big.VLITTLE: On-Demand Data-Parallel Acceleration for Mobile Systems on Chip

Tuan Ta*, Khalid Al-Hawaj, Nick Cebry, Yanghui Ou,
Eric Hall, Courtney Golden, Christopher Batten

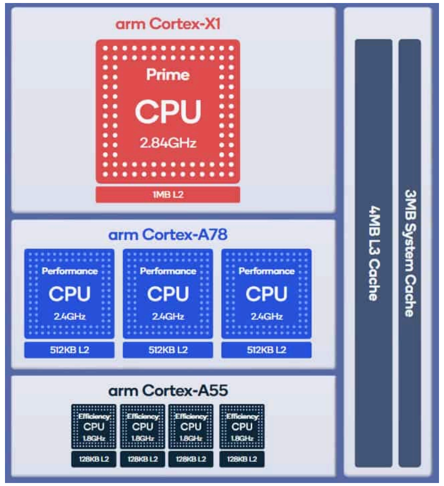
MICRO-55 – Oct 3rd, 2022

Cornell University
Computer Systems Laboratory

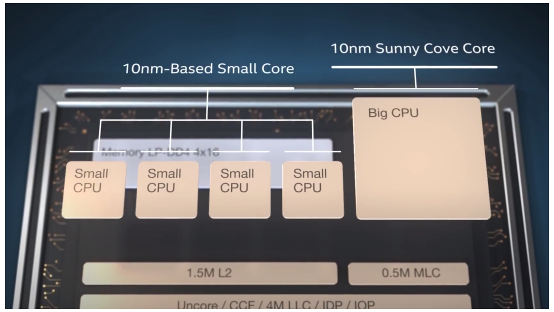
CSL

big.LITTLE architectures dominating mobile SoCs

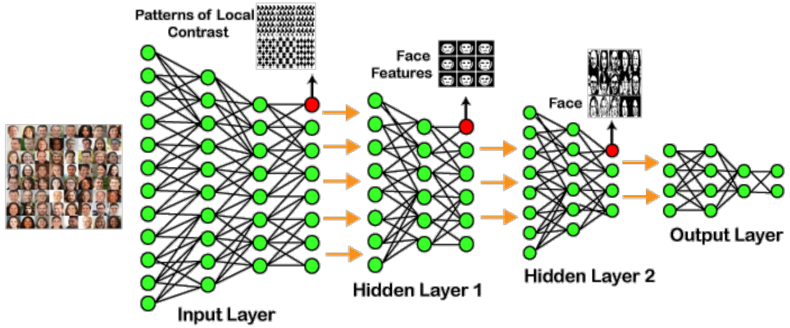
Emerging data-parallel workloads moving to edge devices



(source: Qualcomm)

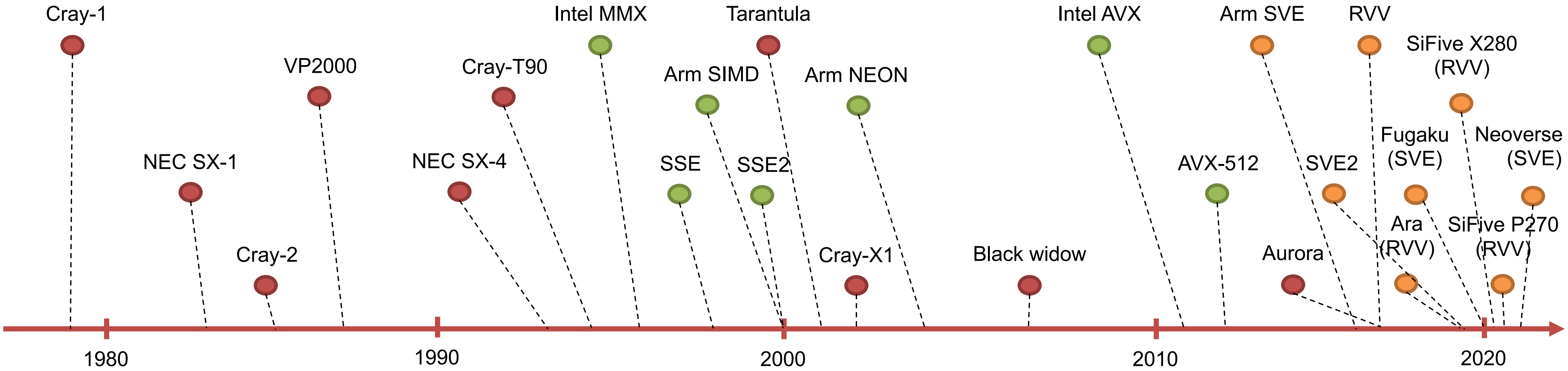


(source: Intel)



How to efficiently accelerate data-parallel workloads on tight area and power budget in big.LITTLE SoCs?

The Resurgence of Vector Architectures



Traditional Long-Vector Archs

- Scalable & long vector length
- Large & decoupled vector engines
- High performance

Traditional Packed-SIMD Archs

- Fixed & short vector length
- Small & integrated vector units
- Low area overhead

