The 38th Annual AAAI Conference on Artificial Intelligence
FEBRUARY 20-27, 2024 | VANCOUVER, CANADA
VANCOUVER CONVENTION CENTRE – WEST BUILDING

Towards Automated RISC-V Microarchitecture Design with Reinforcement Learning

Chen Bai[1], Jianwang Zhai[2], Yuzhe Ma[3], Bei Yu[1], Martin D.F. Wong[4]

[1]The Chinese University of Hong Kong
[2]Beijing University of Posts and Telecommunications
[3]The Hong Kong University of Science and Technology (Guangzhou)
[4]Hong Kong Baptist University

January 11, 2024

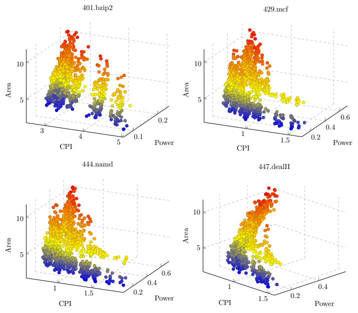
Outline

Introduction

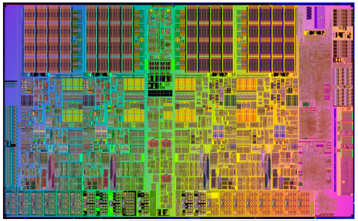
Introduction

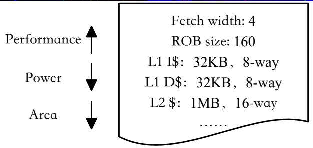
Problem Formulation

Given the microarchitecture design space and target workloads, how do we efficiently search for optimal microarchitectures that can satisfy the pre-determined performance, power, and area (PPA) design targets?



An example microprocessor's design space for SPEC2K6

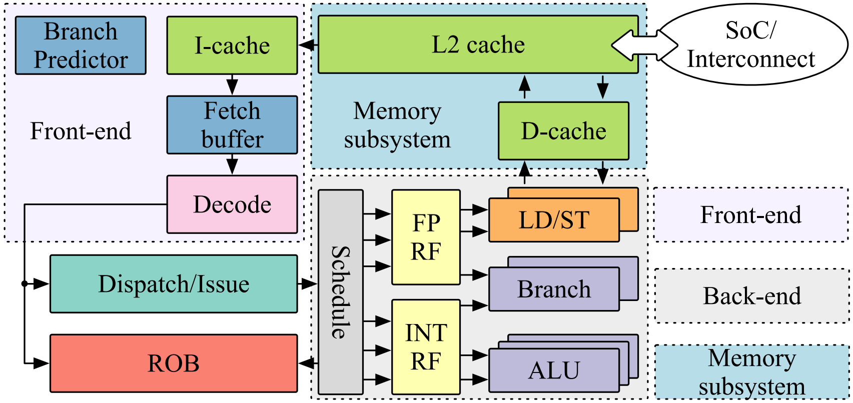




Fetch width: 4
ROB size: 160
L1 I$: 32KB, 8-way
L1 D$: 32KB, 8-way
L2 $: 1MB, 16-way
……

An example of the microarchitecture design space exploration.

Introduction



An overview of the example microprocessor microarchitecture, including different *components*.

Related Work

- Industry: computer architects' expertise.
- Academia:
  - Analytical methodology: interpretable PPA models, explainable search strategy, *etc.*[1][2]
  - Black-box methodology: machine-learning-based PPA modeling and search strategy.[3][4][5]

Limitations:

- Industry solution: architects' personal bias can yield sub-optimal solutions.
- Academic solution: not tightly coupled with expert knowledge & mathematical limitation in the Gaussian process modeling.

[1] Tejas S Karkhanis and James E Smith (2004). "A First-order Superscalar Processor Model". In: *IEEE/ACM International Symposium on Computer Architecture (ISCA)*. IEEE, pp. 338–349.

[2] Tejas S Karkhanis and James E Smith (2007). "Automated design of application specific superscalar processors: an analytical approach". In: *IEEE/ACM International Symposium on Computer Architecture (ISCA)*, pp. 402–411.

