

AutoPMS: A Framework for Automated Power Management System Design via Hierarchical Modeling and Embedded Design Knowledge

Bin Ye and Shuo Li*

Frontier Institute of Chip and System, Fudan University, Shanghai, China

College of Integrated Circuits and Micro-Nano Electronics, Fudan University, Shanghai, China

*Corresponding Author: shuoli@fudan.edu.cn

Abstract—Design automation of analog and mixed-signal (AMS) systems is becoming increasingly critical and challenging as both design complexity and scale continue to grow. Power management systems (PMS), as essential AMS components of modern electronics, must satisfy stringent power delivery specifications, which significantly increase design risks and prolong iteration cycles. Prior work on power management design automation has primarily focused on specific topologies or individual blocks, falling short of meeting full system requirements. To bridge this gap, we propose AutoPMS, an automated framework for large-scale PMS generation. The end-to-end framework automatically generates PMS netlists from user-defined system specifications. AutoPMS employs a hierarchical modeling approach, integrating design-knowledge-based models at both the system and block levels to accelerate the automation process and ensure correctness. In addition, we propose a design-coefficient feedback scheme to enhance modeling accuracy and a weight self-adjusting mechanism to improve the overall success rate of the design flow. To the best of our knowledge, AutoPMS is the first framework to automate multi-output PMS design across a wide dynamic power range, from nanowatts to several watts. This framework offers a practical and scalable solution for generating diverse PMS designs, enabling the agile deployment of various electronic systems.

Index Terms—Analog design automation, Power management system, Hierarchical modeling, Embedded design knowledge

I. INTRODUCTION

Power management systems (PMSs) are essential analog and mixed-signal (AMS) components in modern electronic systems, providing robust power delivery to meet the voltage supply requirements. With the increasing integration and complexity of integrated circuits, designers must simultaneously consider multiple specifications across various power management blocks, imposing substantial manual effort, time costs, and design risks. Thus, PMS design automation has become both practically valuable and an important research focus.

Recent advances in AMS design automation have been driven by methods such as Bayesian optimization (BO) [1]–[4] and reinforcement learning (RL) [5]–[10]. These techniques have shown notable success in automating operational amplifiers and analog-to-digital converters, which typically involve only tens of optimization variables. In contrast, PMSs for chiplets and IoT systems-on-chip (SoCs) often comprise multiple power management units (PMUs), each tailored to the requirements of its power rail, as shown in Figure 1. This results in hundreds of variables, making conventional methods impractical due to

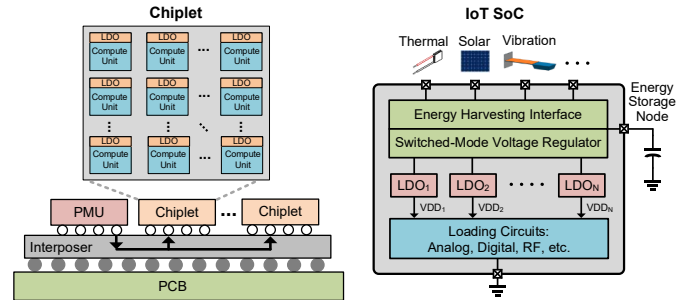


Fig. 1: Multi-output power management systems for chiplets and IoT SoCs.

dimensionality challenges and high computational costs. As a result, approaches effective for small-scale AMS circuits cannot be directly applied to PMS design automation.

Existing work on PMS automation often reduces design dimensionality by optimizing only the power stage or part of the control circuitry while keeping other parameters fixed [11]–[14]. Moreover, most studies focus on a single PMU type, such as buck converters or low-dropout regulators (LDOs), neglecting system-level optimization, and thus fail to address the demands of large-scale PMSs [14]–[16]. Such partial optimization can yield sub-optimal results since modern SoCs integrate PMSs with diverse topologies and multiple specifications.

Driven by system-level design trends, PMS automation must meet several requirements: (1) an end-to-end flow from high-level specifications to complete netlists, (2) flexibility beyond fixed structures, supporting topology exploration, block selection, and parameter optimization, (3) fast optimization for agile, iterative design, and (4) reliability to ensure functional correctness and deployment-ready performance. To address these challenges, we propose an automated design framework that generates PMS netlists from user-defined system specifications. The main contributions are as follows:

- **Framework Development:** We present AutoPMS, an end-to-end framework for PMS automation that generates optimized netlists directly from user-defined system specifications. A scalable power-management block library ensures high performance across diverse power scenarios and facilitates the integration of new blocks spanning a wide dynamic range, from nanowatts to several watts.
- **Hierarchical Modeling:** The hierarchical modeling approach supports the optimization of large-scale systems

by decomposing the problem into the system level and the block level. Experimental results demonstrate that our hierarchical methodology achieves significantly faster runtime in solving high-dimensional design-space optimization problems than conventional non-hierarchical methods.

