Orthrus: Dual-Loop Automated Framework for System-Technology Co-Optimization

Yi Ren[1,2,†], Baokang Peng[3,†], Chenhao Xue[1], Kairong Guo[1], Yukun Wang[4], Guoyao Cheng[3], Yibo Lin[1,5,6], Lining Zhang[3,5,*], Guangyu Sun[1,5,6,*]

[1]*School of Integrated Circuits*, [2]*School of Software and Microelectronics, Peking University*, Beijing, China

[3]*School of Electronic and Computer Engineering, Peking University*, Shenzhen, China

[4]*School of Electronics Engineering and Computer Science, Peking University*, Beijing, China

[5]*Institute of Electronic Design Automation, Peking University*, Wuxi, China

[6]*Beijing Advanced Innovation Center for Integrated Circuits*, Beijing, China

{yiren20, baokangpeng}@stu.pku.edu.cn, {eelnzhang, gsun}@pku.edu.cn

***Abstract*—With the diminishing return from Moore's Law, system-technology co-optimization (STCO) has emerged as a promising approach to sustain the scaling trends in the VLSI industry. By bridging the gap between system requirements and technology innovations, STCO enables customized optimizations for application-driven system architectures. However, existing research lacks sufficient discussion on efficient STCO methodologies, particularly in addressing the information gap across design hierarchies and navigating the expansive cross-layer design space. To address these challenges, this paper presents Orthrus, a dual-loop automated framework that synergizes system-level and technology-level optimizations. At the system level, Orthrus employs a novel mechanism to prioritize the optimization of critical standard cells using system-level statistics. It also guides technology-level optimization via the normal directions of the Pareto frontier efficiently explored by Bayesian optimization. At the technology level, Orthrus leverages system-aware insights to optimize standard cell libraries. It employs a neural network-assisted enhanced differential evolution algorithm to efficiently optimize technology parameters. Experimental results on 7nm technology demonstrate that Orthrus achieves 12.5% delay reduction at iso-power and 61.4% power savings at iso-delay over the baseline approaches, establishing new Pareto frontiers in STCO.**

***Index Terms*—system-technology co-optimization, standard cell library, circuit analysis**

I. Introduction

Fabless-foundry business model serves as a cornerstone of modern VLSI industry, where fabless companies specialize in circuit design while foundries focus on manufacturing. The division of labor narrows the optimization objectives to specific domains, thereby facilitating decades of rapid industrial advancement. Unfortunately, the fabless-foundry model is now facing fundamental limitations. With design methodologies and associated automation tools reaching high maturity, further gains from design-level optimizations alone yield diminishing returns. Additionally, manufacturing process scaling is approaching its physical limits. To sustain the continued growth of the VLSI industry, deeper collaboration between fabless companies and foundries is becoming imperative, requiring a shift towards system-technology co-optimization (STCO) to unlock new performance and efficiency gains. According to Imec's roadmap [1], STCO is expected to play an increasingly vital role, particularly for application-driven system architectures.

Conceptually, STCO aims to integrate multiple design hierarchies listed in Fig. 1(a), encompassing architecture design, logic synthesis, physical design, process design kit (PDK) development, and technology development. Each individual optimization level has been extensively investigated in prior research. At the architectural level, design space exploration (DSE) has been studied on various computing platforms, including CPU [2], [3], AI accelerators [4], [5], high-level synthesis [6], [7], and beyond. Similarly, numerous algorithms have been proposed to optimize the tunable parameters of logic synthesis tools and physical design tools [8]–[10]. At the technology level, considerable research has focused on optimizing process parameters to enhance intrinsic device performance [11]–[13] and standard cell performance [14], [15]. In parallel, numerous studies have concentrated on improving the efficiency of standard cell characterization [16], [17] and the generation of standard cell layouts [18], [19].

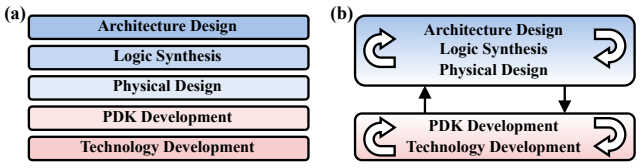


Fig. 1. (a) Full-stack VLSI flow. (b) Our dual-loop STCO.

Unfortunately, despite extensive research on optimizations at individual design levels, the academic community lacks a systematic discussion of holistic optimization across the entire design hierarchy. This gap limits the translation of STCO's theoretical benefits to practical performance improvements.

On the one hand, a straightforward approach involves integrating multiple design hierarchies into a unified design space, where all relevant parameters are jointly optimized to maximize end-to-end quality-of-results (QoR). Although this methodology shows promise for joint optimization of adjacent design levels, such as system level DSE [20]–[22] and design and technology co-optimization (DTCO) [23], [24], it suffers from fundamental scalability limitations when extended to the full STCO optimization chains: Firstly, a full-system evaluation using the complete design flow may take hours to days, rendering iterative optimization impractical; Secondly, the resulting high-dimensional design space exceeds the capabilities of existing DSE algorithms and cannot be efficiently navigated.

On the other hand, we can retain the original design hierarchy and carefully coordinate their interactions to achieve overall benefits. However, establishing effective synergy across design levels remains a fundamental challenge. In the context of STCO, the primary challenge lies in bridging the gap between system-level performance, power, and area (PPA) metrics and technology innovations. Without

This work is supported in part by Beijing Natural Science Foundation (Grant No. L243001), National Natural Science Foundation of China (Grant No. 62032001, 62034007), National Key Research and Development Program of China (Grant No. 2023YFB4402204, 2021ZD0114702), and 111 Project (B18001).

[†]Co-first authors. [*]Corresponding authors.

TABLE I
CROSS-LAYER DESIGN SPACE OF ORTHRUS.

ID	Level	Parameter	Description	Candidate Values	Default Value
1	Architecture	ct_type	compressor tree type	WT,DT	WT
2		cpa_type	carry-propagate adder type	SK,KS,BK	SK
3	Logic Synthesis	clock_period_ns	target clock period	range(0.4,1.0)	0.5
4		syn_generic_effort	generic synthesis effort	low,medium,high	medium
5		syn_map_effort	technology mapping effort	low,medium,high	high
6		syn_opt_effort	post-mapping optimization effort	none,low,medium,high	none
7	Physical Design	place_utilization	floorplan utilization ratio	range(0.5,0.9)	0.8
8		place_glb_cong_effort	effort for relieving congestion in global placement	auto,low,medium,high	auto
9		place_glb_timing_effort	effort for timing-driven global placement	medium,high	medium
10		place_glb_clk_power_driven	enable clock tree power optimization in global placement	true,false	true
11	Technology	phig_n	nmos gate workfunction	range(4.302,4.312)	4.307
12		phig_p	pmos gate workfunction	range(4.8631,4.8731)	4.8681
13		hfin_nm	height of fin	range(28,36)	32
14		tfin_nm	thickness of fin	range(5.8,7.2)	6.5
15		lg_nm	horizontal length of the GATE layer	range(17,23)	20
16		lext_nm	horizontal distance between the gate and the SDT layer	4,5,6	5
17		lct_nm	horizontal length of the SDT layer	range(19,29)	24

visibility into standard cell criticality or well-defined guidance, optimization at the technology level cannot effectively mitigate system performance bottlenecks. While prior work has explored area reduction through merging common standard cell combinations [25], [26], it remains an open problem to jointly address timing optimization, power reduction, and achieving intricate trade-offs among competing PPA objectives.

To address the above challenges, this paper introduces **Orthrus**, an automated framework to enable system-technology co-optimization, as shown in Fig. 1(b). Orthrus employs two synergistic optimization loops: The *system loop* identifies Pareto-optimal parameters and collects data for directing technology optimization. The *technology loop* leverages system-level guidance to selectively optimize process parameters and standard cell layouts. The *inter-loop direction* analyzes data from the system loop and guides the technology loop.