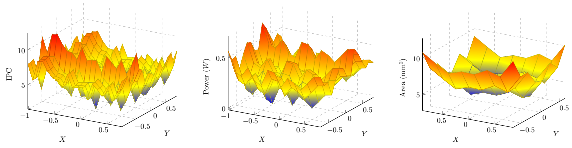
[3] Engin İpek et al. (2006). "Efficiently Exploring Architectural Design Spaces via Predictive Modeling". In: *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 195–206.

[4] Tianshi Chen et al. (2014). "Archranker: A ranking approach to design space exploration". In: *IEEE/ACM International Symposium on Computer Architecture (ISCA)*, pp. 1–12.

[5] Dandan Li et al. (2016). "Efficient design space exploration via statistical sampling and AdaBoost learning". In: *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6.

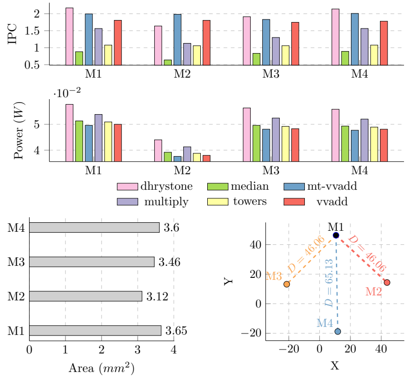
Limitation of Gaussian Process Modeling[6]

The kernel function of the Gaussian process mathematically attributes the PPA differences between two microarchitectures to the microarchitecture embedding distances.



(a) Performance (b) Power (c) Area

A visualization of the design space for 458.sjeng.



An example of different BOOM microarchitectures to demonstrate the claim.

[6] Chen Bai et al. (2021). "BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9.

Introduction

Highlights of our new black-box methodology:

- Remove mathematical limitation in the Gaussian process modeling (*i.e.*, free of unrealistic assumptions).
- Tightly coupled with expert knowledge: *microarchitecture scaling graph*.
- PPA design preference-driven exploration.
- Lightweight agent training environment design to accelerate the learning process.

Preliminaries

RISC-V Microarchitecture Design Space

Table 1: RISC-V Microarchitecture Design Space

Design	Component	Parameters	Candidate
Rocket	Branch predictor	RAS	0 : 12 : 3 [+]
		BTB.nEntries	0 : 56 : 14
		BHT.nEntries	0 : 1024 : 256
	I-cache	nWays	1, 2, 4
		nTLBWays	4 : 32 : 4
	Functional unit	FPU	1, 2
		mulDiv	1, 2, 3
		VM	1, 2
	D-cache	nSets	32, 64
		nWays	1, 2, 4
		nTLBWays	4 : 32 : 4
		nMSHRs	1, 2, 3
Small/Medium Large/Mega Giga SonicBOOM	Branch predictor	Type	1, 2, 3
		maxBrCount	4 : 22 : 2
	IFU	numFetchBufferEntries	6 : 46 : 2
		fetchWidth	4, 8
		ftq.nEntries	12 : 64 : 4
	pipelineWidth		1 : 5 : 1
	ROB		24 : 160 : 4
	PRF	numIntPhysRegisters	40 : 176 : 8
		numFpPRF	34 : 132 : 6
	ISU	numFpPhysRegisters	1 : 5 : 1
		numEntries	6 : 52 : 2
		dispatchWidth	1 : 5 : 1
	LSU	LDQ	6 : 32 : 2
		STQ	6 : 36 : 2
	I-cache	nWays	4, 8
		nSets	32, 64
	D-cache	nWays	4, 8
		nSets	64, 128
		nMSHRs	2 : 10 : 2

[*] The values are start number:end number:stride, e.g., 0 : 12 : 3 denotes the entries of RAS can be 0, 3, 6, etc., until 12.

Microarchitecture PPA Modeling

- Pre-RTL PPA modeling infrastructure:
  - GEM5[7][8] , McPAT[9], *etc.*
- RTL PPA modeling infrastructure:
  - Synopsys VCS[10], Synopsys PrimeTime PX[11], *etc.*

[7]Nathan Binkert, Bradford Beckmann, Gabriel Black, et al. (2011). "The Gem5 Simulator". In: *SIGARCH Comput. Archit. News* 39.2, pp. 1–7.