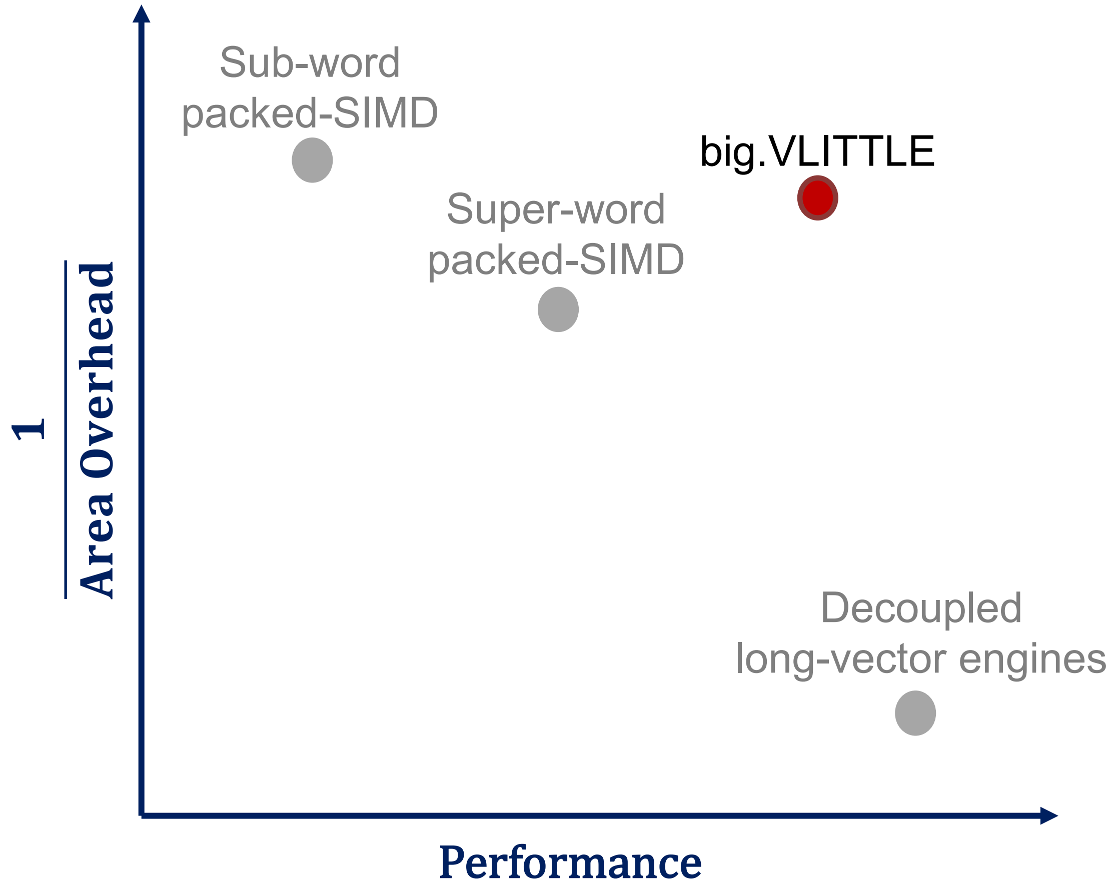
Next-gen Vector Archs

- Scalable vector length
- Either high-performance or low-area-overhead designs

big.VLITTLE Design Goals

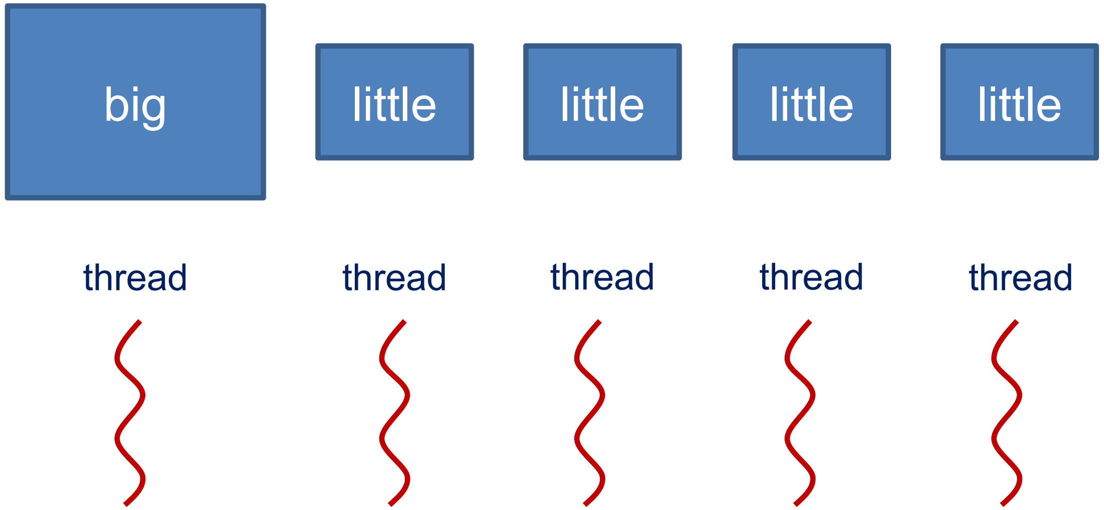
CSL

- ✓ **Next-gen vector architectures**
- ✓ **Low area overhead as integrated packed-SIMD units**
- ✓ **High performance as decoupled long-vector engines**
- ✓ **No performance overhead for task-parallel workloads**



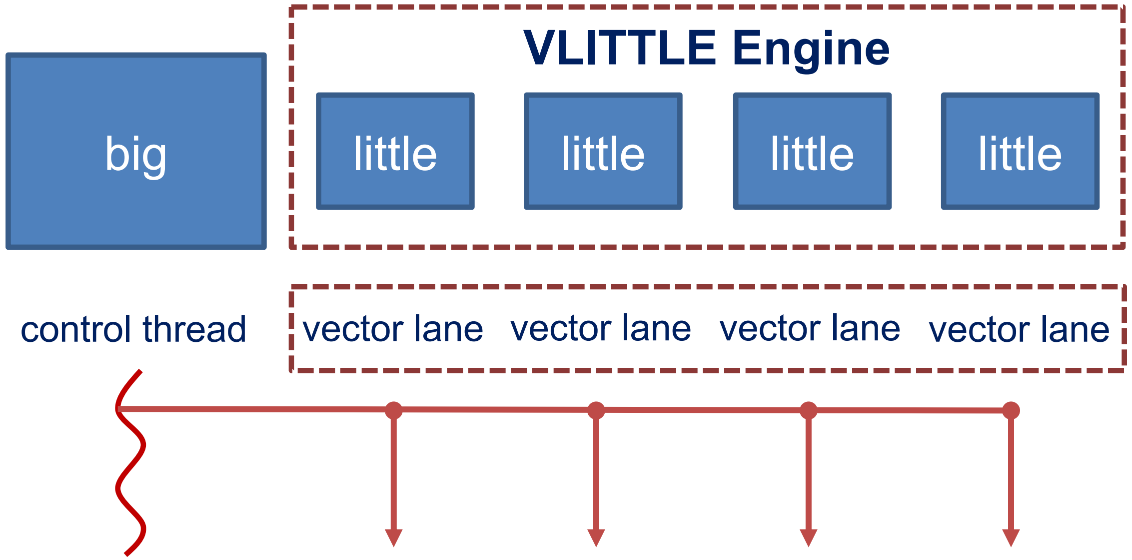
On-Demand Vector Execution

Thread Mode



- Non-vectorized task-parallel workloads
- Exploit thread-level parallelism (TLP)
- Cores execute independently