- **Weight Self-Adjusting and Coefficient Feedback:** We enhance the automation flow by introducing a weight self-adjusting mechanism and a design-coefficient feedback scheme, both of which substantially improve the flow’s success rate and the accuracy of the optimization results.
- **Experimental Validation:** We conduct comprehensive validations of AutoPMS, demonstrating its large-scale automation capability compared to existing approaches. Experimental results show that hierarchical modeling significantly improves optimization speed. We further present experimental results of the feedback techniques used in the flow, highlighting their contributions to overall optimization performance.

II. RELATED WORK

Significant progress has been made in AMS design automation. Although most prior work does not directly target PMSs, these methods provide valuable insights for PMS automation. In this section, we review techniques relevant to our framework.

A. Circuit Modeling Approach

Simulation-based modeling uses simulation netlists, treating the netlist parameters as inputs and the simulation results as outputs. It formulates optimization as a black-box problem, with recent studies frequently using machine learning to accelerate iterations. This approach has shown effectiveness for small-scale circuits (about 20–30 variables) [1]–[4], [17]–[23], yet scales poorly for larger designs (50+ variables) due to exploding runtime and extensive computation. Knowledge-based modeling, in contrast, derives models from circuit principles by embedding designer expertise. It offers higher computational efficiency and adaptability to topology changes, but requires significant manual effort and depends on the quality of expertise. Knowledge-based methods are well-suited for large-scale AMS systems, including PMSs, where automation efficiency and system-level optimization are critical [8], [24], [25].

B. Hierarchical Decomposition Methodology

The broad design space of PMSs makes full-system circuit optimization computationally prohibitive, leading to low automation efficiency. Some studies mitigate this by optimizing only local portions of the system, but this limits overall system quality. By decomposing the system into functionally independent blocks, a high-dimensional problem becomes multiple lower-dimensional ones. Prior study [5] in AMS design automation has shown that modularization can reduce computational growth from exponential to nearly linear. The study [26] also proposed a hierarchical estimation method to alleviate the complexity of analyzing large-scale circuit systems, but it did not integrate the hierarchical modeling technique into design optimization. This hierarchical and modular approach makes complex PMS automation more tractable and scalable.

TABLE I: COMPARISON BETWEEN AUTOPMS AND OTHER AMS DESIGN AUTOMATION METHODOLOGIES

Method	AutoPMS	BO-Based	RL-Based
Principle	Hierarchical Modeling, Analog Block Library, Design Knowledge	Surrogate-based Optimization	Reward-driven Learning
Modeling	Knowledge-based	Simulation-based	Simulation-based
Runtime	Fast	Medium	Slow
Flexibility	High	Medium	Medium
Interpretability	High	Low	Medium
Advantage	High efficiency and strong scalability	Small-scale, efficient, and easily integrated	Adaptive and progressive
Primary Cost	Library Construction, Modeling	Circuit Simulation	Circuit Simulation, Reward Design
Scalability	Large-scale (e.g. SoCs)	Small-scale (< 30 variables)	Compute-Resource Dependent

C. Analog Block Library

The construction of reusable block libraries has long been a common strategy in the EDA domain to enhance design efficiency and scalability, such as standard cell libraries in digital design. Inspired by this concept, many studies have developed analog block libraries to support the automation of analog circuit design [19], [27]–[29]. These flows, offering high flexibility and scalability, have also been applied to PMS design automation research. However, existing power management block libraries typically offer only a few types of PMUs [15], [30], which are inadequate to meet the power delivery requirements of modern SoCs. Consequently, a key challenge in PMS design automation is the development of a comprehensive block library that accommodates a wide power range while ensuring high performance.

D. AMS Design Automation Methodology

Table I provides a comparison between our proposed AutoPMS and mainstream AMS automation methodologies. Bayesian optimization and reinforcement learning rely on simulation-based modeling, where automation efficiency is limited by circuit scale and simulation costs. In contrast, AutoPMS combines hierarchical modeling, an analog block library, and embedded design knowledge, achieving fast runtime, strong scalability, and high interpretability, leading to high automation efficiency. These advantages make AutoPMS particularly well-suited for the automated design of large-scale PMSs.

III. PMU LIBRARY AND HIERARCHICAL MODELING

This section introduces our PMU library and modeling methodology. The proposed library covers widely used PMU blocks, supporting a broad power range and diverse power delivery requirements. Each block includes a circuit schematic and two knowledge-based models for system-level and block-level optimization.

A. PMU Library Development

The PMU library incorporates diverse circuit topologies and control schemes to enhance the flexibility of AutoPMS. Figure 2 illustrates several representative block schematics in the library. Each block contains essential components to function as a standalone PMU, such as a buck converter with its power stage, comparator, control loop, gate driver, and oscillator. The

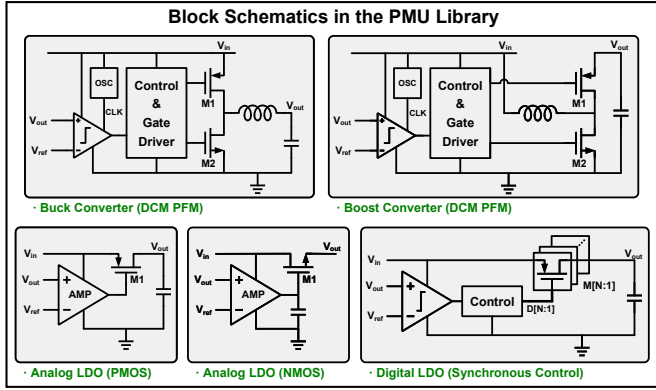


Fig. 2: Representative block schematics in the PMU library.

