

HCAPP: Scalable Power Control for Heterogeneous 2.5D Integrated Systems

Kramer Straube[†], Jason Lowe-Power^{*}, Christopher
Nitta^{*}, Matthew Farrens^{*}, Venkatesh Akella[†]

[†]Department of Electrical &
Computer Engineering
University of California, Davis



^{*}Department of Computer Science
University of California, Davis

Summary

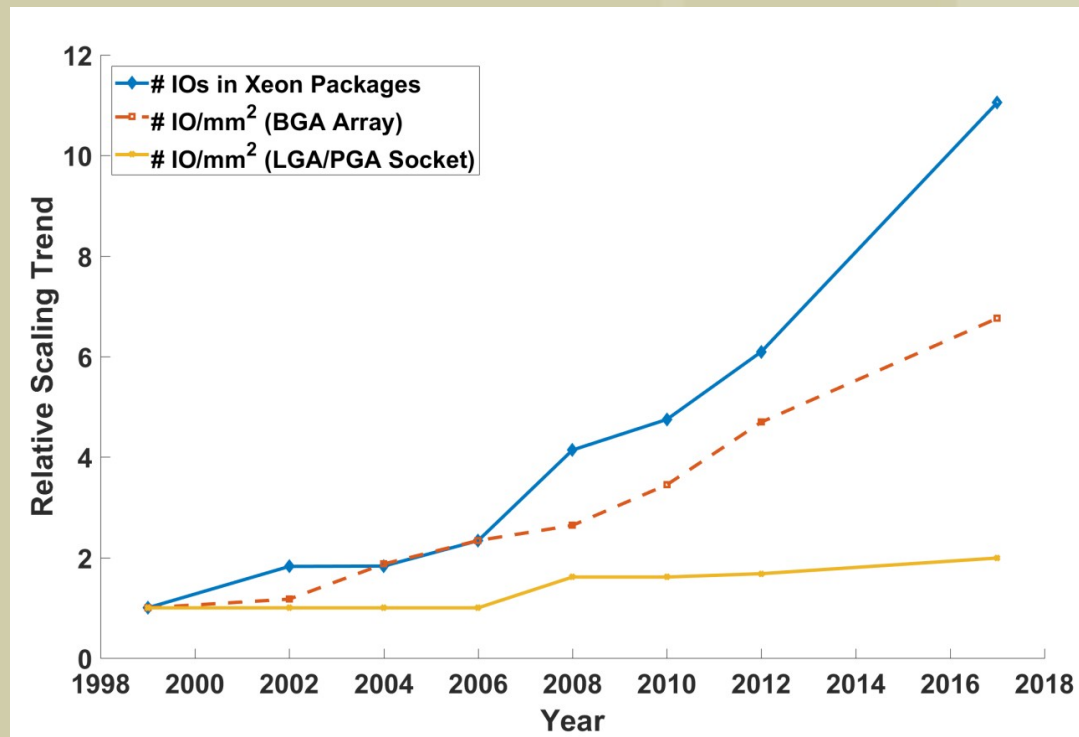
- 2.5D systems are limited by the available package pins
 - Many of these pins are used for supplying power
- Increasing utilization of these pins enables higher performance
- HCAPP
 - ensures a maximum power (eg. package pin limitation)
 - decoupled control through the power supply network
- **21% - 43% geomean speedup**
(on-die and off-die time constraint)

Background

- Next gen computation speedup requires heterogeneous machines
 - Currently CPU+GPU systems exist (Summit and Sierra)
- Accelerators provide speedups that are not reliant on Moore's law
- 2.5D integration (shared interposer + specialized dies) allows increased scalability

Motivation

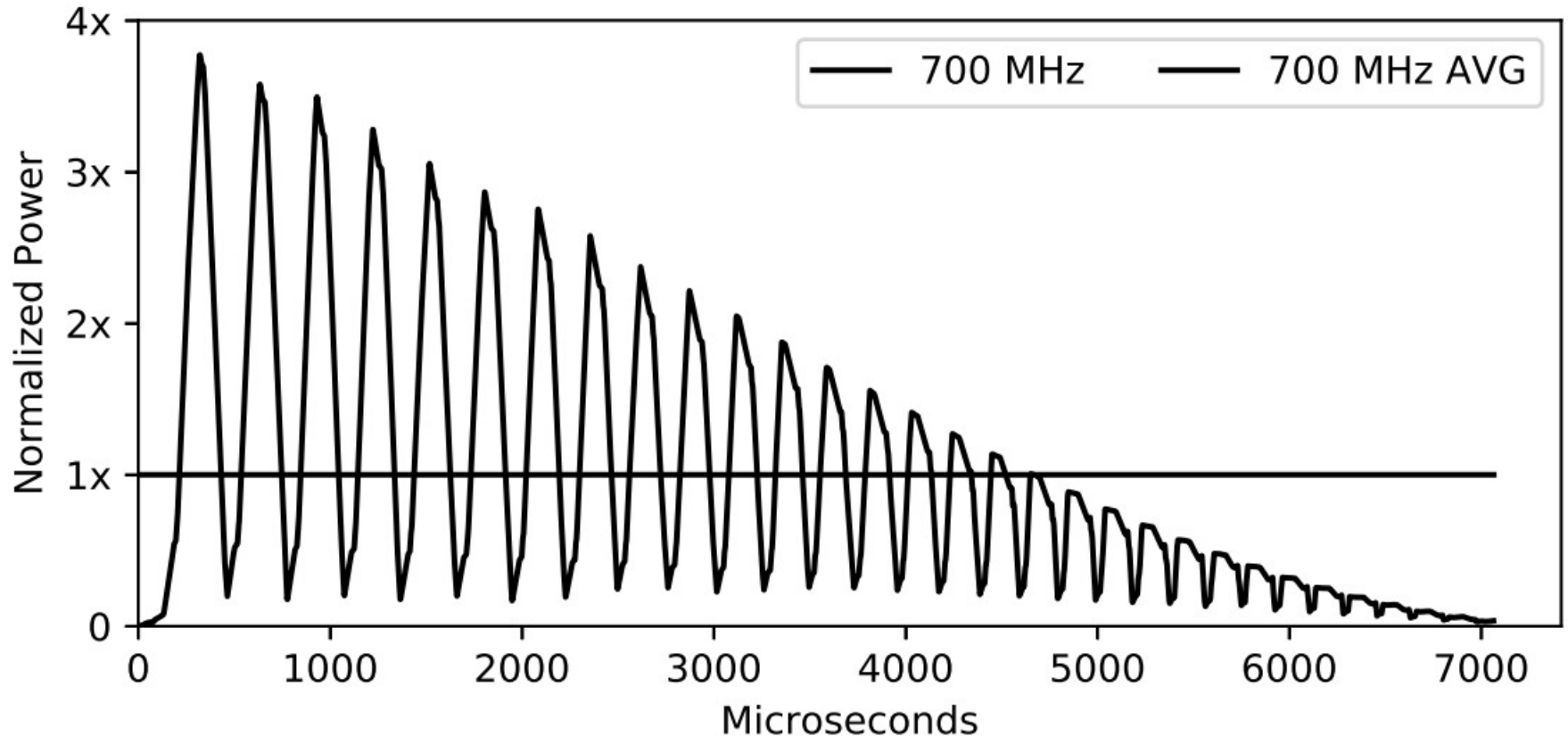
- New problem: multiple dies share single set of package pins
- Increasing need for power and IO (via package pins)



Background

- Power behavior is very **bursty**
 - short high-activity periods followed by longer low-activity periods
 - $P = CV^2 f$
- Keeping the power below the limit = Power Capping

Background



LU Decomposition power consumption at 700 MHz

Background

- Required power behavior detailed by the *power limit specification*
 - Acceptable power level (50 watts)
 - Acceptable time window (20 μ s)
- Time windows dictated by which component will fail first
 - \sim 20 μ s for package pins or \sim 1 ms for an external voltage regulator (VR)

Current Approaches

- Centralized controllers:
 - RAPL/TurboBoost [Intel]
- Software-based control:
 - Isci et al [MICRO39], Joao et al [SIGARCH '13], Lefurgy et al [Cluster Computing v11 '08], SW response time is $>50 \mu\text{s}$ (too slow!)
- Heterogeneous systems:
 - Harmonia [SIGARCH '15], Komoda et al [ICCD '13], DynaCo [SC '13], Pupil [ASPLOS '16], Co-Cap [SAC '16]
 - Focused on saving energy or Software-based (too slow!)

Background

- SoCs have many components (processor cores, GPU compute units, etc) in a single package
 - Nvidia Volta V100 has 5120 CUDA cores (in 80 streaming multiprocessors)
- Large scale means difficult communication for centralized controllers
 - Lots of global wires or use bus

Background

- SoCs have many *different* components
 - CPUs, GPUs, accelerators, FPGAs, etc.
 - Hard to create a single central algorithm for all combinations

Problem Definition

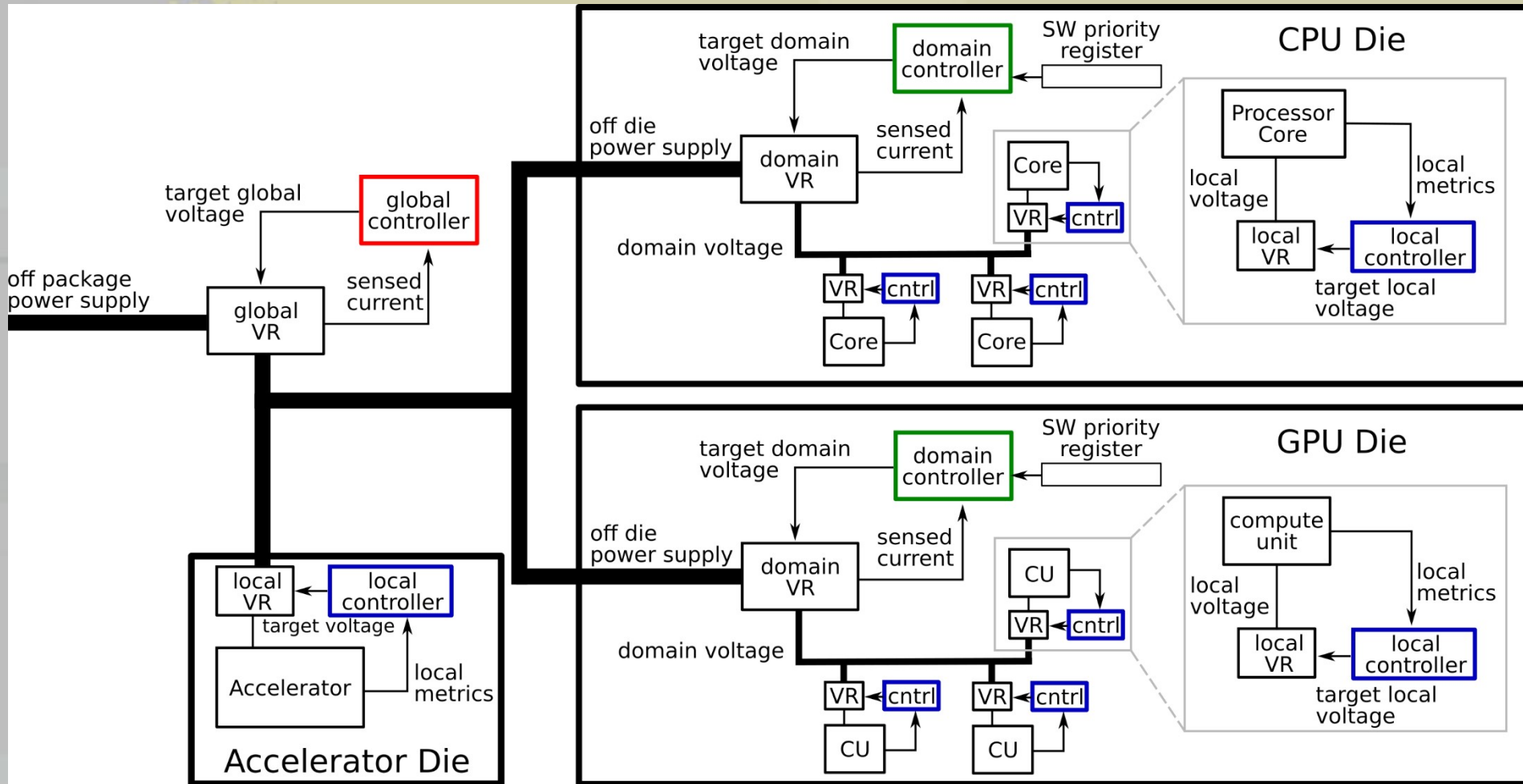
- How can we take advantage of bursty power consumption to improve performance on average?
 - Bring average and peak power closer together
 - Steer the power to where it is needed
- How can we ensure that the approach will scale as 2.5D systems get larger and larger, and support heterogeneity?
 - Cannot have separate communication for each unit
 - Enable swappable support for different architectures

