Knowing The Spec to Explore The Design via Transformed Bayesian Optimization

Donger Luo*
ShanghaiTech University

Qi Sun*
Zhejiang University

Xinheng Li
ShanghaiTech University

Chen Bai
Chinese University of Hong Kong

Bei Yu
Chinese University of Hong Kong

Hao Geng†
ShanghaiTech University

Abstract

AI chip scales expediently in the large language models (LLMs) era. In contrast, the existing chip design space exploration (DSE) methods, aimed at discovering optimal yet often infeasible or unproduceable Pareto-front designs, are hindered by neglect of design specifications. In this paper, we propose a novel Spec-driven transformed Bayesian optimization framework to find expected optimal RISC-V SoC architecture designs for LLM tasks. The highlights of our framework lie in a tailored transformed Gaussian process (GP) model prioritizing specified target metrics and a customized acquisition function (EHRM) in multi-objective optimization. Extensive experiments on large-scale RISC-V SoC architecture design explorations for LLMs, such as Transformer, BERT, and GPT-1, demonstrate that our method not only can effectively find the design according to QoR values from the spec, but also outperforms 34.59% in ADRS over state-of-the-art approach with only 66.67% runtime overhead.

1 Introduction

The astonishing advancement of artificial intelligence (AI) has brought about tremendous changes in computing and associated hardware design paradigms. In particular, generative AI and large language models (LLMs), such as the OpenAI GPT family, which kindled the entire academia and industry, require a large amount of computation for training and deployment. This has led to an explosive increase in demand for AI hardware computility.

Efficient inference of large models requires customized accelerators and System-on-Chip (SoC) designs to reach the satisfying quality of result (QoR) metrics. However, such design is highly complex and resource-intensive, involving collaborative efforts from a sizeable team to systematically address issues and progressively refine the design across several iterations. This dramatically increases costs and time, with an initial design phase spanning 1 to 2 years, followed by several years of optimization. Consequently, agile development is gradually being adopted to reduce chip design costs and

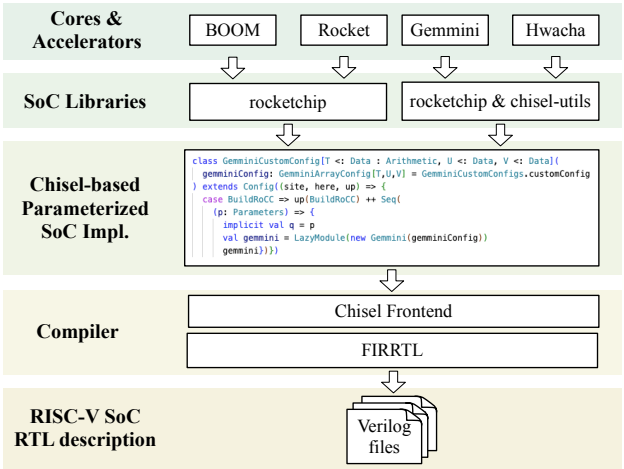


Figure 1: Flow of agile chip development. Processor cores, like BOOM and Rocket, and accelerators, such as Gemmini and Hwacha, are described in parameterized designs based on certain libraries. After that, the compiler takes in Chisel files and generates the SoC RTL description.

accelerate design cycles. Chisel [1], as shown in Figure 1, is a meta-programming, object-oriented, and parameterized hardware description language, allowing for the agile integration of complex hardware designs. Anchored on Chisel, Rocket Core [2] and Berkeley Out-of-Order Machine (BOOM) [3] are *configurable* and *parameterizable* RISC-V processors. Similarly, Gemmini [4] is a systolic array-based DNN accelerator. Integrating the Rocket Core, BOOM, Gemmini, *etc.*, Chipyard [5] is a design, simulation, and implementation framework for custom SoC which has been adopted for LLM inferences. Despite the numerous conveniences afforded by the adoption of configurable design approaches, their optimization remains challenging in the huge parameterized design space. Evaluating each set of parameters may take weeks since the EDA flow is highly time-consuming. Even worse, there is often interdependence or contention between parameters while the multiple design objectives may be conflicting. To tackle the problem, some approaches have been proposed for exploring the design space efficiently. In [6], artificial neural networks are leveraged to describe relationships among design parameters and produce performance estimates for the designs. Lee *et al.* [7] propose a non-linear regression model for exploration in multiprocessor micro-architecture. ELSE [8] combines different regression models for robust design space modeling of microprocessors. Li *et al.* [9] explore the design space by exploiting AdaBoost.

