BlackParrot: An Agile Open Source RISC-V Accelerator Host Multicore

Daniel Petrisko[1], Farzam Gilani[1], Mark Wyse[1], Tommy Jung[1], Scott Davidson[1], Paul Gao[1], Chun Zhao[1], Sripathi Muralitharan[1], Shashank Ranga[1], Zahra Azad[2], Sadullah Canakci[2], Bandhav Veluri[1], Tavio Guarino[1], Ajay Joshi[2], Mark Oskin[1], and Michael Bedford Taylor[1]

[1]University of Washington [2]Boston University

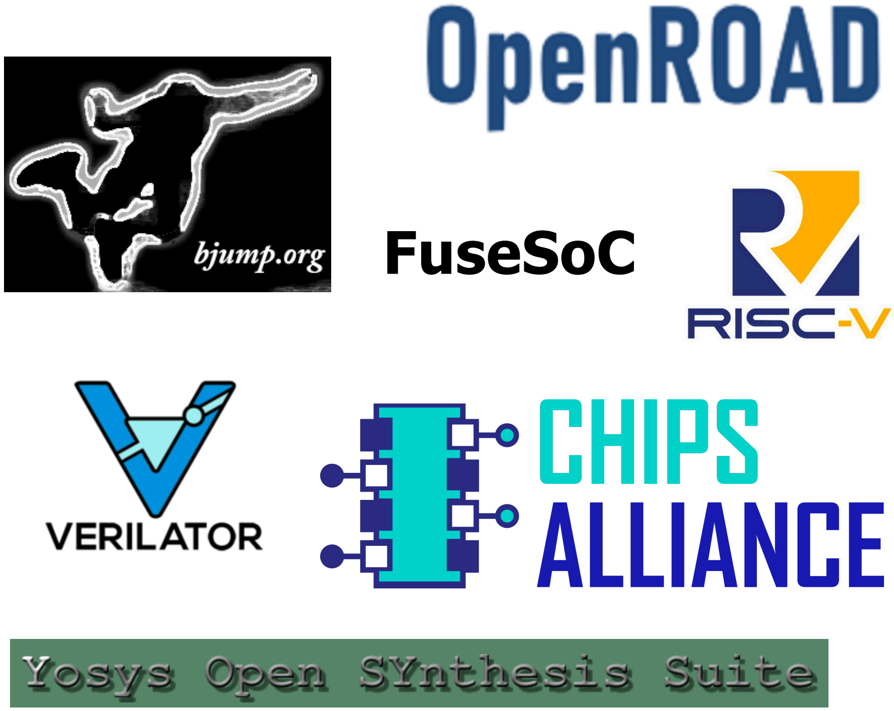
https://github.com/black-parrot

Hardware Development is Going Open-Source

Traditionally, hardware design required expensive, proprietary tooling and IP

Competitive open-source offerings are challenging this status quo

Accessibility and rising demand for custom silicon has led to surge in hardware developers



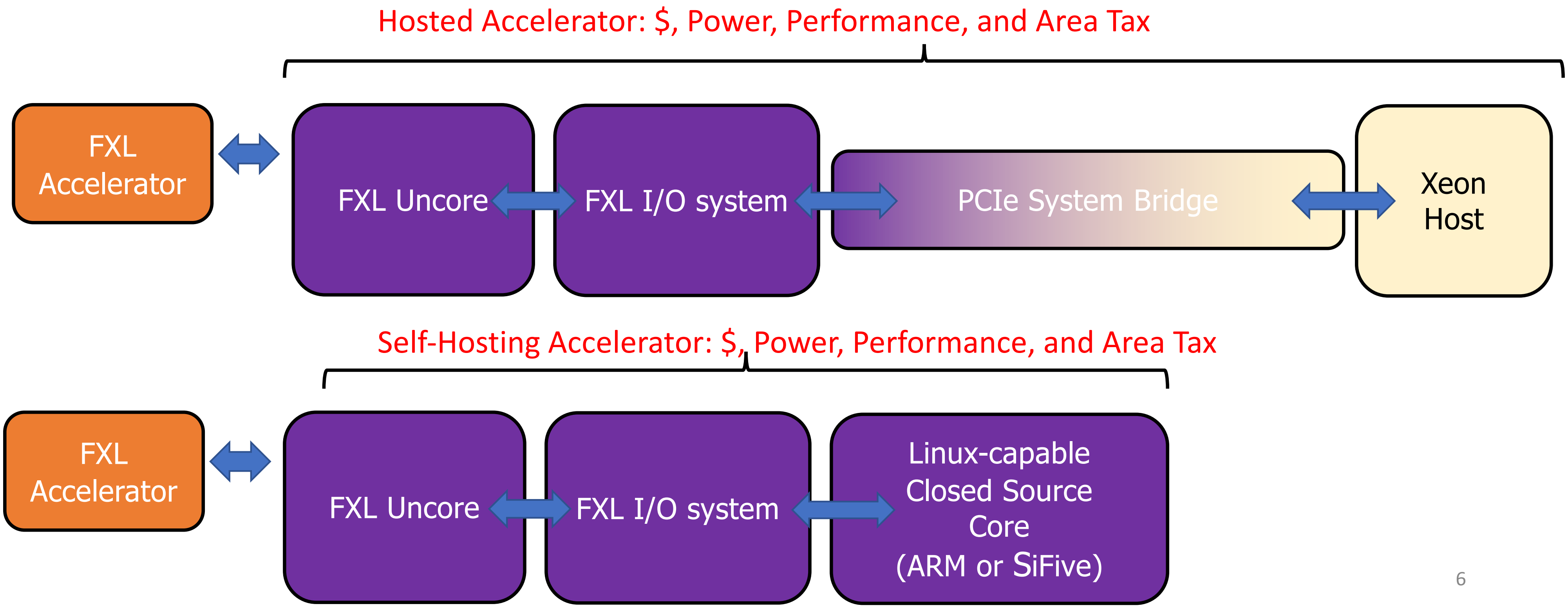
Diverse Workloads Demand Custom Silicon

Accelerators provide performance and energy efficiency benefits

“FXL: Future Accelerator for HPC”

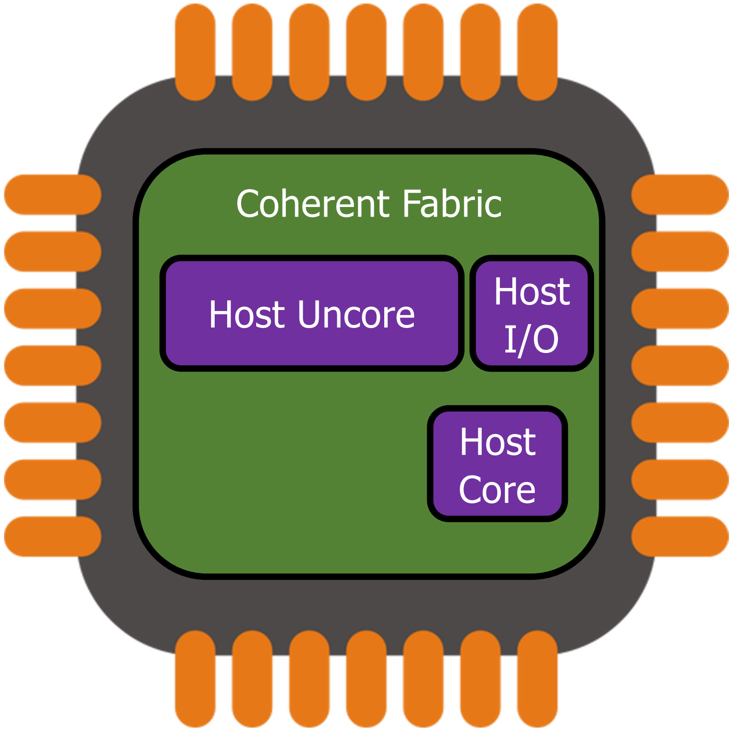
But the Accelerator Is Just One Piece of a System

The system that cares for and feeds the accelerator is often harder to design than the accelerator.



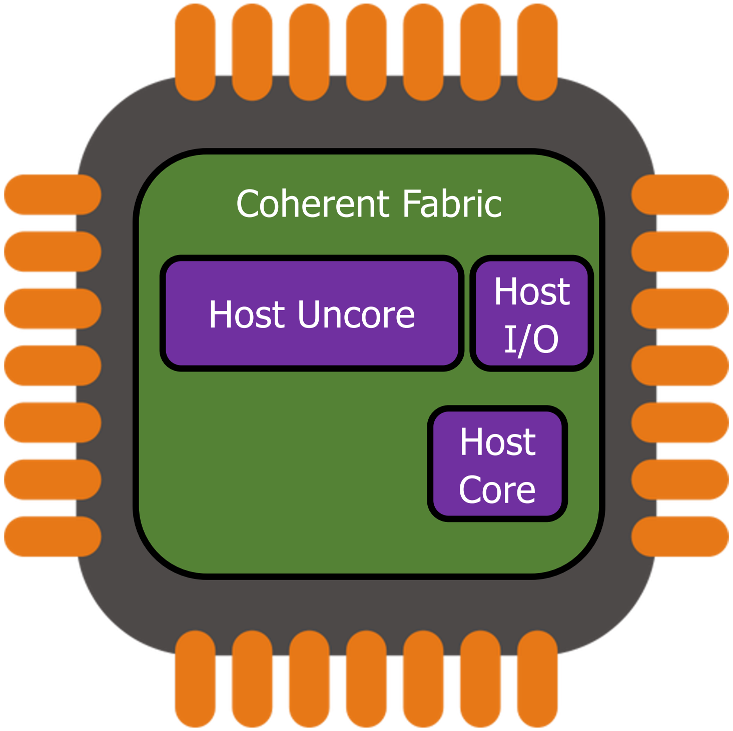
Hosted Accelerator: $, Power, Performance, and Area Tax
FXL Accelerator
FXL Uncore
FXL I/O system
PCIe System Bridge
Xeon Host
Self-Hosting Accelerator: $, Power, Performance, and Area Tax
FXL Accelerator
FXL Uncore
FXL I/O system
Linux-capable Closed Source Core (ARM or SiFive)

