Microelectronics Journal 158 (2025) 106607

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

WiseEDA: LLMs in RF Circuit Design

Hangjiang Jin, Junchao Wang *, Junjie Sheng, Yifan Wu, Jiayu Chen, Yaqi Wang, Jun Liu

Innovation Center for Electronic Design Automation Technology, Hangzhou Dianzi University, Hangzhou, 310018, China

ARTICLE INFO

Keywords:
RFIC
EDA
LLM
Netlist

ABSTRACT

As the complexity of Radio Frequency Integrated Circuit (RFIC) design increases, the significance of Electronic Design Automation (EDA) becomes more pronounced. This paper proposes a circuit design methodology utilizing Large Language Models (LLMs), incorporating tools for topology selection and netlist optimization based on a particle swarm optimization algorithm. These tools are driven by LLMs, enabling engineers to describe their requirements in natural language. The LLM then selects suitable topologies and automatically configures relevant parameters for the optimizer, facilitating the automated netlist circuit design. Experimental results demonstrate that, after equipping the LLM with relevant knowledge through prompt engineering, it can optimize the values of capacitors, inductors, and other parameters in the band-pass filter netlist. This ensures that the filter's S11 and S21 performance meet the specified requirements within a particular frequency band, thereby confirming the feasibility of employing LLMs in the automated circuit design process.

1. Introduction

Radio Frequency Integrated Circuits (RFIC) are a fundamental component of modern communication technology, and their design becomes significantly more challenging as frequency and complexity increase. In traditional design, circuit engineers need to rely on extensive professional knowledge and experience to complete the design through complex calculations and simulations. At the same time, they also need to spend much time on subsequent optimization and validation. This process is time-consuming and labor-intensive, requiring high professional competence. With the rapid advancements in Artificial Intelligence (AI) and machine learning, numerous researchers have begun to apply these technologies to circuit design [1]. Machine learning algorithms, particularly deep learning, with their robust data processing and pattern recognition capabilities, can uncover hidden patterns within vast datasets, providing novel insights and methods for circuit design [2–5]. By training models to learn and simulate expert knowledge and experience in the circuit design process, algorithms can assist engineers in completing design tasks more swiftly and accurately, significantly reducing both difficulty and workload [6,7]. Currently, the optimization of existing algorithms and the combination of multiple optimization algorithms before application have become a prevalent trend [8–10].

Large Language Models (LLMs), which have evolved from natural language processing algorithms, have made groundbreaking progress in recent years. Studies indicate that once the size of a language model surpasses a certain threshold, a notable enhancement in performance occurs [11,12]. LLMs have garnered substantial attention from researchers due to their emergent capabilities [13]. Their powerful text processing and comprehension abilities have demonstrated wide-ranging potential applications across various fields, with many researchers attempting to apply LLMs within their domains, achieving promising results [14]. Autonomous agents have always been a focal point of academic research, and the human-like intelligence exhibited by LLMs has sparked a wave of research into LLM-based autonomous agents [15–18].

The rapid development of LLMs has presented researchers with a new trajectory for the future of Electronic Design Automation (EDA), termed AI+EDA [19]. There are already studies applying LLMs as agents within EDA. For instance, researchers have developed an AI interactive assistant plugin for EDA software called SmartonAI, which leverages large language models to make complex EDA software more user-friendly. This plugin significantly simplifies the PCB design process by transforming complex commands into intuitive, language-based interactions [20]. Haoyuan Wu and others have developed an autonomous agent system named ChatEDA, driven by the large language model AutoMage, designed to generate code for operating EDA tools using natural language instructions [21]. Additionally, research has applied LLMs to generating and simulating SPICE code, constructing a framework called SPICEPilot, and generating related datasets to facilitate SPICE simulation and expedite the design process [22].

* Corresponding author.
E-mail address: junchao@hdu.edu.cn (J. Wang).

https://doi.org/10.1016/j.mejo.2025.106607
Received 2 December 2024; Received in revised form 13 February 2025; Accepted 14 February 2025
Available online 22 February 2025

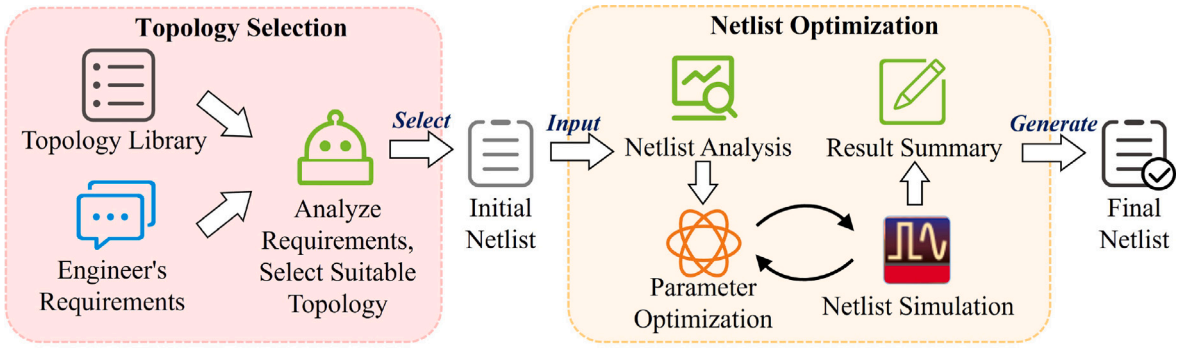

Fig. 1. Overview of the LLM-Based Design Agent Workflow. It consists of the topology selection process on the left and the netlist circuit optimization process on the right, realizing the automation of netlist circuit design through LLM.

In the field of RFIC, no scholarly work has yet introduced the application of large language models for designing circuits represented in netlist form. Netlists, representing abstract circuits, delineate logical relationships and are crucial in integrated circuit design. Since netlists are expressed textually, they can be ingested and interpreted by LLMs. In the traditional form of using algorithms to optimize netlists, engineers need to have a certain level of algorithm knowledge to know how to use algorithms. If engineers want to utilize algorithms' performance fully, they also need to spend a lot of extra time and effort to gain a deep understanding of algorithm knowledge. Although mainstream industrial software may provide basic algorithms, they frequently lack interfaces for configuring algorithm parameters, limiting engineers to default settings. Therefore, this study proposes a novel LLM-based proxy design tool that utilizes an LLM to select topological structures and optimize their corresponding netlists, substituting the traditional circuit design methods employed by engineers.

Compared to previous research, our innovations include:

1. Introduction of an LLM-Based Topology Selection Module: We have introduced an LLM-based topology selection module, engineers can communicate with the LLM using natural language, allowing the LLM to quickly filter suitable topologies based on requirements, thus simplifying the traditional topology selection process.
2. Automation of the Design Optimization Process: The LLM replaces engineers in configuring optimization parameter ranges and algorithm hyperparameters, and can automatically use the optimizer to optimize the circuit. This lowers the design barriers, reduces the need for engineers to acquire algorithm-related knowledge, and enhances design efficiency.

Utilizing prompt engineering [23,24], we equip the LLM with relevant knowledge and tools to select suitable topologies based on user requirements. Subsequently, LLM analyzes the topology structure, sets the parameters in the netlist of the topology structure according to the requirements, and finally automatically sets and calls the PSO algorithm to provide a netlist that meets the requirements. Engineers simply "order" their requirements, akin to ordering in a restaurant, while the LLM autonomously manages the remaining steps. Compared to traditional circuit design methods, our approach automates manual processes, reducing design complexity, enhancing efficiency, and ensuring reliable results.

