Architectural Design Space Exploration for NPUs

NexNPU Tutorial @ ISCA'26

Yuqi Xue

Ph.D. Student, Electrical & Computer Engineering,

Systems Platform & Intelligence Lab @ University of Illinois Urbana-Champaign

yuqixue2@illinois.edu

Overview

Goals of Building an NPU Architectural Design Space Exploration Tool:

1. Quick (and cheap) early exploration of design trade-offs
2. Verify the theoretical benefits of a new idea
3. Evaluate a design choice before full-stack implementation
4. Develop cost models for other downstream tasks

Methodology:

1. Analytical models for all major components on an NPU chip (SA, VU, SRAM, HBM, ICI)
2. Analyze at different granularities: Operator-level, Model-level
3. Report performance, power/energy, and carbon footprint

Case Studies:

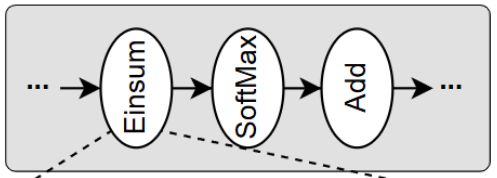
1. Single operator analysis and LLM inference request analysis (with Demo)
2. NPU component-level utilization analysis
3. Power gating & DVFS studies

Introducing NeuSim: An Open-Source Simulator Framework NPUs

NPU Chip Definition

SA: 4
VU: 4
SRAM Capacity: 128 MB
HBM Bandwidth: 2765 GB/s
ICI Link Bandwidth: 100 GB/s
SA Static Power: 8 W
SA Peak Dynamic Power: 30 W
...

DNN Model Definition



LHS Input Shape: [16, 1024, 1024]
RHS Input Shape: [16, 1024, 1024]
Einsum Expr: "bij;bjk;bik"
Data Type: bfloat16
...

Parallelism Config

Tensor Parallelism Degree: 4
Pipeline Parallelism Degree: 4
Data Parallelism Degree: 1
Expert Parallelism Degree: 4
Tensor Parallel Axes: 1
...

NeuSim Core Components

Performance Model
- Systolic Array
- Vector Unit
- On-chip SRAM
- Off-chip HBM
- Inter-chip Interconnect

Power Model
- Static Power
- Dynamic Power
- Power Gating
- DVFS

Carbon Model
- Embodied Carbon
- Operational Carbon

Output Statistics

Execution Time Breakdown:
Execution time, SA time, VU time, SRAM time, ICI time, HBM time, ...

Memory/ICI Traffic Analysis:
HBM traffic, ICI traffic, SRAM tile shape, ...

Hardware Resource Utilization:
FLOPS util., HBM BW util., ...

Power & Energy:
Static SA power, Static VU power, ...
Dynamic SA power, Dynamic VU power, ...

Carbon Footprint:
Total emission (kgCO2e), Carbon efficiency (e.g., kgCO2e per request/token)

Support for Multiple Simulation Modes in NeuSim

Operator-Level Simulation

```
"XlaEinsum(
    a=1x1x8192,
    b=8192x64x128,
    eq=BLM;MND->BLND,
    type=DT_BFLOAT16
)"
```

Tiling | Double Buffering | ICI Traffic Analysis | SA/VU Computation Mapping | HBM Traffic Analysis | ...

```
tot_exe_ns: 136436
SA_exe_ns: 136436
VU_exe_ns: 35356
SRAM_exe_ns: 54326
ICI_exe_ns: 0
HBM_exe_ns: 16478
...
```

Model-Level Simulation

RMSNorm
FFN
...
ML Model Graph Definition

Description	Config	Name	Op Type	Count	Bounded-by	Execution time	Compute time	Memory time	ICI/NVLink tim	ICI/NVLink outl	ICI/NVLink inb
DecodeReceiveIt	InterChipCommI	DecodeReceiveIt	ICINoCompute	512	ICI/NVLink	1842	0	277	1842	0	16384
Attention-servin	LayerNorm(x=1x	Attentionserving	VPU	10240	Memory	277	10	277	0	0	0
Attention-servin	XlaEinsum(a=1x	Attentionserving	VPU	10240	Memory	76817	19275	76817	0	0	0
Attention-servin	XlaEinsum(a=1x	Attentionserving	VPU	10240	Memory	153624	38551	153624	0	0	0
Attention-servin	XlaEinsum(a=1x	Attentionserving	VPU	10240	Memory	38709	9638	38709	0	0	0
Attention-servin	XlaEinsum(a=1x	Attentionserving	VPU	10240	Memory	4848	1205	4848	0	0	0
Attention-servin	Softmax(x=1x1x	Attentionserving	VPU	10240	Memory	676	170	676	0	0	0

LLM Parallelism Configs, Input/Output Sequence Lengths, Batch Size, …

Create Sharded Model Graph | Tensor Shape Analysis

```
prefill_TTFT_ms: 2864
decode_TPOT_ns: 1643758
total_energy_J: 12543
...
```

NeuSim Installation & Setup

1. Install the Python package

```
# inside the cloned NeuSim repo
conda create --name neusim python=3.12.2
conda activate neusim
pip install -e .
# optional: pip install -e ".[dev]"
```

2. Select model / chip / system

configs/chips/*.json

- num_sa, sa_dim, num_vu
- HBM BW, VMEM size, ICI topology

configs/models/*.json

- LLM / DLRM / DiT / GLIGEN
- sequence length, batch, parallelism

3. Launch the test sweep script

```
ray start --head --port=6379
cd neusim/run_scripts
./example_npusim.sh
```

Utilizes Ray to parallelize simulation jobs for architectural DSE sweep.

