2022 ICCAD CAD Contest Problem C: Microarchitecture Design Space Exploration

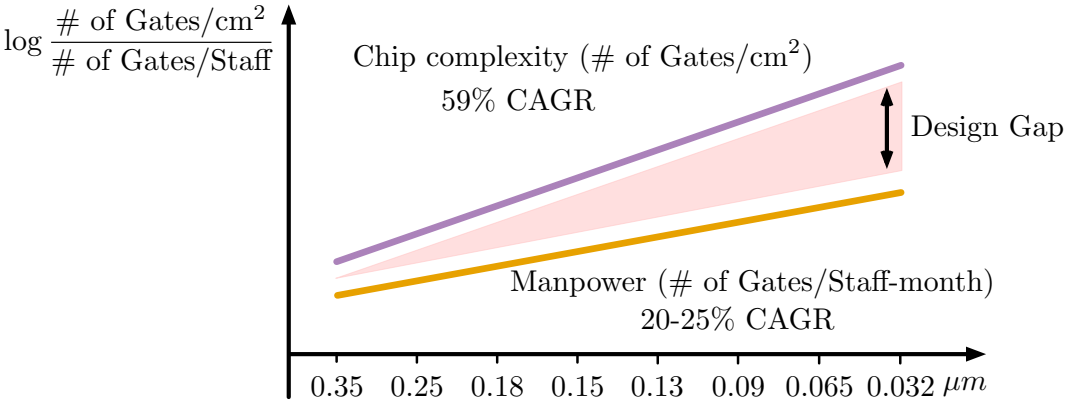
Sicheng Li [1, †], **Chen Bai**[1, 2, †], Xuechao Wei[1], Bizhao Shi[1], Yen-Kuang Chen[1], Yuan Xie[1]
[1]Alibaba Group [2]The Chinese University of Hong Kong
{sicheng.li,baichen.bai}@alibaba-inc.com

Oct. 30, 2022

† The two authors contributed equally to this work.

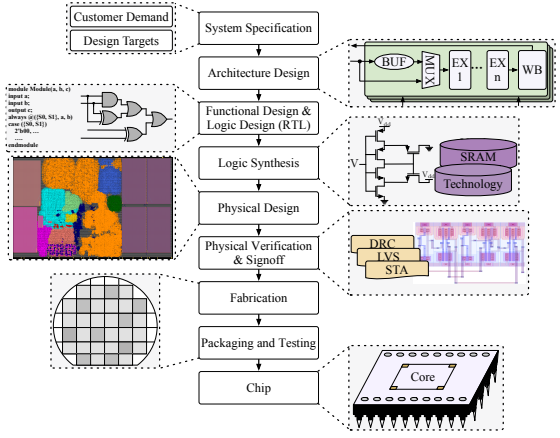
Outline

Introduction

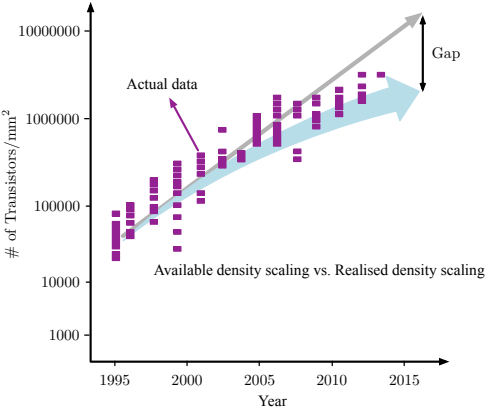


The continuously growing design productivity gap [1]

[1] *SEMICO Research Corp. SoC Silicon and Software 2018 Design Cost Analysis: How Rising Costs Impact SoC Design Starts* 2018



Overview of the VLSI FLow
(Tools and flows have steadily increased in complexity)



Design quality is not scaling

- **Incomplete, fabricatable prototypes** over fully featured models
- **Collaborative, flexible teams** over rigid silos
- **Improvement of tools and generators** over improvement of the instance
- **Response to change** over following a plan

Key Points

- Incomplete, fabricatable prototypes → Full RTL implementation model
- Improvement of tools and generators → Parametric chip generator
- Response to change → Available tape-out candidate for any product deadline