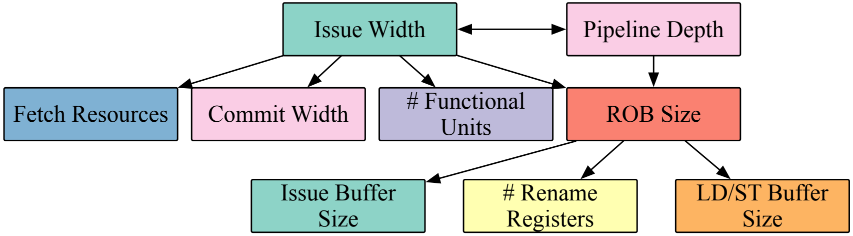
[8]Jason Lowe-Power et al. (2020). "The GEM5 Simulator: Version 20.0+". In: *arXiv preprint arXiv:2007.03152*.

[9]Sheng Li et al. (2009). "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures". In: *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 469–480.

[10]*Synopsys VCS* (2023). https://www.synopsys.com/verification/simulation/vcs.html.

[11]*Synopsys PrimeTime PX Power Analysis* (2023). https://news.synopsys.com/index.php?s=20295&item=123041.

Microarchitecture Scaling Graph



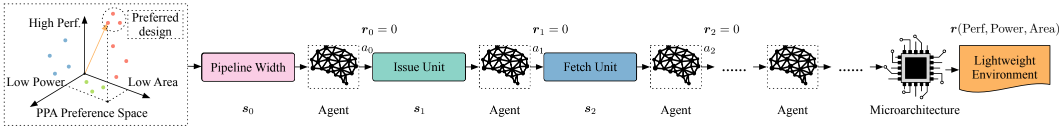
A microarchitecture scaling graph of an example out-of-order microprocessor.

- The microarchitecture scaling graph is a directed graph. Nodes are *components*. Edges are components' decision precedences[12].
- The graph is derived from the interval analysis. And the conclusion from the graph is general for Von Neumann architecture.

[12] Stijn Eyerman et al. (2009). "A Mechanistic Performance Model for Superscalar Out-of-order Processors". In: *Transactions on Computer Systems* 27.2, pp. 1–37.

Reinforcement Learning Methodology

Reinforcement Learning (RL) Methodology



An overview of our reinforcement learning methodology.

- The state space is the microarchitecture design space.
- The action space is the candidate set of components' types of corresponding hardware resources.
- The state transition is defined based on the microarchitecture scaling graph.
- Embed the architect's PPA design preference in the methodology.

PPA Design Preference

Reward scalarization:

$$r = \boldsymbol{r}(\text{Perf}, \text{Power}, \text{Area}) \cdot (\alpha, \beta, \gamma)^{\top} \tag{1}$$[13]

α, β, and γ are weights controlling the PPA trade-off.
Simplex constraints for PPA design preference $\boldsymbol{\phi}$:

$$\begin{gathered} r = \boldsymbol{r}(\text{Perf}, \text{Power}, \text{Area}) \cdot \boldsymbol{\phi}^{\top} \\ \forall i,\ \phi_i \geq 0,\ \sum_i \phi_i = 1 \end{gathered} \tag{2}$$

[13] Perf, Power, Area are normalized values.

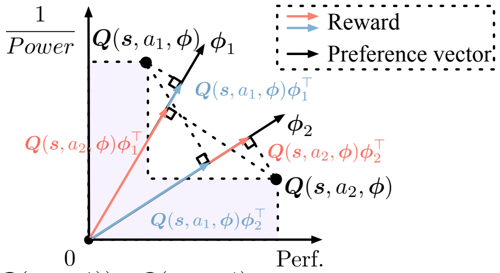
Embed PPA Preference Space into RL

Generalized Bellman optimality equality:

$$\begin{aligned}&\boldsymbol{Q}(\boldsymbol{s},a,\boldsymbol{\phi}) = \\ &\boldsymbol{r}(\boldsymbol{s},a) + \zeta\mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},a)}\mathcal{T}(\boldsymbol{Q}(\boldsymbol{s}',a,\boldsymbol{\phi})), \\ &\mathcal{T}(\boldsymbol{Q}(\boldsymbol{s}',a,\boldsymbol{\phi})) = \underset{\boldsymbol{Q}}{arg}\ \underset{a'\in A,\boldsymbol{\phi}'\in\Phi}{max}\ \boldsymbol{Q}(\boldsymbol{s}',a',\boldsymbol{\phi}')\boldsymbol{\phi}^\top,\end{aligned} \tag{3}$$

ζ is the discount factor, $\boldsymbol{Q}(\boldsymbol{s},a,\boldsymbol{\phi})$ is the state-action vector, and $\boldsymbol{\phi}$ is the PPA design preference.