The design data are expensive and scarce, insufficient for the aforementioned methods. Bayesian optimization (BO) method, which

*Equal contributors
†Corresponding author, also with Shanghai Engineering Research Center of Energy Efficient and Custom AI IC.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3658262

performs efficient global searches to explore the multi-objective Pareto-optimal designs, is promising for such a time-consuming problem. Zhang *et al.* [10] propose to use BO with neural networks for analog circuit synthesis. Ma *et al.* [11] use BO to solve the parameter selection problem for EDA tools. BOOM-Explorer [12] uses a deep kernel learning-based BO method to optimize the power, performance, and area of the BOOM core.

However, these works ignore the critical fact that the Pareto-optimal designs are quite likely to be unknown, cost-unacceptable, and unreachable in practice. Typically, anticipated Quality of Results (QoR) metrics are delineated within the design specification (*abbr.*, spec) prior to initiating the design process, transforming the search for a design that meets these QoR values into a guided endeavor. As extra information, the given spec is underutilized in previous works. We propose a transformed Bayesian optimization-based framework to efficiently find the parameterized SoC architecture that meets the design spec. A transformed Gaussian process model is developed to integrate the QoR metric values given in the spec so that enables efficient exploration. An acquisition function, EHRM, is well designed to enable multi-objective optimization and guide the search based on the transformed model.

Our main contributions are as follows.

- To the best of our knowledge, it is the first exploration framework to efficiently find the parameterized SoC architectures with QoR metric values given in the spec.
- We construct a transformed Gaussian process model that utilizes the information of the given spec.
- A customized acquisition function EHRM is developed to enable multi-objective optimization based on the transformed model.
- We conduct comprehensive experiments on the design space exploration for parameterized RISC-V SoC, benchmarked on some popular LLMs. Experimental results demonstrate the high efficacy and quality of our method, outperforming the previous arts significantly.

The rest of the paper is organized as follows. Section 2 introduces prior knowledge of RISC-V SoC and multi-objective Bayesian optimization and proposes a problem formulation. Section 3 sketches the whole optimization flow. Section 4 describes the technical details of the transformed Bayesian optimization, while Section 5 presents the experimental results followed by a conclusion in Section 6.

2 Preliminaries

2.1 RISC-V SoC and LLM Acceleration

RISC-V SoC: The RISC-V architecture, a reduced instruction set computer architecture, offers efficient hardware design and implementation with a minimal instruction set [13], and has been more and more popular in recent years for its outstanding power efficiency. A notable RISC-V SoC framework is Chipyard [5], an open-source platform enabling SoC customization, integrating various RISC-V CPUs and accelerators, such as BOOM [3], Rocket Core [2], Gemmini [4], and Hwacha [14]. A typical RISC-V SoC orchestrated using Chipyard is shown in Figure 2, integrating a Gemmini accelerator, a RISC-V CPU, an L2 Cache, and DRAM. The Gemmini accelerator comprises a configurable systolic array arranged as tiles of PEs. The scratchpad is the in-accelerator storage made up of configurable

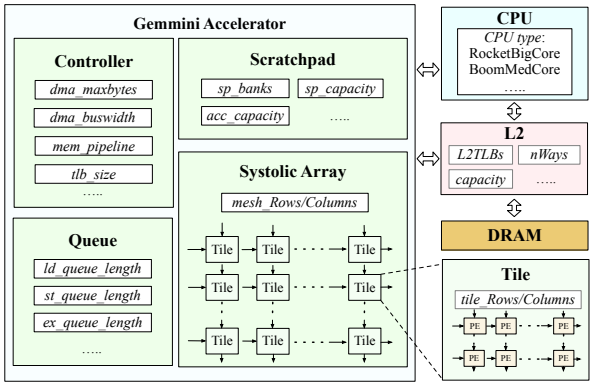


Figure 2: The architecture of RISC-V SoC. It is generated from the highly parameterized Chisel files describing the architecture configurations.

SRAMs. The controller is responsible for scheduling, sending, and receiving the data from the external DRAM with the DMA channel. Queue is implemented for access-execute decoupling, which contains the load, store, and execute instruction queues. The framework can be configured to use one or more RISC-V synthesizable CPUs. The CPUs communicate with accelerators by sending commands to the Rocket custom co-processor (RoCC), a component that can be added to the CPU for communication support. Several configurable hierarchical caches can be added, including L1 and L2 caches.