An Ideal Host Processor Would Be:



An Ideal Host Processor Would Be:

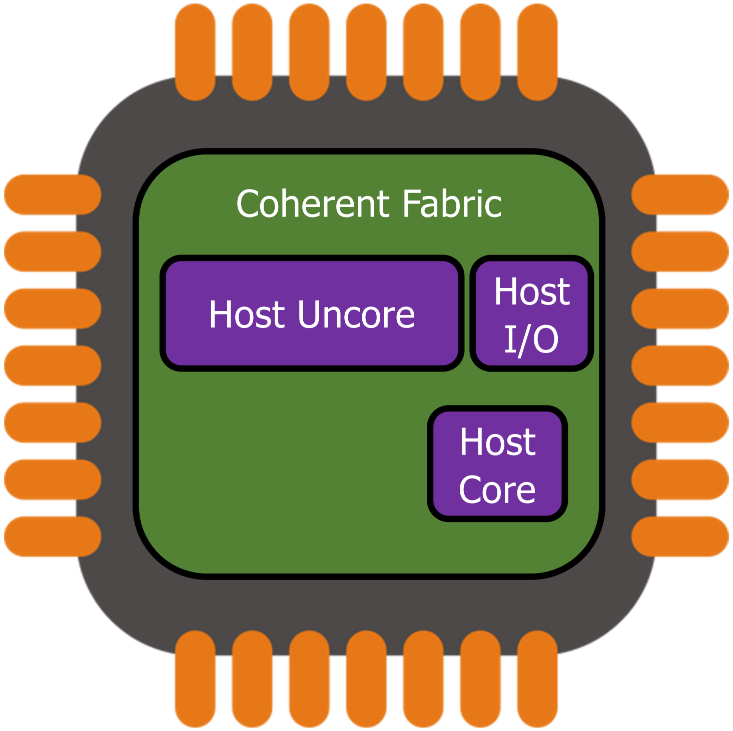
Efficient, performant and reliable



An Ideal Host Processor Would Be:

Efficient, performant and reliable

Easy to integrate with a variety of architectures

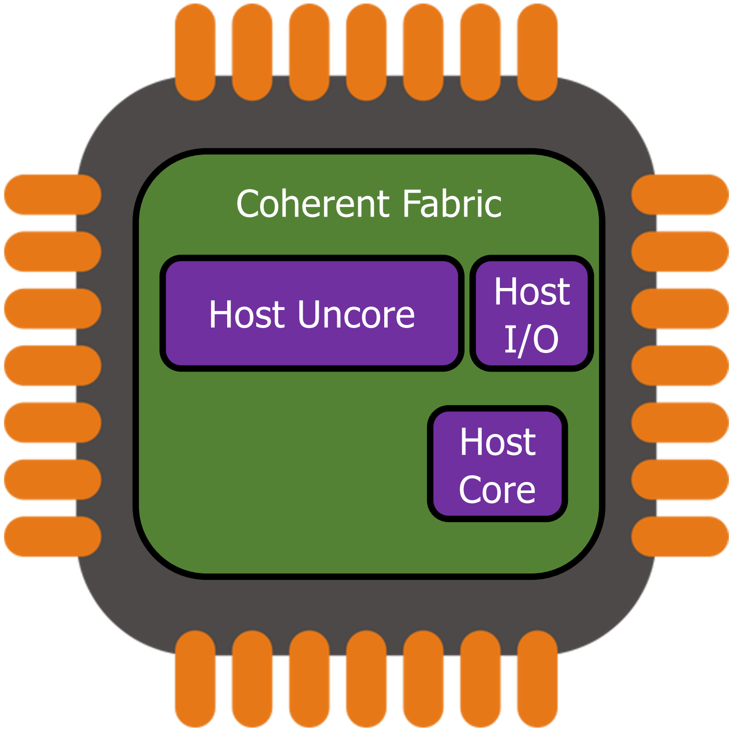


An Ideal Host Processor Would Be:

Efficient, performant and reliable

Easy to integrate with a variety of architectures

Thoroughly verified and silicon-validated



Coherent Fabric

Host Uncore

Host I/O

Host Core

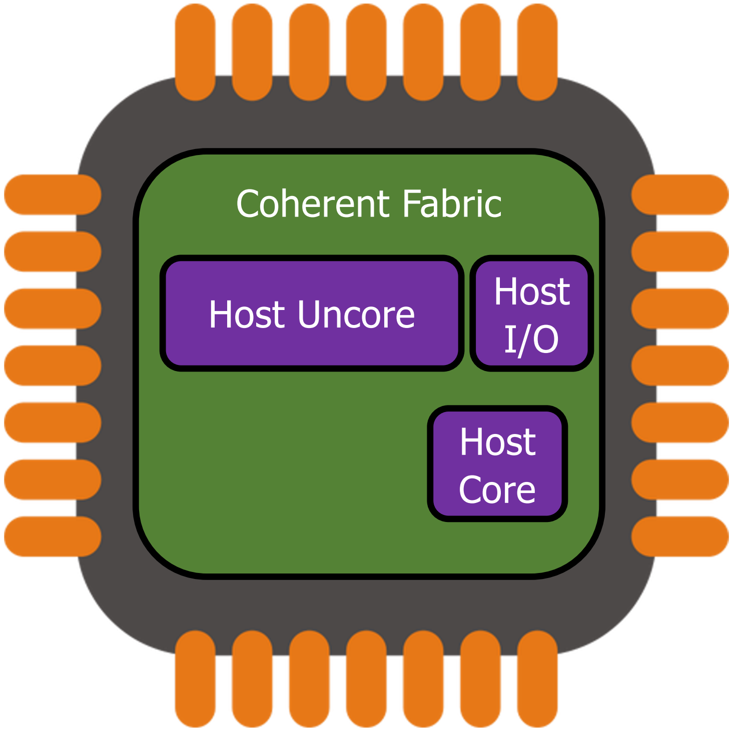
An Ideal Host Processor Would Be:

Efficient, performant and reliable

Easy to integrate with a variety of architectures

Thoroughly verified and silicon-validated

Free and open-source!



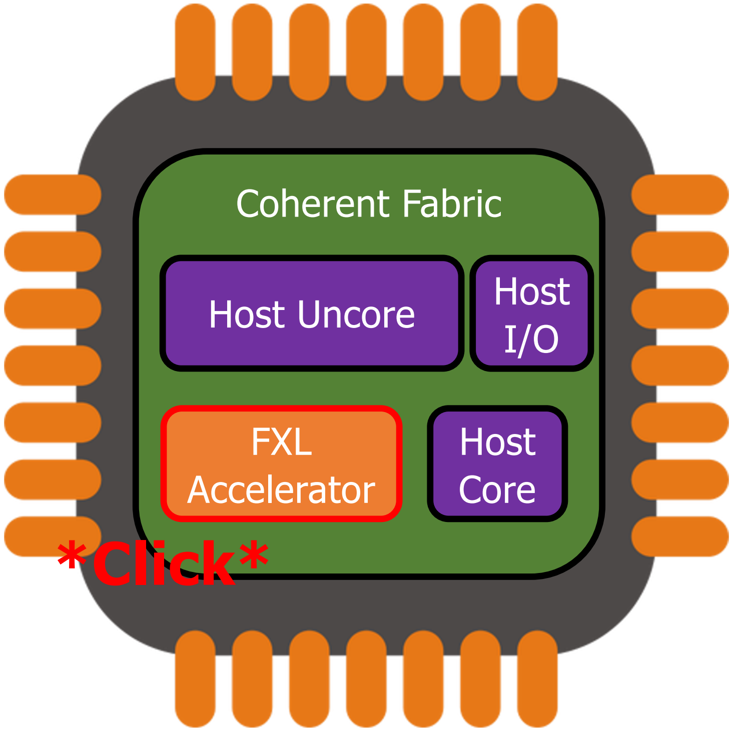
An Ideal Host Processor Would Be:

Efficient, performant and reliable

Easy to integrate with a variety of architectures

Thoroughly verified and silicon-validated

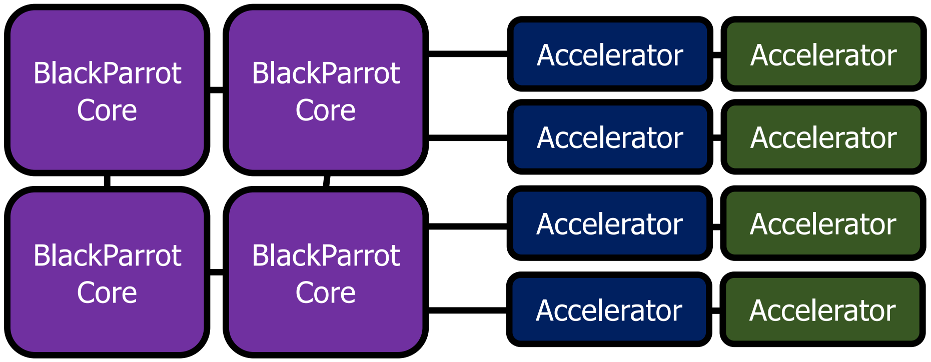
Free and open-source!