**HCAPP: Heterogeneous Constant Average
Power Processing**

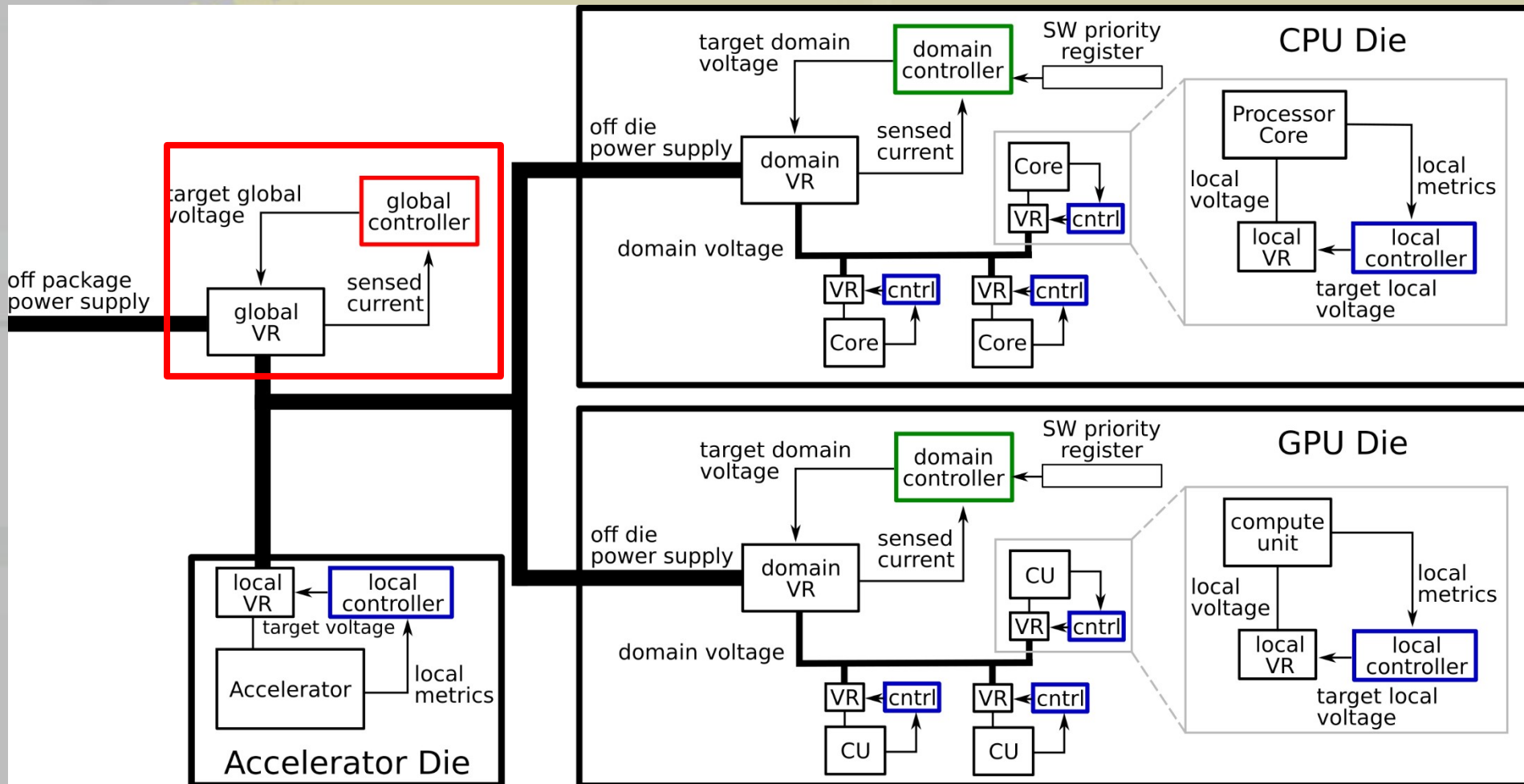
Design Requirements

Requirement	Reason
Scalable to many components	Needs to work for larger and larger designs in the future
Support multiple architectures	Needs to enable multiple different configurations of dies in the 2.5D system
Maintains power cap	Power limit must be upheld for system viability
Uses extra average power	Use as much power as possible since it is already provisioned
Fast reaction time	Power cap must be maintained over short time step (~20 μ s)

HCAPP Design

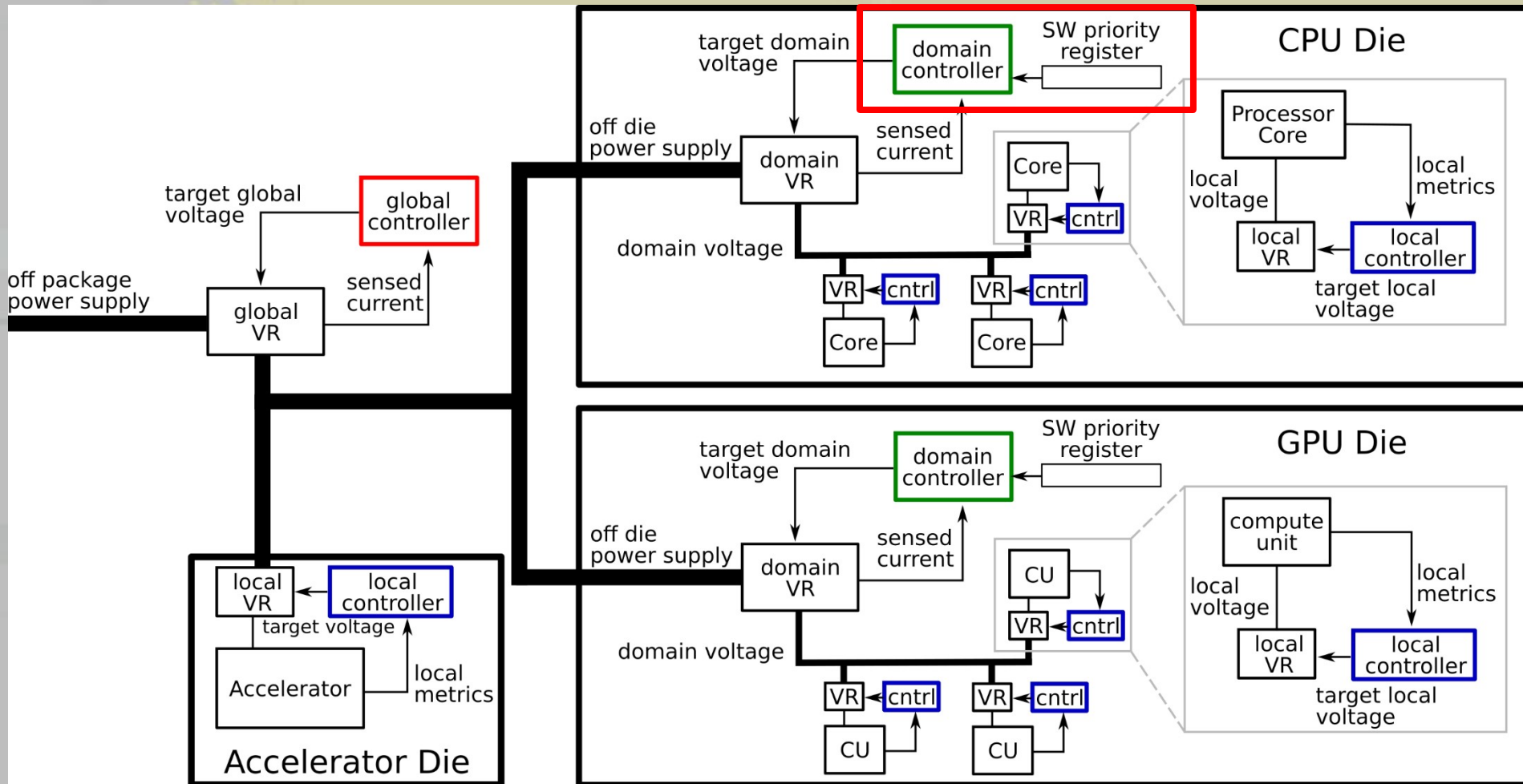


Design



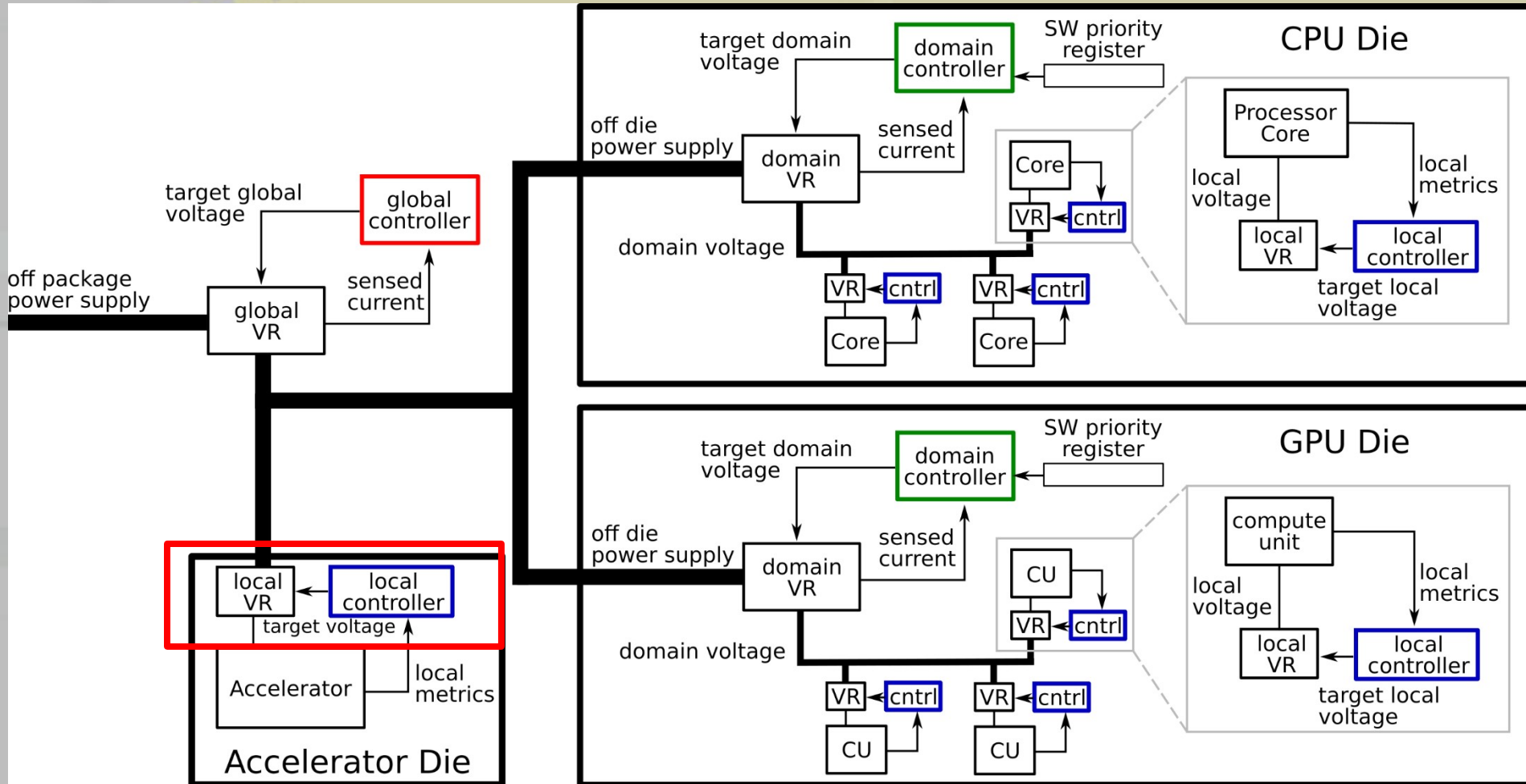
Global controller: maintain power cap through voltage ctrl