Output Files:

raw/
per-operator stats CSV
Aggregated model-level stats JSON

raw_None/
power / energy stats

carbon_.../
carbon footprint stats

Operator-Level Simulation (Demo)

Supported Primitive Operators

- **SA** — MatMul/Einsum, Conv2D, FlashAttention
- **VU** — Elementwise ops, normalization, softmax
- **ICI** — Collectives and explicit transfers
- **Fused Op** — FlashAttention

(More to be supported in the future!)

Single Operator Simulation

```
# chip_json =
json.load(open("configs/chips/tpuv4.json"))
cfg = ChipConfig.model_validate(chip_json)

op = ops_lib.create_einsum_op(
    [8, 4096], [4096, 14336],
    "MK;KN->MN", name="demo_matmul")

op =
op_analysis_lib.fill_operators_execution_info(
  [op], cfg)[0]

print(
  op.stats.execution_time_ns, op.stats.bounded_by
)
print(
  op.stats.tile_shapes_str, op.stats.num_sa_ops
)
```

Inspect Operator.stats

Latency breakdown

- execution_time_ns, bounded_by
- sa_time_ns, vu_time_ns
- memory_time_ns, vmem_time_ns, ici_time_ns

Memory and tiling details

- memory_traffic_bytes
- tile_shapes_str
- num_tiles
- max_vmem_demand_bytes
- num_mxu_ops # internal num_sa_ops

Energy breakdown

…

Workflow

Model-Level Simulation (Demo)

load JSON configs → set batch, seq len, parallelism, etc. → generate sharded operator graph → simulate each operator → dump CSV + aggregate stats

LLM Inference Simulation

```
# system_cfg, npu_cfg, model_cfg come from JSON files
cfg = {**system_cfg, **npu_cfg, **model_cfg}
config = LLMConfig.model_validate(cfg)

# set batch size and parallelism config
config.num_chips = 16
config.tensor_parallelism_degree = 4
config.pipeline_parallelism_degree = 4
config.input_seqlen = 8192
config.output_seqlen = 1024
config.global_batch_size = 16
config.microbatch_size_ici = 4

# launch simulation
ops, prefill_ops, decode_ops = LLMOpsGenerator(config).generate(
    dump_to_file=True, separate_prefill_decode=True)
```

Generated artifacts

- Simulation results for all operators (both prefill and decode phase for LLM inference)

Predefined Models

- LLM: Llama & DeepSeek
- DLRM
- DiT-XL
- GLIGEN

(Easily extendable; more to be supported in the future!)

Post-processing Operator Stats

- prefill TTFT
- decode TPOT
- pipeline-stage stats
- energy / carbon
- bottleneck analysis

Power Gating & DVFS Studies

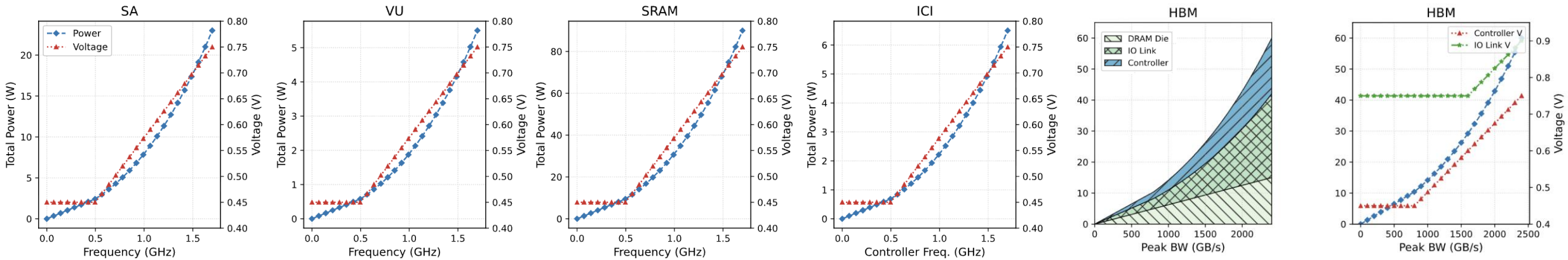
Goals:

1. Understand the power consumption of an NPU chip running different workloads
2. Study component-level utilization to quantify the potential power gating & DVFS opportunities
3. Simulate different power gating granularities to quantify potential benefits
4. Quantify the potential benefits assuming we can adjust the voltage and frequency of each component individually