2. LLM agent design

Fig. 1 illustrates the overall process flow of the LLM-based agent design, which can be broadly divided into two parts: topology selection and netlist optimization.

In the topology selection phase, the LLM acts like a server, analyzing user requirements and identifying corresponding topologies from the topology library to provide to the user.

During the netlist optimization phase, the LLM takes on the perspective of a circuit engineer, analyzing the circuit netlist and emulating the optimization methods of circuit engineers. It designs tools for parameter extraction and substitution, identifies netlist parameters that require optimization along with their ranges, and finally uses an optimization tool with a PSO algorithm to refine the netlist, resulting in a circuit that meets the specified criteria.

2.1. Topology selection module

In selecting topology, the LLM plays a role similar to that of a waiter, undertaking the task of analyzing and interpreting user needs. At the beginning of the task, LLM will understand the specific task process through the pre-set prompts, just like a waiter needs to receive training before starting work. Subsequently, like a skilled waiter recommending the most suitable dishes based on customer preferences, the LLM engages in dialogue with the user to analyze and further comprehend the described needs. It then selects an appropriate topology from our library and presents it to the user. As shown in Fig. 2, the user needs to provide a detailed description of their requirements, such as the type of functional unit, applicable frequency bands, and gain.

Upon receiving the user's needs, the LLM first uses local tools to interpret the topology library relevant to the required components. It searches for metrics that match the user description within this library. The corresponding netlist is provided directly if a topology fully meets the requirements. If not, the LLM offers one or more netlists of topologies with similar metrics, describing the current performance of each structure for user selection. The user can choose the topology they find suitable, request recommendations from the LLM, or even let the LLM decide which topology to select. If the user is unsatisfied with the current options, they can ask the LLM to present a new batch of topologies or redefine their needs.

Additionally, the user can request the LLM to display the netlist structure and schematic of the chosen topology (if available in the library) for a more in-depth understanding. Once the user has determined which topology to use, the selection process concludes, and the LLM forwards the local path of the chosen topology to the subsequent netlist optimization process.

2.2. Netlist optimization module

Fig. 3 illustrates the netlist optimization process, which is divided into three parts. The part of (a) is the optimizer setup, which involves parsing the received netlist to set the optimization parameters and algorithm hyperparameters. The part of (b) is simulation optimization. Parameters and metrics are passed to the optimization algorithm, which employs the PSO algorithm for netlist design optimization. The part of (c) is result generation. The final optimization results are evaluated, and the optimized netlist is output.

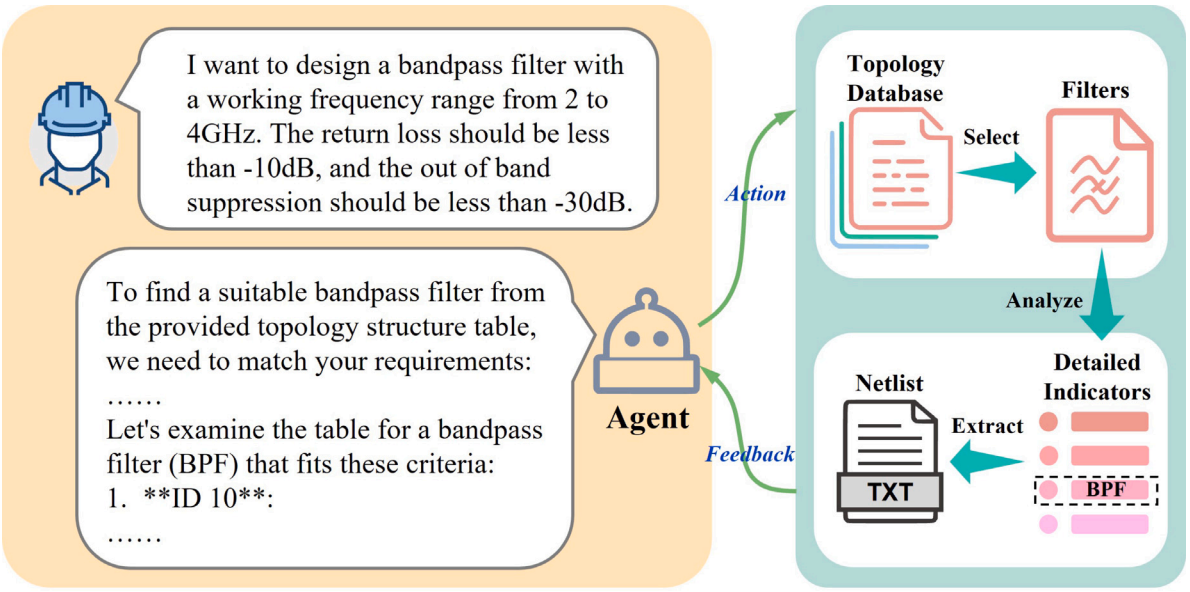

Fig. 2. Example of topology selection. After the user specifies their requirements for a filter, the LLM accesses the local filter topology library, searches for netlists that closely match the requirements, and returns the current performance of each topology.

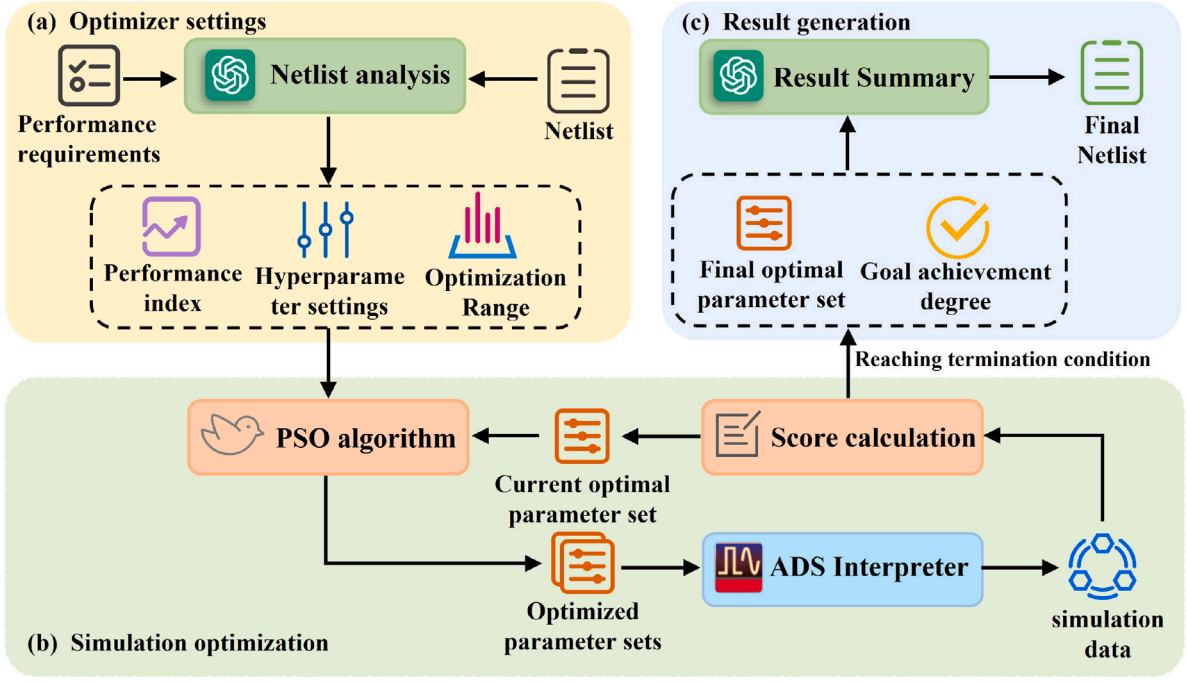

Fig. 3. Netlist optimization process. The LLM first generates the necessary parameters for the optimizer based on the netlist and requirements. The optimizer then parses these parameters to begin iterative simulation and optimization of the netlist. Finally, the LLM summarizes and evaluates the optimizer's results, producing the final netlist.

