Intelligence in Robotic Computing: Cross-Layer Co-Design for Resilient and Efficient Autonomous Machines

Zishen Wan

PhD Student @ Georgia Tech

w/ Profs. Arijit Raychowdhury, Tushar Krishna, Vijay Janapa Reddi




RoboArch Workshop @ MICRO, Senior Student Showcase Session, Nov. 3, 2024

Autonomous Machine Era

- Autonomous Machines on the Rise

Self-Driving Cars

Drones

Legged Robot

AR/VR

Embodied AI Robot

- Wide Application Potential

Package Delivery

Search & Rescue

Agriculture

Manufacture

Space

Challenges and Opportunities

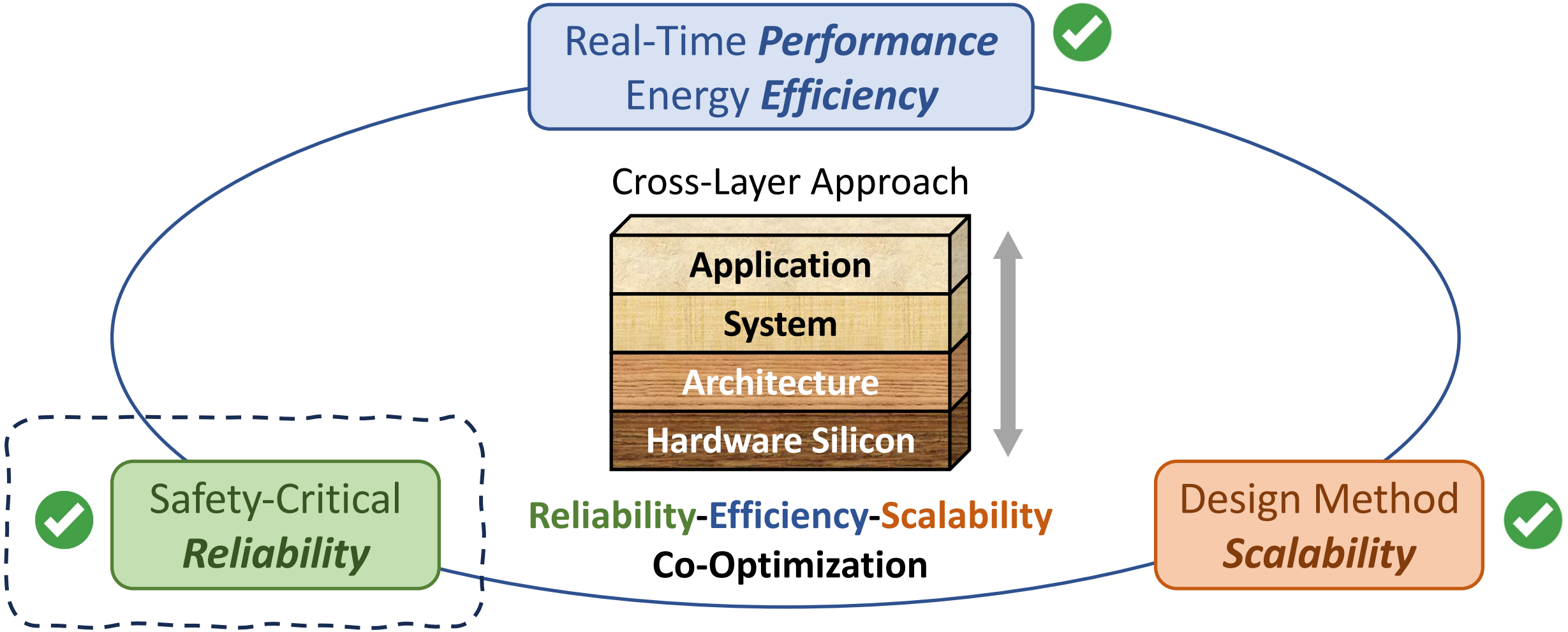
Real-Time ***Performance***
Energy ***Efficiency***

Autonomous Machines

Safety-Critical
Reliability

Design Method
Scalability

My Research: Autonomous Machine Computing



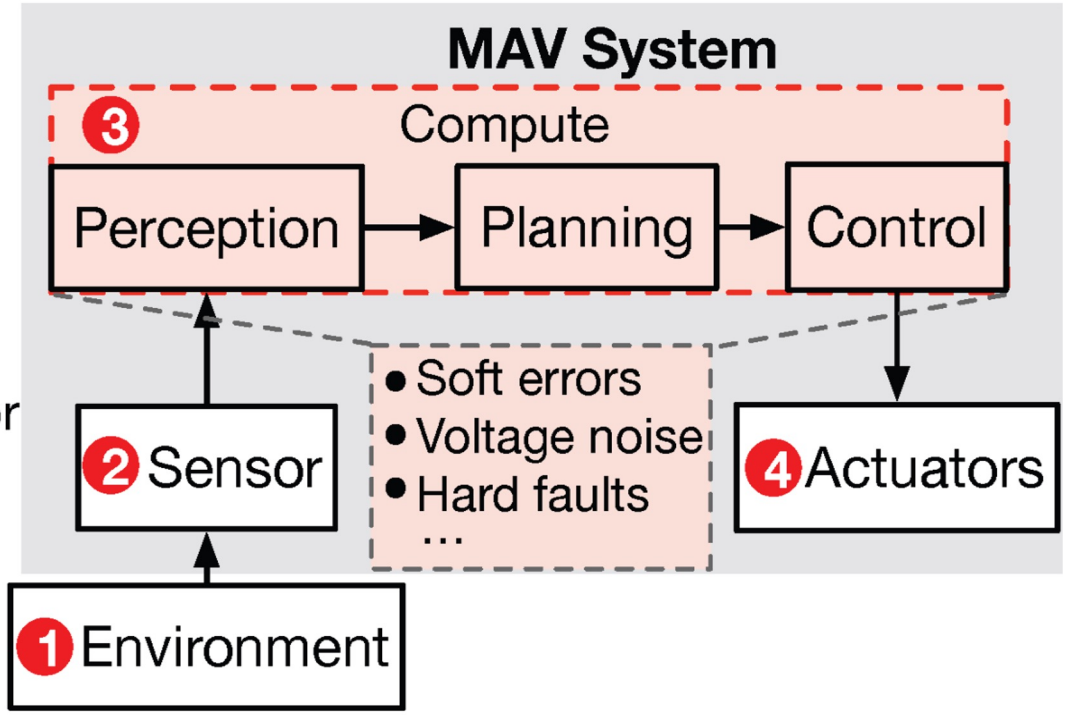
Motivation: Safety and Reliability


[1]


[2]

[1] Telsa Autopilot System Found Probably at Fault in 2018 Crash, The New York Times, 2021
[2] Surviving an In-Flight Anomaly: What Happened on Ingeuity's Sixth Flight, NASA Science, 2021

Motivation: Safety and Reliability



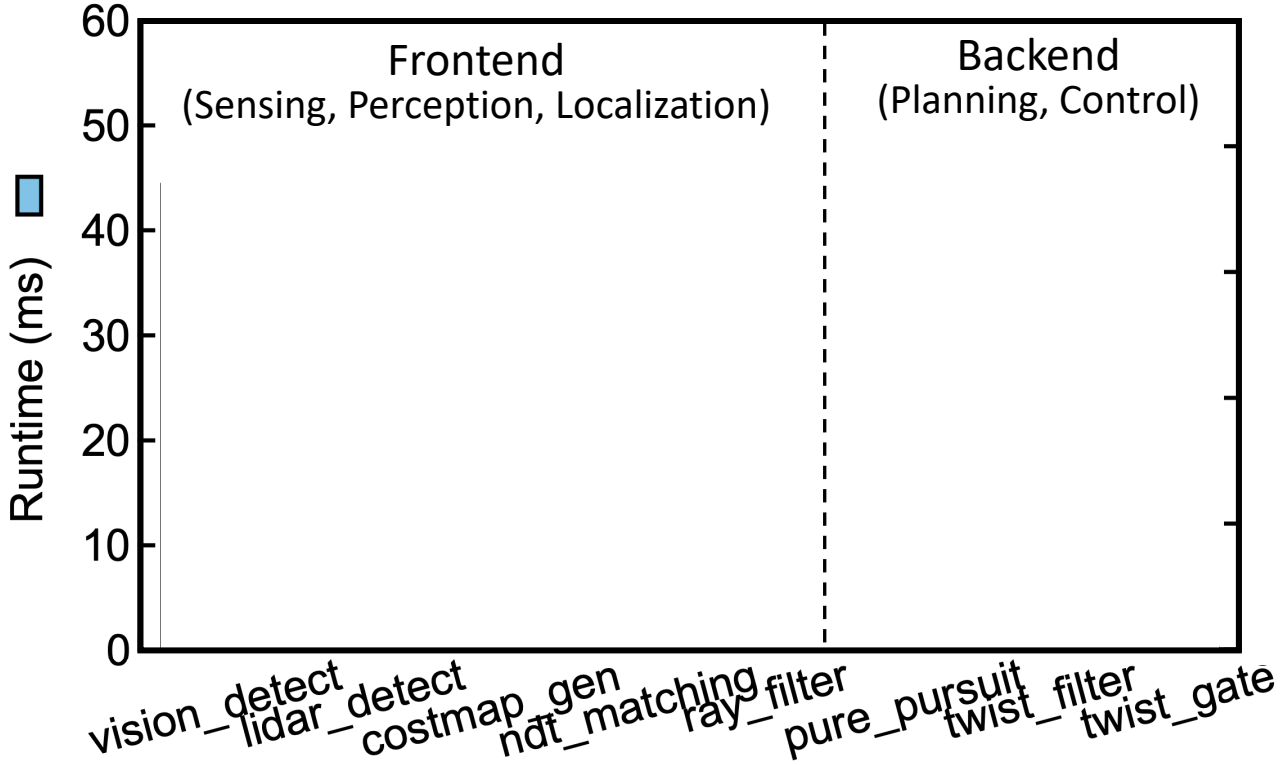
Research Question:

What's the resilience characteristics of autonomous machine systems?

How to provide high protection coverage with little cost?

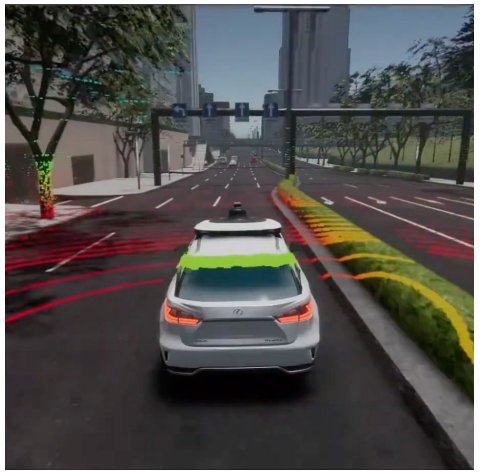
[@Comm. of the ACM'24]

Reliability Characterization – Auto Vehicle

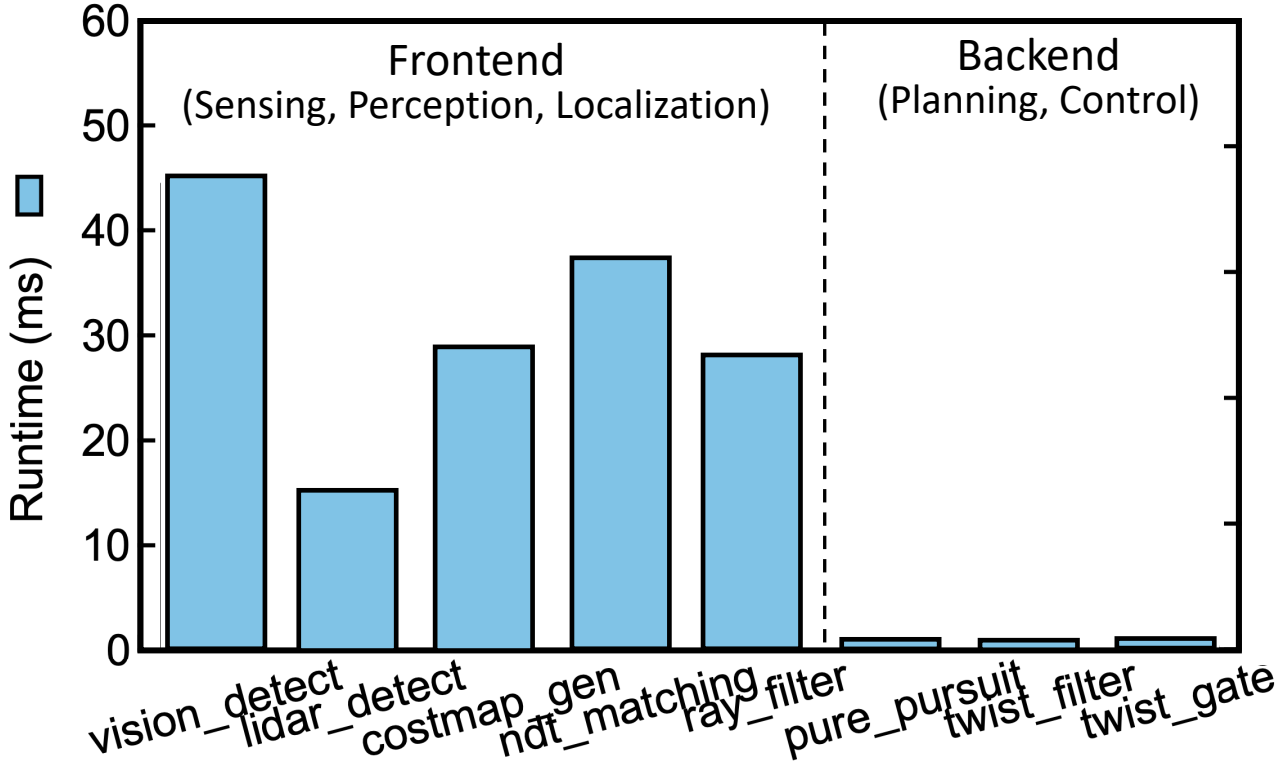


Experimental Setup

- Platform: Autonomous Vehicle (Autoware)

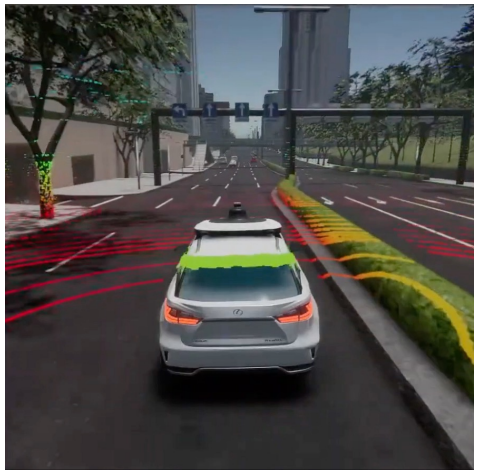


Reliability Characterization – Auto Vehicle



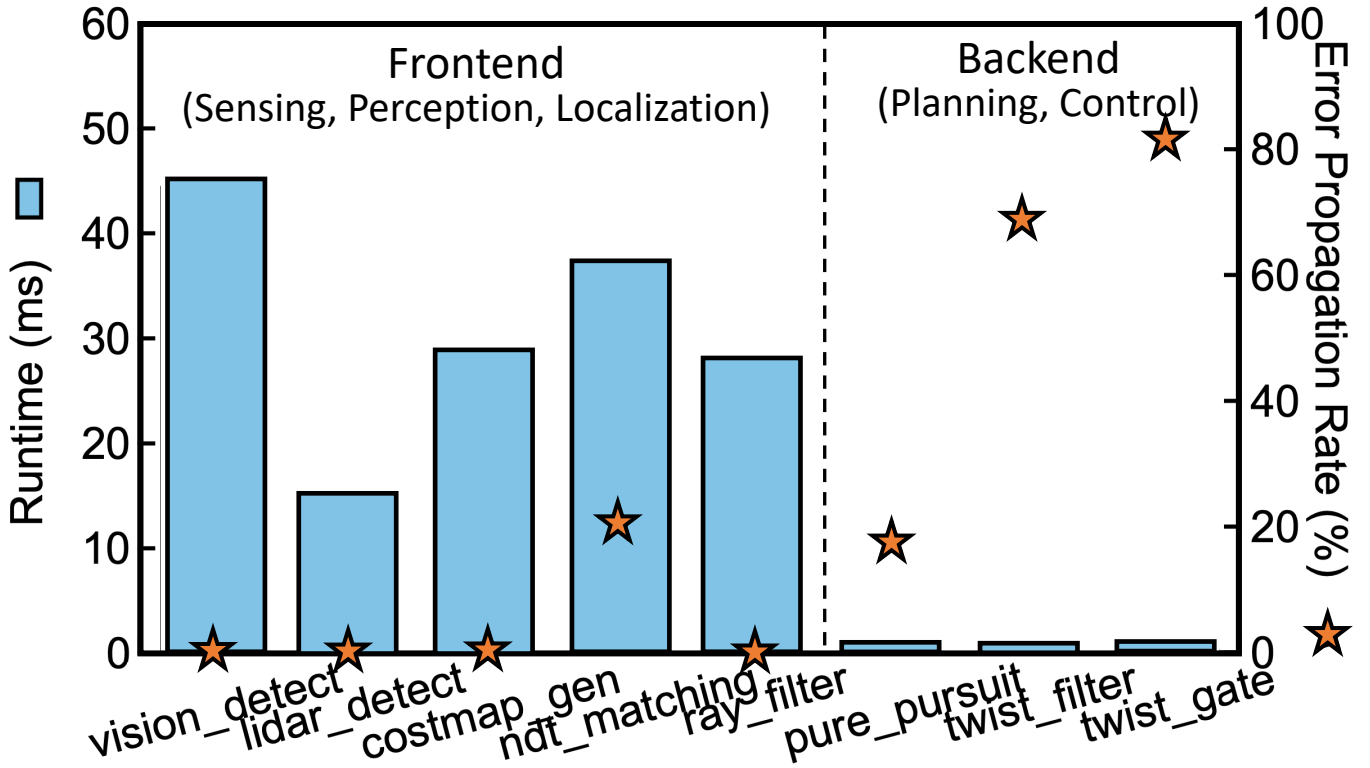
Experimental Setup

- Platform: Autonomous Vehicle (Autoware)



Insight: frontend high latency
backend low latency

Reliability Characterization – Auto Vehicle



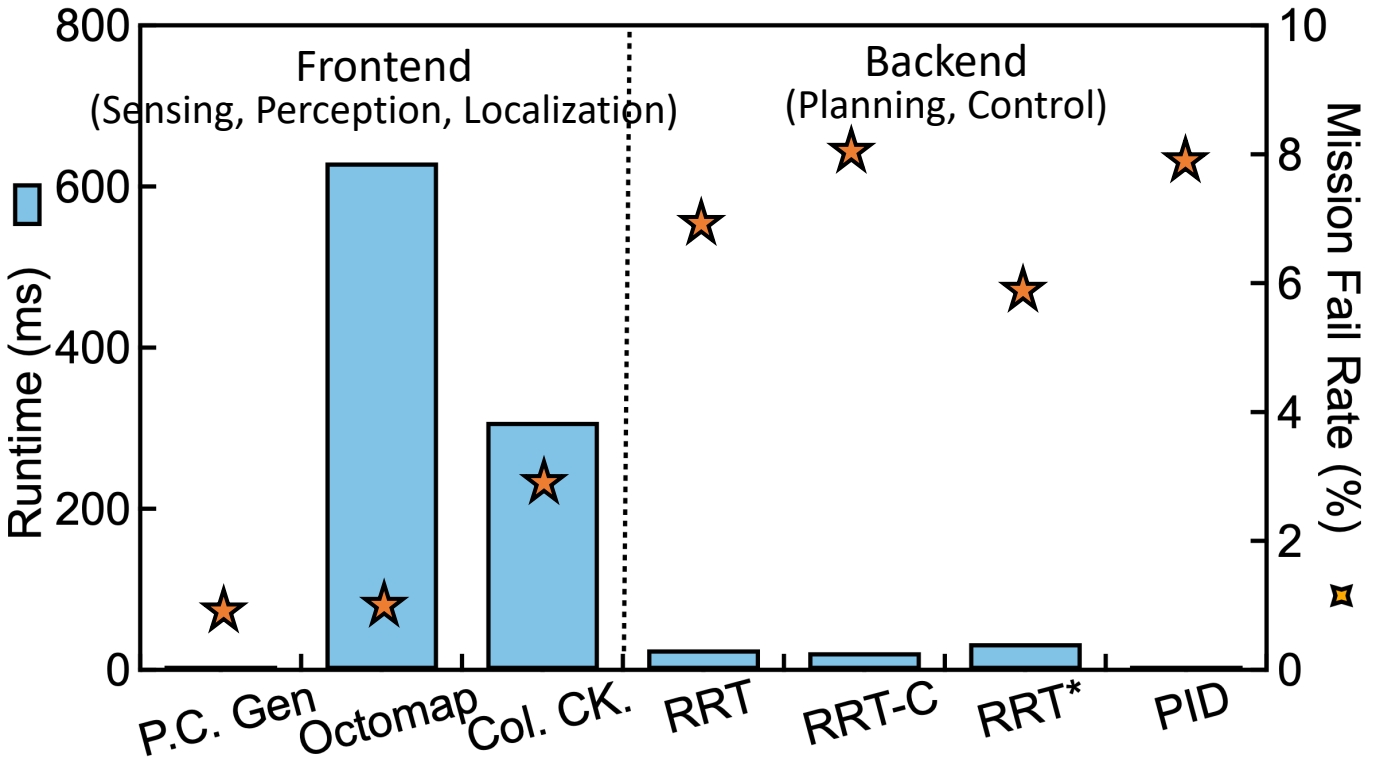
Experimental Setup

- Platform: Autonomous Vehicle (Autoware)
- Reliability: soft errors / single bit flip

Insight: frontend high latency, low vulnerability
backend low latency, high vulnerability

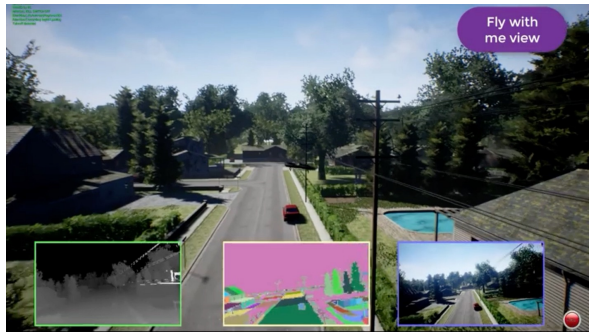
[@Comm. of the ACM'24]

Reliability Characterization - Auto Drone



Experimental Setup

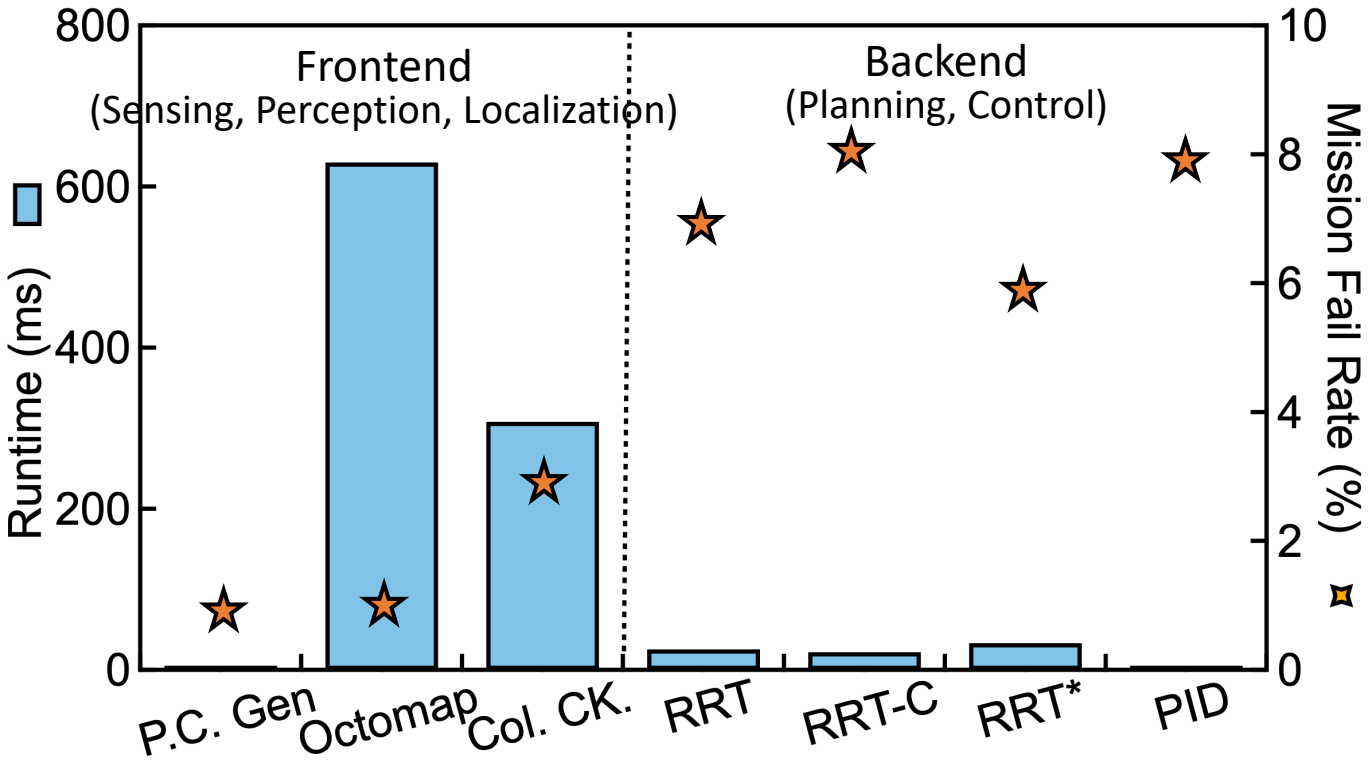
- Platform: Autonomous Drone (MAVBench)
- Reliability: soft errors / silent data corruption



Insight: frontend high latency, low vulnerability
backend low latency, high vulnerability

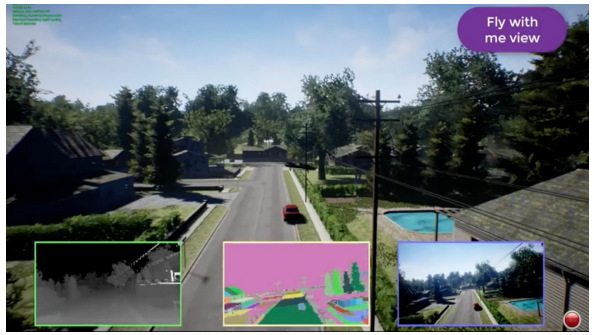
[@Comm. of the ACM'24]

Reliability Characterization - Auto Drone



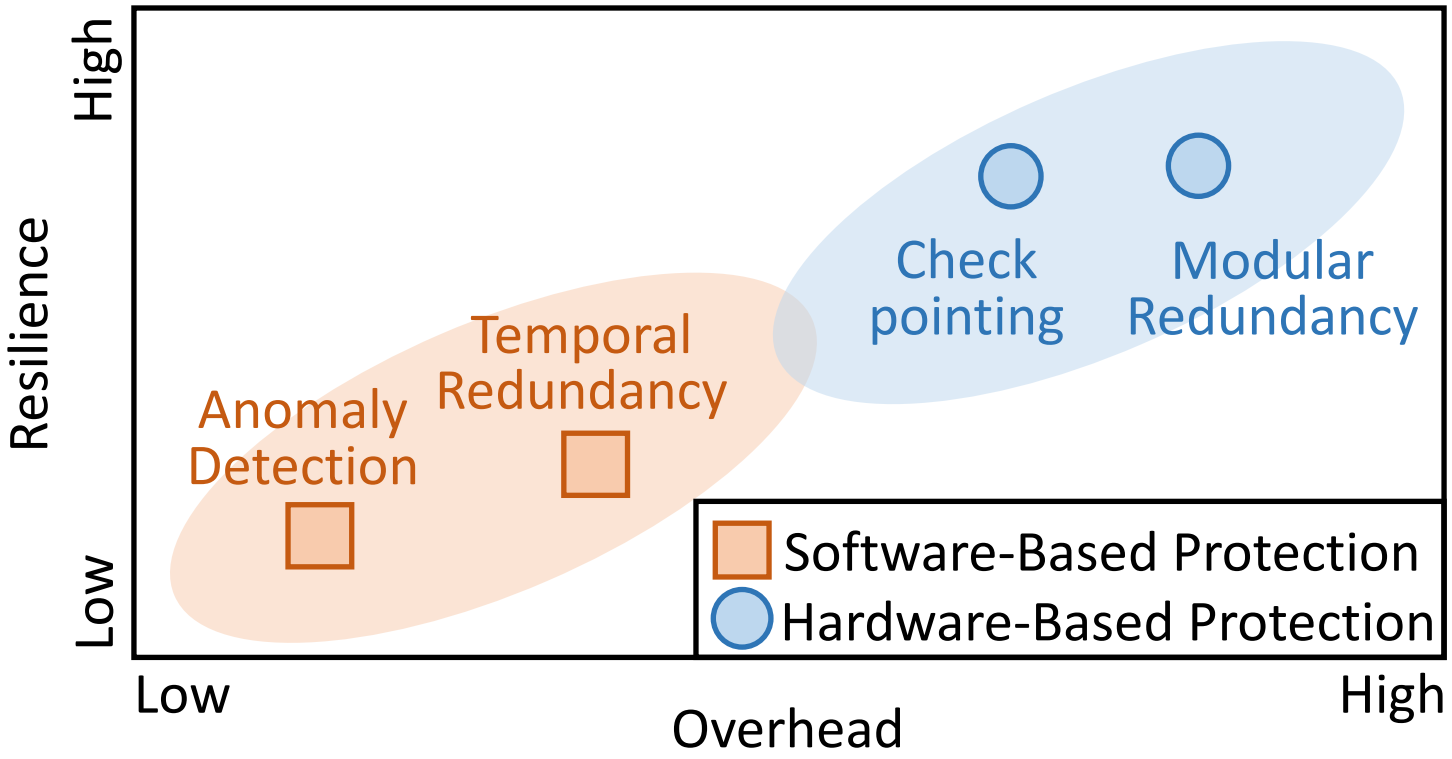
Experimental Setup

- Platform: Autonomous Drone (MAVBench)
- Reliability: soft errors / silent data corruption

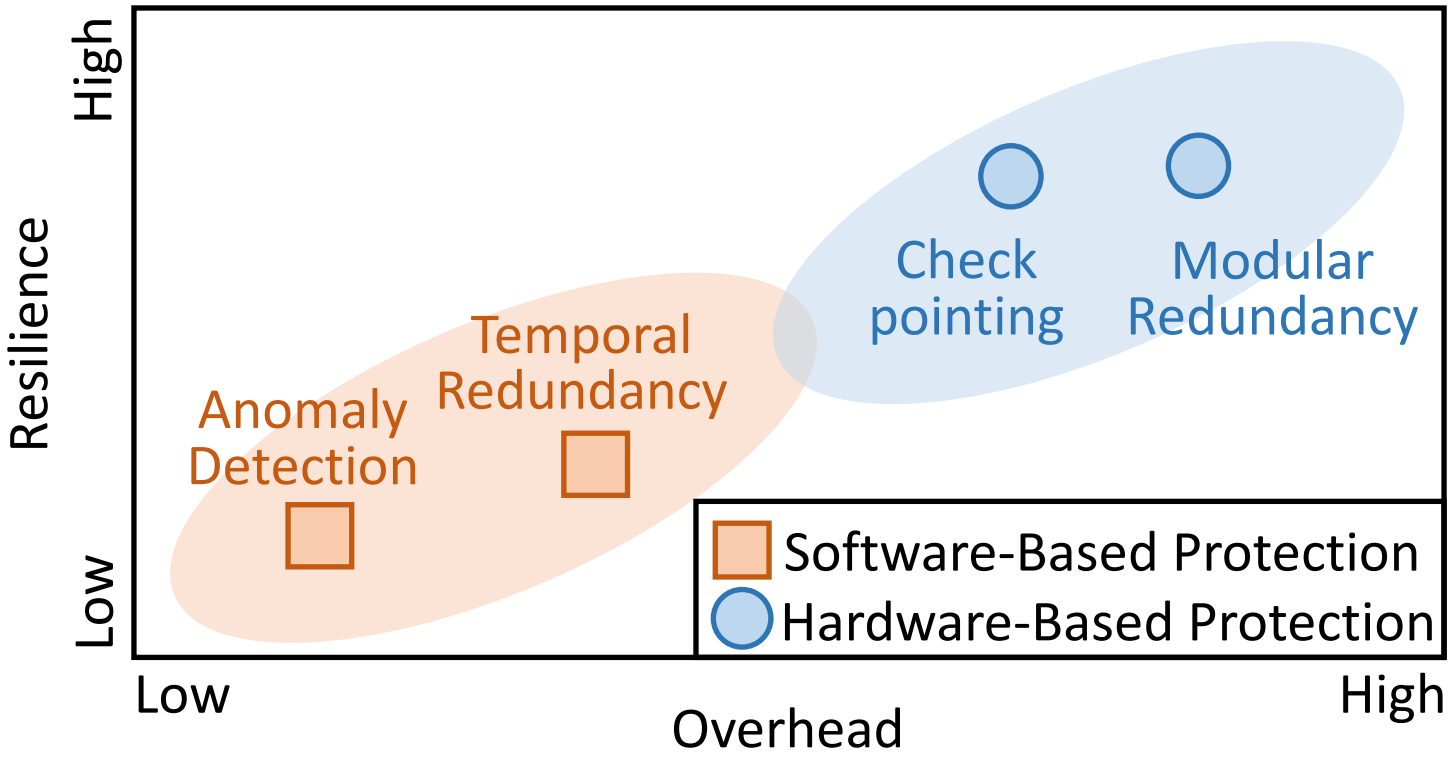


Insight: Autonomous machines exhibit inherent reliability variations

Design Landscape of Protection Techniques



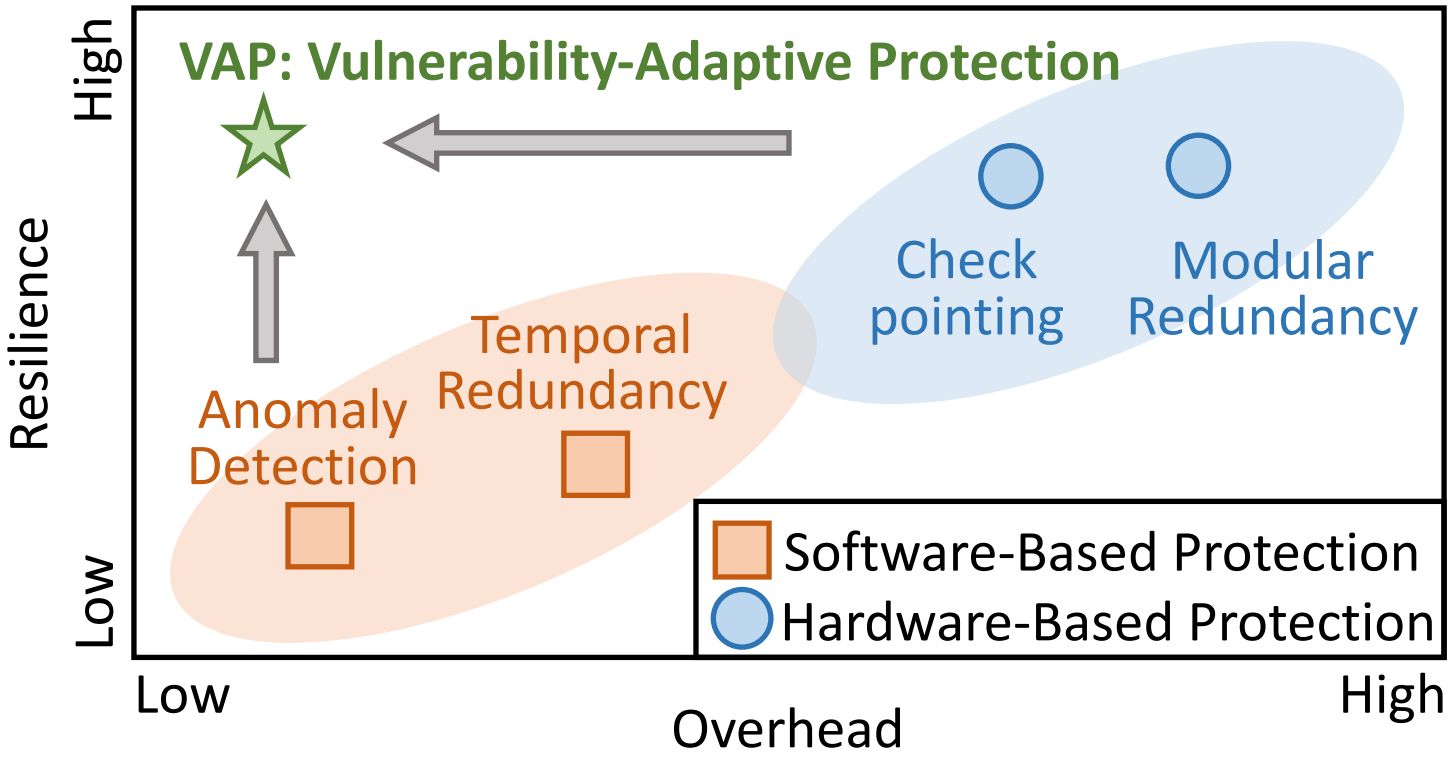
Challenge of Today's Protection Design



Challenge: Today's resiliency solutions are of "*one-size-fits-all*" nature: they use the same protection scheme throughout entire autonomous machine, bringing *trade-offs* between resiliency and cost