Vector Mode



- Vectorized data-parallel workloads
- Exploit data-level parallelism (DLP)
- Little cores execute as vector lanes
- Big core handles scalar control flow

big.VLITTLE Micro-Architecture – Compute

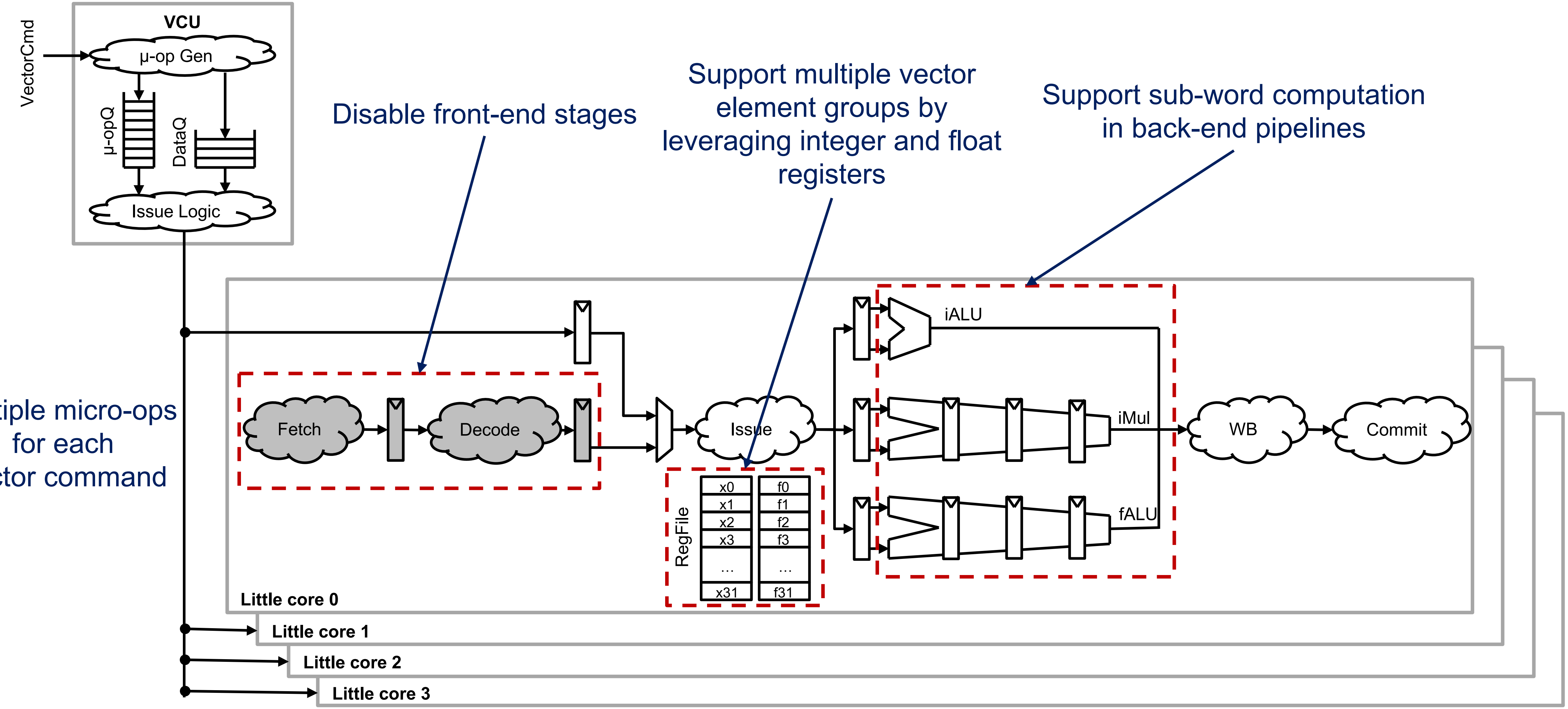
CSL

Disable front-end stages

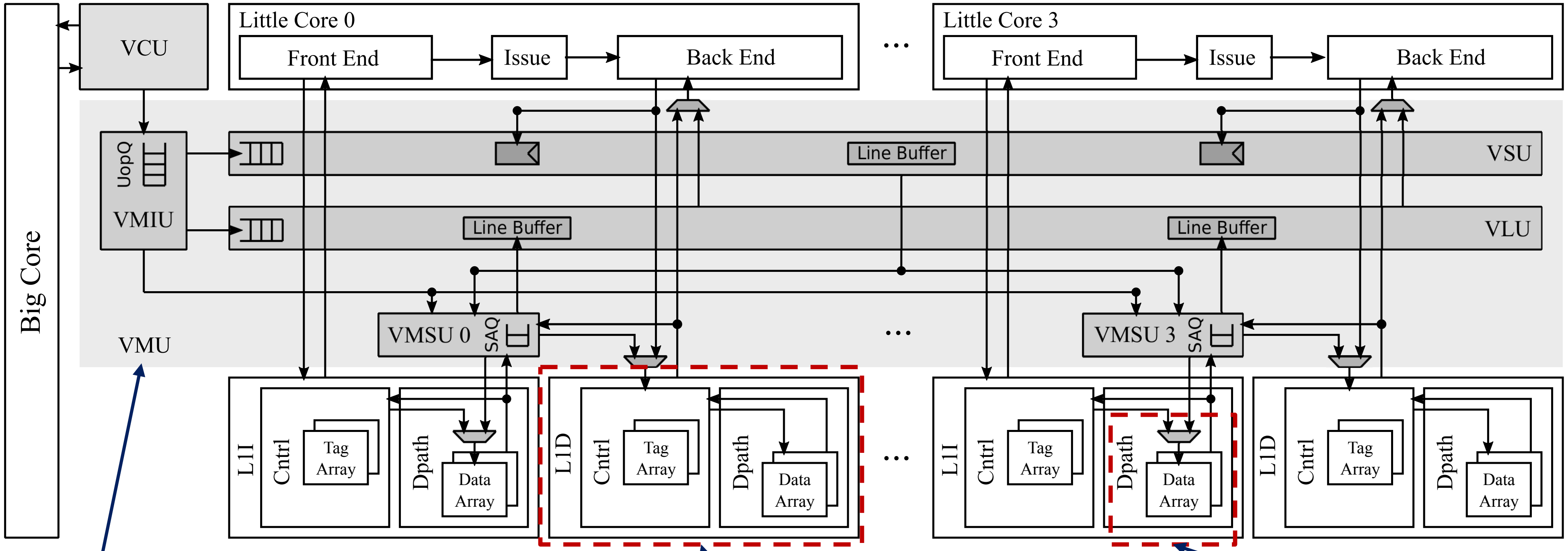
Support multiple vector element groups by leveraging integer and float registers