During the optimizer setup, the netlist analysis module retrieves the netlist from the local repository based on its address and conveys it in natural language to the LLM for comprehension. The LLM then analyzes the complexity of the netlist circuit, determining the variables and their ranges, while automatically configuring the PSO hyperparameters in accordance with the interface requirements of the PSO optimization tool and passing this information to the algorithm optimization module. The optimization range is a critical parameter, as the optimizer needs an appropriate search to ensure the optimization results meet the requirements. Therefore, after setting the ranges for the netlist variables, the LLM presents this information to the user. If the user is unsatisfied with the ranges generated by the LLM and wishes to modify them, they can provide ranges based on their own experience, allowing the LLM to update the information that needs to be conveyed to the subsequent processes according to the user's needs. Additionally, the values of the PSO hyperparameters are also modifiable; if the user has different requirements for the hyperparameter values generated by the LLM, they can provide modification information to the LLM.

The PSO algorithm generates multiple parameter sets during simulation optimization through internal calculations. These parameters are fed into the ADS Interpreter to produce various netlists for simulation. Simulation results are scored and fed back into the optimization algorithm to guide future iterations until termination criteria are met.

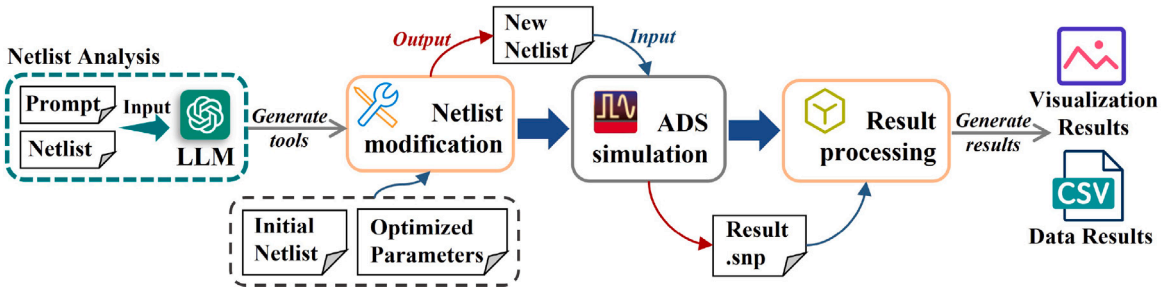

Fig. 4. ADS interpreter. The LLM utilizes prompts and the netlist to generate a netlist modification tool. The initial netlist and optimization parameters are input into this tool to produce a new netlist, which is then passed to the ADS simulator. The generated results are processed by the results handling tool before being returned.

In the result generation step, the result summary module provides the optimal parameter set and scored simulation results to the LLM for evaluation. The LLM will provide feedback on the netlist circuit design's results based on the achievement of the netlist's objectives and provide an optimized netlist.

2.2.1. ADS interpreter

The ADS Interpreter is a tool for interacting between programs and the ADS simulator. Its core functionality involves processing received netlist files and handling the simulation results returned by the ADS simulator. After parsing the current netlist content, the LLM generates the netlist modification tool, allowing adaptation to different netlist formats.

As shown in Fig. 4, after receiving parameters from the optimization tool, the ADS Interpreter uses the netlist modification tool to match and modify parameters in the netlist. This modified circuit netlist is then saved locally in a text format suitable for the ADS simulator. The netlist is transferred to the ADS simulator via the computer's ADS interface for detailed circuit simulation and analysis.

After completing the simulation, the ADS simulator generates an SNP file containing comprehensive simulation results. The ADS Interpreter processes these results to enhance data usability and readability. It removes non-essential information, such as netlist names and simulation dates, to reduce data redundancy. The tool then converts these complex data into decibels (dB) to better assess circuit performance, a common data representation method in RF engineering. Then, the tool will transmit the data to optimize the tool, allowing it to measure the performance of this simulation and confirm the next optimization direction.

Furthermore, the ADS Interpreter will save the data locally in CSV format and convert the simulation results into graphical images. After each simulation finishes, the tool automatically saves the CSV data and converts it into PNG images, with frequency on the x-axis and dB on the y-axis. This visual representation provides users with an intuitive way to observe and analyze each simulation's results.

2.2.2. Circuit optimization tool

The optimization of netlist circuits primarily involves setting the parameter optimization range, defining optimization objectives, and configuring optimization algorithms. Typically, engineers import a netlist with a given topology into relevant software for processing. Industrial software parses the netlist, extracts parameters, and displays them. Engineers use their circuit knowledge and rich design experience to perform initial calculations and manually optimize or set up appropriate optimization algorithms to tune the netlist.

In this work, the LLM replaces engineers in conducting relevant operations. Guided by our prompts, the LLM follows a predefined process to complete tasks. Equipped with a certain level of circuit design knowledge, the LLM analyzes the topology netlist to extract parameters suitable for optimization. It then sets the optimization parameters and design scope, formatting the design objectives to meet the requirements of the algorithm interface. The algorithm iteratively selects parameters and uses feedback from the ADS interpreter to calculate the score for each parameter set, updating selections to move closer to the target. The optimization concludes when the maximum iteration count or a predefined score limit is reached, with the LLM providing the optimized results and netlist to the user.

We employed Particle Swarm Optimization (PSO) to select the optimization algorithm. It is a swarm intelligence-based optimization algorithm that simulates the foraging behavior of bird flocks. It seeks the optimal solution to the problem through information sharing among individuals. Currently, PSO is one of the mainstream optimization algorithms and has been applied in circuit design [25,26]. PSO can efficiently optimize circuit parameters, such as capacitance and inductance, to meet specific performance metrics [27,28]. Each particle in PSO represents a set of circuit parameters, and the algorithm iteratively updates the particles' speed and position to gradually approach the optimal solution. Specifically, each particle's speed is influenced by inertia, its individual best position, and the global best position. In this way, PSO is capable of finding optimal solutions that satisfy design requirements in complex parameter spaces.