[@Comm. of the ACM'24]

Our Insight & Solution



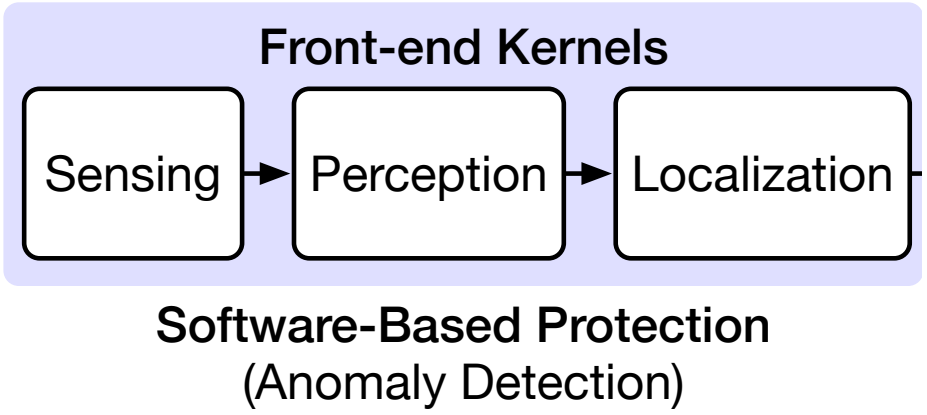
Insight & Solution: exploit the *inherent resilience variations* in autonomous machine system to conduct *vulnerability-adaptive protection (VAP)*

Vulnerability-Adaptive Protection

- **Design Principle**: the protection budget, be it spatially or temporally, should be allocated inversely proportionally to kernel inherent resilience

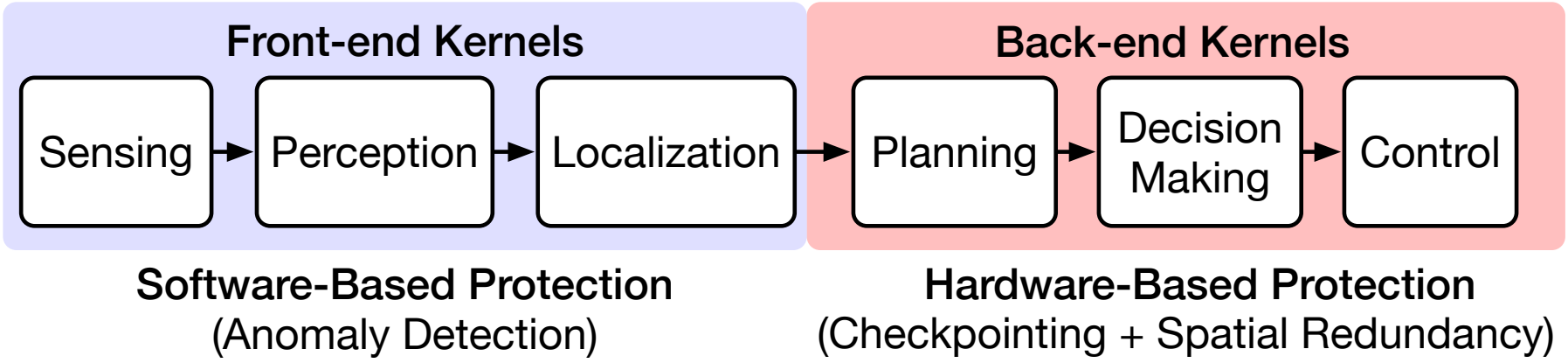
Vulnerability-Adaptive Protection

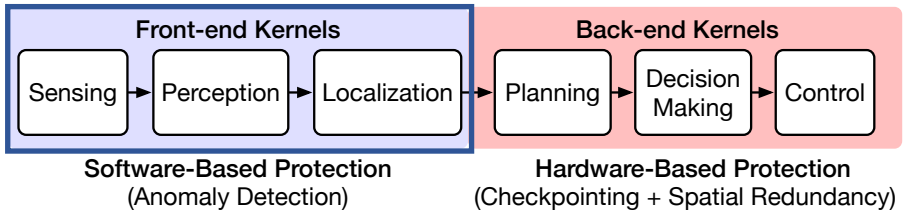
- **Design Principle**: the protection budget, be it spatially or temporally, should be allocated inversely proportionally to kernel inherent resilience
  - **Frontend**: low vulnerability -> lightweight software-based protection



Vulnerability-Adaptive Protection

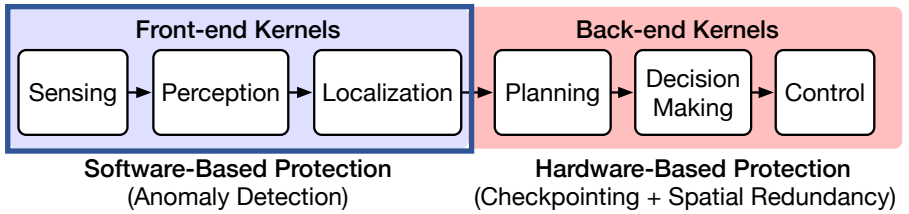
- **Design Principle**: the protection budget, be it spatially or temporally, should be allocated inversely proportionally to kernel inherent resilience
  - **Frontend**: low vulnerability -> lightweight software-based protection
  - **Backend**: high vulnerability -> more protection efforts, hardware-based protection





Frontend: Anomaly Detection

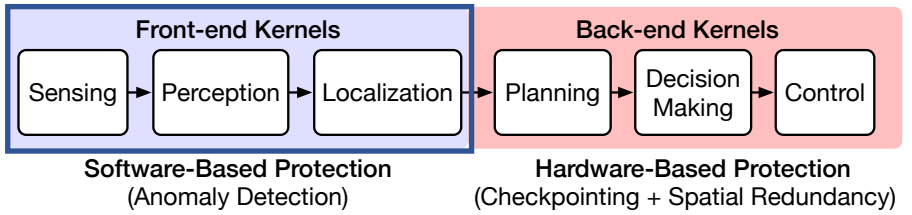
- **Frontend Insights**:
  - Strong **temporal consistency** of inputs and outputs
  - Inherent **error-masking** and error-attenuation capabilities
  - **Rare false positive** detection



Frontend: Anomaly Detection

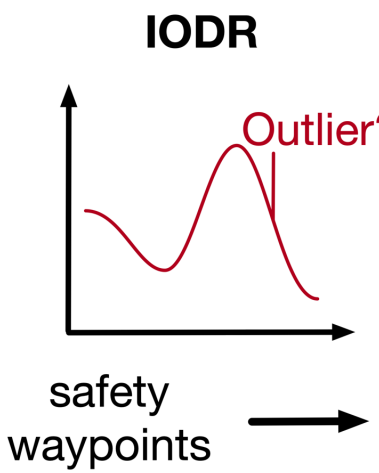
- **Frontend Insights**:
  - Strong **temporal consistency** of inputs and outputs
  - Inherent **error-masking** and error-attenuation capabilities
  - **Rare false positive** detection

IODR: Input Outlier Detection and Resetting



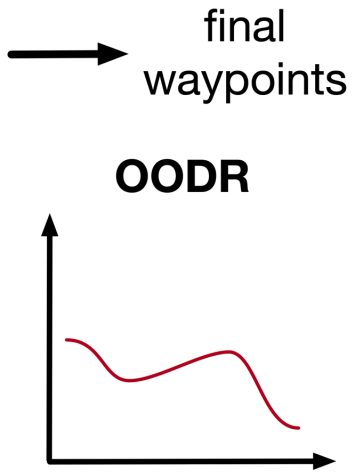
Frontend: Anomaly Detection

- **Frontend Insights**:
  - Strong **temporal consistency** of inputs and outputs
  - Inherent **error-masking** and error-attenuation capabilities
  - **Rare false positive** detection

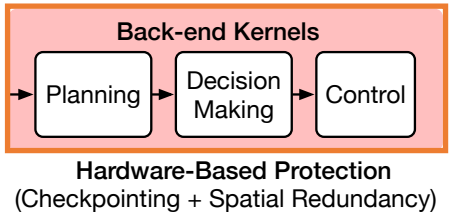


IODR: Input Outlier Detection and Resetting

```
void ChangeWp(const VelocitySetInfo& vs_info, float
safety_wp):
{
    double deceleration = 0.0;
    double velocity_set =0.0;
    cond1 = detect(vs_info);
    if (cond1)
    {
        final_wp = change(safety_wp);
    }
    else
    {
        final_wp = change(safety_wp);
    }
}
```

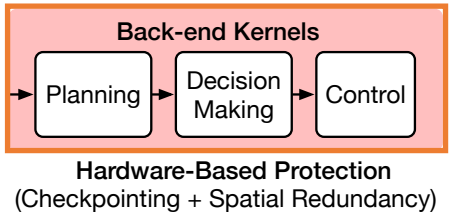


OODR: Output Outlier Detection and Resetting



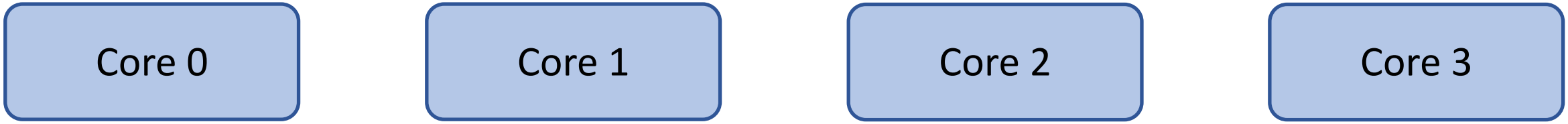
Backend: Redundancy & Checkpointing

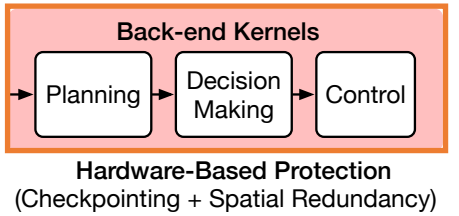
- **Backend Insights**:
  - **Critical** to errors
  - **Extremely lightweight** that do not involve complex computation
  - **More false positive** detection cases



Backend: Redundancy & Checkpointing

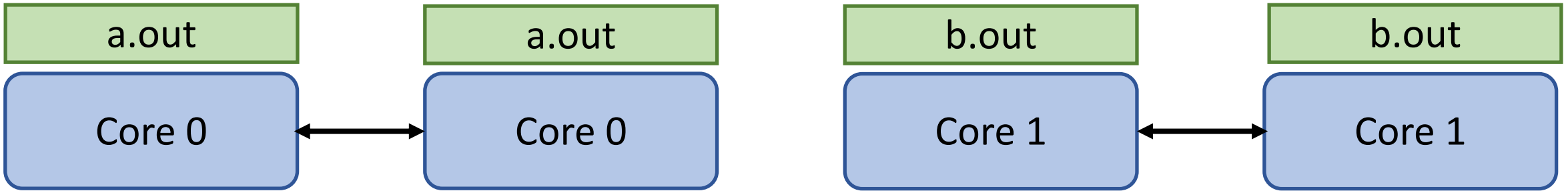
- **Backend Insights**:
  - **Critical** to errors
  - **Extremely lightweight** that do not involve complex computation
  - **More false positive** detection cases

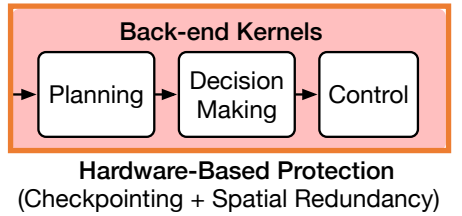




Backend: Redundancy & Checkpointing

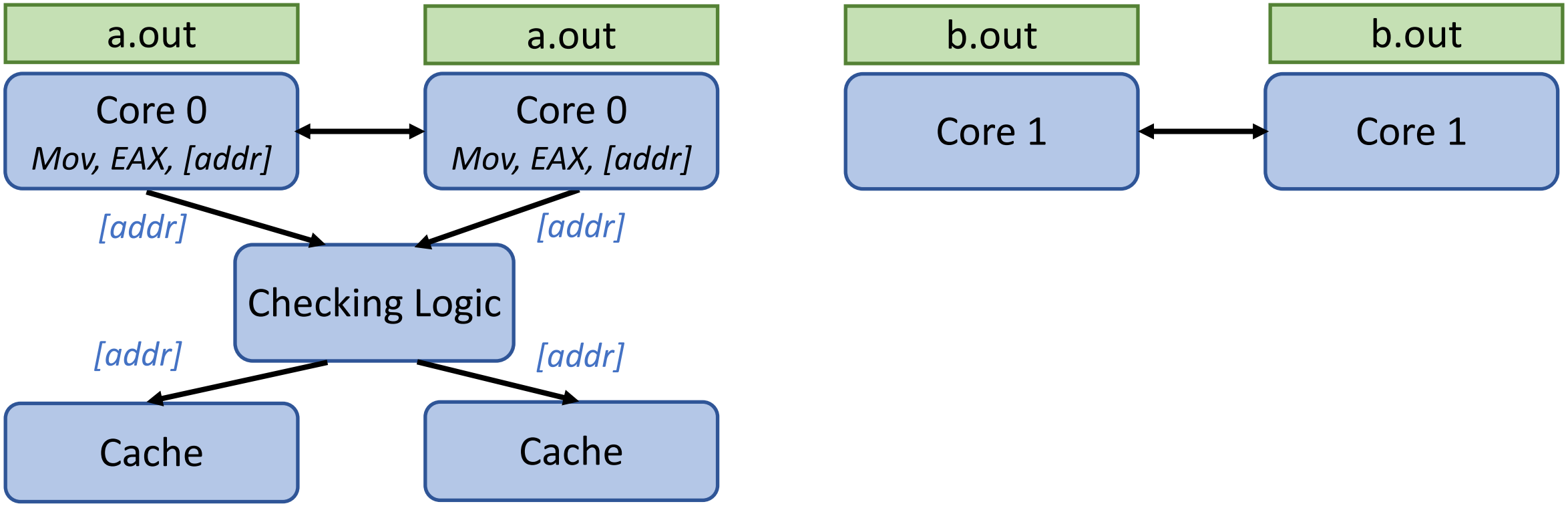
- **Backend Insights**:
  - **Critical** to errors
  - **Extremely lightweight** that do not involve complex computation
  - **More false positive** detection cases

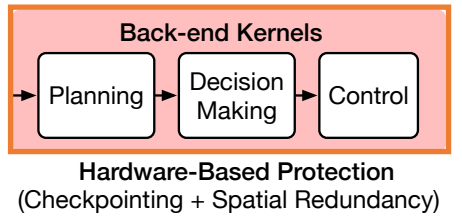




Backend: Redundancy & Checkpointing

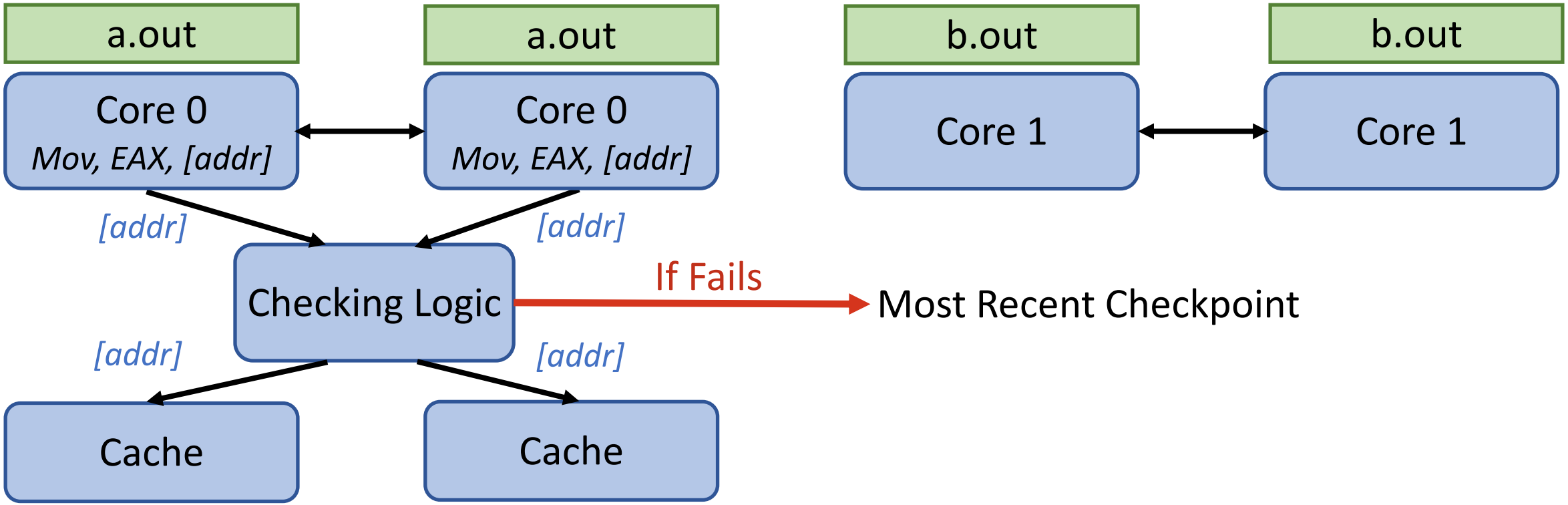
- **Backend Insights**:
  - **Critical** to errors
  - **Extremely lightweight** that do not involve complex computation
  - **More false positive** detection cases





Backend: Redundancy & Checkpointing

- **Backend Insights**:
  - **Critical** to errors
  - **Extremely lightweight** that do not involve complex computation
  - **More false positive** detection cases



VAP Overview

(VAP: Vulnerability-Adaptive Protection)

Insight: Inherent resilience variations of autonomy kernels

Design Principle: Protection budget allocated inversely proportionally to kernel inherent resilience

Evaluation – Autonomous Vehicle

Fault Protection Scheme	
Baseline	No Protection
Software	Anomaly Detection
	Temporal Redundancy
Hardware	Modular Redundancy
	Checkpointing
VAP (Ours) Frontend SW + Backend HW	

Experimental Setup

- Platform: Autonomous Vehicle (Autoware[1])
- Reliability: soft errors

[1] Kato et al, IEEE Micro, 2015

Evaluation – Autonomous Vehicle

Fault Protection Scheme		Resilience: Error Propagation Rate (%)
Baseline	**No Protection**	46.5
Software	**Anomaly Detection**	24.2
	Temporal Redundancy	11.7
Hardware	**Modular Redundancy**	0
	Checkpointing	0
VAP (Ours) Frontend SW + Backend HW		0

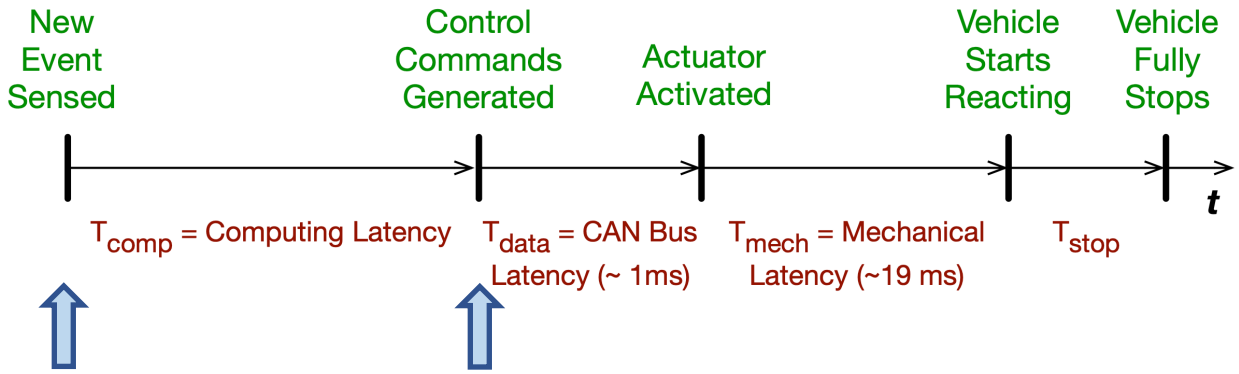
Experimental Setup

- Platform: Autonomous Vehicle (Autoware[1])
- Reliability: soft errors

Takeaway: VAP *improves resilience* and *reduces error propagation rate* by (1) leveraging inherent error-masking capabilities of front-end and (2) strengthening back-end resilience by hardware-based redundancy and checkpointing.

Evaluation – Autonomous Vehicle

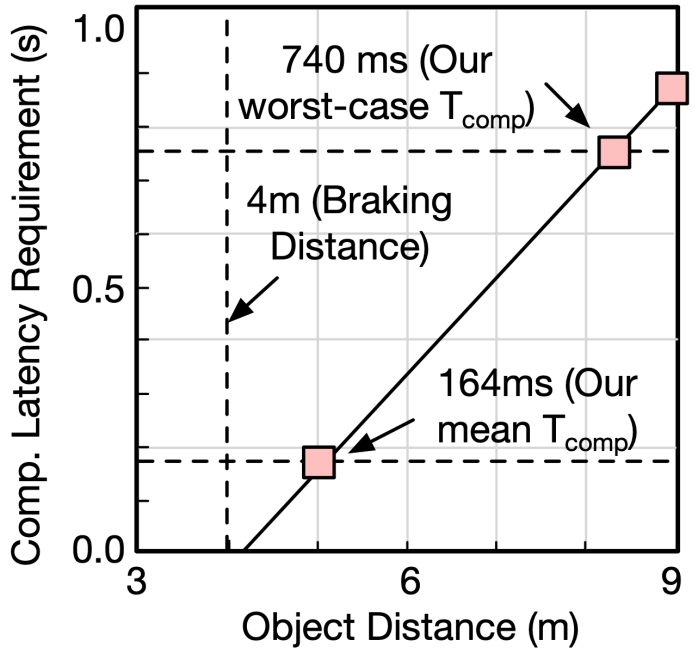
Fault Protection Scheme		Resilience	Latency and Object Distance
		Error Propagation Rate (%)	Compute Latency (ms)
Baseline	**No Protection**	46.5	164
Software	**Anomaly Detection**	24.2	245
	Temporal Redundancy	11.7	347
Hardware	**Modular Redundancy**	0	164
	Checkpointing	0	610
VAP (Ours) Frontend SW + Backend HW		0	173



Takeaway: VAP reduce end-to-end compute latency overhead.

Evaluation – Autonomous Vehicle

Fault Protection Scheme		Resilience	Latency and Object Distance	
		Error Propagation Rate (%)	Compute Latency (ms)	Object Avoidance Distance (m)
Baseline	**No Protection**	46.5	164	5.00
Software	**Anomaly Detection**	24.2	245	5.47
	Temporal Redundancy	11.7	347	6.05
Hardware	**Modular Redundancy**	0	164	5.00
	Checkpointing	0	610	7.56
VAP (Ours) Frontend SW + Backend HW		0	173	5.05



Takeaway: VAP reduce end-to-end compute latency overhead and reduce obstacle avoidance distance.

Evaluation – Autonomous Vehicle

Fault Protection Scheme		Resilience	Latency and Object Distance		Power Consumption and Driving Time	
		Error Propagation Rate (%)	Compute Latency (ms)	Object Avoidance Distance (m)	AD Component Power (W)	AD Energy Change (%)
Baseline	**No Protection**	46.5	164	5.00	175	-
Software	**Anomaly Detection**	24.2	245	5.47	175	+33.14
	Temporal Redundancy	11.7	347	6.05	175	+75.24
Hardware	**Modular Redundancy**	0	164	5.00	473	+170.29
	Checkpointing	0	610	7.56	324	+91.52
VAP (Ours) Frontend SW + Backend HW		0	173	5.05	175	+4.09

Takeaway: VAP reduce autonomous driving compute power and energy overhead.

Evaluation – Autonomous Vehicle

Fault Protection Scheme		Resilience	Latency and Object Distance		Power Consumption and Driving Time			
		Error Propagation Rate (%)	Compute Latency (ms)	Object Avoidance Distance (m)	AD Component Power (W)	AD Energy Change (%)	Driving Time (hour)	Revenue Loss (%)
Baseline	**No Protection**	46.5	164	5.00	175	-	7.74	-
Software	**Anomaly Detection**	24.2	245	5.47	175	+33.14	7.20	-6.99
	Temporal Redundancy	11.7	347	6.05	175	+75.24	6.62	-14.52
Hardware	**Modular Redundancy**	0	164	5.00	473	+170.29	5.59	-27.78
	Checkpointing	0	610	7.56	324	+91.52	6.42	-17.13
VAP (Ours) Frontend SW + Backend HW		0	173	5.05	175	+4.09	7.67	-0.92

Takeaway: VAP reduce autonomous driving compute power and energy overhead, thus enable longer driving time.

Evaluation – Autonomous Vehicle

Fault Protection Scheme		Resilience	Latency and Object Distance		Power Consumption and Driving Time				Cost
		Error Propagation Rate (%)	Compute Latency (ms)	Object Avoidance Distance (m)	AD Component Power (W)	AD Energy Change (%)	Driving Time (hour)	Revenue Loss (%)	Extra Dollar Cost
Baseline	**No Protection**	46.5	164	5.00	175	-	7.74	-	-
Software	**Anomaly Detection**	24.2	245	5.47	175	+33.14	7.20	-6.99	negligible
	Temporal Redundancy	11.7	347	6.05	175	+75.24	6.62	-14.52	negligible
Hardware	**Modular Redundancy**	0	164	5.00	473	+170.29	5.59	-27.78	(CPU+GPU)x2
	Checkpointing	0	610	7.56	324	+91.52	6.42	-17.13	(CPU+GPU)x1
VAP (Ours) Frontend SW + Backend HW		0	173	5.05	175	+4.09	7.67	-0.92	negligible

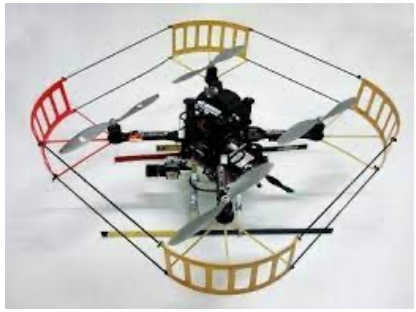
Takeaway: VAP reduces compute latency, energy and system overhead by taking advantage of (1) low cost and false-positive detection in front-end and (2) low latency in back-end. Conventional "one-size-fits-all" techniques are limited by tradeoffs in resilience and overhead.

Evaluation – Autonomous Drone

Fault Protection Scheme		Resilience	Latency and Flight Time			Power Consumption and Flight Energy				Cost
		Mission Failure Rate (%)	Compute Latency (ms)	Avg. Flight Velocity (m/s)	Mission Time (s)	Compute Power (W)	Mission Energy (kJ)	Num. of Missions	Endurance Reduction (%)	Extra Dollar Cost
Baseline	**No Protection**	12.20	871	2.79	107.53	15	60.69	5.62	-	-
Software	**Anomaly Detection**	6.44	1201	2.51	119.52	15	66.79	5.05	-10.04	negligible
	Temporal Redundancy	3.02	1924	2.14	140.18	15	78.34	4.31	-23.30	negligible
Hardware	**Modular Redundancy**	0	871	2.74	109.49	45	63.13	5.34	-3.79	TX2 x 2
	Checkpointing	0	3458	1.75	171.43	30	96.76	3.49	-37.90	TX2 x 1
VAP (Ours) Frontend SW + Backend HW		0	897	2.77	108.30	15	60.52	5.58	-0.72	negligible

Experimental Setup

- Platform: Autonomous Drone (MAVBench[2])
- Reliability: soft errors



[2] Boroujerdian et al, MICRO, 2018

Evaluation – Autonomous Drone

Fault Protection Scheme		Resilience	Latency and Flight Time			Power Consumption and Flight Energy				Cost
		Mission Failure Rate (%)	Compute Latency (ms)	Avg. Flight Velocity (m/s)	Mission Time (s)	Compute Power (W)	Mission Energy (kJ)	Num. of Missions	Endurance Reduction (%)	Extra Dollar Cost
Baseline	**No Protection**	12.20	871	2.79	107.53	15	60.69	5.62	-	-
Software	**Anomaly Detection**	6.44	1201	2.51	119.52	15	66.79	5.05	-10.04	negligible
	Temporal Redundancy	3.02	1924	2.14	140.18	15	78.34	4.31	-23.30	negligible
Hardware	**Modular Redundancy**	0	871	2.74	109.49	45	63.13	5.34	-3.79	TX2 x 2
	Checkpointing	0	3458	1.75	171.43	30	96.76	3.49	-37.90	TX2 x 1
VAP (Ours) Frontend SW + Backend HW		0	897	2.77	108.30	15	60.52	5.58	-0.72	negligible

Takeaway: For small form factor autonomous machines (e.g., drones), extra compute latency and payload weight brought by fault protection schemes impact drone safe flight velocity, further impacting end-to-end system mission time, mission energy, and flight endurance.

Evaluation – Autonomous Drone

Fault Protection Scheme		Resilience	Latency and Flight Time			Power Consumption and Flight Energy				Cost
		Mission Failure Rate (%)	Compute Latency (ms)	Avg. Flight Velocity (m/s)	Mission Time (s)	Compute Power (W)	Mission Energy (kJ)	Num. of Missions	Endurance Reduction (%)	Extra Dollar Cost
Baseline	**No Protection**	12.20	871	2.79	107.53	15	60.69	5.62	-	-
Software	**Anomaly Detection**	6.44	1201	2.51	119.52	15	66.79	5.05	-10.04	negligible
	Temporal Redundancy	3.02	1924	2.14	140.18	15	78.34	4.31	-23.30	negligible
Hardware	**Modular Redundancy**	0	871	2.74	109.49	45	63.13	5.34	-3.79	TX2 x 2
	Checkpointing	0	3458	1.75	171.43	30	96.76	3.49	-37.90	TX2 x 1
VAP (Ours) Frontend SW + Backend HW		0	897	2.77	108.30	15	60.52	5.58	-0.72	negligible

Takeaway: VAP generalizes well to small-scale drone system *with improved resilience and negligible overhead*. By contrast, the large overhead from conventional "one-size-fits-all" protection results in severer performance degradation in SWaP-constrained systems.

