OpenSAR: An Open Source Automated End-to-end SAR ADC Compiler

Mingjie Liu, Xiyuan Tang, Keren Zhu, Hao Chen, Nan Sun, and David Z. Pan
ECE Department, The University of Texas at Austin, Austin, TX, USA
{jay_liu, xitang, keren.zhu, haoc}@utexas.edu, nansun@mail.utexas.edu, dpan@ece.utexas.edu

***Abstract*—Despite recent developments in automated analog sizing and analog layout generation, there is doubt whether analog design automation techniques could scale to system-level designs. On the other hand, analog designs are considered major roadblocks for open source hardware with limited available design automation tools. In this work, we present OpenSAR, the first open source automated end-to-end successive approximation register (SAR) analog-to-digital converter (ADC) compiler. OpenSAR only requires system performance specifications as the minimal input and outputs DRC and LVS clean layouts. Compared with prior work, we leverage automated placement and routing to generate analog building blocks, removing the need to design layout templates or libraries. We optimize the redundant non-binary capacitor digital-to-analog converter (CDAC) array design for yield considerations with a template-based layout generator that interleaves capacitor rows and columns to reduce process gradient mismatch. Post layout simulations demonstrate that the generated prototype designs achieve state-of-the-art resolution, speed, and energy efficiency.**

I. Introduction

Analog-to-digital converters (ADCs) are integral building blocks of modern system-on-chips (SoCs) since they are the interface between analog front-end sensory to back-end digital signal processing. Successive approximation register (SAR) ADCs, is an ADC type that has gained increasing attention due to its great versatility and range from ultralow-power to ultrahigh-speed designs [1]. The increasing market demand for Internet of Things (IoT) devices [2] has made SAR ADCs crucial design components because of their moderate resolution and high energy efficiency.

However, analog designs still heavily rely on manual efforts. Traditionally, designers have to select system architecture, circuit topology, size devices, and layout engineers to draw device placement and wire routing. This high customization and manual involvement in analog design has greatly limited the turn-around-time and design scale, making it challenging to keep up with the pace of expanding IoT market demands. The highly automated digital design counterparts have scaled to billions of transistor count. Thus new analog design methodologies and development in analog automation tools are desired to speed up the current manual design flows.

Despite such challenges, there has been a significant amount of research in analog design automation to speed up the current manual design flows. Advancements in machine learning and black-box optimization techniques, such as Bayesian optimization [3] and reinforcement learning [4], have enabled efficient automated transistor sizing with high fidelity transistor-level simulations which characterize circuit performance. Template-based procedural layout automation tools, such as Berkeley Analog Generator (BAG) [5], have enabled easy layout adaptation in technology migration and some flexibility in circuit sizing. Optimization-based layout generation, such as MAGICAL [6] and ALIGN [7], focus on fully automated layout generation, where the device placement is determined by numerically minimizing the wirelength [8] and automatically routed through algorithms such as maze routing [9]. Although there has been increasing attention in academia for fully automated analog design solutions, it is still challenging to scale fully automated transistor sizing and layout generation methods to system-level designs, and generate high-quality and reliable solutions.

On the other hand, open source hardware is gaining momentum in the design ecosystem [10], where hardware designs can be freely used, altered, or distributed. The recent introduction of open source process design kits (PDKs) [11] has opened opportunities for circuit designers to explore open source hardware solutions without considering non-disclosure agreement (NDA) restrictions. OpenRAM [12] proposed a end-to-end open source memory compiler. The OpenRoad [13] project has further enhanced researchers to develop synthesizable digital-friendly designs on traditional analog designs such as OpenSerDes [14]. FaSOCs [15] have been shown to generate various analog circuit solutions, including PLLs, LDOs, and temperature sensors, by leveraging digital automatic place and route (APR) tools. However, SAR ADCs still require analog building blocks such as sample and hold (S&H) circuits and comparators. Thus it is difficult for SAR ADCs to be adapted to fully synthesized digital solutions while still achieving acceptable performance and design flexibility.

In this work, we present an open source automated end-to-end SAR ADC compiler. The entire framework requires top-level design specifications and generates a SAR ADC layout design as output. We leverage recent developments in automated transistor sizing and optimization-based layout automation to remove human-in-the-loop, significantly reducing the design effort. Prior work in automating SAR ADC designs [16], [17] rely on template- layout generation requires manual effort in layout template design, greatly restricting the design flexibility and offering little to no automation in device sizing. In OpenSAR, the front-end sizing and back-end layout generation process of placement and routing are fully automated. Our contributions are summarized as follows:

- We present a fully automated end-to-end SAR ADC compiler that requires only top-level specifications as the minimal input.
- Our framework is an integral of recent analog automation developments from device sizing, placement, and routing, which removes manual effort in both front-end and back-end design of analog building blocks.
- We develop a template-based non-binary capacitor digital-to-analog converter (CDAC) array layout generator to ensure its layout quality.
- Our SAR ADC is based on a redundant non-binary search algorithm, where the bit redundancy design problem is formally formulated and optimized for yield.
- Post layout simulations of prototype designs demonstrate state-of-the-art resolution, speed, and energy efficiency.
- To the best of our knowledge, this work is the first **open source**[1] framework for generating SAR ADC designs and layouts from

[1] https://github.com/magical-eda/OpenSAR

top-level specifications.

II. Synthesized ADCs and SAR ADCs

In this section, we first review prior work on synthesized ADCs in Sec. II-A where the back-end design is realized with digital automated place and route (APR) tools. We then give a basic introduction on SAR ADCs with an emphasis on the prior attempts of design automation in Sec. II-B.

A. Synthesized ADCs

Synthesizable ADCs typically only consist of standard digital logic cells, described entirely in Verilog code, allowing the back-end process to be fully automated by digital APR tools. Stochastic flash ADCs [18] introduces a comparator based on only digital NAND gates. It removes the resistor ladder in traditional flash ADCs and relies on the random offset of comparators in a large array. However, this typically results in low power efficiency and resolution. The work of [19] proposes a time-mode circuit employing a VCO-based multi-bit quantifier with first-order noise-shaping. The design is then described in Verilog, and the resulting layout is implemented with digital APR tools. Similarly, the work of [20] presents a fully synthesized VCO-based $\Delta\Sigma$ modulator from digital standard cells and a few resistors. Only one-time modifications on the standard cell library are needed for adding resistor custom cells. Although synthesizable ADCs can fully automate the back-end layout implementation, the design procedure still involves careful manual effort, and designs only using standard logic severely limits the design flexibility.