14



$\frac{1}{Power}$
$\boldsymbol{Q}(\boldsymbol{s},a_1,\boldsymbol{\phi})$
$\boldsymbol{\phi}_1$
Reward
Preference vector
$\boldsymbol{Q}(\boldsymbol{s},a_1,\boldsymbol{\phi})\boldsymbol{\phi}_1^\top$
$\boldsymbol{\phi}_2$
$\boldsymbol{Q}(\boldsymbol{s},a_2,\boldsymbol{\phi})\boldsymbol{\phi}_1^\top$
$\boldsymbol{Q}(\boldsymbol{s},a_2,\boldsymbol{\phi})\boldsymbol{\phi}_2^\top$
$\boldsymbol{Q}(\boldsymbol{s},a_2,\boldsymbol{\phi})$
$\boldsymbol{Q}(\boldsymbol{s},a_1,\boldsymbol{\phi})\boldsymbol{\phi}_2^\top$
0
Perf.

$$\mathcal{T}(\boldsymbol{Q}(\boldsymbol{s},a,\boldsymbol{\phi})) = \boldsymbol{Q}(\boldsymbol{s},a_2,\boldsymbol{\phi}) = \underset{\boldsymbol{Q}}{argmax}\begin{Bmatrix}\boldsymbol{Q}(\boldsymbol{s},a_1,\boldsymbol{\phi})\boldsymbol{\phi}_1^\top, \boldsymbol{Q}(\boldsymbol{s},a_1,\boldsymbol{\phi})\boldsymbol{\phi}_2^\top \\ \boldsymbol{Q}(\boldsymbol{s},a_2,\boldsymbol{\phi})\boldsymbol{\phi}_1^\top, \boldsymbol{Q}(\boldsymbol{s},a_2,\boldsymbol{\phi})\boldsymbol{\phi}_2^\top\end{Bmatrix}$$

Optimization procedure with the generalized Bellman optimality equality.

[14] Axel Abels et al. (2019). "Dynamic Weights In Multi-objective Deep Reinforcement Learning". In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 11–20.

Reinforcement Learning (RL) Methodology

- We adopt the asynchronous advantage actor-critic (A3C)[15].
- We utilize the conditioned neural network design[16].

Gradients of the actor weights $\boldsymbol{\theta}_a$:

$$\begin{aligned}\nabla\boldsymbol{\theta}_a =& \kappa \nabla_{\boldsymbol{\theta}_a} H(\pi(\boldsymbol{s}_t;\boldsymbol{\theta}_a))+ \\ & \mathbb{E}_{\xi\sim\pi}[\sum_{t=0}^{\infty} \nabla_{\boldsymbol{\theta}_a} \log \pi_{\boldsymbol{\theta}_a}(a_t \mid \boldsymbol{s}_t) A(\boldsymbol{s}_t, a_t, \boldsymbol{\phi}')\boldsymbol{\phi}^\top],\end{aligned} \tag{4}$$

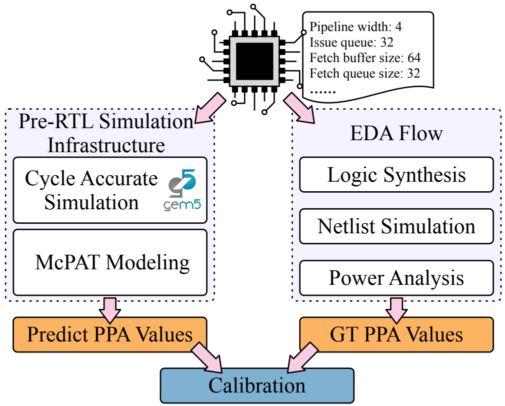
Loss function of the critic:

$$\begin{aligned}L_c =& \rho\|(\boldsymbol{Q}^* - \boldsymbol{Q}(\boldsymbol{s}, a, \boldsymbol{\phi}'; \boldsymbol{\theta}_c))\boldsymbol{\phi}^\top\|_2^2 + \\ & (1-\rho)\|\boldsymbol{Q}^* - \boldsymbol{Q}(\boldsymbol{s}, a, \boldsymbol{\phi}'; \boldsymbol{\theta}_c)\|_2^2,\end{aligned} \tag{5}$$