Support sub-word computation in back-end pipelines

Multiple micro-ops for each vector command



big.VLITTLE Micro-Architecture – Memory



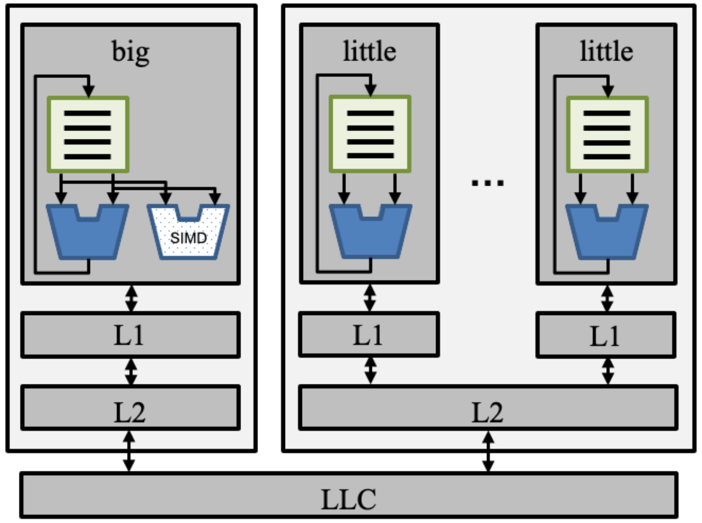
VMU – Decouple vector memory accesses from vector computation

Reconfigure L1D caches as one multi-bank L1 cache for VLITTLE

Reconfigure L1I data arrays as data buffers in VMU

Baseline & big.VLITTLE Systems

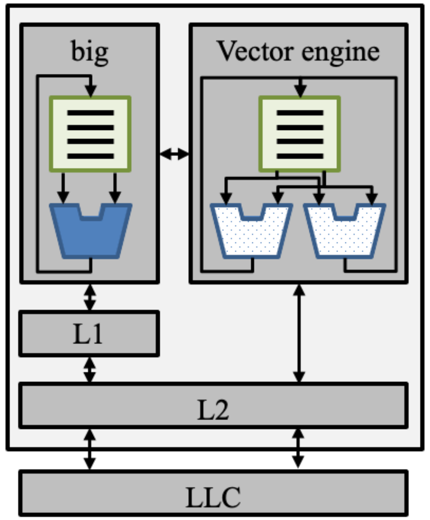
big.LITTLE w/ Integrated Vector Unit



Integrated Vector Unit

- 128-bit vector length
- Exec pipelines:
  - 2x 4-lane 32-bit, or
  - 2x 2-lane 64-bit
- Speculative & out-of-order issue
- Similar to 128-bit packed SIMD unit

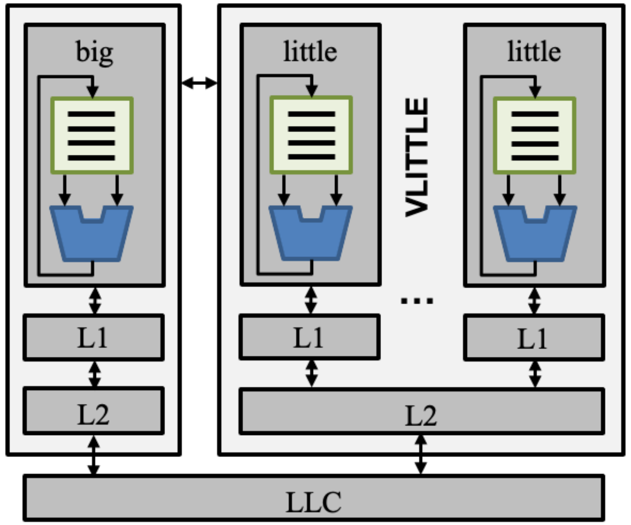
Big Core w/ Decoupled Vector Engine



Decoupled Vector Engine

- 2048-bit vector length
- 3x 16-lane 32-bit exec pipelines
- Comparable to Tarantula