B. SAR ADC Design Automation

SAR ADC has become the architecture of choice for medium-speed/medium-resolution requirements, where it managed to achieve the best efficiency amongst other ADC architectures [21]. It has a straightforward implementation with the topology shown in Fig. 1. The analog input V_{in} is sampled with arrays of capacitors also implements a charge-redistribution feedback DAC. The comparator with the logic and DAC performs a binary search algorithm to approximate the sampled input in N steps, where N is the final resolution of digital output code D_{out}.

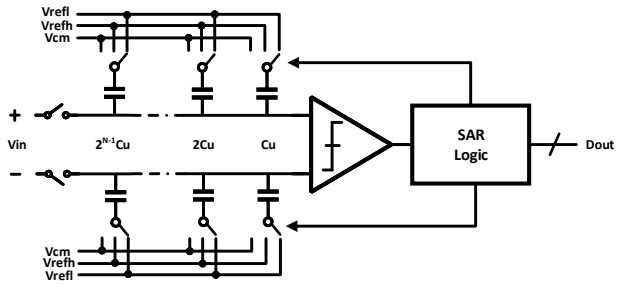


Fig. 1: SAR ADC architecture.

Several works have proposed various methods to automate the design process. Seo et al. [22] propose a code-reusable design methodology for synthesizable SAR ADCs. The authors replace traditional custom-designed analog building blocks with synthesizable digital circuits, including bootstrapped switch and comparator. The design requires a power gating multi-threshold technology and manual design of the capacitor standard cell. Huang et al. [23] proposed a systematic design methodology for designing SAR ADCs, where the sizing procedure for analog components is automated first by equation-based methods and then finetuned with spice simulations. The framework does not include automatic layout generation. Wulff et al. [16] presents a low-power compiled SAR design in 28-nm FDSOI technology. It implements a template-based layout method, using a limited set of circuit blocks with restrictions on the selection of transistor sizing, and needs the manual definition of routing solutions. Ding et al. [17] propose a hybrid design automation tool for SAR ADCs. The automatic sizing procedure is a hybrid approach based on the equation-based method, and simulation-based finetuning. The top-level routing requires additional manual scripting. Preparing the layout templates and the comparator lookup library also requires a significant amount of manual labor and setup overhead.

III. Analog Design Automation

Analog design usually consists of two stages: front-end and back-end design. The front-end design stage consists of architecture, circuit topology design, and device sizing. The back-end design stage mainly focuses on layout implementation, where devices are placed and routed to achieve DRC and LVS clean solutions. By convention, both the front-end and back-end design stages are manual and often involve iterative optimizations since layout parasitics and effects are often difficult to predict in the early front-end design stage.

In this section, we introduce recent developments in analog automation, with automatic device sizing in Sec. III-A and layout automation in Sec. III-B. Specifically, we emphasize Bayesian optimization [3] and MAGICAL [6] that are the main components of the OpenSAR framework.

A. Automatic Device Sizing

Numerous methods have previously been proposed for automated transistor sizing. Early works relied on symbolic AC models [24] and equation-based method [25], where the analog circuit needs to be modeled first. Recent works focus on directly using results from high fidelity performance spice simulations and leverage black-box gradient-free optimization methods, including differential evolution [26], Bayesian optimization [3], and reinforcement learning [4].

Bayesian optimization is a sequential model-based optimization algorithm. A probabilistic surrogate model (usually Gaussian process regression GPR) is updated with the new simulated result during each optimization iteration. Then the next sample point to query the black box is decided by optimizing the acquisition function, a loss function that describes the objective optimality while balancing exploration and exploitation. We adopt the constrained single objective formulation for automated transistor sizing problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & c_i(x) \leq b_i,\ i = 1, \ldots, m, \end{aligned} \tag{1}$$

where x is the d-dimensional design variable, $f_0(x)$ is the selected cost to minimize (typically power), and $c_i(x)$ are m performance metrics to meet the target of b_i. In our context, during Bayesian optimization, the GPR model predicts a mean μ_x and variance σ_x^2 for any input x. The next sample data is selected by maximizing the weighted expected improvement, a product of expected improvement (EI) and probability of feasibility (PoF):

$$EI(x) = E_{f_0 \sim N(\mu_0, \sigma_0)}[max(f^* - f_0, 0)] \tag{2}$$

$$PoF(x) = \prod_{i=1}^{m} Pr(c_i(x) < 0) \tag{3}$$

Bayesian optimization has received increasing attention due to its high sample efficiency and flexibility to cover the vast need of designers, such as extensions to multi-objective [27], and constrained

optimization [3]. Several works have leveraged additional techniques to improve scalability [28], parallelism [29] and layout parasitic considerations [30] for analog sizing. In OpenSAR, we integrate BoTorch [31], an open source library for Bayesian optimization, as the optimization kernel for automatic sizing.

B. Analog Layout Automation

In this section, we introduce the recent advancements in analog layout automation. We first introduce template-based layout generation in Sec. III-B1 and optimization based layout generation in Sec. III-B2. We specifically emphasize MAGICAL since it is the core layout engine used for analog circuit components in OpenSAR. In OpenSAR, we leverage the best of both methods. The CDAC array (Sec. IV-B) is generated with a template-based method to ensure layout quality with yield considerations. The S&H circuit and comparator (Sec. IV-D) are generated with MAGICAL, removing the manual effort in template design and allowing the flexibility to explore different circuit topology and sizing at ease.

1) Template-based Layout Generation: Template-based layout generators refer to the design method implemented circuit layout by combining primary devices and structures using module generators. These module generators usually generate layout structures according to predefined rules or template layout designs. Berkeley Analog Generator (BAG) [5] is an open-source framework of template-based layout generation. This method typically requires designers to specify device placement and route (such as routing topology, metal layer, and via cuts) in a parameterized fashion, allowing designers to modify circuit sizing. Since the designer has complete control over each device placement and routing, it could produce high-quality layouts scalable to system-level designs verified post-silicon measurements, such as SAR ADCs [17], [16], SerDes transceivers [32], and memory cells [12]. Nonetheless, it requires a significant amount of manual effort to design general layout templates or PCELLs, where device layouts are still manually programmed to be placed and routed.

2) Optimization-based Layout Generation: Optimization-based method [6], [7] formulates the layout placement and routing as an optimization problem and aims at producing a fully automated layout. The device placements are determined by numerically minimizing the wirelength and automatically routed with a maze routing algorithm. Compared with template-based layout generation, optimization-based methods generate fully automated layout solutions that do not require additional effort to design layout templates. However, it further requires constraint recognition, such as device symmetry [33], building block symmetry [34], and common-centroid matching [35], to ensure the layout quality.