The LLM has a basic understanding of algorithms and can independently generate and optimize algorithms [29,30]. Therefore, to further achieve automation and enhance the LLM's role in the optimization process, we delegate some of the PSO algorithm's hyperparameters to the LLM, including maximum iterations (max iter), population size (pop size), inertia weight (w), personal learning factor (c1), global learning factor (c2), and maximum velocity (max velocity rate). Typically, we recommend that the LLM use our commonly used values, namely a population size of 50, an inertia factor of 0.8, local and global learning factors of 1.5, and a maximum velocity of 0.2. At the same time, we use the minimization objective function to execute our PSO algorithm, and in each optimization result, the algorithm will select the group with the lowest score as the optimal solution. For example, when using the suggested parameters for experimentation, the particle swarm algorithm generates 50 different parameter groups based on the provided optimization parameter range. These 50 parameter groups will then respectively generate 50 corresponding S-parameter results through the ADS Interpreter. Next, we calculate each result according to the formula and pass the optimal parameter group and its score to the PSO algorithm, updating the speed and position of its internal particles. Through continuous iteration and updating, the particles eventually converge to a very small area, and the score calculation for the generated parameter group is 0, indicating that the optimization objective has been achieved. Algorithm 1 provides detailed information on the PSO algorithm. In the algorithm, the position of each particle corresponds to a set of circuit parameter values.

2.3. Prompt engineering

Prompt engineering is a key technique for guiding LLMs to complete specific tasks by setting scenarios, roles, and task objectives, enabling them to efficiently carry out design tasks. In this study, we adopted two

Algorithm 1 Particle Swarm Optimization

Input: Optimize parameter names and ranges, optimization objectives, netlist addresses, algorithm hyperparameters
Output: Optimal parameter set (global optimal position)
Initialization
- Initialize the positions and velocities of the particle swarm based on the given parameter ranges
- Initialize the individual best positions and the global best position
- Initialize the individual best fitness values and the global best fitness value

while stopping criteria are not met **do**
for each particle `in` swarm **do**
Calculate the fitness value of the current particle
Update circuit parameters based on the particle's position
Modify the netlist file
Call the simulation tool to perform simulation
Extract optimization target data from the simulation results
Calculate the fitness value (compute the loss function based on the optimization target)
if current fitness value is better than the individual best fitness value **then**
Update the individual best position
Update the individual best fitness value
end if
if current fitness value is better than the global best fitness value **then**
Update the global best position
Update the global best fitness value
end if
Update the particle's velocity and position according to the PSO update formulas
end for
end while

main prompting methods: the Chain of Thought (COT) method and the ReAct method, to achieve automation in RF functional unit design.

The core of the COT method lies in breaking down complex problems into smaller sub-problems, guiding the LLM to gradually complete tasks [31]. This method allows the LLM to better understand task requirements and its current progress, thus advancing the design process step by step. In the topology selection phase of this study, we utilized a method called zero-shot chain of thought, which has been proven to be simple yet very effective [32]. We provided the LLM with the prompt "Let's think step by step" and asked it to analyze user requirements progressively, matching suitable netlists from the topology library. We instructed the LLM to filter topologies that meet the frequency band and performance criteria and further refine the selection to ultimately present 3–4 optimal topologies for the user to choose from. This method not only improves task clarity but also enhances the LLM's execution ability in complex tasks.

The ReAct method, which combines reasoning and acting, allows the LLM to gather information and complete tasks through external tools [33]. Through the ReAct approach, the LLM can analyze problems through reasoning and use external tools in conjunction with the environment to assist in problem-solving. In this study, the ReAct method is primarily reflected in providing the LLM with tools for external interaction, including reading topology libraries and calling optimizers. When reading the topology library, the LLM retrieves different libraries based on specific needs, allowing it to select the correct topology netlist. Additionally, we equipped it with tools to extract local netlists and schematics, enabling it to display the selected netlists and schematics when requested by the user, further enhancing its interactivity with the user. In the optimization design phase, after considering the current design stage, the LLM decides when to call the optimizer and sets the optimization parameter ranges based on the optimizer's format to execute the optimization process.

In this study, prompt engineering is crucial for guiding the LLM in gradually achieving RF functional unit design. In addition to the aforementioned two prompting methods, we also provided the LLM with other general prompts, guiding it through the entire design process.

In the topology selection process, the LLM was cast in the role of an engineer with circuit design expertise. It was provided with a library of available topologies and their characteristics, and tasked with matching topologies based on user requirements. The LLM prioritizes the selection of topologies whose frequency bands and performance characteristics closely align with those of the original topology netlist. After presenting the recommended topologies, the LLM can further provide additional topologies or detailed performance information of a specific topology upon user request.

Before executing the optimization process, we instructed the LLM to generate a parameter extraction tool compatible with the current netlist. We also constrained its format and function names to facilitate the subsequent extraction and application of the function in the later stages. During the optimization process, we described the capabilities of the optimization tools to the LLM and guided their use, including setting metrics, hyperparameters, and ranges for optimization parameters. In addition, we specified the generation of optimization parameter ranges by the LLM. Drawing on our prior design experience, we provided a relatively simple default value, requiring the LLM to generate ranges that are approximately 1.5 times above and below the original parameters by default. This range is suitable for requirements where the performance gap with the original netlist is not significant. We also guided the LLM to display the modified parameters to the user and, if the user requests changes, to adjust the ranges based on the user's feedback. For the results summary module, we required it to analyze the degree to which criteria were met based on the optimization scores, backfill the optimized parameter set into the original netlist to generate a new one and present the performance of this final netlist.

3. Proposed method

3.1. LLM and GPT

Currently, there are many mainstream LLMs. Proprietary models include ERNIE Bot, GPT, and Claude, while open-source models include Qwen, LLaMA, and Gemma. Among these, GPT stands out due to its robust performance in various fields [34,35]. The usage of the GPT model is straightforward, and the GPT-4 series boasts strong reasoning capabilities, capable of addressing a wide range of problems [36]. Moreover, the GPT model has shown significant potential in the field of circuit design, with researchers already successfully designing chips through conversational interfaces [37]. Thus, we chose GPT as our assistant agent and selected the most advanced GPT-4o model to attempt the automation of the filter design process.

3.2. The bandpass filters

In RF circuits, filters selectively pass or suppress signals within a specific frequency range. Based on their operating principles, filters can be categorized into several types, including high-pass, low-pass, bandpass, and band-stop filters. Due to the clear performance metrics and relatively low design complexity, this study selects bandpass filters as the design target for GPT.

Fig. 5 shows the schematic of a bandpass filter from our topology library in ADS, exhibiting an overall symmetrical structure. It consists of multiple resonant circuits composed of capacitors and inductors, thereby achieving the filtering function. The TermG on the left and right sides of the circuit serve as the input and output ports, respectively, while BSV in our PDK acts as a via for grounding.