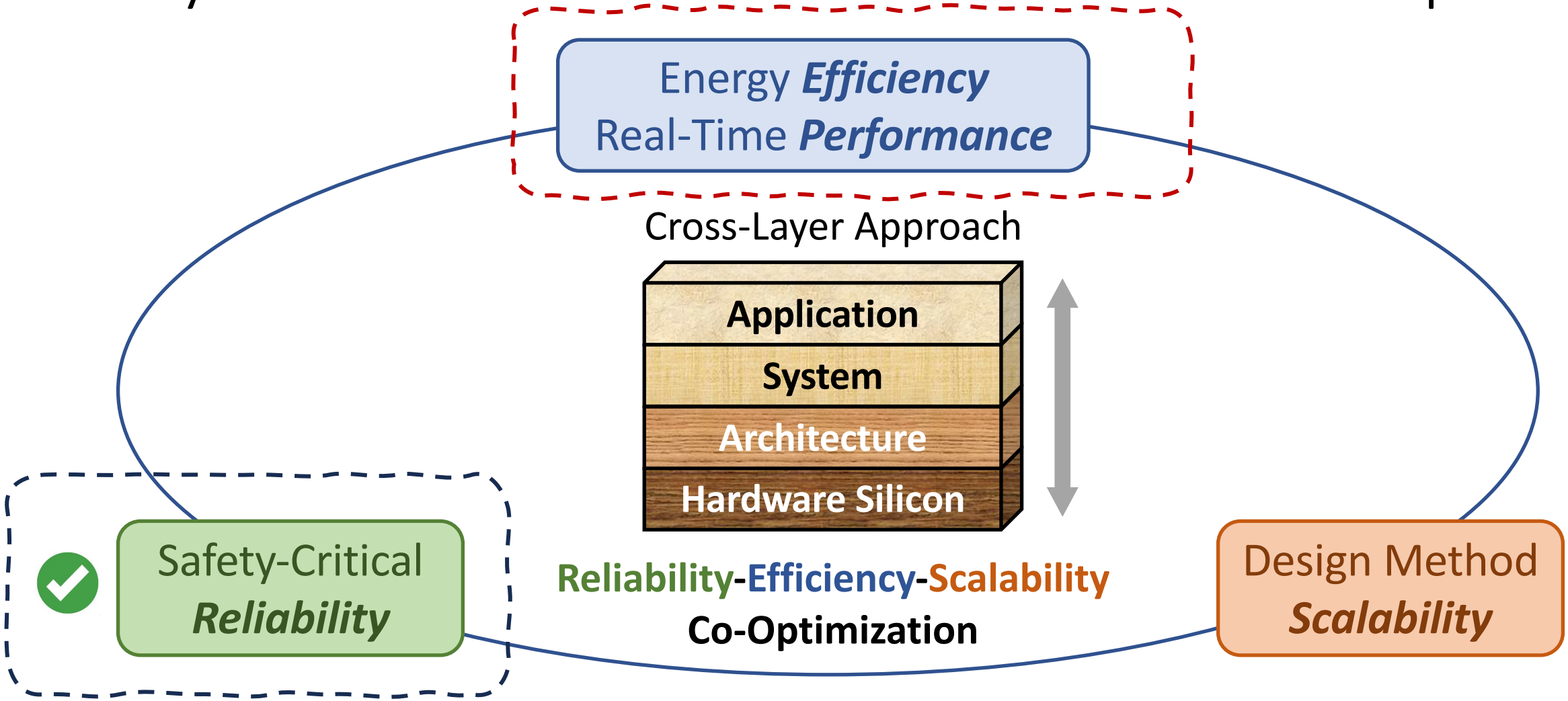
[@Comm. of the ACM'24]

Key Takeaways:

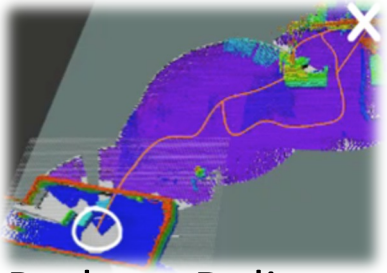
Autonomous machines have inherent resilience variations: frontend reliable, backend vulnerable

Vulnerability-adaptive protection:
allocate protection budget based on kernel resilience

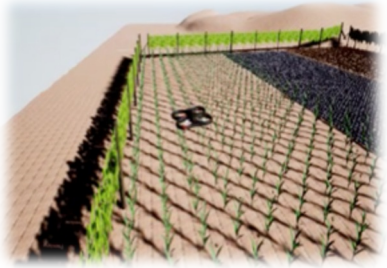
My Research: Autonomous Machine Computing



Motivation: Need Energy-Efficient Computing

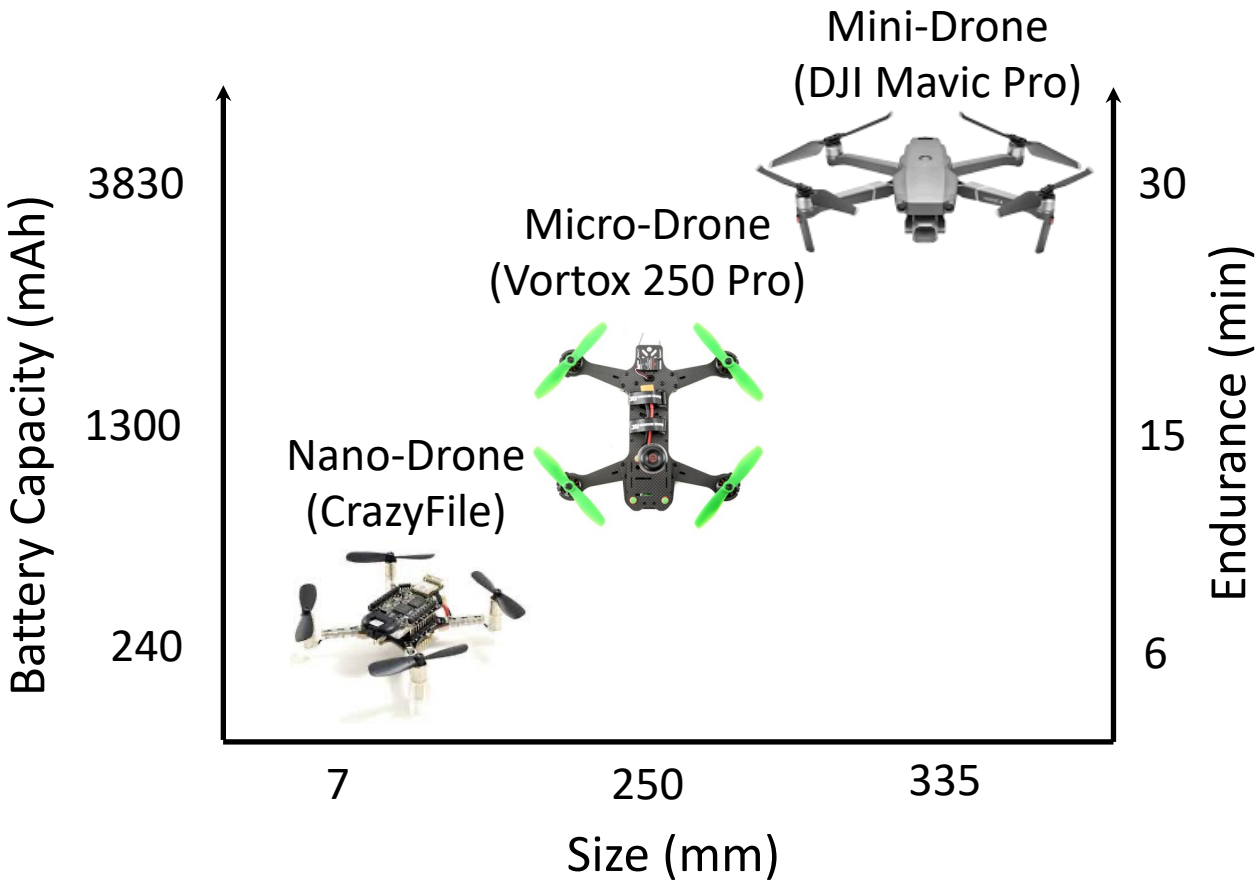


Package Delivery



Scanning

Search and Rescue



Drones are size, weight, and power (SWaP) constrained

Imagine a Deployment Scenario

For a single flight mission:

- Average fight energy: 75.80 J

Under battery support:

- Total num of completed missions: 40

For a single flight mission:

- Average fight energy: **61.42 J**

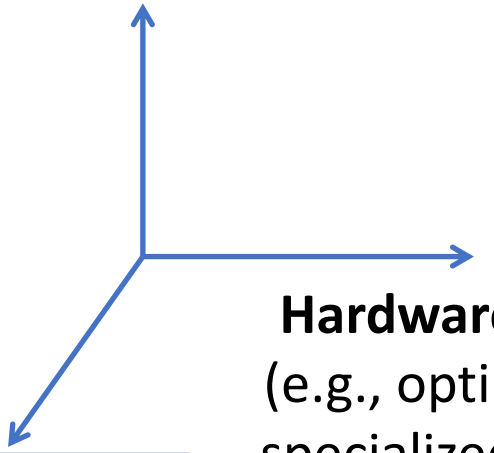
Under battery support:

- Total num of completed missions: **49**

Our Goal!

Low-Voltage Processing Reduces Energy

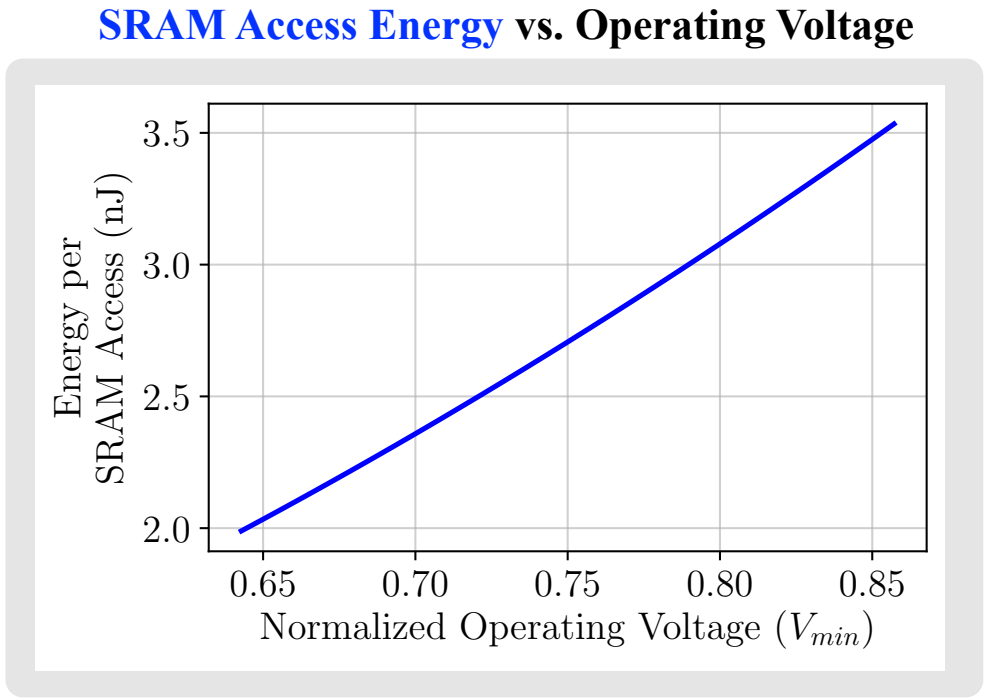
Software Optimization
(e.g., quantization, sparsity)



Hardware Optimization
(e.g., optimized dataflow, specialized compute unit)

Lower processor operating voltage

Energy ∝ Voltage ^{2}

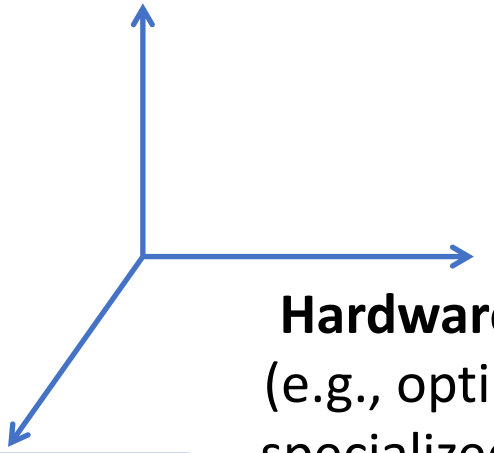


Data measured from 14nm FinFET SRAM chips

Lower operating voltage quadratically reduces energy

Low-Voltage Processing Bring Variations

Software Optimization
(e.g., quantization, sparsity)

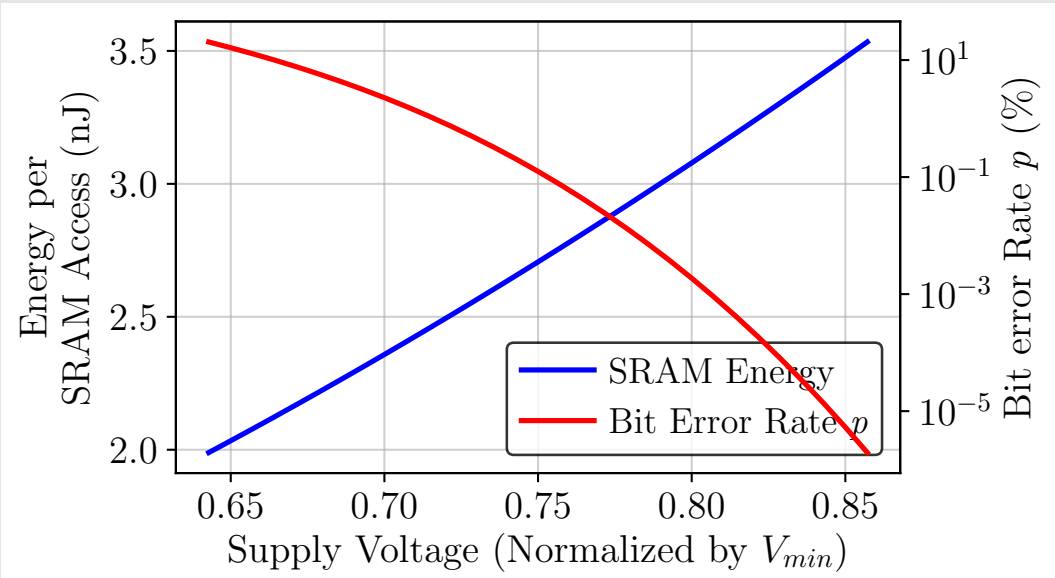


Hardware Optimization
(e.g., optimized dataflow, specialized compute unit)

Lower processor operating voltage

Energy ∝ Voltage ^{2}

SRAM Access Energy / Bit Error Rate vs. Operating Voltage



Data measured from 14nm FinFET SRAM chips

Lower operating voltage bring chip variations/errors

Research Question:

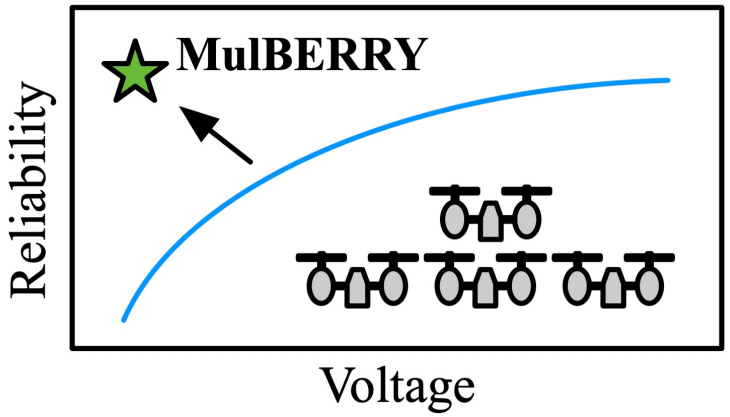
Can we achieve aggressive *energy-savings* under low-voltage operation, *yet remain computationally-resilient for* autonomous drones*?*

MulBERRY System Design Principle

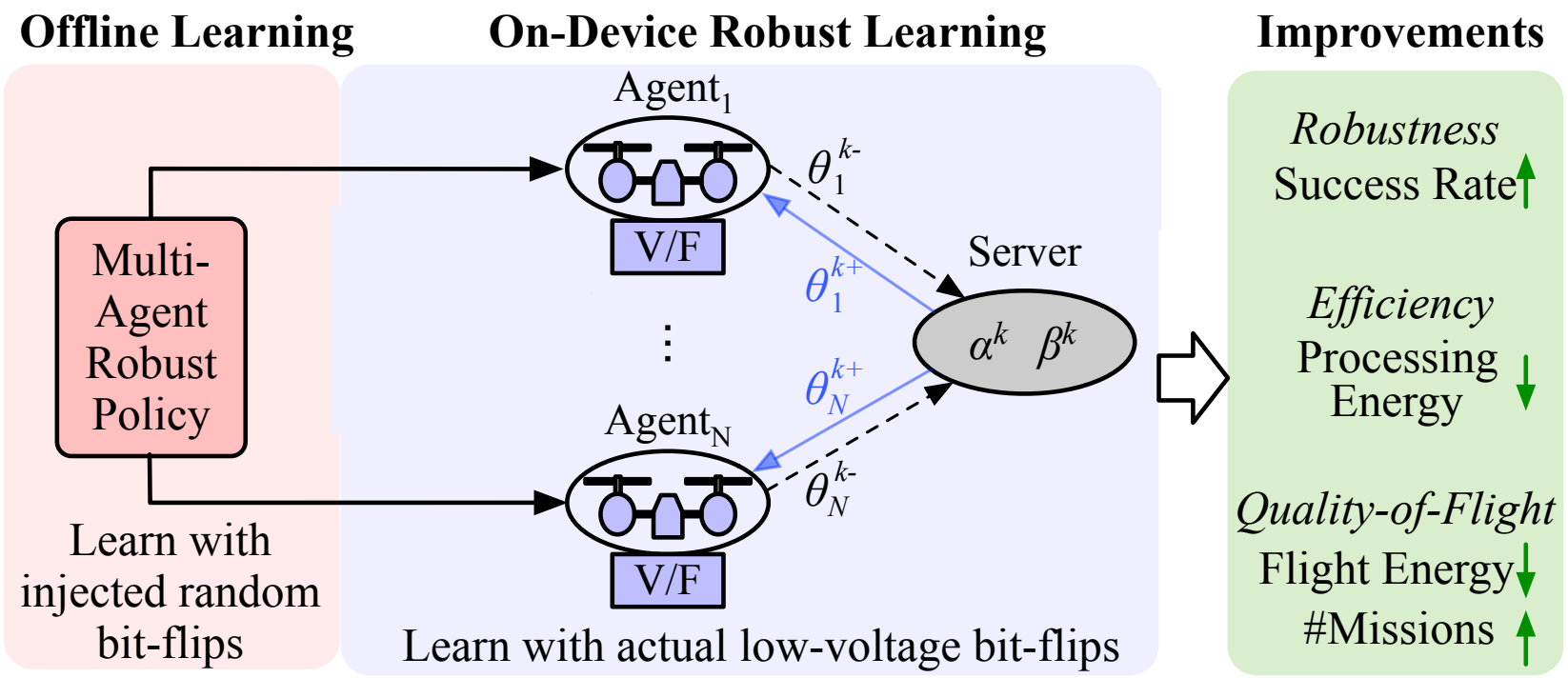
- **Design Principle**: Cross-layer swarm robust learning framework, integrates *algorithm-level* error-aware learning with *system-level* collaborative server-agent optimization and *hardware-level* thermal-voltage adaptive adjustment.

MulBERRY Design Objective

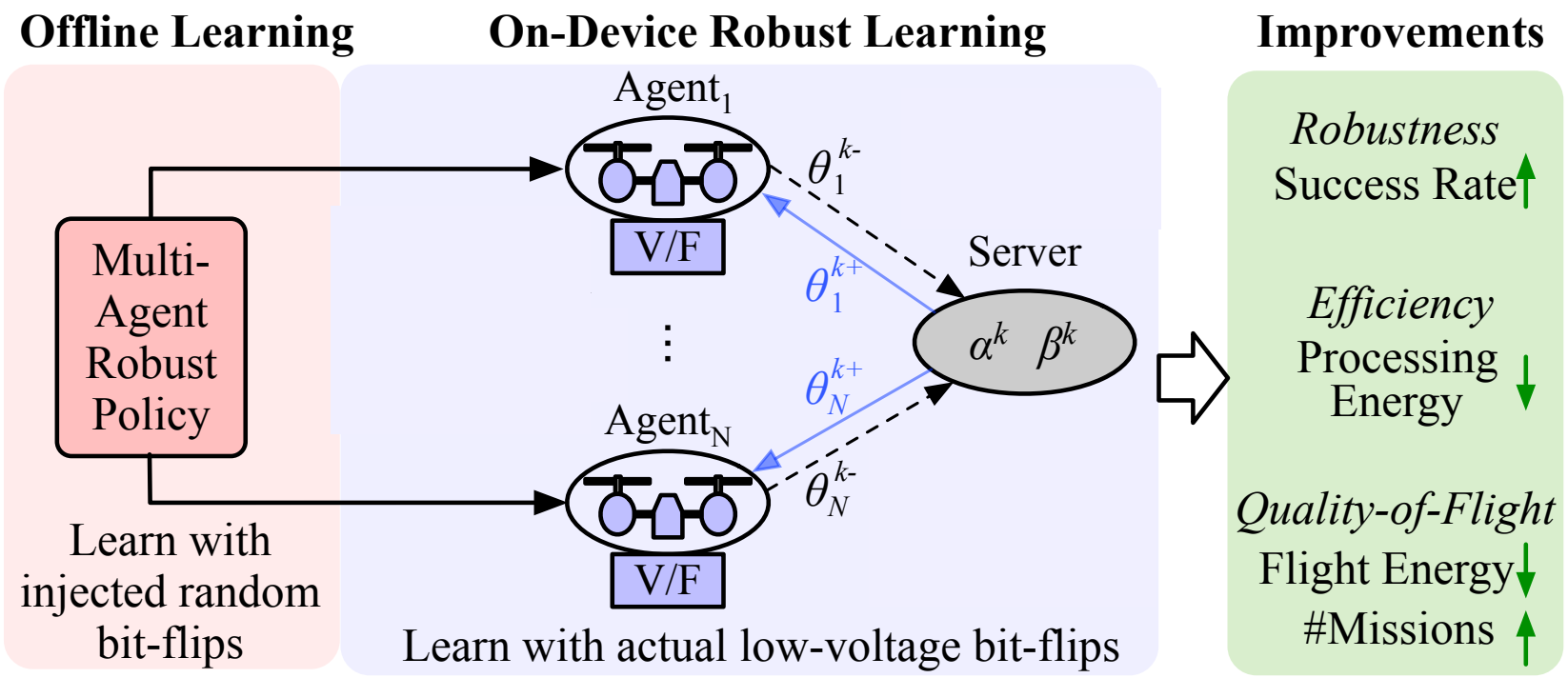
- **Design Principle**: Cross-layer swarm robust learning framework, integrates *algorithm-level* error-aware learning with *system-level* collaborative server-agent optimization and *hardware-level* thermal-voltage adaptive adjustment.
- **Achieve**: Aggressive *energy-savings* under *low-voltage operation*, yet *computationally-resilient* for swarm autonomous drone systems.



MulBERRY Key Techniques

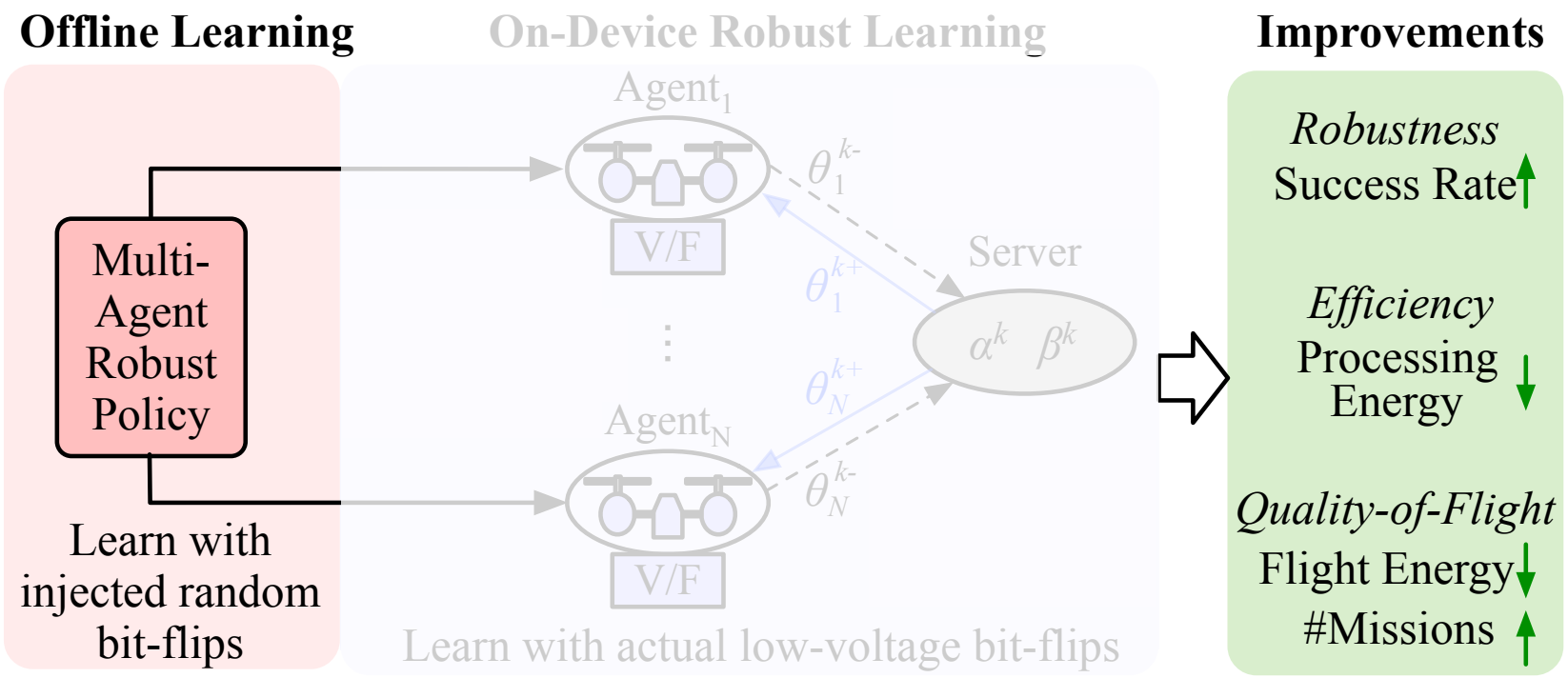


MulBERRY Key Techniques



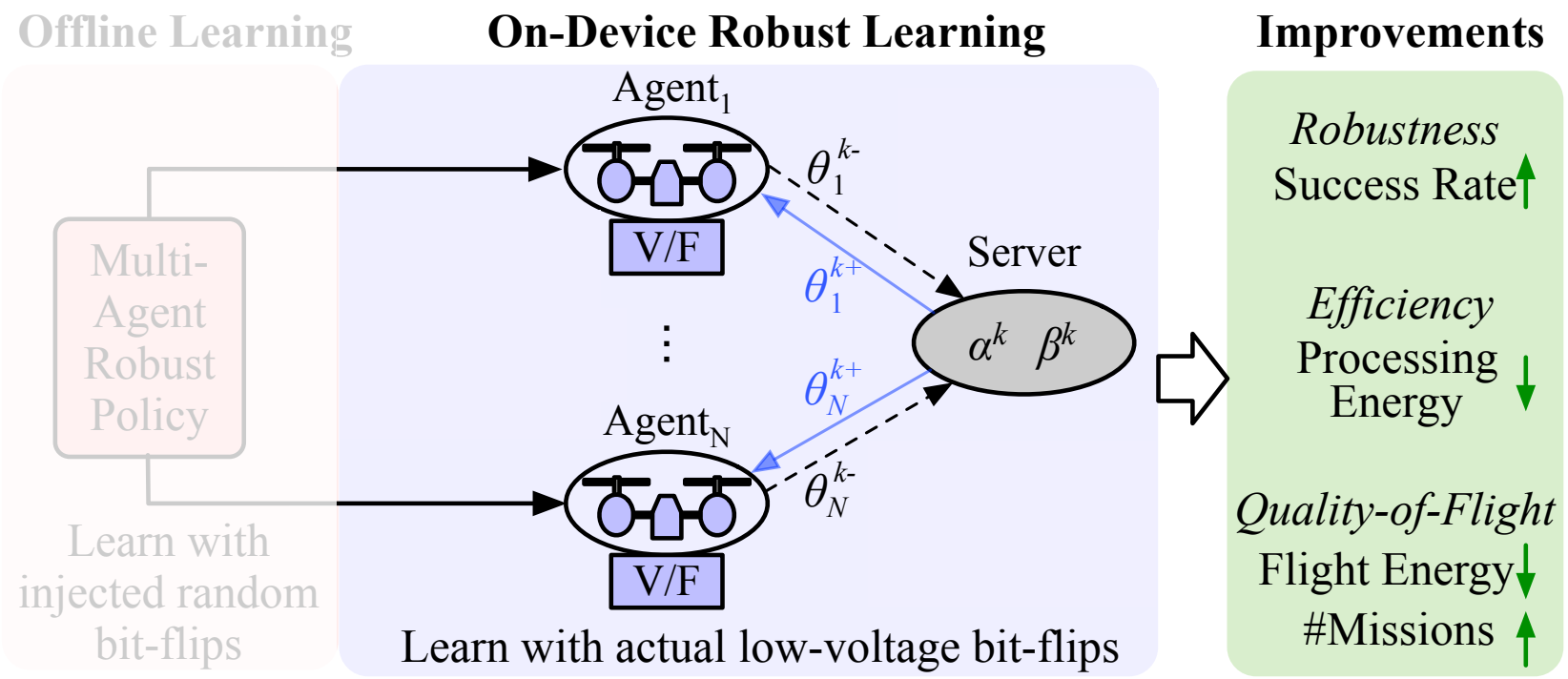
Two-Stage Swarm Robust Learning

MulBERRY Key Techniques



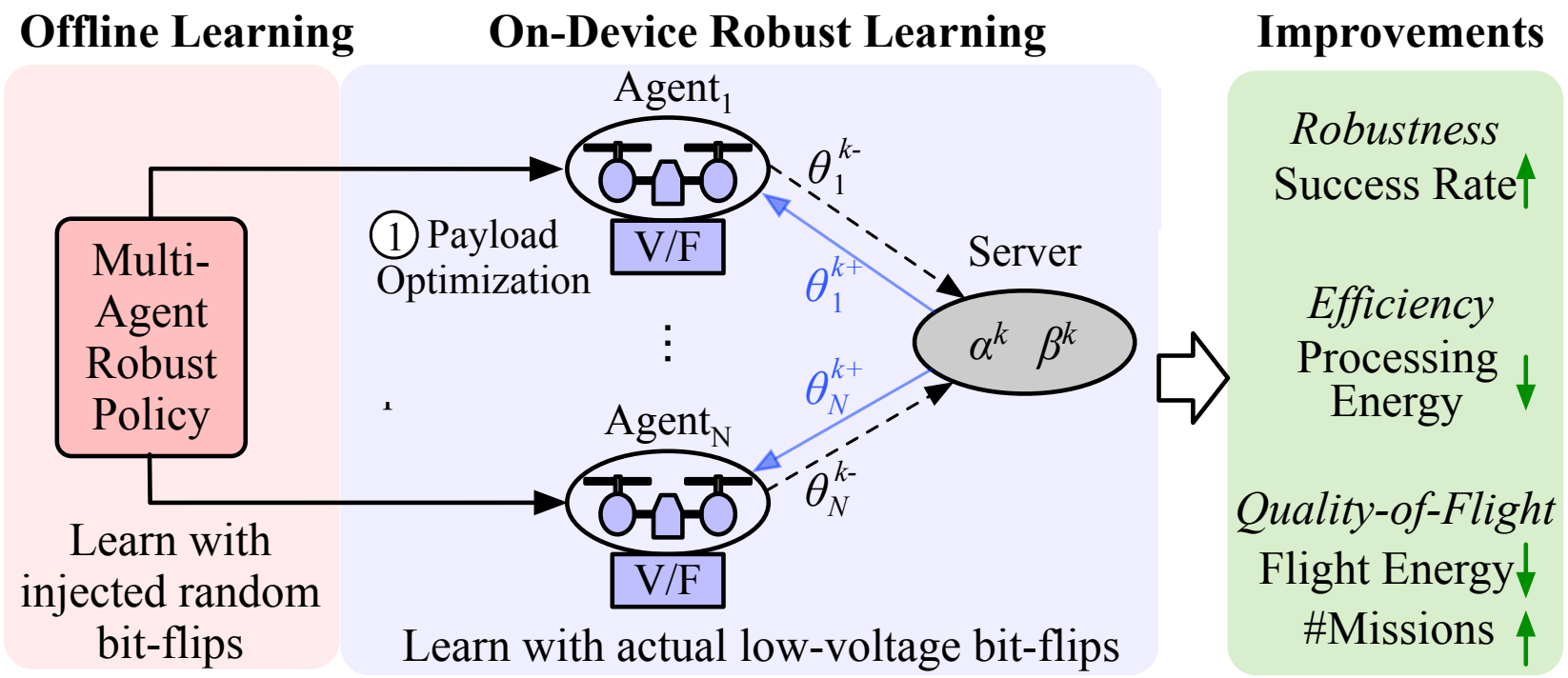
Two-Stage Swarm Robust Learning

MulBERRY Key Techniques



Two-Stage Swarm Robust Learning

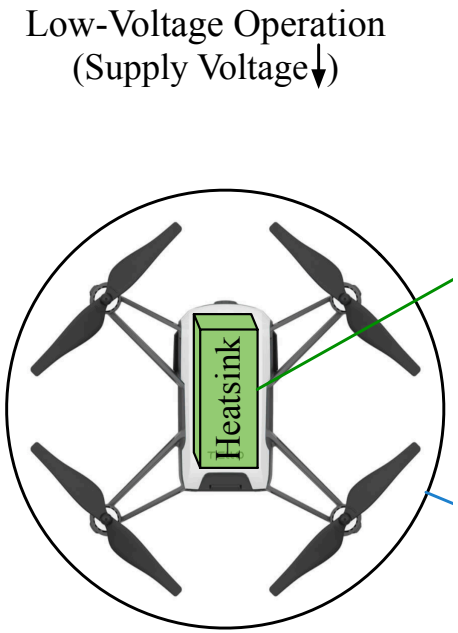
MulBERRY Key Techniques



Two-Stage Swarm Robust Learning

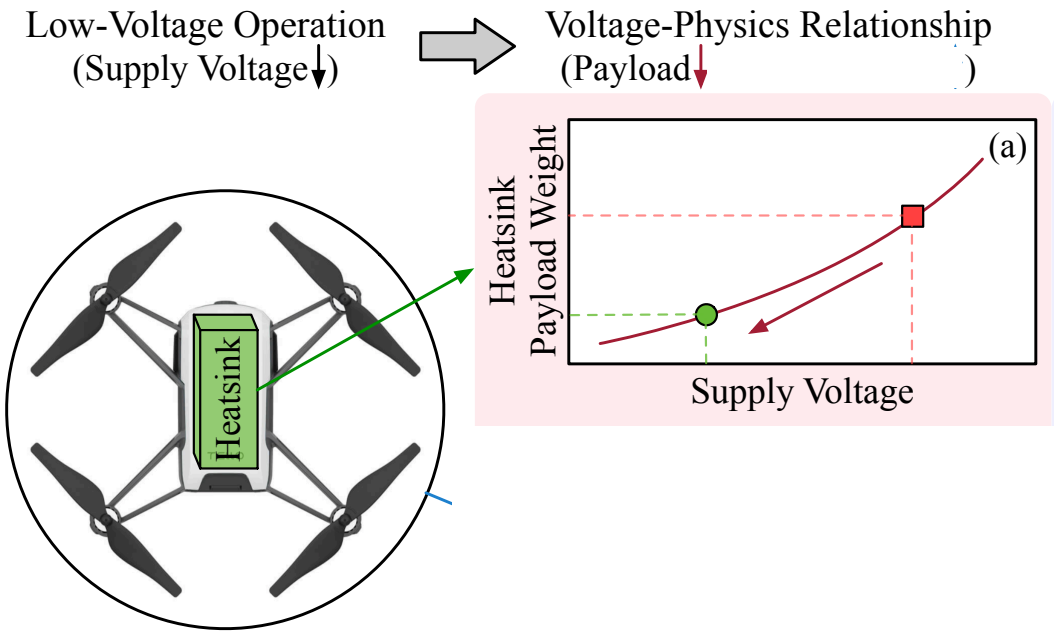
Low-Voltage Payload Optimization

Low-Voltage Thermal-Payload Optimization



Low-voltage operation

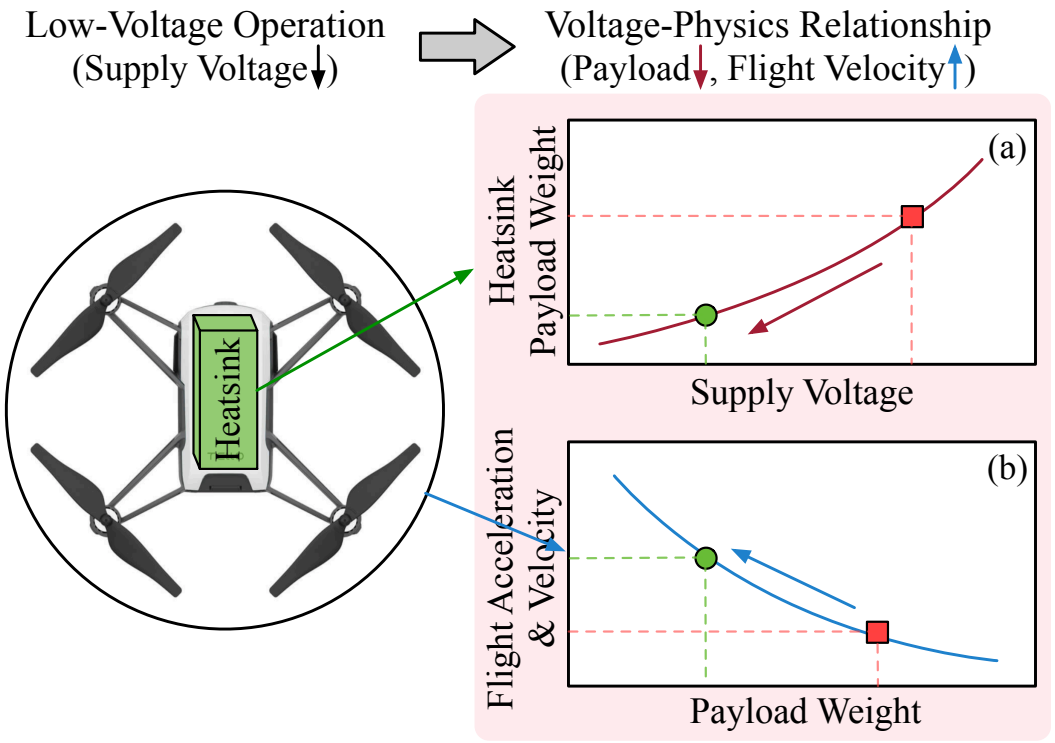
Low-Voltage Thermal-Payload Optimization



HotSpot analysis[1] + heatsink modeling[2]
[1] Hotspot 6.0: Validation, acceleration and extension
[2] Celsia Heatsink Size Simulator

Low-voltage operation ⇒ **Payload weight↓**
(Peak temperature↓, heatsink size and weight↓)