MAGICAL [6] is an open source layout generator with silicon proved measurements [36], where only the circuit netlist is required as the input. The placement is formulated as a minimization problem as follows:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & f^{WL}(\boldsymbol{x},\boldsymbol{y}), \\ s.t. \quad & O^{OVL}, O^{ASYM} = 0, \end{aligned} \tag{4}$$

where $(\boldsymbol{x}, \boldsymbol{y})$ denotes the device placement coordinate, f^{WL} is the weighted sum of wirelength for all nets, and O^{OVL} and O^{ASYM} are the device overlap and symmetry constraints. A gradient-based non-linear optimization method [8] is used to solve the device global placement solution and then is legalized to ensure the constraints in non-overlap and device symmetry are met. The grid-based router [9] uses a maze routing algorithm while considering net symmetry based on the pin placement. A robust rip-up and re-routing scheme is implemented to ensure technology design rules and resolve routing congestion. The final DRC and LVS clean GDSII layout are generated as output.

IV. The OpenSAR Framework

In this section, we present the details of OpenSAR framework. We first introduce the design flow in Sec. IV-A. The template-based CDAC array layout generator is presented in Sec. IV-B. We formally formulate yield optimization for optimizing bit redundancy in Sec. IV-C and solved with integer non-linear programming with layout constraints based on the CDAC layout generator. The design for custom analog circuit components contains automatic sizing and optimization-based layout with MAGICAL is introduced in Sec. IV-D. The digital SAR control logic is implemented with commercial digital flow in Sec. IV-E. Finally, the detailed system-level layout integration is discussed in Sec. IV-F.

A. Framework Overview

OpenSAR leverages a common-mode based charge recovery switching method [37] as shown in Fig. 1. All CDAC capacitors' bottom-plates are connected to V_{cm} during the sampling phase. The MSB to LSB bits are switched in a differential manner based on the comparator decision in all binary search-based SAR ADCs. For example, if the positive capacitor array bit is connected to V_{refh} based on the comparator decision, then the corresponding negative capacitor array bit should be connected to V_{refl}.

The entire design flow for OpenSAR ADC compiler is shown in Fig. 2. OpenSAR requires minimal designer input, with only design specifications, technology files, and DRC rules. The minimal input specification includes the required ADC resolution, speed, number of redundant bits (Sec. IV-C), clock cycles for sampling (default to 1). The top-level specifications are translated to component design specifications based on equations. The components are separately generated first. The CDAC array is first optimized based on the allowed number of redundancy bits with yield considerations and layout generated with a template-based layout generator for high layout quality. The analog components, including bootstrap sampling switch and comparator, are sized automatically with Bayesian optimization. The designer should provide the analog circuit topology and simulation testbench for automated device sizing. The analog component layouts are generated by the optimization-based layout generation tool MAGICAL without any additional input. The layout generation process can also be included inside the Bayesian optimization loop to ensure layout quality [38]. Finally, the system-level layout is integrated. The floorplan and placement step would adjust the component layout by applying additional layout boundary constraints to the default MAGICAL placement objective (Sec. IV-D). The top-level routing includes two steps, where sensitive nets are routed first and digital last. Component placement spacing in the top-level placement and CDAC layout could be adjusted for pin access ability and routability considerations. If the post layout simulated performance does not satisfy design specifications, the user can intervene in the design process by adjusting component specifications, sizing exit criteria, etc.

Table I compares OpenSAR with prior work on SAR ADC automation. OpenSAR provides complete end-to-end ADC compilation with little manual effort in setup requirements. By leveraging optimization-based layout automation MAGICAL, OpenSAR removes the need to provide layout templates or design libraries. Designers only need to provide circuit topology and simulation testbench for analog circuit components such as bootstrap sampling and comparator, offering more design flexibility. Bayesian optimization automates the device

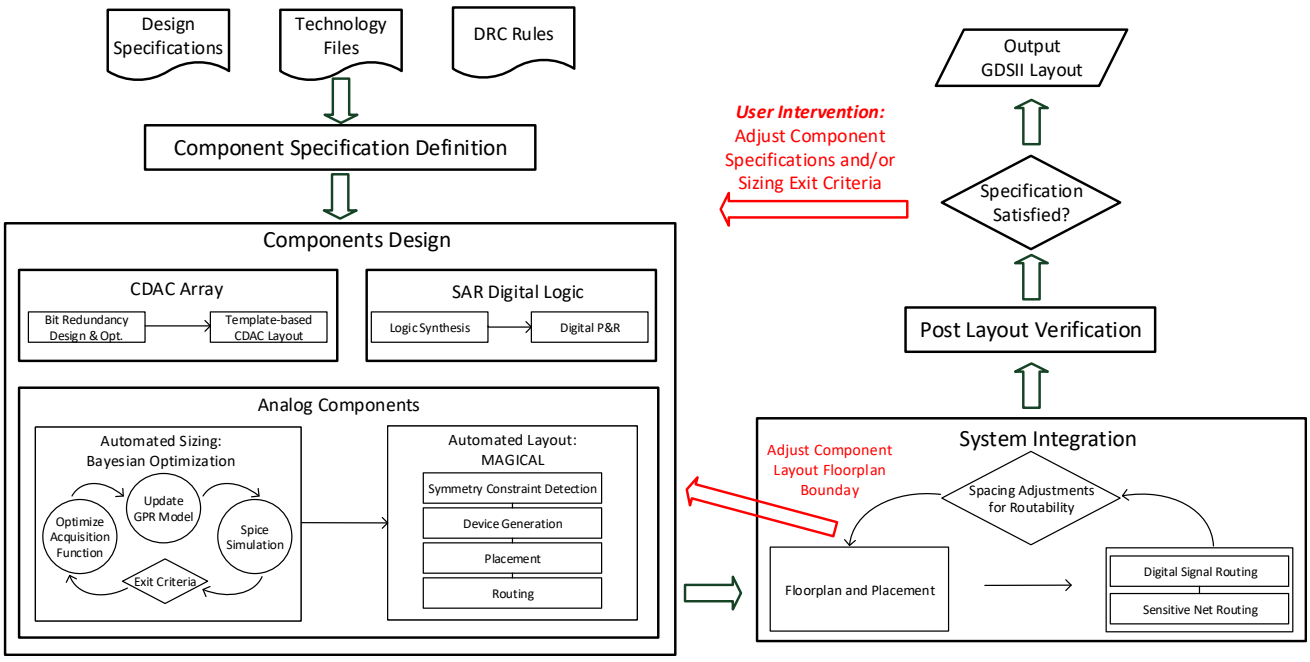


Fig. 2: OpenSAR framework design flow.