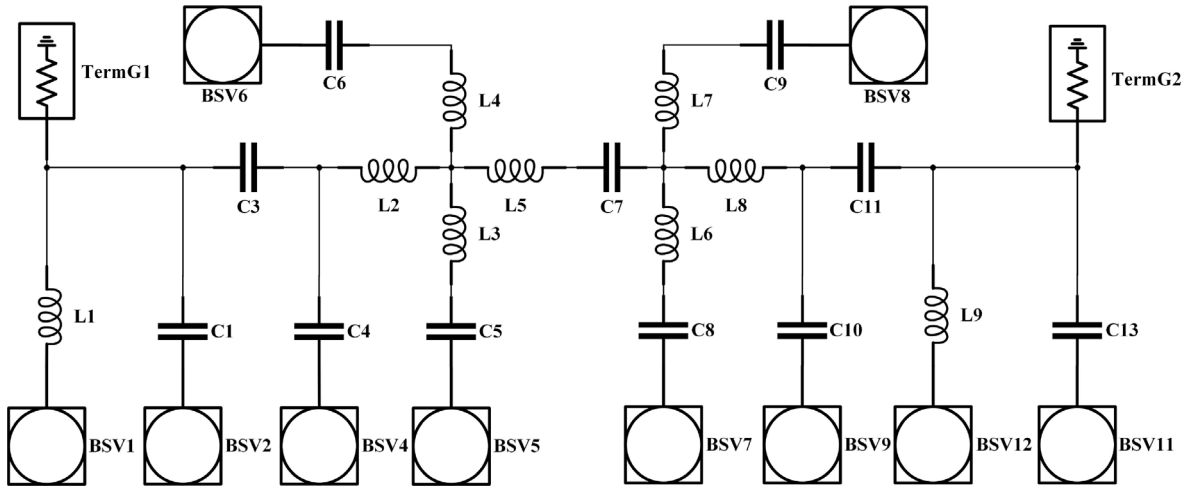

Fig. 5. Schematic of the band-pass filter.

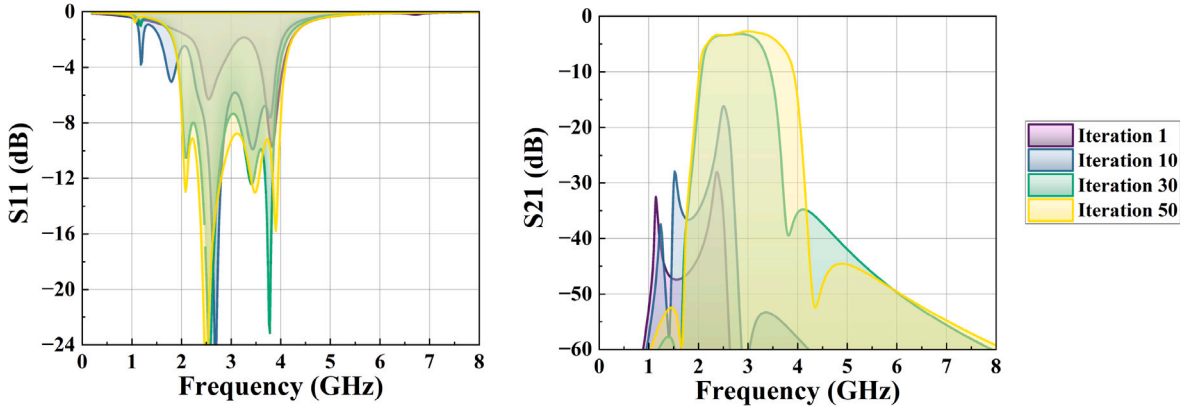

Fig. 6. Iterative process of filter optimization by the optimizer. Starting from the first generation, the performance of the bandpass filter gradually approaches the set target.

For filters, performance is typically evaluated using data in dB format. Since the results from the ADS simulator interface are provided as real and imaginary components, it is necessary to transform this data into dB format. The conversion formula is as follows:

$$\text{magnitude [dB]} = 20 \cdot \log_{10}\left(\sqrt{\text{Re}^2 + \text{Im}^2}\right) \tag{1}$$

In the formula, Re and Im represent the real and imaginary parts of the S-parameters, respectively. A two-port filter has four sets of S-parameters: S11, S12, S21, S22. In our design process, we performed comprehensive optimization across all these parameters, possibly leading to S11 and S22 not being perfectly equal. However, our primary objective is to achieve efficient signal transmission in a specific direction. Thus, we simplified our focus to primarily concentrate on S11 and S21. Here, S11 represents the reflection coefficient at Port 1 with Port 2 matched, and S21 represents the forward transmission coefficient from Port 1 to Port 2 with Port 2 matched.

4. Results and discussion

4.1. Optimization design of bandpass filter

We score results by calculating the area between the current result curve and the target. The score is zero for areas meeting the target; otherwise, the area between the result and target line segments is calculated and returned as the score. The area calculation formula is as follows:

$$\text{score} = \left| \int_a^b f(x)\,dx \right| \tag{2}$$

In the formula, a and b represent the start and end values of the frequency band, and f(x) is the S parameter curve within the required frequency range. If there are multiple objectives, we aggregate the scores for each objective by summing them. Several parameter sets equal to the population size are generated during each iteration of the PSO algorithm. When these parameters are applied to the netlist for simulation, scores are calculated using the above formula, returning a set of scores of equal quantity.

We tasked the LLM with designing a bandpass filter where S11 should be less than −8 dB within the 2–4 GHz range, and S21 should be less than −40 dB below 1.7 GHz and above 4.3 GHz. From the multiple topologies proposed by the LLM, we selected the structure mentioned above for optimization and allowed the LLM to set the optimization automatically ranges for capacitance and inductance. Fig. 6 illustrates the approximate performance changes of this filter during the optimization process. The figure shows that the S11 results consistently decrease with iterative steps, particularly with noticeable changes within the 2–4 GHz range. For S21, the passband gradually shifts towards the 2–4 GHz band during the optimization, with the 30th generation showing improved performance, eventually broadening the passband by the 50th generation to meet our specified requirements.

We conducted a simulation of the optimized bandpass filter with a sampling frequency of 0.02 GHz, and its performance is presented in

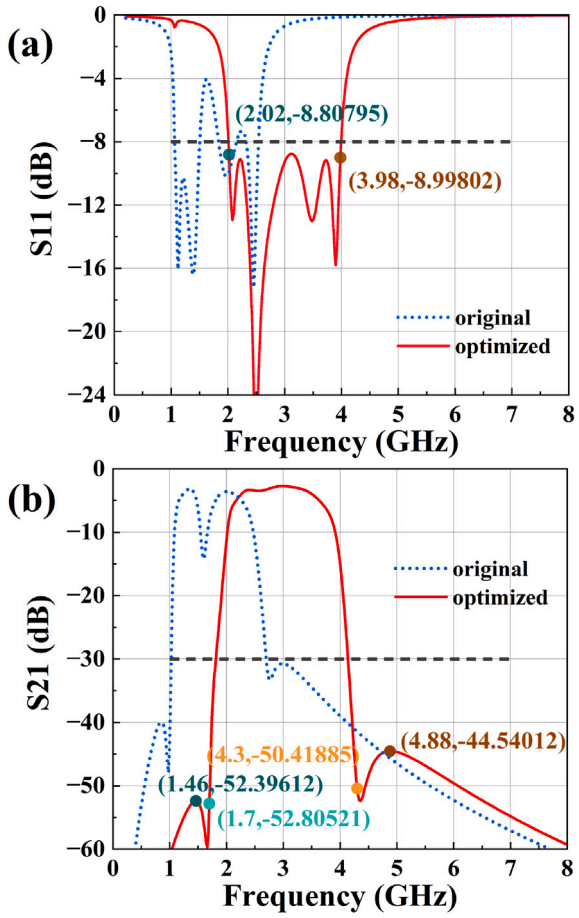

Fig. 7. The simulation results of the optimized netlist. (a) shows the S11 results, with the black line indicating the −8 dB threshold; (b) shows the S21 results, with the black line indicating the −30 dB threshold.