[15] Volodymyr Mnih, Adria Puigdomenech Badia, et al. (2016). "Asynchronous Methods for Deep Reinforcement Learning". In: *International Conference on Machine Learning (ICML)*. vol. 48, pp. 1928–1937.

[16] Axel Abels et al. (2019). "Dynamic Weights In Multi-objective Deep Reinforcement Learning". In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 11–20.

Accelerate Learning via Lightweight Environment



An overview of the PPA calibration.

- PPA models calibration flow[17].
- PPA models update policy.

[17] Jianwang Zhai et al. (2021). "McPAT-Calib: A Microarchitecture Power Modeling Framework for Modern CPUs". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, pp. 1–9.

Experiments

Experiments Environment

- RTL implementation: Rocket & BOOM[18][19].
- Technology node: 7-nm ASAP7 PDK[20].
- EDA tools: Cadence Genus 18.12-e012_1, Synopsys VCS M-2017.03, PrimeTime PX R-2020.09-SP1, *etc.*
- Server: 80 $\times$ Intel(R) Xeon(R) CPU e7-4803 v2 @ 2.20GHz, 1TB main memory.

Baselines

- ISCA'14: ArchRanker[21]
- DAC'16: AdaBoost[22]
- ICCAD'21: BOOM-Explorer[23]

[18] Krste Asanović, Rimas Avizienis, Jonathan Bachrach, et al. (2016). *The Rocket Chip Generator*. Tech. rep. University of California, Berkeley.

[19] Jerry Zhao et al. (2020). "SonicBOOM: The 3rd Generation Berkeley Out-of-order Machine". In: *Workshop on Computer Architecture Research with RISC-V (CARRV)*.

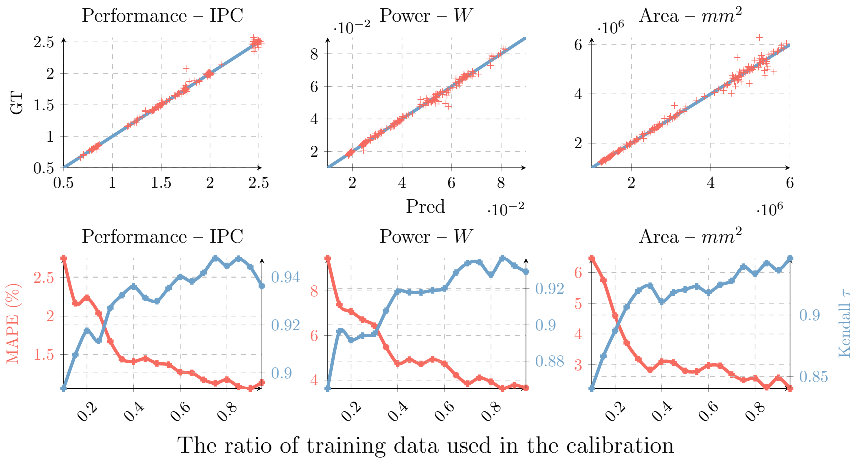
[20] Lawrence T Clark et al. (2016). "ASAP7: A 7-nm FinFET Predictive Process Design Kit". In: *Microelectronics Journal* 53, pp. 105–115.

[21] Tianshi Chen et al. (2014). "Archranker: A ranking approach to design space exploration". In: *IEEE/ACM International Symposium on Computer Architecture (ISCA)*, pp. 1–12.

[22] Dandan Li et al. (2016). "Efficient design space exploration via statistical sampling and AdaBoost learning". In: *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6.