		CDAC Array	Bootstrap Sampling	Comparator	System Integration
[22]	Design	NA –	Standard Cells	Standard Cells	NA –
	Layout	Template	Digital APR	Digital APR	Digital APR
[23]	Design	NA –	Equation	Equation	Iterative Opt.
	Layout	NA –			
[16]	Design	NA –			
	Layout	Template	Template	Template	Script Routing
[17]	Design	NA –	Knowledge	Library	NA –
	Layout	Template	Template	Library	Script Routing
OpenSAR	Design	Redundancy Opt.	Bayesian Opt.	Bayesian Opt.	Manual
	Layout	Template	MAGICAL	MAGICAL	Automated P&R

TABLE I: Comparison with prior work.

sizing procedural and further optimizes the system energy efficiency. Almost all efforts on layout automation of prior work are based on template-based layouts and script-based routing, restricting the flexibility for device sizing and circuit topology. OpenSAR also considers yield optimization during the CDAC array design stage by leveraging bit redundancy. Furthermore, OpenSAR includes top-level floorplan, where component layouts aspect ratios are automatically adjusted. The dual-stage top-level automated routing preserves signal integrity and layout quality while removing manual effort in scripted routing. OpenSAR is flexible for designers to alter any part of the design process with custom designs. Different component circuit topologies can be explored efficiently, and downstream layout placement and routing can be integrated seamlessly with manually drawn component layouts.

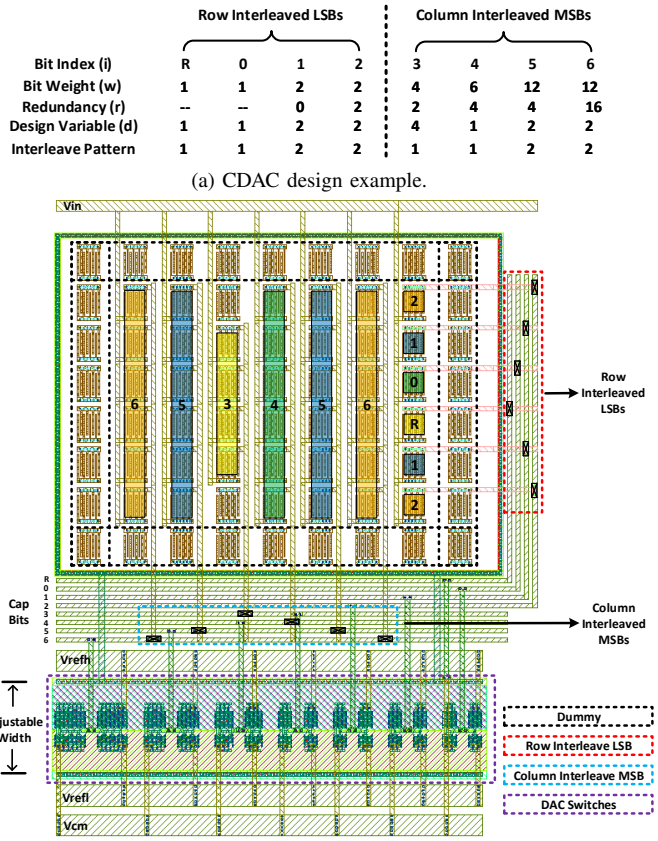
B. Capacitor Digital-to-Analog Converter Array Generator

The capacitor digital-to-analog converter (CDAC) array layout is implemented with a template-based layout generation, where the unit capacitor width, length, and finger number can all be adjusted. Several prior work [39], [40], [41] go to extremes at dispersing the placement of capacitors, resulting in high parasitics overhead and routing complexity.

This work proposes a simpler approach with row and column-based interleaving for LSB and MSB capacitor dispersion. Figure 3 is an example of the proposed CDAC layout generation. The 7-bit array design is shown in Fig. 3(a), where non-binary weights are assigned to different bit indexes. In the shown example, bit R is the reference unit cap with the bottom plate always connected to V_{cm}. The LSB bits from R to 2 are row interleaved, and the MSB bits from 3 to 6 are column interleaved, as shown in Fig. 3(b). The two lowest columns interleaved MSBs (3 and 4) consist of a single column, with 4 equal to the sum of row interleaved LSB bits. The rest MSBs (5 and 6) must be even multiples of the single column. The interleave patterns can automatically be generated and assigned by placing the corresponding contacts in the routing region. In this way, the generated row and column capacitors are separately common-centroid and further interleaved to mitigate random process gradient mismatch effects. The following rules need to be obeyed when determining the CDAC for bit redundancy design to ensure layout feasibility,

1) The weights must be non decreasing.
2) The weights of the two lowest row interleaved LSBs (bit R and 0) are 1.
3) The weights of other LSBs must be even.
4) The weight of the lowest column interleaved MSB must not be larger than the sum of LSBs.
5) The weight of the second-lowest MSB must be exactly equal to the sum of LSBs.
6) The weights of other MSBs must be even multiples of the sum of LSBs.

The DAC switches for each bit are also placed and routed beneath the generated CDAC array. The switch width is automatically determined through transient simulations, such that the DAC settling time could be achieved for each cap bit load. Furthermore, the spacing of the switch can be adjusted. This adjustable width could be further finetuned in the subsequent system integration step IV-F, such that during routing, there is enough spacing for pin accessibility.



(a) CDAC design example.

(b) CDAC layout example.

Fig. 3: CDAC array automated layout generation.

C. Bit Redundancy Design and Optimization

OpenSAR leverages a redundant non-binary search algorithm [42] such that mistakes of comparator decisions can be digitally corrected. This improves the SAR ADC robustness, sampling speed, and tolerance to layout mismatch.

The CDAC bit design shown in Fig. 3(a) is an example with redundancy. The redundancy for bit i is calculated as:

$$r_i = w_R + \sum_{j=0}^{i-1} w_j - w_i. \tag{5}$$

The conventional SAR ADC with binary weighted CDAC array would have redundancy of 0 for every bit. Assume mismatch in unit capacitors are independent Gaussian distributions with mean C_u and standard deviation of $\sigma_u C_u$, then the ith bit redundancy r_i is the following Gaussian distribution:

$$\mu_i = (w_R + \sum_{j=0}^{i-1} w_j - w_i) \cdot C_u, \tag{6}$$

$$\sigma_i = \sigma_u C_u \cdot \sqrt{w_R^2 + \sum_{j=0}^{i} w_j^2}. \tag{7}$$

Thus for the redundancy to able to calibrate an error, r_i needs to be positive. Assuming tolerating k_i variance $\mu_i - k_i\sigma_i = 0$,

$$k_i = \frac{\mu_i}{\sigma_i} = \frac{w_R + \sum_{j=0}^{i-1} w_j - w_i}{\sigma_u \sqrt{w_R^2 + \sum_{j=0}^{i} w_j^2}} \tag{8}$$