Design



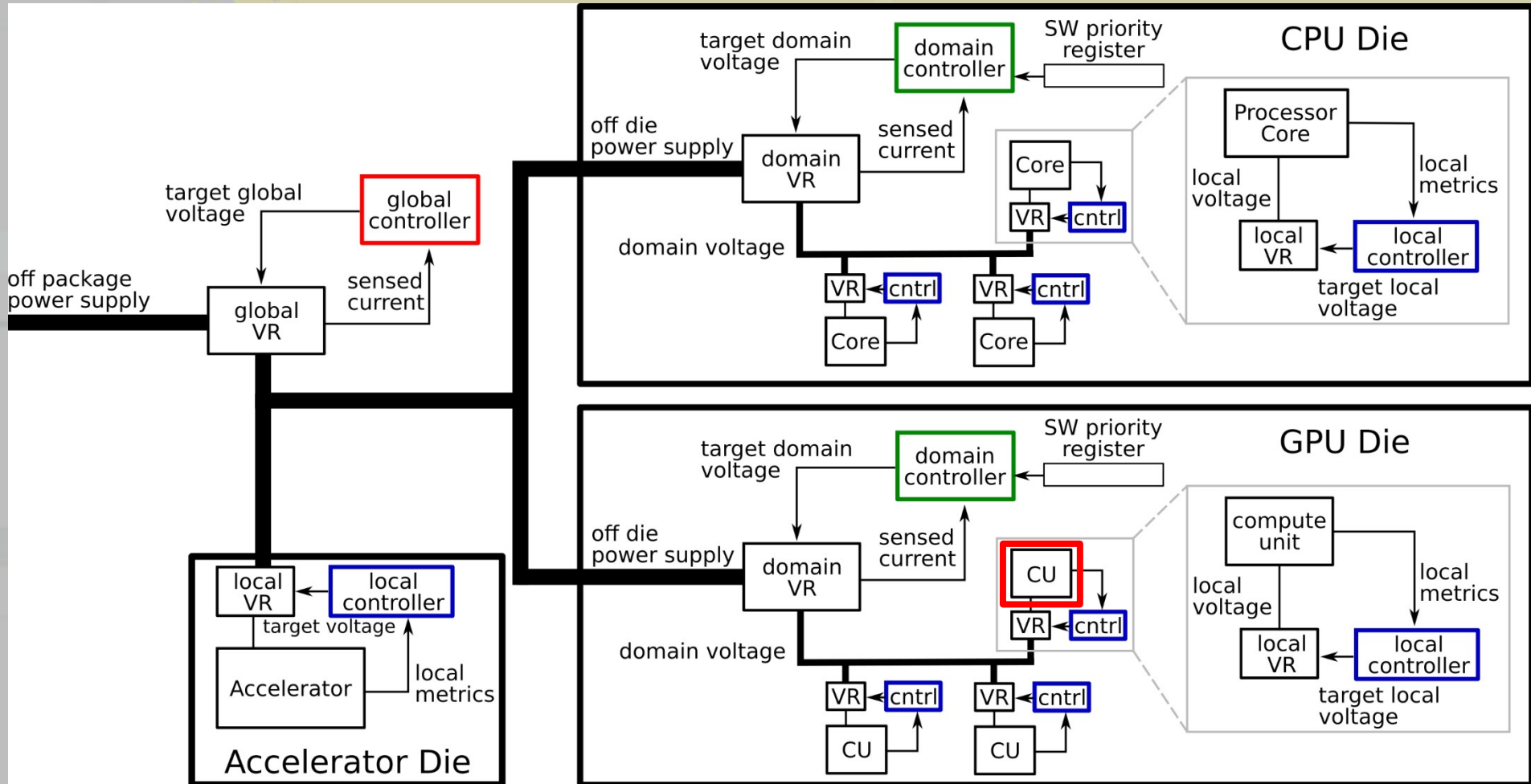
Domain controller: Scale voltage for die, SW interface

Design



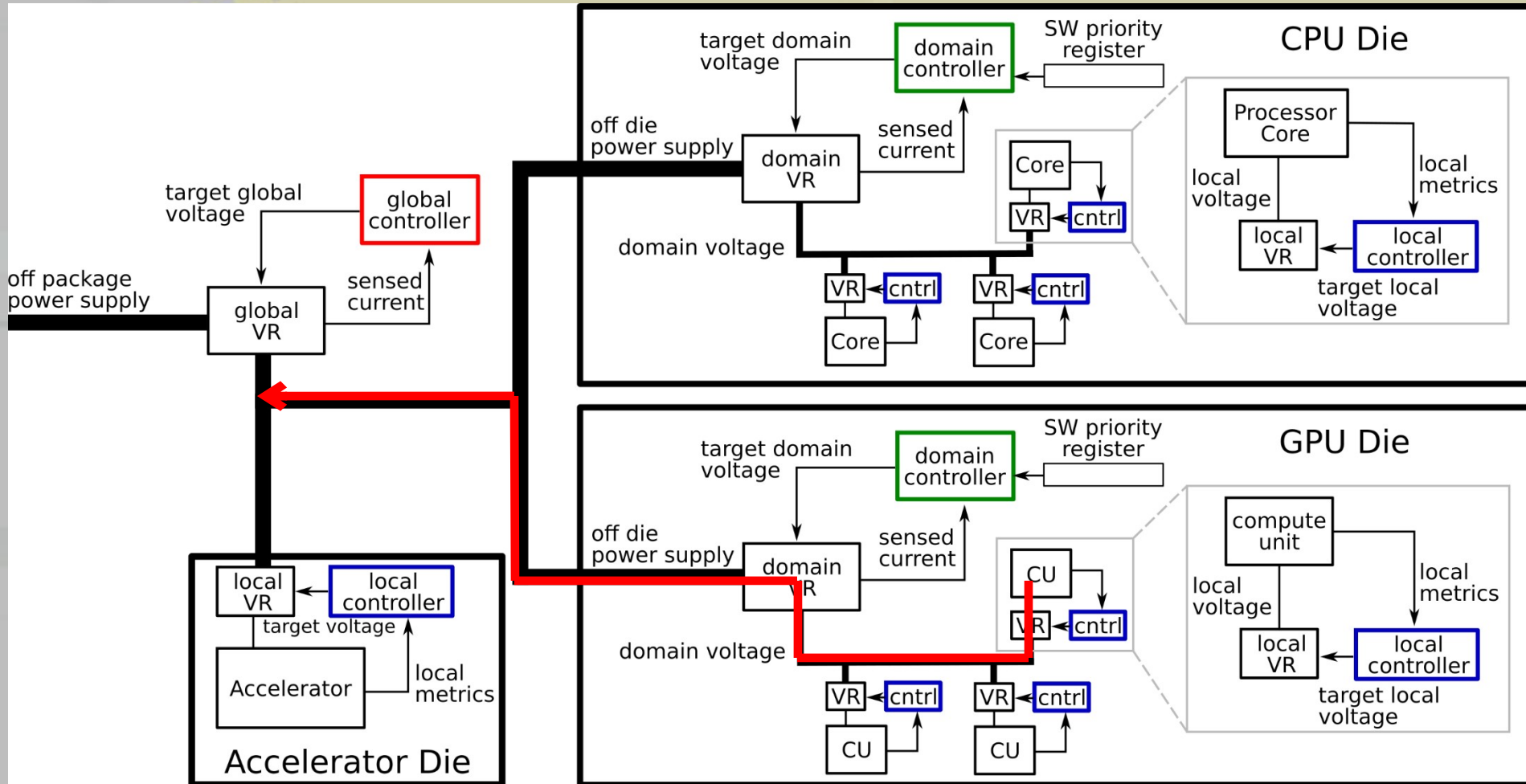
Local controller: Use local metric to improve efficiency