library covers mainstream DC–DC converter topologies, including buck, boost, and buck–boost, and supports control schemes including discontinuous conduction mode (DCM), continuous conduction mode (CCM), pulse frequency modulation (PFM), and pulse width modulation (PWM). In addition, it provides LDOs with either analog or digital control. The library is highly scalable, allowing new blocks to be added with a one-time integration effort. By incorporating new topologies and models into the PMU library, AutoPMS can directly utilize these blocks within the design flow. This extensible and modular architecture ensures adaptability to emerging power management technologies and portability across process nodes.

B. Hierarchical Modeling Methodology

System Modeling: The purpose of system modeling is to allocate user-defined PMS specifications to individual blocks, reducing the dimensionality of the variable space in the optimization problem. Table II illustrates the modeling methodology using the DCM PFM Buck block as an example. In the system modeling, the variables are the specifications of the buck block, enabling system performance optimization by allocating block specifications. The system modeling equations include design coefficients K , whose values relate to internal design parameters. To improve modeling accuracy, the design coefficient feedback mechanism is introduced as detailed in later sections. To ensure feasible system-level optimization, each block specification is limited by empirical bounds, which define the decision space for optimization. For example, the efficiency of a buck converter typically does not exceed 95%. Furthermore, multiple blocks are connected according to topologies in PMS design. This introduces additional constraints in system-level optimization to match at the interfaces between two blocks. These new constraints are integrated into the system-topology templates and invoked during the system-level optimization process to complete the optimization problem.

Block Modeling: Block modeling is used to optimize block performance with transistor-level design parameters to meet the allocated block specifications. As shown in Table II, these parameters are treated as optimization variables, while the performance specifications of the blocks serve as optimization objectives. To account for process-dependent effects, technology coefficients are introduced to improve modeling accuracy.

TABLE II: HIERARCHICAL MODELS OF A BUCK CONVERTER

	Model Equations
Equations in the System Model	$\text{Efficiency} = \frac{P_{\text{out}}}{P_{\text{out}} + P_{\text{sw}} + P_{\text{cond}} + P_{\text{static}}}$ $P_{\text{cond}} = K_{\text{Pcond}} I_{\text{out}} I_{L_peak}$ $P_{\text{sw}} = K_{\text{Psw}} \frac{I_{\text{out}}}{V_{\text{ripple}}}$ $V_{\text{ripple}} = K_{\text{Vripple}} \cdot \frac{I_{L_peak}^2}{V_{\text{in}} - V_{\text{out}}}$
Block Specifications	$V_{\text{in}}, V_{\text{out}}, P_{\text{static}}, V_{\text{ripple}}, I_{L_peak}, f_{\text{sw}}$
Equations in the Block Model	$P_{\text{cond}} = 1/3 f_{\text{sw}} I_{L_peak}^2 (R_1 T_{\text{source}} + R_2 T_{\text{load}})$ $f_{\text{sw}} = \frac{I_{\text{out}}}{V_{\text{ripple}} C_{\text{out}}}$ $I_{L_peak} = \frac{V_{\text{in}} - V_{\text{out}}}{R_1} (1 - e^{-\frac{R_1 T_{\text{source}}}{L_{\text{ind}}}})$
Design Parameters	$W_1, L_1, W_2, L_2, L_{\text{ind}}, R_{\text{ESR}_L}, C_{\text{out}}, R_{\text{ESR}_C}$

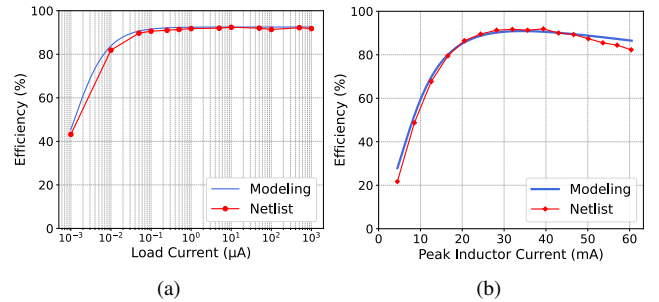


Fig. 3: Block modeling accuracy against spice netlist: (a) efficiency vs. load current; (b) efficiency vs. peak inductor current.

Block modeling guides detailed circuit design, with its accuracy directly affecting the deviation between modeled and actual results. Empirical bounds are also defined for each design parameter to ensure physical realizability. Figure 3 compares the modeling and simulation results for the efficiency-related characteristic curves of the buck converter. The model achieves an efficiency accuracy exceeding 95% across the tested operating points, measured using mean squared error (MSE).