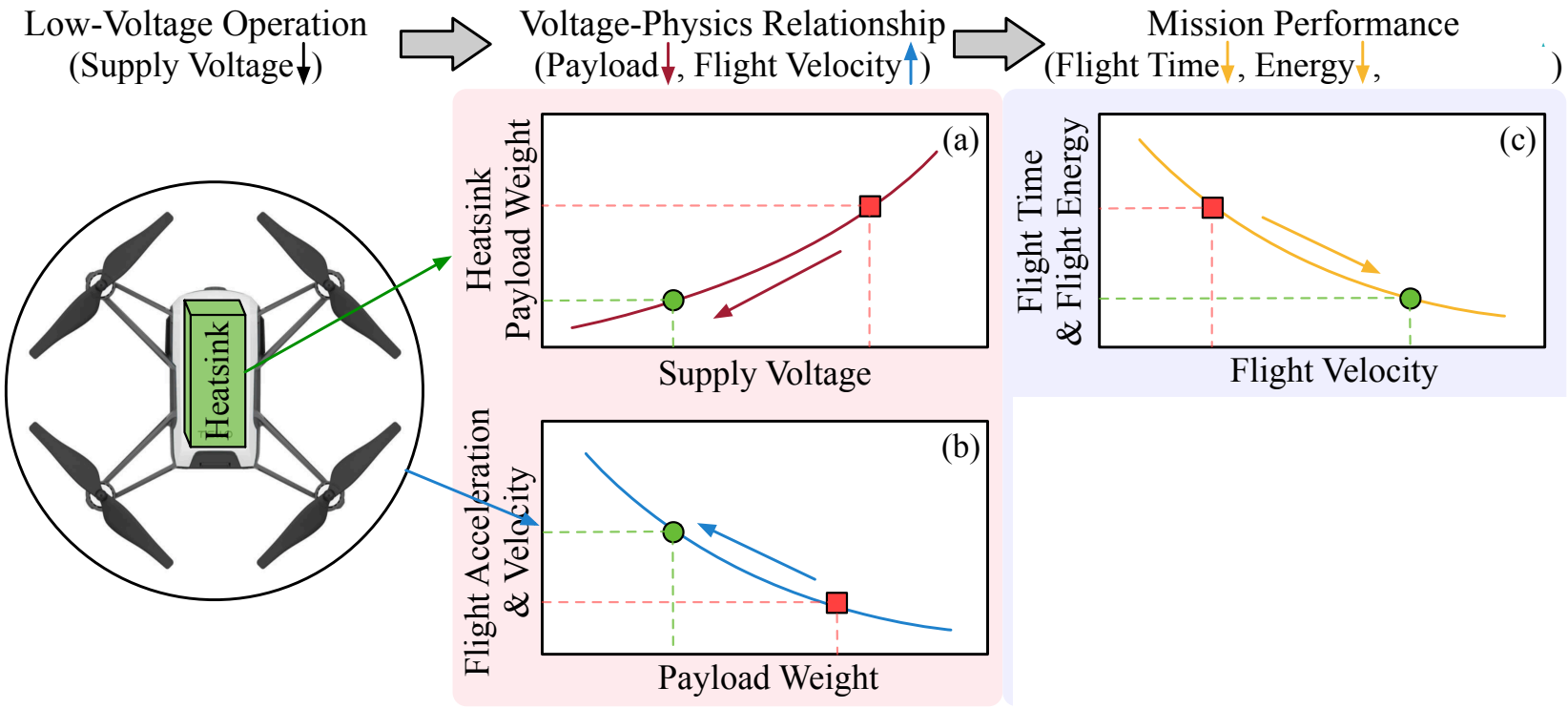
Low-Voltage Thermal-Payload Optimization



Low-voltage operation → Payload weight ↓

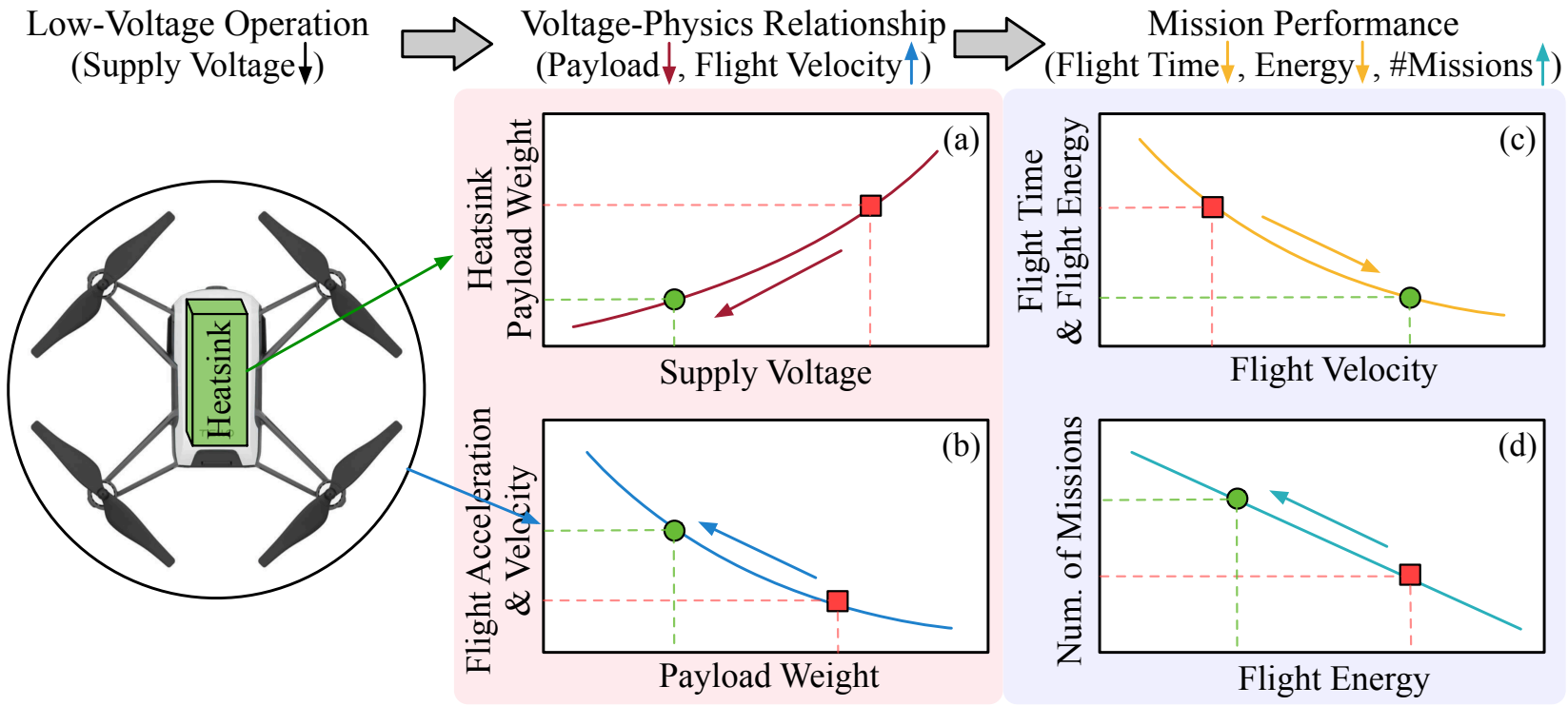
→ **Safe flight velocity↑**

Low-Voltage Thermal-Payload Optimization



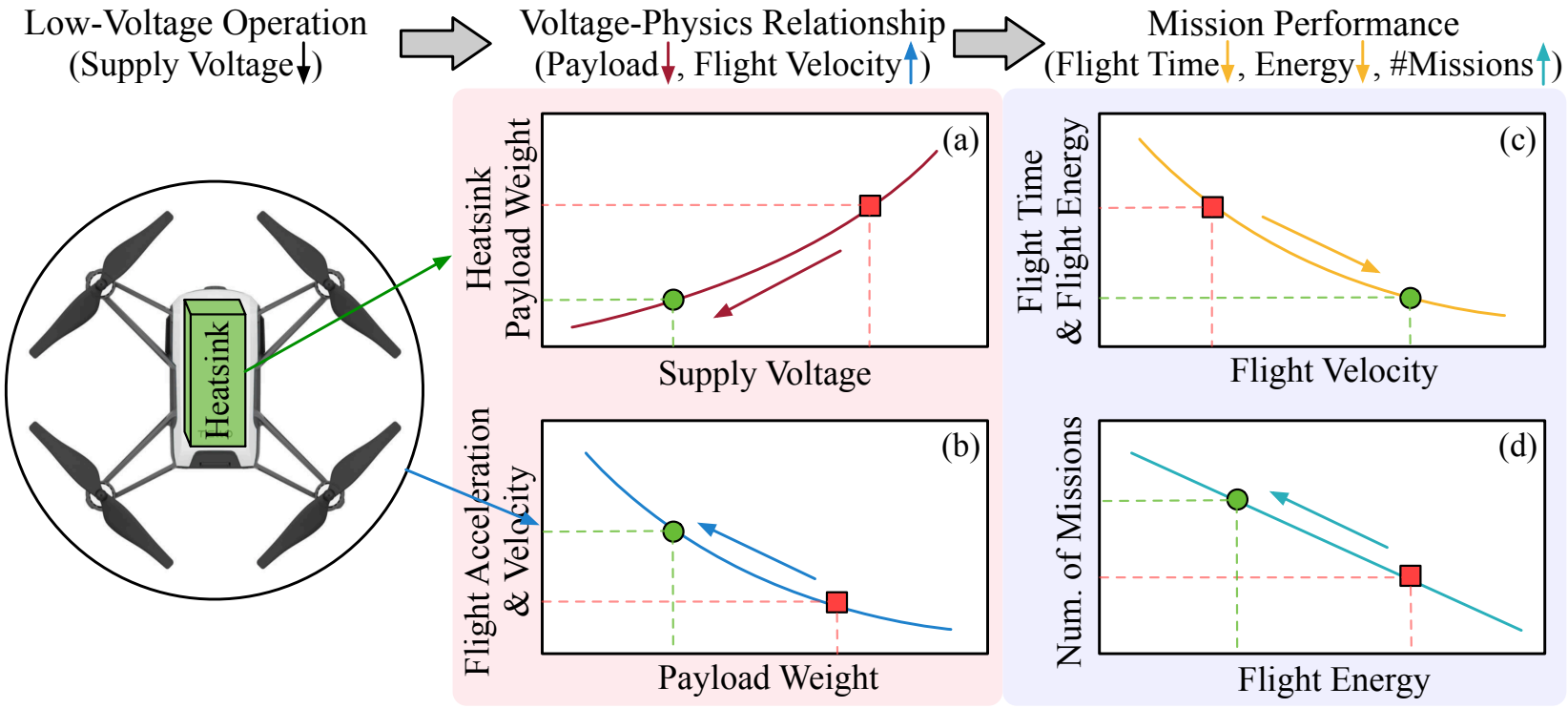
Low-voltage operation → Payload weight ↓ → Safe flight velocity ↑ → **Flight time & energy ↓**

Low-Voltage Thermal-Payload Optimization



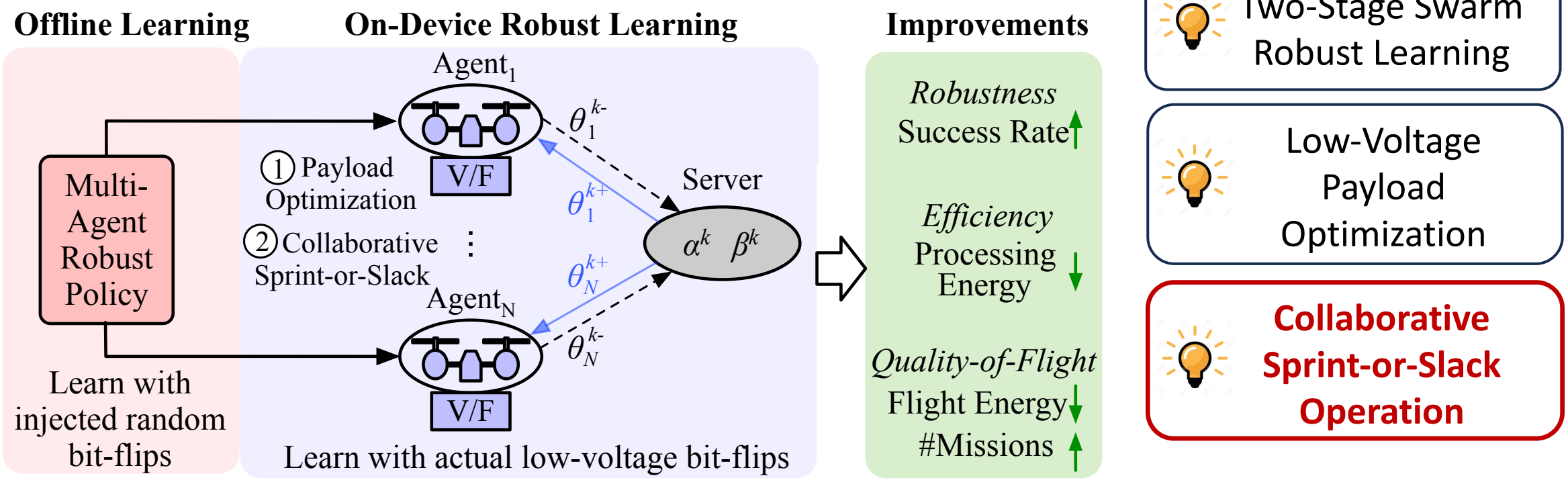
Low-voltage operation → Payload weight ↓ → Safe flight velocity ↑ → Flight time & energy ↓ → **Number of missions ↑**

Low-Voltage Thermal-Payload Optimization

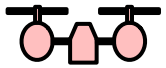


Under low-voltage, MulBERRY reduces drone payload, leading to increased safe flight velocity, thus reducing mission time and energy

MulBERRY Key Techniques



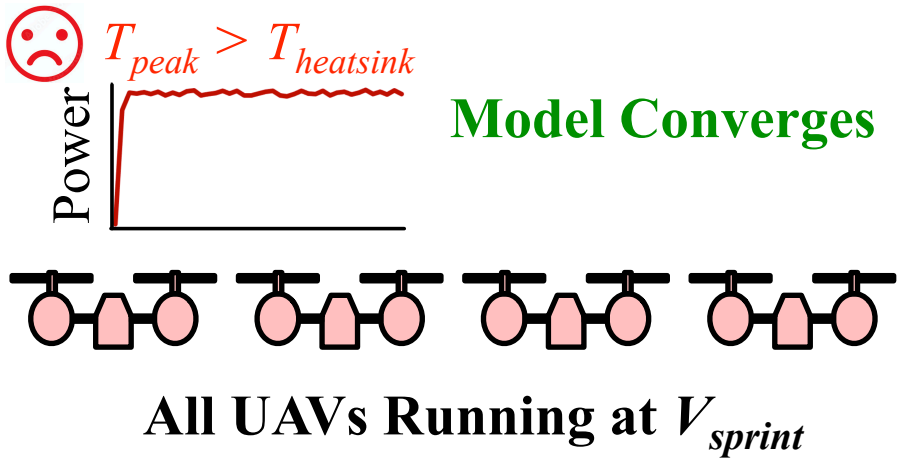
Sprint-or-Slack Operation


UAV Sprint: operate at nominal voltage (no error, high energy)


UAV Slack: operate at low voltage (with error, low energy)

Sprint-or-Slack Operation

- UAV Sprint: operate at nominal voltage (no error, high energy)
- UAV Slack: operate at low voltage (with error, low energy)

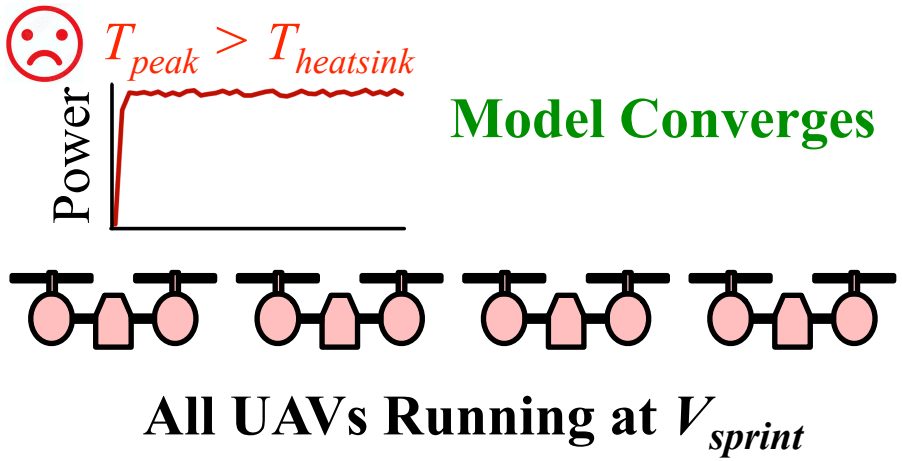


All UAVs are Sprinting

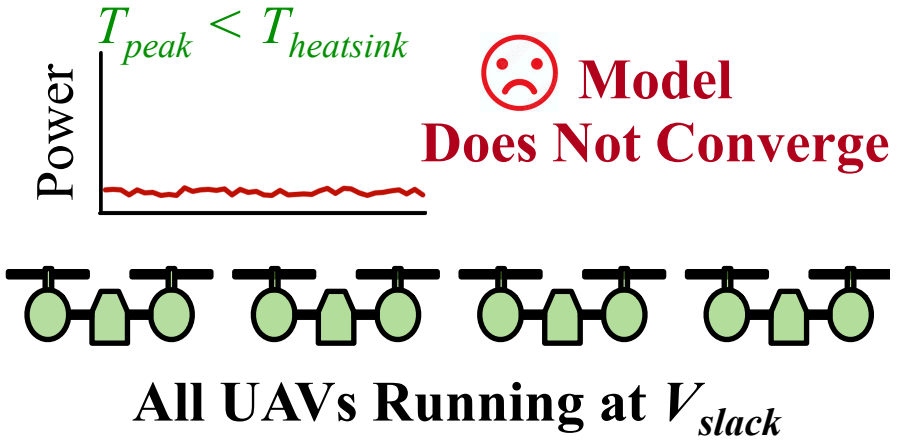
Sprint-or-Slack Operation

UAV Sprint: operate at nominal voltage (no error, high energy)

UAV Slack: operate at low voltage (with error, low energy)

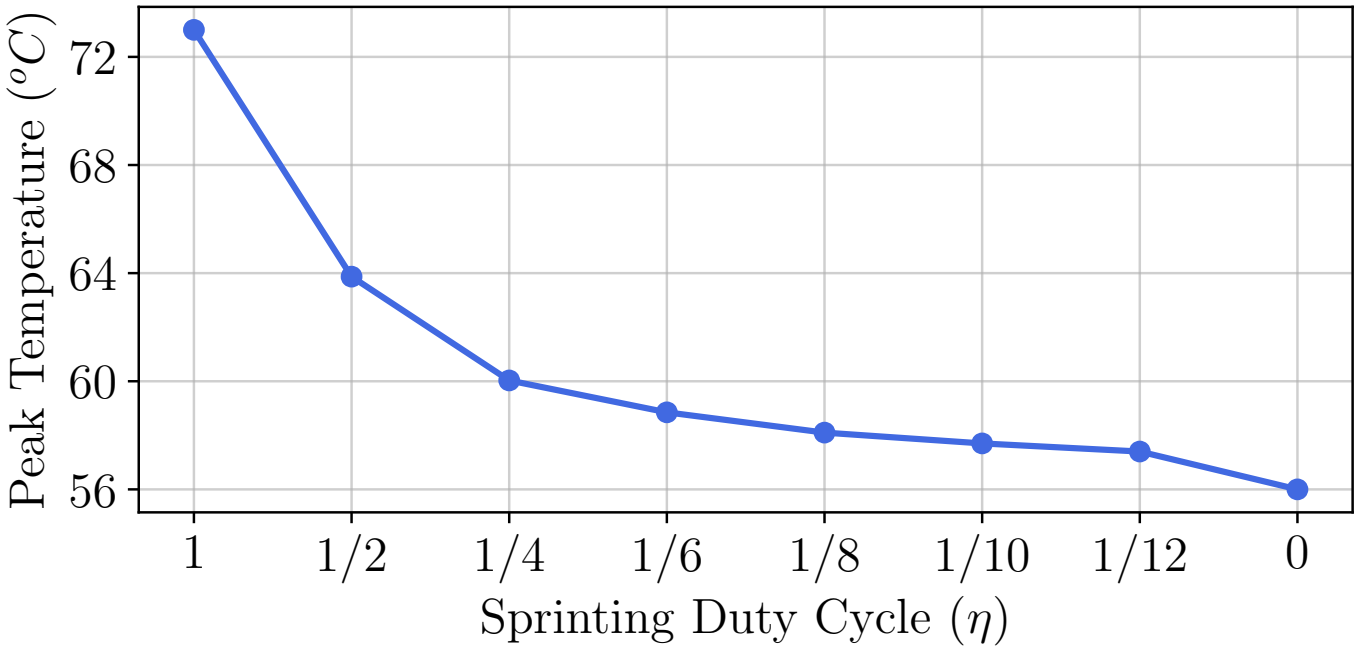


All UAVs are Sprinting

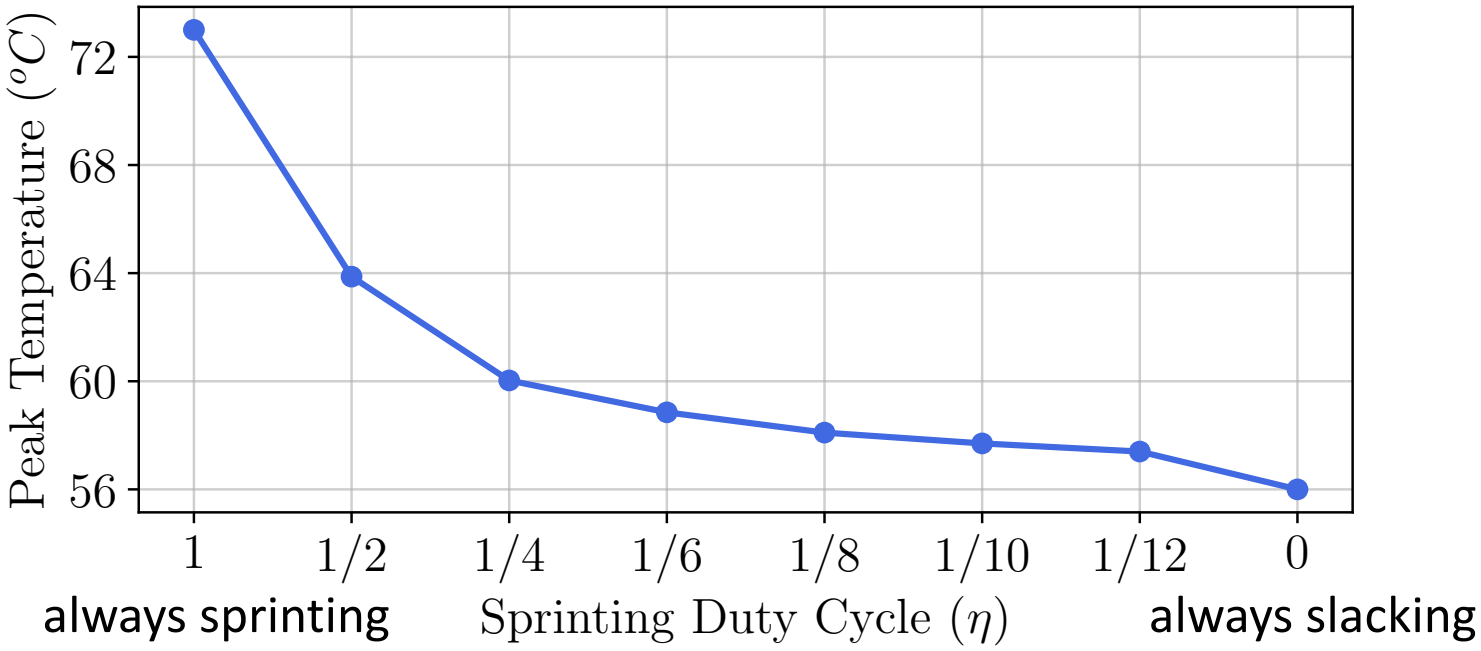


All UAVs are Slacking

Sprint-or-Slack Operation

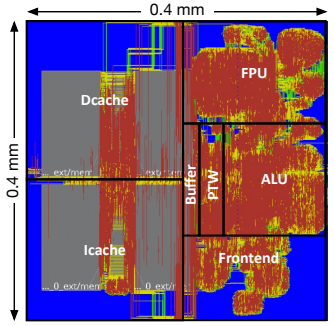


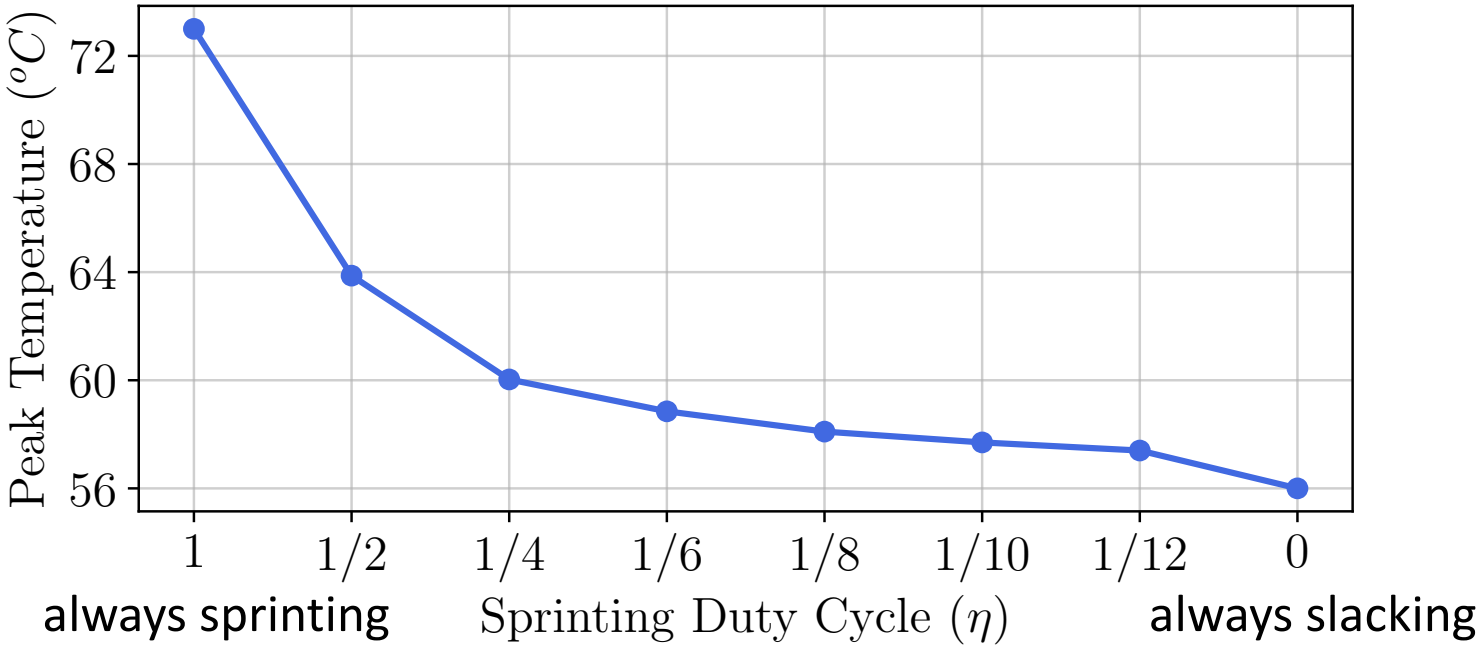
Sprint-or-Slack Operation



Sprinting Duty Cycle (η): fraction of period for which the UAV is sprinting

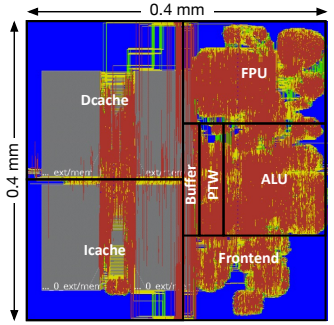
Sprint-or-Slack Operation



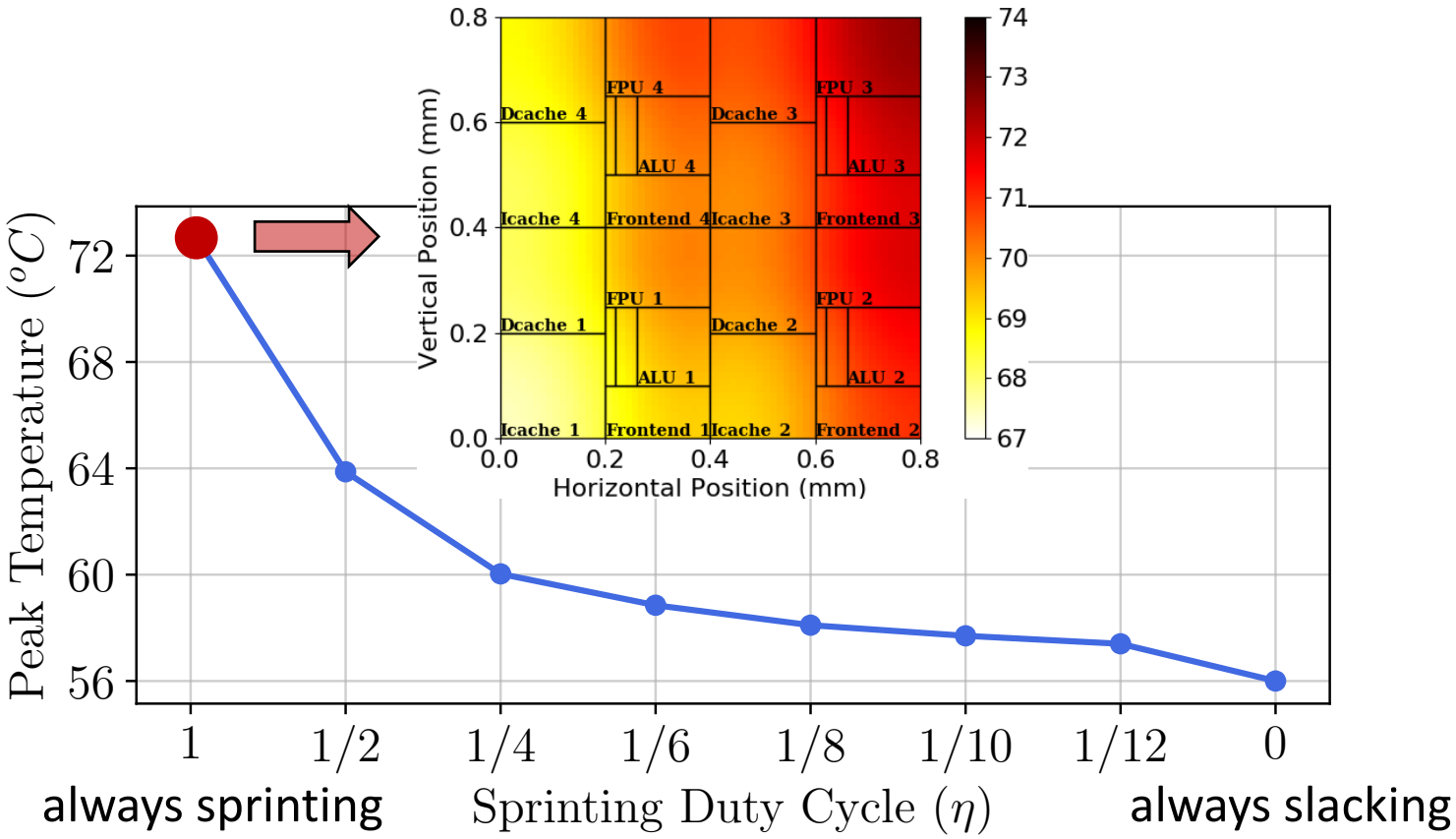


Sprinting Duty Cycle (η): fraction of period for which the UAV is sprinting

Sprint-or-Slack Operation

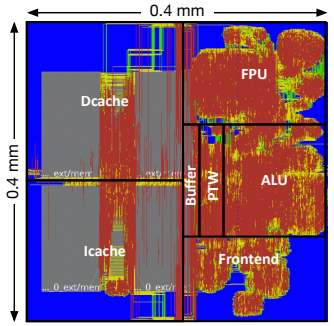


Temperature heatmap η=1

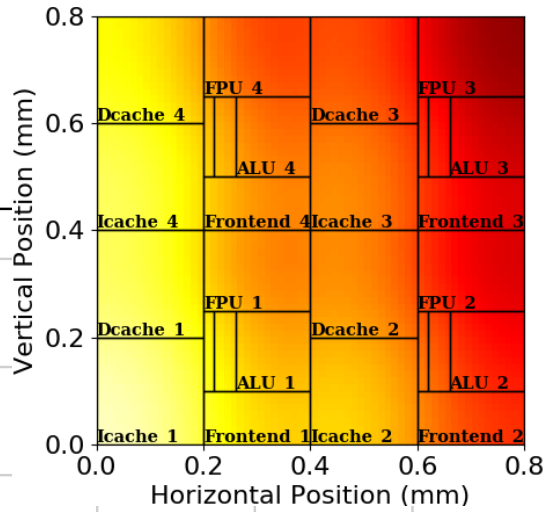


Sprinting Duty Cycle (η): fraction of period for which the UAV is sprinting

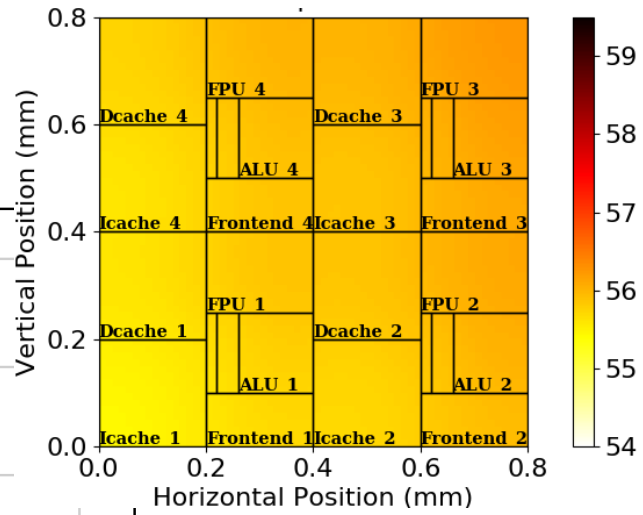
Sprint-or-Slack Operation



Temperature heatmap η=1



Temperature heatmap η=0



Note: for better visualization, color scales are different

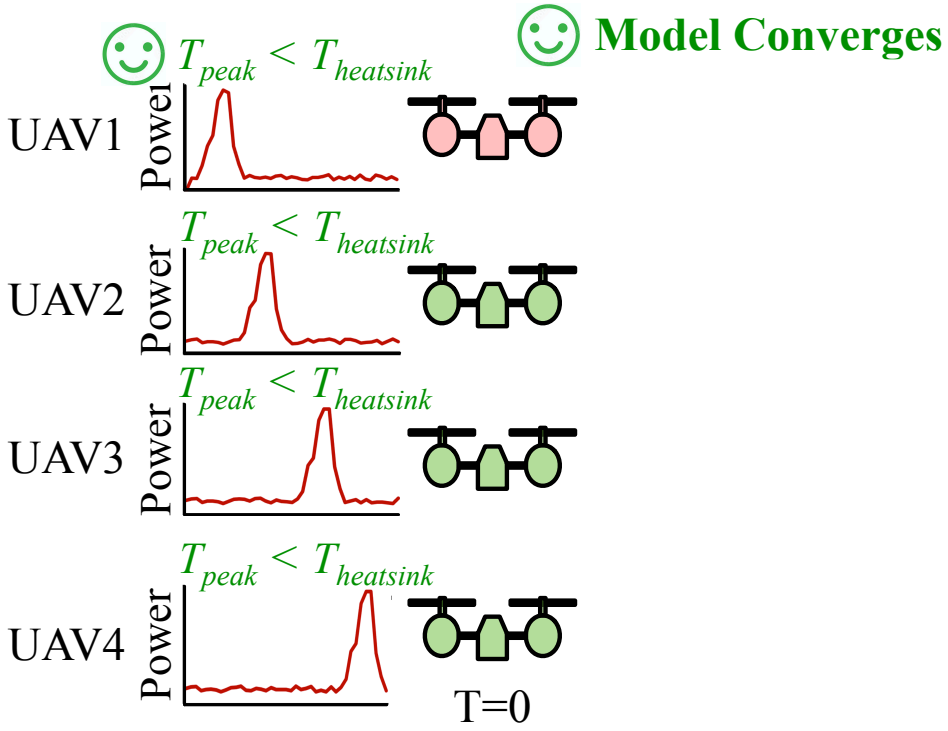
Sprinting Duty Cycle (η): fraction of period for which the UAV is sprinting