RISC-V Implementations

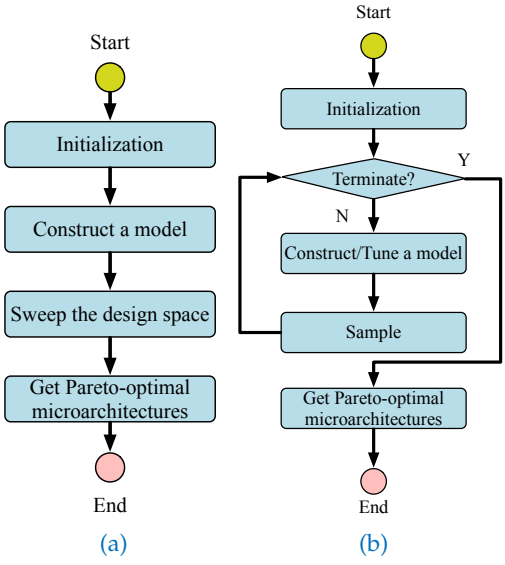
- Infrastructure
  - Bluespec System Verilog [Nikhil 2004], ROHD [2], Chisel [Bachrach et al. 2012], PyMTL [Lockhart, Zibrat, and Batten 2014], SpinalHDL [3]
  - Mamba [Jiang, Ilbeyi, and Batten 2018], FireMarshal [Pemberton and Amid 2021], Dromajo [Kabylkas et al. 2021], ChiselVerifyer [Dobis et al. 2021]
  - Hammer [Wang et al. 2020]
  - ...
- Methodology
  - BOOM-Explorer [Bai et al. 2021]
  - ...
- Platform
  - OpenPiton [Balkind et al. 2016], Chipyard [Amid et al. 2020], CHIPKIT [Whatmough et al. 2020]
  - ...

[2] *Rapid Open Hardware Development (ROHD) Framework* (2021). https://github.com/intel/rohd

[3] *SpinalHDL: A Language to Describe Digital Hardware* (2020). https://github.com/SpinalHDL/SpinalHDL

Design Space Exploration

Design Space Exploration



(*a*) Offline design space exploration flow. (*b*) Online design space exploration flow.

Domination Criterion

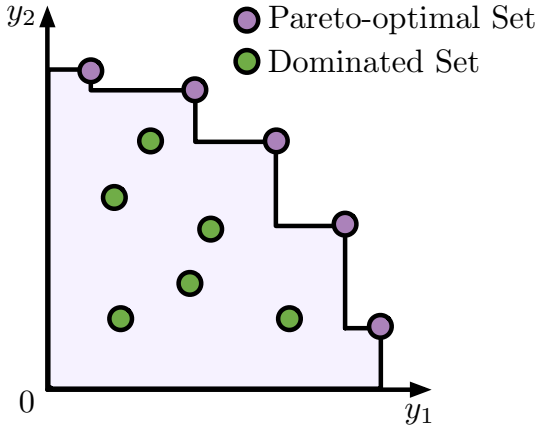
For an n-objective maximization problem, a vector of objective values $\boldsymbol{y}'$ is said to dominate $\boldsymbol{y}$ if

$$\begin{aligned} &\forall i \in [1, n], \quad \boldsymbol{y}' \geq \boldsymbol{y}; \\ &\exists j \in [1, n], \quad \boldsymbol{y}' > \boldsymbol{y}, \end{aligned} \tag{1}$$

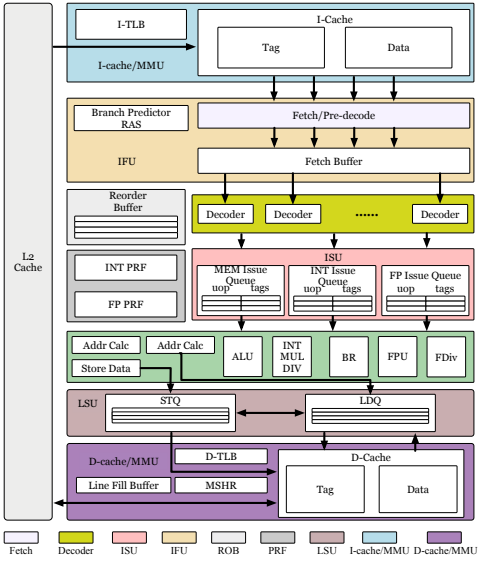
Hence, we denote $\boldsymbol{y}' \succeq \boldsymbol{y}$. Otherwise, $\boldsymbol{y}' \not\succeq \boldsymbol{y}$.