BlackParrot: A “Base Class” for Accelerator SoCs

Goal:
Be the default Linux-capable host multicore for and by the world

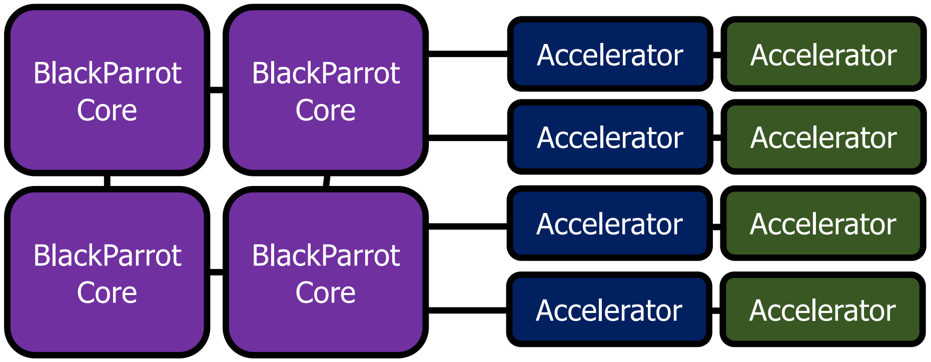
- RV64GC (IMAFDC) ISA
- MSU Privilege Mode
- SV39 Virtual Memory
- 32kB VIPT I$ and D$
- Directory-based coherence
- Scalable, distributed L2 cache
- Linux-capable
- BSD-3 licensed



BlackParrot: A “Base Class” for Accelerator SoCs

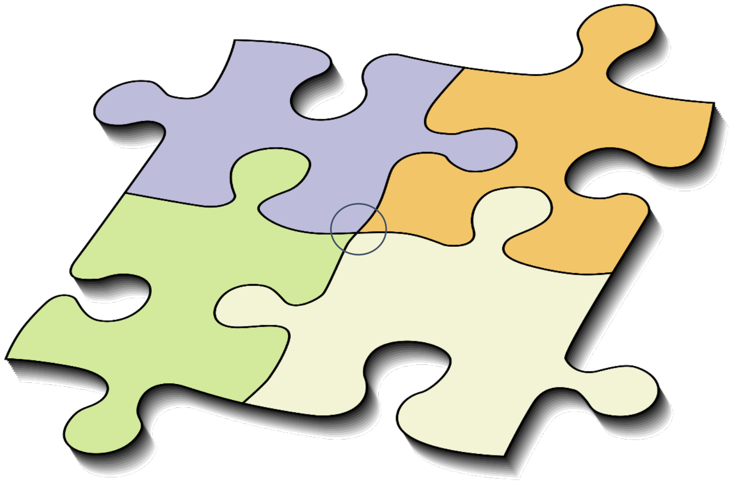
Goal:
Be the default Linux-capable host multicore for and by the world

- RV64GC (IMAFDC) ISA
- MSU Privilege Mode
- SV39 Virtual Memory
- 32kB VIPT I$ and D$
- Directory-based coherence
- Scalable, distributed L2 cache
- Linux-capable
- BSD-3 licensed



Not our goal: “Destroy all x86 servers! Watch out Intel!” =)

BlackParrot: Four Success Metrics



BlackParrot: Four Success Metrics

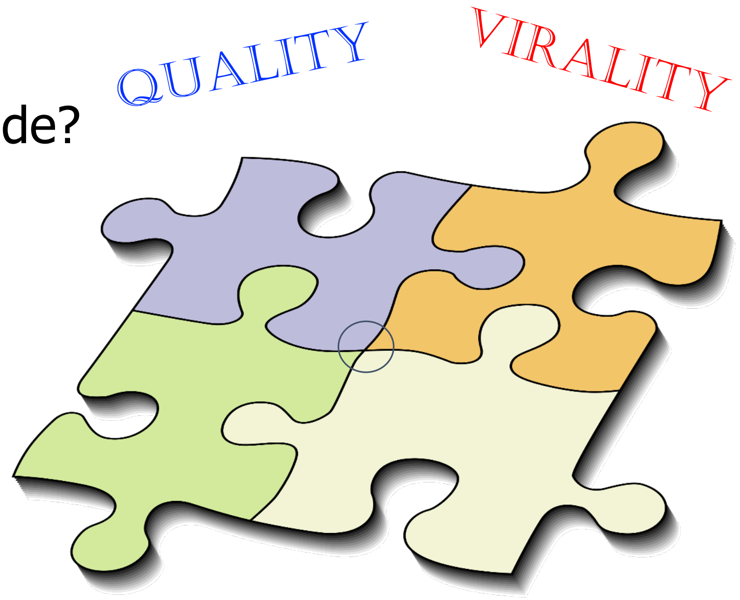
Will people trust our code?

- Is it easy to understand?
- Is it secure?
- Is it validated?
- Will you put it in Silicon?

BlackParrot: Four Success Metrics

Will people trust our code?

- Is it easy to understand?
- Is it secure?
- Is it validated?
- Will you put it in Silicon?



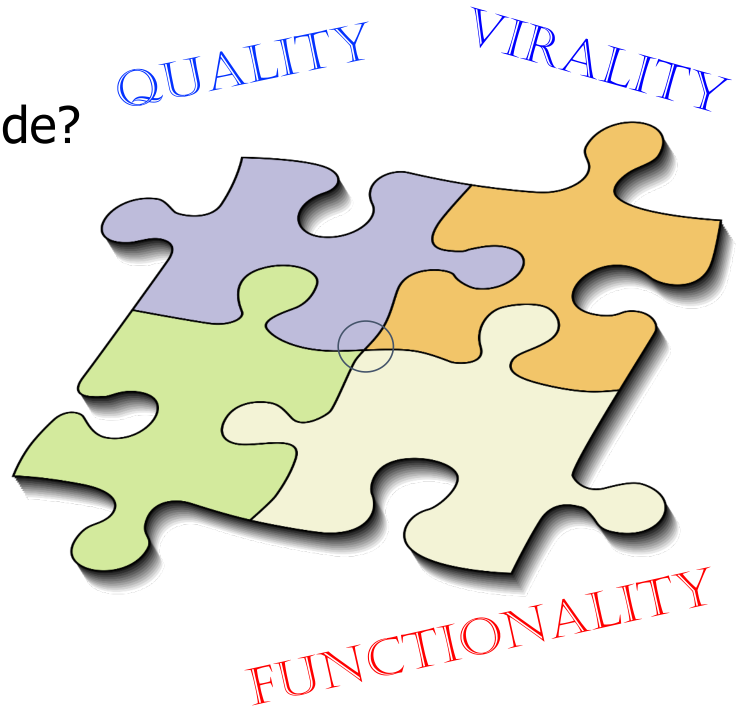
Will people use our code?

- Can we convince the smartest people in the world to improve it?
- Can it scale to many users and developers?
- Will companies invest and become stewards of the code?

BlackParrot: Four Success Metrics

Will people trust our code?

- Is it easy to understand?
- Is it secure?
- Is it validated?
- Will you put it in Silicon?

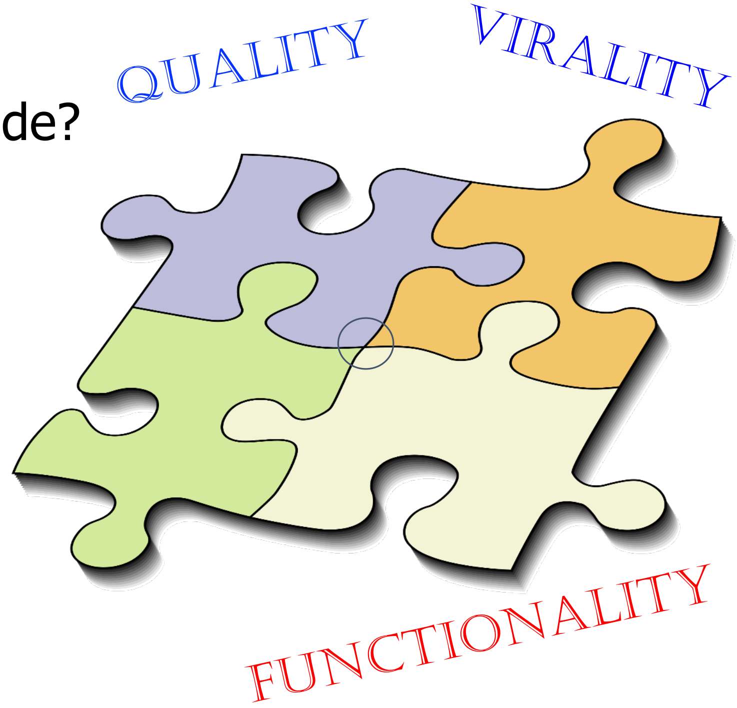


Will people use our code?

- Can we convince the smartest people in the world to improve it?
- Can it scale to many users and developers?
- Will companies invest and become stewards of the code?

Does the code have the features people need?

BlackParrot: Four Success Metrics



Will people trust our code?

- Is it easy to understand?
- Is it secure?
- Is it validated?
- Will you put it in Silicon?

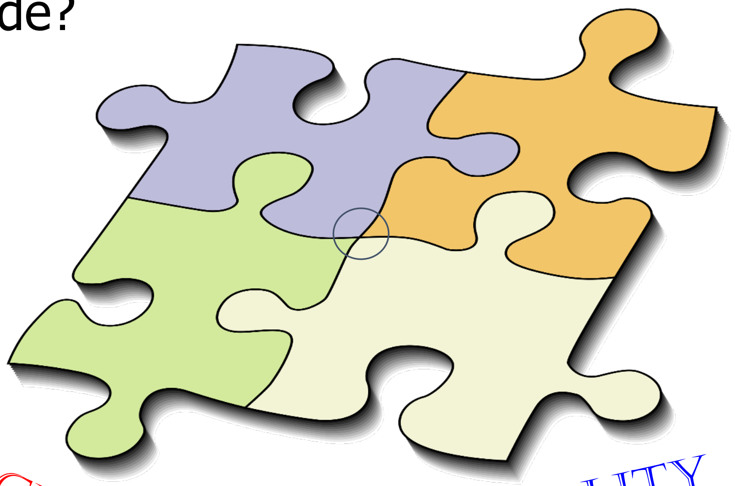
Will people use our code?