big.VLITTLE

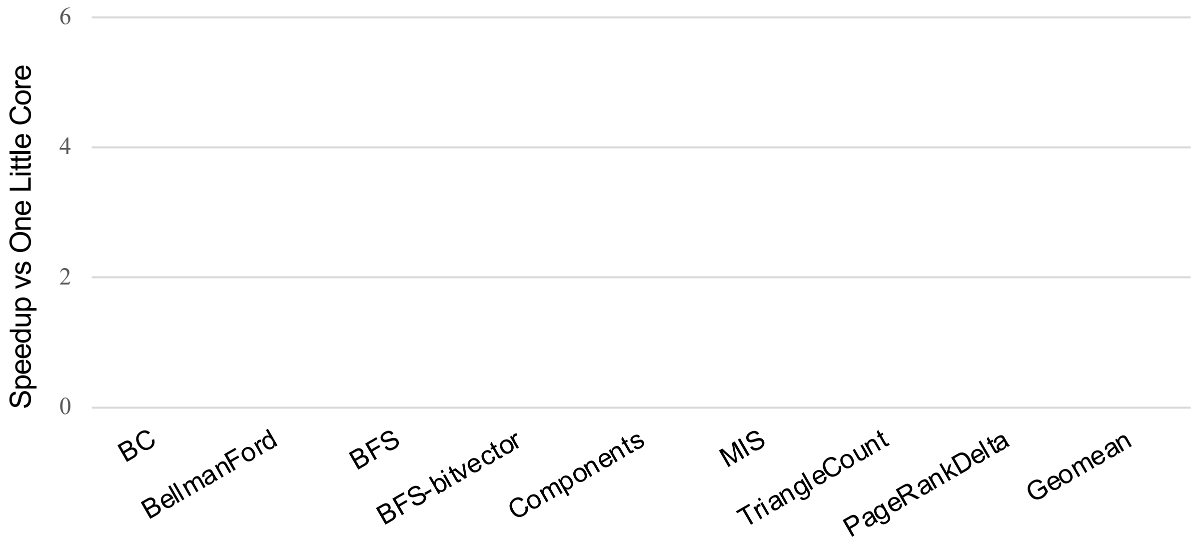


VLITTLE Engine

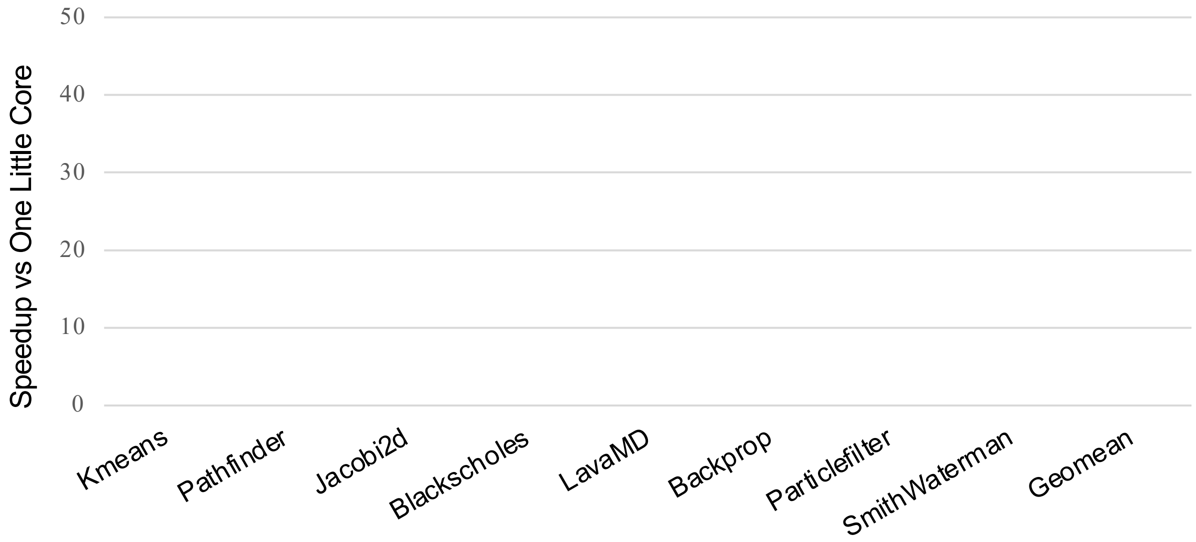
- 512-bit vector length
- 4x little cores
- 2x element groups (chimes)
- Packed sub-word elements

Task- & Data-Parallel Applications

Non-Vectorized Task-Parallel Applications



Vectorized Data-Parallel Applications



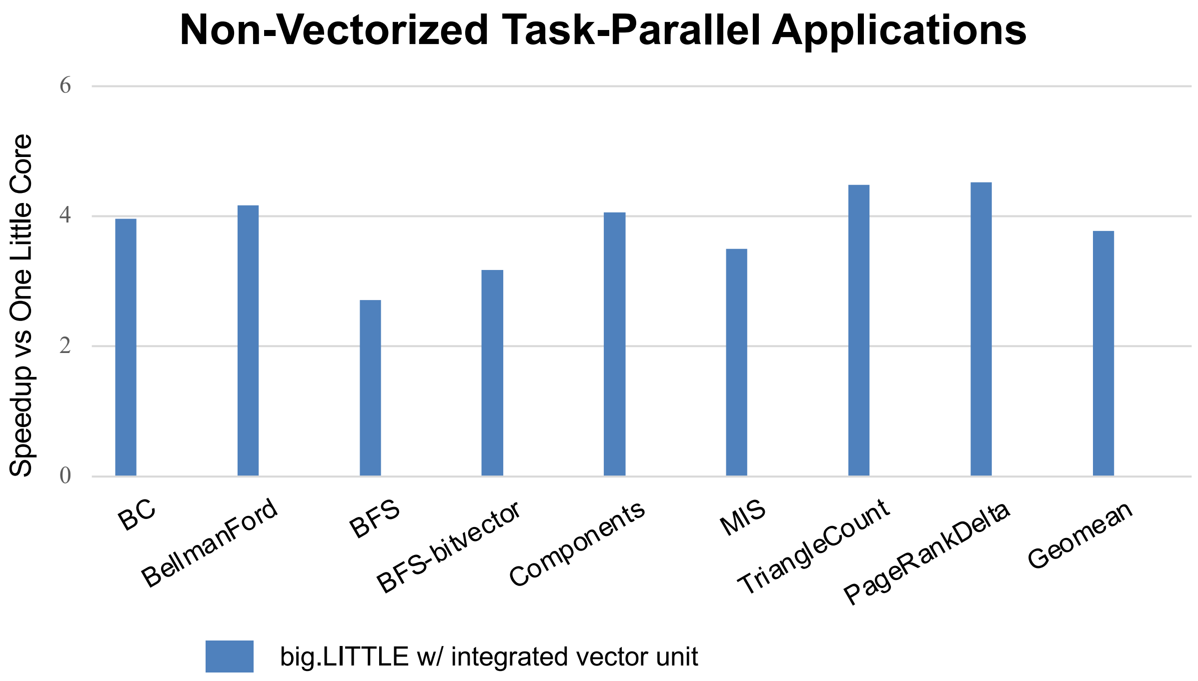
- Irregular task-parallel graph applications
- Ligra suite