Component-Level Power Model in NeuSim

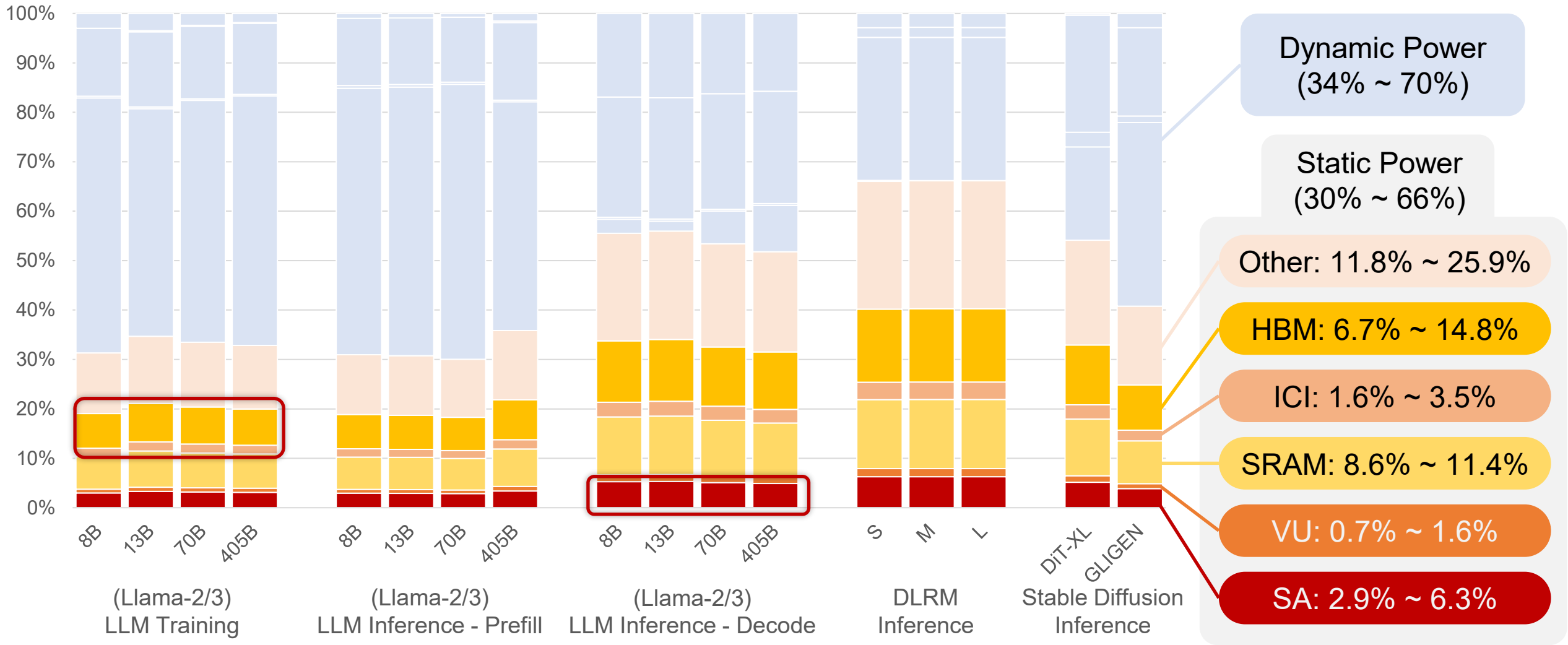
Simulation Pipeline

Generate iso-performance stats (peak V/f) → **Scale execution times based on new frequency** → **Look up V/f table of each component** → **Update power and energy stats**



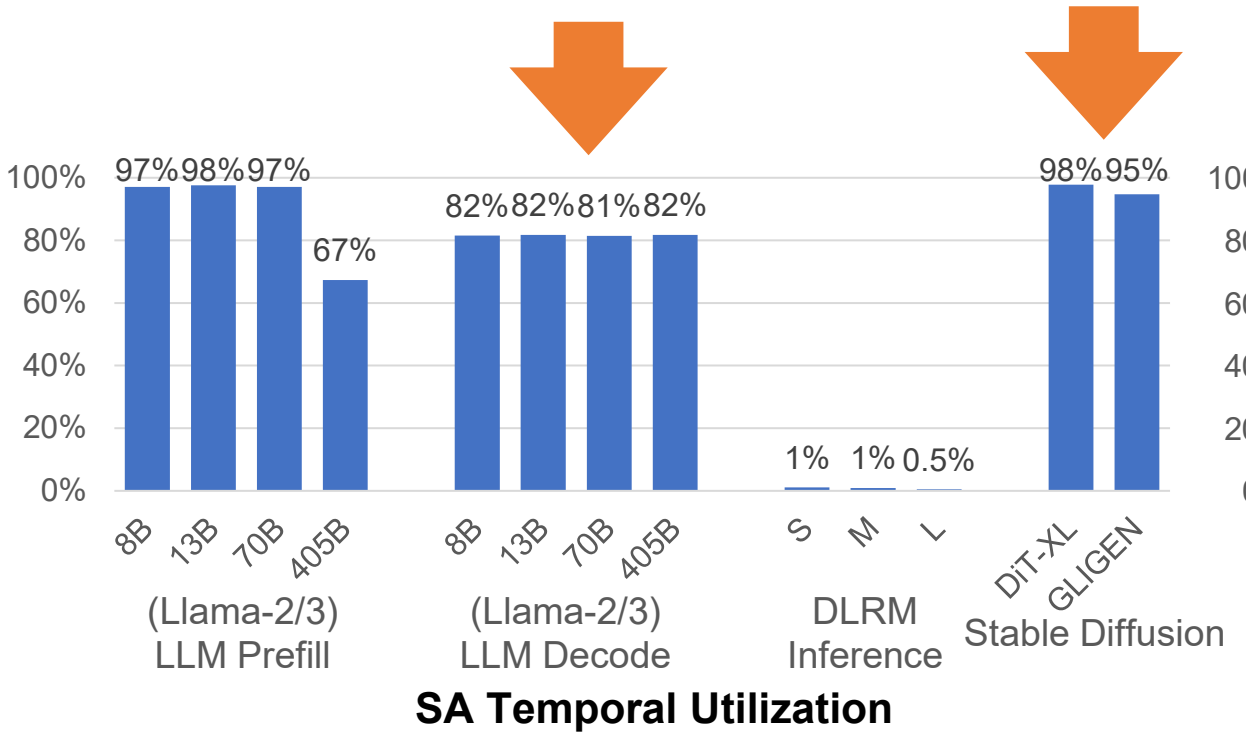
V/f relations of different NPU components based on synthesized results with ASAP7 PDK