Design



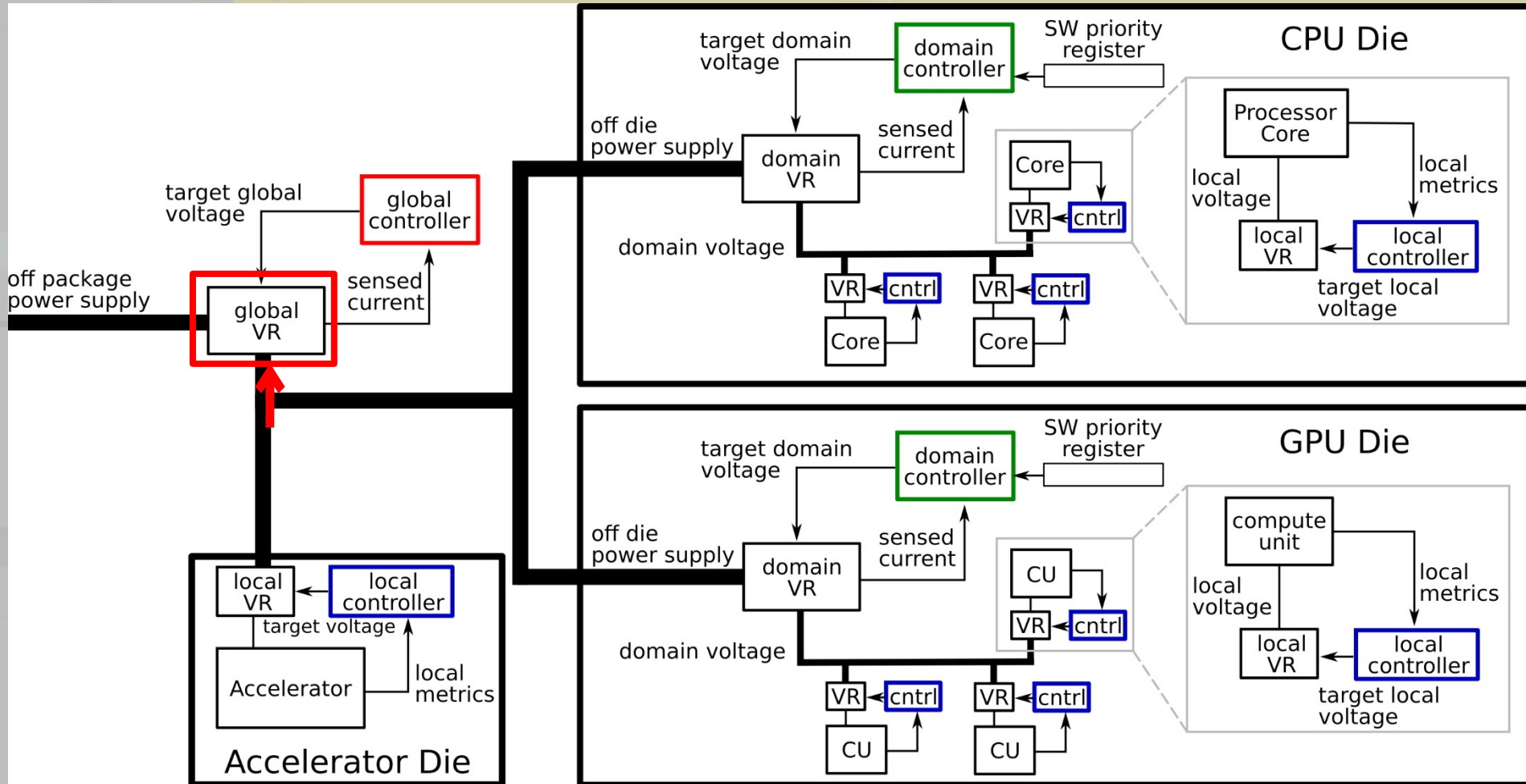
Step 1: Activity change in a component

Design



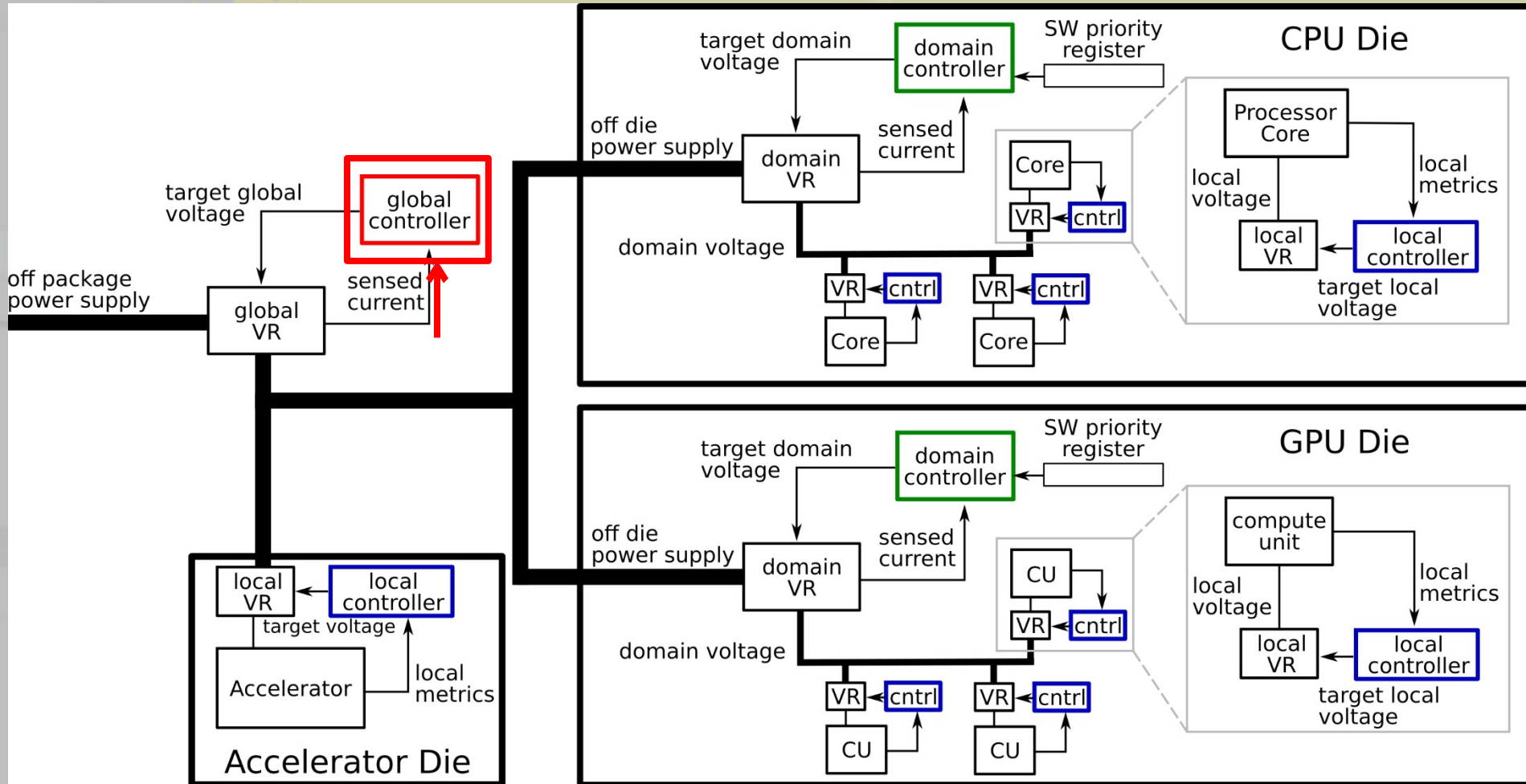
Step 2: Power draw propagates back to global VR

Design



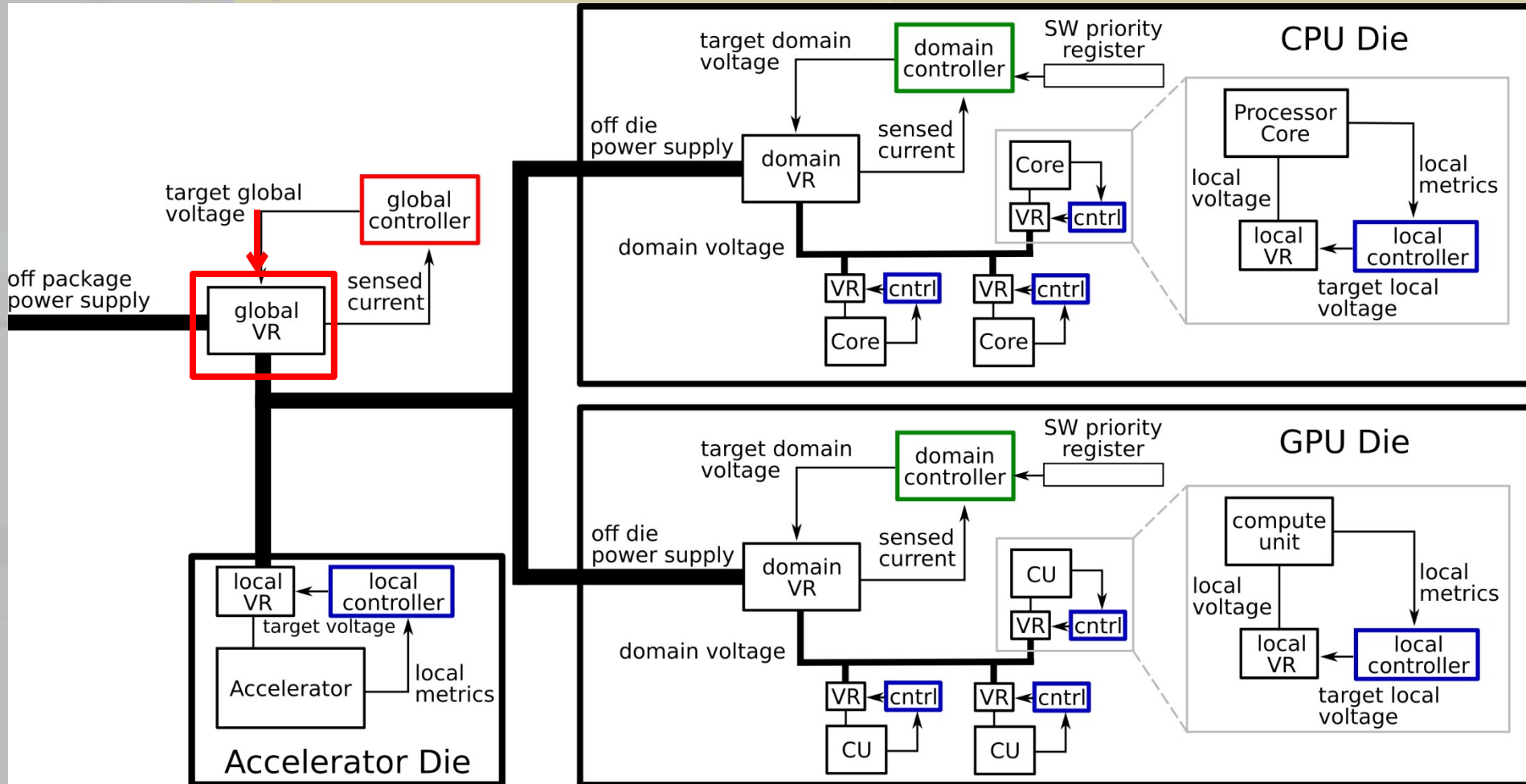
Step 3: Global VR senses new current draw

Design



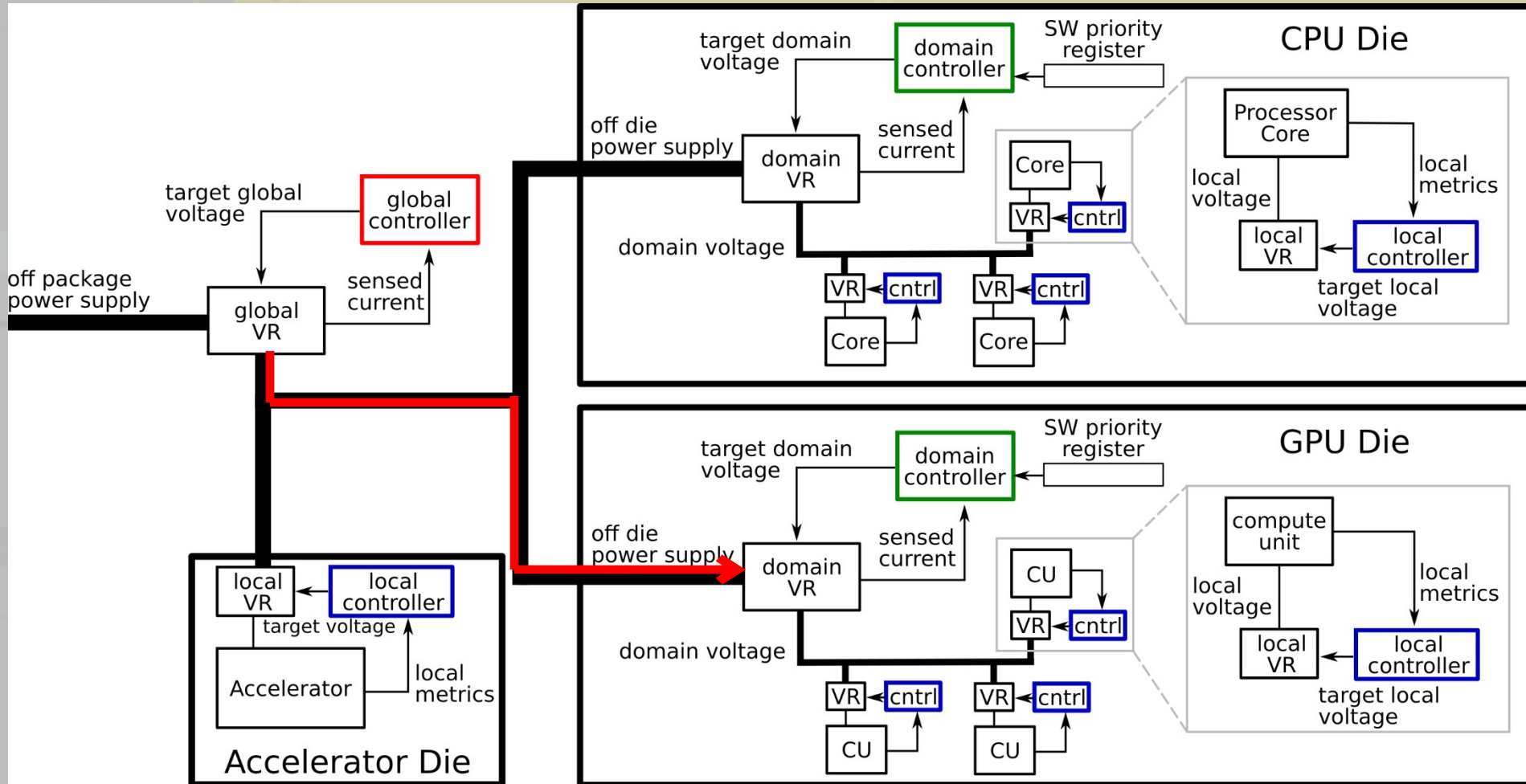
Step 4: Global controller calculates next voltage (PID)

Design



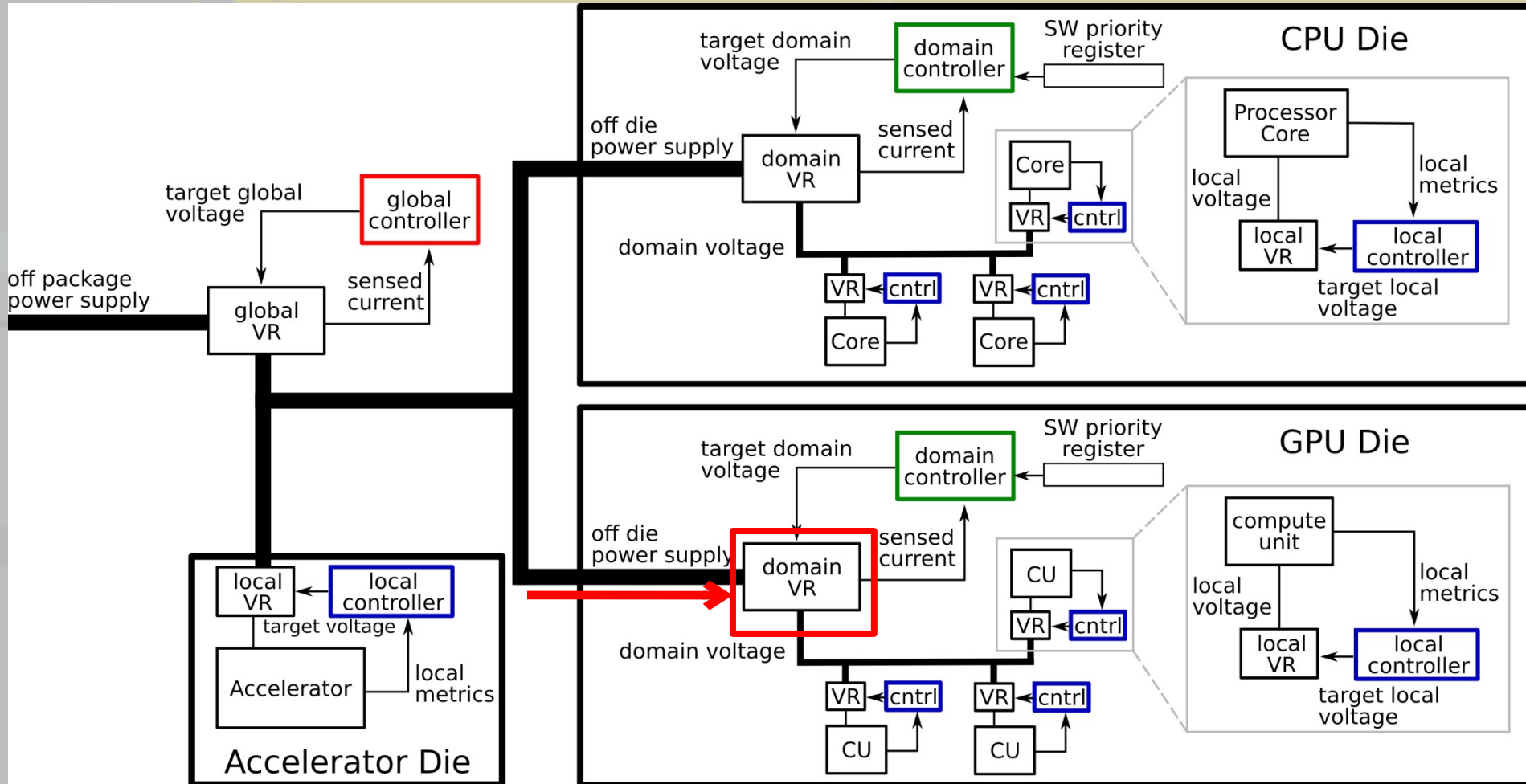
Step 5: Global VR assigns new global voltage