LLM Acceleration: The RISC-V SoC has demonstrated remarkable capabilities in the acceleration of convolution networks, supporting a variety of complex model computations and requirements [4]. Divergent from traditional convolutional networks, LLM tasks exhibit a more regularized or uniform model structure, with large-scale matrix multiplication constituting the primary module and performance bottleneck. To adapt the architecture depicted in Figure 2 for LLM tasks, numerous factors must be considered. These include the selection of an appropriate CPU, cache sizes, accelerator configurations, *etc.*, to achieve a synergistic optimization of performance, area, and power consumption, thereby fulfilling the design specification.

2.2 Multi-objective Bayesian Optimization

Bayesian optimization (BO) [15] is a sequential design strategy for the global optimization of noisy black-box functions. It builds surrogate models to mimic the unknown black-box objective functions, *i.e.*, the complicated SoC architecture in our context. The Gaussian process (GP) model is widely used as the surrogate model, which provides a posterior distribution of functions given prior and observed data.

Real-world optimization problems often involve multiple conflicting objectives. Multi-objective Bayesian optimization (MOBO) extends the standard BO to tackle such multi-objective problems. Let functions $\{f_i(\boldsymbol{x})\}_{i=1}^{m}$ denote the m-dimension QoR metrics to be minimized and $\boldsymbol{X}$ denotes the parameter space. A parameter vector $\boldsymbol{x}^* \in \boldsymbol{X}$ is said to dominate $\boldsymbol{x}' \in \boldsymbol{X}$, denoted as $\boldsymbol{x}^* \succeq \boldsymbol{x}'$, if the following two conditions are met in this minimization problem:

(1) $f_i(\boldsymbol{x}') \geq f_i(\boldsymbol{x}^*), \forall i \in \{1, ..., m\}$,
(2) There exists at least one j such that $f_j(\boldsymbol{x}') > f_j(\boldsymbol{x}^*)$.

The set of parameter vectors that are not dominated by other vectors is called the Pareto-optimal set, denoted as $\boldsymbol{X}_{\text{Pareto}}$, which is

the target of MOBO:

$$X_{\text{Pareto}} = \{x^* \in X | \nexists x' \in X, x' \succeq x^*\}. \quad (1)$$

If $y^* \succeq y'$, it means that (x^*, y^*) dominate (x', y'). For the problem with m optimization objectives, Bayesian optimization builds m surrogate models for these objectives, denoted as M, to mimic the unknown black-box objective functions. Further, an acquisition function $Ac(M, x)$ is built based on the surrogate models and estimates the quality of a parameter configuration x with respect to finding the Pareto set, without going through the time-consuming EDA flow.

2.3 Problem Formulation

Definition 1 (Hypervolume)**.** In multi-objective problems, the notion of solution is that of Pareto fronts. The quality of such fronts can be measured by hypervolume:

$$\text{HV}\left(f^{ref}, \text{X}\right) = \Lambda\left(\bigcup_{\text{X}_n \in \text{X}} \left[f_1\left(\text{X}_n\right), f_1^{ref}\right] \times \cdots \times \left[f_m\left(\text{X}_n\right), f_m^{ref}\right]\right), \quad (2)$$

where f^{ref} refers to a chosen reference point, generally, it is set as a very bad performance value point. f_m means the m th metric i.e. cycles, power and area. And Λ refers to the Lebesgue measure.

Definition 2 (ADRS)**.** Another metric to measure the quality of Pareto fronts is the average distance to reference set (ADRS):

$$\text{ADRS}(\Gamma, \Omega) = \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} \min_{\omega \in \Omega} D(\gamma, \omega), \quad (3)$$

where D is the Euclidean distance function. Γ is the real Pareto-optimal set and Ω is the learned Pareto-optimal set. ADRS is often exploited to measure how close a learned Pareto-optimal set is to the real Pareto-optimal set of the design space.

In industrial scenarios, a set of expected values of QoR metrics are usually provided by specifications in advance. With the above knowledge, our problem can be formulated as follows.

Problem 1 (Spec-driven SoC design space exploration)**.** Given the expected value of QoR metrics from specifications, the objective of SoC architecture design space exploration is to automatically search parameterized designs whose associated performance is closest to expected metric values.

3 Overall Flow

In this section, we introduce a multi-objective Bayesian optimization approach. The primary goal of the approach is to optimize the parameterized SoC architecture to meet the expected metrics.