NPU Power Consumption Breakdown



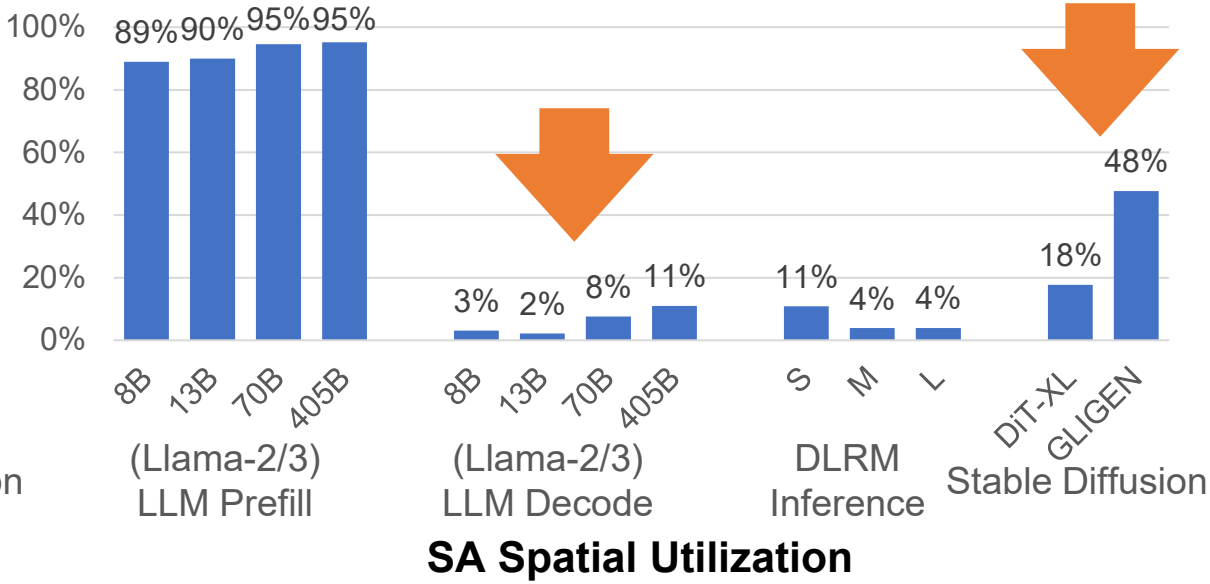
Static power accounts for 30% ~ 66% of energy consumption on an NPU chip