Pareto-optimal Set

Given $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_n\}$, the Pareto-optimal set is to be defined as
$\mathcal{P}(Y) = \{\boldsymbol{y}_i \in Y \mid \boldsymbol{y}_j \not\succeq \boldsymbol{y}_i, \forall j \in Y \setminus \{\boldsymbol{y}_i\}\}$.



Overview of Pareto-optimal Set



Overview of RISC-V BOOM [Asanovic, Patterson, and C. Celio 2015; C. P. Celio 2017; Zhao et al. 2020]

Table: Microarchitecture Design Space of BOOM

Design	Boxes									Total
	Fetch	Decoder	ISU	IFU	ROB	PRF	LSU	I-cache/MMU	D-cache/MMU	
sub-design-1	1	1	1, 2, 3	1, 2, 3	1, 2, 3, 4	1, 2, 3	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	
sub-design-2	1	2	4, 5, 6	4, 5, 6	5, 6, 7	4, 5, 6	4, 5, 6	1, 2, 3, 4	1, 2, 3, 4	
sub-design-3	2	3	7, 8, 9	7, 8, 9	8, 9, 10	7, 8, 9	7, 8, 9	5, 6, 7	5, 6, 7	15633
sub-design-4	2	4	10, 11, 12	10, 11, 12	11, 12, 13	10, 11, 12	10, 11, 12	5, 6, 7	8, 9, 10	
sub-design-5	2	5	13, 14, 15	12, 13, 14	14, 15, 16	11, 12, 13	10, 11, 12	5, 6, 7	8, 9, 10	

Table: Components I

Index	Fetch	Decoder	LSU [1]		I-cache/MMU				D-cache/MMU					
	Width	Width	LDQ	STQ	Sets	Ways	I-TLBSets	I-TLBWays	Sets	Ways	RP [2]	MSHR	D-TLBSets	D-TLBWays
1	4	1	8	8	64	4	1	32	64	4	0	2	1	8
2	8	2	6	6	32	8	1	32	64	4	0	2	1	32
3		3	12	12	32	4	1	16	64	6	1	2	1	8
4		4	16	16	64	1	1	16	64	4	1	2	1	32
5		5	12	12	64	8	1	32	64	2	0	4	1	32
6			20	20	64	8	2	16	64	4	1	4	1	32
7			24	24	64	8	2	32	64	4	1	8	1	32
8			22	22	64	8	2	32	64	8	0	8	2	32
9			28	28					64	8	1	8	2	32
10			32	32					64	8	1	8	1	32
11			28	28										
12			36	36										

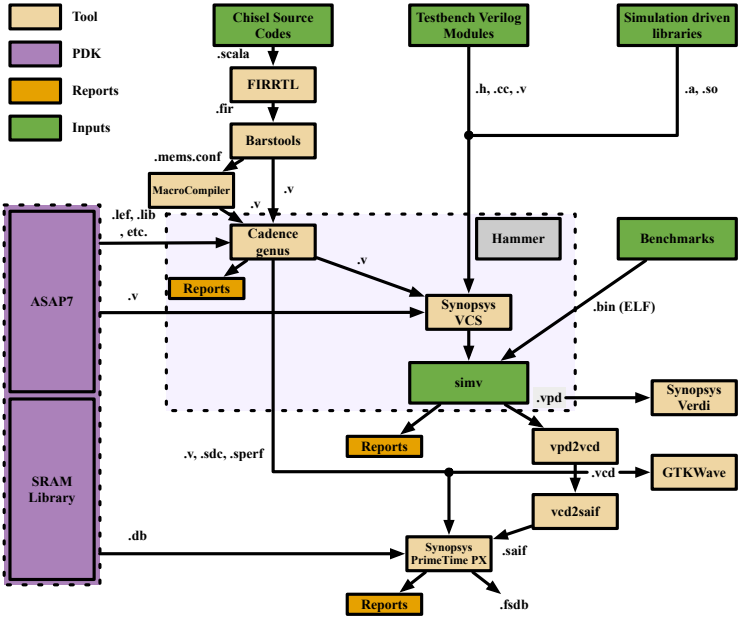
[1] LDQ and STQ are shored for load and store queue entries, respectively.