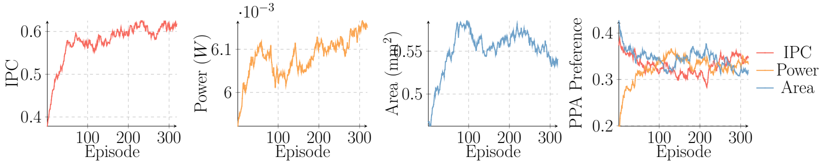
[23] Chen Bai et al. (2021). "BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9.

Accuracy of Lightweight PPA Models

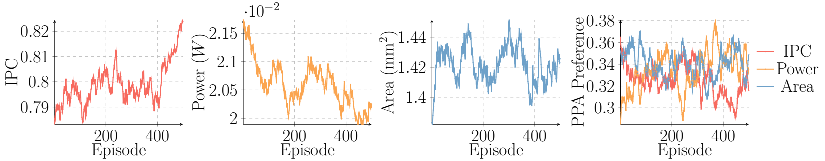


The accuracy of lightweight PPA models, and MAPE and Kendall τ curves *w.r.t.* the calibration data size.

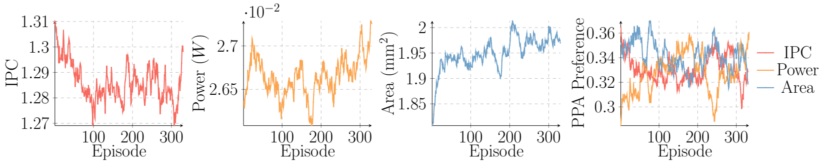
RL Training I



(a) Rocket

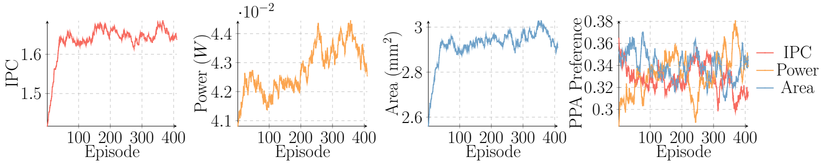


(b) Small-scale BOOM

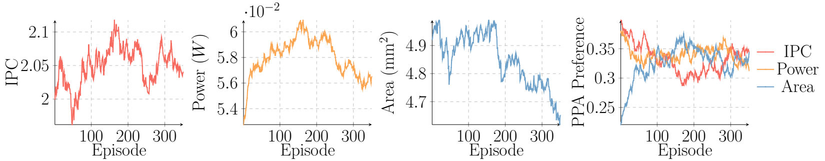


(c) Medium-scale BOOM

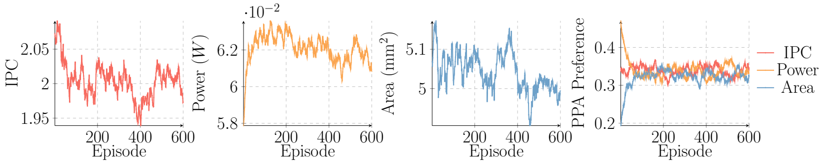
RL Training II



(a) Large-scale BOOM



(b) Mega-scale BOOM



(c) Giga-scale BOOM

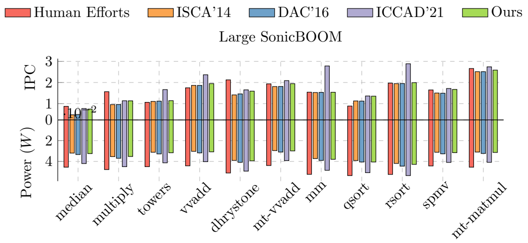
Main Evaluation Results

Table 2: Comparison w. Human Efforts & Prior Arts

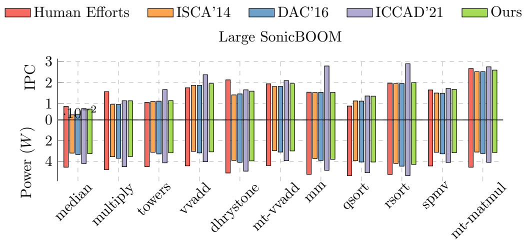
Design	Method	Performance	Power	Area	Perf / Power		Perf / Area		(Perf × Perf) / (Power × Area)		Runtime
		IPC	W	mm^2	Val.	Ratio	Val.	Ratio	Val.	Ratio	
Rocket	Human Efforts	0.7338	0.0027	0.9082	267.4708	–[1]	0.8080	–	216.1090	–	–
	ISCA'14	0.8157	0.0023	0.7943	**359.3222**	**1.3434**×	1.0270	1.2710×	369.0075	1.7075×	8.6111×
	DAC'16	0.5485	0.0018	0.5337	305.3090	1.1415×	1.0278	1.2721×	313.8042	1.4527×	5.8961×
	ICCAD'21	0.7278	0.0021	0.7448	352.7177	1.3187×	0.9771	1.2093×	344.6327	1.5947×	1.5011×
	Ours	0.7278	0.0023	0.5762	313.6958	1.1728×	**1.2631**	**1.5633**×	**396.2335**	**1.8335**×	**1.0000**
Small SonicBOOM	Human Efforts	0.7837	0.0203	1.5048	38.6057	–	0.5209	–	20.1062	–	–
	ISCA'14	0.8197	0.0150	1.2838	54.7692	1.4187×	0.6385	1.2260×	34.9710	1.7393×	5.8033×
	DAC'16	0.8076	0.0147	1.2512	54.8119	1.4198×	0.6454	1.2393×	35.3765	1.7594×	4.7918×
	ICCAD'21	0.8469	0.0200	1.5026	42.3436	1.0968×	0.5636	1.0821×	23.8645	1.1869×	1.3053×
	Ours	0.8403	0.0152	1.2538	**55.2813**	**1.4320**×	**0.6702**	**1.2868**×	**37.0491**	**1.8427**×	**1.0000**