Analysis of NPU Component Utilization: Systolic Arrays



SA Temporal Utilization

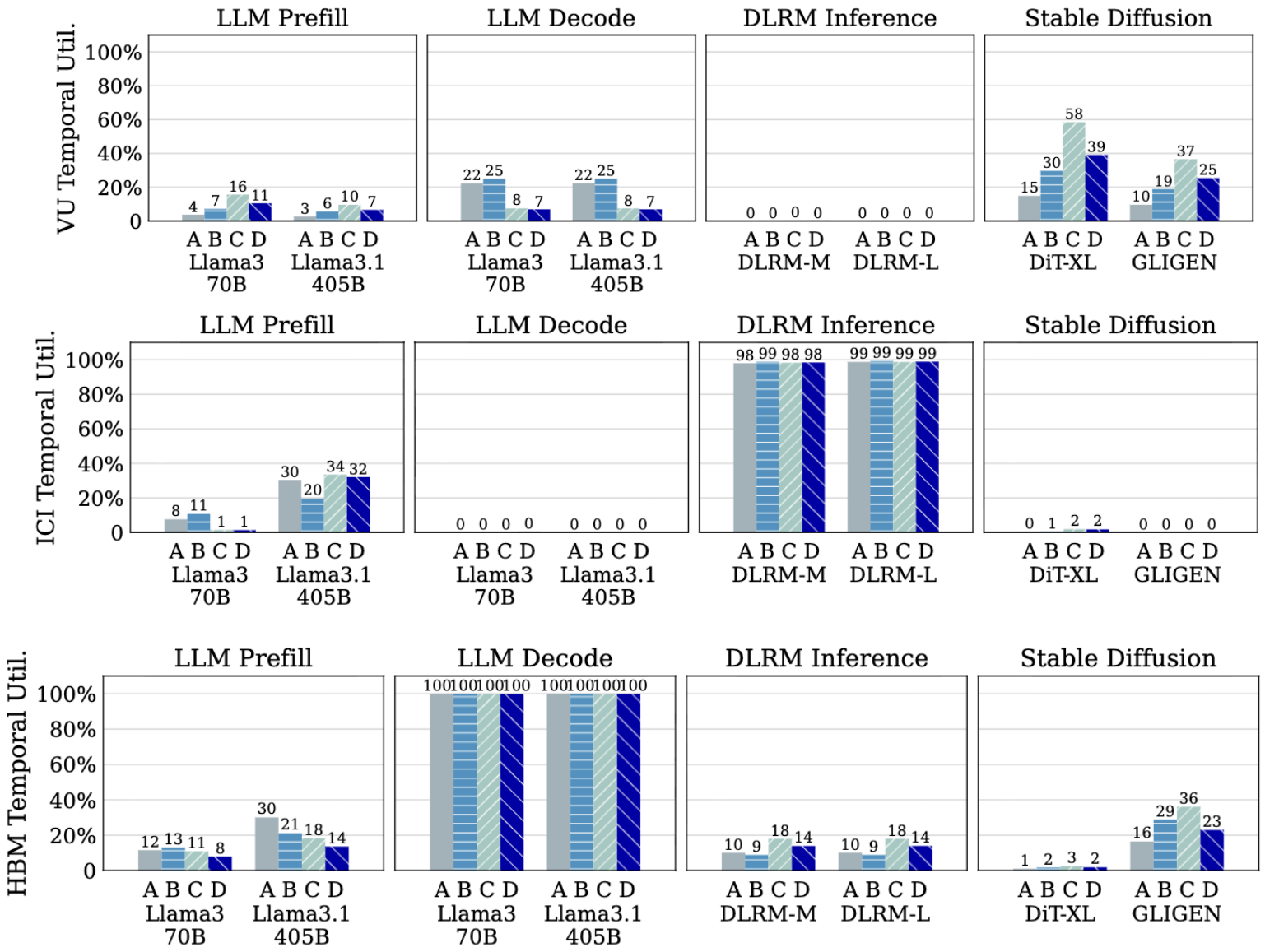
Low spatial utilization



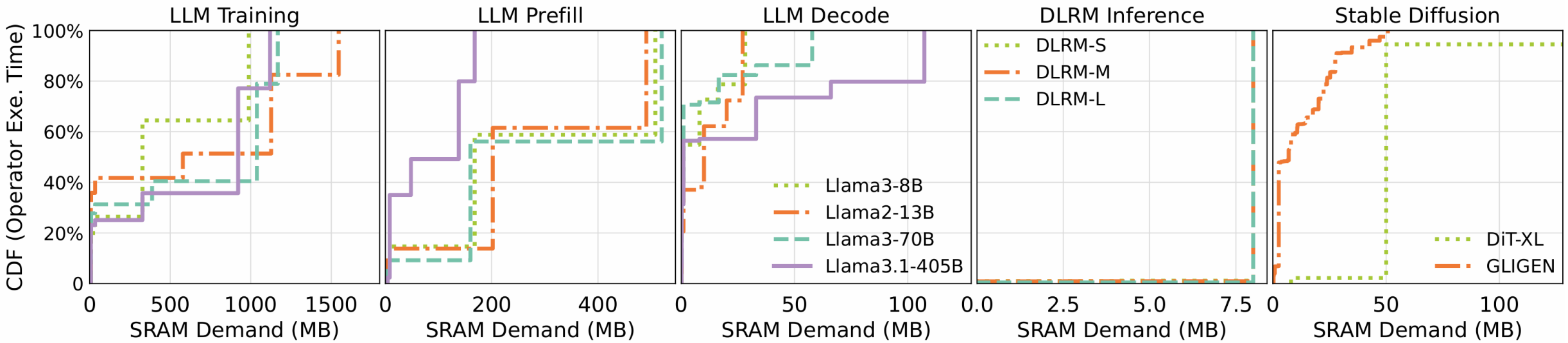
SA Spatial Utilization

Systolic array should be power gated at the processing element (PE) level

Analysis of NPU Component Utilization: Vector Units, HBM, and ICI



Analysis of NPU Component Utilization: SRAM Capacity



SRAM should be resized and power gated by the compiler based on the requirement of the ML program