Fig. 7. The blue dotted line represents the initial performance of the bandpass filter, the red solid line indicates the optimized performance, and the black dashed line denotes the threshold specified by our requirements. As seen in Fig. 7(a), the filter's S11 remains below −8 dB, staying beneath the black dashed line within the 2.02–3.98 GHz range, demonstrating excellent input matching that aligns well with our performance criteria. In Fig. 7(b), the S21 value at 1.7 GHz and 4.3 GHz is less than −30 dB, with the out-of-band peaks on the left and right sides being −52.4 dB and −44.5 dB, respectively, both below the −30 dB dashed line. These results satisfy the requirement that S21 remain below −30 dB for frequencies less than 1.7 GHz and greater than 4.3 GHz, indicating that the LLM successfully fulfilled this design task.

Table 1 presents all optimization parameters used in this design, including their values before and after optimization, as well as the optimization range for each parameter determined by GPT.

4.2. Optimization runtime breakdown

After repeated testing and experimentation, we set the default number of iterations for the optimizer to 50. This is a relatively reasonable value, which can provide a balanced optimization effect and computational cost in most optimization scenarios. For more complex circuit designs, 50 iterations can explore the design space and find approximate optimal solutions. Additionally, we also set the default population size for PSO to an intermediate value, that is, 50 individuals.

In our previous optimization process, the most time-consuming part was calling the ADS simulator to perform simulations. In each iteration of the optimizer, there were 50 groups of optimization parameters, corresponding to 50 PCBs that needed to be simulated. This means that we need to perform 2500 simulations after completing the entire optimization process, which would undoubtedly consume a lot of time. According to our server's benchmarking, it takes approximately 0.8–1.1 s to simulate a filter PCB under an idle situation. However, if other programs occupy the CPU, the simulation time may be extended to 1.2 s or even 1.5 s.

We have performed statistics on the runtime of each step in the optimization process, and the specific situation is shown in the Table 2.

During the entire optimization process, the simulation time occupies more than 99% of the overall optimization time. In the remaining steps, the time-consuming part is the processing time for each set of simulation data, which accounts for approximately 0.58% of the optimization time. The time consumption for the remaining steps is very small.

Table 1
Optimization parameters and ranges used in this design.

Parameter	Value		Upper	Lower
	Before-opt	After-opt		
L_9 (nH)	3.16	1.56	0.3	6.33
L_8 (nH)	1.44	1.17	0.1	2.88
L_7 (nH)	3.5	1.35	0.35	7
L_6 (nH)	3.5	1.32	0.35	7
L_5 (nH)	1.1	1.47	0.1	2.21
L_4 (nH)	1.17	1.51	0.1	2.35
L_3 (nH)	3.5	3.93	0.35	7
L_2 (nH)	1.44	0.38	0.1	2.88
L_1 (nH)	3.16	1.44	0.3	6.33
C_{13} (fF)	1139	1635	100	2279
C_{12} (fF)	3204	2402	300	6409
C_{10} (fF)	2846	2147	300	5693
C_9 (fF)	2846	1059	280	14770
C_8 (fF)	7385	8936	700	14770
C_7 (fF)	7129	1810	800	16259
C_6 (fF)	2846	5584	280	5693
C_5 (fF)	2846	8722	700	14770
C_4 (fF)	3204	2319	300	6409
C_3 (fF)	3204	2014	300	6409
C_1 (fF)	1139	1492	100	2279

Table 2
Statistics on the duration of each step in the optimization process.

Step	Time cost (s)	Proportion
Netlist simulation	2636.22	99.3757%
Data processing	15.62	0.5888%
Score calculation	0.13	0.0049%
Parameter update	0.5	0.0188%
Others	0.31	0.0117%
Total time consumption	2652.78	

4.3. Comparison of different algorithms

To comprehensively evaluate the performance of the PSO algorithm in circuit optimization, we have compared it with other several common optimization algorithms, including the differential evolution algorithm (DE) and the non-dominated sorting genetic algorithm II (NSGAII) [8]. The differential evolution algorithm is a population-based heuristic random search algorithm that uses the difference information between individuals in the population to generate new solutions, thereby achieving global optimization of the search space. The non-dominated sorting genetic algorithm II is a classic multi-objective optimization algorithm that is mainly used to solve optimization problems with multiple conflicting objectives. By using non-dominated sorting and crowding factor calculation, it can optimize multiple objectives.

Fig. 8 shows the comparison of the three algorithms' error reduction over the 50 iterations process.

Table 3
Experimental results across different metrics.

Target	S11 requirement	S21 requirement	Score	Completion
1	In-band(0.8–1.5 GHz) < −10 dB	Out-band(<0.5 GHz, >1.8 GHz) < −30 dB	0.0	✓
2	In-band(1–2 GHz) < −15 dB	Out-band(<0.8 GHz, >2.2 GHz) < −40 dB	0.27	✓
3	In-band(1–2 GHz) < −10 dB	Out-band(<0.8 GHz, >2.2 GHz) < −30 dB	0.93	✓
4	In-band(1.3–2.3 GHz) < −10 dB	Out-band(<1 GHz, >2.6 GHz) < −30 dB	0.0	✓
5	In-band(1.3–2.3 GHz) < −10 dB	Out-band(<1 GHz, >2.6 GHz) < −40 dB	0.0	✓
6	In-band(2–3.5 GHz) < −15 dB	Out-band(<1.7 GHz, >3.8 GHz) < −40 dB	0.0	✓

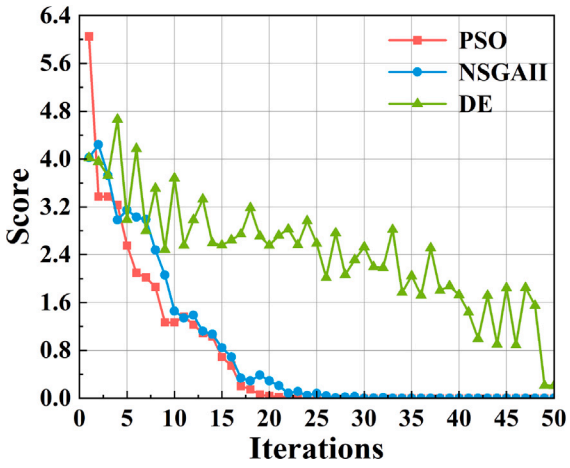

Fig. 8. Comparison of different algorithm optimizations. In the case of the same goal, there is a certain difference in the decline of scores among the three different algorithms during 50 iterations.

In the case where a smaller score indicates smaller error, it can be seen from the figure that PSO's first-generation error is relatively large, but its error reduction speed is very fast, and it is reduced to 0 at the 23rd generation. Although NSGA2's first-generation error is smaller than PSO's, its error reduction speed is slightly inferior to PSO's, and it is reduced to 0 at the 33rd generation. DE's reduction speed is relatively slow, and even at the 50th generation, the score is still around 0.2, failing to achieve perfect optimization.

4.4. Additional filter optimization attempts

To verify this automated design process's reliability, we conducted multiple designs using this topology. The performance metrics and optimization scores are presented in the Table 3.

We conducted simulations and plotted the results for the optimized filter based on the targets outlined in the table, as shown in Fig. 9. For the S11 parameter of this bandpass filter, we require it to be below a specific value within the designated frequency band to achieve higher transmission efficiency. For the S21 parameter, we require the out-of-band suppression to be below a certain value to enhance signal quality.