IV. PROPOSED AUTOPMS DESIGN AUTOMATION FLOW

A. AutoPMS Flow Overview

Figure 4 illustrates the proposed AutoPMS design flow, which begins with user-defined system specifications. Before the flow starts, the framework imports the PMU library and technology coefficients based on the selected technology node (65nm CMOS in this work). At this stage, user-defined preference weights are also read in, enabling customization of optimization priorities.

Phase I: AutoPMS performs a preliminary analysis of the system specifications (e.g., boost vs. buck, number of outputs) and generates system topology candidates using the predefined system topology templates. The score of each candidate topology, E_{topo} , is given by:

$$E_{\text{topo}} = \sum_{i=1}^n w_i \frac{|P_{\text{topo}}^i - S_{\text{user}}^i|}{S_{\text{user}}^i} \quad (1)$$

s.t. $C_i(P_{\text{topo}}^i, S_{\text{user}}^i) < 0,$
 $\forall i \in \{1 \dots n\}.$

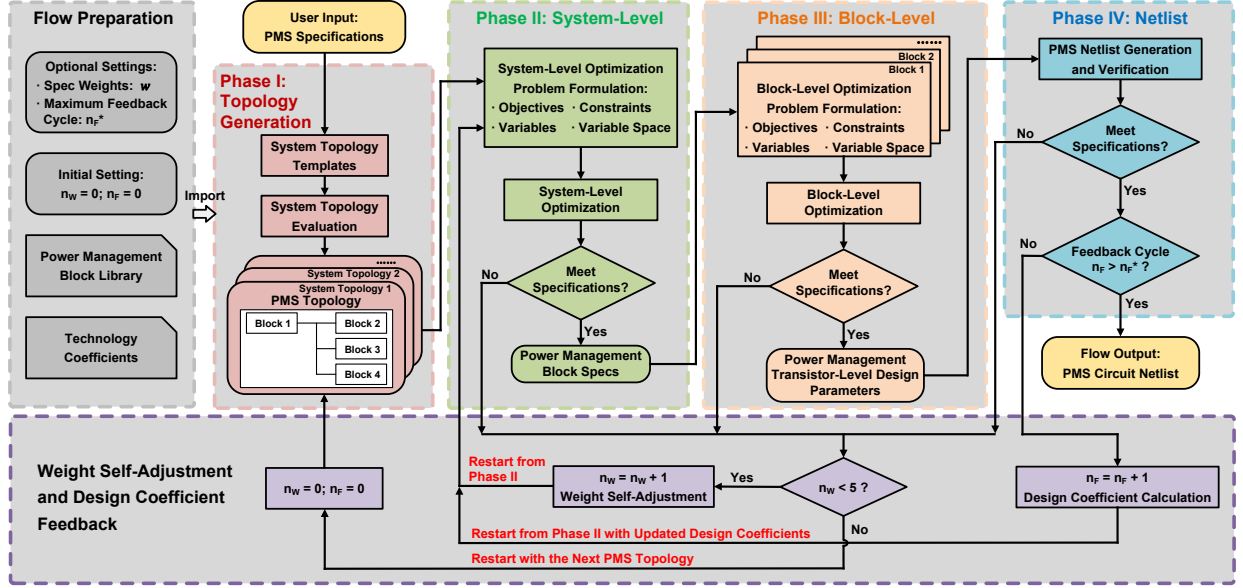


Fig. 4: The proposed AutoPMS design automation flow.

where P_{topo}^i , S_{user}^i , w_i , and C_i represent the i -th expertise-based performance value of a given topology, the i -th user-specified system specification, the associated specification weight, and the specification constraint, respectively. E_{topo} measures how well a given system topology satisfies the user's input specifications. All system topology candidates are ranked by this score, with the top design selected for optimization and the others retained as alternatives.

Phase II: System-level optimization is performed based on the system topology selected in Phase I. The framework then constructs the system-level optimization problem by invoking the system-level models of all involved blocks, as given by:

$$\begin{aligned}
 & \text{maximize } E_{\text{sys}} = \sum_{i=1}^n w_i f_{\text{sys}}^i(\mathbf{x}) \\
 & \text{s.t. } C_i(\mathbf{x}, S_{\text{user}}^i) < 0, G_j(\mathbf{x}) < 0, \\
 & \quad \mathbf{S}_{\text{blk_min}} \leq \mathbf{x} \leq \mathbf{S}_{\text{blk_max}}, \\
 & \quad \forall i \in \{1 \dots n\}, j \in \{1 \dots m\}.
 \end{aligned} \quad (2)$$

where E_{sys} , f_{sys}^i , \mathbf{x} , C_i , and G_j represent the score after system-level optimization, the normalized performance corresponding to the i -th system specification, the values of optimization variables (block specifications), system specification constraints, and physics-related constraints, respectively. System modeling is optimized using heuristic algorithms. NSGA-II is adopted as the default optimization algorithm in AutoPMS because it effectively balances convergence speed and search capability. System-level optimization allocates user-defined specifications to individual blocks, decomposing the high-dimensional problem into lower-dimensional subproblems.