Based on the above analysis, we formally formulate the bit redundancy design with the following max-min objective:

$$\max \min_{i \geq 2} k_i(w). \tag{9}$$

This objective is to maximize the worst capacitor error tolerance for any bit. Suppose a target ADC resolution of N bit, with bit 0 to N_r row interleaved LSBs, bit $N_r + 1$ to $N_r + N_c$ column interleaved MSBs, adhering to the layout related rules described in Sec. IV-B:

$$\begin{aligned} w_i &\leq w_{i+1}, 0 \leq i \leq N_r + N_c \\ w_R &= w_0 = 1 \\ w_i &= 2d_i, 1 \leq i \leq N_r \\ w_{(N_r+1)} &\leq w_R + \sum_{i=0}^{N_r} w_i \\ w_{(N_r+2)} &= w_R + \sum_{i=0}^{N_r} w_i \\ w_i &= w_{(N_r+2)} \cdot 2d_i, N_r + 3 \leq i \leq N_r + N_c \end{aligned} \tag{10}$$

Furthermore, the entire design need to cover the N bit requirement, while not excessive capacitor usage:

$$2^N < w_R + \sum_{i=0}^{N_r+N_c} w_i < 2^N + 2^{N_r} \tag{11}$$

The free design variables are d_i and need to be integers. Thus the formulated problem has a non-linear objective with linear integer constraints, which can be solved with open-sourced (mixed) integer non-linear programming (MINLP) solvers such as APOPT [43], [44].

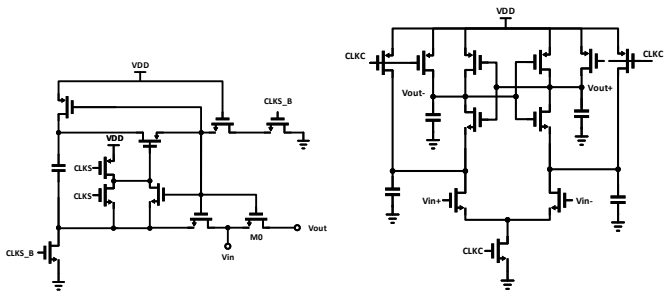
D. Analog Circuit Components

The analog circuit components, including bootstrap sampling switch and comparator, are sized automatically with Bayesian optimization described in Sec. III-A. The designer only needs to input the circuit topology and simulation testbench. The designer can further specify when the optimization terminates, such as the maximum number of simulations or when the design criterion is met. The simulation objective is usually to minimize power consumption. Performance constraints of Eq. (1) can be calculated based on equations, such that the system achieves the desired performance specifications. The layout is generated by MAGICAL and requires no additional input. The automatic placement and routing entirely remove any additional manual efforts. We can add additional constraints to the placement objective in Eq. (4) to restrict the placement aspect ratio.

1) Bootstrap Sampling Switch: The circuit topology is shown in Fig. 4(a). Only the sampling switch size M_0 need to be adjusted for different sampling speed and resolution.

2) Strong-arm Latch Comparator: The circuit topology is shown in Fig. 4(b). All devices are sized automatically to minimize power consumption. The corresponding performance constraints are input-referred noise V_{noise} and delay t_d calculated as follows:

$$V_{noise} < \frac{1}{4} V_{LSB} = \frac{V_{DD}}{4\sqrt{2} \cdot 2^{N-1}} \tag{12}$$



(a) Bootstrap sampling switch. (b) Strong-arm latch.

Fig. 4: Analog circuit components.

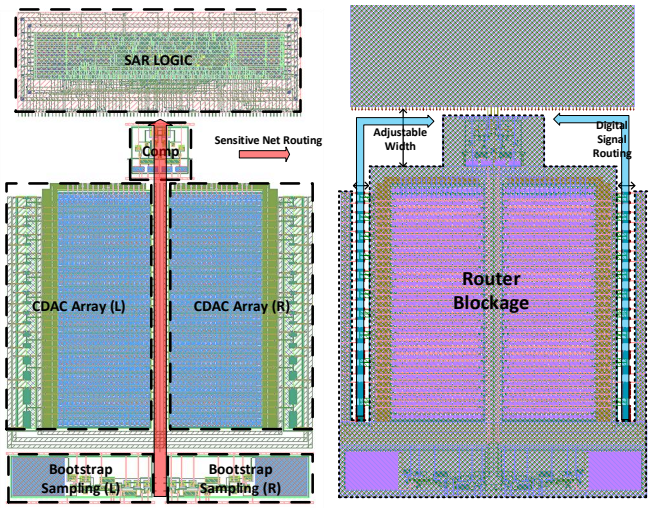
$$t_d < \frac{1}{2F_s N_{cycle}} \tag{13}$$

with V_{DD} the power supply voltage, N bit the targeted resolution, F_S the sampling rate, and N_{cycle} the number of clock cycles convert one sample.

E. SAR Digital Logic

The control logic for SAR can be easily synthesized and implemented with digital APR tools. The aspect ratio can be altered with floorplan scripts. It is also important to plan the pins connecting to DAC switches carefully to ensure routability in top level routing.

F. System Integration



(a) Floorplan and sensitive routing. (b) Digital signal routing.

Fig. 5: Top level layout integration

The top-level layout is integrated with automatic floorplan, placement, and routing. The floorplan for SAR architecture is shown in Fig. 5(a), which allows the critical signal to pass symmetrically in the center of the layout directly. After generating the CDAC array layout, other components' layout floorplan boundaries, such as bootstrap sampling and SAR logic, can be adjusted accordingly. We do not manipulate the floorplan of the comparator since it is found that extreme aspect ratio results in large routing parasitics and degrades the performance. The building blocks are automatically placed symmetrically along the center axis according to the floorplan.

The automated routing is performed with the maze router in MAGICAL in [9]. The pin locations could be calculated based on the component placement. The pin connections are defined in a separate file, which only specifies the pin names that need to be connected. The routing consists of two separate stages. In the first stage, the sensitive analog nets, including clock signals for sampling, CDAC bottom-plate connections etc., are routed. The router automatically recognizes and satisfies net symmetry constraints based on the pin locations. In the second stage, the digital signal nets connecting SAR logic to CDAC switches are routed. During this stage, we create a blockage to the router as shown in Fig. 5(b) so that the strong digital aggressors would not couple to the sensitive analog nets. The spacing of DAC switches and component placement can be automatically adjusted for better pin accessibility and routing congestion.