Fig. 9(a) shows the optimization results for Target 1. The designed filter's S11 is less than −10 dB within the 0.8–1.5 GHz range, and the S21 is below −30 dB outside the band, indicating successful design according to these criteria. Fig. 9(b) presents the optimization results for Target 2. Within the 1–2 GHz range, the filter's S11 slightly exceeds −8 dB at 1.9 GHz, but the S21 meets the out-of-band requirement of being below −30 dB, making the design largely successful. Fig. 9(c) illustrates the optimization results for Target 3. In the 1–2 GHz range, the maximum S11 value approaches −15 dB at 1.2 GHz yet remains compliant. The S21 slightly exceeds −40 dB at the left extremum of 2.5 GHz but meets requirements elsewhere, marking the design as generally successful. Fig. 9(d) displays the optimization results for Target 4. The filter's S11 meets the criterion of being below −10 dB within the 1.3–2.3 GHz range, and the out-of-band S21 is below −30 dB, fully conforming to the specifications. Fig. 9(e) shows the optimization results for Target 5. The filter's S11 is entirely below −10 dB within the 1.3–2.3 GHz range, and the out-of-band S21 is below −40 dB, indicating the complete success of the design. Fig. 9(f) presents the optimization results for Target 6, where the filter's S11 is below −10 dB across the 1.5–2.5 GHz band, and the out-of-band S21 fully meets the criterion of being below −30 dB, marking the design as entirely successful.

5. Conclusion

We propose an automated circuit design method based on LLMs. This approach includes a topology selection tool and a PSO netlist optimizer for the LLM, enhanced by prompt engineering to provide decision-making and tool invocation capabilities. The LLM emulates the operational workflow of engineers, automatically selecting suitable topologies based on requirements, setting optimizer parameters, designing the netlist, and summarizing the output. Through this method, the LLM can achieve the preliminary design of a bandpass filter, with the design results generally meeting performance requirements. To some extent, this approach has enabled the application of LLMs in the RFIC domain, offering more possibilities for future AI development in this field. In future research, we plan to incorporate additional optimization algorithms to provide the LLM with more options for selection. We will also apply this methodology to design more complex RF functional units, such as power amplifiers and low-noise amplifiers. Furthermore, we will continue to improve our optimizers and expand our topology library to meet a broader range of design requirements and to make them suitable for more complex circuit systems.

CRediT authorship contribution statement

Hangjiang Jin: Writing – original draft, Methodology, Software. **Junchao Wang:** Conceptualization, Writing – review & editing. **Junjie Sheng:** Data curation. **Yifan Wu:** Writing – review & editing. **Jiayu Chen:** Writing – review & editing. **Yaqi Wang:** Supervision, Writing – review & editing. **Jun Liu:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by the National Natural Science Foundation of China, No. 62206081.

Data availability

Data will be made available on request.

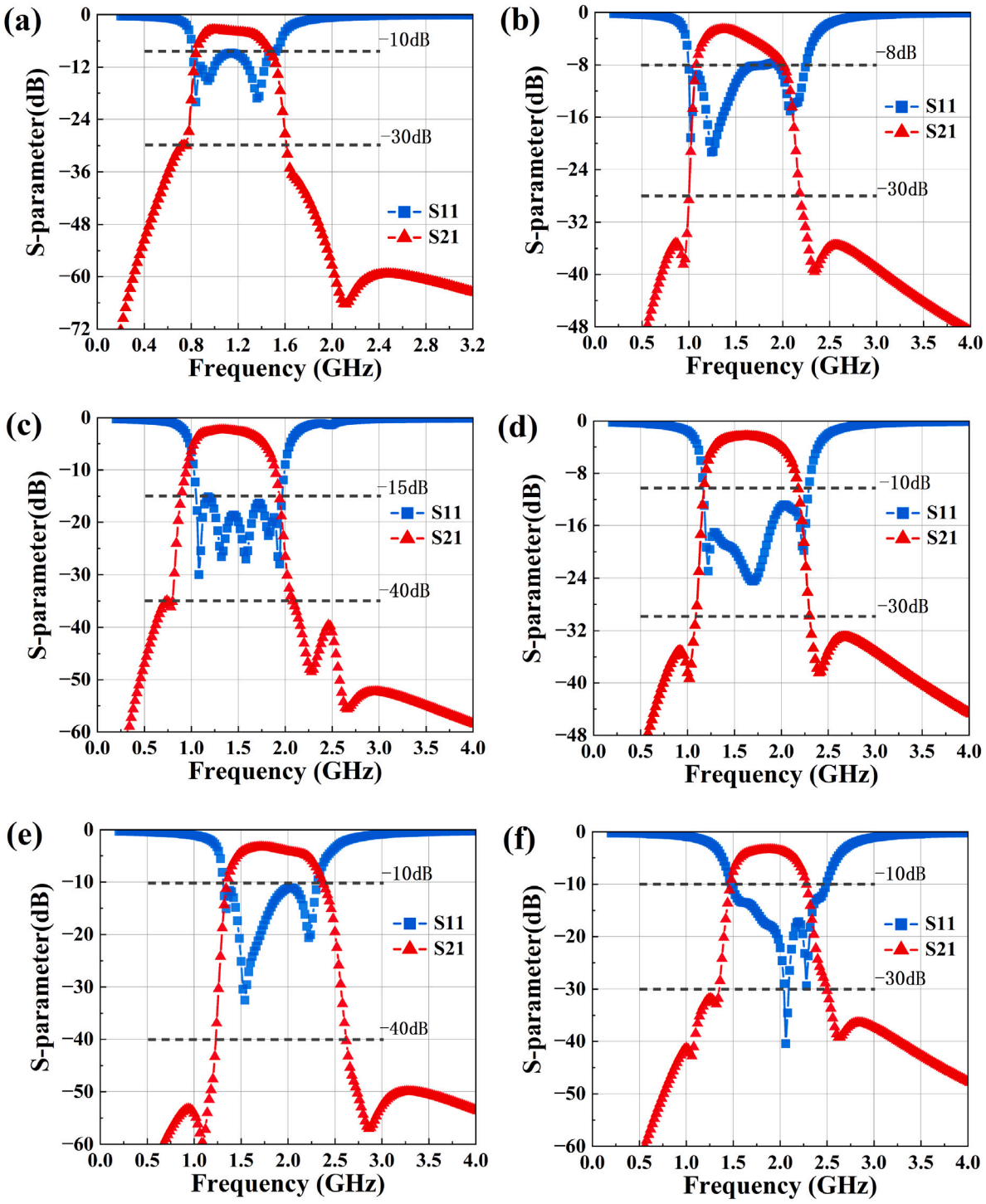

Fig. 9. Simulation results for designs. Figures (a) to (f) correspond to the design outcomes for Targets 1 through 6, as listed in the table.

References

[1] Abdelaziz Lberni, Malika Alami Marktani, Abdelaziz Ahaitouf, Ali Ahaitouf, Efficient butterfly inspired optimization algorithm for analog circuits design, Microelectron. J. 113 (2021) 105078.

[2] Qi-Jun Zhang, Kuldip C. Gupta, Vijay K. Devabhaktuni, Artificial neural networks for RF and microwave design-from theory to practice, IEEE Trans. Microw. Theory Tech. 51 (4) (2003) 1339–1350.