- Can we convince the smartest people in the world to improve it?
- Can it scale to many users and developers?
- Will companies invest and become stewards of the code?

Does the code have the features people need?

And leave out the ones they don't?

BlackParrot: Four Success Metrics



QUALITY

Will people trust our code?

- Is it easy to understand?
- Is it secure?
- Is it validated?
- Will you put it in Silicon?

VIRALITY

Will people use our code?

- Can we convince the smartest people in the world to improve it?
- Can it scale to many users and developers?
- Will companies invest and become stewards of the code?

FUNCTIONALITY

Does the code have the features people need?

EFFICIENCY

Is the code Pareto optimal in terms of power, performance, and area?

The BlackParrot Manifesto

Be TINY

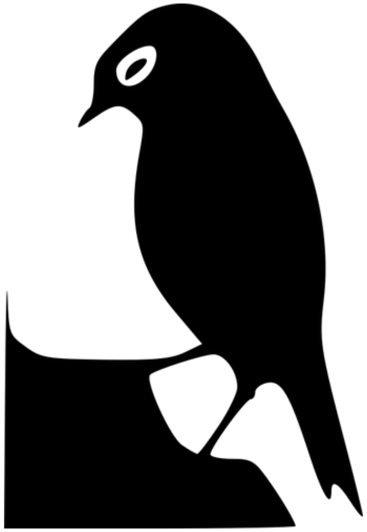
- Place a premium on small, understandable code base
- Minimize unused features or configurations that increase complexity

Be Modular

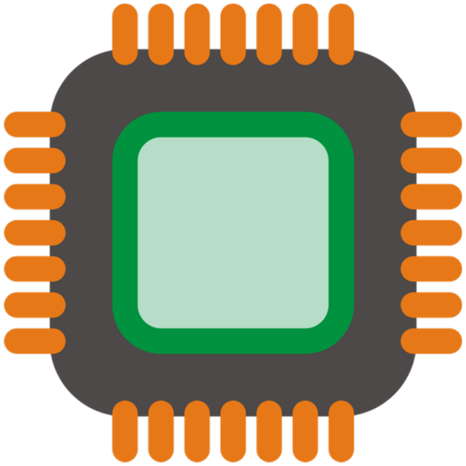
- Use well-defined interfaces to enable scalable, global participation
- Maintain modular testability and continuous integration

Be Friendly

- Combat “Not Invented Here” Syndrome
- Welcome contributions and distributed ownership



BlackParrot System Architecture

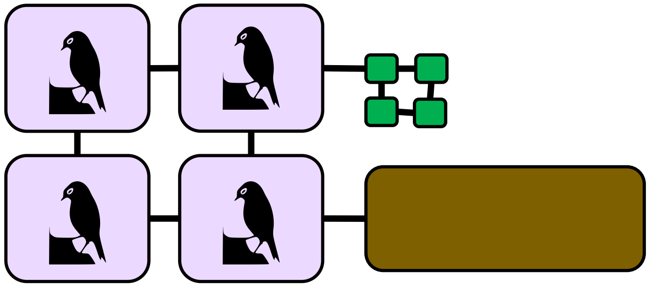


Fully Coherent Multicore

Designed as a host core for accelerator-based SoCs

…so make it as easy to connect accelerators!

Coherent, streaming, even multicore accelerators are supported



BedRock Cache Coherence System

First ever open-source programmable directory-based coherence controller

Directory-based, ***race-free by design***, MESI protocol

- Fully inclusive coherence directory
- Connected by 3 point-to-point ordered networks for coherence traffic

Cache Coherence Engine (CCE): coherence directory controller

- Manages coherence for a subset of the physical address space
- Backed by non-inclusive L2 cache

Local Cache Engine (LCE): L1 I$, L1 D$, I/O cache, accelerator, etc.

- Standard SystemVerilog module used by any processor or accelerator to allow it to participate in the cache coherence protocol
- Must request read/write permissions from CCE for cache blocks

Cache coherence is a well-known source of bugs in commercial processors.
Being race-free by design is a key component of our strategy for being a globally maintainable source base. By keeping complexity low, we can scale the # of contributors to the source.

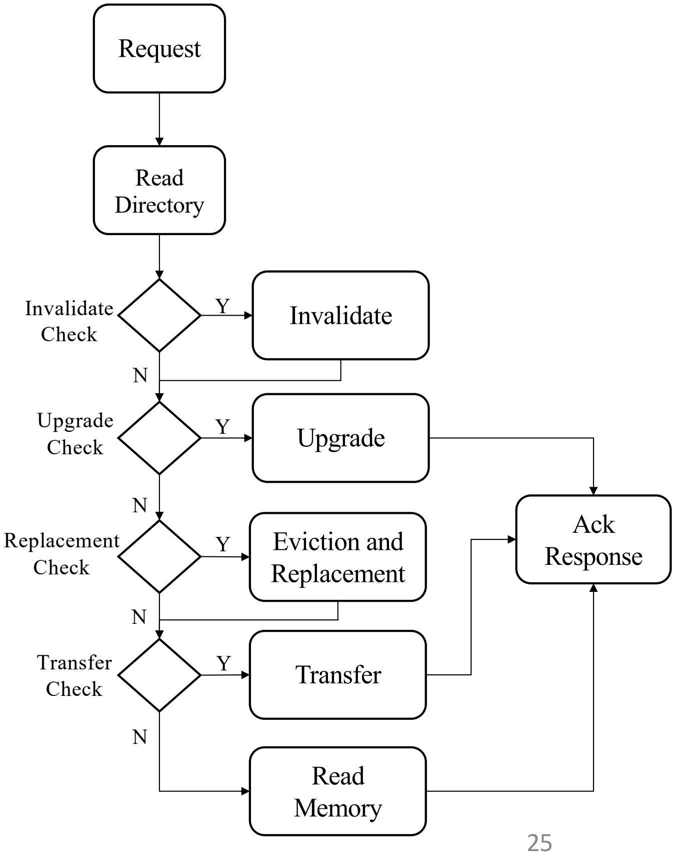
BedRock: A Race-Free Directory-Based Coherence Protocol

Atomic transactions, zero transient states in protocol!

Significantly faster to verify correctness compared to traditional MESI

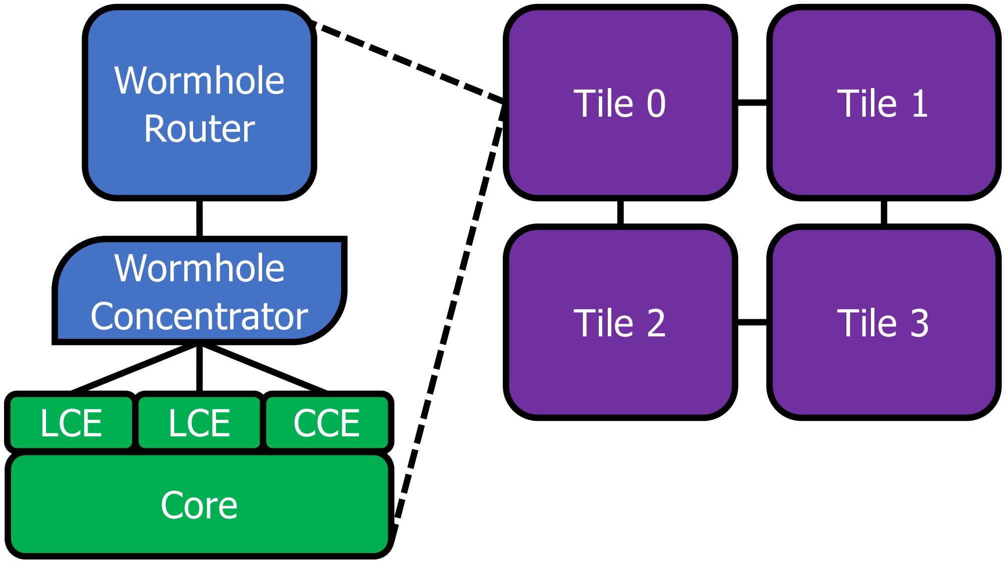
- 26.9x faster for a 6-cache system
- Verified correct in 3.5 hours for 7-cache system while traditional MESI protocol did not finish verification within 72 hours