Component-Level Power Gating for NPUs

Software-Managed Power Gating

Component Idleness Analysis

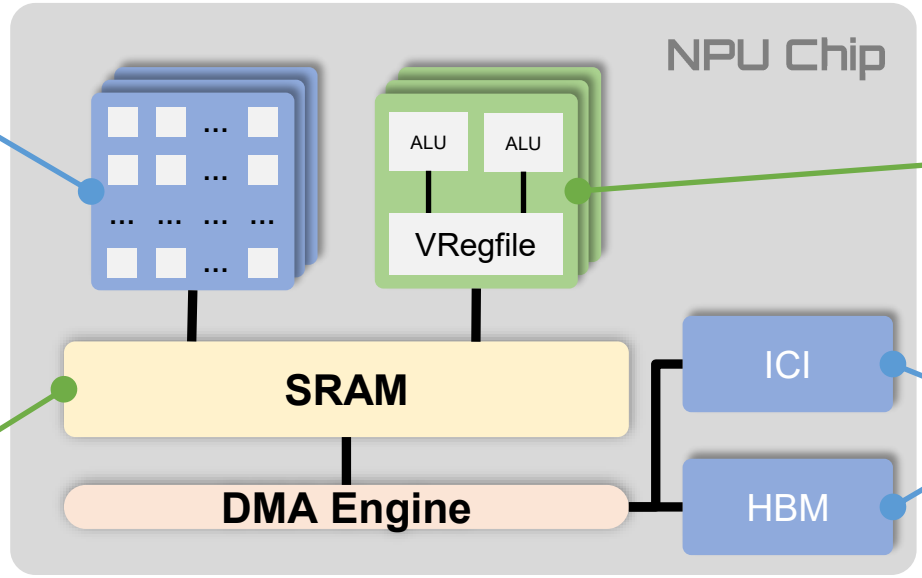
Power Gating Instruction Instrumentation

ISA Extension for Fine-Grained Power Management

Hardware Support for Power Gating

Systolic Array: HW-Managed Power Gating at PE-level

SRAM: SW-Managed Segment-wise Power Gating



Vector Unit: SW-Managed BET-based Power Gating

ICI & HBM: HW-Managed Idleness Detection

Power Gating Granularity Experimental Setup

Benchmarks

LLM Prefill	Llama3-8B, Llama2-13B, Llama3-70B, Llama3.1-405B; Input/Ouput Tokens: 4096/512
LLM Decode	
Recommendation	DLRM-S/M/L (20/45/98 GB)
Stable Diffusion	DiT-XL, GLIGEN; 512x512 image