[3] Vijaya Kumar Devabhaktuni, Mustapha CE Yagoub, Yonghua Fang, Jianjun Xu, Qi-Jun Zhang, Neural networks for microwave modeling: Model development issues and nonlinear modeling techniques, Int. J. RF Microw. Comput.-Aided Eng.: Co-Spons. Cent. Adv. Manuf. Packag. Microw. Opt. Digit. Electron. (CAMPmode) Univ. Color. Boulder 11 (1) (2001) 4–21.

[4] Rui Cheng, Lin-Zi Yin, Zhao-Hui Jiang, Xue-Mei Xu, Gate-level circuit partitioning algorithm based on clustering and an improved genetic algorithm, Entropy 25 (4) (2023) 597.

[5] Dmitry Bulakh, Sergey Zhestkov, Pavel Volobuev, A pattern-based algorithm for transistor-level combinational circuits netlists visualization, in: 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus, IEEE, 2019, pp. 2194–2197.

[6] Engin Afacan, Nuno Lourenço, Ricardo Martins, Günhan Dündar, Machine learning techniques in analog/RF integrated circuit design, synthesis, layout, and test, Integration 77 (2021) 113–130.

[7] Arijit Majumdar, Soumyo Chatterjee, Sayan Chatterjee, Sheli Sinha Chaudhari, Dipak Ranjan Poddar, Optimization of intrinsic elements from small signal model

of GaN HEMT by using PSO, in: 2015 IEEE Applied Electromagnetics Conference, AEMC, IEEE, 2015, pp. 1–2.
[8] Dongyu Zhang, Hongliang Lv, Silu Yan, Yanghui Hu, Qijun Zhang, Chao Han, Ranran Zhao, Yuming Zhang, Multi-objective neural network modeling and applications to microwave power amplifiers, Microelectron. J. 149 (2024) 106244.
[9] Xianming Liu, Shihong Wu, Wenrun Xiao, Chenhui Zhao, Chao Huang, Donghui Guo, Performance improvement for the CMOS rail-to-rail amplifier via APSO-based design and SNN's training, Microelectron. J. 146 (2024) 106131.
[10] KG Shreeharsha, RK Siddharth, Charudatta G Korde, MH Vasantha, Nithin Kumar YB, An exponential variation based PSO for analog circuit sizing in constrained environment, AEU-Int. J. Electron. Commun. 187 (2024) 155531.
[11] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al., Emergent abilities of large language models, 2022, arXiv preprint arXiv:2206.07682.
[12] Yao Fu, Hao Peng, Tushar Khot, How does gpt obtain its ability? Tracing emergent abilities of language models to their sources, Yao Fu's Notion (2022).
[13] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al., A survey of large language models, 2023, arXiv preprint arXiv:2303.18223.
[14] Tyler A. Chang, Benjamin K. Bergen, Language model behavior: A comprehensive survey, Comput. Linguist. 50 (1) (2024) 293–350.
[15] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al., A survey on large language model based autonomous agents, Front. Comput. Sci. 18 (6) (2024) 186345.
[16] Hui Yang, Sifu Yue, Yunzhong He, Auto-gpt for online decision making: Benchmarks and additional opinions, 2023, arXiv preprint arXiv:2306.02224.
[17] Daniil A. Boiko, Robert MacKnight, Ben Kline, Gabe Gomes, Autonomous chemical research with large language models, Nature 624 (7992) (2023) 570–578.
[18] Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, Dongmei Zhang, Demonstration of InsightPilot: An LLM-empowered automated data exploration system, 2023, arXiv preprint arXiv:2304.00477.
[19] Lei Chen, Yiqi Chen, Zhufei Chu, Wenji Fang, Tsung-Yi Ho, Yu Huang, Sadaf Khan, Min Li, Xingquan Li, Yun Liang, et al., The dawn of ai-native eda: Promises and challenges of large circuit models, 2024, arXiv preprint arXiv:2403.07257.
[20] Boyu Han, Xinyu Wang, Yifan Wang, Junyu Yan, Yidong Tian, New interaction paradigm for complex eda software leveraging gpt, 2023, arXiv preprint arXiv:2307.14740.
[21] Haoyuan Wu, Zhuolun He, Xinyun Zhang, Xufeng Yao, Su Zheng, Haisheng Zheng, Bei Yu, Chateda: A large language model powered autonomous agent for eda, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2024).
[22] Deepak Vungarala, Sakila Alam, Arnob Ghosh, Shaahin Angizi, SPICEPilot: Navigating SPICE code generation and simulation with AI guidance, 2024, arXiv preprint arXiv:2410.20553.
[23] Jindong Gu, Zhen Han, Shuo Chen, Ahmad Beirami, Bailan He, Gengyuan Zhang, Ruotong Liao, Yao Qin, Volker Tresp, Philip Torr, A systematic survey of prompt engineering on vision-language foundation models, 2023, arXiv preprint arXiv:2307.12980.
[24] Sabit Ekin, Prompt engineering for ChatGPT: a quick guide to techniques, tips, and best practices, Authorea Prepr. (2023).
[25] Subhash Patel, Rajesh A. Thakker, Automatic circuit design and optimization using modified pso algorithm, J. Eng. Sci. Technol. Rev. 9 (1) (2016).
[26] Yanning Chen, Dongyan Zhao, Fang Liu, Jie Gao, Hui Zhu, Thermal layout optimization for 3D stacked multichip modules, Microelectron. J. 139 (2023) 105882.
[27] Haiyi Cai, Jincan Zhang, Min Liu, Shi Yang, Shaowei Wang, Bo Liu, Juwei Zhang, Adaptive particle swarm optimization based hybrid small-signal modeling of GaN HEMT, Microelectron. J. 137 (2023) 105834.
[28] Ria Rashid, Nandakumar Nambath, Area optimisation of two stage miller compensated op-amp in 65 nm using hybrid PSO, IEEE Trans. Circuits Syst. II: Express Briefs 69 (1) (2021) 199–203.
[29] Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, Qingfu Zhang, Evolution of heuristics: Towards efficient automatic algorithm design using large language model, in: Forty-First International Conference on Machine Learning, 2024.
[30] Niki van Stein, Diederick Vermetten, Thomas Bäck, In-the-loop hyper-parameter optimization for LLM-based automated design of heuristics, 2024, arXiv preprint arXiv:2410.16309.
[31] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, Adv. Neural Inf. Process. Syst. 35 (2022) 24824–24837.
[32] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa, Large language models are zero-shot reasoners, Adv. Neural Inf. Process. Syst. 35 (2022) 22199–22213.
[33] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, Yuan Cao, React: Synergizing reasoning and acting in language models, 2022, arXiv preprint arXiv:2210.03629.
[34] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al., Gpt-4 technical report, 2023, arXiv preprint arXiv:2303.08774.
[35] Katikapalli Subramanyam Kalyan, A survey of GPT-3 family large language models including ChatGPT and GPT-4, Nat. Lang. Process. J. (2023) 100048.
[36] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al., Sparks of artificial general intelligence: Early experiments with gpt-4, 2023, arXiv preprint arXiv:2303.12712.
[37] Jason Blocklove, Siddharth Garg, Ramesh Karri, Hammond Pearce, Chip-chat: Challenges and opportunities in conversational hardware design, in: 2023 ACM/IEEE 5th Workshop on Machine Learning for CAD, MLCAD, IEEE, 2023, pp. 1–6.