Design



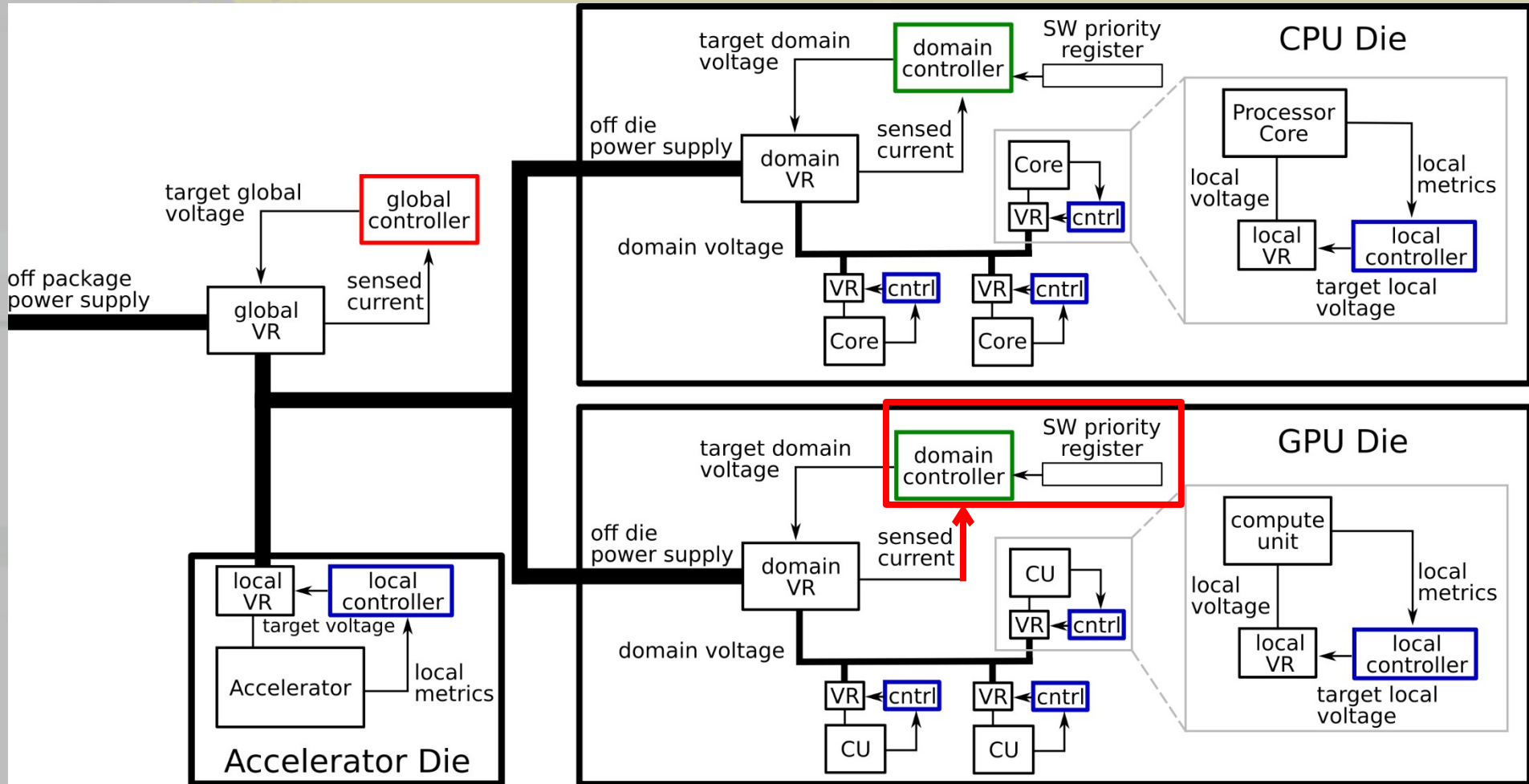
Step 6: Global voltage propagates to domain VR

Design



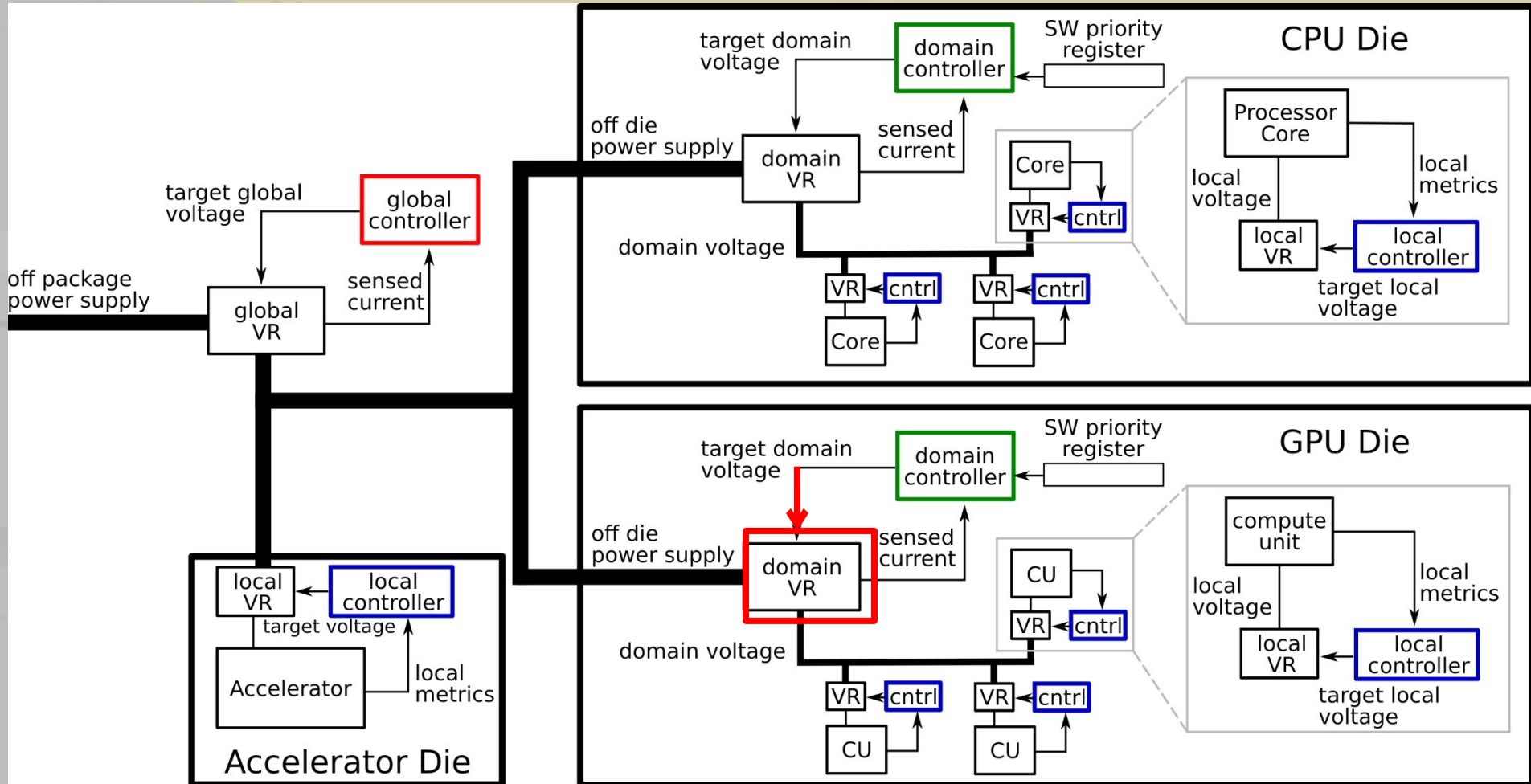
Step 7: Domain VR senses new global voltage and current

Design



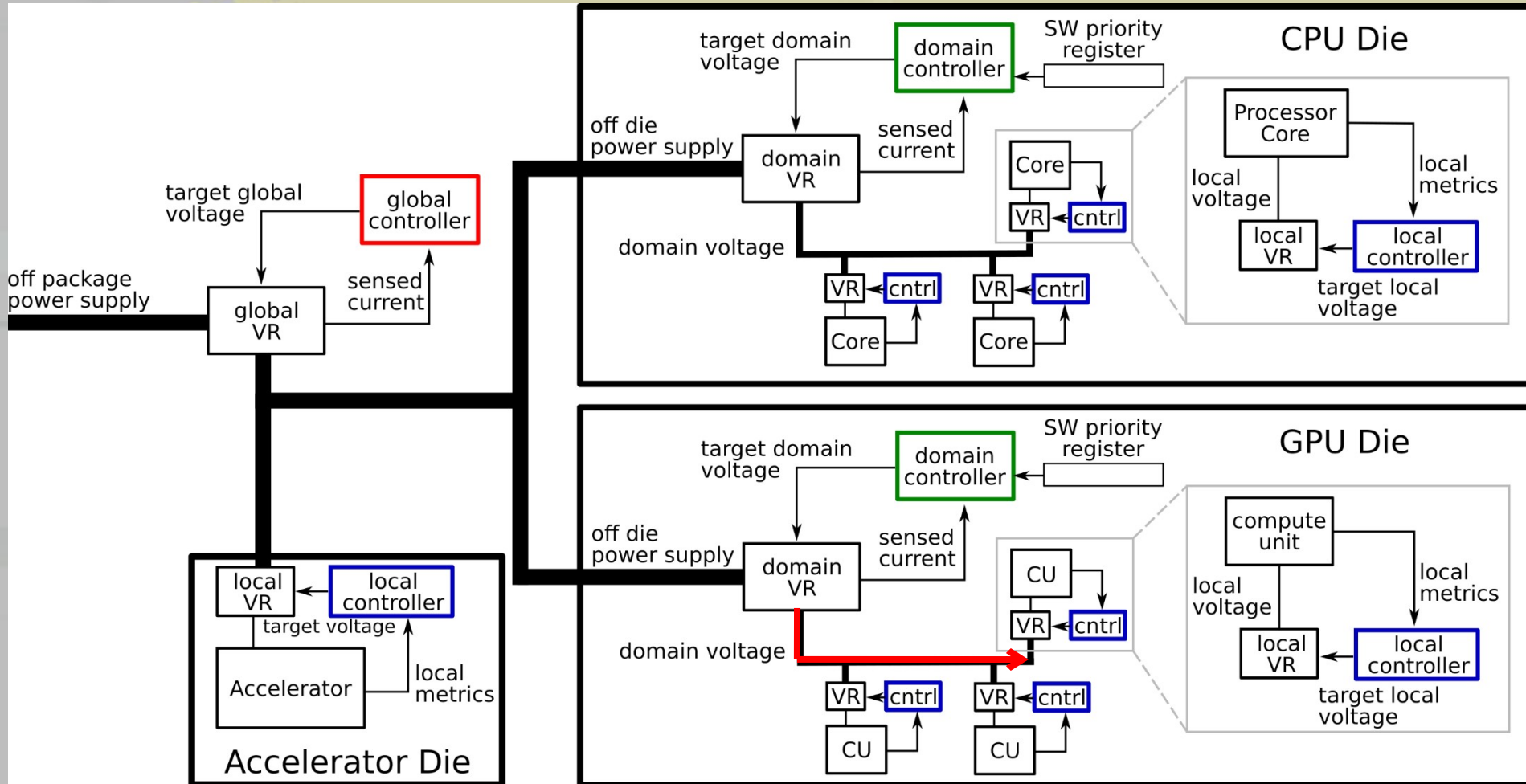
Step 8: Domain ctrl calculates new domain voltage