The overall architecture of our approach is shown in Figure 3. Chipyard takes in the suggested parameterized SoC architecture as Chisel files and generates corresponding Verilog files. Then, performance information is generated using the simulator. After the Verilog files are synthesized with EDA tools, power and area information can be read from report files. The QoR metrics, including performance, power, and area, are appended to the data Y, along with the parameterized SoC architecture as X. A transformed Gaussian process regression model which utilizes the expected value from the spec is built for each metric. These models offer mean and variance

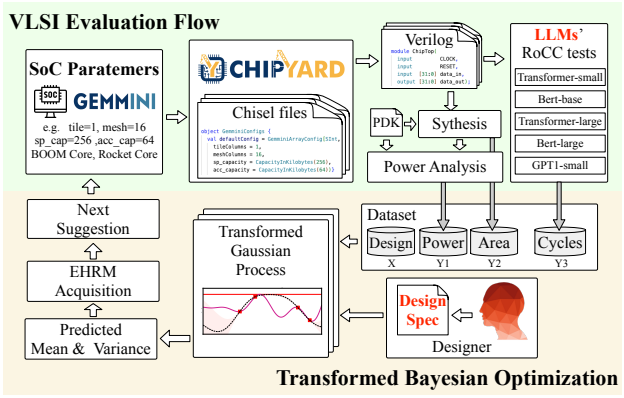


Figure 3: Overall flow. The whole flow is separated into two parts. The green part in the figure takes in the suggestion and evaluates it through the simulator and EDA tools. After that, QoR metrics are collected in the yellow part to generate the next suggestion through transformed Bayesian optimization.

Algorithm 1 BO Approach with Given Spec

Input: Spec's metrics values f^*, parameter search space X, acquisition function AC, simulation tool $\mathbb{SIM}$, EDA tool $\mathbb{EDA}$, optimization budget t_{max}.
Output: The set of solutions closest to spec value X_{spec}.

1: **Initialization**: sample initial input X_0 with $X_0 \subset X$, $Y_0 = [\mathbb{SIM}(X_0), \mathbb{EDA}(X_0)]$, $D_0 = \{X_0, Y_0\}$;
2: **for** $t \leftarrow 1$ to t_{max} **do**
3: Build m transformed Gaussian process models: $M_{t;1,\ldots,m} \leftarrow \mathcal{GP}(f^*, X_{t-1}, Y_{t-1})$; ▷ *Equation* (7)
4: Select the next candidate to evaluate: $X_t \leftarrow \text{AC}(f^*, M_{t;1,\ldots,m}, x), \forall x \in X$; ▷ *Equation* (12)
5: Evaluate next candidate $Y_t = [\mathbb{SIM}(X_t), \mathbb{EDA}(X_t)]$;
6: Update $D_t \leftarrow \{X_t, Y_t\}$;
7: **end for**
8: Select ideal configurations X_{spec} according to $D_{t_{max}}$;
9: **return** X_{spec}.

information for the acquisition function to select the next set of parameterized SoC architecture.

Algorithm 1 outlines the steps taken in each iteration of the optimization process. In line 1, the algorithm begins by randomly sampling the initial parameter configurations X_0 from the parameter search space. X_0 is then simulated and synthesized using the simulator and EDA tools to get the corresponding performance, power, and area values. For simplicity, we denote the simulator as $\mathbb{SIM}$ and EDA tools as $\mathbb{EDA}$. In each iteration (line 2 to 7), a transformed GP model, which takes in the knowledge of expected value f^* in spec, is built for each metric (line 3). A corresponding acquisition function, which also needs f^*, selects the next candidate to be evaluated (line 4). After that candidate is simulated and synthesized, the results are added to the sample set D for the next iteration (line 5 to 6). This process is repeated until the maximum number of iterations is reached. Finally, the algorithm returns the parameterized SoC architecture closest to the given spec.

In a nutshell, the proposed algorithm takes in the knowledge of the spec to explore and exploit the SoC architecture design space

using the Gaussian process model, thereby providing a satisfactory SoC architecture.

4 Bayesian Optimization With Given Spec

As aforementioned, each parameter in a parameterized SoC architecture has its elusive effect on metrics like performance, power, and area. Some parameters may conflict, making it even harder to pick an ideal SoC architecture from the design space. This section details the methodology we employed to carry out Bayesian optimization (BO) for optimizing QoR with its values given in spec. We aim to utilize the knowledge of that given value f^* in spec to find a corresponding SoC architecture quickly.