LCEs are “dumb”, all control logic is in CCE

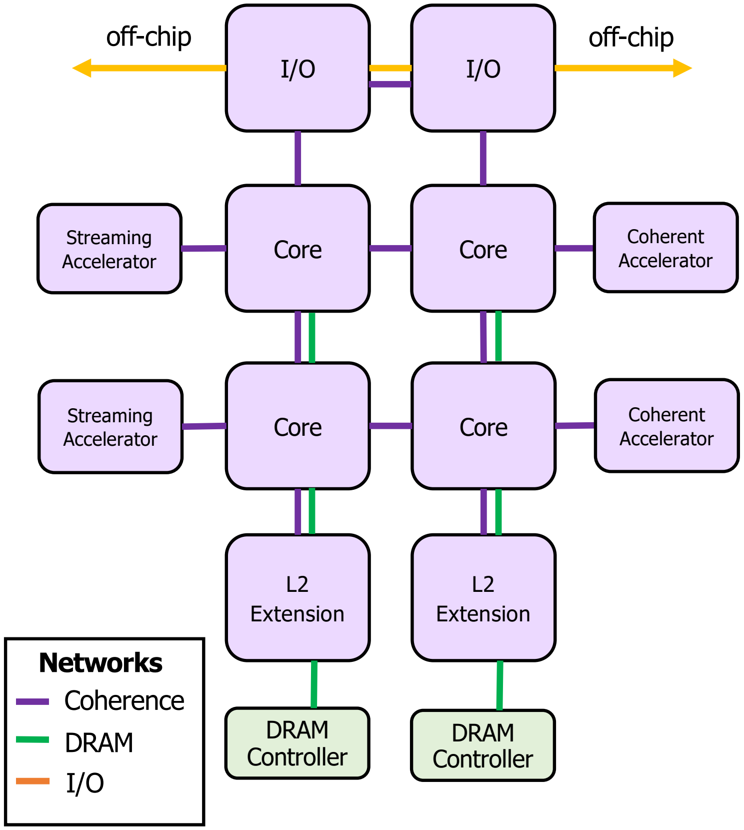


Tiled NoC-Based SoC Provides Scalability

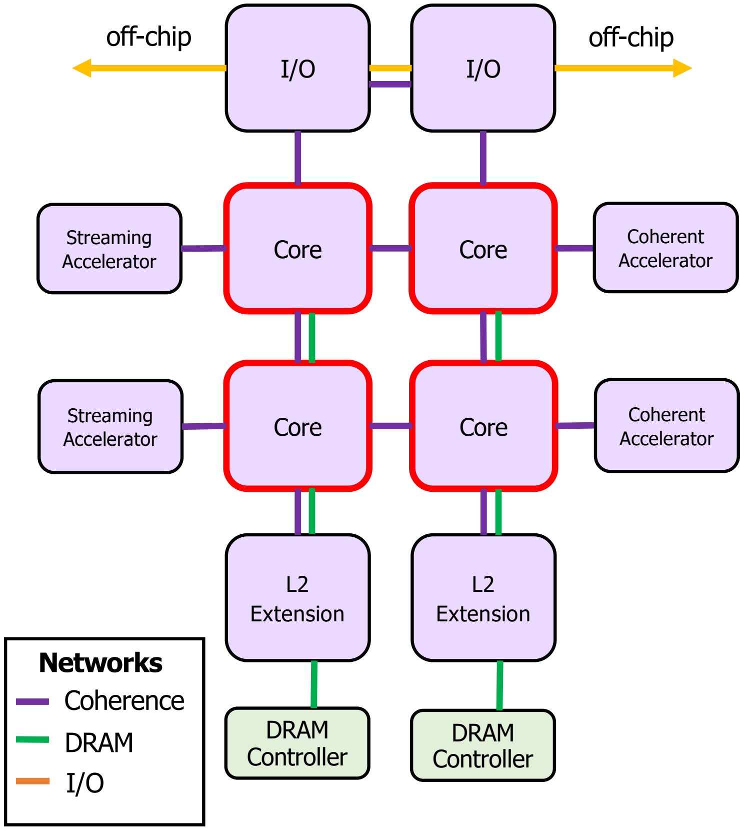
- BlackParrot comprises 3 NoCs
  - BedRock Network for coherence
  - Memory network for DRAM
  - I/O network for off-chip access
- Wormhole routing allows flexibility between routing congestion, bandwidth and latency
- Regularized tiles are efficient for hierarchical CAD flows



BlackParrot Supports Diverse Tile Types



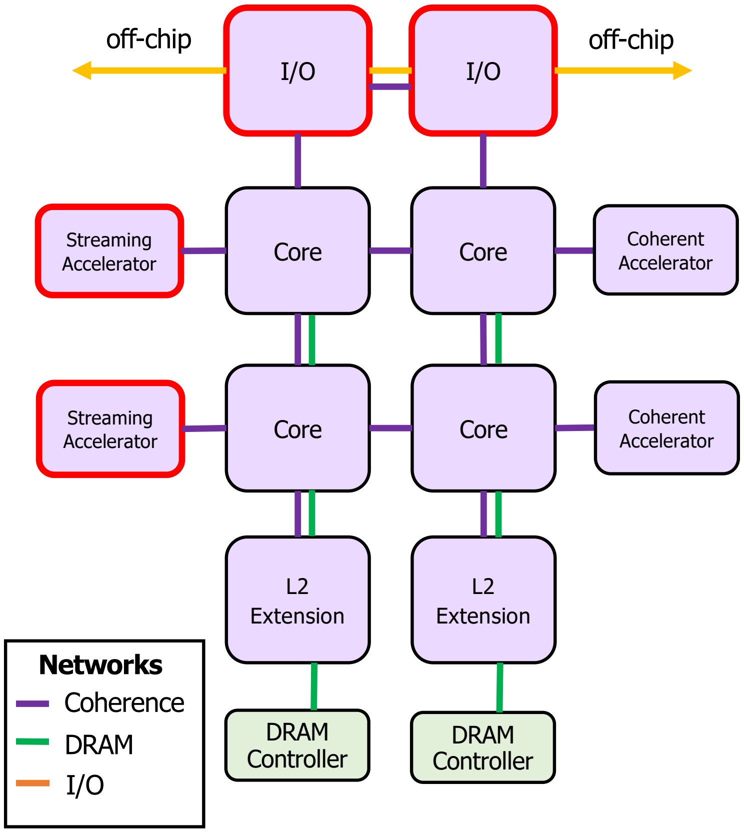
BlackParrot's Core Tiles



Core Tile

- Most common tile type
- Contains single-coherent core of a multicore system
- Contains uncore components to communicate with the rest of the system

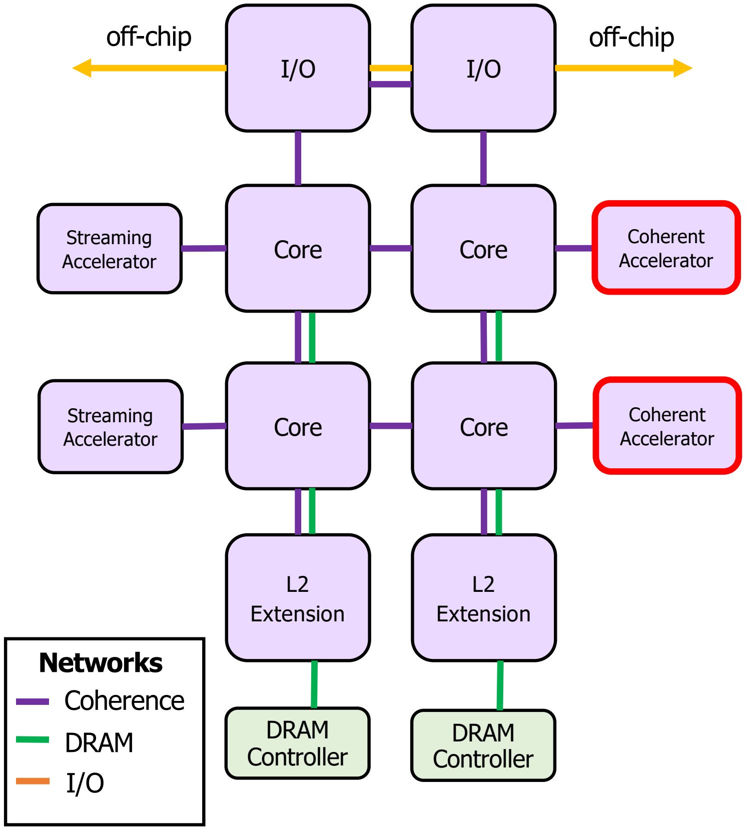
BlackParrot’s Streaming Tiles



Streaming Accelerator Tile

- Streams data to and from cores as uncached requests
- Does not have cache coherent memory
- Can be used for a broad range of accelerators – from fixed function accelerator to integrated GPUs

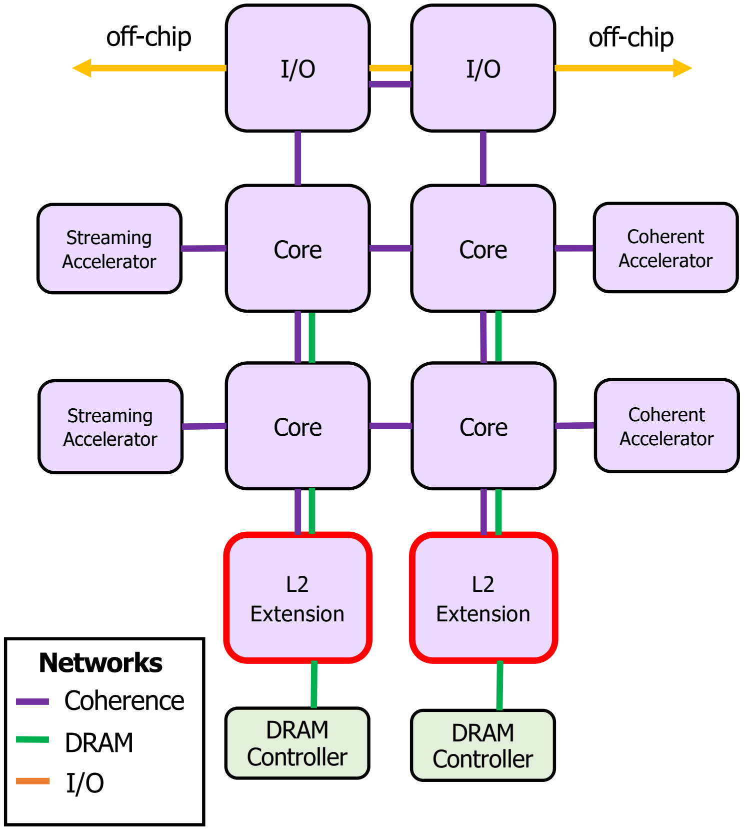
BlackParrot’s Coherent Accelerator Tiles



Coherent Accelerator Tile

- Supports accelerators with cache coherent memory
- Allows fine-grained communication between accelerator and core
- Works well for co-operative workloads