Design



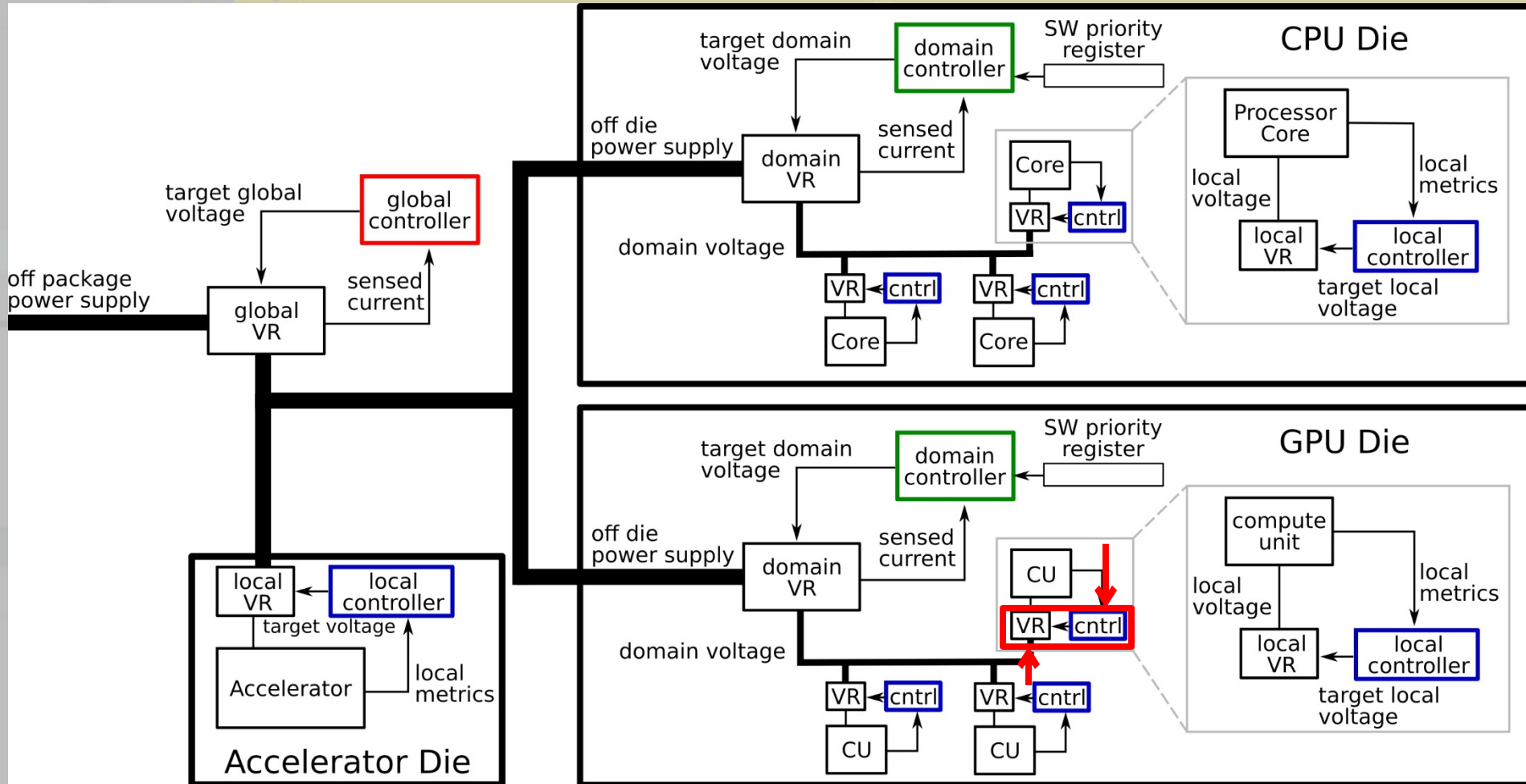
Step 9: Domain VR applies new domain voltage

Design



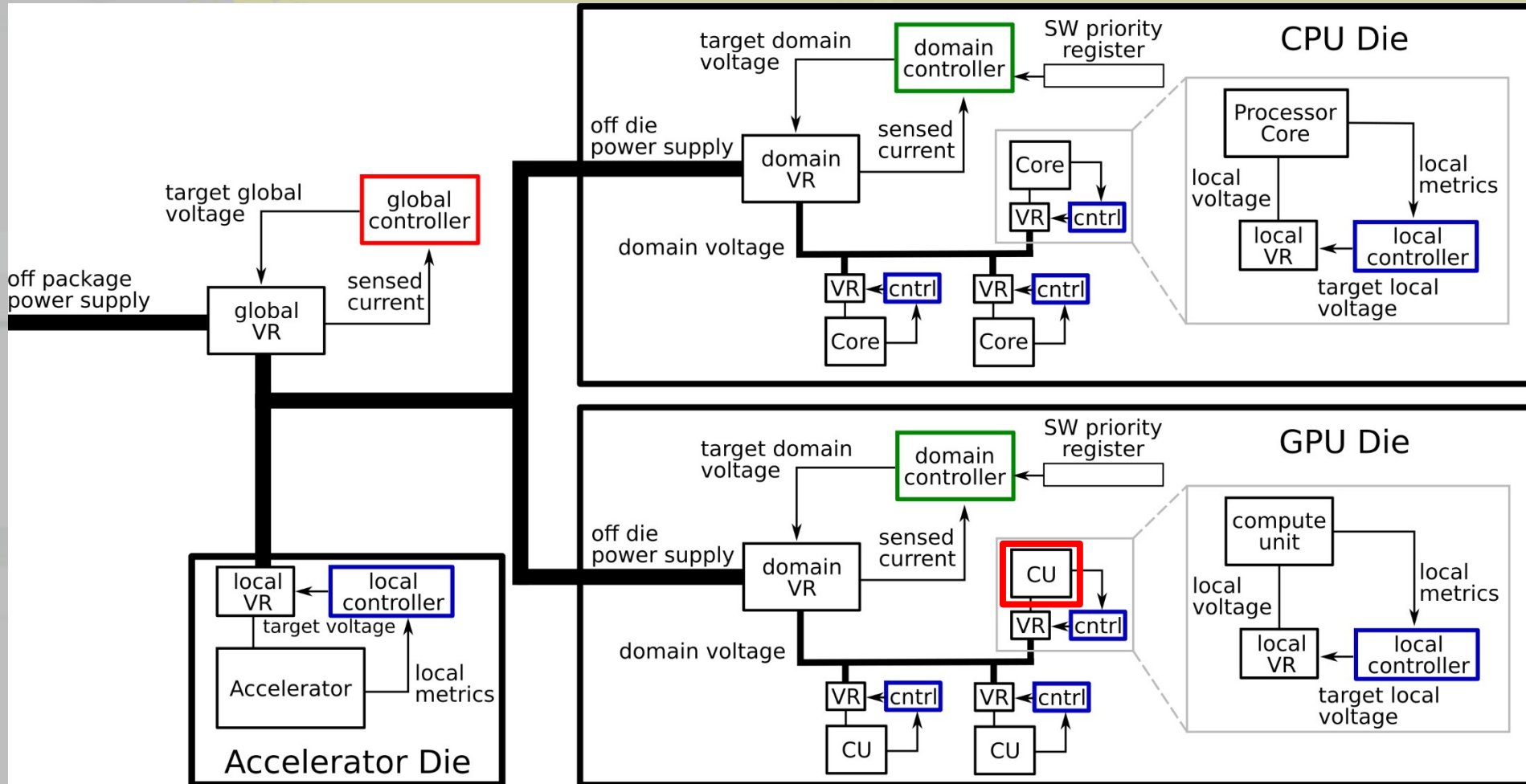
Step 10: New domain voltage propagates to component

Design



Step 11: New local voltage determined from domain voltage and local controller

Design



Step 12: Component uses new local voltage and frequency

Design

- PID Tuning
 - Done manually with general methodology
 - First, **increase proportional** (K_P)
 - Then, **increase integral** until steady state error is acceptable (K_I)
 - Derivative component not used in this controller

$$V_{ERROR} = \sqrt[3]{P_{SPEC} - P_{NOW}}$$

$$V_{NEXT} = V_{FeedForward} + K_P * V_{ERROR} + K_I * \int V_{ERROR} dt + K_D * \frac{dV_{ERROR}}{dt}$$

Component-Specific Design

- Local controllers designed to take advantage of local architecture metrics (such as IPC or warp occupancy)
- Scale voltage locally based on metrics to push power to components that need the power
- Used high IPC (CPU) and dynamic warp controllers (GPU) from CAPP and GPU-CAPP work

Global Controller Speed

Component	Response time (ns)
Voltage Regulator	$(36-226) \times 2 = 72-452$
Sensing Circuitry	50-60
Controller	10-30
Power Supply Network	$(3-15) \times 5 = 15-75$
Total	147-617
HCAPP Cycle Time	1000

Design Summary

Requirement	HCAPP Related Feature	Status
Scalable to many components	Decentralized control through power network	PASS
Support multiple architectures	Architecture-specific domain controller and local controller logic	PASS
Maintains power cap	PID power control tuned to ensure cap	PASS
Uses extra average power	PID power control increases voltage when power is below cap	PASS
Fast reaction time	Speed of CAPP control is 1 μ s	PASS

Experimental Setup (System)

- System was defined as:
 - 1 CPU
 - 1 GPU
 - 1 SHA Accelerator
- Focused on execution time of one benchmark run on each starting at the same time
 - Combinations chosen based on benchmark characteristics

Experimental Setup (Models)

- CPU modeled using Sniper simulator with McPAT power model
- GPU modeled using GPUSim with GPUWattch
- Accelerator modeled as SHA Accelerator [Suresh et al, ESSCIRC'18]

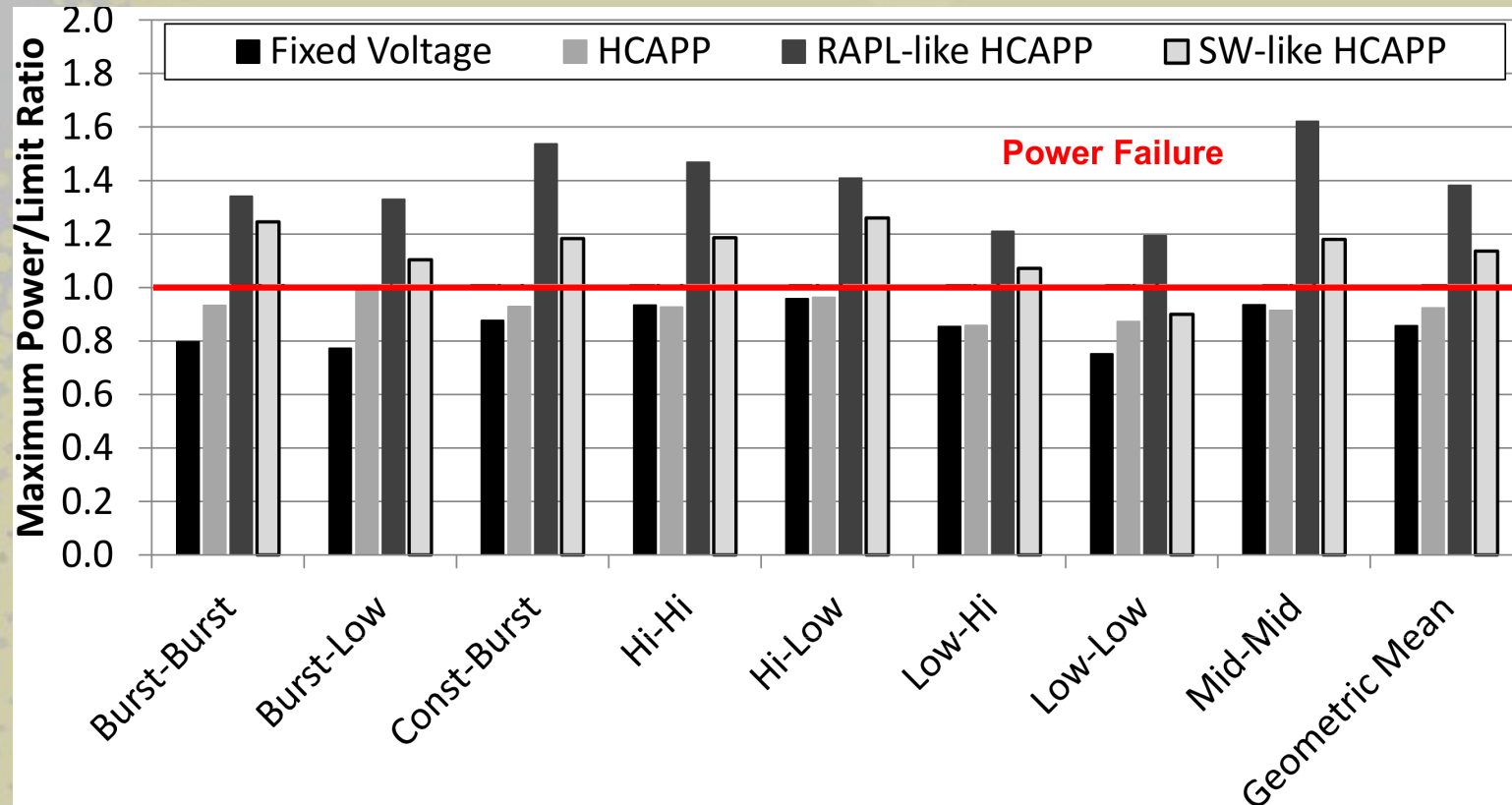
Experimental Setup (Benchmarks)