4.1 Transformed Gaussian process

The key idea of the transformation is that the function value $f(x)$ should reach the value f^* in spec but does not need to be greater than f^*. Based on the idea, the Gaussian process is transformed as:

$$f(\mathbf{x}) = f^* - \frac{1}{2}g^2(\mathbf{x}) \quad g(\mathbf{x}) \sim GP\left(m_0, K\right), \tag{4}$$

where m_0 and K are the prior mean and covariance of $g(x)$ respectively, so $f(x)$ will not beyond the given spec's value f^*. Doing so can save lots of effort from the attempt to go greater than f^*. Given a set of single-objective observations $\boldsymbol{D} = (\boldsymbol{x}_i, y_i)_{i=1}^N$, we can compute the observations as: $\boldsymbol{D_g} = (\boldsymbol{x}_i, g_i)_{i=1}^N$, where $g_i = \sqrt{2\left(f^* - y_i\right)}$. Then the posterior distribution of the transformed Gaussian process can be described as:

$$p\left(g \mid \boldsymbol{D}_g, f^*\right) \sim \mathcal{N}\left(\mu_g(\mathbf{x}), \sigma_g(\mathbf{x})\right), \tag{5}$$

However, the posterior distribution becomes a non-central process because of the transformation. An approximation technique is performed to overcome the problem. A local-linearization of the transformation $t(\mathbf{x}) = f^* - \frac{1}{2}g^2(\mathbf{x})$ is taken around g_0 and f is obtained as:

$$f \approx t\left(g_0\right) + t'\left(g_0\right)\left(g - g_0\right), \tag{6}$$

where the gradient $t'\left(g_0\right)$ equals $-g$. We set $g_0 = \mu_g$ to model the posterior distribution, so f can be expressed as:

$$\begin{aligned} f(\mathbf{x}) &\approx f^* - \frac{1}{2}\mu_g^2(\mathbf{x}) - \mu_g(\mathbf{x})\left[g(\mathbf{x}) - \mu_g(\mathbf{x})\right] \\ &= f^* + \frac{1}{2}\mu_g^2(\mathbf{x}) - \mu_g(\mathbf{x})g(\mathbf{x}). \end{aligned} \tag{7}$$

Now, the posterior distribution has a form for $p\left(f \mid .\right) = \mathcal{N}(f \mid \mu, \sigma)$, where

$$\mu(\mathbf{x}) = f^* - \frac{1}{2}\mu_g^2(\mathbf{x}), \tag{8}$$

and

$$\sigma(\mathbf{x}) = \mu_g(\mathbf{x})\sigma_g(\mathbf{x})\mu_g(\mathbf{x}). \tag{9}$$

By implementing such transformations, we can exploit the information of the given spec and force the Gaussian process to get close to f^* without getting above it, as shown in Figure 4.

4.2 Expected hyper-regret minimization

The acquisition function is crucial in Bayesian optimization, determining where to sample next. Based on the idea of getting close to the given value f^* in spec, a multi-objective acquisition function is developed, called expected hyper-regret minimization (EHRM).

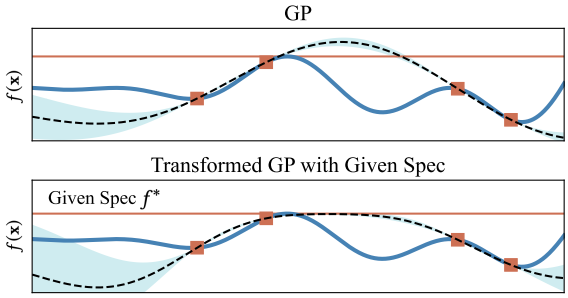


Figure 4: Comparison between ordinary Gaussian process and our transformed Gaussian process. The brown square in the figure is the evaluated point, and the blue curve is the real function $f(x)$. The dotted line and the cyan area represent the predicted mean and variance, respectively. It can be seen that the predicted mean (dotted line) and variance (cyan area) do not go above the target value from the spec in the transformed Gaussian process, while the ordinary Gaussian process does.

Here, the word 'regret' means the miss-distance between the metrics of the current parameter configuration and f^*, which is defined as:

$$r(\mathbf{x}) = f^* - f(\mathbf{x}). \tag{10}$$

In a normal posterior distribution, the probability of $r(\mathbf{x})$ is

$$P(r) = \frac{1}{\sqrt{2\pi}\sigma(\mathbf{x})}\exp\left(-\frac{1}{2}\frac{\left[f^* - \mu(\mathbf{x}) - r(\mathbf{x})\right]^2}{\sigma^2(\mathbf{x})}\right). \tag{11}$$

When it comes to multiple QoR metrics to be optimized, which means $\boldsymbol{f^*}$ is now a vector, so we introduce hyper-regret $Hr(\mathbf{x})$ to express the regret in high-dimension circumstances.

Hyper-regret measures the distance between the current metrics and the spec's value in a normalized space. It is used to quantify the disappointment of the metrics of a specific SoC architecture when the spec is given.