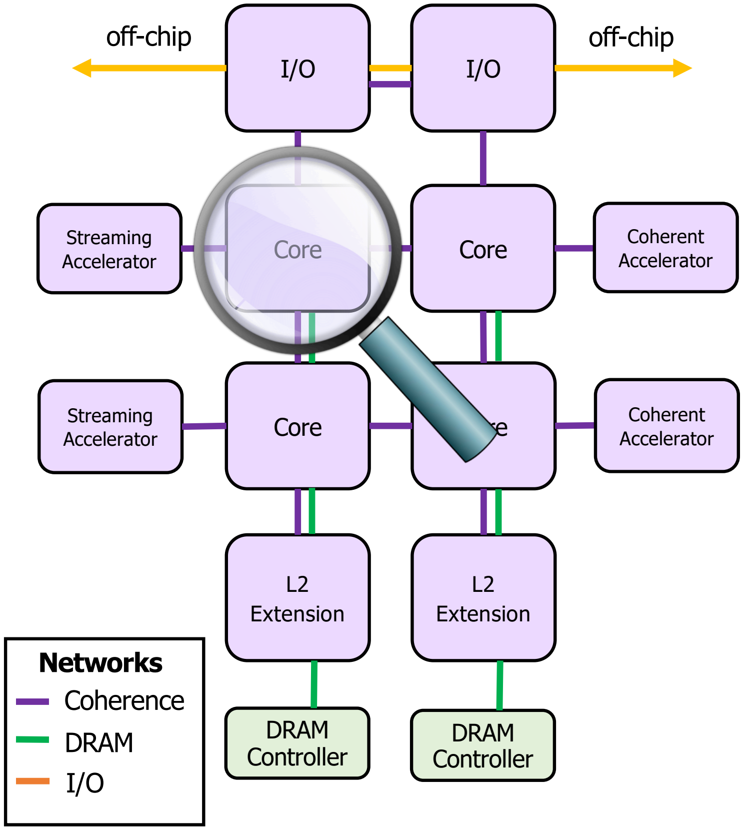
BlackParrot’s L2 Tiles



L2 Tile

- Supports large, distributed L2 cache and directory tags
- Provides higher memory-to-compute ratio