The main contributions of this paper are as follows:

- We propose Orthrus, an automated STCO framework equipped with synergetic optimization loops.
- We propose a novel coordination mechanism that synergizes the system loop and technology loop by analyzing cell contributions, subcircuit frequencies, and PPA optimization directions.
- We propose a system optimization loop that leverages multi-objectives Bayesian optimization to efficiently identify the Pareto frontier while collecting data.
- We propose a technology optimization loop that leverages system-level guidance and employs a neural networks-assisted differential evolution algorithm to efficiently optimize technology parameters.
- Orthrus achieves a 33.2% PPA hypervolume improvement under advanced 7nm technology, delivering 12.5% delay reduction at iso-power and 61.4% power savings at iso-delay.

The remainder of this paper is organized as follows: Section II provides preliminaries on graph matching and problem formulation. Section III details the Orthrus framework. Section IV presents the evaluation results. Finally, Section V concludes the paper.

II. PRELIMINARIES

A. Graph Matching

Graph matching is a fundamental problem concerned with establishing correspondences or identifying structural similarities between graphs. It finds broad application in domains such as computer vision, pattern recognition, and circuit design. Classical approaches

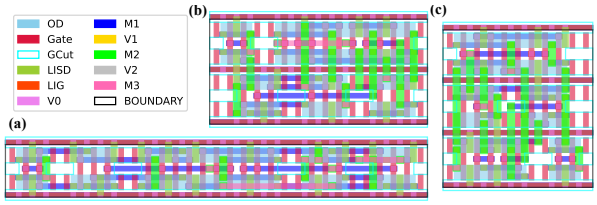


Fig. 2. Layout of fused full-adder circuit (38 transistors): (a) Single-row configuration (num_rows = 1) with 31 CPP width; (b) Two-row folded layout (num_rows = 2) with 16 CPP width; (c) Three-row folded arrangement (num_rows = 3) with 12 CPP width.

to graph matching include backtracking, depth-first search (DFS), and constraint-based pruning [27]. In Electronic Design Automation (EDA), standard cell netlists are commonly represented as graphs, making graph matching techniques highly relevant. Subgraph isomorphism detection, a key aspect of graph matching, plays a crucial role in tasks such as Layout vs Schematic (LVS) verification and positioning of Integrated Clock Gating (ICG) cells [28]. Specifically, a standard cell netlist G is isomorphic to netlist H if there exists a bijection mapping between their standard cell sets that preserves cell interconnections. Subgraph isomorphism detection aims to find all subgraphs within netlist G that are isomorphic to an arbitrary query subgraph $H \subseteq G$.

The discovery of isomorphic subgraphs enables various optimization opportunities, including standard cell merging. Prior works such as AutoCellLibX [25] and TeMACLE [26] propose to merge frequent subcircuits for area reduction, utilizing the blank space within simple cells. However, these approaches focus solely on area reduction, neglecting cell delay and power consumption. As shown in Fig. 2, Orthrus overcomes this limitation by incorporating multirow standard cell layout synthesis, which shortens the critical net length for improved delay and lower power dissipation [29].

B. Problem Formulation

Orthrus employs a fully automated design flow that integrates EDA tools across multiple design levels. These tools offer a wide range of tunable parameters, creating an enormous cross-layer design space. TABLE I summarizes the target design hierarchy and its associated parameters, detailed as follows:

- *Architecture*: In Orthrus, we validate the efficacy of STCO methodology on application-driven system architectures, as customized

TABLE II
ADOPTED STANDARD CELLS FROM ASAP7 6T LIBRARY

Category	Standard Cells	Row Count
Basic Cell	AND2x2, AND2x4, AND3x1, NAND2x1, NAND2x2, NAND3x1, OR2x2, OR2x4, OR3x1, NOR2x1, XNOR2x2, XOR2x2 INVx1, INVx2, INVx4, INVx8, BUFx2, BUFx4, BUFx8 MAJx1, MAJx2, AOI21x1, AO21x1, AO22x1, OA21x1, OA22x1	1
Fused Cell	*Extracted from frequent subcircuit patterns*	{1,2,3}

design optimization for these architectures is expected to yield substantial practical benefits. Specifically, Orthrus targets the multiply-accumulator (MAC) arrays, a key component in AI accelerators that play a crucial role in determining the PPA of the entire system [30]. We employ an 8×8 systolic array with MAC units interconnected via pipeline registers. Each MAC unit incorporates a parallel multiplier architecture, comprising a partial product generator, a compressor tree, and a carry-propagate adder. We select compressor tree from Wallace Tree (WT) and Dadda Tree (DT), and carry-propagate adder from Sklansky adder (SK), Kogge-Stone adder (KS), and Brent-Kung adder (BK). An in-house RTL generator is developed to translate the MAC array configuration into Verilog HDL codes.

- *Logic Synthesis*: The logic synthesis tool converts RTL implementations into standard cell netlists. Orthrus employs Cadence Genus for logic synthesis and adjusts the target frequency as well as synthesis efforts.
- *Physical Design*: The physical design tool places and routes the standard cell netlists into a manufacturable circuit layout. In Orthrus, we employ Cadence Innovus for physical implementation. We mainly consider the design options at the global placement stage, since these options demonstrate a significant impact on PPA outcomes [10].
- *Technology*: To explore the parameters involved in technology optimization, Orthrus utilizes a customized ASAP7 open-source PDK [31] as an exemplary demonstration platform. We employ the calibrated model card from ASAP7 as the baseline model and adjust several model instance parameters. Besides, we adjusted the layout-related parameters of the standard cell and ensured that these adjustments satisfied the constant CPP requirement, as defined by the following equation:

$$CPP = L_g + 2 * L_{ext} + L_{ct} \tag{1}$$

For the cell layout, M1 and M3 are configured with 1D horizontal routing, while LISD and M2 use 1D vertical routing. For each standard cell, we validate the layout using Mentor Calibre to perform Design Rule Check (DRC), LVS, and Parasitic Extraction (PEX) checks, and extract the corresponding parasitics. Additionally, we use Cadence Liberate for delay and power characterization of the standard cells, generating the timing library (.lib). Cadence Abstract is employed to generate the physical library (.lef).

Definition 1 (Tunable Parameter Design Space) *A tunable parameter configuration* $\mathbf{p}$ *is defined as a combination of candidate values given in TABLE I. The feature vector* $\mathbf{p} = (\mathbf{p}_{arch}, \mathbf{p}_{ls}, \mathbf{p}_{pd}, \mathbf{p}_{tech})$ *can be decomposed into multiple segments, each corresponds to the tunable parameters of a specific design level. The complete parameter design space* $\mathcal{D}_{param}$ *constitutes the set of all feasible parameter configurations.*

In addition to adjusting the tunable parameters of EDA tools, Orthrus investigates the layout customization of individual standard cells. As illustrated in TABLE II, Orthrus selects several fundamental standard cells from ASAP7 to establish the initial standard cell library, and extend the library by fusing subcircuits into new standard cells. We develop a C++ program for the automatic generation of multi-row standard cell layouts following [29], abbreviated as `StdGen`. In a nutshell, given the SPICE netlist of specific standard cells and target number of rows, `StdGen` systematically explores transistor placement while considering intra-cell routability, followed by SAT-based routing to ensure compliance with design rules. The generated layouts undergo DRC, LVS, PEX, and characterization, yielding optimized standard cells that replace the original ones in the subsequent design stages.