Phase III: AutoPMS performs detailed circuit design automation for each block in parallel. Block modeling from the PMU library is utilized with block specifications allocated from the system-level optimization in Phase II. NSGA-II is again employed to explore the block-level design space and identify optimal solutions. The resulting design variables are

then mapped onto predefined netlist templates for complete circuit implementations.

Phase IV: The generated PMS design is then validated in Phase IV. AutoPMS produces the complete system netlist by combining the system topology with the block designs obtained in earlier phases. These netlists are simulated using a SPICE simulator to verify performance against the user-defined specifications. Before generating the final output netlist, several feedback cycles are executed through the design-coefficient feedback mechanism to improve modeling accuracy.

B. Design Coefficient Feedback

The accuracy of the system modeling strongly depends on the design coefficients shown in Table II, which capture internal circuit parameters such as transistor sizes, capacitances, and inductances. Before the automation flow begins, the framework relies on default design coefficients provided by designers, inevitably introducing modeling inaccuracies. To mitigate this issue, Phase IV incorporates a feedback mechanism that updates system-level coefficients once block-level designs are available. As illustrated in Figure 4, SPICE simulations evaluate PMS performance, and the resulting data are used to update the coefficients in system models. These updated coefficients are fed back into Phase II, yielding refined models with higher accuracy for the next automation cycle. This feedback loop enhances modeling fidelity, accelerates convergence, and calibrates the optimization process to actual circuit behavior, thereby improving both design quality and efficiency.

C. Weight Self-Adjustment

The user-defined weights (ranging from 0 to 1 with a default value of 0.5) encode preferences for specific specifications and guide their relative importance in optimization. To maximize the likelihood of meeting system input requirements, the framework integrates an internal weight self-adjustment mechanism.

Algorithm 1 Weight Self-Adjustment

Require: User-defined specification weights $\mathbf{w} = \{w_1, w_2, \dots, w_n\}$ and specifications $\mathbf{S}^* = \{S_1^*, S_2^*, \dots, S_n^*\}$ (system specifications in Phase II and IV and block specifications in Phase III)

- 1: Initialize \mathbf{w}
- 2: Validate specifications \mathbf{S} in Phase II, III, or IV
- 3: **for** $i = 1, 2, \dots, n$ **do**
- 4: **if** specification S_i fails to meet requirements S_i^* **then**
- 5: $w_i \leftarrow w_i + 0.1$
- 6: **end if**
- 7: **end for**
- 8: Restart the design flow at Phase II with the updated weights \mathbf{w}

When optimization results violate constraints, AutoPMS identifies the unsatisfied specifications and increases their weights (by 0.1 in this work), as illustrated in Figure 4. The flow then returns to Phase II, prioritizing the violated specifications and steering the search toward feasible solutions. To avoid excessive iterations on an unsuitable topology, AutoPMS employs a self-adjustment counter. If more than five adjustments fail to yield a feasible design, the current topology candidate is discarded, and the flow proceeds with the next option in the ranking list. This weight self-adjustment mechanism adapts optimization without manual intervention, enhances design-space exploration, and increases the likelihood of satisfying user-defined specifications.

V. EVALUATION

The proposed framework is implemented and evaluated based on the TSMC 65nm CMOS technology. All simulations and optimizations are carried out on a workstation with an Intel® Core™ i7-10875H CPU with 2.3 GHz base frequency and 32 GB memory.

A. Functionality and Runtime Evaluation

This section evaluates AutoPMS using an example of automatic PMS design. To reflect practical design requirements, all specification weights are set to 0.5, the feedback cycle limit is set to 5, and a total of 21 system specifications are provided as input. The generated system topology comprises two power stages and delivers three distinct output voltage rails: a buck converter in the first stage and three LDOs in the second stage, including two PMOS ALDOs and one NMOS ALDO. The breakdown of static power consumption is 44% for the buck converter, 32% and 19% for the PMOS LDOs, and 5% for the NMOS LDO, indicating a balanced static power distribution. The system achieves a peak efficiency of 74.6% and a minimum efficiency of 60.2%, meeting all system efficiency specifications.

With the generated system topology, the PMS contains 91 design variables across four blocks. Hierarchical modeling decomposes the original high-dimensional problem into system-level optimization (26 variables) and block-level optimization (25, 22, 22, and 22 variables). This decomposition significantly reduces computational cost by mitigating the exponential growth associated with dimensionality. Table III summarizes the runtime distribution across different phases within a single AutoPMS cycle. Phase IV netlist verification is the primary bottleneck due to the intrinsic cost of circuit-level simulations.