[2] RP is shorted for a replacement policy. Specifically, 0 denotes LRU, and 1 represents Pseudo LRU.

Table: Components II

Index	ISU [1]									IFU [2]			ROB	PRF [3]	
	MEM.DW	MEM.IW	MEM.QE	INT.DW	INT.IW	INT.QE	FP.DW	FP.IW	FP.QE	Tag	FBE	FTQ	Entries	INT	FP
1	1	1	8	1	1	8	1	1	8	8	8	16	32	52	48
2	1	1	6	1	1	6	1	1	6	6	6	14	30	42	38
3	1	1	10	1	1	12	1	1	12	10	12	20	34	62	58
4	2	1	12	2	2	20	2	1	16	12	16	32	36	80	64
5	2	2	12	2	2	20	2	2	16	10	14	30	64	70	54
6	2	2	14	2	2	24	2	2	20	14	20	36	60	90	74
7	3	1	16	3	3	32	3	1	24	16	24	32	72	100	96
8	3	2	16	3	3	32	3	2	24	14	21	30	96	90	86
9	3	2	20	3	3	36	3	3	28	18	30	36	90	110	106
10	4	2	24	4	4	40	4	2	32	20	32	40	108	118	118
11	4	2	28	4	4	44	4	4	36	22	36	44	128	128	128
12	4	2	22	4	4	36	4	2	28	24	40	48	132	138	138
13	5	2	24	5	5	40	5	2	32	20	35	40	136	146	146
14	5	2	26	5	4	44	5	4	36	26	45	50	130		
15	5	2	28	5	5	48	5	5	40				120		
16													140		

[1] DW, IW, and QE are shorted for dispatch width, issue width, and queue entries.
[2] FBE and FTQ are shorted for fetch buffer entries and fetch target queue entries, respectively.
[3] INT and FP are shorted for the number of integer and floating-point physical registers, respectively.



Overview of BOOM VLSI Flow

Problem Formulation

Our Solution Encoding

Microarchitecture embedding is defined as a combination of candidate values for each component of a microarchitecture, as shown in Table 1, Table 2, and Table 3.

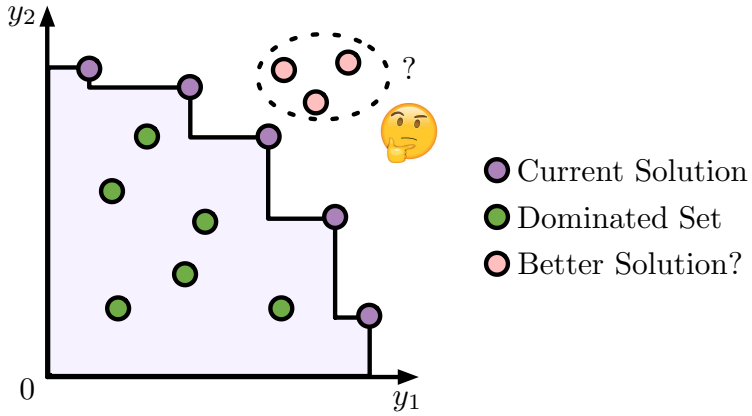
Our Optimization Target

Performance, power, and area (PPA) are three optimization targets.

- Microarchitecture embedding is denoted as a feature vector $\boldsymbol{x}$.
- A better microarchitecture has higher performance and lower power and area.

Our Problem

Given a design space $\mathcal{D} = \{x_1, x_2, ..., x_n\}$, experiment f maps the design space $\mathcal{D}$ to PPA metric value space $Y = \{y_1, y_2, ..., y_n\}$.
The microarchitecture design space exploration is to find $X \subseteq \mathcal{D}$, whose objective values are $\mathcal{P}(Y)$ as far as possible.



Where is the Pareto-optimal Set?