Definition 2 (Cell Layout Design Space) *Given standard cell* c*, let* $\mathcal{L}(c, R_c)$ *denote the set of* c*'s feasible layouts whose row count falls in set* R_c*. For a standard cell library* $\mathcal{C}$ *given in TABLE II, the cell layout design space* $\mathcal{D}_{cell} = \prod_{c \in \mathcal{C}} \mathcal{L}(c, R_c)$ *is defined as the Cartesian product of feasible layout sets for all standard cells in* $\mathcal{C}$*.*

Through joint optimization of the tunable parameters and standard cell layouts, Orthrus targets system-level improvements in PPA. Typically, these objectives are conflicting, where advancing one may degrade others. At the cell level, reducing the threshold voltage improves latency while increasing leakage power, and transistor width expansion enhances drive strength at the expense of a larger area. These trade-offs propagate to the system level, where performance gains incur either increased power dissipation or area overhead. In this context of multi-objective optimization, the optimal solutions form a Pareto frontier, where no PPA metrics can be further improved without deteriorating others. Since the true Pareto set cannot be obtained within limited trials in practical STCO scenarios, our objective is to advance the explored Pareto frontier, which is quantitatively measured by hypervolume improvement w.r.t. a reference point. Formally, our problem formulation and the related terminologies are defined as follows:

Definition 3 (Performance) *The performance is defined as the maximum attainable frequency of the MAC array, which is determined by the maximum delay of all timing paths.*

Definition 4 (Power) *The power is defined as the average power dissipation when the MAC array operates at the maximum attainable frequency.*

Definition 5 (Area) *The area is defined as the size of the floorplan in which the MAC array is placed and routed.*

Definition 6 (Pareto Frontier) *Let objective vector* $\mathbf{y}$ *denote the PPA metrics.* $\mathbf{y}$ *is said to be Pareto-dominated by* $\mathbf{y}'$ *(denoted as* $\mathbf{y} \preceq \mathbf{y}'$*) if the following condition satisfies:*

$$\begin{aligned} &\forall i \in [1,3], \quad \mathbf{y}'[i] \leq \mathbf{y}[i]; \\ &\exists j \in [1,3], \quad \mathbf{y}'[j] < \mathbf{y}[j]. \end{aligned} \tag{2}$$

Given a set of objective vectors $\mathcal{Y}$*, its Pareto frontier is defined as a subset* $\mathcal{Y}^* = \{\mathbf{y} | \mathbf{y} \npreceq \mathbf{y}', \forall \mathbf{y}' \in \mathcal{Y}\}$*.*

Definition 7 (Hypervolume) *Given a set of objective vectors* $\mathcal{Y}$ *and a reference point* $\mathbf{y}_{ref}$ *that is strictly dominated by all* $\mathbf{y} \in \mathcal{Y}$*, the hypervolume (HV) is calculated as the Lebesgue measure of the dominated space:*

$$HV(\mathcal{Y}^*, \mathbf{y}_{ref}) = \int_{\mathbb{R}^3} \mathbf{1}[\exists \mathbf{y}' \in \mathcal{Y}^*, \mathbf{y}' \preceq \mathbf{y} \preceq \mathbf{y}_{ref}] \, d\mathbf{y} \tag{3}$$

Problem 1 (System-Technology Co-Optimization) *For subset* $\mathcal{X} \subset \mathcal{D}_{param} \times \mathcal{D}_{cell}$ *sampled from the joint design space of tunable parameters and standard cell layouts, its corresponding set of PPA metric* $\mathcal{Y}$ *can be obtained through the VLSI flow. Given limited invocation of the VLSI evaluation flow, the objective of Orthrus is to obtain* $\mathcal{X}$ *such that the hypervolume* $HV(\mathcal{Y}, \mathbf{y}_{ref})$ *can be maximized.*

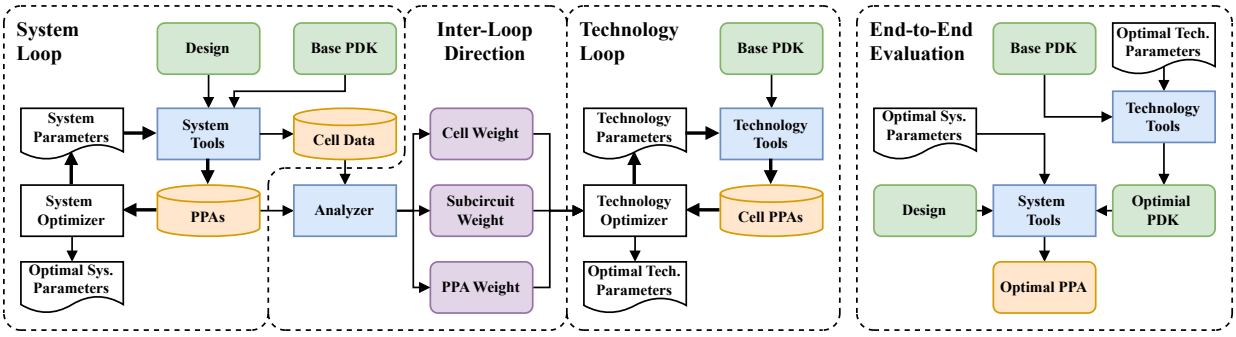


Fig. 3. Overview of Orthrus. The system loop and technology loop search for optimal parameters at their respective levels. The inter-loop direction analyzes system loop data to guide technology loop optimization. End-to-end evaluation provides the PPA of optimal parameters.

Algorithm 1: Bayesian Optimization

Input : Parameter Space $\mathcal{D}$, Maximum Iteration t_{max}
Output: Pareto frontier $\mathcal{Y}^*$ and corresponding Parero set $\mathcal{X}^*$

Initialize $\mathcal{X}_0$ via random sampling from $\mathcal{D}$;
Evaluate $\mathcal{Y}_0$ via toolchain;
for $t \leftarrow 1$ **to** t_{max} **do**
  Train surrogate model M on $(\mathcal{X}_{t-1}, \mathcal{Y}_{t-1})$;
  Select $\mathbf{x}_t = \arg\max \alpha(\mathbf{x})$;
  Evaluate $\mathbf{y}_t$ via toolchain;
  Update $\mathcal{X}_t = \mathcal{X}_{t-1} \cup \{\mathbf{x}_t\}$, $\mathcal{Y}_t = \mathcal{Y}_{t-1} \cup \{\mathbf{y}_t\}$;
return Pareto frontier $\mathcal{Y}^*$ and corresponding Parero set $\mathcal{X}^*$

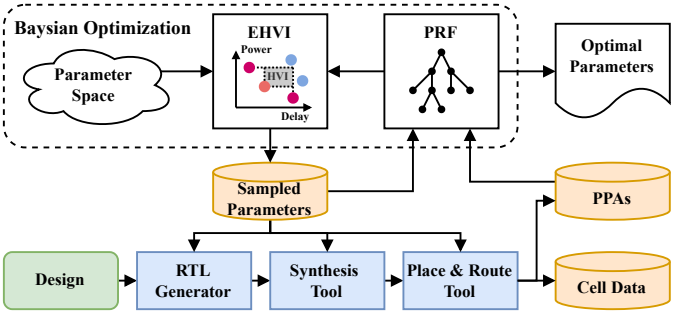


Fig. 4. System loop uses EHVI and PRF for Bayesian optimization, along with an EDA toolchain to extract PPA metrics and cell data.

III. Methodology

A. Framework Overview

The overview of the Orthrus framework is shown in Fig. 3. First, the *system loop* explores the system design space using Bayesian optimization, identifies Pareto-optimal parameters, and collects cell data (Section III-B). Next, the *inter-loop direction* analyzes this data through a novel mechanism to prioritize critical cells, suggest fusion candidates, and guide technology optimization (Section III-C). Then, the *technology loop* optimizes technology parameters based on these system-aware insights via a neural network-assisted heuristic algorithm (Section III-D). Finally, *end-to-end evaluation* provides the final PPA results of these optimal parameters.