Since the information of each metric's mean and variance can be calculated from the transformed Gaussian process, the expected hyper-regret is defined as the expectation of $Hr(\mathbf{x})$ with respect to the posterior predictive distribution of the transformed GP. EHRM aims to find a configuration of SoC architecture parameters with an expectation closet to the given spec, in other words, with the smallest expected hyper-regret:

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathscr{X}} \mathbb{EHRM}(\mathbf{x}) = \arg\min_{\mathbf{x}\in\mathscr{X}} \mathbb{E}[Hr(\mathbf{x})]. \tag{12}$$

EHRM utilizes mean and variance similarly to EHVI (Expected Hypervolume Improvement) [16], while the ideas behind them are different. EHVI prefers high means and high variance, which means it always desires to find a better QoR result in high-variance areas. However, the same desire for EHRM decreases with the result reaching the given value in the spec.

5 Experiment

5.1 Experimental Setup and Benchmarks

We test our flow on a Gemmini-based RISC-V SoC benchmarked on some popular LLMs, including Transformer-small [17], Bert-base [18], Transformer-large, Bert-large, and GPT1-small [19]. Chipyard 1.9.1 is leveraged to generate RTL code from parameterized Chisel

Table 1: Examples of Parameters

Parameters	Stage	Candidates
cpu_type	CPU core	RocketBig/BoomMed/BoomLarge
L2TLBs	L2 Cache	512, 1024
nWays		4, 8, 16
capacityKB		512, 1024
tile_Rows/Columns	Accelerator	1, 4, 8
mesh_Rows/Columns		8, 16, 32, 64
sp_capacity		256, 512, 1024, 2048, 4096
dma_buswidth		128, 256

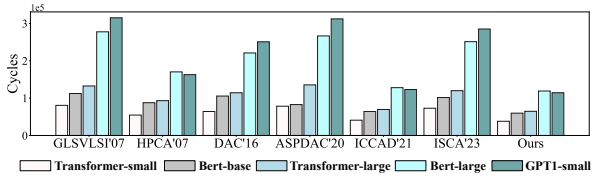


Figure 5: Cycles of language model tests on different methods' output optimal parameters.

files. A bespoke simulator built on Spike is used for fast cycle simulation which returns the number of cycles to measure LLMs' inference cycles on the SoC designs. Cadence Genus 201 is invoked to synthesize the RTL design based on the ASAP7 7nm PDK. Cadence Joules 201 and PrimeTime 2018.06-SP1 analyze the power consumption. The multiple optimization objectives include power, area, and cycles.

The architecture design space is composed of 19 parameters, resulting in 3×10^8 parameter configurations. After refining our selection to exclude infeasible parameter configurations and sampling the remaining ones, we have included a total of 1124 parameter configurations in our offline dataset. Table 1 illustrates the parameters. Their definitions are shown below:

- *cpu_type* specifies the type of CPU cores, which is selected between different BOOM and Rocket cores.
- *L2TLBs* assigns the number of entries of translation look-aside buffer in L2 Cache.
- *nWays* assigns the number of associate sets of L2 Cache.
- *capacityKB* specifies the capacity of L2 Cache in Kilobytes.
- *tile_Rows/Columns* and *mesh_Rows/Columns* specify the dimensions of systolic arraies. Each tile is an array of PEs and each mesh is an array of tiles with connecting registers.
- *sp_capacity* determines the capacity of scratchpad memory.
- *dma_buswidth* assigns the bus width of the accelerator's DMA transaction.

The following metrics are employed to compare each SoC DSE method: hypervolume (HV), cycles-power hypervolume ($HV_{0,1}$), cycles-area hypervolume ($HV_{0,2}$), and average distance to reference set (ADRS). ADRS computes the average distance between the found Pareto designs and the ground-truth Pareto designs. The cycle here is the average cycle of the benchmarks' inference performance on the SoC. We compare our method with the following baselines, each with a 150-hour execution quota (including EDA tool runtimes):

- GLSVLSI'07 [20]: Support vector machine-based method.
- HPCA'07 [7]: Regression-based model with non-linear transformation.
- DAC'16 [9]: Method based on AdaBoost learning.
- ASPDAC'20 [21]: XGBoost-based method.

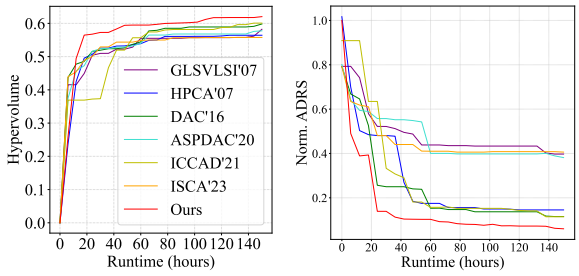


Figure 6: (a) The visualizations of different methods' normalized hypervolume changes with runtime (including EDA tools). (b) The visualizations of normalized ADRS changes with runtime.