Collaborative Sprint-or-Slack Operation

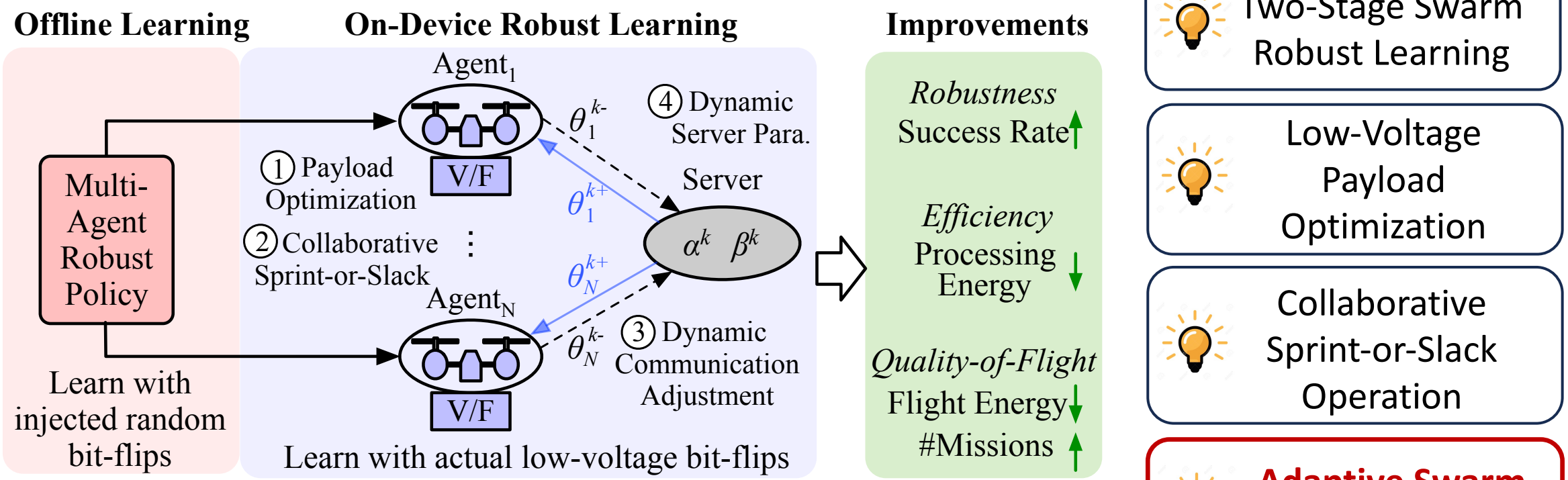
UAV Sprint: operate at nominal voltage (no error, high energy)

UAV Slack: operate at low voltage (with error, low energy)

Collaborative Sprint-or-Slack

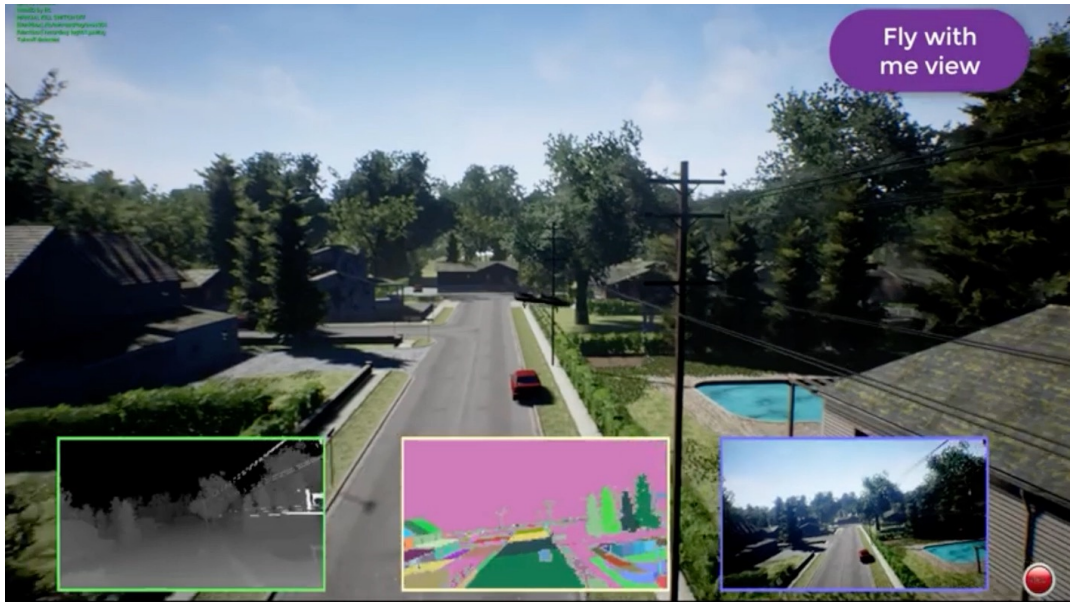


MulBERRY Key Techniques



Evaluation: Swarm UAVs Setup

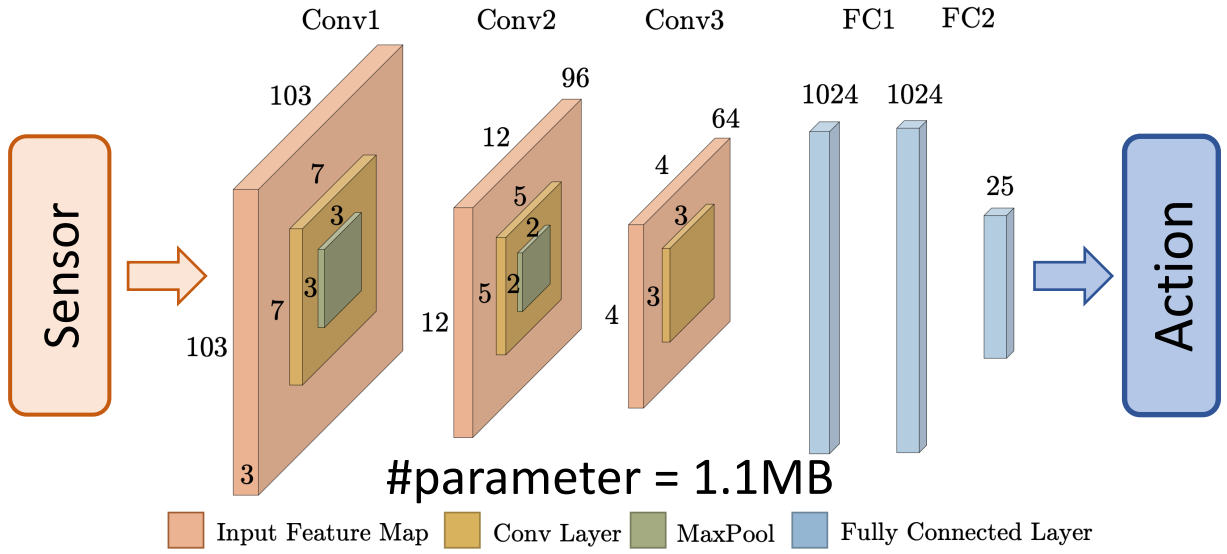
- Simulation Platform:



Unreal Engine + AirSim

(3D realistic environments) (Drone dynamics)

- Task: collaborative package delivery or surveillance
- Policy Architecture of each UAV:



- Swarm size: 4-UAV, 8-UAV, 12-UAV

Swarm UAVs Experimental Setup (UAV Platform)

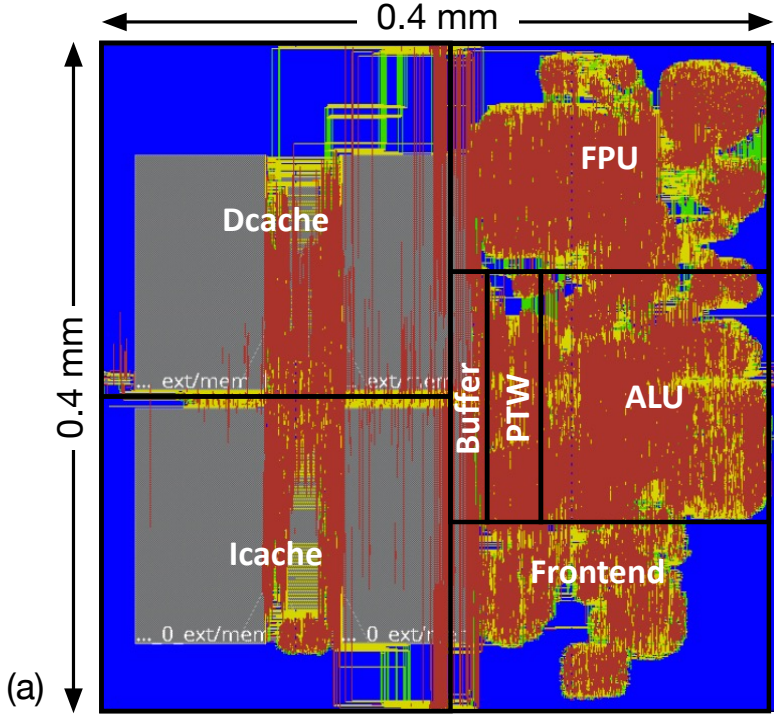
Bitcraze Crazyflie UAV

Nano-Drone
27g takeoff weight
15g max payload
250mAh battery

DJI Tello UAV

Micro-Drone
80g takeoff weight
70g max payload
1100mAh battery

Swarm UAVs Experimental Setup (Hardware)



Layout of one RISC-V Rocket core

(b)

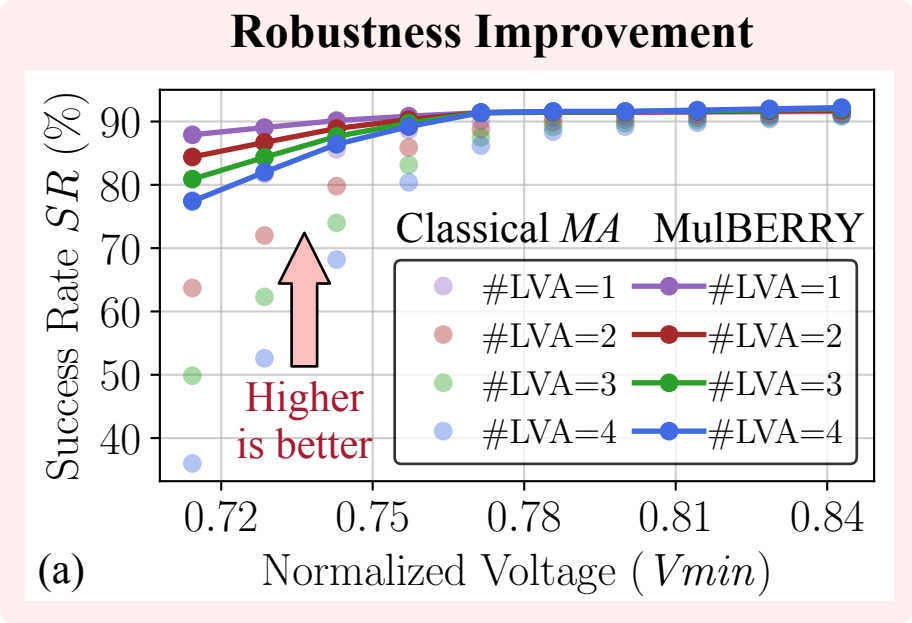
Hardware Configuration Parameters	
Technology	GF 12*nm*
Core Type	4 x RISC-V Rocket Cores
Cache	16KB 4-way I+D Caches
Routed Core Area	0.4*mm* x 0.4*mm*
Voltage Range	0.54*V* to 1*V*
Power	117*mW* to 399*mW*

Evaluation Metrics

- Compute-level:
  - Processing Energy
- System-level:
  - Avg. flight success rate
  - Avg. flight time
  - Avg. flight energy
  - Avg. #missions

All reported results are averaged from 500 runs

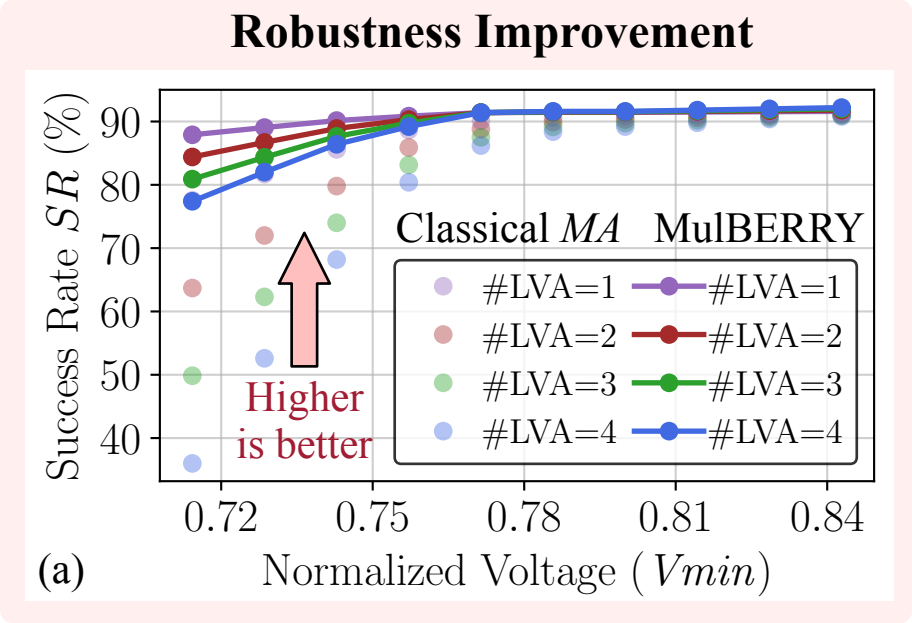
Robustness & Mission Efficiency Improvement

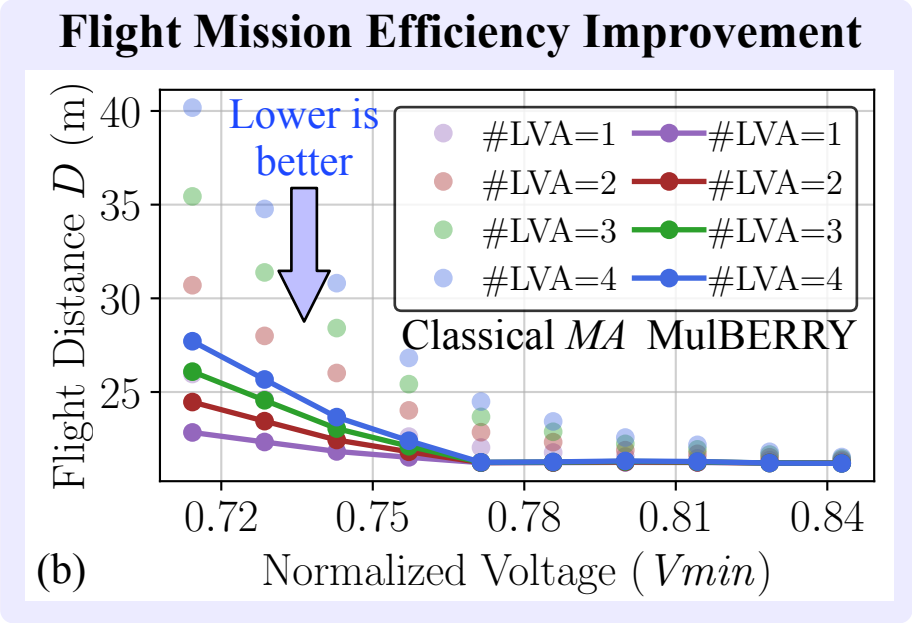


(#LVA: number of low-voltage UAVs)

MulBERRY improves mission robustness under low-voltage operation

Robustness & Mission Efficiency Improvement

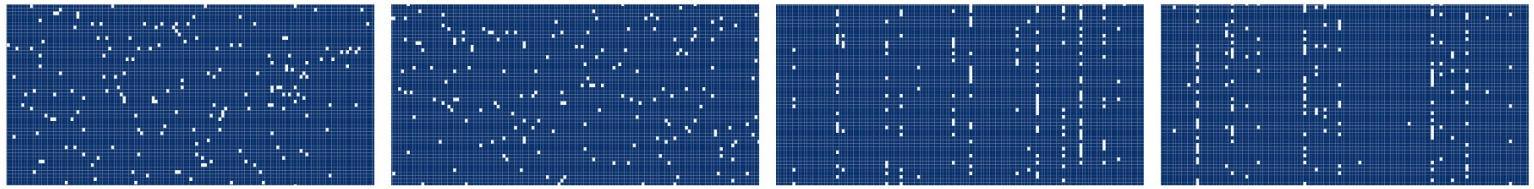




(#LVA: number of low-voltage UAVs)

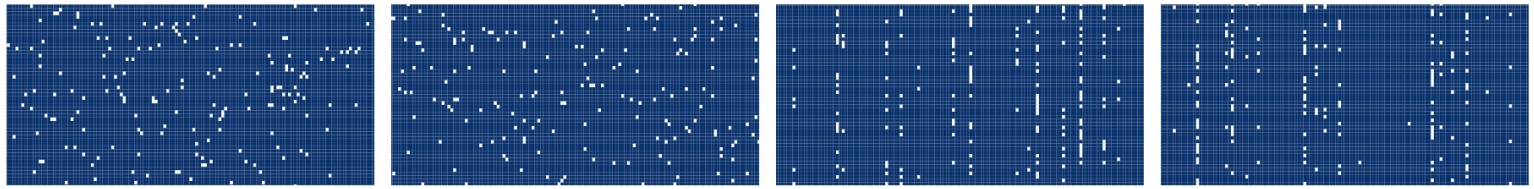
MulBERRY improves mission robustness under low-voltage operation

Effectiveness Across Voltages and Chips



UAV 1 (Chip 1) UAV 2 (Chip 2) UAV 3 (Chip 3) UAV 4 (Chip 4)

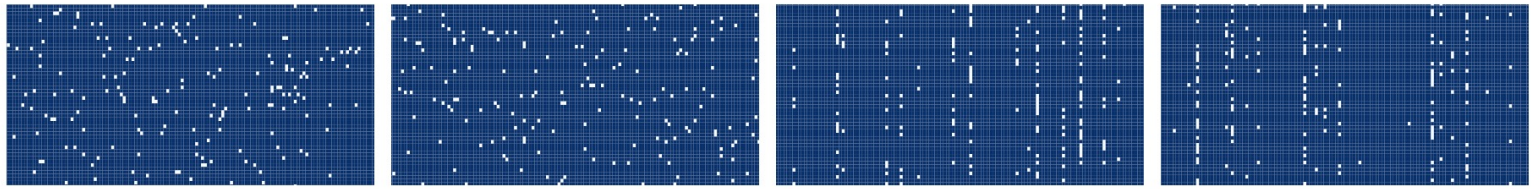
Effectiveness Across Voltages and Chips



UAV 1 (Chip 1) UAV 2 (Chip 2) UAV 3 (Chip 3) UAV 4 (Chip 4)

Voltage / BER (p)	**Metric**	**UAV 1**	**UAV 2**	**UAV 3**	**UAV 4**
Baseline $1V$ (p=0)					
$0.77V_{min}$ / (p=0.025%)	Success Rate (%)				
	Flight Energy (J)				
$0.74V_{min}$ / (p=0.203%)	Success Rate (%)				
	Flight Energy (J)				

Effectiveness Across Voltages and Chips



UAV 1 (Chip 1) UAV 2 (Chip 2) UAV 3 (Chip 3) UAV 4 (Chip 4)

Voltage / BER (p)	Metric	UAV 1	UAV 2	UAV 3	UAV 4
Baseline $1V$ (p=0)	Success Rate = 91.4%, Flight Energy = 75.80J				
$0.77V_{min}$ / (p=0.025%)	Success Rate (%)	91.6	91.4	90.2	90.6
	Flight Energy (J)	63.90	64.06	66.16	65.47
$0.74V_{min}$ / (p=0.203%)	Success Rate (%)	91.4	91.6	90.4	90.2
	Flight Energy (J)	63.15	62.95	64.37	64.78

MulBERRY is scalable across voltages and chips, and consistently improves efficiency and robustness

Effectiveness Across Environments

Sparse Obstacle

Medium Obstacle

Dense Obstacle

Effectiveness Across Environments


Sparse Obstacle


Medium Obstacle


Dense Obstacle

Environment	**Sparse**		**Medium**		**Dense**	
	Flight Energy (J)	Num. of Missions	Flight Energy (J)	Num. of Missions	Flight Energy (J)	Num. of Missions
Baseline @1V						
MulBERRY (optimal)						

Effectiveness Across Environments


Sparse Obstacle


Medium Obstacle


Dense Obstacle

Environment	Sparse		Medium		Dense	
	Flight Energy (J)	Num. of Missions	Flight Energy (J)	Num. of Missions	Flight Energy (J)	Num. of Missions
Baseline @1V	52.41	58.56	75.80	40.15	102.4	28.04
MulBERRY (optimal)	42.02 @0.69V_{min}	71.63	61.42 @0.70V_{min}	49.01	85.77 @0.73V_{min}	33.79

MulBERRY is adaptive across environments, and consistently improves efficiency;
Sparse obstacle environments enable lower operating voltage

Key Takeaways:

Low-voltage operation leads to **energy savings** in both **compute** and **end-to-end mission energy**

Optimizing autonomous system **cyber** components (compute) impacts its **physical** performance

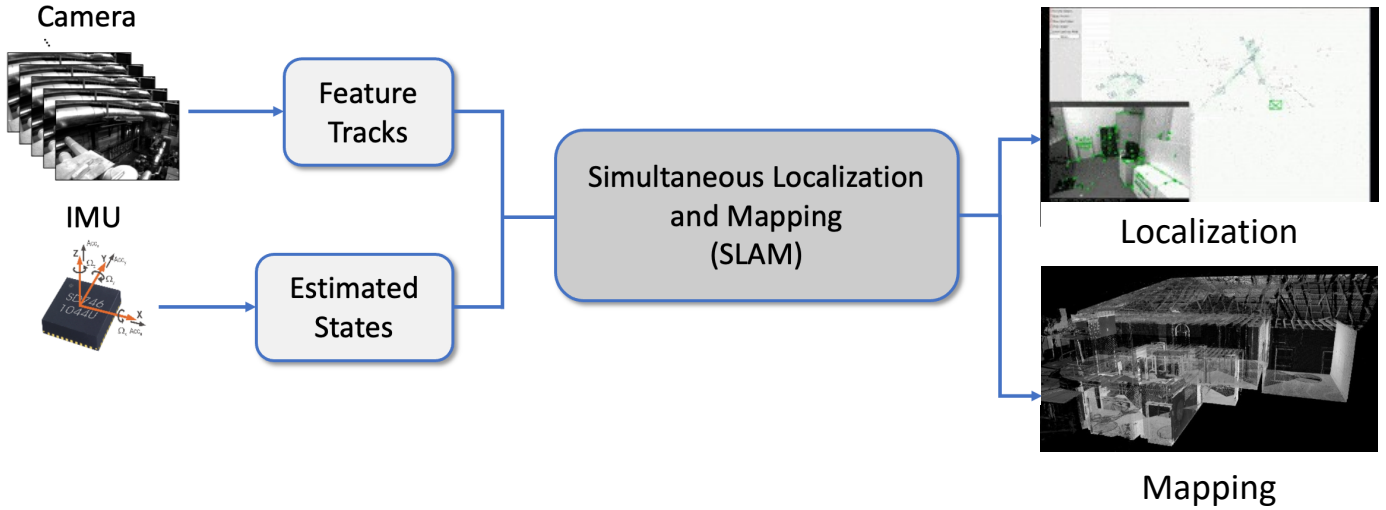
[@CICC'22]

Domain-Specific Architecture (DSA)

Software Optimization
(e.g., quantization, sparsity)

Hardware Optimization
(e.g., optimized dataflow, compute unit)

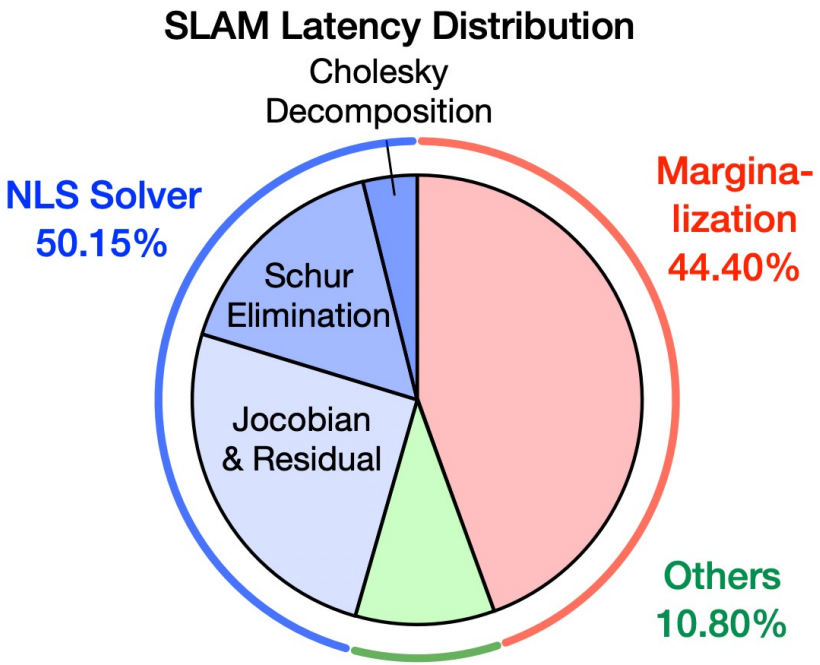
Lower processor operating voltage



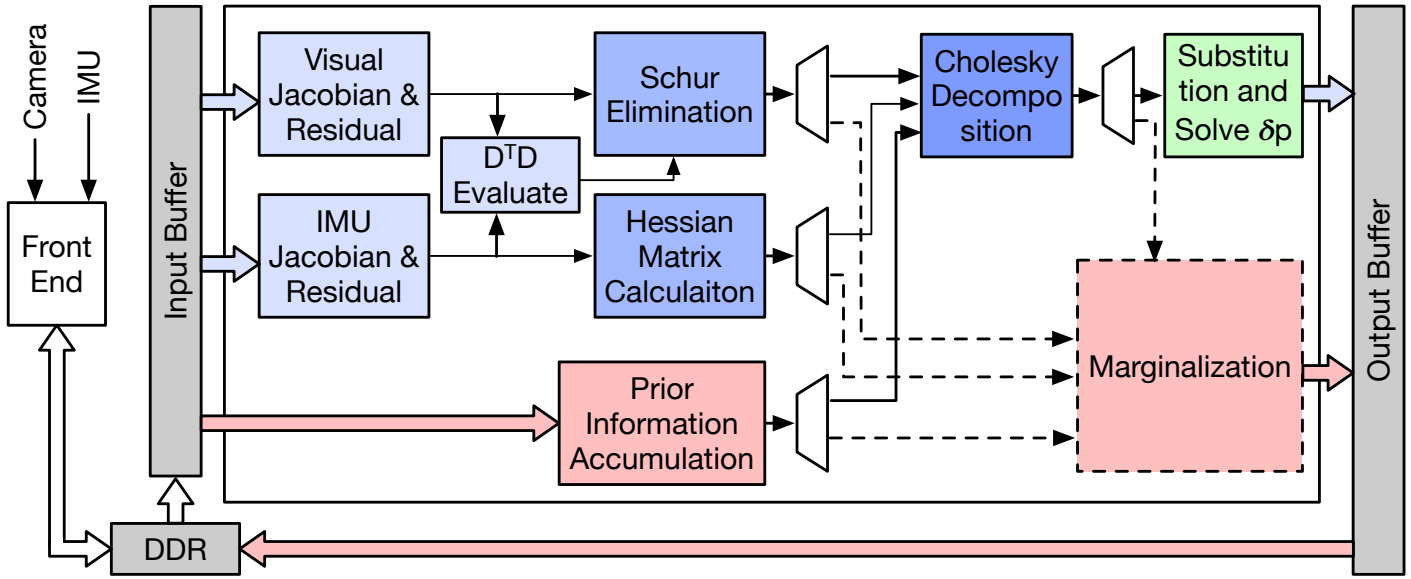
Example: Simultaneous Localization and Mapping (SLAM)

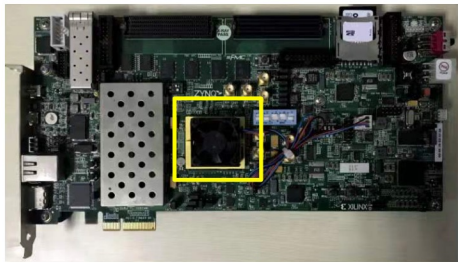
System Profiling & Hardware Architecture

- SLAM System Profiling



- Hardware Architecture Overview





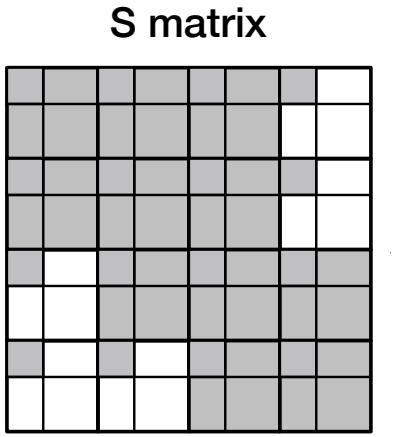
FPGA Zynq-7000 SoC ZC706 with XC7Z045 FFG900-2

Key Features:

- **Energy-efficient** localization & mapping
- **Dynamic reconfiguration** at runtime
- **Real-time** performance of 61 fps at 3.45W (56mJ/frame)

Key Design Technique 1

- Memory Optimization (Data Layout & Symmetry & Sparsity)

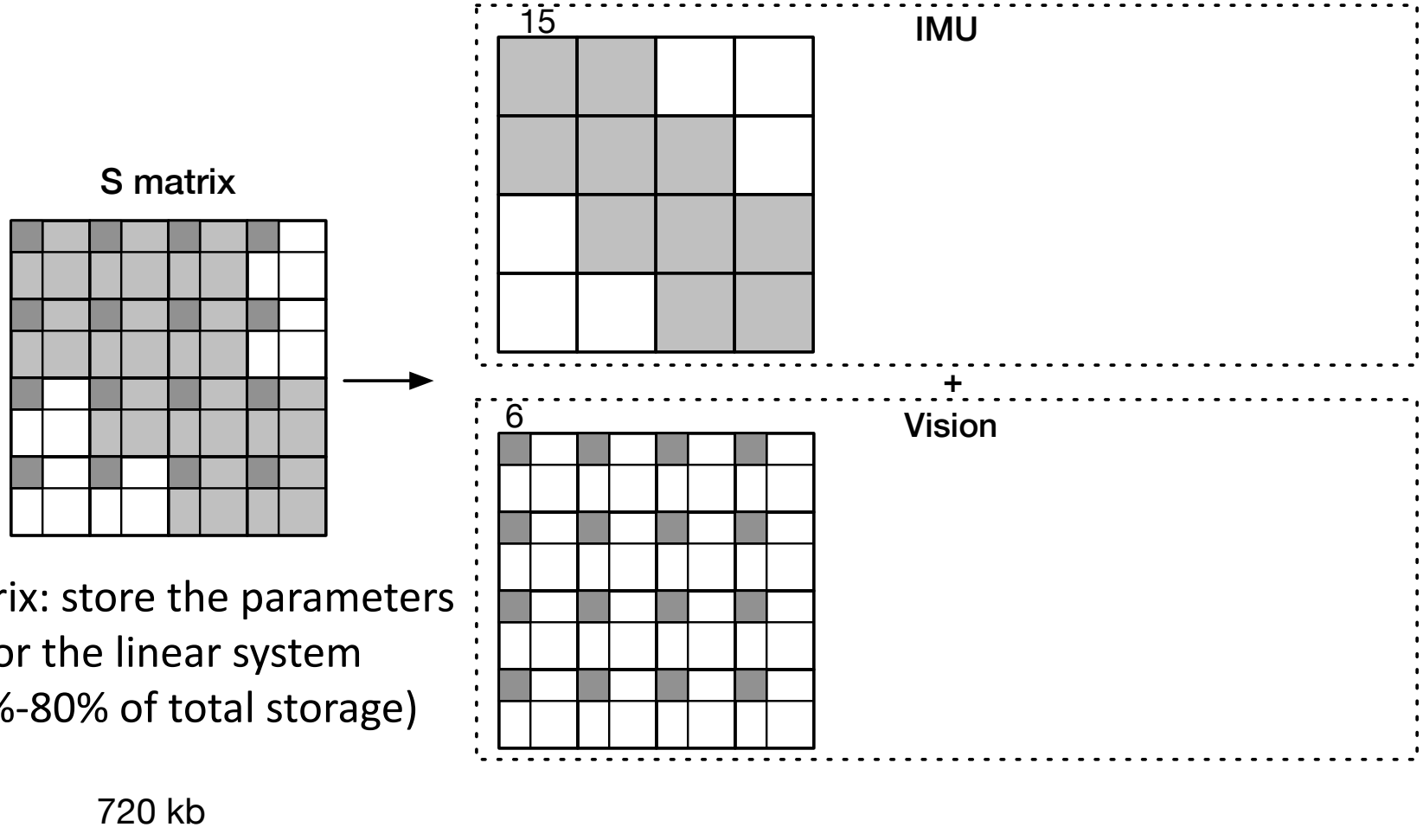


S matrix: store the parameters for the linear system (40%-80% of total storage)

720 kb

Key Design Technique 1

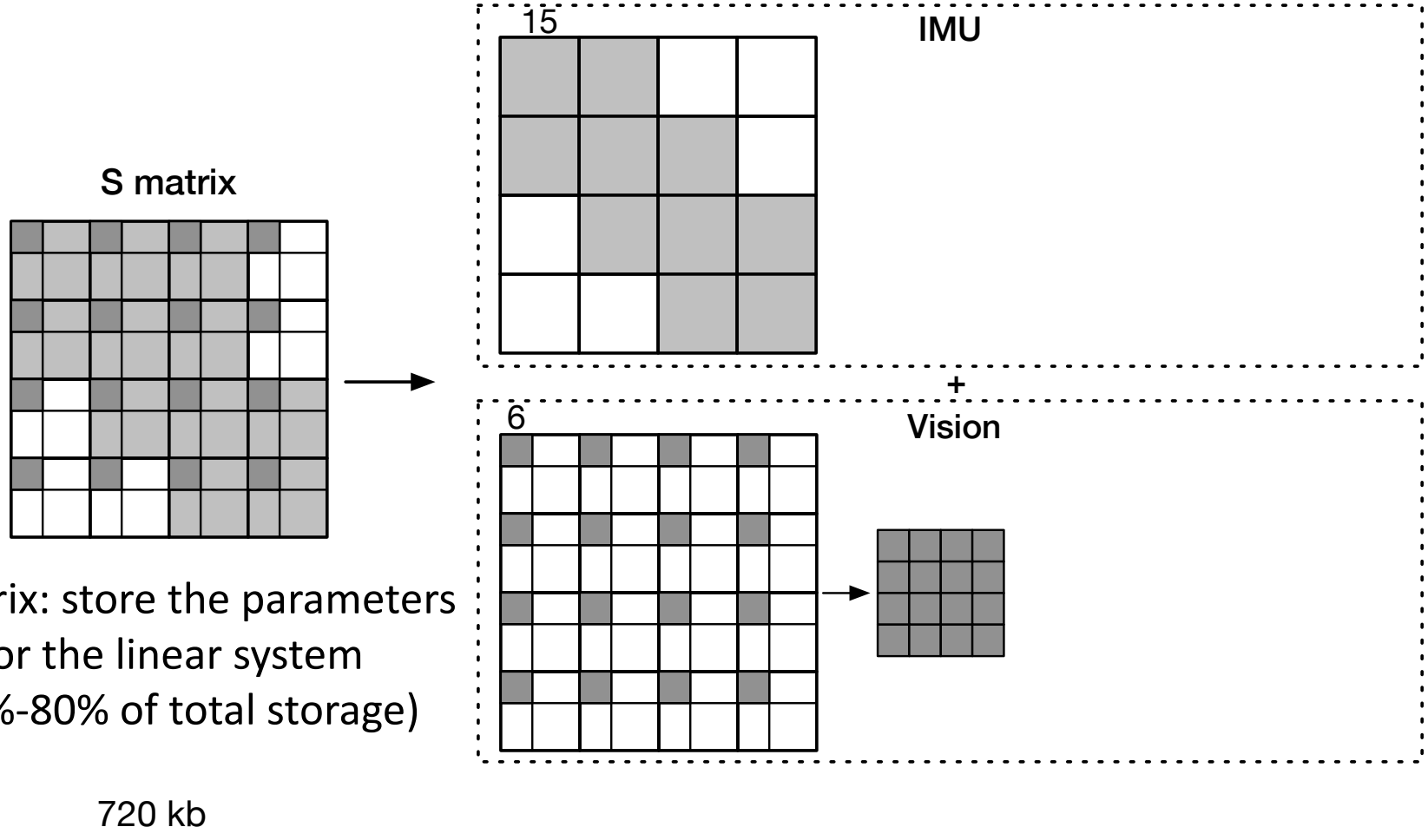
- Memory Optimization (Data Layout & Symmetry & Sparsity)