TABLE III: RUNTIME ANALYSIS OF ONE AUTOMATION CYCLE

	Function	Runtime (One Cycle)	
		Hierarchical Modeling	Non-Hierarchical Modeling
Phase I	Flow Preparation	< 30 s	< 30 s
	Topology Generation	~ 1 min	~ 1 min
Phase II	System-Level Optimization	~ 3 mins	~ 32 mins
Phase III	Block-Level Optimization	~ 5 mins	
Phase IV	Netlist Verification	~ 20 mins	~ 20 mins
	Design Coefficients Calculation	< 10 s	-
Total Runtime		~ 31 mins	~ 53 mins

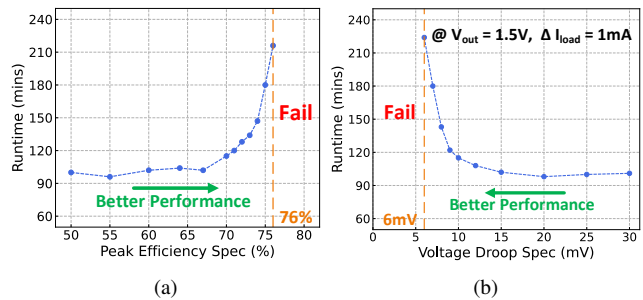


Fig. 5: Design-space exploration for: (a) peak efficiency; (b) voltage droop.

In contrast, the non-hierarchical modeling approach, which optimizes the PMS directly using block models, results in substantially higher runtime overhead.

We further analyze the design-space exploration capability of AutoPMS. Figure 5 shows the exploration results for system efficiency and transient response performance, with all other input specifications held constant. As the requirements on peak efficiency or transient response voltage droop approach their limits, the runtime of the design flow increases sharply, reflecting the boundaries of the feasible design space. Once the specifications exceed practical bounds, the design flow fails. These results demonstrate that the proposed AutoPMS framework can effectively explore the circuit design space and identify optimal solutions.

B. Performance Evaluation

Hierarchical modeling plays a central role in improving computational efficiency in AutoPMS. As illustrated in Figure 6(a), this approach substantially reduces both simulation overhead and computational cost, with the benefits becoming more pronounced as system complexity increases. For a PMS comprising seven blocks, hierarchical modeling achieves a 4.8× design efficiency improvement compared with the non-hierarchical modeling approach using block models. Figure 6(b) illustrates the effect of the weight self-adjustment mechanism by comparing the success rates of automated PMS design with and without this mechanism. For a fixed system structure, 50 sets of system specifications were randomly generated as inputs to the design flow, and the success rate of the automated design was evaluated. As the complexity of the PMS increases and the number of blocks grows, the weight self-adjustment mechanism significantly improves the likelihood of generating designs that satisfy all user requirements. When the system contains 5

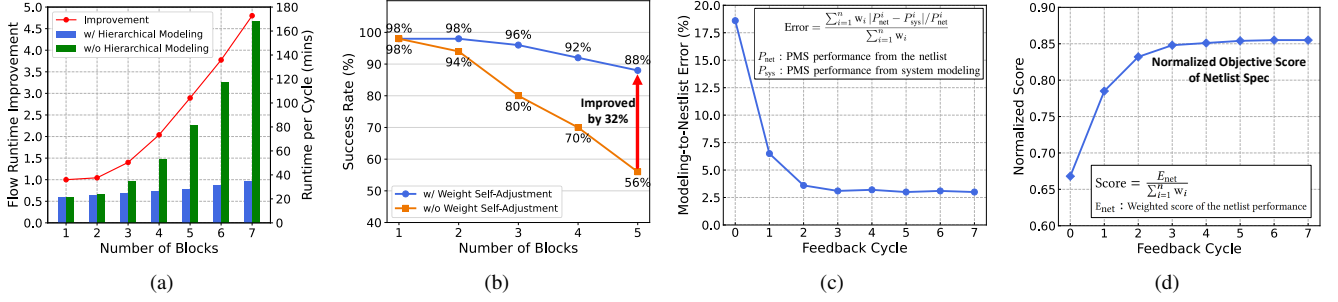


Fig. 6: Performance evaluation of the proposed PMS automation flow: (a) hierarchical modeling; (b) weight self-adjustment; (c) system modeling error vs. design coefficient feedback cycle; (d) normalized score vs. design coefficient feedback cycle.