BlackParrot: Peaking inside



off-chip

I/O

I/O

off-chip

Streaming Accelerator

Core

Core

Coherent Accelerator

Streaming Accelerator

Core

e

Coherent Accelerator

L2 Extension

L2 Extension

DRAM Controller

DRAM Controller

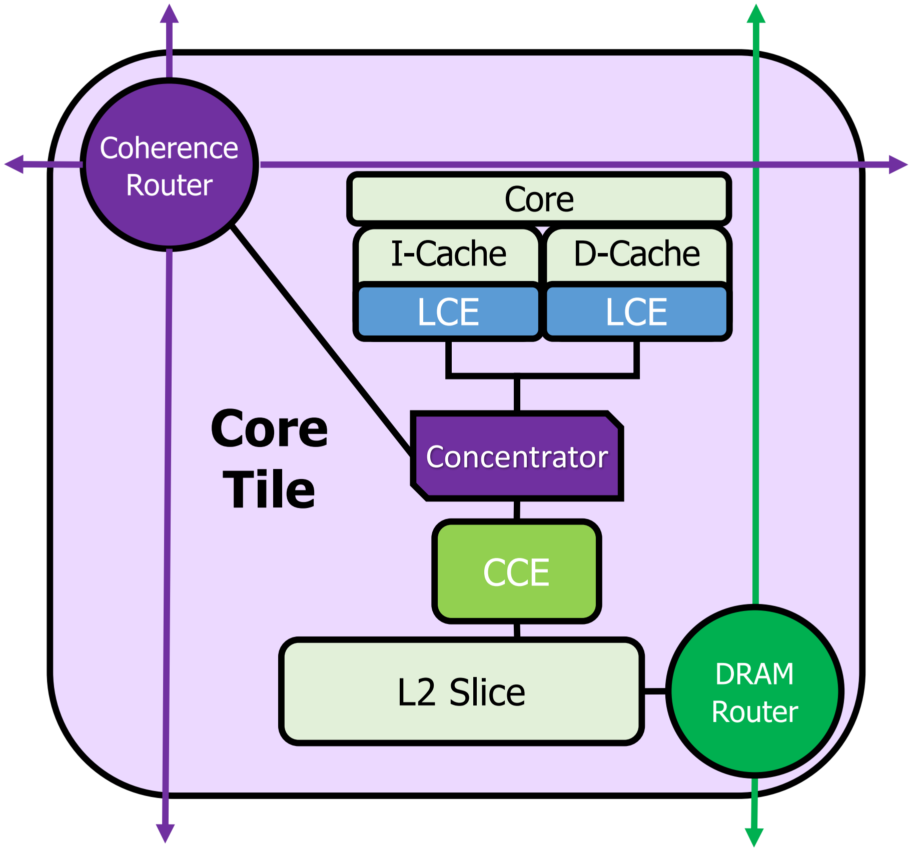
Networks

Coherence

DRAM

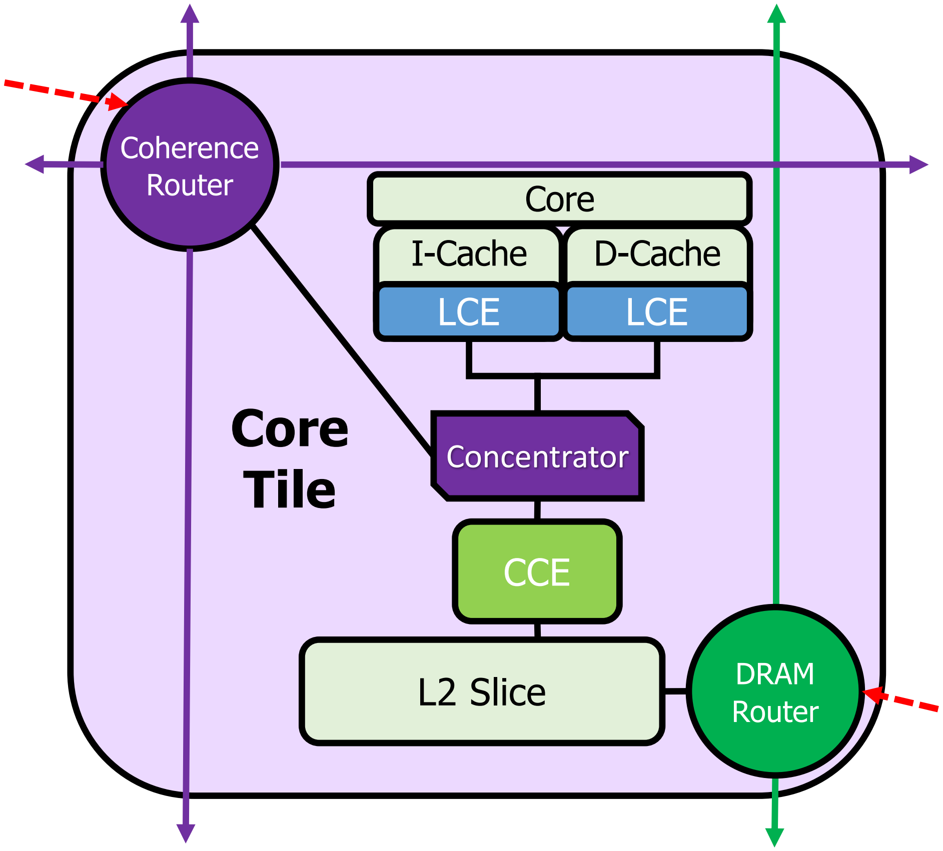
I/O

Assemble Tiles Out of Reusable Components



Assemble Tiles Out of Reusable Components

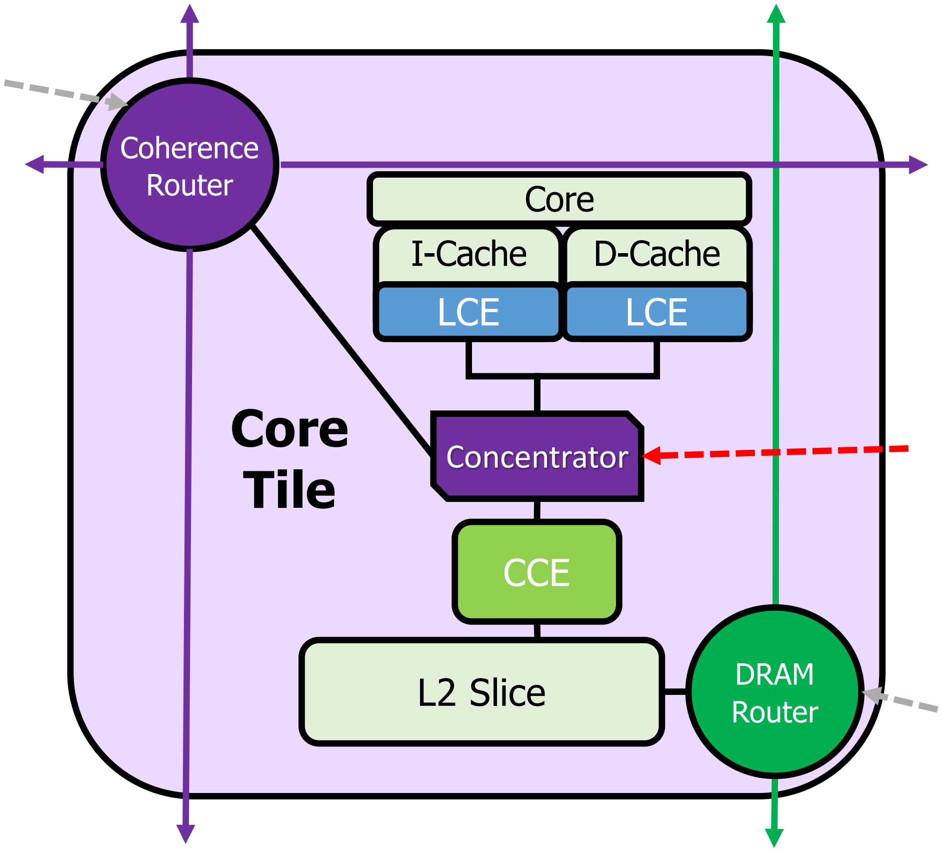
BaseJump STL Wormhole Router



BaseJump STL Wormhole Router

Assemble Tiles Out of Reusable Components

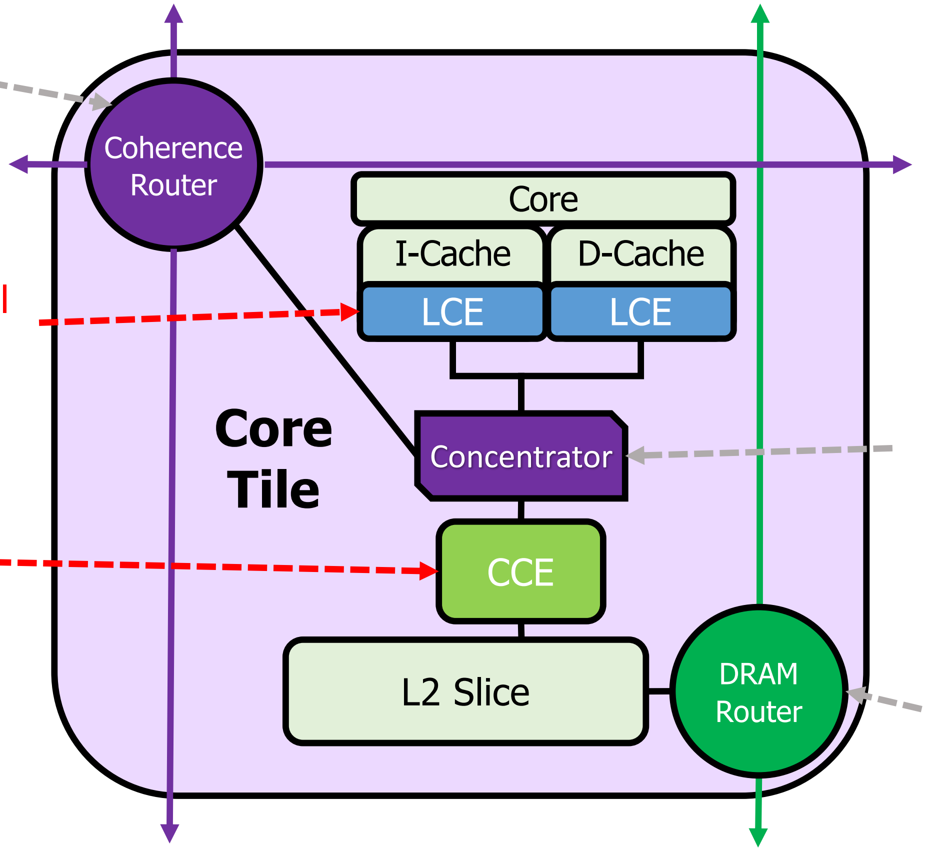
BaseJump STL
Wormhole
Router



BaseJump STL
Wormhole
Concentrator

BaseJump STL
Wormhole
Router

Assemble Tiles Out of Reusable Components



BaseJump STL Wormhole Router

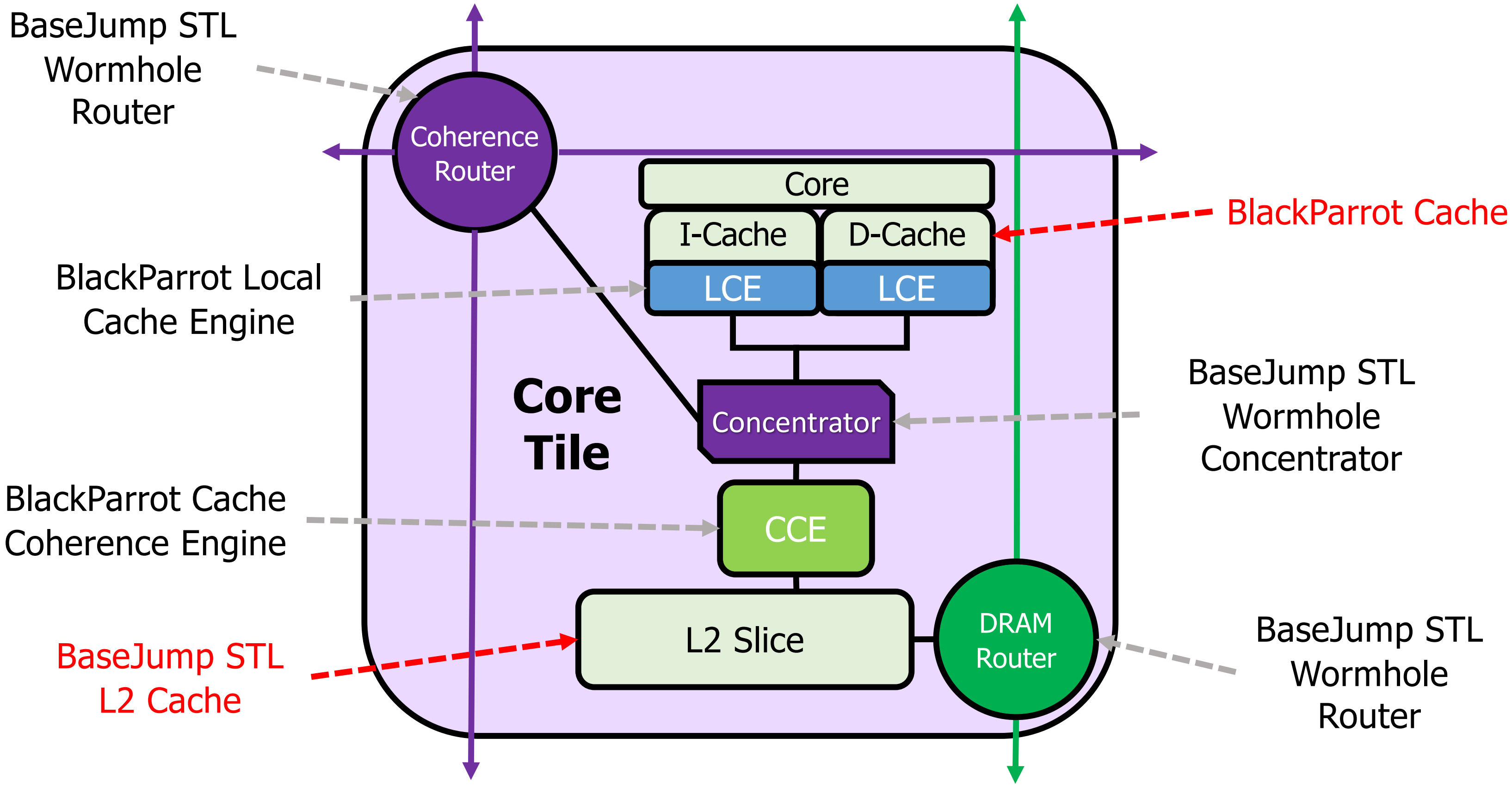
BlackParrot Local Cache Engine

BlackParrot Cache Coherence Engine

BaseJump STL Wormhole Concentrator

BaseJump STL Wormhole Router

Assemble Tiles Out of Reusable Components



BaseJump STL