B. System Loop

The proposed system loop framework, as shown in Fig. 4, aims to identify Pareto-optimal parameters and collect cell data for directing technology optimization. Bayesian Optimization (BO) [32] is adopted to navigate high-dimensional parameter spaces efficiently, overcoming the computational infeasibility of brute-force sampling in multi-objective optimization. By synergizing surrogate modeling with automated design toolchains, the framework balances exploration of under-sampled regions and exploitation of known high-performance solutions, while efficiently correlating system parameters with PPA.

Bayesian Optimization. The BO algorithm iteratively refines parameter selections using a surrogate model M and an acquisition function $\alpha(\cdot)$. Let $\mathcal{X}_t = \{\mathbf{x}_i\}_{i=1}^t$ and $\mathcal{Y}_t = \{\mathbf{y}_i\}_{i=1}^t$ denote the evaluated parameters $\mathbf{x}_i$ and their objective vectors $\mathbf{y}_i$. At each iteration, the surrogate model approximates the posterior distribution of $\mathbf{y}$, and the acquisition function $\alpha(\mathbf{x})$ prioritizes candidate points. The pseudocode is shown in Algorithm 1.

Initialization and Acquisition Function. The framework initializes with random sampling to ensure spatial coverage of the parameter space. Subsequent iterations employ Expected Hypervolume Improvement (EHVI) [33] as the acquisition function to maximize hypervolume gains on the Pareto frontier. For a candidate $\mathbf{x}$, EHVI quantifies the expected improvement over the current Pareto frontier $\mathcal{Y}^*$:

$$\text{EHVI}(\mathbf{x}) = \mathbb{E}\left[\max\left(0, HV(\mathcal{Y}^* \cup \mathbf{y}(\mathbf{x})) - HV(\mathcal{Y}^*)\right)\right], \quad (4)$$

where $HV(\cdot)$ is the simplified notion for $HV(\cdot, \mathbf{y}_{ref})$, and the expectation integrates over the surrogate model's predictive distribution.

Surrogate Model. To avoid incompatibility or high computational complexity in Bayesian optimization, Probabilistic Random Forest (PRF) [34] serves as the surrogate model, extending standard random forests by outputting Gaussian distributions for each objective. For an ensemble of B regression trees, PRF predicts the mean $\mu(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B}\mu_b(\mathbf{x})$ and variance $\sigma^2(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B}(\mu_b(\mathbf{x}) - \mu(\mathbf{x}))^2$ for each objective. This probabilistic formulation enables uncertainty-aware EHVI computation, crucial for balancing exploration-exploitation trade-offs.

EDA toolchain. The toolchain integrates three stages: RTL Generator synthesizes parameterized hardware descriptions, Synthesis Tool maps RTL to gate-level netlists, and Place & Route Tool generates physical layouts and reports the PPA $\mathbf{y}$ for BO. Cell data—extracted from the final netlist—includes the timing of critical paths and power/area of each cell, forming the database for inter-loop analysis. This closed-loop system automates parameter-to-PPA translation, enabling system-aware technology optimization.

C. Inter-Loop Direction

This section presents the coordination mechanism that synergizes system loop and technology loop. As shown in Fig. 3, Orthrus analyzes the post-routing netlists alongside corresponding system-level PPA metrics to guide technology optimization. The resulting

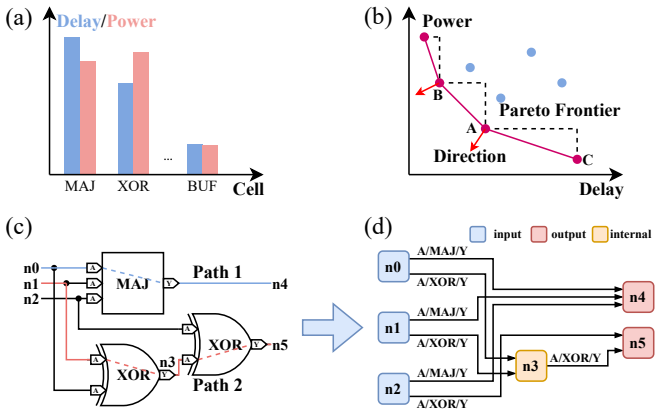


Fig. 5. Inter-loop analysis. (a) Per-cell delay/power contributions. (b) PPA directions on Pareto frontier. (c) Per-cell critial timing path. (d) Standard cell netlists modeled with net-centric directed acyclic graph (DAG).

inter-loop direction consists of three key components: (1) the PPA contribution of each standard cell type; (2) the occurrence frequency of cell combinations; (3) the optimization direction for specific system-level parameter configurations.

In the following subsections, we will introduce the details of each type of inter-loop direction.

Cell Contribution Analysis. As shown in Fig. 5(a), we quantify the contribution of each cell to system performance and power consumption, which enables prioritized optimization on critical cells. Our study focuses on power and timing impact, considering that cell area remains unchanged after process parameter tuning and cell layout exploration.

The power contribution of standard cell c can be derived from its aggregate power consumption divided by the total system power. Formally, given a post-routing standard cell netlist $\mathcal{G}$, the power contribution of c is calculated by:

$$w_c^{power} = \frac{\sum_{g \in \mathcal{G}} power(g) \cdot \mathbf{1}[Type(g) = c]}{power(\mathcal{G})} \tag{5}$$

The timing contribution of standard cell c is determined through path-based analysis, where cells appearing more frequently on critical timing paths are considered to have a greater impact on system-level timing performance. The rationale stems from the observation that such cells are either essential components of timing-critical functional modules or favored by synthesis tools to mitigate timing violations. In either case, optimizing these standard cells can effectively enhance overall performance. To quantify the timing contribution of cell type c, we first compute the timing contribution of each individual cell instance g (Equation (6)), then derive the aggregated contribution for c by averaging the score across all corresponding instances (Equation (7)).

$$w_g^{delay} = \exp(\lambda \cdot \max\{delay(p) \mid g \in p, p \in \mathcal{P}\}) \tag{6}$$

$$w_c^{delay} = \frac{\sum_{g \in \mathcal{G}} w_g^{delay} \cdot \mathbf{1}[Type(g) = c]}{\sum_{g \in \mathcal{G}} w_g^{delay}} \tag{7}$$

In Equation (6), a timing path p is a signal route across the cell netlist, as illustrated in Fig. 5(c). The set of all timing paths is denoted as $\mathcal{P}$. The timing path delays are obtained using the static timing analysis engine within Innovus. Conceptually, the contribution of standard cell instance g diminishes exponentially with larger quantity and greater criticality of competing timing paths. The hyperparameter λ is introduced to modulate the weighting mechanism's sensitivity to competitive effects.

Algorithm 2: Frequent Subcircuit Mining

Input : Netlist $G = (V_G, E_G)$, max depth d_{max}, max output number o_{max}, max input number i_{max}
Output: Subgraph occurence count $M : \Sigma^* \mapsto \mathbb{N}$

$M \leftarrow \{s \mapsto 0 \mid \forall s \in \Sigma^*\}$;
forall $V_H \subseteq V_G, |V_H| \leq o_{max}$ **do**
  /* Explore subcircuits using DFS. */
  $V_P \leftarrow \{v \in V_G \mid \exists u \in H, v$ can reach u in $G\}$;
  $P \leftarrow G[V_P]$ // Induced subgraph
  $\mathcal{S}_H \leftarrow \texttt{DFS}(P, d_{max}, i_{max})$;
  /* Count subcircuit patterns. */
  forall $S \in \mathcal{S}_H$ **do**
    $s \leftarrow \texttt{CanonicalRepr}(S)$;
    $M[s] \leftarrow M[s] + 1$;
return M;

Subcircuit Analysis. In Orthrus, we propose to synthesize multi-row standard cells for frequently occurring cell combinations, aiming to improve system-level PPA. A customized subgraph isomorphism detection algorithm is employed to identify these common cell combinations.