S matrix: store the parameters for the linear system (40%-80% of total storage)

Key Design Technique 1

- Memory Optimization (Data Layout & Symmetry & Sparsity)

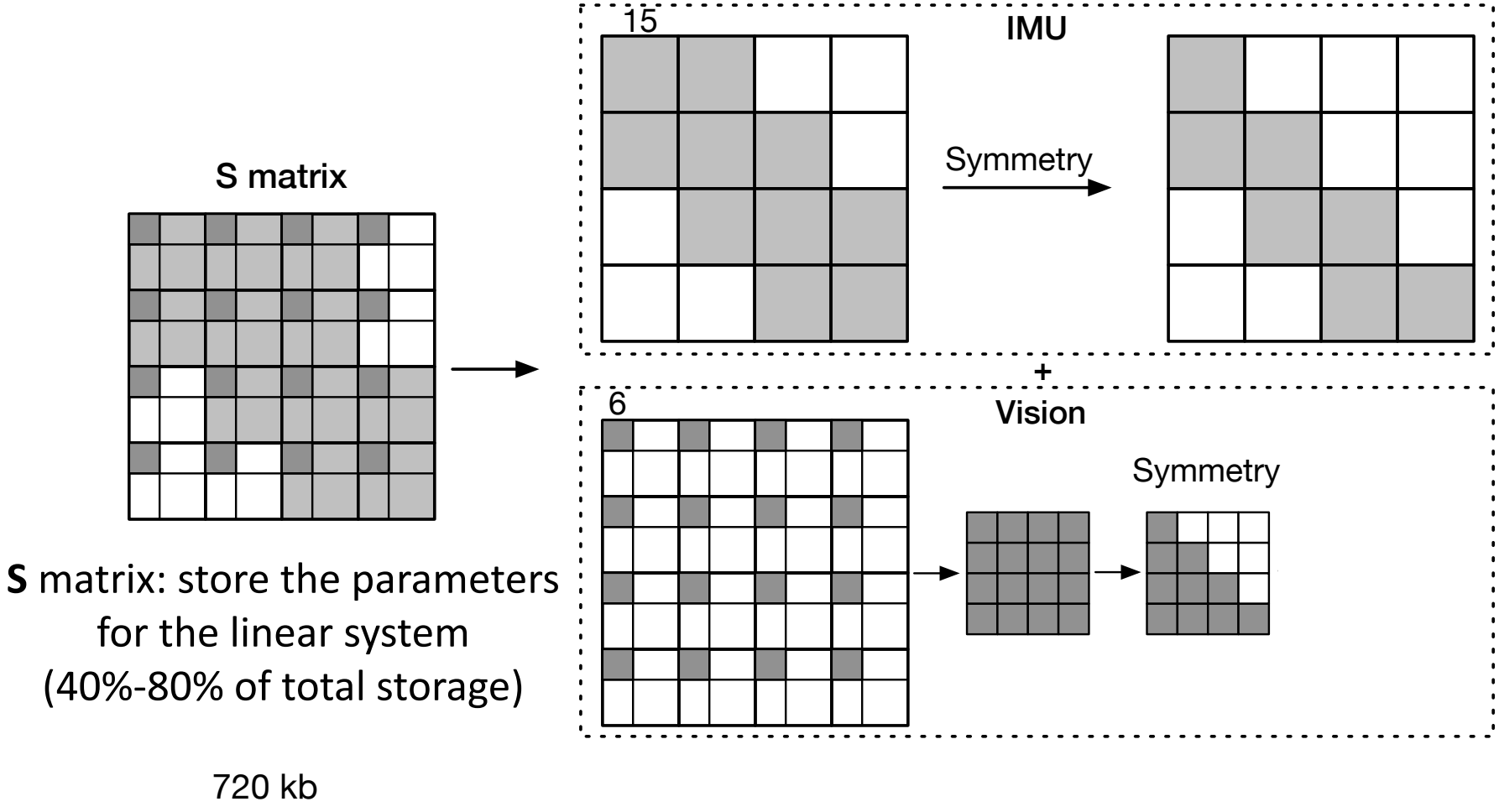


S matrix: store the parameters for the linear system (40%-80% of total storage)

720 kb

Key Design Technique 1

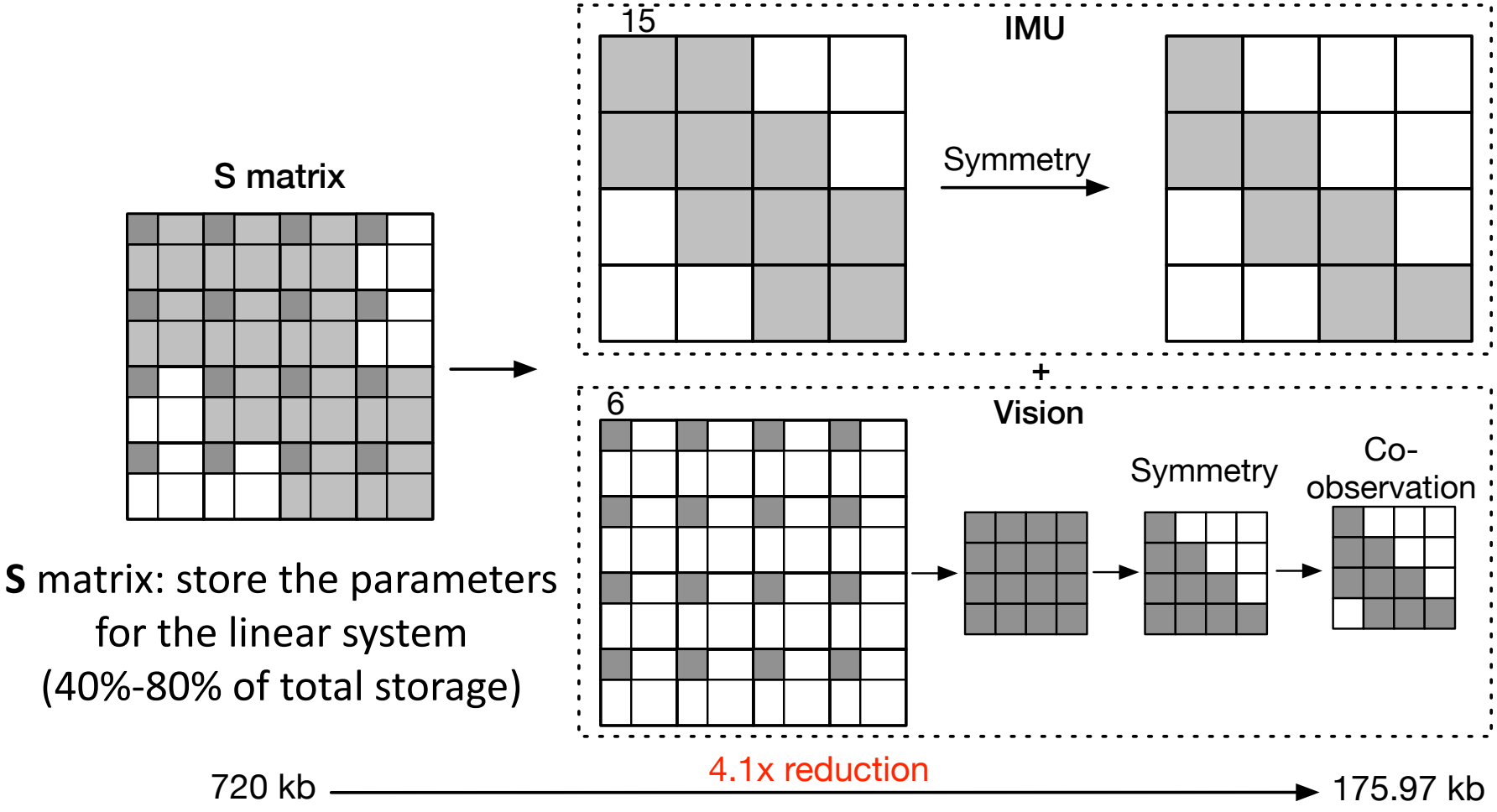
- Memory Optimization (Data Layout & Symmetry & Sparsity)



S matrix: store the parameters for the linear system (40%-80% of total storage)

Key Design Technique 1

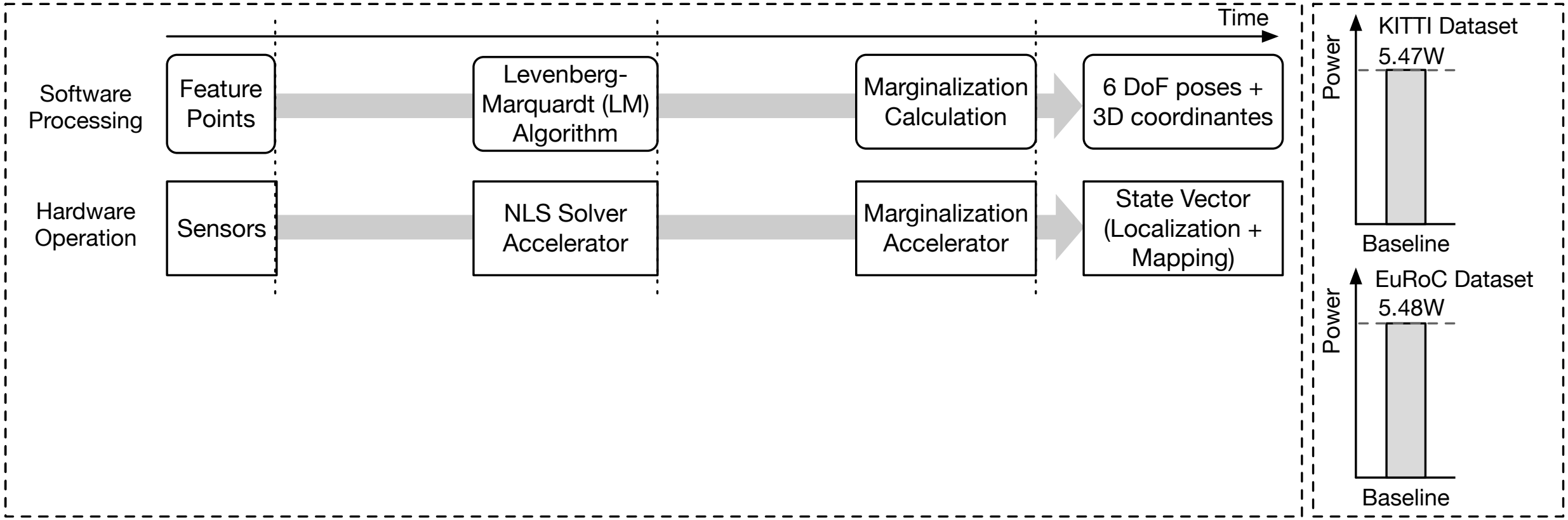
- Memory Optimization (Data Layout & Symmetry & Sparsity)



S matrix: store the parameters for the linear system (40%-80% of total storage)

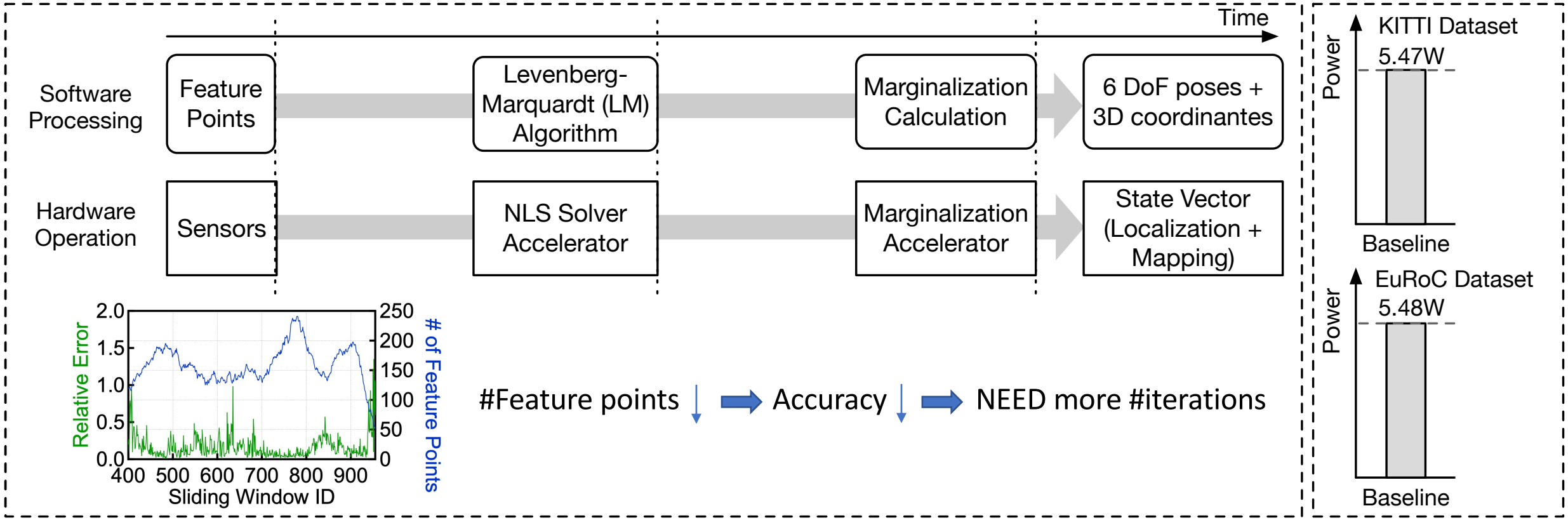
Key Design Technique 2

- Runtime Reconfiguration & Clock Gating



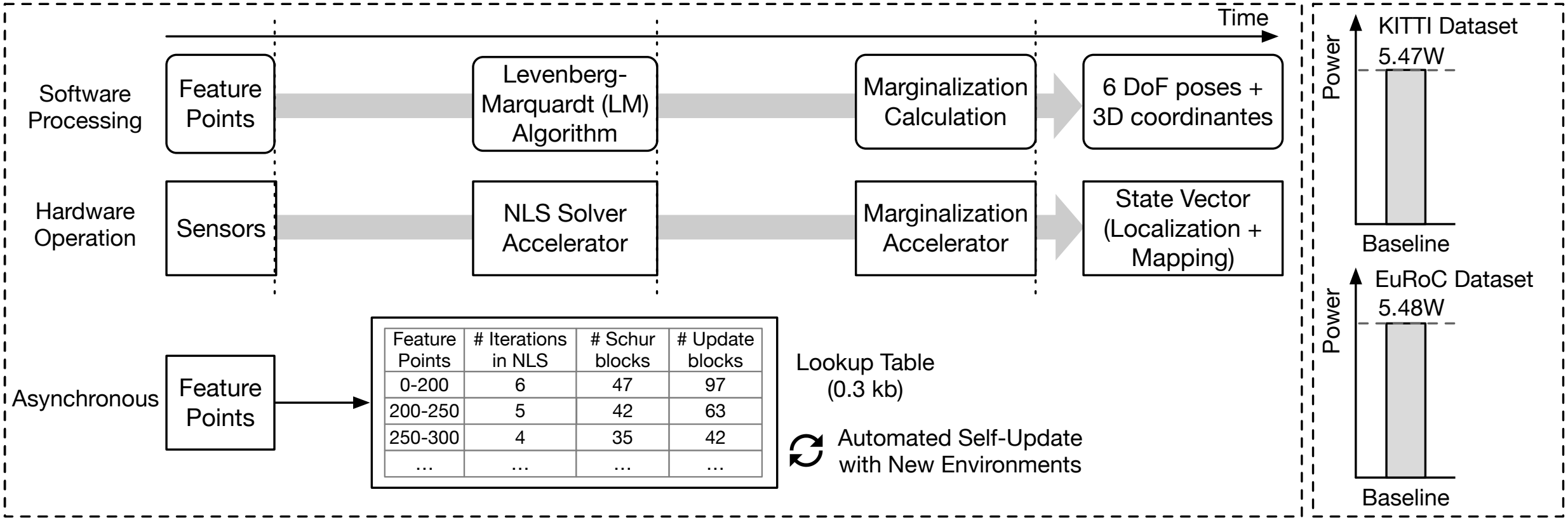
Key Design Technique 2

- Runtime Reconfiguration & Clock Gating



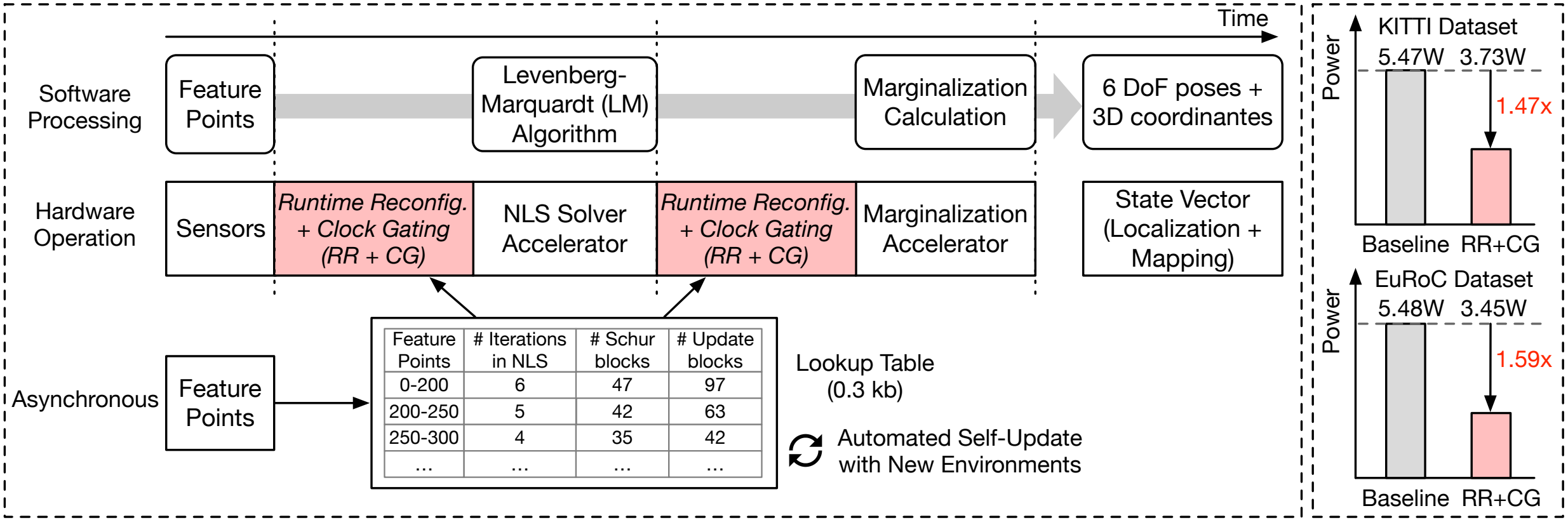
Key Design Technique 2

- Runtime Reconfiguration & Clock Gating

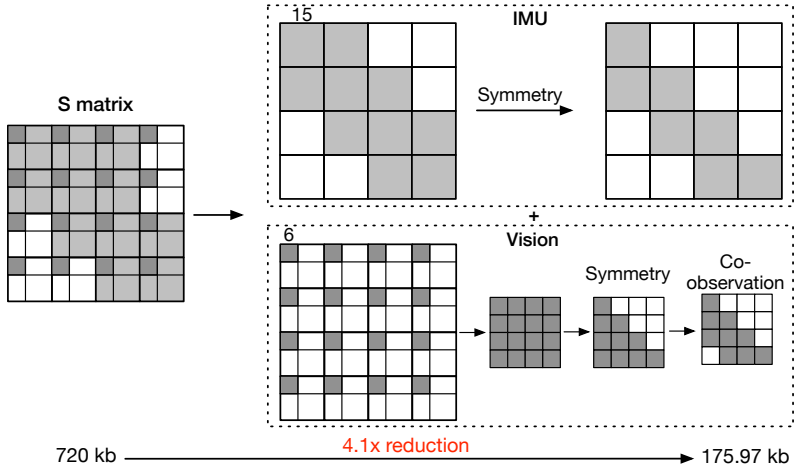


Key Design Technique 2

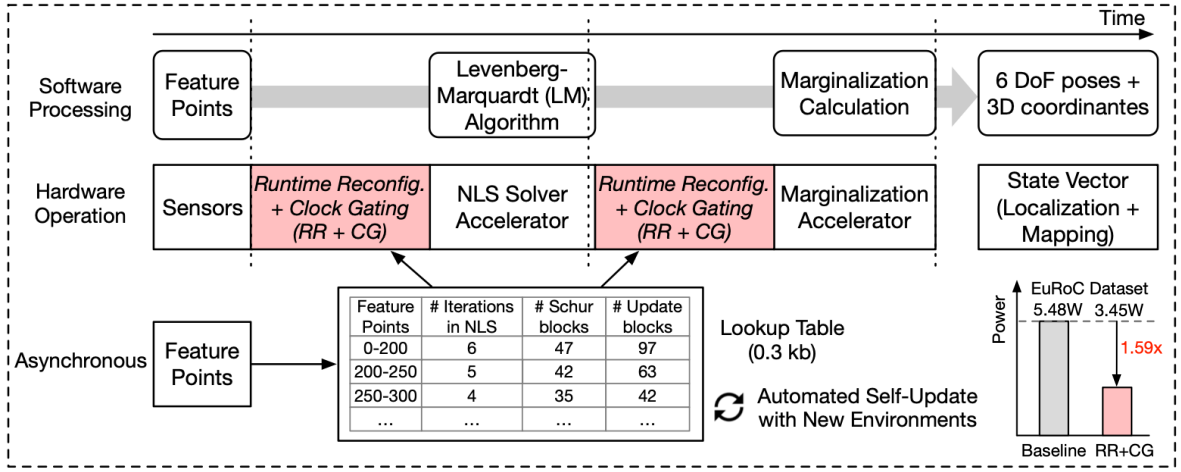
- Runtime Reconfiguration & Clock Gating



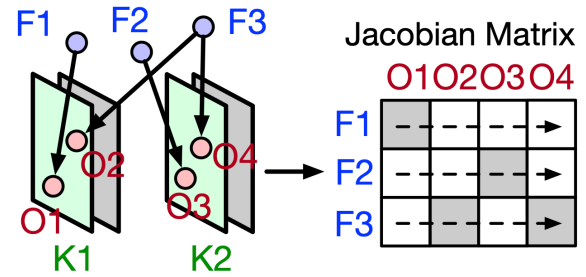
Key Design Techniques



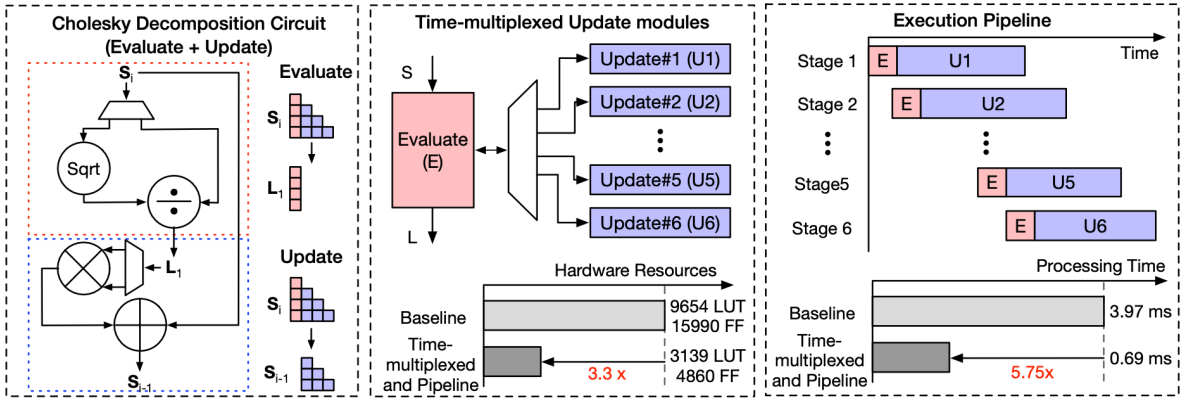
Memory layout, symmetry, sparsity



Runtime reconfigurability and clock gating



Data reuse and dataflow



Time-multiplexing and pipelining

[@CICC'22]

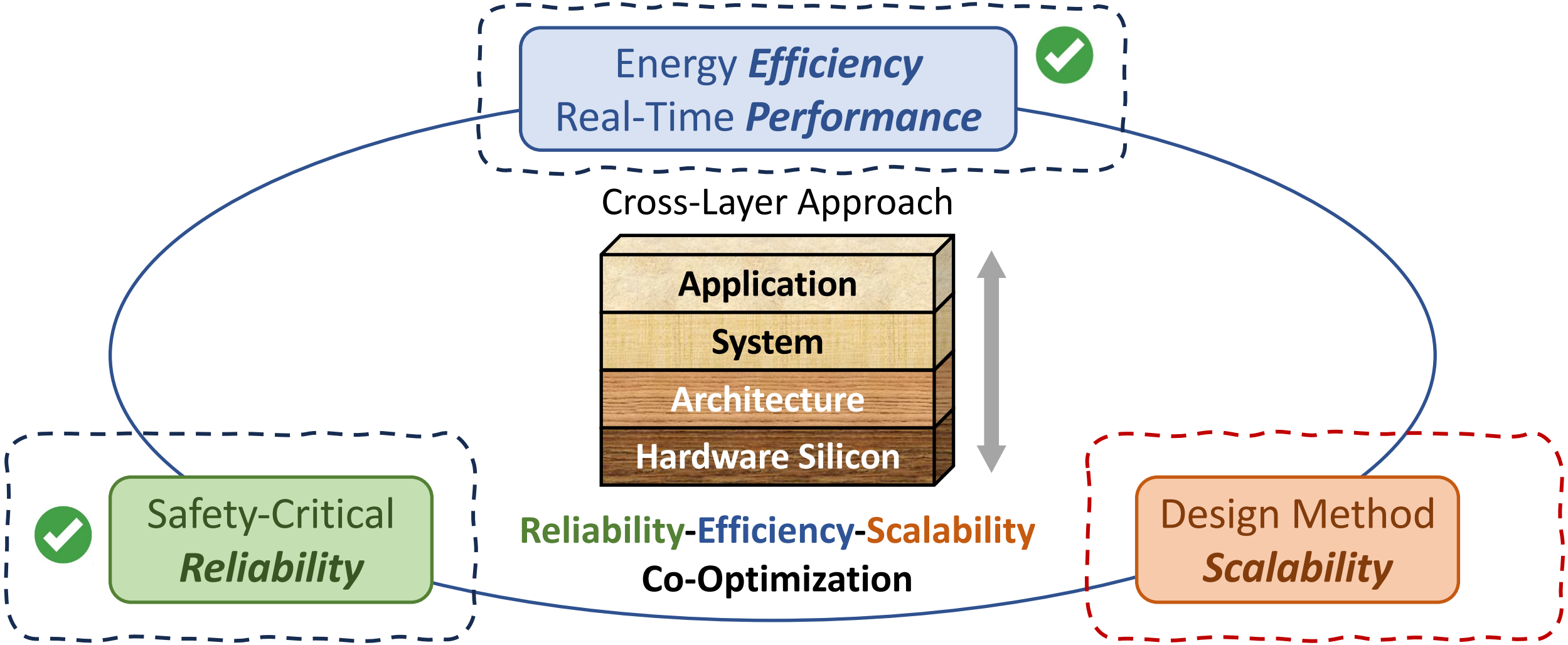
[@ISSCC'23]

Key Takeaways:

Domain-specific co-design and **design-technology co-optimization** unlock system performance and efficiency

Autonomous machines need **spatial-aware computing** that consider environment dynamics and heterogeneity

My Research: Autonomous Machine Computing



Current Robotic Domain-Specific Accelerators

Robot Applications


sweeper


Manipulator

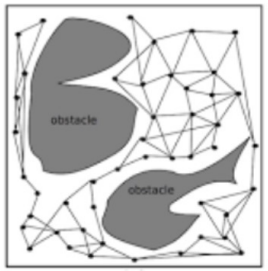

Vehicle


Drone

Robot Algorithms


Localization


Planning



Control

Solution 1

Programmable Accelerators

Strength: High generality

Weakness: Less effective in exploiting specific sparse structures

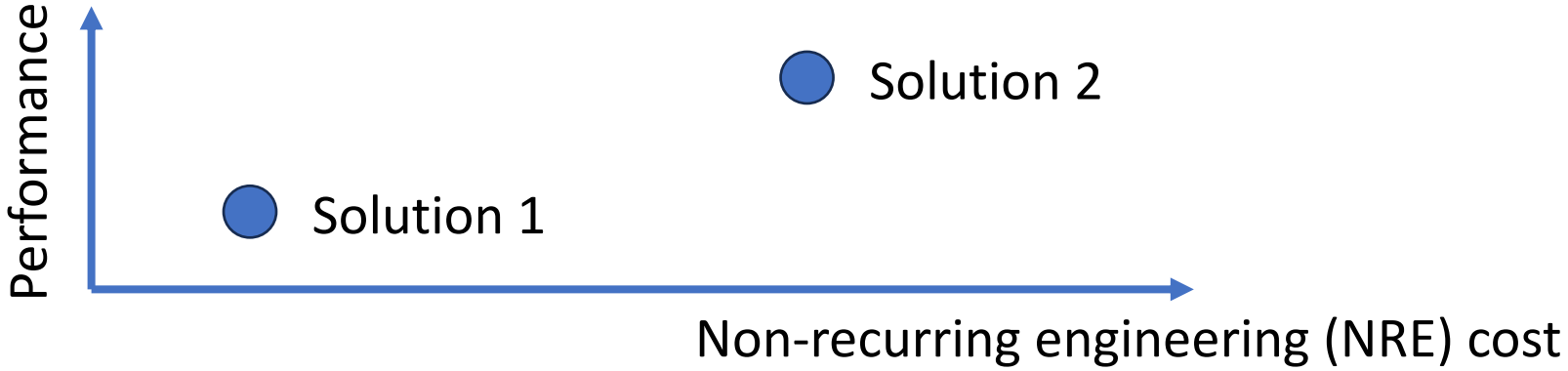
Solution 2

Dedicated Accelerators

Strength: High performance

Weakness: High NRE costs; Stacking accelerators requires large chip area

Current Robotic Domain-Specific Accelerators



Solution 1

Programmable Accelerators

Strength: High generality
Weakness: Less effective in exploiting specific sparse structures

Solution 2

Dedicated Accelerators

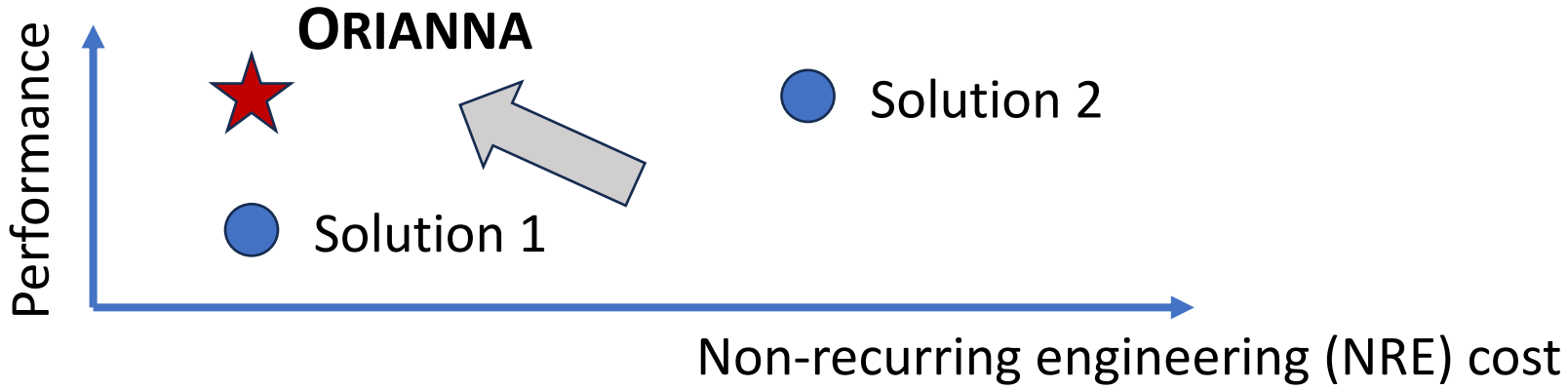
Strength: High performance
Weakness: High NRE costs; Stacking accelerators requires large chip area

Research Question:

How can we improve robotics domain-specific accelerators **design adaptability and scalability**?