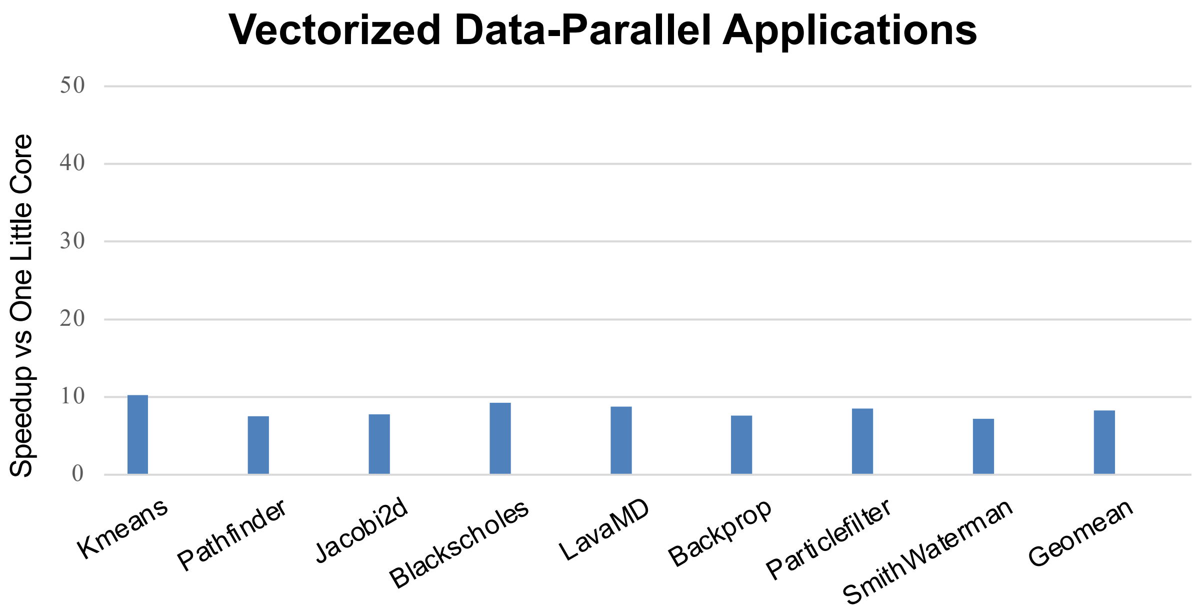
- Regular data-parallel applications
- Rodinia, RiVEC & Genomics suites

Both types of applications are equally important in modern mobile SoCs

big.LITTLE w/ Integrated Vector Unit

CSL





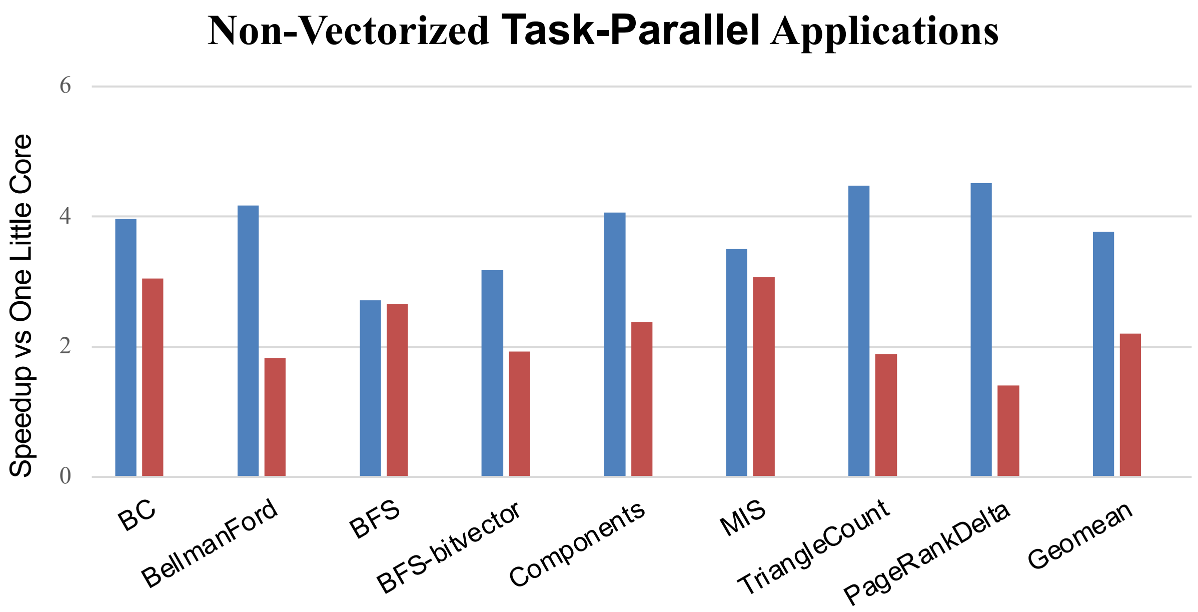
Exploit only thread-level parallelism

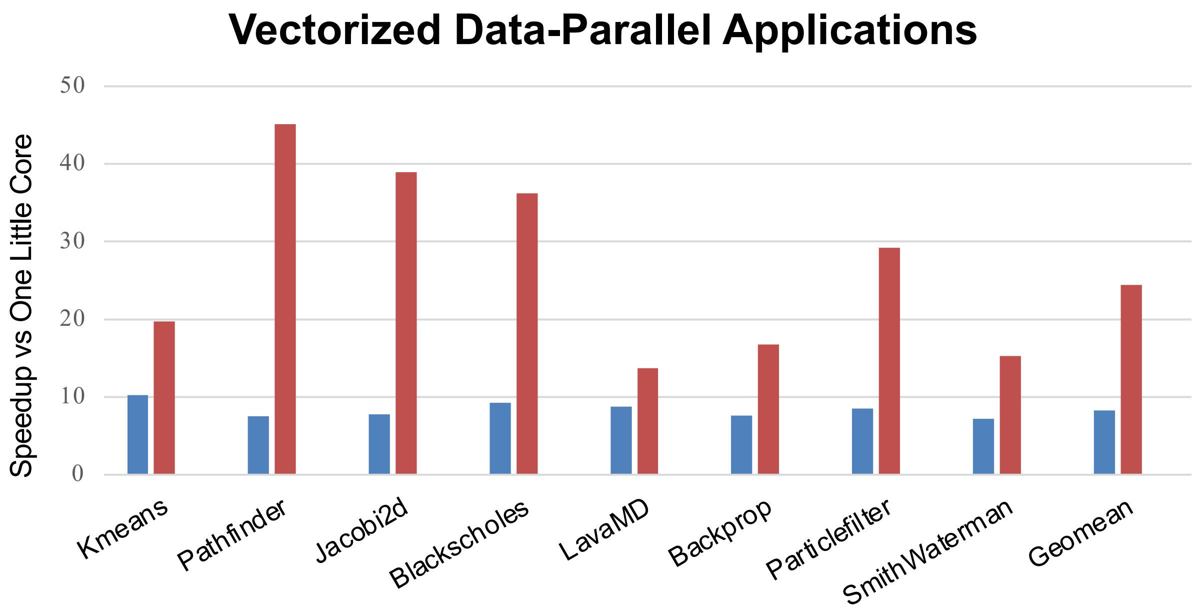
Exploit both thread-level and data-level parallelism