TABLE IV: COMPARISON OF THIS WORK WITH RELATED POWER MANAGEMENT DESIGN AUTOMATION WORKS

	[16] ISLPED '2019	[14] CICC '2020	[25] ISCAS '2023	[31] ICCAD '2023	AutoPMS
Technology	65nm, 130nm	65nm	65nm, 130nm	130nm	65nm
Methodology	Design-knowledge-based modeling + Toolchain integration	Design-knowledge-based modeling + Toolchain integration	Auxiliary cell library + Toolchain integration	Reinforcement Learning + Graph Neural Network	Design-knowledge-based hierarchical modeling + PMU library
Topologies	Buck	Buck	LDO	LDO	Buck, Boost, Buck-Boost, LDO
Control Circuit	Yes	Yes	Yes	Yes	Yes
Control Scheme	DC-DC: PWM+CCM	DC-DC: PWM+CCM	LDO: Synchronous Digital	LDO: Analog	DC-DC: PWM+CCM, PFM+DCM; LDO: Analog, Synchronous/Asynchronous Digital
Power Range	180mW ~ 1.5W [†] 8.33 ×	18mW ~ 45mW [†] 2.5 ×	300μW ~ 32.5mW 108.33 ×	18μW ~ 18mW 1000 ×	8nW ~ 4.8W 600000 ×
Multi-Output Support	No	No	No	No	Yes
Peak Efficiency	76%	79.3%	94.8% (current efficiency)	N/R	74.6%
Transient Response V_{droop}	15mV* @ $I_{\text{load}} = 55\text{mA}$, $V_{\text{out}} = 0.75\text{V}$.	70mV* @ $I_{\text{load}} = 30\text{mA}$, $V_{\text{out}} = 0.75\text{V}$.	N/R	200mV @ $I_{\text{load}} = 10\text{mA}$, $V_{\text{out}} = 1.8\text{V}$.	74.4mV @ $I_{\text{load}} = 10\text{mA}$, $V_{\text{out}} = 1.8\text{V}$.
Settling Time	133ns @ $I_{\text{load}} = 55\text{mA}$, $V_{\text{out}} = 0.75\text{V}$.	200ns @ $I_{\text{load}} = 30\text{mA}$, $V_{\text{out}} = 0.75\text{V}$.	2.5μs @ $I_{\text{load}} = 25\text{mA}$, $V_{\text{out}} = 1.25\text{V}$.	N/R	560ns @ $I_{\text{load}} = 10\text{mA}$, $V_{\text{out}} = 1.8\text{V}$.

N/R = Not Reported, N/A = Not Applicable; [†]Originate from the generated design; *Calculated from the paper

blocks, the weight self-adjustment mechanism increases the design success rate by 32%.

We further analyze the impact of the design-coefficient feedback on the final optimization results of the flow. The feedback mechanism reduces the discrepancy between the model and the actual netlist, thereby enhancing the overall optimization performance. Figure 6(c) and (d) illustrate the evolution of the optimization process for a PMS composed of three blocks (21 input specifications) by disabling the weight self-adjustment to ensure that the optimization scores across successive cycles remain comparable. The results show the reduction of the model-netlist discrepancy and corresponding improvement in the optimization objective. As the number of cycles increases, the optimization objective gradually approaches a steady value. By improving model accuracy and accelerating convergence, the design-coefficient feedback enhances the quality of the final design solution. For more complex PMS structures with multiple output rails, additional feedback cycles are typically required. Users can adjust the number of cycles to balance design quality and computational cost.

C. Comprehensive Comparison

Table IV summarizes the comparison between the proposed AutoPMS framework and prior automated PMU design works. Unlike existing methods, which are limited to single topologies and narrow power ranges (typically from 10^2 to 10^3), AutoPMS

supports buck, boost, buck-boost, and LDO topologies with multiple control schemes, extending the power range from 8nW to 4.8W. This achieves a dynamic range of 6×10^5 , which is $10^2 \times$ broader than state-of-the-art works. To the best of our knowledge, AutoPMS is the first framework to enable multi-output PMS design automation. In terms of efficiency and transient performance, AutoPMS delivers results competitive with existing works while offering much greater flexibility for diverse application scenarios.

VI. CONCLUSION

This paper presents AutoPMS, an end-to-end automation framework for the design of large-scale PMSs aimed at reducing the manual cost and improving design efficiency. AutoPMS achieves system-level optimization by combining hierarchical modeling with a PMU library and automatically generates system netlists that satisfy user-defined system specifications. We further propose weight self-adjustment and design-coefficient feedback mechanisms to enhance the robustness and modeling accuracy of the framework. Experimental validation in TSMC 65nm technology confirms the effectiveness and competitiveness of AutoPMS. This work addresses a critical gap in the automated design of large-scale, multi-output PMSs and contributes to more practical design automation for power delivery in modern electronic systems.