V. Experimental Results

In this section, we present two prototype designs based on the OpenSAR framework. Both designs are generated in TSMC 40nm technology. All experiments are conducted on a Linux workstation with an 8-core Intel 3.0GHz CPU with 64GB memory. All results are based on post-layout R+C+CC extraction with Mentor PEX and spectre simulation with Cadence.

A. SAR ADC Designs

The first design is a 10bit resolution SAR ADC at 100MS/s. The supply voltage is set to 1.2V so that the SAR logic can function at maximum speed. The design has 1-bit redundancy. For each sample conversion, two cycles are used for sampling, with a total of 13 clock cycles per conversion.

The second design is a 12bit resolution SAR ADC at 1MS/s. Since it targets a lower speed but high resolution, the supply voltage is set to 0.7V to lower the power consumption of SAR logic circuits. The design has 2-bit redundancies for better yield considering its high resolution. For each sample conversion, two cycles are used for sampling, with a total of 16 clock cycles per conversion.

Both of the designs are generated from the OpenSAR framework. The comparator is automatically sized, placed, and routed. During the sizing, we set the Bayesian optimization to stop after 200 iterations. Since the bootstrap switch only has one device to size, a simple variable linear sweep can obtain the sizing; thus, it is performed manually. The SAR logic is synthesized, placed, and routed with Cadence tools. The generated SAR ADC layouts are DRC and LVS clean.

B. Runtime

The runtime for generating the two designs is shown in Table II. The layout generation time for the DAC switch is included in CDAC array. The design of the bootstrap switch is a linear sweep done manually with runtime not counted. It can be observed that the layout generation process is extremely fast. The majority of the time for design is spent on spice simulations for sizing the comparator. Nonetheless, the entire SAR ADC can be generated end-to-end in about 2 hours.

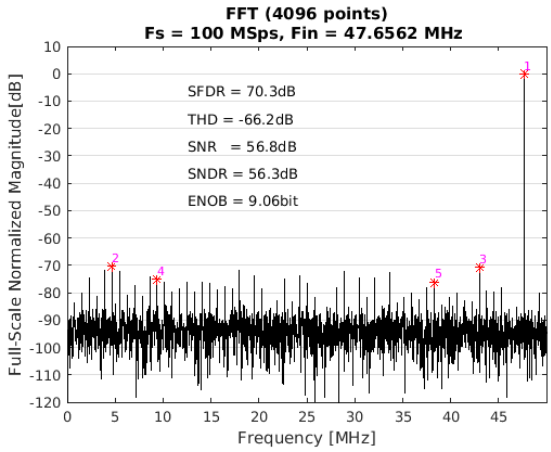
C. Simulated Performance

The post-layout simulation is conducted with extracted R+C+CC layout parasitics and transient noise. The obtained simulated spectra of transient simulations are shown in Fig. 6. It also presents detailed performance results, including SNDR, SFDR, etc. The spice level R+C+CC extracted simulations are extremely time-consuming. As a reference, for the 12bit design, it took more than four weeks to obtain the 4096-point FFT simulated spectrum.

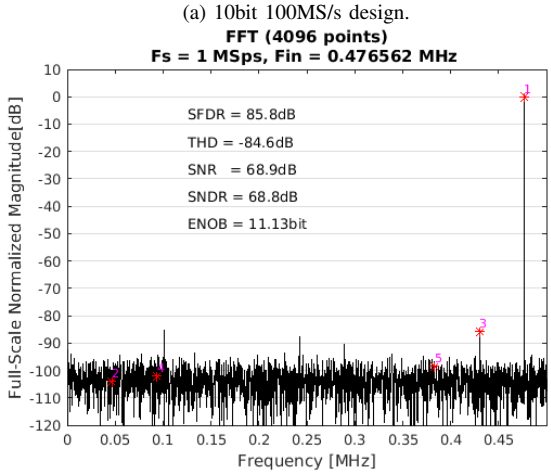
The simulated power consumption breakdown is shown in Fig. 7. For the 10bit 100MS/s design, 71.5% of the power is consumed by

TABLE II: Runtime Breakdown

	10bit		12bit	
	Design	Layout	Design	Layout
DAC Switch	6m33.6s	-	7m51.7s	-
CDAC Array	6.4s	11.7s	42.0s	39.1s
Bootstrap Switch	-	6.9s	-	8.4s
Comparator	1h45m47.3s	4.7s	1h52m32.7s	6.1s
SAR Logic	15.5s	2m56.1s	20.8s	2m49.3s
Top Floorplan	-	26.2s	-	1m4.2s
Top Routing	-	1m40.3s	-	3m15.6s
Total	1h52m42.8s	4m59.7s	2h1m27.2s	8m2.7s



(a) 10bit 100MS/s design.



(b) 12bit 1MS/s design.

Fig. 6: Simulated spectrum.

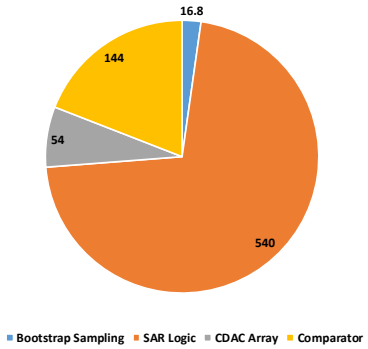
SAR logic. The power consumption could be further improved if we leverage dynamic logic instead of CMOS standard logic; however, we would then be unable to use digital APR tools. In the 12bit 1MS/s design, 47.6% of the power is consumed by the comparator. This demonstrates that OpenSAR is successful at optimizing the system performance at both ends of the design specifications. For high-speed designs, timing is more challenging to achieve; thus, digital logic power would dominate. For high-resolution designs, thermal noise is the limiting factor; thus, the comparator consumes more power to tradeoff for less noise.

Overall, the 10bit 100MS/s design achieves an SNDR of 56.3dB, consuming 754.8μW of power. The 12bit 1MS/s design achieves an SNDR of 68.8dB consuming 9.6μW. We calculated both the Schreier and Walden Figure of Merit (FOM):

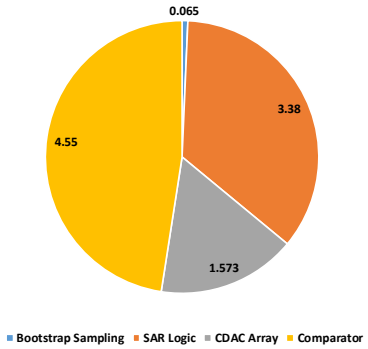
$$FOM_S = SNDR + 10\log_{10}\frac{f_s}{2P} \tag{14}$$

$$FOM_W = \frac{P}{f_s \cdot 2^{ENOB}} \tag{15}$$

where $SNDR$ is the signal-to-noise-distortion ratio, f_s is the sampling frequency, $ENOB$ is the effective number of bits, and P is the power consumption. The results are shown in Table III and compares with prior work on SAR ADC automation. It can be seen that OpenSAR achieves state-of-the-art results in terms of resolution, speed, and FOM.