- ICCAD'21 [12]: Method with active learning-based initial dataset sampling and Gaussian process with deep kernel learning.
- ISCA'23 [22]: Method based on randomized decision forests.

Our method and the baselines share the same randomly initialized parameter configurations, except ICCAD'21 [12] which utilizes its meticulously selected initial configurations. To evaluate our exploration ability and simultaneously perform a fair comparison against other Pareto-driven methods, the associated QoR metric values of Pareto-optimal designs are given in several separate specifications to feed into our approach. Our method is invoked in parallel, and the total time cost is summed up for each process.

5.2 Comparisons Against SOTA Works

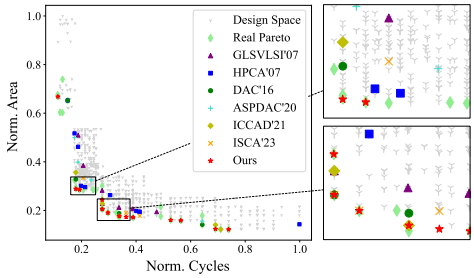
Table 2 shows the comparison of our method with existing methods on RISC-V SoC. Although these methods have the same runtime quota, our method can better explore the search space. In the normalized space of the 3 metrics (cycles, power, and area), our method acquires the highest hypervolume in all subspace ($HV_{0,1}$, $HV_{0,2}$ and HV). On average of these three metrics, our method is 9.02% higher than GLSVLSI'07 [20], 7.03% higher than HPCA'07 [7], 4.17% higher than DAC'16 [9], 6.30% higher than ASPDAC'20 [21], 4.02% higher than ICCAD'21 [12], 9.54% higher than ISCA'23 [22]. Visualization of the Pareto frontier in two QoR metrics is shown in Figure 7. Some Pareto results of different language model tests given by each method are shown in Figure 5. The SoC design output by our method can reach the least cycles in each of the language model tests.

Figure 6(a) and Figure 6(b) show the increase of hypervolume and the decrease of ADRS of each method. The red curve rises faster than others in hypervolume and falls faster than others in ADRS, which means our method outperforms others remarkably. In ADRS, our method converged to 0.107 after 40 hours of runtime, while DAC'16 [9] converged to 0.154 after 60 hours of runtime. It indicates that our method is more efficient in exploring the design space.

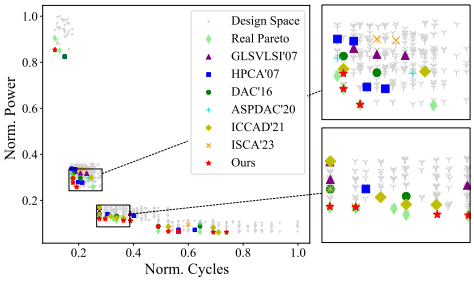
The associated performance of suggestion and the given spec QoR metric values are visualized in Figure 8(a). Color of the suggestion's performance is related to the time it is suggested. The later it is suggested, the darker its color is. It can be seen that the suggestion keeps getting close to the given spec QoR values and finally reaches them. The area ratio of the suggested SoC design's components is listed in Figure 8(b). The systolic mesh and the scratchpad occupy

Table 2: Comparisons against SOTA works

Metric \ Method	GLSVLSI'07 [20]	HPCA'07 [7]	DAC'16 [9]	ASPDAC'20 [21]	ICCAD'21 [12]	ISCA'23 [22]	**Ours**
$HV_{0,1}$	0.6320	0.6491	0.6789	0.6610	0.6716	0.6398	**0.7063**
$HV_{0,2}$	0.7129	0.7255	0.7231	0.7144	0.7251	0.6929	**0.7472**
HV	0.5577	0.5636	0.5891	0.5758	0.5975	0.5609	**0.6208**
Average	0.6342	0.6460	0.6637	0.6504	0.6647	0.6312	**0.6914**
Ratio(%)	91.72	93.43	95.99	94.07	96.13	91.29	**100**

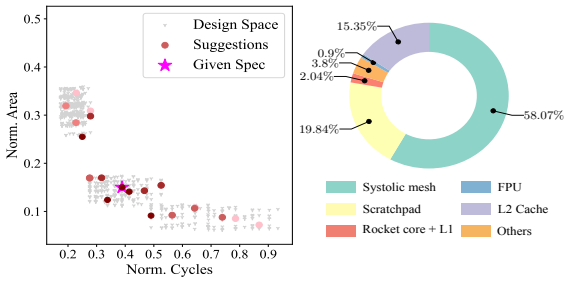


(a)



(b)

Figure 7: (a) The visualizations of selected Pareto frontiers in cycles-area metric space. (b) The visualizations of selected Pareto frontiers in cycles-power metric space.