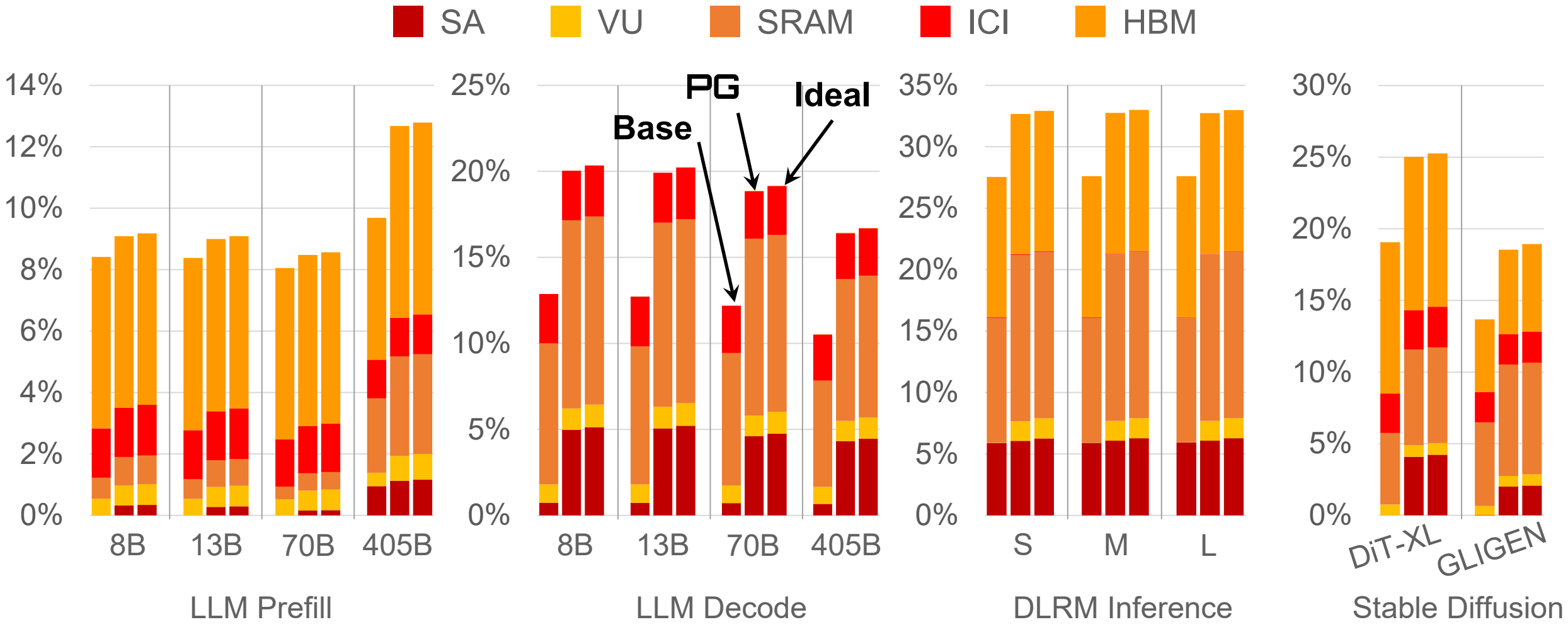
Experimental Setup

Base:
idleness detection-based power gating for each component

PG:
fine-grained power gating with SW/HW co-design

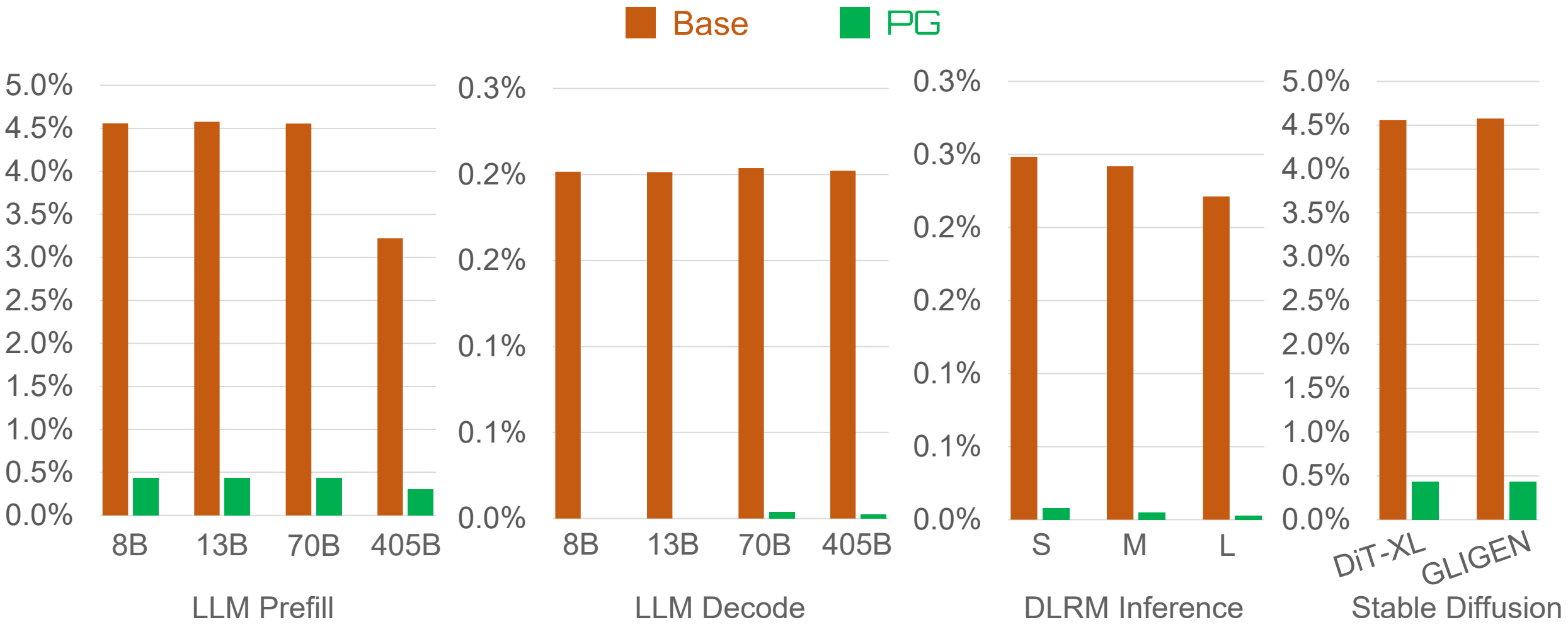
Ideal:
roofline design;
no wake-up delay;
all idle intervals are perfectly power-gated