- CPU: PARSEC benchmark subset
- GPU: Rodinia benchmark subset
- SHA Accelerator
 - Analytical model with fixed amount of input work
- Benchmarks selected to create combinations of interesting power behaviors

Experimental Setup

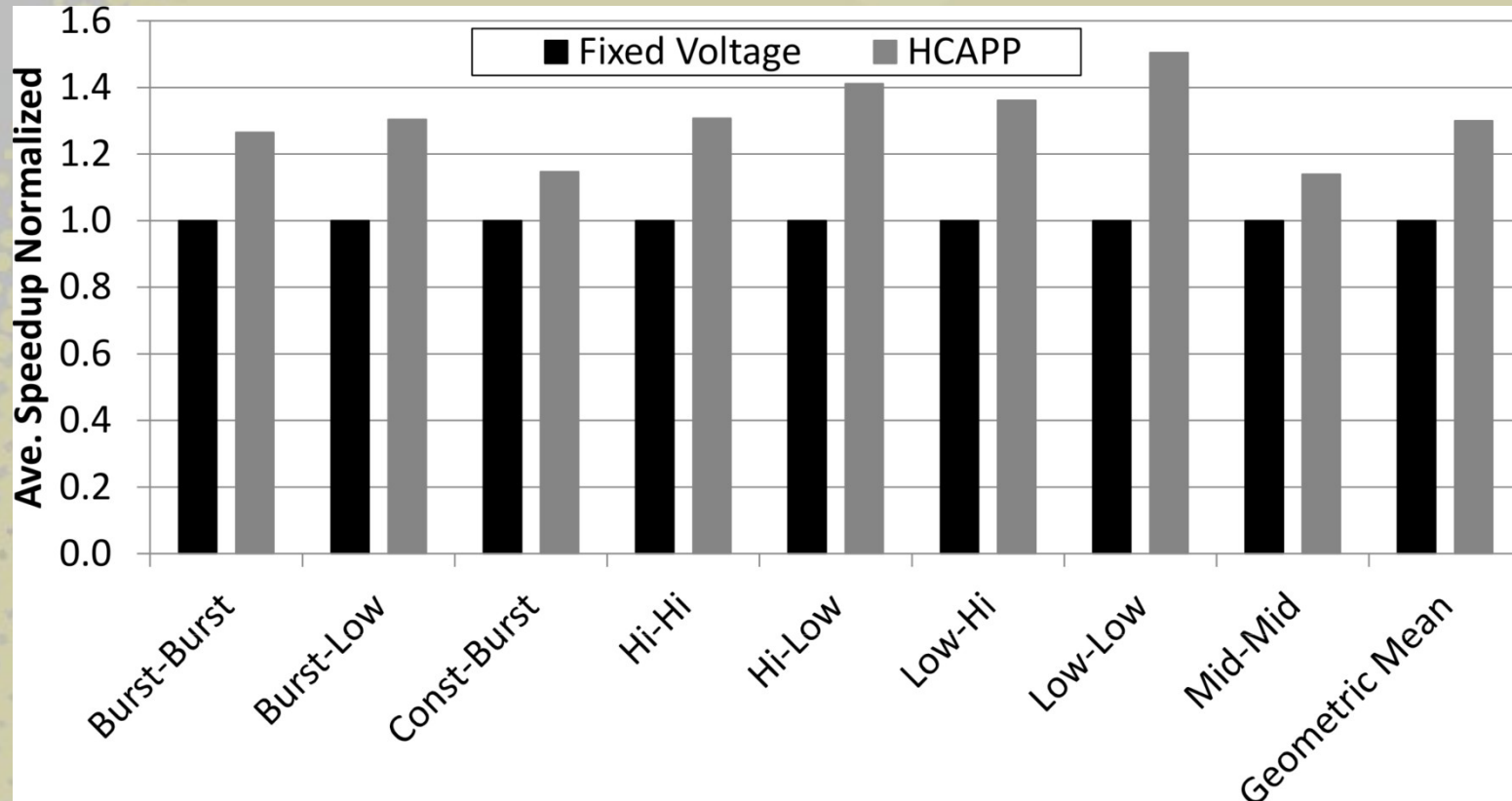
- Baseline: system with a single fixed global voltage and no local controllers
- Comparison systems:
 - HCAPP with $1\mu\text{s}$ control period
 - HCAPP with $100\mu\text{s}$ control period (RAPL-like equivalent)
 - HCAPP with 10ms control period (SW equivalent)
- Constraints: 100 W ($20\mu\text{s}$ window) and 100 W (1 ms window)

HCAPP Maximum Power



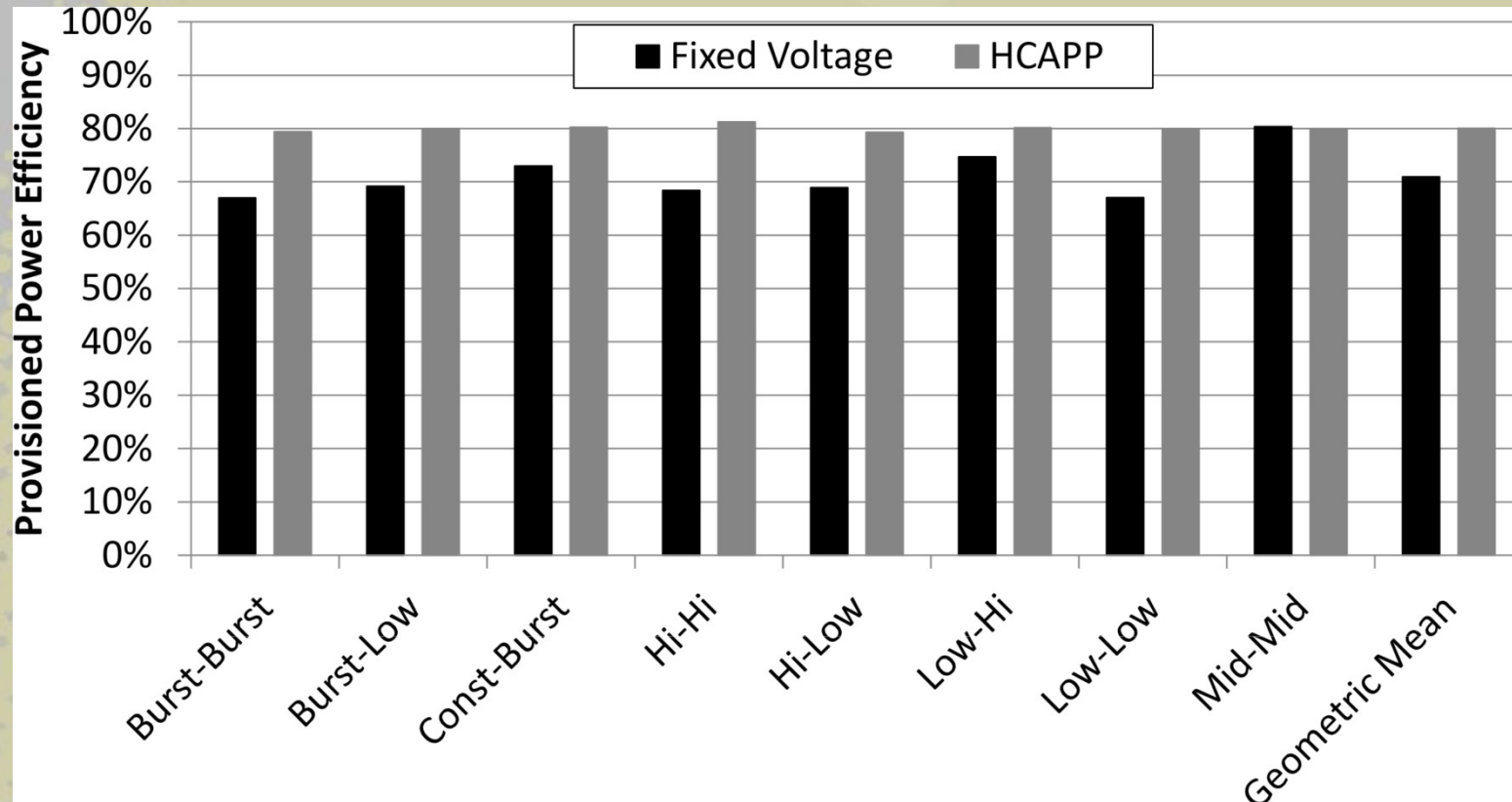
RAPL-like and SW-like greatly exceed maximum power
20 μ s time window