(a) (b)

Figure 8: (a) The visualizations of transformed Bayesian optimization's suggestion getting close to given spec. (b) The area ratio of each component in our method's suggested design.

most of the design's area. The systolic mesh size of the suggested SoC design is 32, which is better fitted to the hidden dimensions and heads of LLMs on the benchmark.The capacity of the scratchpad is 2048 (KiB), large enough to store the result of the LLMs' inference.

6 Conclusion

In this paper, we have proposed an architecture design space exploration method based on the transformed Bayesian optimization approach to find an ideal RISC-V SoC design with given QoR metric values from the specification. The constructed model utilizes the given spec QoR metric values as additional information to learn and force the Gaussian process to get close to the target QoR values without beyond them. A tailored acquisition function is developed for optimization in multiple metrics (*e.g.*, cycles, power, and area). Experiments on 5 large language models under an advanced 7nm technology node have demonstrated the efficiency and effectiveness of the proposed framework.

Acknowledgment

This work is sponsored by Shanghai Pujiang Program (Project No. 22PJ1410400) and Open Project No. SKLICS-K202313 of State Key Laboratory of Integrated Chips and Systems, Fudan University.

References

[1] J. Bachrach *et al.*, "Chisel: Constructing Hardware in A Scala Embedded Language," in *Proc. DAC*, 2012.
[2] K. Asanovic *et al.*, "The Rocket Chip Generator," *EECS Department, University of California, Berkeley, Tech. Rep.*, 2016.
[3] ——, "The Berkeley Out-of-order Machine (boom): An Industry-competitive, Synthesizable, Parameterized RISC-V Processor," *University of California at Berkeley Berkeley United States, Tech. Rep*, 2015.
[4] H. Genc *et al.*, "Gemmini: Enabling Systematic Deep-learning Architecture Evaluation via Full-stack Integration," in *Proc. DAC*, 2021.
[5] A. Amid *et al.*, "Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs," *IEEE Micro*, 2020.
[6] E. Ïpek *et al.*, "Efficiently Exploring Architectural Design Spaces via Predictive Modeling," in *Proc. ASPLOS*, 2006.
[7] B. C. Lee and D. M. Brooks, "Illustrative Design Space Studies with Microarchitectural Regression Models," in *Proc. HPCA*, 2007.
[8] Q. Guo *et al.*, "Robust Design Space Modeling," *ACM TODAES*, 2015.
[9] D. Li *et al.*, "Efficient Design Space Exploration via Statistical Sampling and AdaBoost Learning," in *Proc. DAC*, 2016.
[10] S. Zhang *et al.*, "Bayesian Optimization Approach for Analog Circuit Synthesis Using Neural Network," in *Proc. DATE*, 2019.
[11] Y. Ma *et al.*, "CAD Tool Design Space Exploration via Bayesian Optimization," in *Proc. MLCAD*, 2019.
[12] C. Bai *et al.*, "BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework," in *Proc. ICCAD*, 2021.
[13] E. Blem *et al.*, "Power Struggles: Revisiting The RISC vs. CISC Debate on Contemporary ARM and x86 Architectures," in *Proc. HPCA*, 2013.
[14] Y. Lee *et al.*, "The Hwacha Microarchitecture Manual, Version 3.8." *University of California, Berkeley, Tech. Rep.*, 2015.
[15] J. Mockus, "The Application of Bayesian Methods for Seeking the Extremum," *Towards global optimization*, 1998.
[16] M. T. Emmerich *et al.*, "Hypervolume-based Expected Improvement: Monotonicity Properties and Exact Computation," in *Proc. CEC*, 2011.
[17] A. Vaswani *et al.*, "Attention is All You Need," *Proc. NeurIPS*, 2017.
[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[19] A. Radford *et al.*, "Improving Language Understanding by Generative Pre-training," https://openai.com/research/language-unsupervised/, 2018.
[20] M. Barros *et al.*, "GA-SVM Feasibility Model and Optimization Kernel Applied to Analog IC Design Automation," in *Proc. GLSVLSI*, 2007.
[21] Z. Xie *et al.*, "FIST: A Feature-importance Sampling and Tree-based Method for Automatic Design Flow Parameter Tuning," in *Proc. ASPDAC*, 2020.
[22] S. Krishnan *et al.*, "ArchGym: An Open-Source Gymnasium for Machine Learning Assisted Architecture Design," in *Proc. ISCA*, 2023.