Before diving into algorithmic details, we first describe the graph construction process. As illustrated in Fig. 5(d), we employ a net-centric representation to model the standard cell netlists. Specifically, a combinatorial standard cell netlist can be represented as a directed acyclic graph (DAG). The vertices correspond to inter-cell nets and are categorized into three types (`Input/Output/Internal`). Edges represent intra-cell connections and are annotated with the cell type and associated I/O pins. For netlists that contain sequential elements (e.g. registers and latches), we partition the design into multiple combinatorial subcircuits and apply the graph matching algorithm independently.

Algorithm 2 outlines the frequent subcircuit mining process. In a nutshell, for netlist $G = (V_G, E_G)$ represented with net-centric DAG, the algorithm systematically explores all connected subcircuits within a bounded size and records the occurrence frequency of subcircuit patterns. To ensure tractability, we impose the constraints that candidate subcircuit $S \subseteq G$ must have input net count $\texttt{NumIn}(S) \leq i_{max}$, output net count $\texttt{NumOut}(S) \leq o_{max}$, and logic depth $\texttt{Depth}(S) \leq d_{max}$. In practice, we set $i_{max} = 4, o_{max} = 2, d_{max} = 3$, respectively. Each traversed subcircuit S is hashed into a unique key using an established colored DAG hashing method [35]. We record the frequency of its corresponding subcircuit pattern via bucket counting.

PPA Direction Analysis. As illustrated in Fig. 5(b), the direction of technology optimization is derived from the geometric properties of the Pareto frontier identified in the system loop. Given the computational overhead of iteratively evaluating the technology toolchain, we formulate a single-objective optimization for the technology loop by weighting the PPA metrics. Specifically, for each Pareto-optimal point, we calculate the normal vector to the local Pareto frontier using Singular Value Decomposition (SVD) on its k-nearest neighbors $\mathcal{N}_k$. This vector defines the trade-off sensitivity between delay and power, expressed as $\mathbf{v}_2^\top = [-W_{delay}, -W_{power}]$, the last row of $V^\top$:

$$V^\top = \begin{bmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \end{bmatrix}, \quad \mathcal{N}_k = U \Sigma V^\top. \tag{8}$$

Additionally, we flip the direction if it is far from the origin. As shown

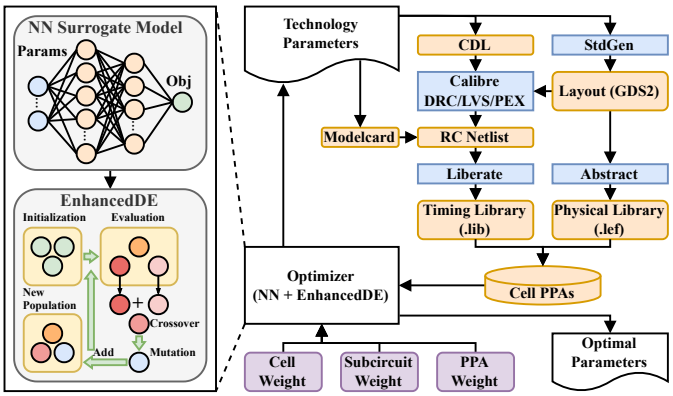


Fig. 6. Technology loop diagram, utilizing an EDA toolchain to extract PPA metrics for each standard cell, accepting weight inputs, employing neural network as a surrogate model, and using EnhancedDE as the optimizer to output the optimal parameters.

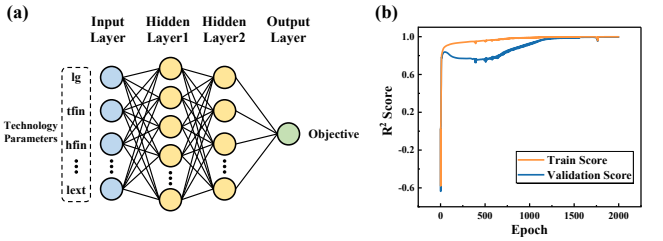


Fig. 7. (a) Schematic diagram of the neural network surrogate model; (b) R^2 scores for the training and validation sets as a function of epochs during the neural network training process.

in Fig. 5(c), with this direction, the optimization balances power and delay at the balanced point A, while pushing the delay to the limit at the low-delay point B.

D. Technology Loop

The proposed technology loop framework, as illustrated in Fig. 6, aims to optimize technology parameters for system-level performance. The framework consists of three primary steps: `CellSimulate`, `PPADirected`, and `Optimizer`. These steps enable efficient simulation and optimization of technology parameters for system-level design.

The `CellSimulate` function takes as input the parameters at the Technology level from TABLE I and outputs the PPA of each standard cell. The simulation process can be broken down into the following stages:

- *Parameter Adjustment*: The input parameters are used to adjust the circuit netlist, model card, and `StdGen` configuration. This ensures that the simulation aligns with the target technology parameters.
- *Layout Generation*: The `StdGen` function takes the configuration parameters and generates the corresponding standard cell layouts. The generated layouts are then validated through DRC, LVS, and PEX to verify the correctness of the layout and extract the parasitic netlists.
- *Library Generation*: Using the validated layouts, the physical library (.lef) is created. Additionally, the extracted parasitic netlists, along with the model card, are used to generate the timing library (.lib), which contains the necessary delay and power characterizations for each standard cell. This results in the power, delay, and area information for each standard cell, which corresponds to the output of the `CellSimulate` function.

Algorithm 3: Neural Network-Assisted EnhancedDE for Technology Optimization.

Input : Initial samples N, NN model training epochs E, maximum iteration for enhancedDE I_{max}, DE generation n_{gen}, population size s_{pop}, top size s_{top}, mutation factor MF, penalty factor PF, penalty threshold PT, crossover probability CR, historical data $\mathbf{X}_{his}$, $\mathbf{C}_{his}$