HCAPP Performance



Average speedup of **+21%**
20 μ s time window

HCAPP PPE

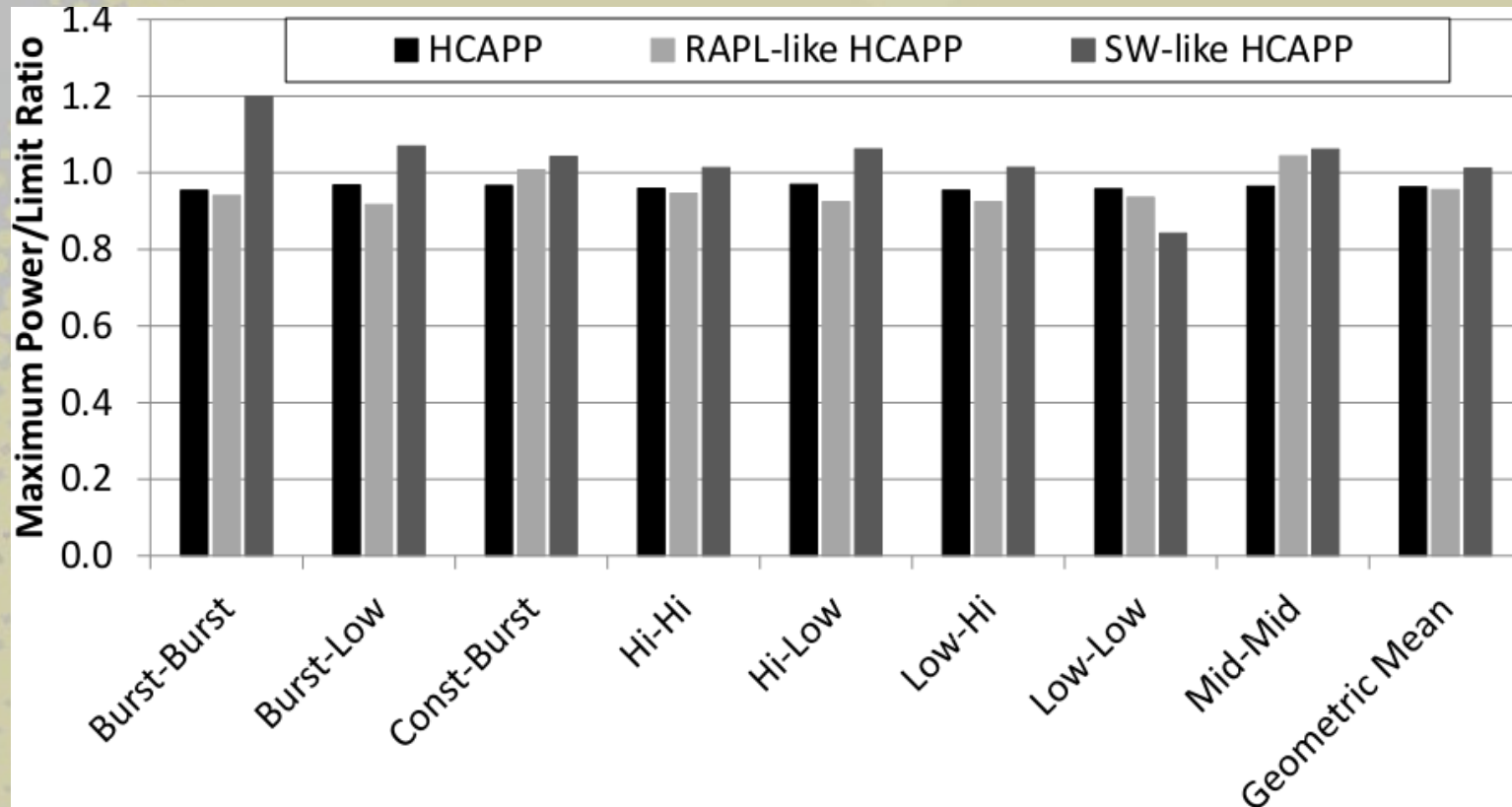


Provisioned Power Efficiency = Average Power / Power Limit

Average PPE improved from 69.1% to **79.3%**

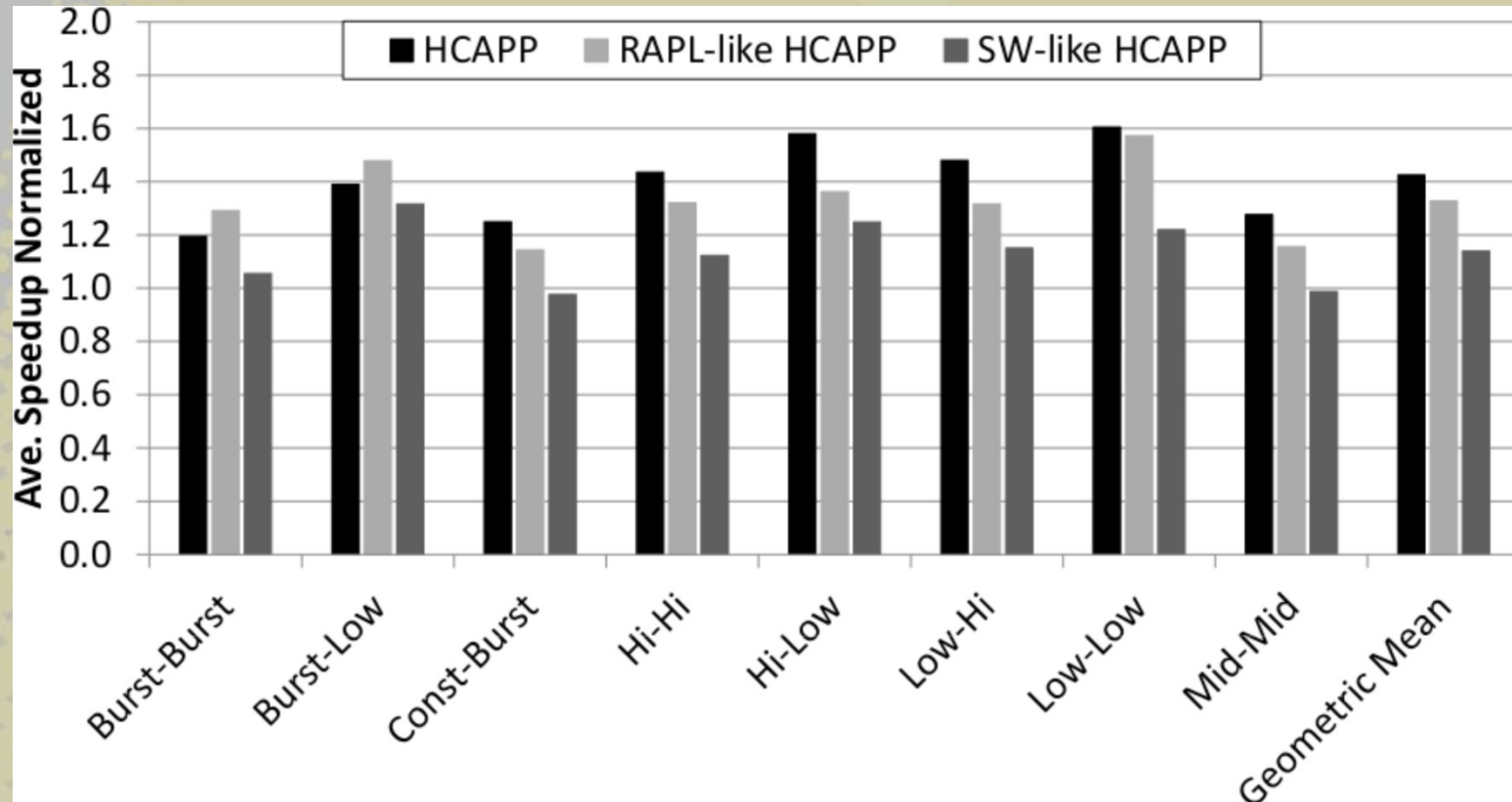
20 μ s time window

HCAPP Maximum Power



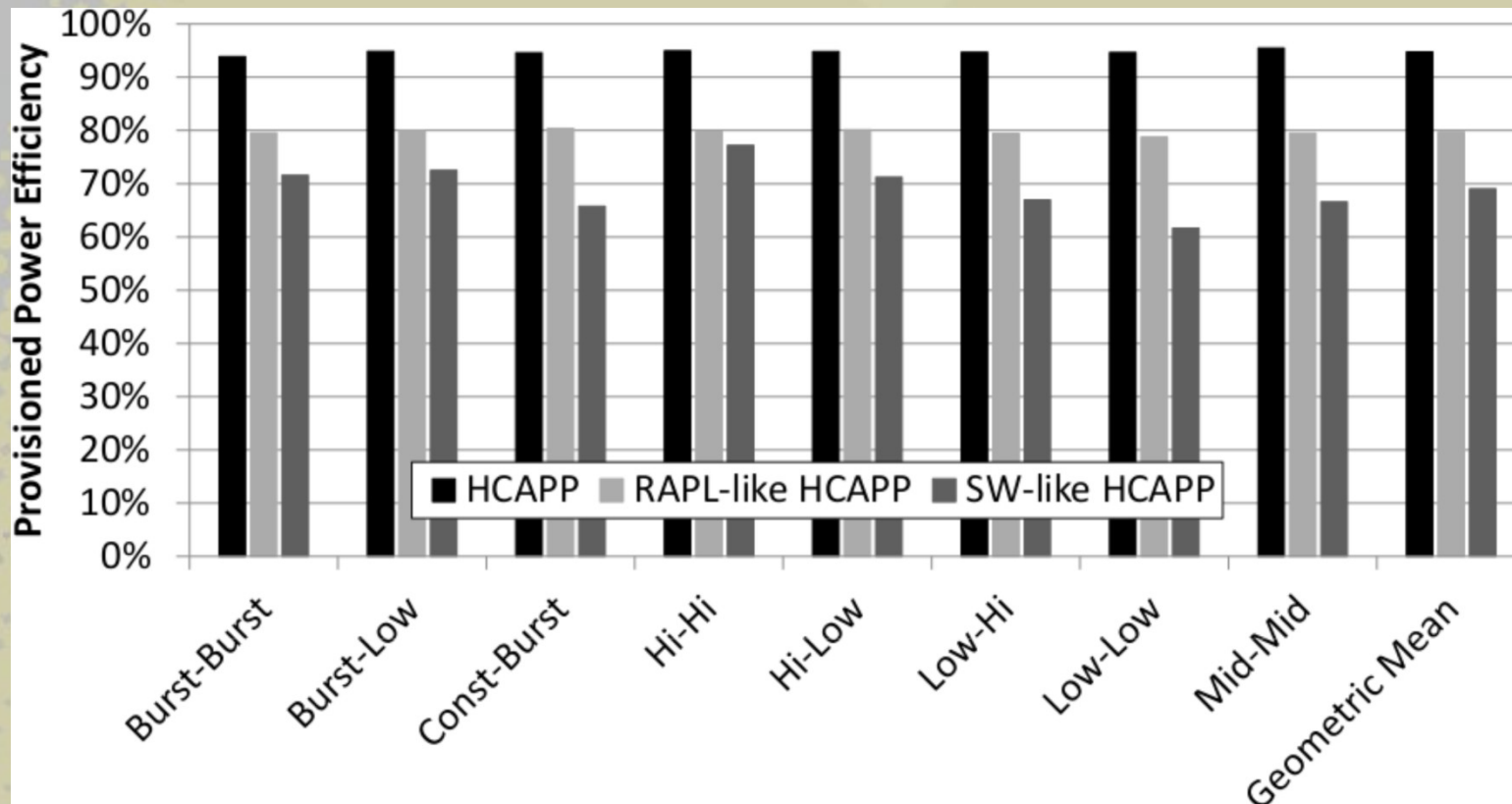
RAPL-like and SW-like still exceed limit, RAPL-like approaches viability
1 ms time window

HCAPP Performance



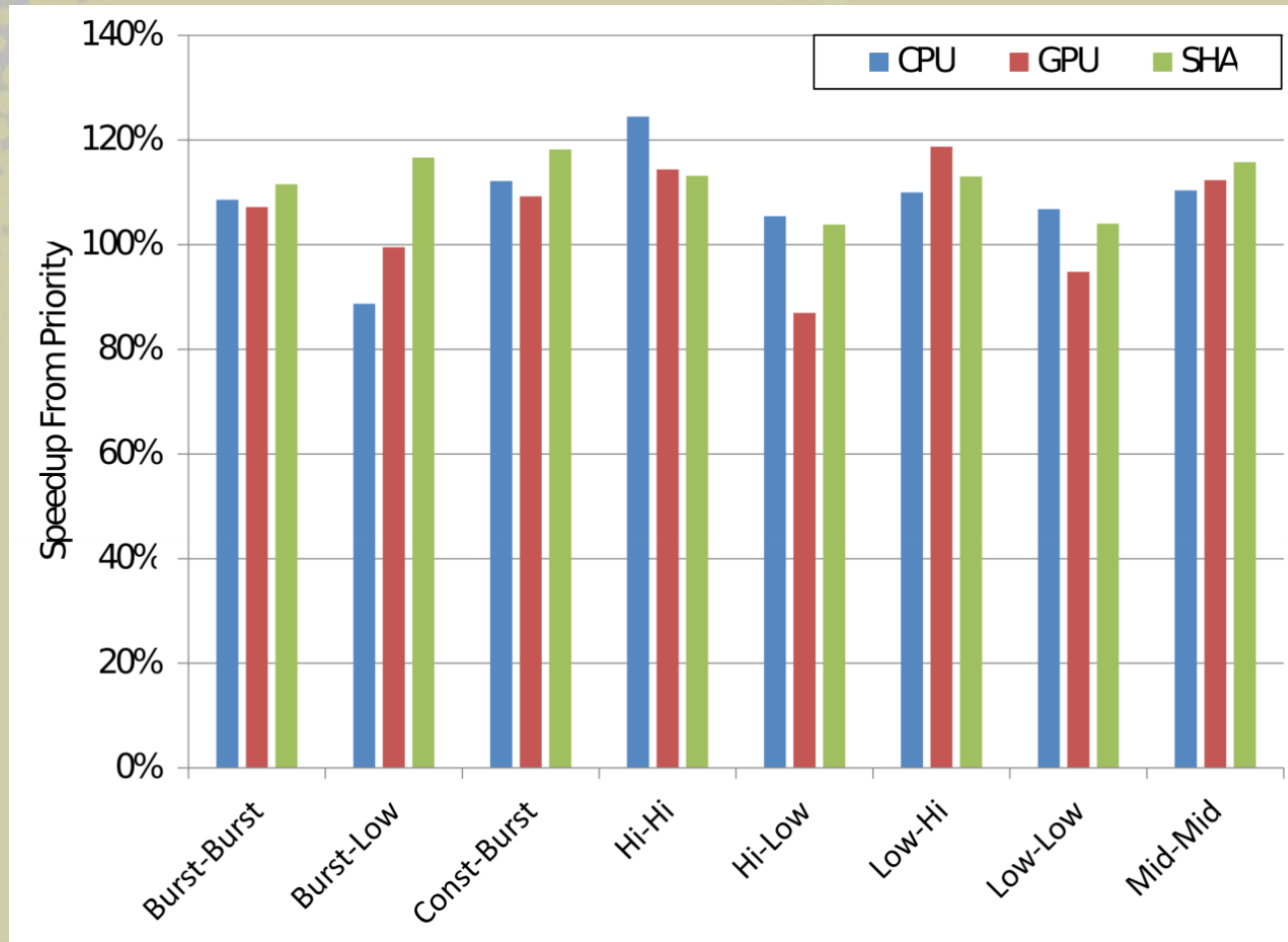
Average speedup of **+43%** (compared to 36% for RAPL)
1 ms time window

HCAPP PPE



Average PPE improved from 69.1% to **93.9%** (RAPL: 79.7%)
1 ms time window

HCAPP SW Interface



Simple SW prioritization results in average speedups of **+8.3%** (CPU), **+5.4%** (GPU), and **+12.0%** (SHA)

Final Thoughts

HCAPP is a power management architecture that can:

- Manage heterogeneous systems
- Scale with increasingly large systems in a single package
- Maximize performance under a power limit

Application	Pin power limit	VR power limit
Speedup	+21%	+43%
PPE	+10%	+35%

Thank you for watching