(a) 10bit 100MS/s design.



(b) 12bit 1MS/s design.

Fig. 7: Simulated power consumption in μW.

VI. Conclusion

In this work, we present OpenSAR, an open source automated end-to-end SAR ADC compiler. The framework requires only system specifications as the minimal input, while highly flexible and allows designers to explore different building block circuit topologies, sizing, and layout solutions at ease. We leverage recent developments in analog design automation, automating both the device sizing and layout generation process. The design of a redundant non-binary weighted CDAC array is optimized for yield considerations. Post layout simulations demonstrate that generated prototype designs achieve state-of-the-art resolution, speed, and energy efficiency.

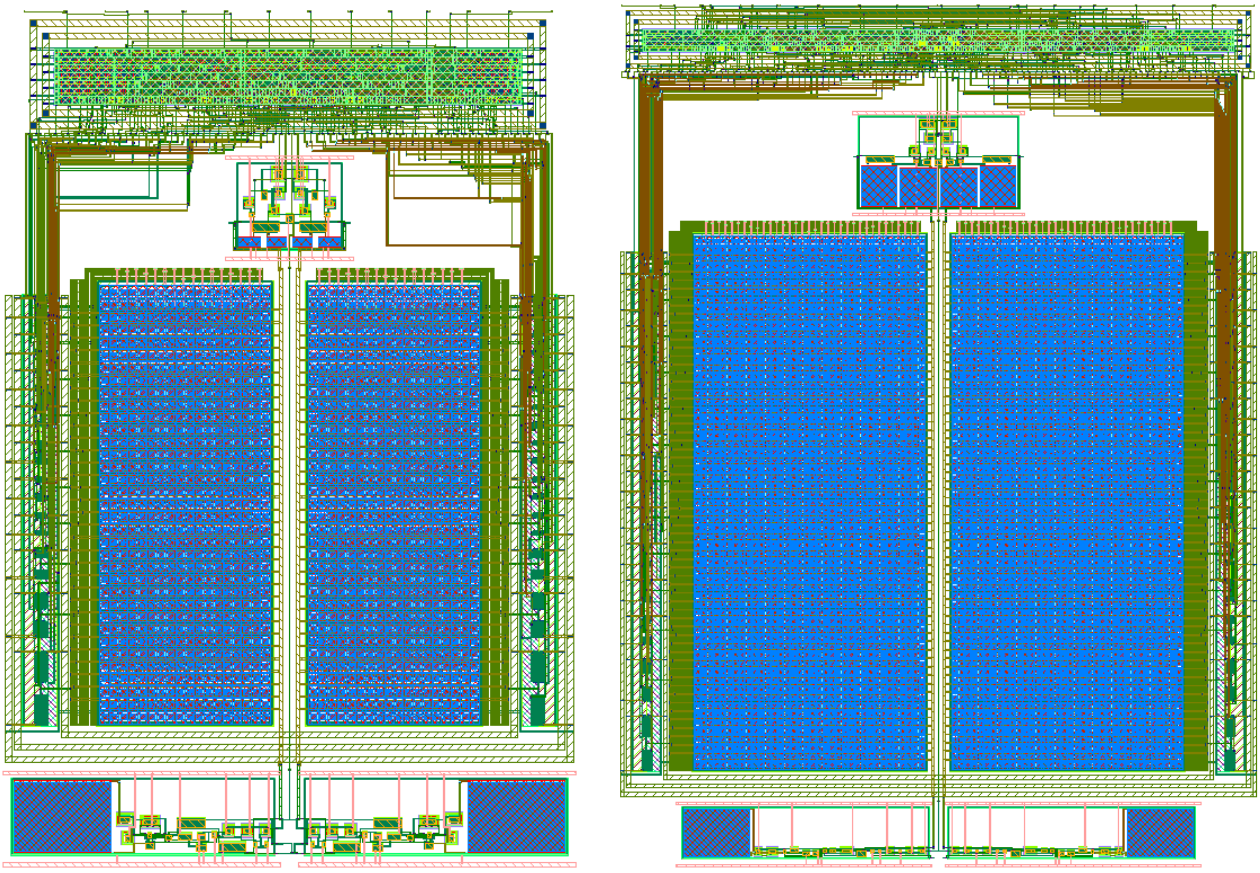
Acknowledgement

This work is supported in part by the NSF under Grant No.1704758, and the DARPA ERI IDEA program.

TABLE III: Performance Comparison with Prior Work on SAR ADC Automation

	Huang† [23]		Seo† [22]		Wulff† [16]		Ding† [17]		OpenSAR*	
Auto-Design	Yes		Yes (Digital)		No		Yes		Yes	
Auto-Layout	No		Yes		Yes		Yes		Yes	
Resolution	12	10	12	11	9		8	12	10	12
Technology (nm)	180	90	180	28	28 FDSOI		40		40	
ENOB	10.1	8.4	10.2	9.1	7.4	7.8	7.6	9.9	9.1	11.1
SNDR (dB)	62.7	52.1	63.3	56.8	46.4	48.8	47.4	61.1	56.3	68.8
BW(MHz)	0.3	25	0.05	25	1	10	16	0.5	50	0.5
FOM_S (dB)	152.2	159.7	155.3	164.8	166.8	166.8	156.7	165.8	166.8	176.0
FOM_W (fJ/c.step)	500	26.7	265.5	14.1	2.7	3.5	30.7	18.1	10.8	4.3

*Simulated results with layout R+C+CC parasitic extraction. †Tapeout measurement.



(a) 10bit 100MS/s design.

(b) 12bit 1MS/s design.

Fig. 8: OpenSAR generated layouts.

REFERENCES

[1] B. Murmann, "The race for the extra decibel: A brief review of current adc performance trajectories," *IEEE Solid-State Circuits Magazine*, vol. 7, no. 3, pp. 58–66, 2015.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi *et al.*, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] W. Lyu, P. Xue, F. Yang *et al.*, "An efficient bayesian optimization approach for automated optimization of analog circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1954–1967, 2018.

[4] H. Wang, K. Wang, J. Yang *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)*, 2020, pp. 1–6.

[5] J. Crossley, A. Puggelli, H. Le *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)*, 2013, pp. 74–81.

[6] B. Xu, K. Zhu, M. Liu *et al.*, "Magical: Toward fully automated analog ic layout leveraging human and machine intelligence: Invited paper," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.