Medium SonicBOOM	Human Efforts	1.1938	0.0256	1.9332	46.6952	–	0.6175	–	28.8363	–	–
	ISCA'14	1.2362	0.0196	1.6242	**62.9622**	**1.3484**×	**0.7611**	**1.2324**×	**47.9192**	**1.6618**×	5.6879×
	DAC'16	1.3757	0.0254	1.9247	54.0894	1.1584×	0.7148	1.1574×	38.6609	1.3407×	4.6966×
	ICCAD'21	1.4454	0.0271	2.1583	53.3342	1.1422×	0.6697	1.0844×	35.7170	1.2386×	1.2793×
	Ours	1.2872	0.0206	1.7351	62.5886	1.3404×	0.7419	1.2014×	46.4339	1.6103×	**1.0000**
Large SonicBOOM	Human Efforts	1.4871	0.0446	3.2055	33.3430	–	0.4639	–	15.4686	–	–
	ISCA'14	1.4900	0.0309	2.5420	48.2184	1.4461×	0.5861	1.2634×	28.2626	1.8271×	5.8920×
	DAC'16	1.4919	0.0324	2.6744	45.9976	1.3795×	0.5578	1.2024×	25.6592	1.6588×	4.8651×
	ICCAD'21	1.9162	0.0409	3.6715	46.8507	1.4051×	0.5219	1.1250×	24.4520	1.5808×	1.3252×
	Ours	1.5882	0.0314	2.5643	**50.6324**	**1.5185**×	**0.6193**	**1.3350**×	**31.3580**	**2.0272**×	**1.0000**
Mega SonicBOOM	Human Efforts	1.9500	0.0578	4.8059	33.7571	–	0.4058	–	13.6972	–	–
	ISCA'14	2.4957	0.0566	5.3676	44.0942	1.3062×	0.4650	1.1459×	20.5020	1.4968×	5.5443×
	DAC'16	2.4995	0.0562	5.3797	44.4483	1.3167×	0.4646	1.1451×	20.6513	1.5077×	4.5780×
	ICCAD'21	2.4823	0.0607	4.7008	40.9170	1.2121×	**0.5281**	**1.3014**×	21.6066	1.5774×	1.2470×
	Ours	2.5232	0.0557	5.2512	**45.3005**	**1.3420**×	0.4805	1.1842×	**21.7674**	**1.5892**×	**1.0000**
Giga SonicBOOM	Human Efforts	1.8717	0.0716	5.0691	26.1538	–	0.3692	–	9.6572	–	–
	ISCA'14	2.2528	0.0622	6.0010	36.2192	1.3849×	0.3754	1.0167×	13.5970	1.4080×	5.6321×
	DAC'16	2.2522	0.0773	5.5995	29.1480	1.1145×	**0.4022**	**1.0893**×	11.7236	1.2140×	4.6505×
	ICCAD'21	2.2650	0.0745	5.8652	30.4162	1.1630×	0.3862	1.0459×	11.7460	1.2163×	1.2668×
	Ours	2.2692	0.0595	5.7459	**38.1587**	**1.4590**×	0.3949	1.0695×	**15.0696**	**1.5605**×	**1.0000**

[1] "–" denotes not applicable.