REFERENCES

- [1] Wenlong Lyu, Pan Xue, Fan Yang, Changhao Yan, Zhiliang Hong, Xuan Zeng, and Dian Zhou. An efficient bayesian optimization approach for automated optimization of analog circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(6):1954–1967, 2018.
- [2] Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3306–3314. PMLR, 10–15 Jul 2018.
- [3] Jialin Lu, Liangbo Lei, Fan Yang, Changhao Yan, and Xuan Zeng. Automated compensation scheme design for operational amplifier via bayesian optimization. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 517–522, 2021.
- [4] Wei W Xing, Weijian Fan, Zhuohua Liu, Yuan Yao, and Yuanqi Hu. Kato: Knowledge alignment and transfer for transistor sizing of different design and technology. In *Proceedings of the 61st ACM/IEEE Design Automation Conference, DAC '24*, New York, NY, USA, 2024. Association for Computing Machinery.
- [5] Jinxin Zhang, Jiarui Bao, Zhangcheng Huang, Xuan Zeng, and Ye Lu. Automated design of complex analog circuits with multiagent based reinforcement learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2023.
- [6] Hanrui Wang et al. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.
- [7] Keertana Settaluri, Zhaokai Liu, Rishubh Khurana, Arash Mirhaj, Rajeev Jain, and Borivoje Nikolic. Automated design of analog circuits using reinforcement learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(9):2794–2807, 2022.
- [8] Somayaji N.S. Karthik, Hanbin Hu, and Peng Li. Prioritized reinforcement learning for analog circuit optimization with design knowledge. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1231–1236, 2021.
- [9] Zhenxin Zhao and Lihong Zhang. Analog integrated circuit topology synthesis with deep reinforcement learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(12):5138–5151, 2022.
- [10] Supriyo Maji, Ahmet F. Budak, Souradip Poddar, and David Z. Pan. Toward end-to-end analog design automation with ml and data-driven approaches. In *Proceedings of the 29th Asia and South Pacific Design Automation Conference, ASPDAC '24*, page 657–664. IEEE Press, 2024.
- [11] Shaoze Fan, Ningyuan Cao, Shun Zhang, Jing Li, Xiaoxiao Guo, and Xin Zhang. From specification to topology: Automatic power converter design via reinforcement learning. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9, 2021.
- [12] Chen-Chia Chang et al. Lamagic: language-model-based topology generation for analog integrated circuits. In *Proceedings of the 41st International Conference on Machine Learning, ICML'24*. JMLR.org, 2024.
- [13] Van-Hai Bui et al. Deep neural network-based surrogate model for optimal component sizing of power converters using deep reinforcement learning. *IEEE Access*, 10:78702–78712, 2022.
- [14] Venkata Chaitanya Krishna Chekuri, Nael Mizanur Rahman, Edward Lee, Arvind Signh, and Saibal Mukhopadhyay. A fully synthesized integrated buck regulator with auto-generated gds-ii in 65nm cmos process. In *2020 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, 2020.
- [15] Yaswanth Kumar Cherivirala and David Wentzloff. A technology-agnostic method for digital ldo synthesis and layout automation. In *Proceedings of the 29th ACM/IEEE International Symposium on Low Power Electronics and Design, ISLPED '24*, page 1–6, New York, NY, USA, 2024. Association for Computing Machinery.
- [16] Venkata Chaitanya Krishna Chekuri et al. Automatic gdsii generator for on-chip voltage regulator for easy integration in digital socs. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6, 2019.
- [17] Jialin Lu, Liangbo Lei, Jiangli Huang, Fan Yang, Li Shang, and Xuan Zeng. Automatic op-amp generation from specification to layout. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(12):4378–4390, 2023.
- [18] Zhongkai Wang et al. An automated and process-portable generator for phase-locked loop. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 511–516, 2021.
- [19] Ming Ding et al. A circuit-design-driven tool with a hybrid automation approach for sar adcs in iot. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 672–675, 2018.
- [20] Biying Xu, Shaolan Li, Nan Sun, and David Z. Pan. A scaling compatible, synthesis friendly vco-based delta-sigma adc design and synthesis methodology. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2017.
- [21] Keertana Settaluri, Ameer Haj-Ali, Qijing Huang, Kourosh Hakhamaneshi, and Borivoje Nikolic. Autockt: Deep reinforcement learning of analog circuit designs. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 490–495, 2020.
- [22] Pedro Toledo et al. A 300mv-supply, sub-nw-power digital-based operational transconductance amplifier. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(9):3073–3077, 2021.
- [23] Kourosh Hakhamaneshi, Nick Werblun, Pieter Abbeel, and Vladimir Stojanović. Late breaking results: Analog circuit generator based on deep neural network enhanced combinatorial optimization. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–2, 2019.
- [24] Ashish Kumar Singh, Kareem Ragab, Mario Lok, Constantine Caramanis, and Michael Orshansky. Predictable equation-based analog optimization based on explicit capture of modeling error statistics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(10):1485–1498, 2012.
- [25] Yaswanth K. Cherivirala, Mehdi Saligane, and David D. Wentzloff. An open source compatible framework to fully autonomous digital ldo generation. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2023.
- [26] Engin Deniz and Günhan Dündar. Hierarchical performance estimation of analog blocks using pareto fronts. In *6th Conference on Ph.D. Research in Microelectronics & Electronics*, pages 1–4, 2010.
- [27] J. Crossley et al. Bag: A designer-oriented integrated framework for the development of ams circuit generators. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 74–81, 2013.
- [28] Eric Chang, Jaeduk Han, Woorham Bae, Zhongkai Wang, Nathan Narevsky, Borivoje Nikolic, and Elad Alon. Bag2: A process-portable framework for generator-based ams circuit design. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–8, 2018.
- [29] Juzheng Liu et al. From specification to silicon: Towards analog/mixed-signal design automation using surrogate nn models with transfer learning. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9, 2021.
- [30] Shourya Gupta, Shuo Li, and Benton H. Calhoun. Scalable all-analog ldos with reduced input offset variability using digital synthesis flow in 65-nm cmos. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 32(1):190–194, 2024.
- [31] Zonghao Li and Anthony Chan Carusone. Design and optimization of low-dropout voltage regulator using relational graph neural network and reinforcement learning in open-source sky130 process. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 01–09, 2023.