Evaluation

- Dataset of 15,633 RISC-V BOOM microarchitectures
- Contest Benchmarking Platform
- Evaluation Metrics
- Onling Ranking Platform

Microarchitecture Embedding	Performance	Power	Area	VLSI Time
4 1 1 1 8 1 ····· 2 1 8	0.9611	0.0265	1508807.5460	11011.3573
4 1 1 1 6 1 ····· 2 1 8	0.9389	0.0224	1499039.5220	10202.0257
4 1 1 1 10 1 ····· 2 1 8	0.9410	0.0225	1525471.5550	10099.7964
⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋱ ⋮ ⋮ ⋮	⋮	⋮	⋮	⋮
8 5 5 2 24 5 ····· 8 1 32	2.8075	0.0727	6295347.7110	36556.3131
8 5 5 2 26 5 ····· 8 1 32	2.8788	0.0872	6013093.0060	40801.0153
8 5 5 2 28 5 ····· 8 1 32	2.9200	0.0711	5856744.2460	35900.8144

Overview of the dataset, each PPA values of a microarchitecture is obtained from Figure 6

- Get started w. the contest benchmarking platform

```
1    pip3 install iccad-contest
```

- Contest solution implementation: Python3.5+

Listing: Template of the Solution Implementation

```
1
from iccad_contest.abstract_optimizer import
    AbstractOptimizer
3 from iccad_contest.design_space_exploration import
    experiment

5 class YourAlgorithm(AbstractOptimizer):
      primary_import = "iccad_contest"
7
      def __init__(self, design_space):
9         #
          #     build a wrapper class for an optimizer.
11        #
          #     parameters
13        #     ----------
```

```
        #     design_space: <class "
            MicroarchitectureDesignSpace">
15      #
        AbstractOptimizer.__init__(self, api_config)
17      # do whatever other setup is needed
        # ...
19
    def suggest(self):
21      #
        #     get a suggestion from the optimizer.
23      #
        #     returns
25      #     -------
        #     next_guess: <list> of <list>
27      #         list of `self.n_suggestions`
            suggestion(s).
```

```
        #         each suggestion is a microarchitecture
            embedding.
29      #
        # do whatever is needed to get the parallel
            guesses
31      # ...
        return x_guess
33
    def observe(self, X, y):
35      #
        #     send an observation of a suggestion back
            to the optimizer.
37      #
        #     parameters
39      #     ----------
        #     x: <list> of <list>
41      #         the output of 'suggest'.
```

Solution Implementation IV

```
            #     y: <list> of <list>
43          #         corresponding values where each 'x' is
                mapped to.
            #
45          # update the model with new objective function
                observations
            # ...
47          # no return statement needed

49  if __name__ == "__main__":
        # this is the entry point for experiments, so pass
            the class to
51      # 'experiment_main_entry' to use this optimizer.
        # this statement must be included in the wrapper
            class file:
53      experiment(YourAlgorithm)
```

Listing: Data structure Definition of the Design Space

```
descriptions = {
        "sub-design-1": {
                "Fetch": [1],
                "Decoder": [1],
                "ISU": [1, 2, 3],
                ...
        }
}

components_mappings = {
        "Fetch": {
                "description": ["FetchWidth"],
                "1": [4]
        },
        "Decoder": {
```

```
16                "description": ["DecodeWidth"],
                  "1": [1]
18        },
          "ISU": {
20                "description": [
                          "MEM_INST.DispatchWidth", "
                             MEM_INST.IssueWidth"
22                        "MEM_INST.NumEntries", "
                             INT_INST.DispatchWidth",
                          "INT_INST.IssueWidth", "
                             INT_INST.NumEntries",
24                        "FP_INST.DispatchWidth", "
                             FP_INST.IssueWidth",
                          "FP_INST.NumEntries"
26                ],
                  "1": [1, 1, 8, 1, 1, 8, 1, 1, 8],
28                "2": [1, 1, 6, 1, 1, 6, 1, 1, 6],
```

Key Data Structure III

```
                "3": [1, 1, 10, 1, 1, 12, 1, 1, 12]
        },
        "IFU": {
                "description": ["BranchTag", "
                   FetchBufferEntries", "
                   FetchTargetQueue"]
                "1": [8, 8, 16],
                ...
        },
        ...
}
```