[@ASPLOS'24]

Orianna: Accelerator Generation Framework for Optimization-Based Robotic Applications



Solution 1

Programmable Accelerators

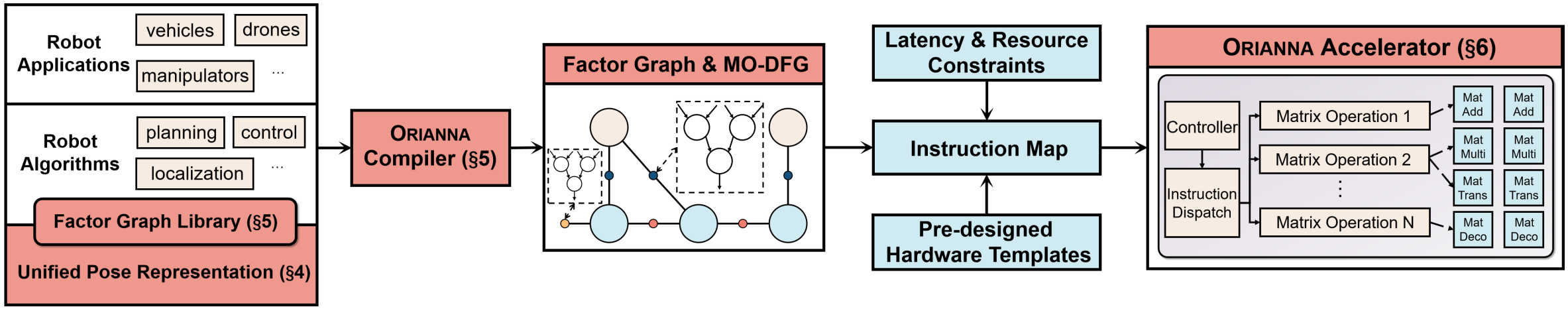
Strength: High generality
Weakness: Less effective in exploiting specific sparse structures

Solution 2

Dedicated Accelerators

Strength: High performance
Weakness: High NRE costs; Stacking accelerators requires large chip area

Orianna Framework



ORIANNA Software

Use **factor graph** as unified abstraction; Build flexible **factor graph software library**

Performance ↑

ORIANNA Compiler

Compiler transforms diverse algorithms into **unified factor graph instructions** and **data flow graph**

Generality & Scalability ↑

ORIANNA Accelerator

Different robotic algorithms can be executed **concurrently** on the **Factor Graph Accelerator**

Resource Utilization ↑

Key Takeaways:

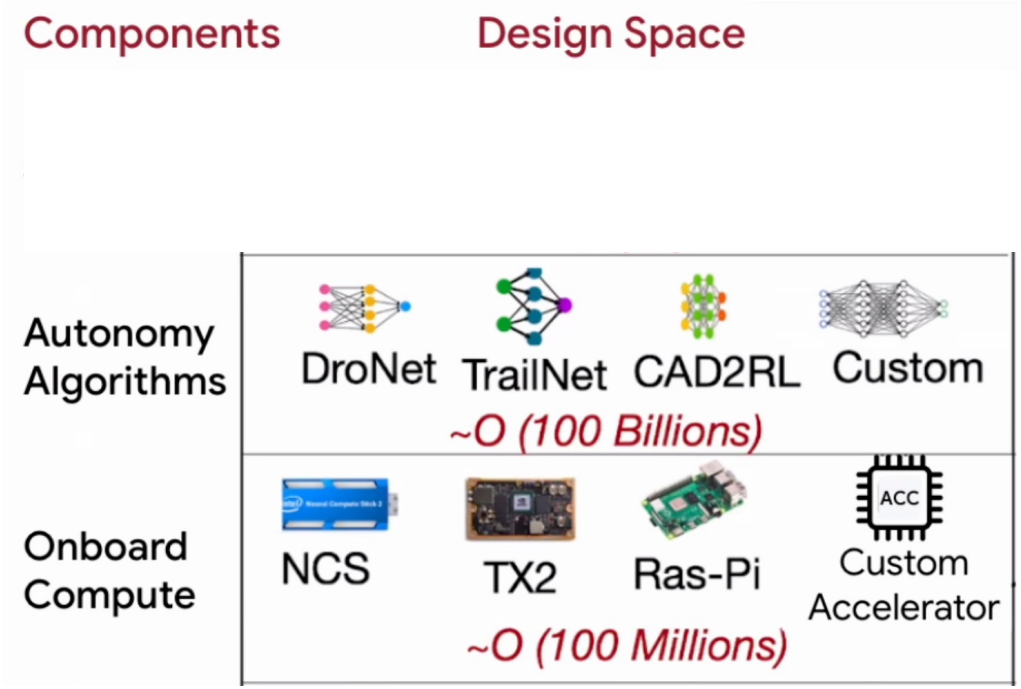
Focusing on **fragments in isolation** leads to **overlooking interdependencies and system-wide implications**.

Prioritizing **system morphology** provides greater visibility, that can lead to greater DSA adaptability and efficiency.

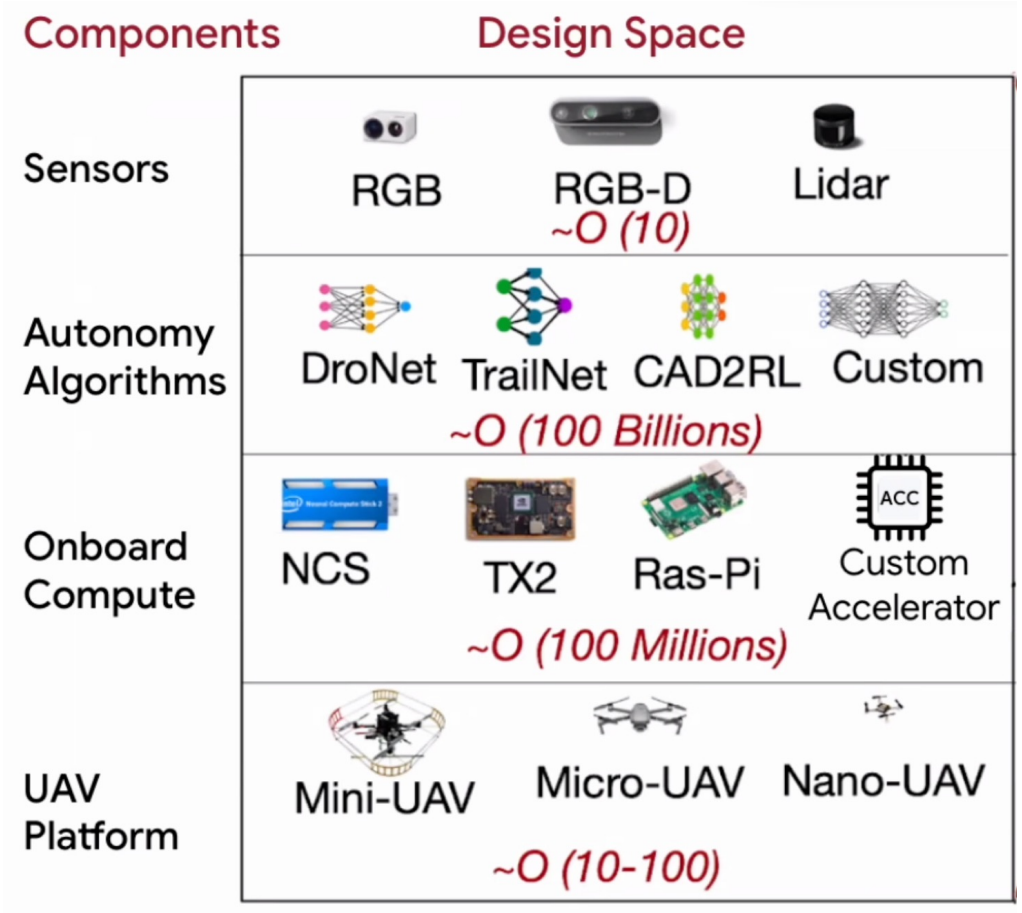
Large Design Space

Components	Design Space
Onboard Compute	

Large Design Space



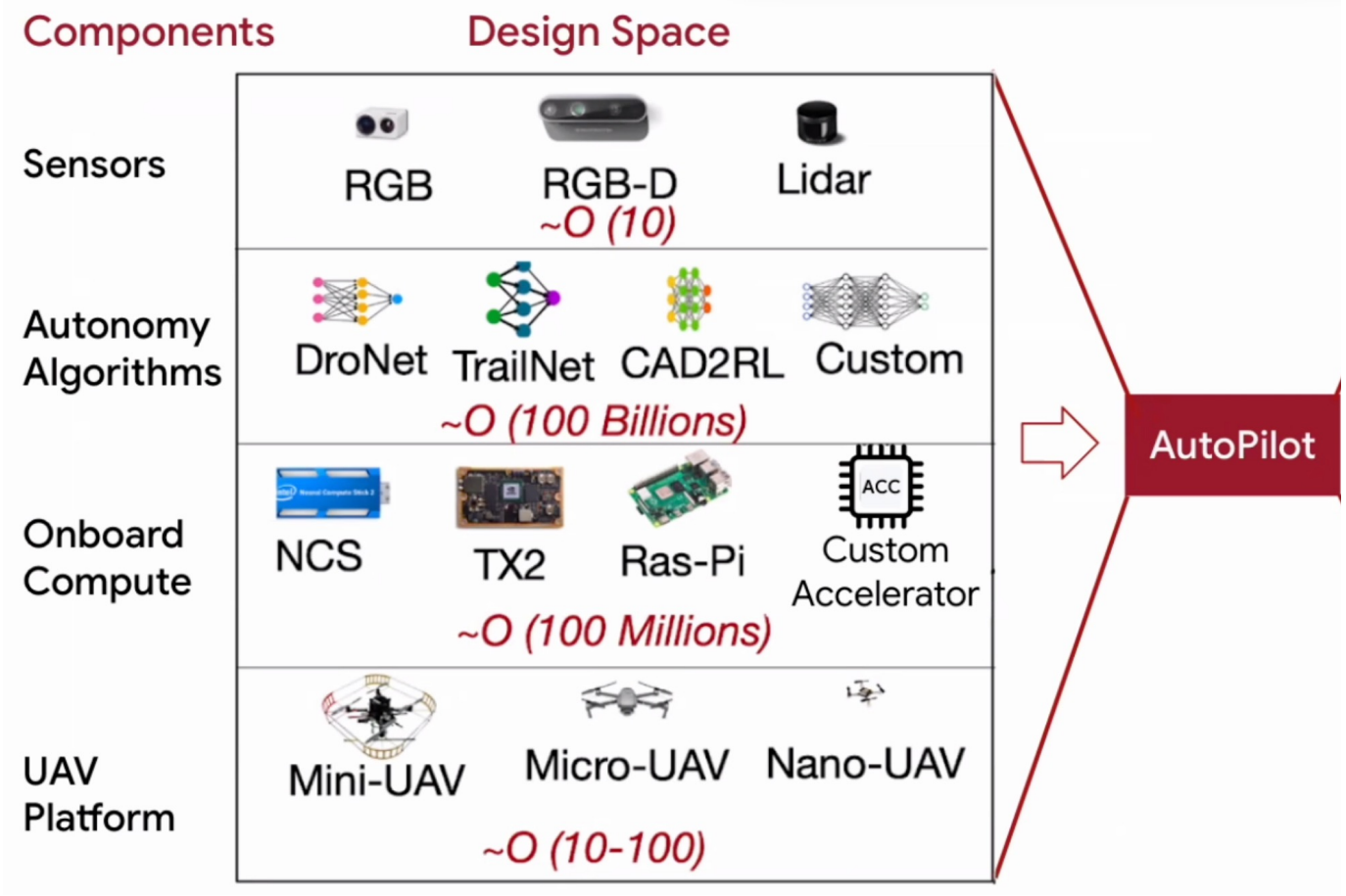
Large Design Space



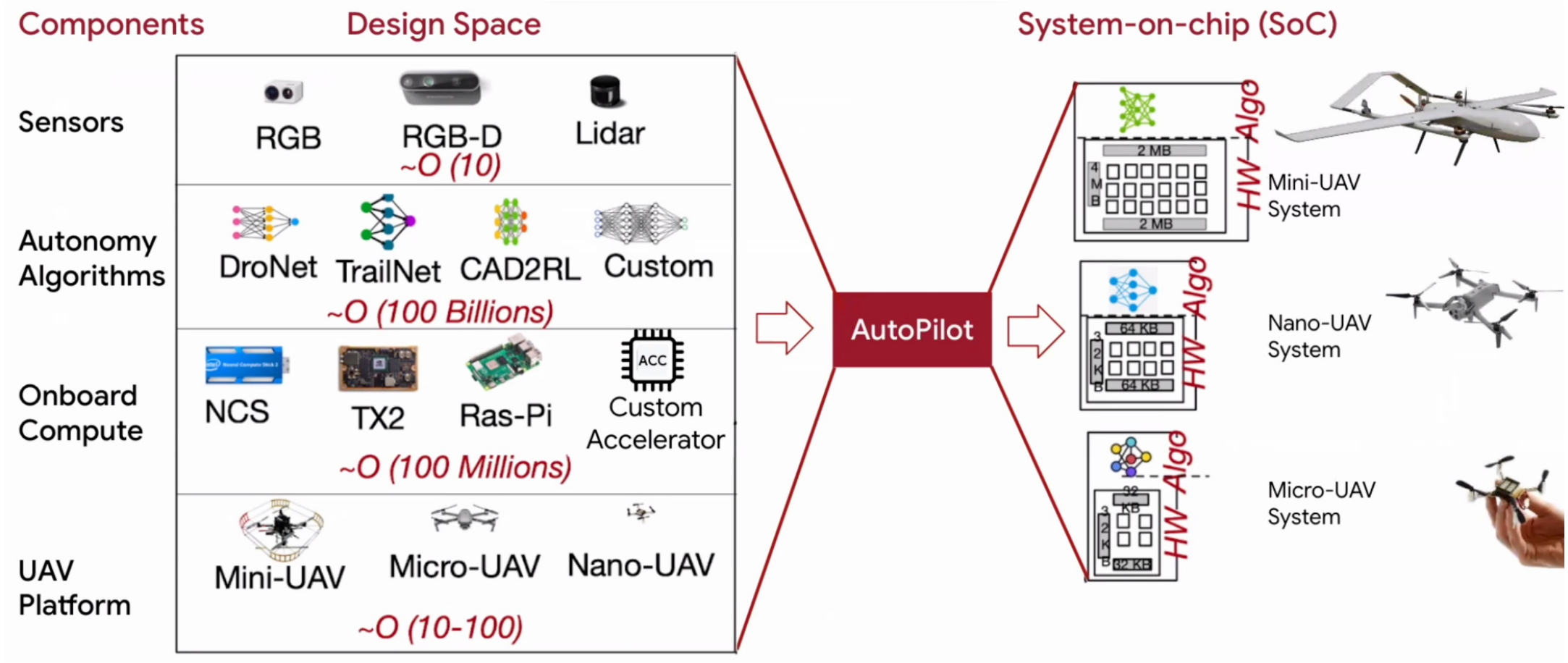
Research Question:

How can we design DSAs to handle the increasing levels of system complexity?

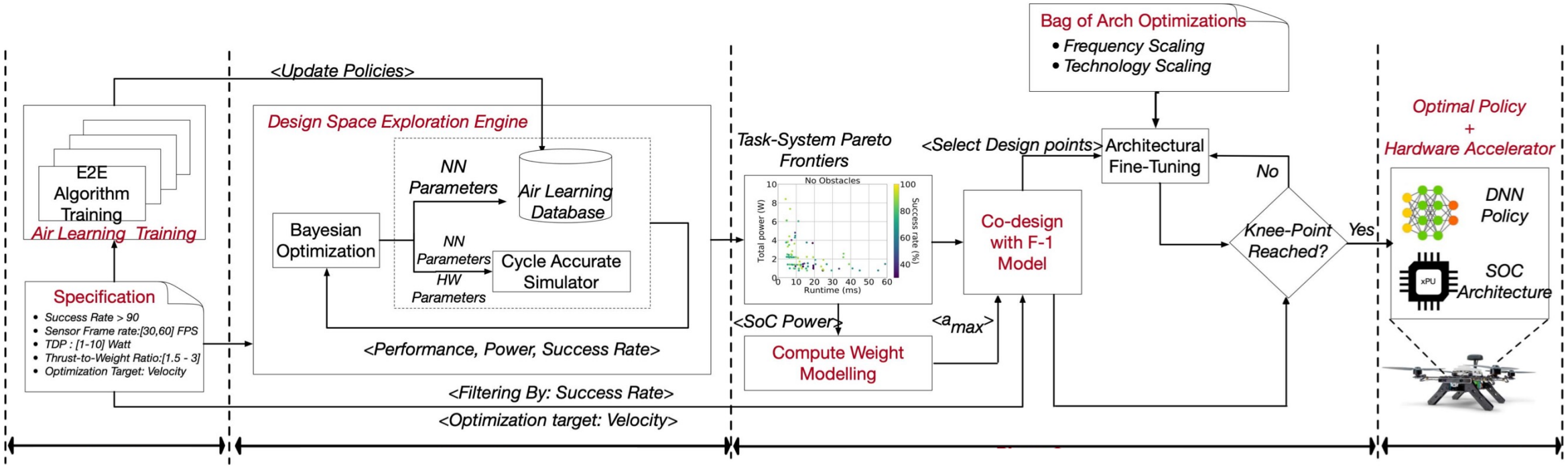
Our Solution: AutoPilot



Our Solution: AutoPilot

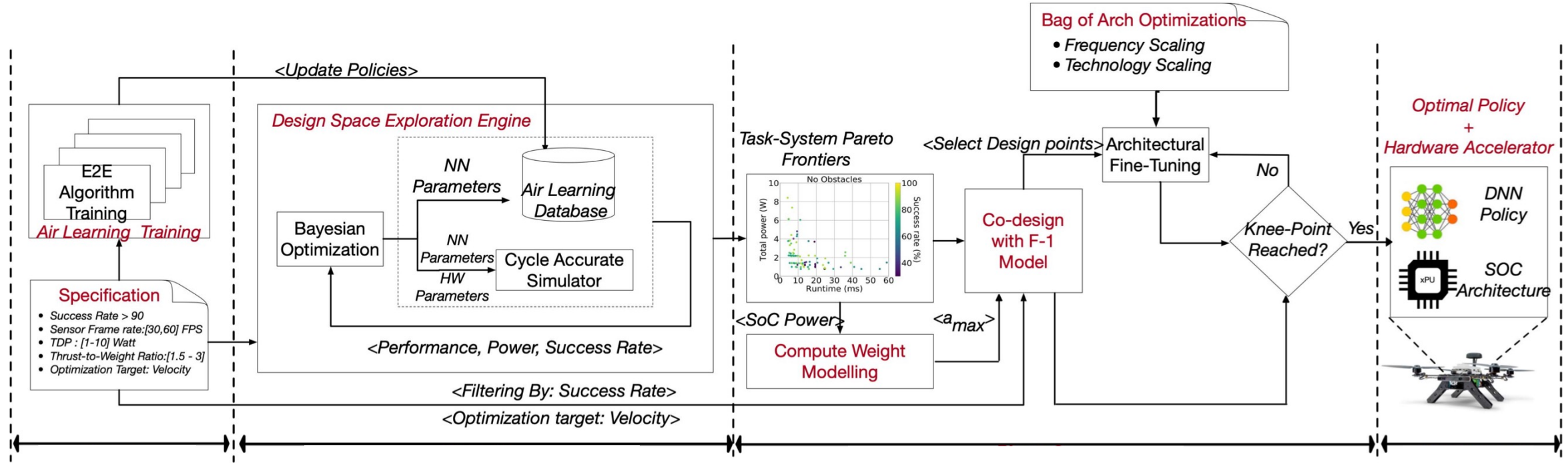


AutoPilot Framework



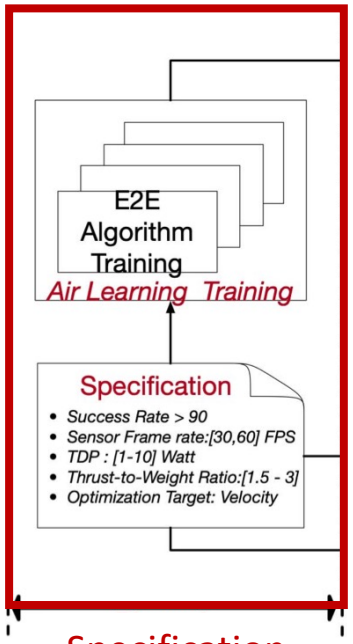
Automating SoC Design Space Exploration for SWaP Constrained Autonomous UAVs

AutoPilot Framework



Specification Phase | Optimization Phase | Mapping Phase | Deployment Phase

Domain Specific | Domain Agnostic | Domain Specific | Target System



Specification Phase

(Domain-specific)

Input the specification

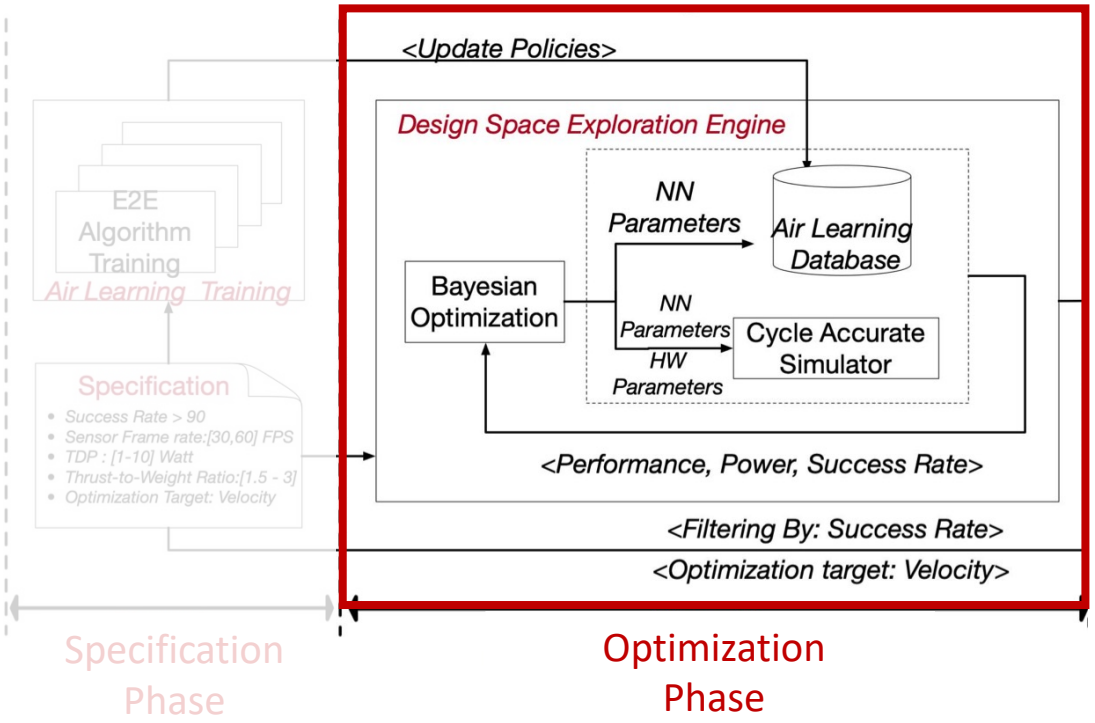
Specification

- Success Rate >90
- Sensor Frame rate: [30, 60] FPS
- TDP: [1-10] Watt
- Thrust-to-Weight Ratio: [1.5-3]
- Optimization Target: Velocity

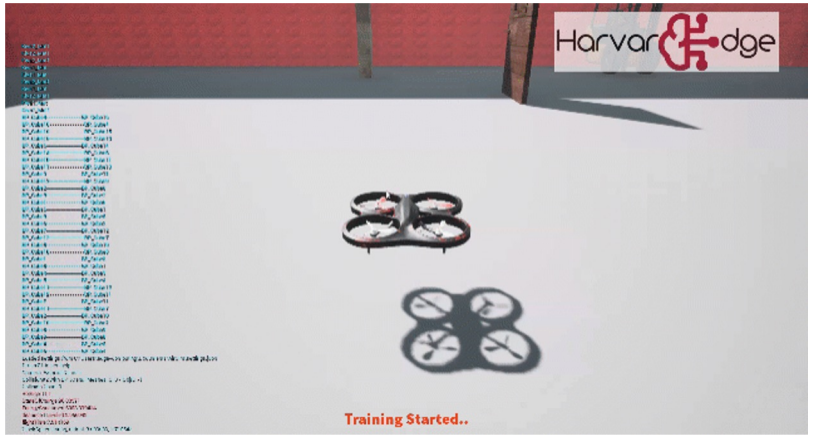
Sensor configuration
- Frame rate
- RGB
- LIDAR
- …

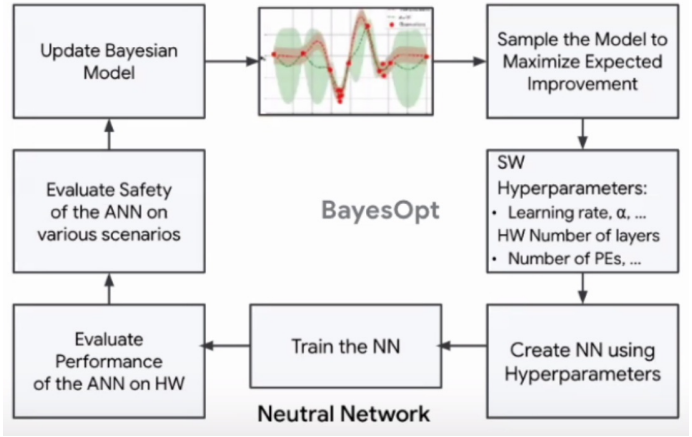
Compute configuration
- TDP
- Latency
- Throughput
- …

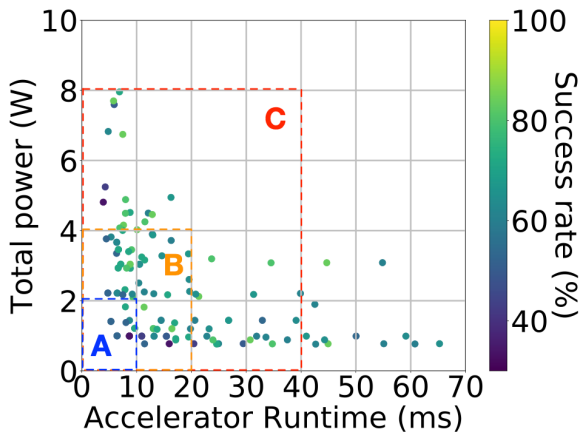
System configuration
- Weight
- Thrust
- …



(Domain-agnostic) Algorithm-Hardware Co-Design Optimization



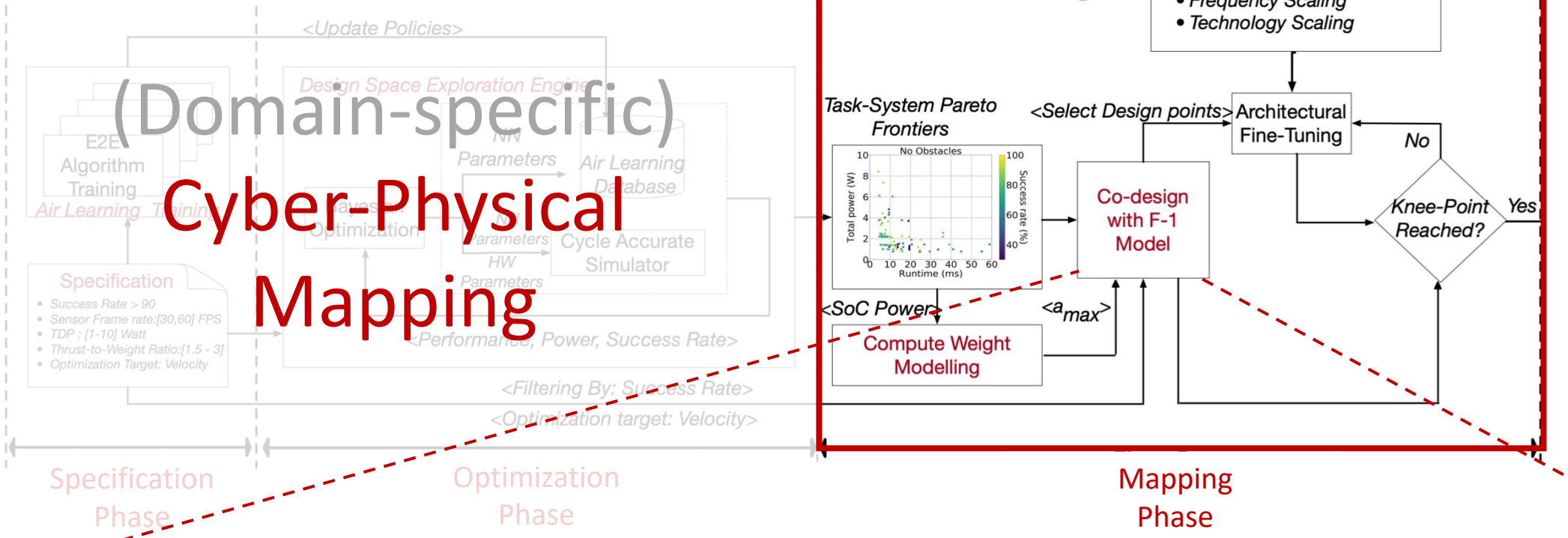


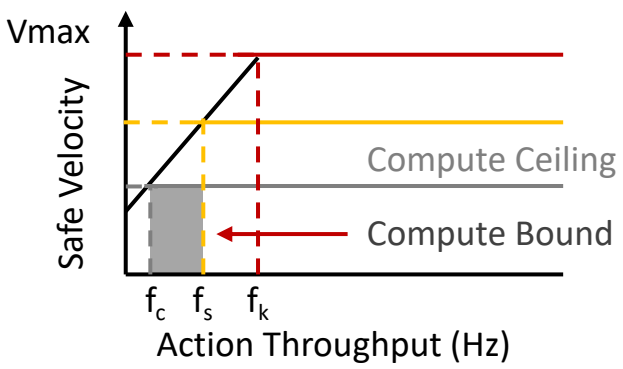


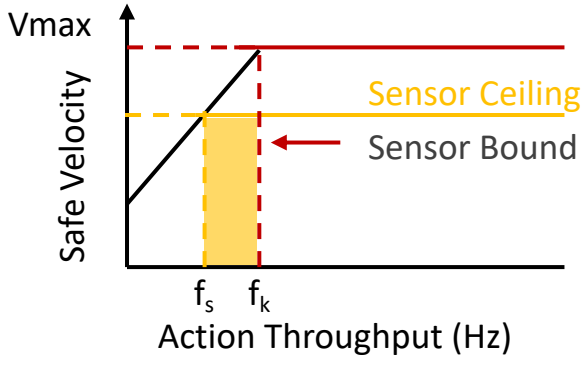
(Domain-specific) Cyber-Physical Mapping

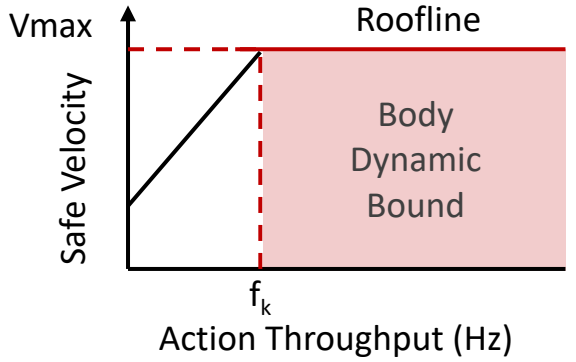
[@ISPASS'23]

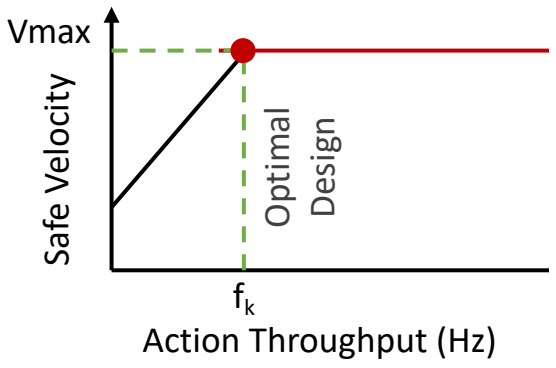
[@CAL'20]

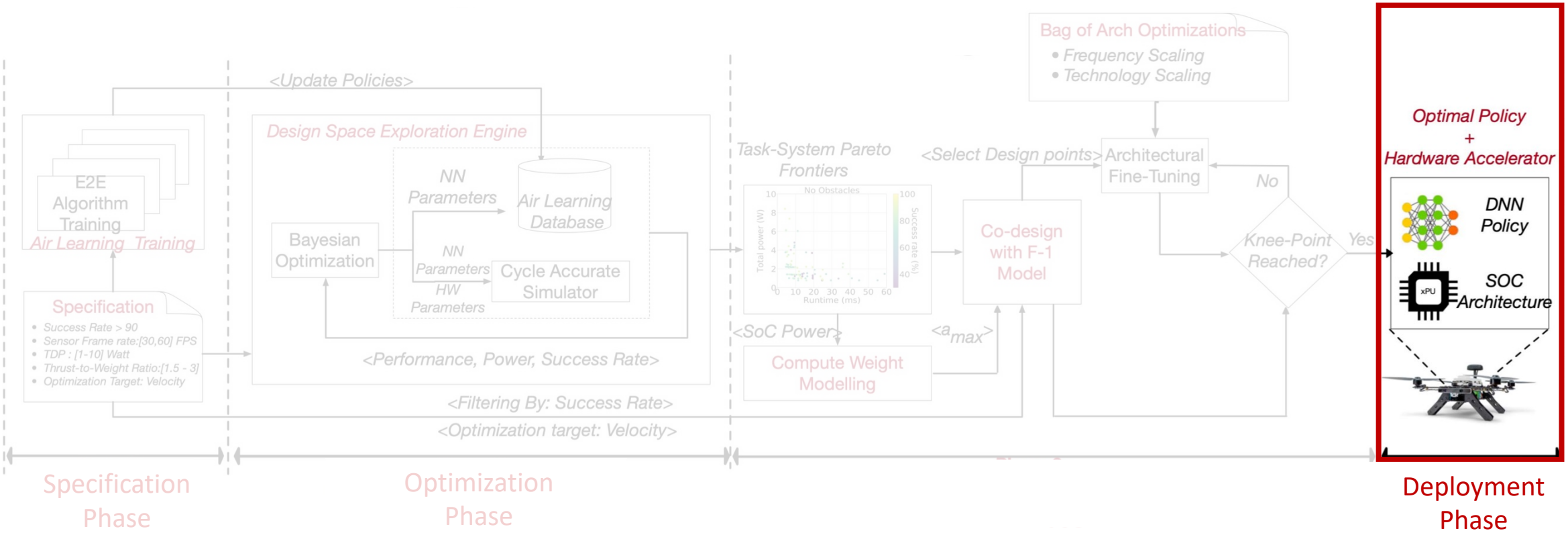






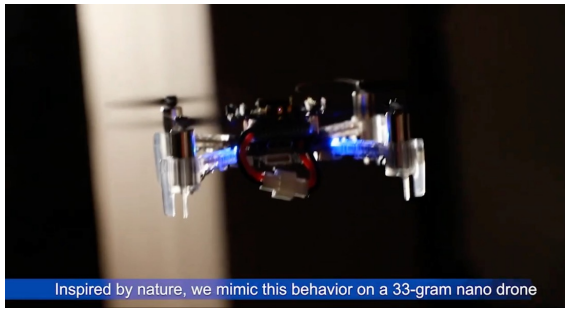






Mini UAV

Micro UAV



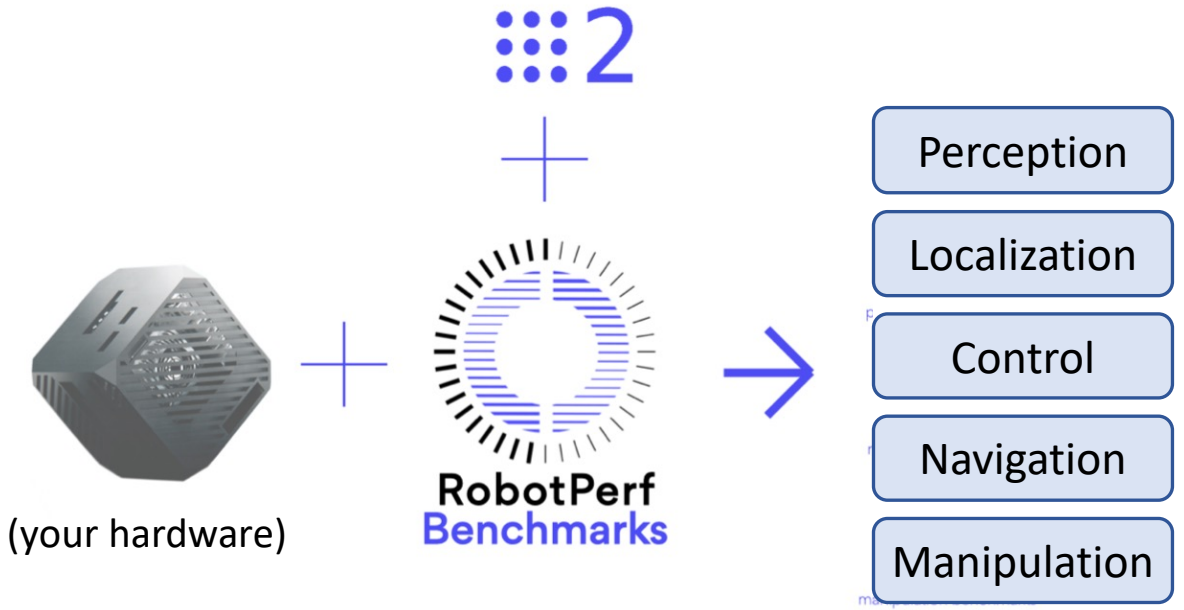
Nano UAV

[@ICRA'24]

RobotPerf

"If You Can't Measure It, You Can't Improve it" - Peter Drucker

- A Benchmarking Suite for Evaluating Robotics Computing Performance



Collaborative efforts across 10+ universities & industries

[@MICRO'23]

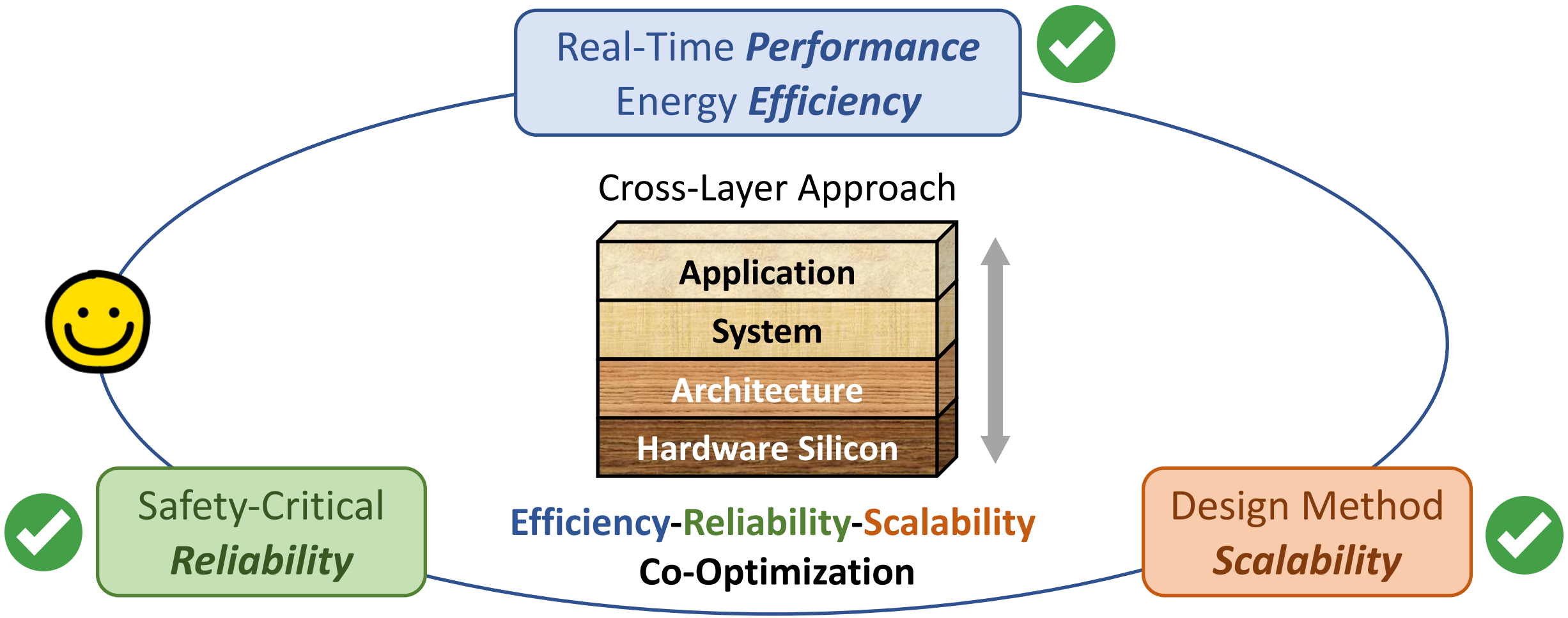
[@ISPASS'23]

Key Takeaways:

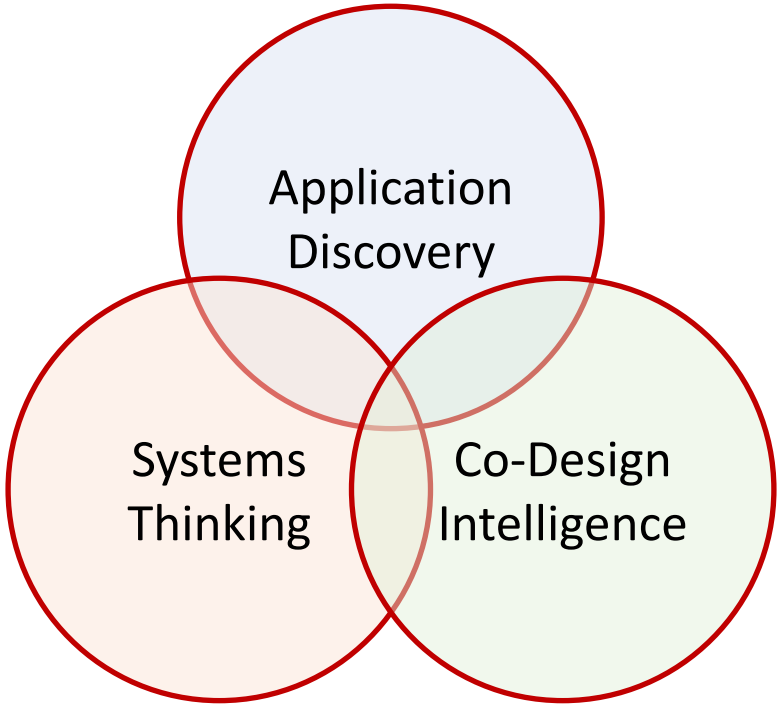
Focusing on **isolated individual components** for DSAs can cause us to miss the forest for the trees

An **end-to-end, holistic approach** to system design needs to more **practical solutions**.