[7] K. Kunal, M. Madhusudan, A. K. Sharma *et al.*, "Align – open-source analog layout automation from the ground up," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–4.

[8] K. Zhu, H. Chen, M. Liu *et al.*, "Effective analog/mixed-signal circuit placement considering system signal flow," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–9.

[9] H. Chen, K. Zhu, M. Liu *et al.*, "Toward silicon-proven detailed routing for analog and mixed-signal circuits," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–8.

[10] T. Ansell and M. Saligane, "The missing pieces of open design enablement: A recent history of google efforts," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–8.

[11] "SkyWater." [Online]. Available: https://github.com/google/skywater-pdk

[12] M. R. Guthaus, J. E. Stine, S. Ataei *et al.*, "Openram: An open-source memory compiler," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–6.

[13] T. Ajayi, V. A. Chhabria, M. Fogaça *et al.*, "Invited: Toward an open-source digital flow: First learnings from the openroad project," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–4.

[14] G. Kumar, B. Chatterjee, and S. Sen, "OpenSerDes: an open source process-portable all-digital serial link," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021.

[15] T. Ajayi, S. Kamineni, Y. K. Cherivirala *et al.*, "An open-source framework for autonomous soc design with analog block generation," in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*, 2020, pp. 141–146.

[16] C. Wulff and T. Ytterdal, "A compiled 9-bit 20-ms/s 3.5-fj/conv.step sar adc in 28-nm fdsoi for bluetooth low energy receivers," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 7, pp. 1915–1926, 2017.

[17] M. Ding, P. Harpe, G. Chen *et al.*, "A hybrid design automation tool for sar adcs in iot," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 12, pp. 2853–2862, 2018.

[18] S. Weaver, B. Hershberg, and U. Moon, "Digitally synthesized stochastic flash adc using only standard digital cells," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 1, pp. 84–91, 2014.

[19] V. Unnikrishnan and M. Vesterbacka, "Time-mode analog-to-digital conversion using standard cells," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 12, pp. 3348–3357, 2014.

[20] S. Li, B. Xu, D. Z. Pan *et al.*, "A 60-fj/step 11-enob vco-based ctdsm synthesized from digital standard cell library," in *2019 IEEE Custom Integrated Circuits Conference (CICC)*, 2019, pp. 1–4.

[21] P. Harpe, H. Li, and Y. Shen, "Low-power sar adcs: trends, examples and future," in *ESSCIRC 2019 - IEEE 45th European Solid State Circuits Conference (ESSCIRC)*, 2019, pp. 25–28.

[22] M. Seo, Y. Roh, D. Chang *et al.*, "A reusable code-based sar adc design with cdac compiler and synthesizable analog building blocks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 12, pp. 1904–1908, 2018.

[23] C. Huang, J. Lin, Y. Shyu *et al.*, "A systematic design methodology of asynchronous sar adcs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 5, pp. 1835–1848, 2016.

[24] G. Gielen, H. Walscharts, and W. Sansen, "Analog circuit design optimization based on symbolic simulation and simulated annealing," in *ESSCIRC '89: Proceedings of the 15th European Solid-State Circuits Conference*, 1989, pp. 252–255.

[25] M. d. Hershenson, S. P. Boyd, and T. H. Lee, "Optimal design of a cmos op-amp via geometric programming," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 1–21, 2001.

[26] Bo Liu, F. V. Fernandez, and G. Gielen, "Fuzzy selection based differential evolution algorithm for analog cell sizing capturing imprecise human intentions," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 622–629.

[27] W. Lyu, F. Yang, C. Yan *et al.*, "Multi-objective bayesian optimization for analog/rf circuit synthesis," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[28] S. Zhang, W. Lyu, F. Yang *et al.*, "Bayesian optimization approach for analog circuit synthesis using neural network," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 1463–1468.

[29] S. Zhang, F. Yang, D. Zhou *et al.*, "An efficient asynchronous batch bayesian optimization approach for analog circuit synthesis," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[30] M. Liu, W. Turner, G. Kokai *et al.*, "Parasitic-aware analog circuit sizing with graph neural networks and bayesian optimization," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021.

[31] M. Balandat, B. Karrer, D. R. Jiang *et al.*, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization," in *Advances in Neural Information Processing Systems 33*, 2020.

[32] S. Han, S. Jeong, C. Kim *et al.*, "Gui-enhanced layout generation of ffe sst txs for fast high-speed serial link design," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[33] M. Eick, M. Strasser, K. Lu *et al.*, "Comprehensive generation of hierarchical placement rules for analog integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 2, pp. 180–193, 2011.

[34] M. Liu, W. Li, K. Zhu *et al.*, "S3DET: Detecting system symmetry constraints for analog circuits with graph similarity," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 193–198.

[35] A. Sharma, M. Madhusudan, S. Burns *et al.*, "Common-centroid layouts for analog circuits: Advantages and limitations," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021.

[36] H. Chen, M. Liu, X. Tang *et al.*, "Magical 1.0: An open-source fully-automated ams layout synthesis framework verified with a 40-nm 1GS/s $\Delta\Sigma$ adc," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*, 2021.

[37] Y. Zhu, C. Chan, U. Chio *et al.*, "A 10-bit 100-ms/s reference-free sar adc in 90 nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 6, pp. 1111–1121, 2010.

[38] M. Liu, K. Zhu, X. Tang *et al.*, "Closing the design loop: Bayesian optimization assisted hierarchical analog layout synthesis," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[39] C. Lin, J. Lin, Y. Chiu *et al.*, "Common-centroid capacitor placement considering systematic and random mismatches in analog integrated circuits," in *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2011, pp. 528–533.

[40] M. P. Lin, Y. He, V. W. Hsiao *et al.*, "Common-centroid capacitor layout generation considering device matching and parasitic minimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 7, pp. 991–1002, 2013.

[41] Y. X. Ding, F. Burcea, H. Habal *et al.*, "Pastel: Parasitic matching-driven placement and routing of capacitor arrays with generalized ratios in charge-redistribution sar-adcs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 7, pp. 1372–1385, 2020.

[42] T. Ogawa, H. Kobayashi, M. Hotta *et al.*, "Sar adc algorithm with redundancy," in *APCCAS 2008 - 2008 IEEE Asia Pacific Conference on Circuits and Systems*, 2008, pp. 268–271.

[43] "APOPT." [Online]. Available: https://apopt.com/

[44] L. Beal, D. Hill, R. Martin *et al.*, "Gekko optimization suite," *Processes*, vol. 6, no. 8, p. 106, 2018.