evolutionary computation*, vol. 18, no. 4, pp. 577–601, 2013.

[19] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 4, pp. 656–667, 2002.

[20] J. Zhang, H. S.-H. Chung, and W.-L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," *IEEE Transactions on evolutionary Computation*, vol. 11, no. 3, pp. 326–335, 2007.

[21] M. Li and J. Zheng, "Spread assessment for evolutionary multi-objective optimization," in *International conference on evolutionary multi-criterion optimization*. Springer, 2009, pp. 216–230.

[22] H. Ji and C. Dai, "A simplified hypervolume-based evolutionary algorithm for many-objective optimization," *Complexity*, vol. 2020, no. 1, p. 8353154, 2020.

[23] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[25] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of machine learning research*, vol. 22, no. 268, pp. 1–8, 2021.