Summary: Autonomous Machine Computing



Summary: Core Research Methodology



01. Application Discovery: Deeply understanding an application through deep characterization to identify and address the underlying problem space.

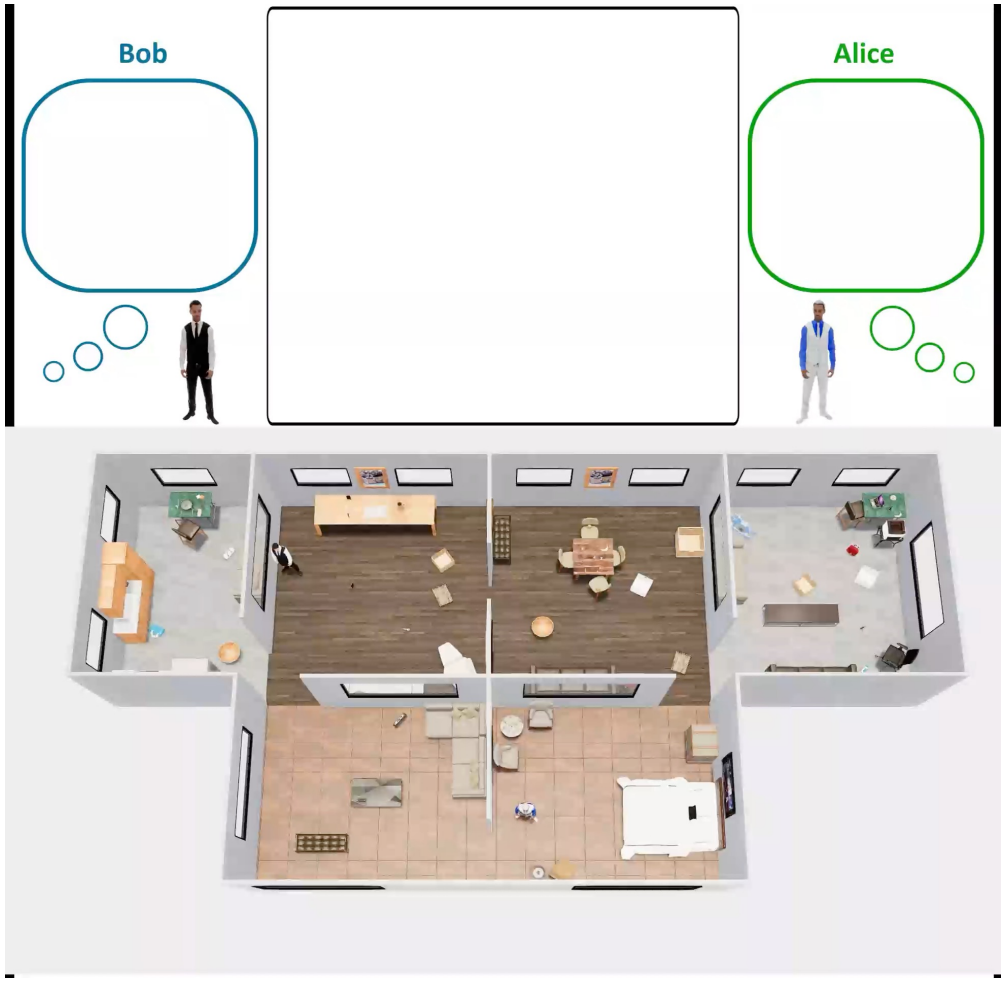
02. Systems Thinking: End-to-end design of complex systems, where every element is considered as interconnected and part of a larger, integrated whole

03. Co-Design Intelligence: Developing tools, methodologies, and frameworks that incorporate insights from application discovery and systems thinking to automatically design optimized solutions

Next-Step: Embodied AI Agent System

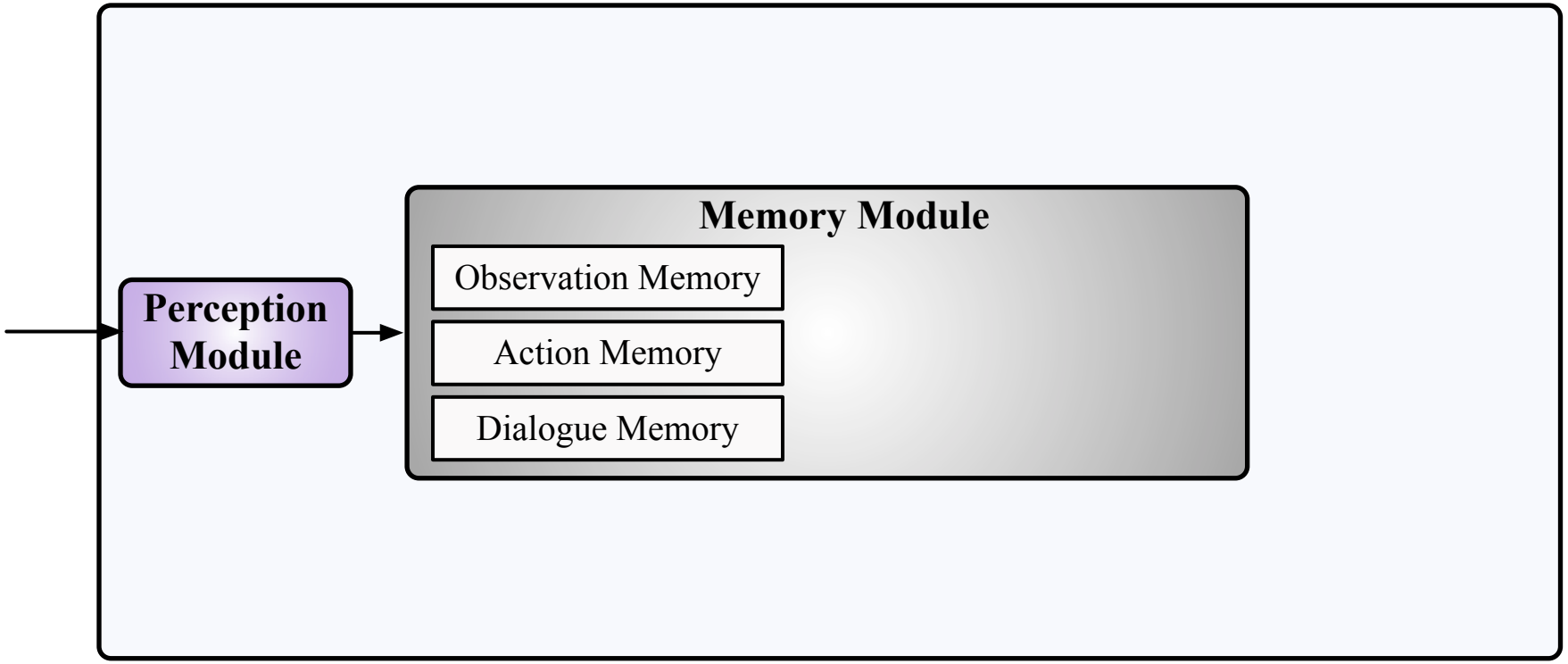
- **Task**: long-horizon multi-objective task and motion planning
  - Examples: household tasks, transport objects, make meal, set up table, cook…

Demo: Long-Horizon Multi-Objective Planning

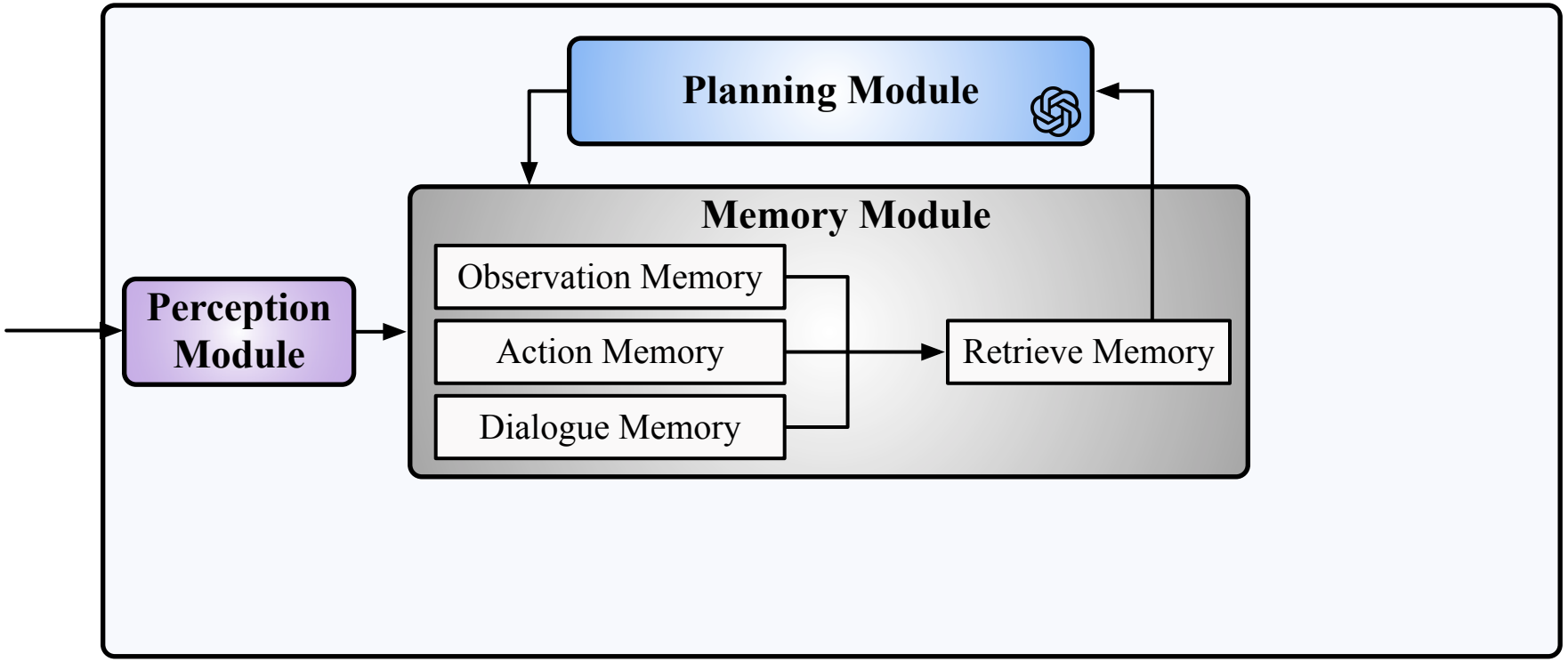


Embodied AI Agent Workflow

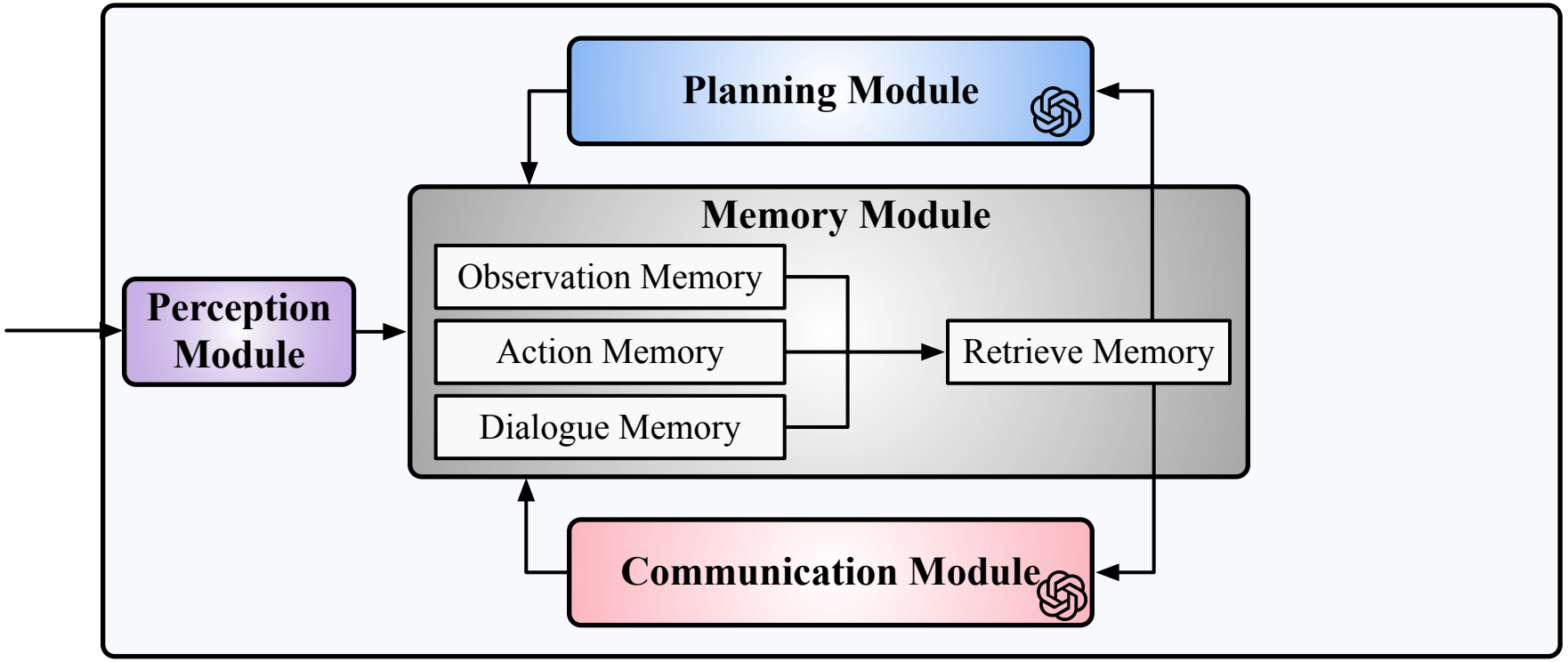
Embodied AI Agent Workflow



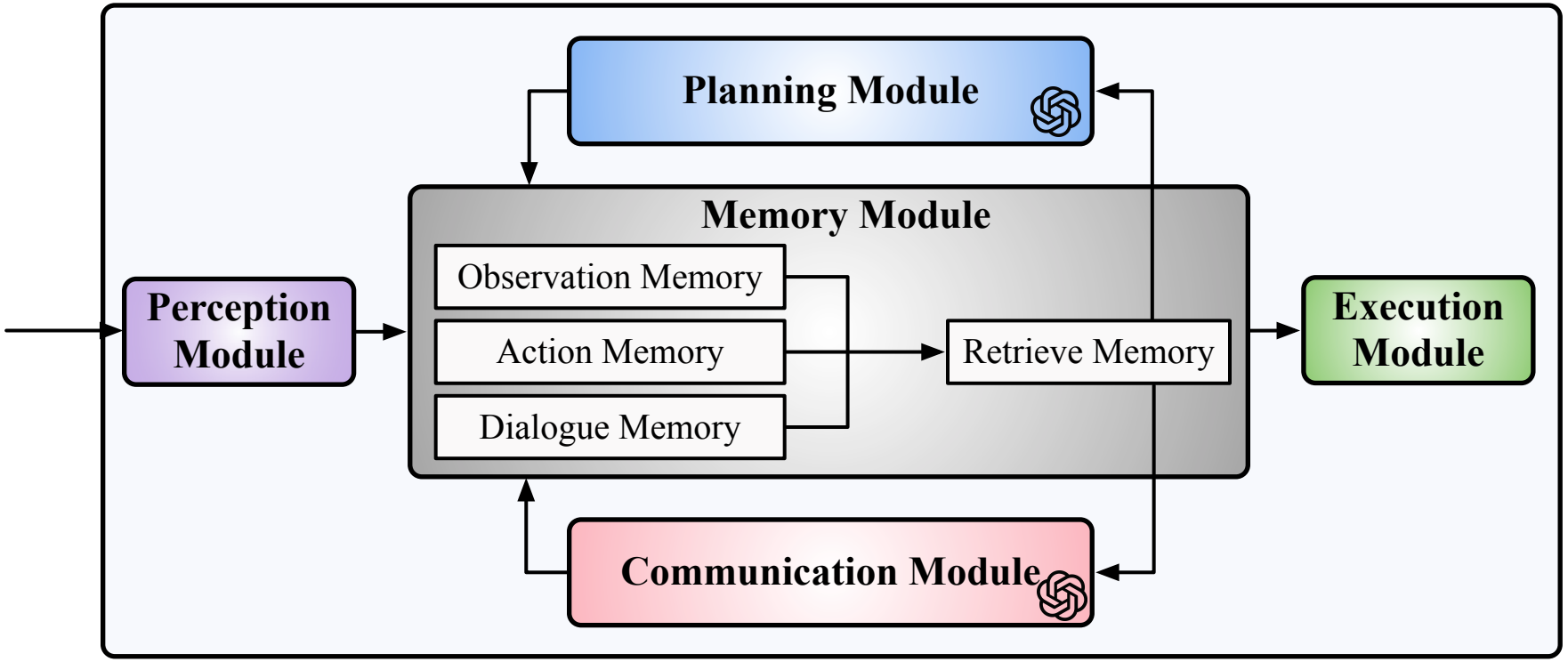
Embodied AI Agent Workflow



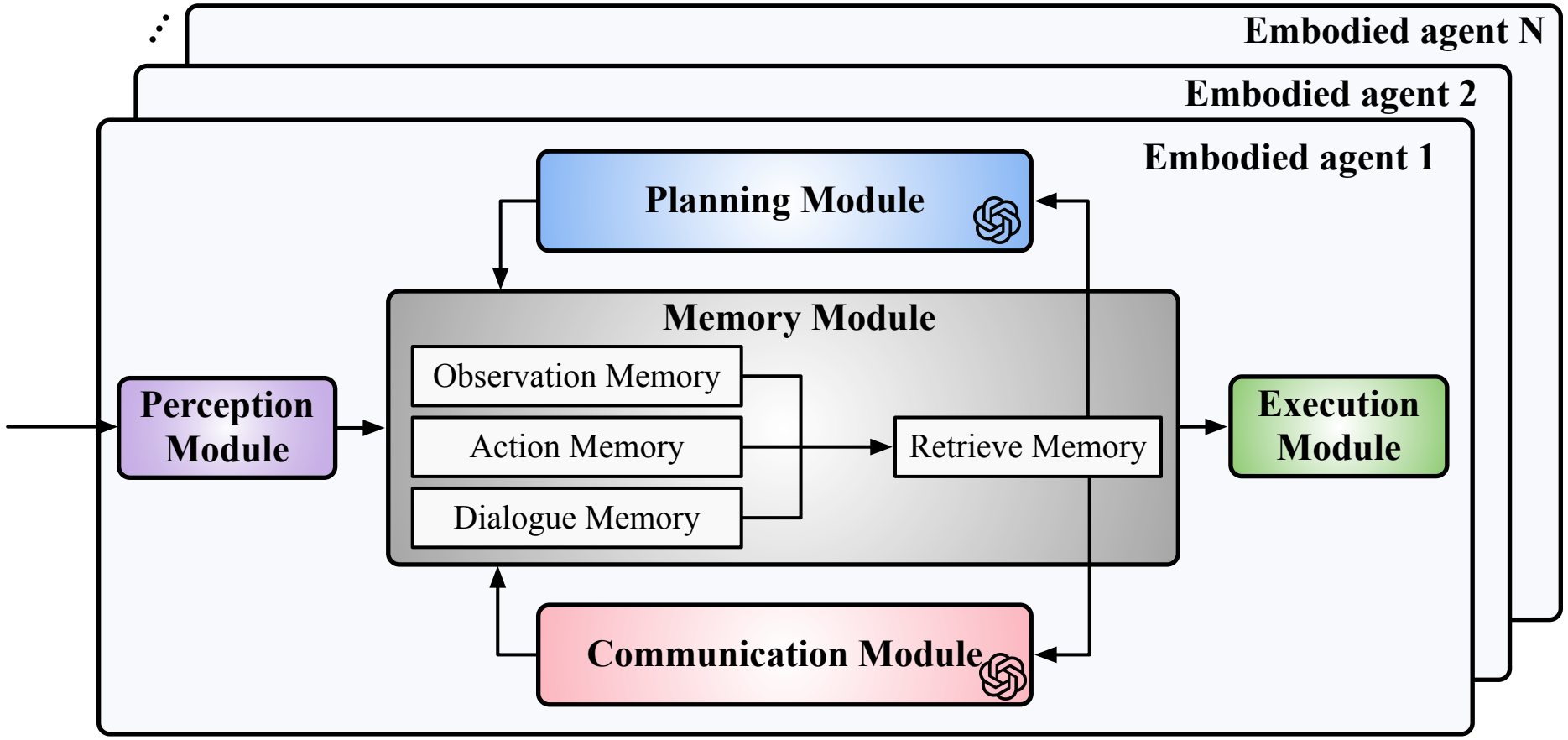
Embodied AI Agent Workflow



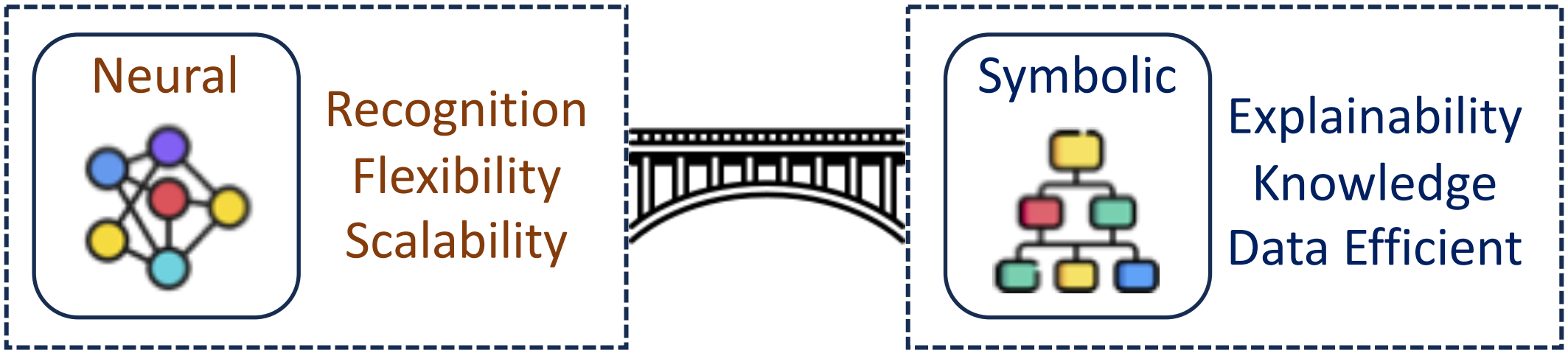
Embodied AI Agent Workflow



Embodied AI Agent Workflow

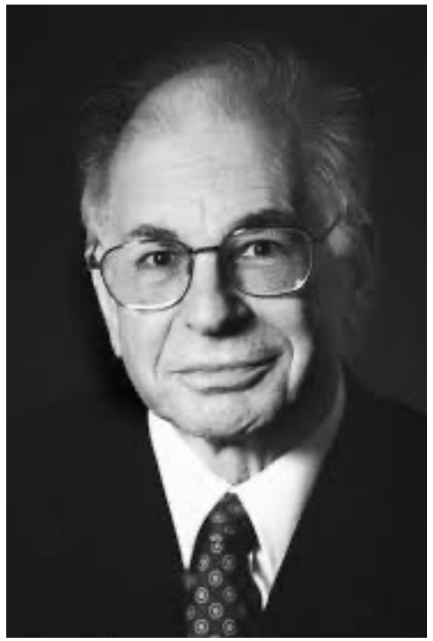


Next-Step: Neurosymbolic Cognitive Agent

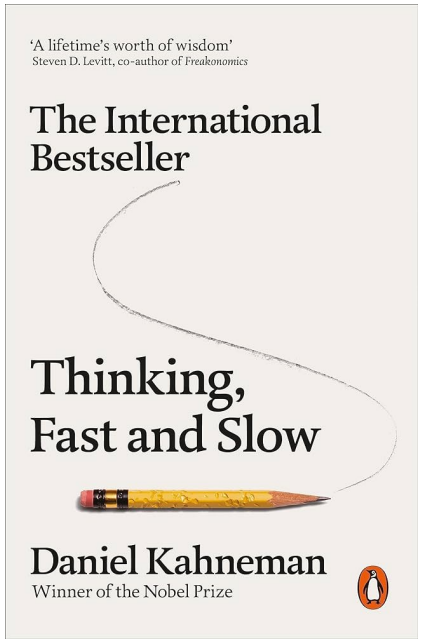


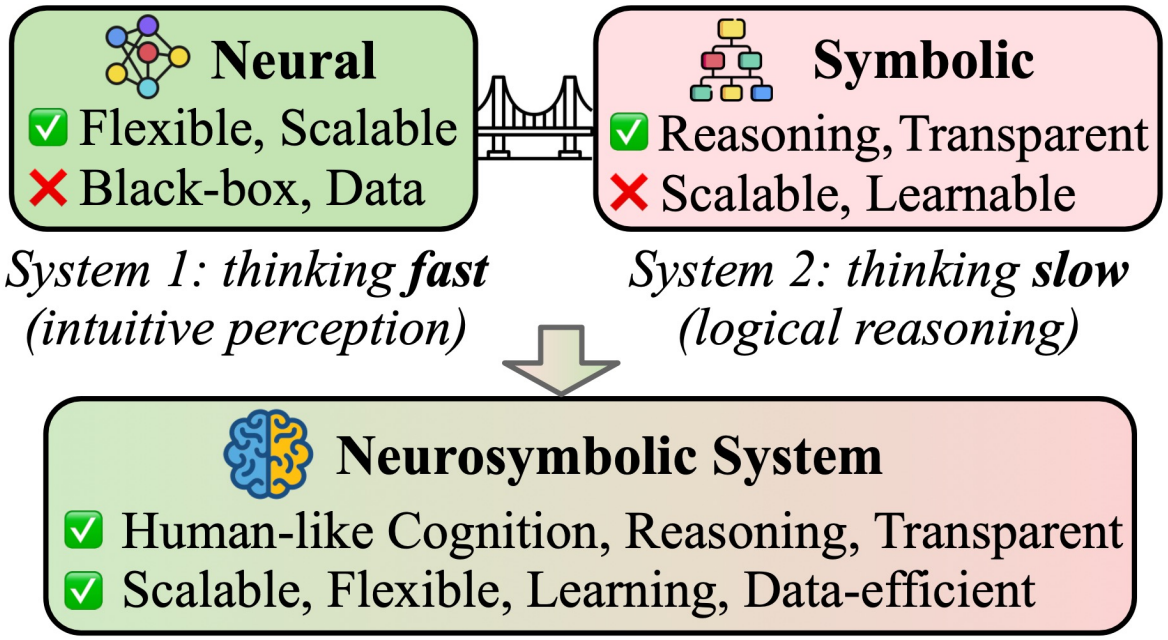
Towards Cognitive and Trustworthy AI Agent Systems

Relationship to Human Minds



Daniel Kahneman (1934-2024)

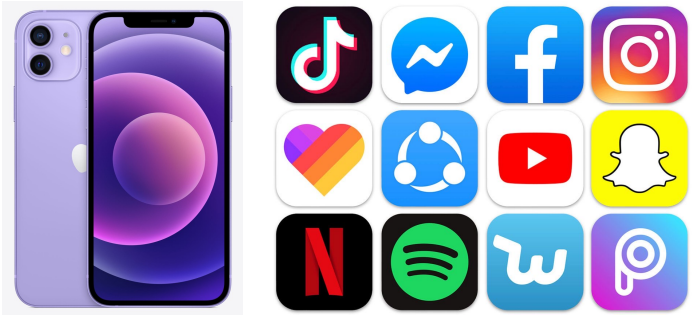




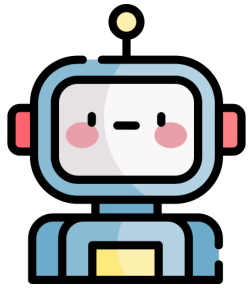
Our Vision

90% basic functions
10% end-user applications

10% basic functions
90% end-user applications

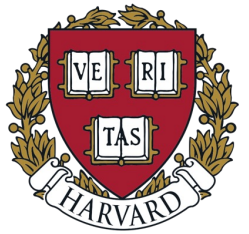


90% basic functions
10% end-user applications



10% basic functions
90% **<You Can Decide>**

The Future of
Autonomous Machine Computing
is **Bright** and **At-Scale**!



Intelligence in Robotic Computing: Cross-Layer Co-Design for Resilient and Efficient Autonomous Machines

Zishen Wan

PhD Student @ Georgia Tech

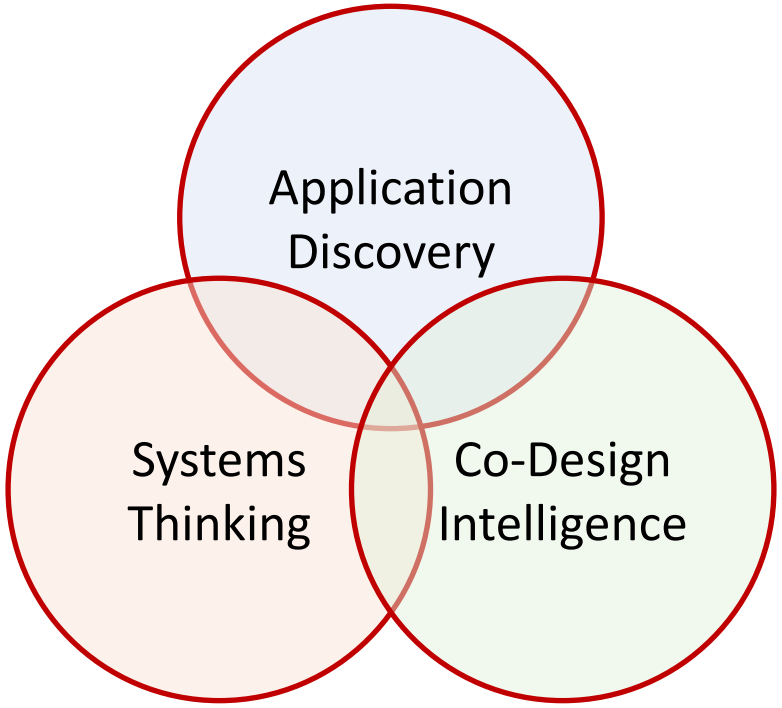
w/ Profs. Arijit Raychowdhury, Tushar Krishna, Vijay Janapa Reddi

Web: https://zishenwan.github.io

Email: zishenwan@gatech.edu

RoboArch Workshop @ MICRO, Student Showcase Session, Nov. 3, 2024

Summary: Core Research Methodology



01. Application Discovery: Deeply understanding an application through deep characterization to identify and address the underlying problem space.

02. Systems Thinking: End-to-end design of complex systems, where every element is considered as interconnected and part of a larger, integrated whole

03. Co-Design Intelligence: Developing tools, methodologies, and frameworks that incorporate insights from application discovery and systems thinking to automatically design optimized solutions