Output: Optimal parameters $\mathbf{x}^*$

/* Phase 1: Initial Sampling */
$\mathbf{X}_{init} \leftarrow \texttt{LHS}(N)$ // Latin Hypercube Sampling
$\mathbf{C}_{init} \leftarrow \texttt{CellSimulate}(\mathcal{P}_{init})$ // standard cell simulation
$(\mathbf{Y}_{init}, \mathbf{Y}_{his}) \leftarrow \texttt{PPACalculation}(\mathbf{C}_{init}, \mathbf{C}_{his})$;
$\mathbf{D} \leftarrow (\mathbf{X}_{init}, \mathbf{C}_{init}, \mathbf{Y}_{init})$ // Initial dataset
$\mathbf{D} \leftarrow (\mathbf{D}.\mathbf{X} \cup \mathbf{X}_{his}, \mathbf{D}.\mathbf{C} \cup \mathbf{C}_{his}, \mathbf{D}.\mathbf{Y} \cup \mathbf{Y}_{his})$;
for $i = 1 \rightarrow I_{max}$ **do**
  /* Phase 2: Surrogate Model Construction */
  $\texttt{NNSurrogate}(\cdot) \leftarrow \texttt{TrainMLP}(\mathbf{D}, \text{epochs} = E)$;
  /* Phase 3: EnhancedDE Optimization */
  $\mathbf{P} \leftarrow \texttt{InitPopulation}(s_{pop}, \mathbf{D}.\mathbf{X})$;
  $\mathbf{T} \leftarrow \emptyset$ // Elite solution archive
  for $j = 1 \rightarrow n_{gen}$ **do**
    forall $x \in P$ **do**
      if $\texttt{rand}() < 0.2$ *and* $|\mathbf{T}| < 2$ **then**
        $a \leftarrow \texttt{RandomSelect}(\mathbf{P} \setminus \{x\})$;
        $b, c \leftarrow \texttt{RandomSelect}(\mathbf{T}, 2)$;
      else
        $a, b, c \leftarrow \texttt{RandomSelect}(\mathbf{P} \setminus \{x\}, 3)$;
      $m \leftarrow a + MF \times (b - c)$ // mutant
      $t \leftarrow \texttt{CrossOver}(x, m, CR)$ // trial
      $f_{base} \leftarrow \texttt{NNSurrogate}(t)$;
      $d_{min} \leftarrow \texttt{MinDistance}(t, \mathbf{T} \cup \mathbf{P})$;
      $f_{penalized} \leftarrow f_{base} + PF \times \max(0, PT - d_{min})$;
      if $f_{penalized} < \texttt{fitness}(x)$ **then**
        $\mathbf{P}.\texttt{update}(x, t, f_{penalized})$;
      $\mathbf{T} \leftarrow \texttt{UpdateElites}(\mathbf{T} \cup \{t\}, PT)$
  /* Phase 4: Design Verification & Update */
  $\mathbf{X}_{candidates} \leftarrow \texttt{SelectDiverse}(\mathbf{T}, s_{top})$;
  $\mathbf{C}_{true} \leftarrow \texttt{CellSimulate}(\mathbf{X}_{candidates})$;
  $\mathbf{Y}_{true} \leftarrow \texttt{PPACalculation}(\mathbf{C}_{true})$;
  $\mathbf{D} \leftarrow (\mathbf{D}.\mathbf{X} \cup \mathbf{X}_{candidates}, \mathbf{D}.\mathbf{C} \cup \mathbf{C}_{true}, \mathbf{D}.\mathbf{Y} \cup \mathbf{Y}_{true})$;
/* Phase 5: Final Parameter Extraction */
$k^* \leftarrow \texttt{argmin}(\mathbf{D}.\mathbf{Y})$;
$\mathbf{x}^* \leftarrow \mathbf{D}.\mathbf{X}[k^*]$;
return $\mathbf{x}^*$;

The `PPACalculation` function uses PPA weights along with the delay and power of each cell and returns the corresponding weighted PPA objective y for the technology loop. y is computed according to the formula in Equations (9)-(11). First, the normalized cell delay and power are weighted by their respective contribution (see cell contribution analysis in Section III-C). Normalization references original ASAP7 cells for those from the initial library, and the initial single-row version for the fused cells. Next, system-level direction weights aggregate delay and power metrics to evaluate the optimization objective (see PPA direction analysis in Section III-C).

$$Delay(\mathbf{C}) = \sum_{c \in \mathbf{C}} w_c^{delay} \times norm_{delay}(c) \tag{9}$$

$$Power(\mathbf{C}) = \sum_{c \in \mathbf{C}} w_c^{power} \times norm_{power}(c) \tag{10}$$

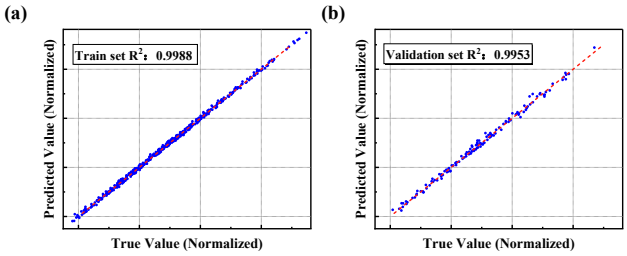


Fig. 8. Comparison between surrogate model predictions and ground truth values: (a) Training set performance showing strong agreement (R^2 = 99.88%), with data points closely distributed along the diagonal trend line; (b) Test set performance (20% holdout) demonstrating model generalizability, maintaining good correlation (R^2 = 99.53%). Dashed lines represent perfect prediction ($y = x$).

$$y = W_{delay} \times Delay(\mathbf{C}) + W_{power} \times Power(\mathbf{C}) \tag{11}$$

The `Optimizer` step, detailed in Algorithm 3, involves leveraging a neural network as a surrogate model to reduce reliance on the time-consuming `CellSimulate` function, while employing an Enhanced Differential Evolution (EnhancedDE) algorithm to optimize the technology parameters. The specific process is outlined as follows:

- *Initialization Sampling:* The initialization sampling employs Latin Hypercube Sampling (LHS) for broad parameter space coverage, with historical values from the previous technology loop integrated to expand the neural network's training dataset. This helps avoid the selection of optimal solutions that overlap with historical solutions, improving the diversity of the sampling process.
- *Neural Network Surrogate Model:* We employ a fully connected neural network (as shown in Fig. 7(a)) to predict the objective y based on normalized input parameters, as listed in TABLE I. The model's output is the predicted objective value $\hat{y}$. The loss function is defined as follows:

$$Loss = MSE(\hat{y} - y) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y)^2 \tag{12}$$

  To prevent overfitting, we use batch processing for the training set and incorporate L2 regularization. The training and testing dataset variations with respect to the number of epochs are shown in Fig. 7(b), and the validation results for the training and testing datasets are shown in Fig. 8. The R^2 value for the training dataset is 99.88%, for the testing dataset is 99.53%. The results demonstrate that the model effectively predicts the target values with a high degree of accuracy.
- *Differential Evolution Optimization:* The `Optimizer` uses an EnhancedDE algorithm, which builds on traditional differential evolution and adds a distance penalty to improve the diversity of the solution. While the neural network surrogate model accelerates the evaluation process, the EnhancedDE algorithm utilizes the population search strategy to perform extensive exploration of the parameter space, ultimately identifying the optimal set of solutions. To further ensure solution diversity, a distance penalty is introduced. This penalty evaluates the Euclidean distance between candidate solutions and existing solutions, checking if the distance exceeds a predefined penalty threshold (PT). If the candidate solution fails to meet the distance requirement, a penalty is applied, discouraging the selection of similar solutions and promoting diversity in the final batch of optimal solutions. Upon completion of one iteration, the optimal solutions returned are evaluated using the `CellSimulate` and `PPADirected` functions to obtain the true target value, $\mathbf{Y}_{true}$, which then updates the dataset $\mathbf{D}$. Once the iteration reaches the predefined I_{max}, the optimal parameter vector $\mathbf{x}^*$ corresponding to the best target value is returned.

The `Optimizer` is responsible for adjusting both technology parameters $\mathbf{p}_{tech}$ and cell-specific hyperparameters `num_rows`. Since the latter presents more complexity, we hereby make further elaborations on this process. Given a base standard cell library derived from the original ASAP7 library, Orthrus selects the N_{ext} most frequent subcircuit patterns for cell fusion (detailed in subcircuit analysis from Section III-C). The selected subcircuits are assigned to an initial cell row count `num_rows` = 1 and are incorporated into the library extension. Whereas the hyperparameter of the remaining subcircuit patterns is permanently assigned to 0, meaning that these subcircuits will not be fused and added to the library. As discussed in Section II-A, `num_rows` serves as a key hyperparameter for fused cells with numerous transistors, which balances area compactness and critical path length. In the subsequent invocation of technology loop, we adjust `num_rows` of the selected fused cells between 1 and 3 to explore this trade-off. For all standard cells adopted from the initial ASAP7 library, `num_rows` is fixed to 1 to reduce the complexity of the surrogate model.

IV. Evaluation

A. Setup

Platform. The automated STCO framework runs on a Linux-based platform with an Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz and 1536 GiB of memory. Cadence Genus 19.12-s121_1 and Cadence Innovus v21.14-s109_1 are used to synthesize, place, and route every sampled design. Cadence Liberate 19.2.1.215, Cadence Abstract 6.1.8, Cadence Spectre 18.1.0, and Mentor Calibre v2019.3_15.11 are used for characterizing the timing library, physical library, circuit simulation, layout verification, and parasitic extraction. The Bayesian Optimization in the system loop is implemented via `ParallelOptimizer` in OpenBox 0.8.4 [36] with default settings.