- TLP across big and little cores
- DLP in the big core's integrated vector unit

Work-stealing between threads to maximize performance & utilization

Big core w/ Decoupled Vector Engine





big.LITTLE w/ integrated vector unit

Big core w/ decoupled vector engine

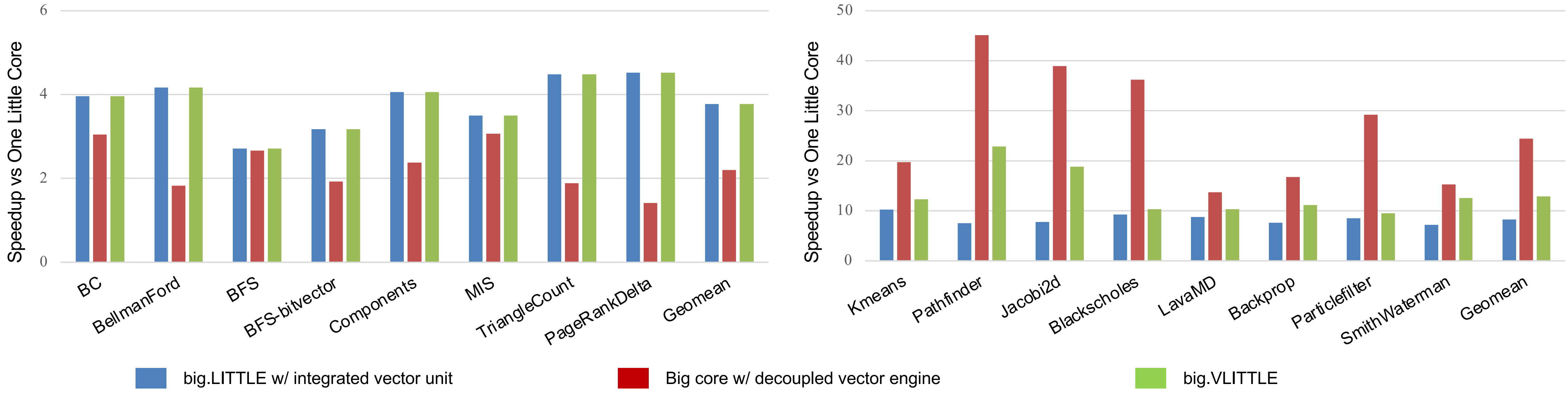
No vector code → can only utilize the big core

Best-in-class performance for vectorized workloads

Decoupled vector engine is specialized for data-level parallelism

big.VLITTLE - Reconfigure to Adapt to Both

Non-Vectorized Task-Parallel Applications | **Vectorized Data-Parallel Applications**



Work as big.LITTLE system to exploit TLP

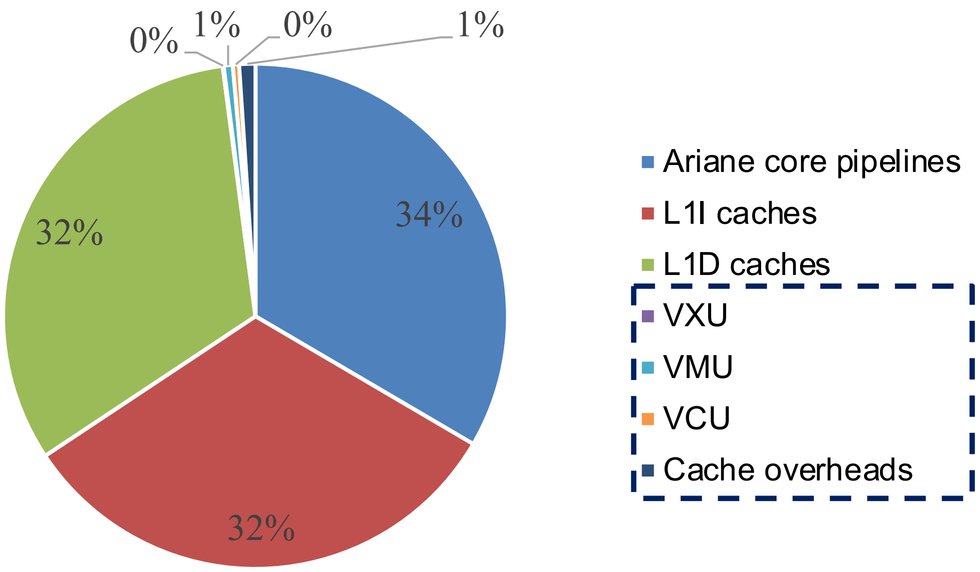
Work as a decoupled vector engine to exploit DLP

big.VLITTLE bridges the performance gap between integrated vector unit and decoupled vector engine through on-demand reconfigurability