Contest Benchmarking Platform Help Menu I

Listing: Benchmarking Platform Help Menu

```
$ python3 random-search-optimizer.py -h

usage: random-search-optimizer.py [-h] [-o OUTPUT_PATH]
    [-u UUID] [-s SOLUTION_SETTINGS] [-q NUM_OF_QUERIES
   ]

ICCAD'22 Contest Platform - solutions evaluation

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT_PATH, --output-path OUTPUT_PATH contest
     output path specification
  -u UUID, --uuid UUID  universally unique identifier (
     UUID) specification
```

```
11    -s SOLUTION_SETTINGS, --solution-settings
         SOLUTION_SETTINGS solution submission
         specification
      -q NUM_OF_QUERIES, --num-of-queries NUM_OF_QUERIES
         the number of queries specification
```

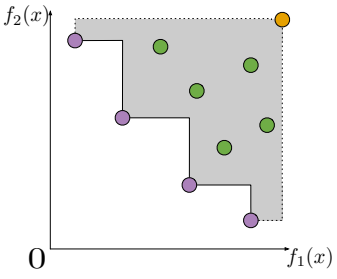
Listing: Example Command of Benchmarking Platform

```
$ python3 random-search-optimizer.py -o output -u "00
    ef538e88634ddd9810d034b748c24d" -q 20
...
[INFO]: summary for the solution, the best Pareto
    hypervolume: 37.59654046710655, the best Pareto
    hypervolume difference: 66.67984662813456 cost:
    164902.3422778734.
...
```

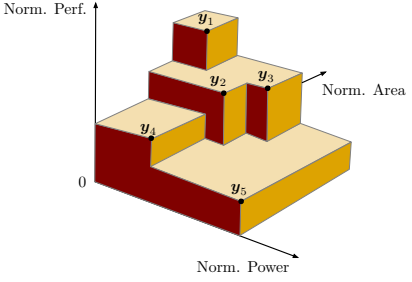
- Pareto hypervolume
- Overall running time: the total time of algorithms including VLSI flow runtime cost.
- A final score is calculated based on Pareto hypervolume and overall running time

Pareto hypervolume [Bai et al. 2021]

$$\mathrm{PVol}_{\boldsymbol{v}_{\mathrm{ref}}}(\mathcal{P}(Y)) = \int_{Y} \mathbb{1}[\boldsymbol{y} \not\succeq \boldsymbol{v}_{\mathrm{ref}}][1 - \prod_{\boldsymbol{y}_* \in \mathcal{P}(Y)} \mathbb{1}[\boldsymbol{y}_* \not\succeq \boldsymbol{y}]]\mathrm{d}\boldsymbol{y}, \tag{2}$$



(a)



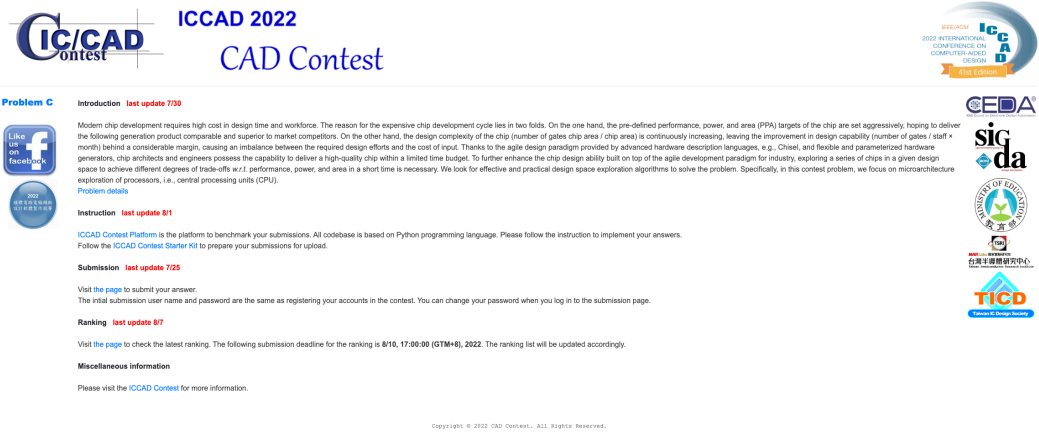
(b)

(a). An example overview of the Pareto hypervolume in the two dimensional space (b). An example overview of the Pareto hypervolume in the three dimensional space, *i.e.*, power, performance, and area.