Hyperparameters. We set the sensitivity parameter $\lambda = 10$ in Equation (6) for computing cell timing contribution. We choose $k = 2$ neighbors in Equation (8) for finding the normal vector. The neural network surrogate model in the technology loop adopts an architecture with two hidden layers (16 and 8 neurons respectively) utilizing sigmoid activation functions. An initial learning rate of 0.02 is configured with the Adam optimizer for parameter updates. The training epoch E for the neural network is set to 1500 epochs to ensure convergence. In the optimization algorithm, the parameters are set as follows: PT = 0.1, PF = 1e3, MF = 0.8, CR = 0.9, s_{pop} = 100, n_{gen} = 20, s_{top} = 5, and I_{max} = 2.

Baseline. To demonstrate the efficacy of technology optimization, the baseline approach only adjusts system-level parameters $(\mathbf{p}_{arch}, \mathbf{p}_{ls}, \mathbf{p}_{pd})$. We use the default technology parameters from TABLE I and the basic standard cells from TABLE II.

B. Result Analysis

Pareto frontier and Hypervolume. We identify two key techniques for significant PPA improvement: (1) Standard cell recharacterization (Rechar), which adjusts technology parameters $\mathbf{p}_{tech}$ and `StdGen` hyperparameter `num_rows`; (2) Subcircuit fusion (Fusion), which merges common subcircuit patterns into new standard cells. We ablate their individual and combined contributions to expanding PPA Pareto frontiers, as demonstrated in Fig. 9 and TABLE III. Without Fusion, adjusting only $\mathbf{p}_{tech}$ achieves a 6.5% hypervolume improvement over the baseline. Without Rechar, fusing subcircuits into single-row standard cells yields a 7.7% hypervolume improvement over the

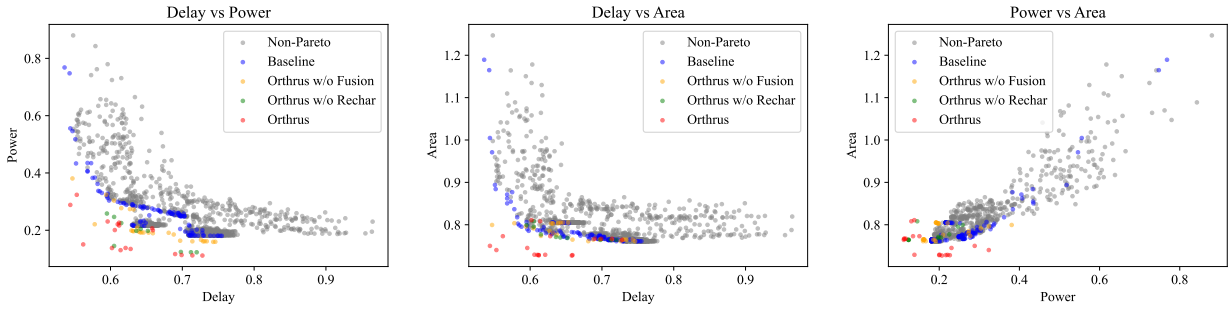
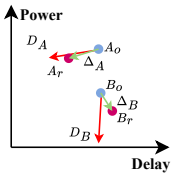


Fig. 9. Normalized Pareto frontier of baseline, Orthrus without subcircuit fusion, Orthrus without standard cell recharacterization, and Orthrus. The reference point for computing hypervolume is $(1, 1, 1)$.



Category	Power	Delay	phig_n	phig_p	hfin_nm	tfin_nm	lg_nm	lext_nm	lct_nm	num_rows	Cosine
A_o	0.313	0.622	4.307	4.8681	32	6.5	20	5	24	1	0.989
A_r	0.289	0.544	4.302	4.8683	36	7.1	17	6	25	3	
B_o	0.145	0.605	4.307	4.8681	32	6.5	20	5	24	1	0.822
B_r	0.131	0.614	4.312	4.8680	28	5.8	18	6	24	1	

Fig. 10. Directional alignment between the optimized parameter vector D and the actual optimization trajectory Δ (subfigure). The corresponding power and delay metrics for each category, along with the technology level parameters and cosine similarity (subtable).

TABLE III
HYPERVOLUME OF EACH METHOD

Method	Baseline	Orthrus w/o Fusion	Orthrus w/o Rechar	Orthrus
HV ($\times 10^{-2}$)	8.055	8.582	8.679	10.727
	-	+6.5%	+7.7%	+33.2%

baseline. When these techniques are combined, we observe significant reductions in delay and power along with moderate area savings, resulting in a substantial hypervolume improvement of 33.2%. We further measured the optimization results of individual metrics while maintaining others constant (allowing a tolerance of 1e-3). Due to the observed power-area correlation ($r = 0.88$), we focus on delay-power tradeoffs: achieving 61.4% power savings at iso-delay and 12.5% delay reduction at iso-power conditions.

Effectiveness of Inter-Loop Direction. To validate the effectiveness of inter-loop direction, we quantify the cosine similarity between the optimization direction and the actual Rechar path. As depicted in Fig. 11, which plots the *sorted* cosine similarity across all optimization points, the vast majority of values are positive (clustering near or reaching 1.0). This strong alignment confirms that the optimization path adheres closely to the inter-loop direction. Additionally, we evaluated the optimization results using naive cell weighting (i.e., treating all cells as equally important). The final hypervolume of 9.443×10^{-2} represents a 12.0% reduction compared to the results of Orthrus. This outcome demonstrates the effectiveness of our prioritized cell weighting approach during optimization.

Subcircuit Fusion. Statistical analysis based on the methodology introduced in Section III-C reveals that Full Adders (FAs) and Half Adders (HAs) account for the majority of the delay (53.1%), power (65.2%), and area (75.7%) overhead. Consequently, we specifically optimize these two subcircuits through fusion techniques.

Case study. To further investigate the effectiveness of our proposed method, we present two optimization examples shown in Fig. 10. As seen, the optimized direction D closely aligns with the actual optimization path Δ. From the table in Fig. 10, it can be observed that the primary objective for the A parameter combination is to optimize timing. The corresponding parameter set adjusts the work function to reduce the threshold voltage, increases the drive current

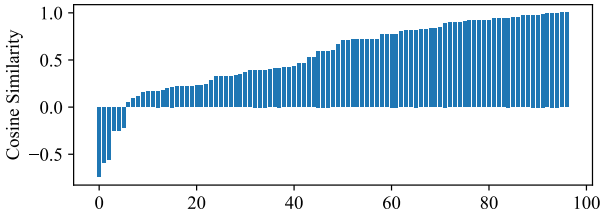


Fig. 11. The sorted cosine similarity between the PPA optimization direction and the actual recharacterization path.

by modifying `hfin`, `tfin`, and `lg`, and enhances both intra-cell and inter-cell routability by increasing `num_rows`. These results align with physical expectations. Additionally, from the B parameter combination, it is clear that the main objective is to reduce power. This leads to an opposite trend compared to the A combination, where the work function is adjusted to increase the threshold voltage, reduce the drive current, and reduce `num_rows` to minimize parasitic capacitance, thereby decreasing dynamic power. This analysis further validates the effectiveness of our proposed method.