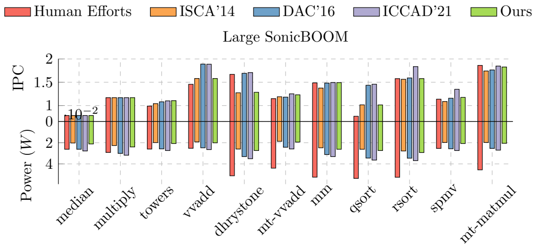
Analysis w. More Workloads



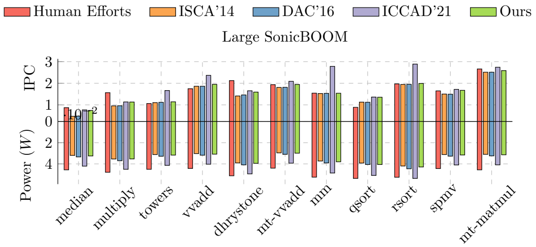
(a) Rocket



(b) Small-scale BOOM

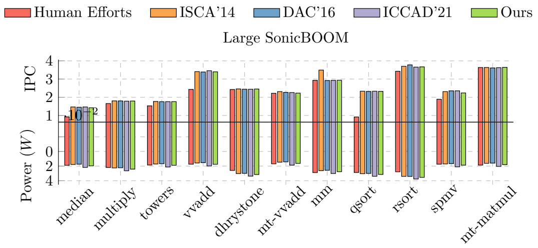


(c) Medium-scale BOOM

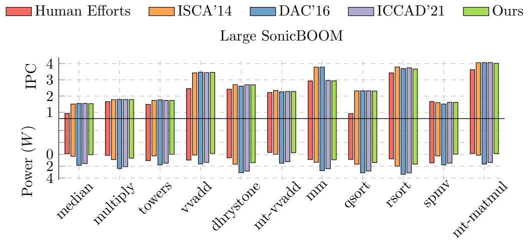


(d) Large-scale BOOM

Analysis w. More Workloads (cont.)



(e) Mega-scale BOOM



(f) Giga-scale BOOM

Summary:

- Our solutions achieve an average of 24.64%, 17.13%, and 6.33% than ICCAD'21, DAC'16, and ISCA'14 in PPA trade-off, respectively with 4.07$\times$ higher efficiency than baselines.
- For large-scale BOOM, compared to human implementations, our solution demonstrates significant improvements in three metrics by factors of 1.35$\times$, 1.23$\times$, and 1.66$\times$, respectively.

Conclusion

Conclusion

- Remove mathematical limitation in the Gaussian process modeling (*i.e.*, free of unrealistic assumptions).
- Tightly coupled with expert knowledge: *microarchitecture scaling graph*.
- PPA design preference-driven exploration.
- Lightweight agent training environment design to accelerate the learning process.
- Experiments show that our method achieves an average of 16.03% improvement in PPA trade-off with 4.07× higher efficiency than previous state-of-the-art approaches.

Conclusion

- Remove mathematical limitation in the Gaussian process modeling (*i.e.*, free of unrealistic assumptions).
- Tightly coupled with expert knowledge: *microarchitecture scaling graph*.
- PPA design preference-driven exploration.
- Lightweight agent training environment design to accelerate the learning process.
- Experiments show that our method achieves an average of 16.03% improvement in PPA trade-off with 4.07× higher efficiency than previous state-of-the-art approaches.

Discussion:

- Why do we choose A3C instead of PPO or SAC?
- Why does our method not perform very well on medium-scale SonicBOOM?
- Is the RL method the sole remedy to resolve the mathematical limitation (unrealistic assumption)?

THANK YOU!