Final Score

$$\text{score} = \text{PVol}_{v_{\text{ref}}} \cdot \begin{cases} \alpha - \dfrac{\text{ORT} - \theta}{\theta}, \text{ORT} \geq \theta \\ \alpha + |\dfrac{\text{ORT} - \theta}{\theta}|, \text{ORT} < \theta \end{cases}, \tag{3}$$

α is an ORT score baseline, equal to 6, and θ is a pre-defined ORT budget, equivalent to 2625000.



Overview of the online ranking platform.

Ranking list

- The online ranking platform will be maintained after the contest.
- Everybody can register and participate in solving the problem.
- Evaluation according to fixed frequency.
- Address: `http://47.93.191.38/`

Acknowledgement

Acknowledgement

We would like to thank

- **Prof. Yuzhe Ma**, from the Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou)
- **Prof. Bei Yu**, from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong SAR.

They suggested valuable ideas and helped to review the problem.

THANK YOU!

References I

Alon Amid et al. (2020). "Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs". In: *IEEE Micro* 40.4, pp. 10–21.

Krste Asanovic, David A Patterson, and Christopher Celio (2015). *The Berkeley Out-of-order Machine (BOOM): An Industry-competitive, Synthesizable, Parameterized RISC-V Processor*. Tech. rep. University of California at Berkeley.

Jonathan Bachrach et al. (2012). "Chisel: Constructing Hardware in a Scala Embedded Language". In: *ACM/IEEE Design Automation Conference (DAC)*, pp. 1212–1221.

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)*. IEEE, pp. 1–9.

Jonathan Balkind et al. (2016). "OpenPiton: An Open Source Manycore Research Framework". In: *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* 51.4, pp. 217–232.

Christopher Patrick Celio (2017). *A Highly Productive Implementation of an Out-of-Order Processor Generator*. eScholarship, University of California.

References II

Andrew Dobis et al. (2021). “ChiselVerify: An Open-source Hardware Verification Library for Chisel and Scala”. In: *IEEE Nordic Circuits and Systems Conference (NorCAS)*. IEEE, pp. 1–7.

Shunning Jiang, Berkin Ilbeyi, and Christopher Batten (2018). “Mamba: Closing the Performance Gap in Productive Hardware Development Frameworks”. In: *ACM/IEEE Design Automation Conference (DAC)*. IEEE, pp. 1–6.

Nursultan Kabylkas et al. (2021). “Effective Processor Verification with Logic Fuzzer Enhanced Co-simulation”. In: *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 667–678.

Yunsup Lee et al. (2016). “An Agile Approach to Building RISC-V Microprocessors”. In: *IEEE Micro* 36.2, pp. 8–20.

Derek Lockhart, Gary Zibrat, and Christopher Batten (2014). “PyMTL: A Unified Framework for Vertically Integrated Computer Architecture Research”. In: *IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, pp. 280–292.

References III

Rishiyur Nikhil (2004). "Bluespec System Verilog: Efficient, Correct RTL from High Level Specifications". In: *ACM/IEEE International Conference on Formal Methods and Models for Co-Design (MEMCOD)*. IEEE, pp. 69–70.

Nathan Pemberton and Alon Amid (2021). "Firemarshal: Making HW/SW Co-design Reproducible and Reliable". In: IEEE, pp. 299–309.

Rapid Open Hardware Development (ROHD) Framework (2021). https://github.com/intel/rohd.

SEMICO Research Corp. SoC Silicon and Software 2018 Design Cost Analysis: How Rising Costs Impact SoC Design Starts (2018). https://semico.com/content/soc-silicon-and-software-2018-design-cost-analysis-how-rising-costs-impact-soc-design-starts.

SpinalHDL: A Language to Describe Digital Hardware (2020). https://github.com/SpinalHDL/SpinalHDL.

Edward Wang et al. (2020). "A Methodology for Reusable Physical Design". In: *IEEE International Symposium on Quality Electronic Design (ISQED)*. IEEE, pp. 243–249.

References IV

Paul N Whatmough et al. (2020). “CHIPKIT: An Agile, Reusable Open-source Framework for Rapid Test Chip development”. In: *IEEE Micro* 40.4, pp. 32–40.

Jerry Zhao et al. (2020). “SonicBOOM: The 3rd Generation Berkeley Out-of-order Machine”. In: *Fourth Workshop on Computer Architecture Research with RISC-V*. Vol. 5.