V. CONCLUSION

This paper introduces Orthrus, a dual-loop automated framework for system-technology co-optimization (STCO). Orthrus combines system-level and technology-level optimizations through an interloop coordination mechanism, bridging the gap between system requirements and technology innovations while optimizing both levels simultaneously. Evaluated on 7nm technology, Orthrus achieves 12.5% delay reduction at iso-power and 61.4% power savings at iso-delay compared to baseline approaches, complemented by a 33.2% PPA hypervolume improvement that redefines Pareto optimality for cross-layer design. Overall, Orthrus offers a promising solution to the challenges of scaling in the VLSI industry, providing a comprehensive and efficient methodology for STCO that can adapt to evolving technological demands. In the future, we aim to expand Orthrus to support a broader range of architectures and process technologies, further enhancing its versatility and impact in optimizing future VLSI designs.

REFERENCES

[1] D. Biswas, J. Myers, S. B. Samavedam, and J. Ryckaert, "Stco: driving the more than moore era," in *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2024, pp. 7–8.

[2] C. Bai, Q. Sun, J. Zhai, Y. Ma, B. Yu, and M. D. Wong, "Boom-explorer: Risc-v boom microarchitecture design space exploration framework," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[3] C. Bai, J. Huang, X. Wei, Y. Ma, S. Li, H. Zheng, B. Yu, and Y. Xie, "Archexplorer: Microarchitecture exploration via bottleneck analysis," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 268–282.

[4] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, 2015, pp. 161–170.

[5] Q. Xiao, S. Zheng, B. Wu, P. Xu, X. Qian, and Y. Liang, "Hasco: Towards agile hardware and software co-design for tensor computation," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 1055–1068.

[6] J. Wang, L. Guo, and J. Cong, "Autosa: A polyhedral compiler for high-performance systolic arrays on fpga," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2021, pp. 93–104.

[7] L. Jia, Z. Luo, L. Lu, and Y. Liang, "Tensorlib: A spatial accelerator generation framework for tensor algebra," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 865–870.

[8] H. Geng, Q. Xu, T.-Y. Ho, and B. Yu, "Ppatuner: Pareto-driven tool parameter auto-tuning in physical design via gaussian process transfer learning," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1237–1242.

[9] Z. Xie, G.-Q. Fang, Y.-H. Huang, H. Ren, Y. Zhang, B. Khailany, S.-Y. Fang, J. Hu, Y. Chen, and E. C. Barboza, "Fist: A feature-importance sampling and tree-based method for automatic design flow parameter tuning," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 19–25.

[10] R. Liang, J. Jung, H. Xiang, L. Reddy, A. Lvov, J. Hu, and G.-J. Nam, "Flowtuner: A multi-stage eda flow tuner exploiting parameter knowledge transfer," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[11] C. Gilardi, G. Zeevi, S. Choi, S.-K. Su, T. Y. Hung, S. Li, N. Safron, Q. Lin, T. Srimani, M. Passlack, G. Pitner, E. Chen, I. Radu, H.-S. P. Wong, and S. Mitra, "Barrier booster for remote extension doping and its dtco for 1d & 2d fets," in *2023 International Electron Devices Meeting (IEDM)*, 2023, pp. 1–4.

[12] S. Kim, S.-J. Min, S.-G. Jung, and H.-Y. Yu, "Multi-objective optimization and inverse design of complementary field-effect transistor using combined approach of machine learning and non-dominated sorting genetic algorithms for next-generation semiconductor devices," *Engineering Applications of Artificial Intelligence*, vol. 137, p. 109064, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197624012223

[13] H. Zhang, Y. Jing, and P. Zhou, "Machine learning-based device modeling and performance optimization for finfets," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 4, pp. 1585–1589, 2023.

[14] U. Kwon, T. Okagaki, Y.-s. Song, S. Kim, Y. Kim, M. Kim, A.-y. Kim, S. Ahn, J. Shin, Y. Park, J. Kim, D. S. Kim, W. Qi, Y. Lu, N. Xu, H.-H. Park, J. Wang, and W. Choi, "Intelligent dtco (idtco) for next generation logic path-finding," in *2018 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, 2018, pp. 49–52.

[15] X. Wang, R. Kumar, S. B. Prakash, P. Zheng, T.-H. Wu, Q. Shi, M. Nabors, S. C. Gadigatla, S. Realov, C.-H. Chen, Y. Zhang, M. Kaizad, A. Yeoh, I. Post, C. Auth, and A. Madhavan, "Design-technology co-optimization of standard cell libraries on intel 10nm process," in *2018 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2018, pp. 28–2.

[16] Z. Chen, C. Guo, Z. Song, G. Feng, S. Wang, L. Zhang, X. Yin, Z. Wu, Z. Yan, and C. Zhuo, "Boosting standard cell library characterization with machine learning," in *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, 2025, pp. 385–391.

[17] T. Ma, Z. Deng, X. Sun, and L. Shao, "Fast cell library characterization for design technology co-optimization based on graph neural networks," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024, pp. 472–477.

[18] H. Cho, H. Seo, S. Chung, K.-M. Choi, and T. Kim, "Standard cell layout generator amenable to design technology co-optimization in advanced process nodes," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–6.

[19] S. Choi, J. Jung, A. B. Kahng, M. Kim, C.-H. Park, B. Pramanik, and D. Yoon, "Probe3. 0: a systematic framework for design-technology pathfinding with improved design enablement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 4, pp. 1218–1231, 2023.

[20] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, "Cross-layer optimization for high speed adders: A pareto driven machine learning approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 12, pp. 2298–2311, 2018.

[21] Y.-F. Liu, C.-Y. Hsieh, and S.-Y. Kuo, "Boomerang: Physical-aware design space exploration framework on risc-v sonicboom microarchitecture," in *2023 IEEE 34th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2023, pp. 85–93.

[22] Y. Ren, C. Xue, J. Zhang, C. Zhang, Q. Xu, Y. Lin, L. Zhang, and G. Sun, "Diffuse: Cross-layer design space exploration of dnn accelerator via diffusion-driven optimization," *arXiv preprint arXiv:2503.23945*, 2025.

[23] M. Liu, "1.1 unleashing the future of innovation," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 9–16.

[24] K. Zhang, "1.1 semiconductor industry: Present & future," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67, 2024, pp. 10–15.

[25] T. Liang, J. Chen, L. Li, and W. Zhang, "Autocelllibx: Automated standard cell library extension based on pattern mining," *arXiv preprint arXiv:2207.12314*, 2022.

[26] R. Fu, C. Wang, B. Yu, and T.-Y. Ho, "Temacle: A technology mapping-aware area-efficient standard cell library extension framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

[27] J. Yan, X.-C. Yin, W. Lin, C. Deng, H. Zha, and X. Yang, "A short survey of recent advances in graph matching," in *Proceedings of the 2016 ACM on international conference on multimedia retrieval*, 2016, pp. 167–174.

[28] Q. He and Y. Li, "An efficient circuit matching algorithm based on hash extraction of features," in *2024 2nd International Symposium of Electronics Design Automation (ISEDA)*. IEEE, 2024, pp. 222–228.

[29] K. Guo and Y. Lin, "Multi-row standard cell layout synthesis with enhanced scalability," in *2025 3nd International Symposium of Electronics Design Automation (ISEDA)*, 2025, pp. 1–6.

[30] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[31] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002626921630026X

[32] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[33] S. Daulton, M. Balandat, and E. Bakshy, "Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[34] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, ser. LION'05. Berlin, Heidelberg: Springer-Verlag, 2011, p. 507–523. [Online]. Available: https://doi.org/10.1007/978-3-642-25566-3_40

[35] C. Helbling, "Directed graph hashing," *arXiv preprint arXiv:2002.06653*, 2020.

[36] Y. Li, Y. Shen, W. Zhang, Y. Chen, H. Jiang, M. Liu, J. Jiang, J. Gao, W. Wu, Z. Yang, C. Zhang, and B. Cui, "Openbox: A generalized black-box optimization service," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 3209–3219.