End-to-End Energy Saving of ReGate



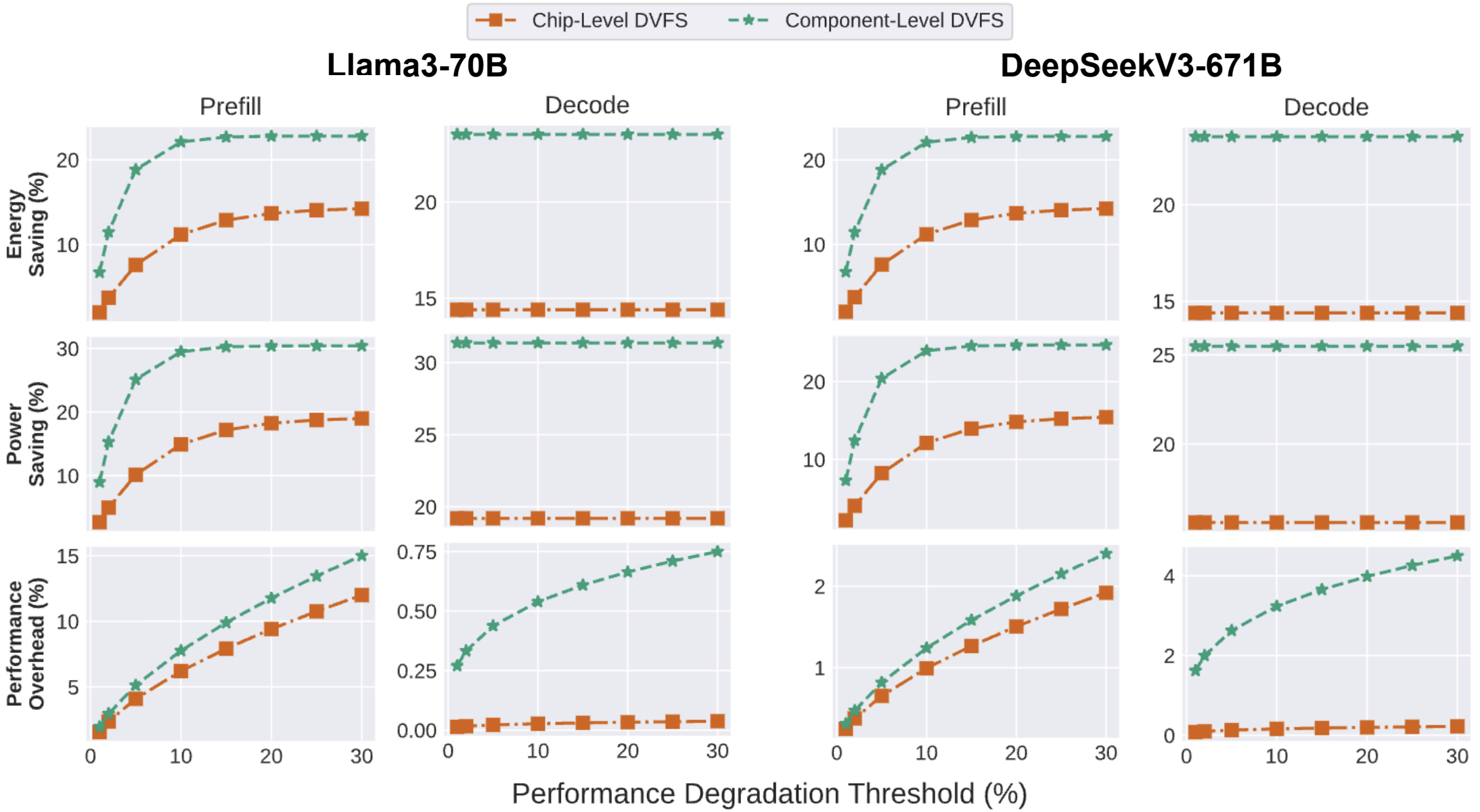
Power gating reduces the energy consumption of NPU chips by 8.5% ~ 32.8%

Performance Overhead of Power Gating

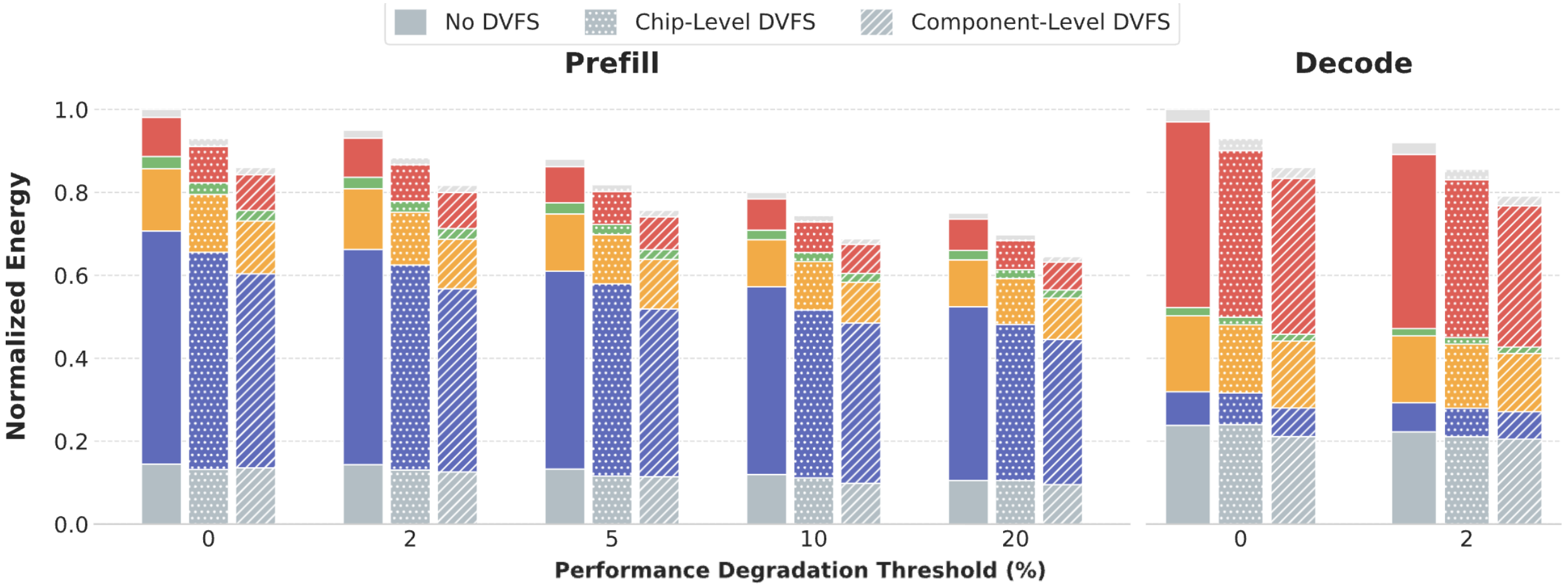


Power gating incurs negligible (less than 0.5%) performance overhead while significantly reducing energy consumption

Benefits of Component-Level DVFS



Benefit Breakdown of Component-Level DVFS



Energy saving breakdown of DVFS for Llama3-70B