big.VLITTLE Brings Area & Power Efficiency

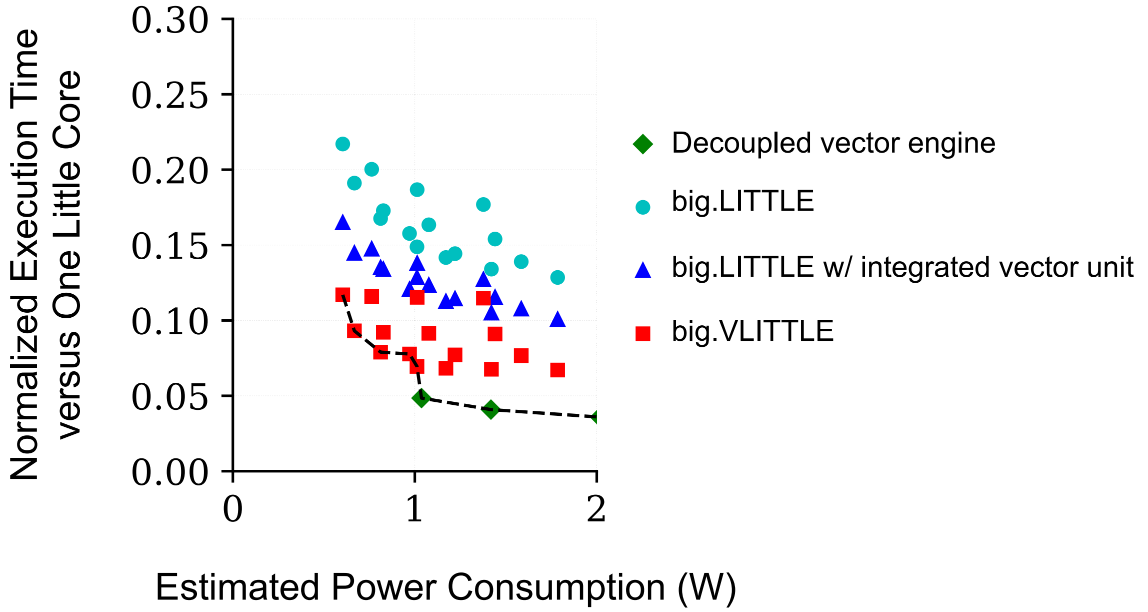
CSL

Area Overheads of big.VLITTLE



Less than 3% area overhead compared to four little cores and their L1 caches

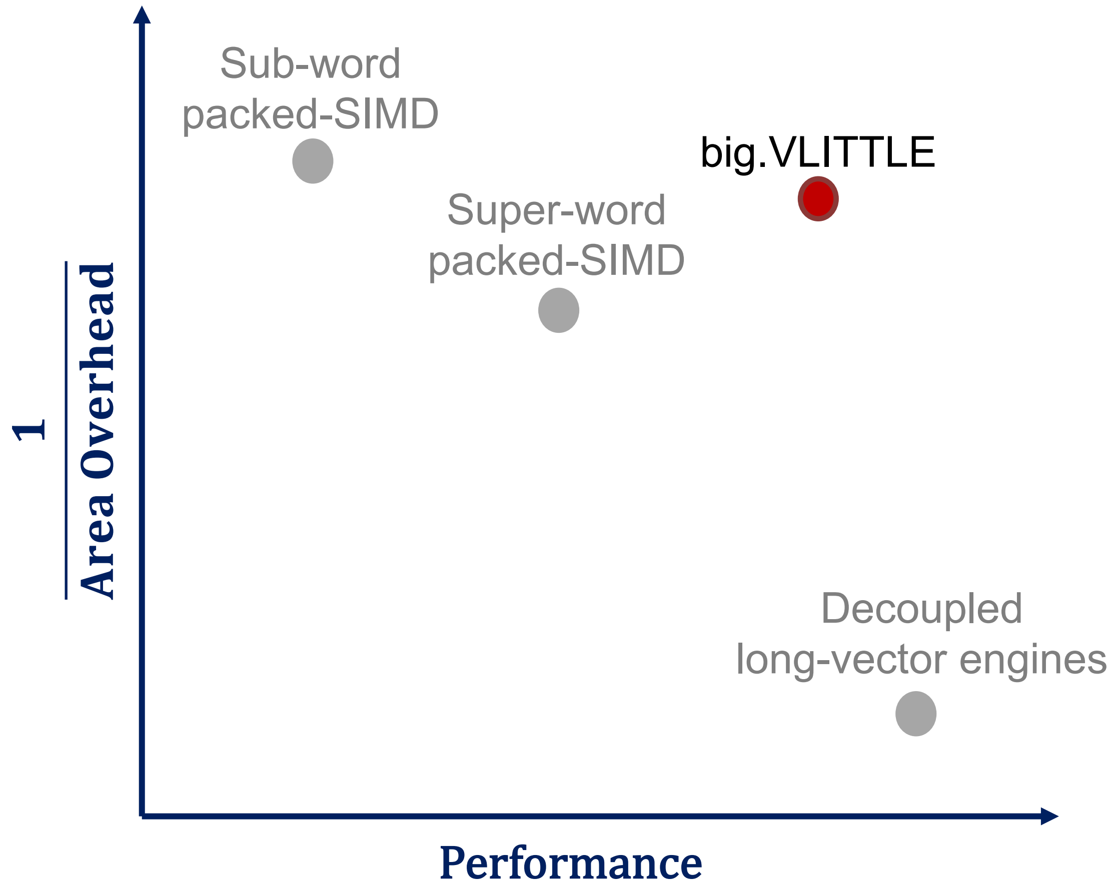
Performance vs Power Consumption for Data-Parallel Applications (geomean) at Different Voltage/Frequency Points



- big.VLITTLE is more power-efficient than big.LITTLE w/ integrated vector unit
- big.VLITTLE is close to decoupled vector engine in terms of power efficiency

big.VLITTLE: Takeaways

CSL



- Reconfigure on demand to maximize performance of **both task- and data-parallel** workloads
- **Leverage little cores as vector lanes** in vector mode
- **Reconfigure existing parts of L1 caches** to deliver **high memory throughput** in vector mode
- **Higher performance per area and power** than a big.LITTLE system with an integrated vector unit
- **Close to decoupled vector engine** which is best-in-class for DLP in terms of **area and power efficiency without trading off task-parallel performance**

This work was supported in part by NSF, the Center for Applications Driving Architectures (ADA), one of six centers of JUMP, a Semiconductor Research Corporation program co-sponsored by DARPA, and equipment, tool, and/or physical IP donations from Intel, Synopsys, and Cadence.