Wormhole
Router

Coherence Router

Core

I-Cache

D-Cache

BlackParrot Cache

LCE

LCE

BlackParrot Local
Cache Engine

Core Tile

Concentrator

BaseJump STL
Wormhole
Concentrator

BlackParrot Cache
Coherence Engine

CCE

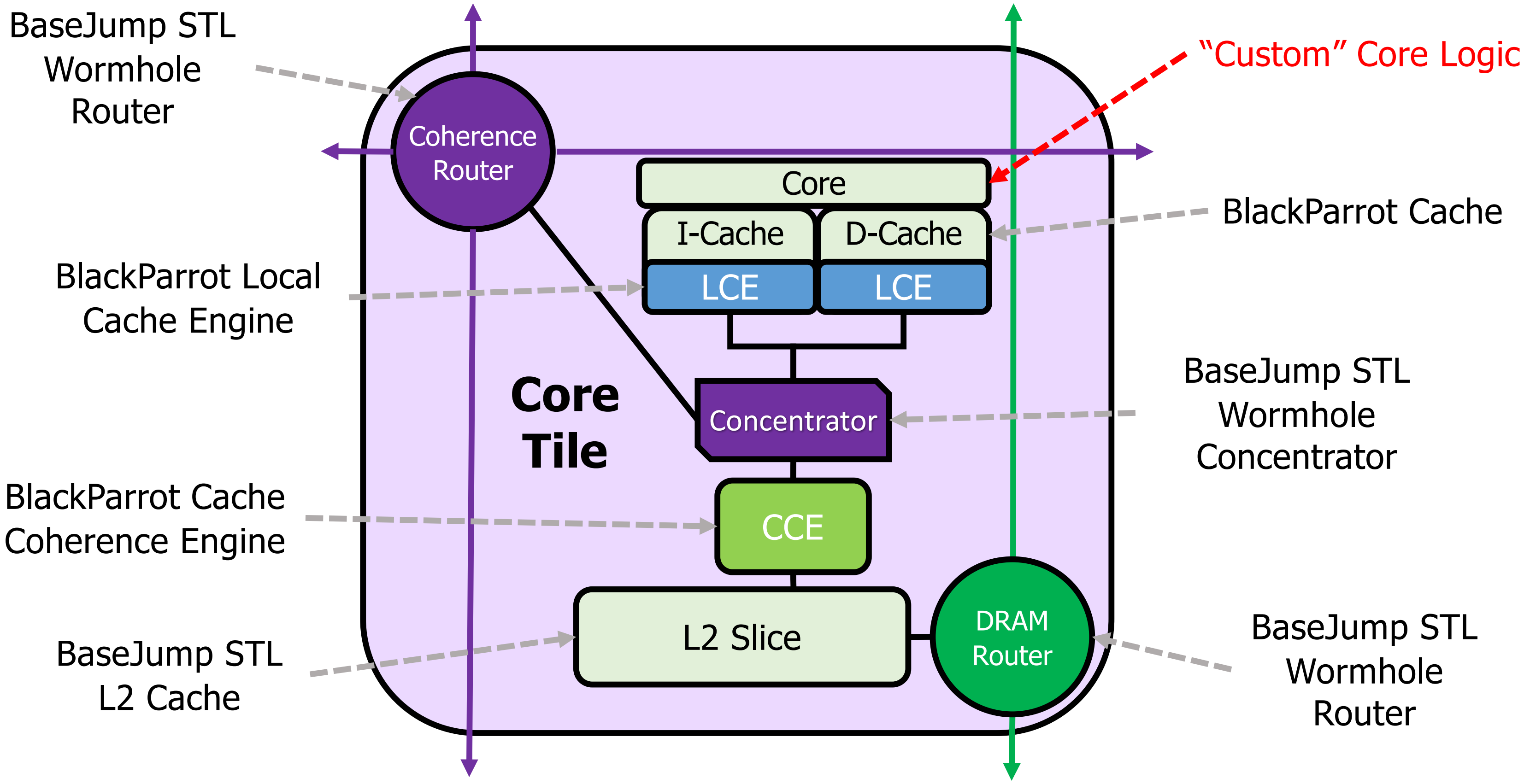
L2 Slice

DRAM Router

BaseJump STL
L2 Cache

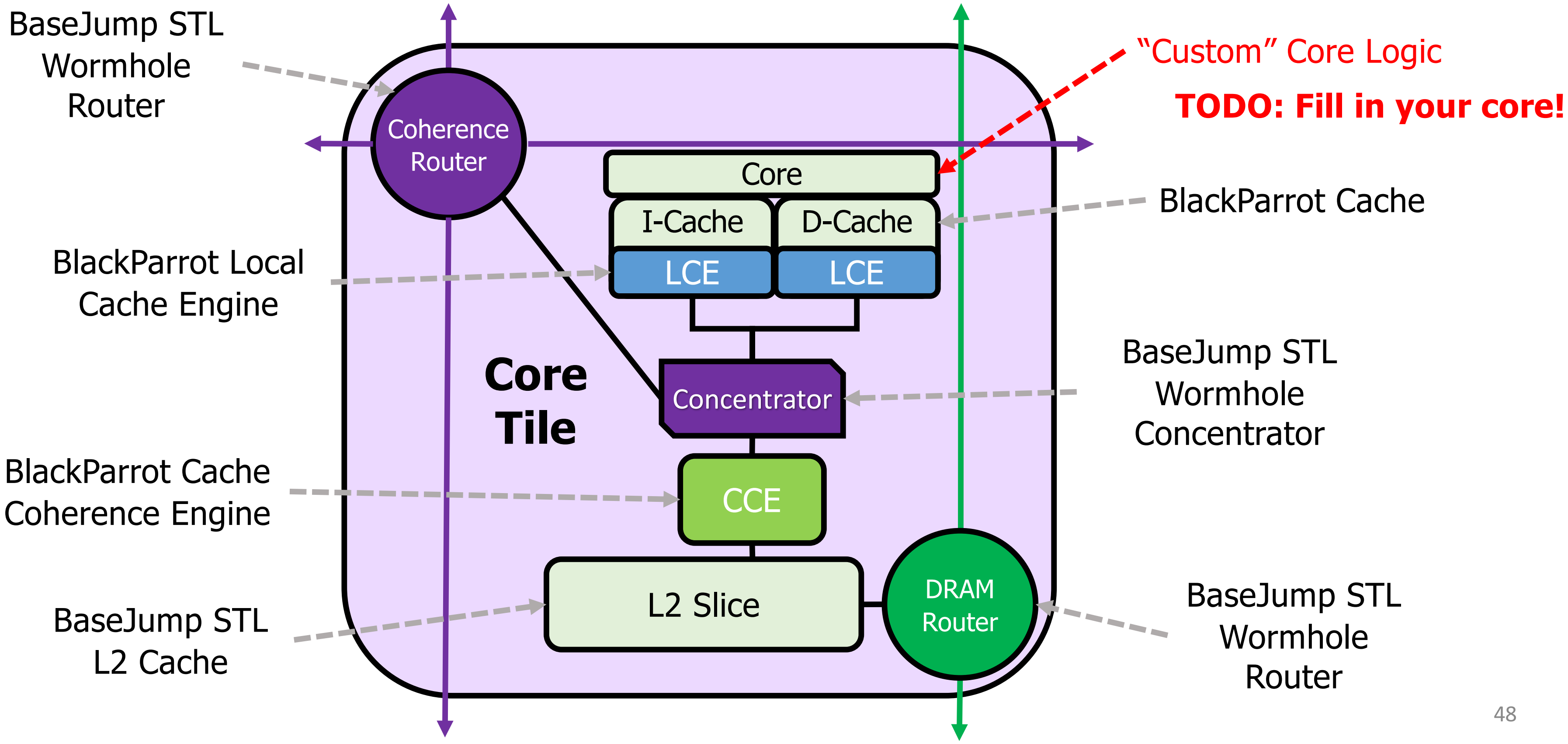
BaseJump STL
Wormhole
Router

Assemble Tiles Out of Reusable Components



BaseJump STL Wormhole Router
Coherence Router
"Custom" Core Logic
Core
I-Cache
D-Cache
BlackParrot Cache
BlackParrot Local Cache Engine
LCE
LCE
Core Tile
Concentrator
BaseJump STL Wormhole Concentrator
BlackParrot Cache Coherence Engine
CCE
L2 Slice
DRAM Router
BaseJump STL L2 Cache
BaseJump STL Wormhole Router

Assemble Tiles Out of Reusable Components



BaseJump STL Wormhole Router

"Custom" Core Logic

TODO: Fill in your core!

BlackParrot Cache

BlackParrot Local Cache Engine

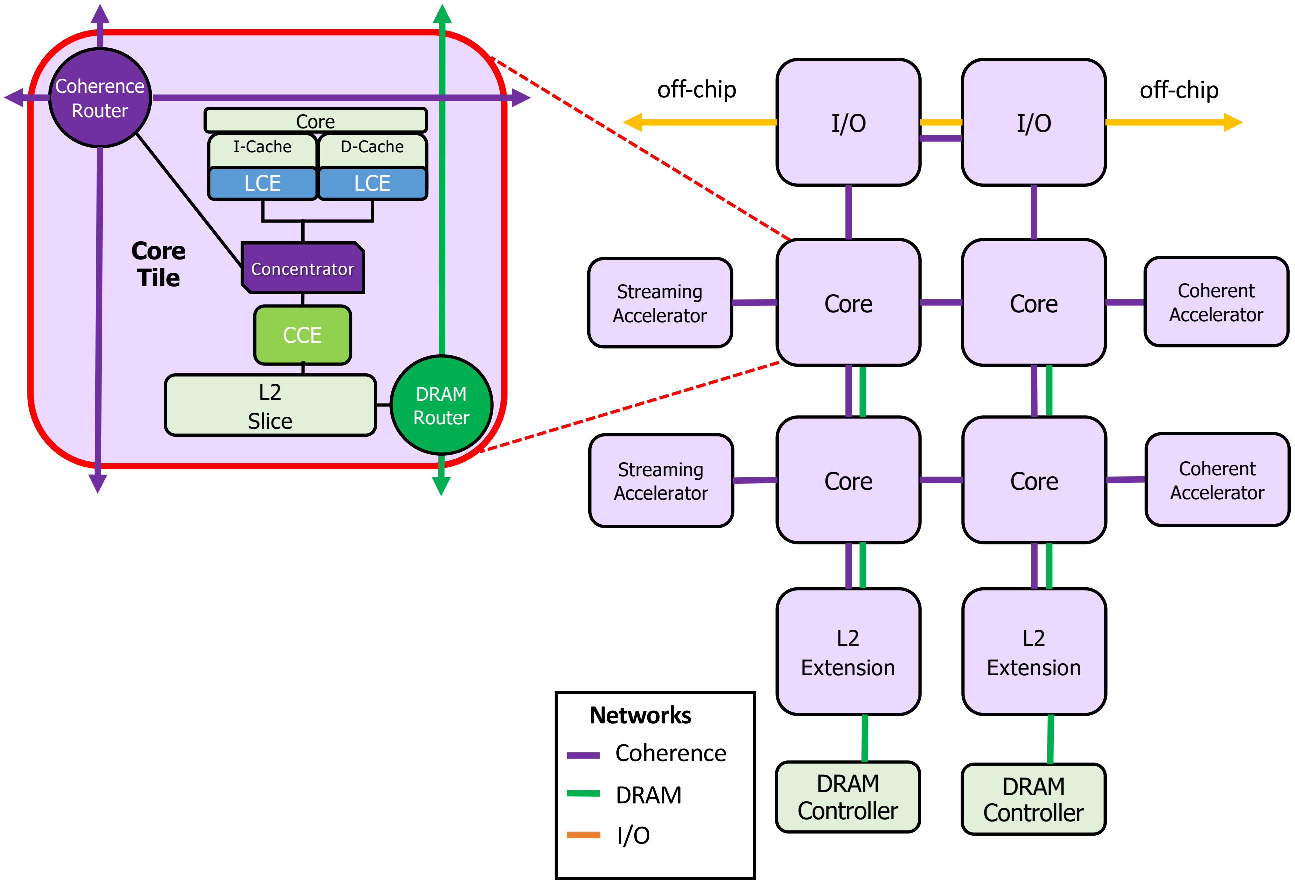
BaseJump STL Wormhole Concentrator

BlackParrot Cache Coherence Engine

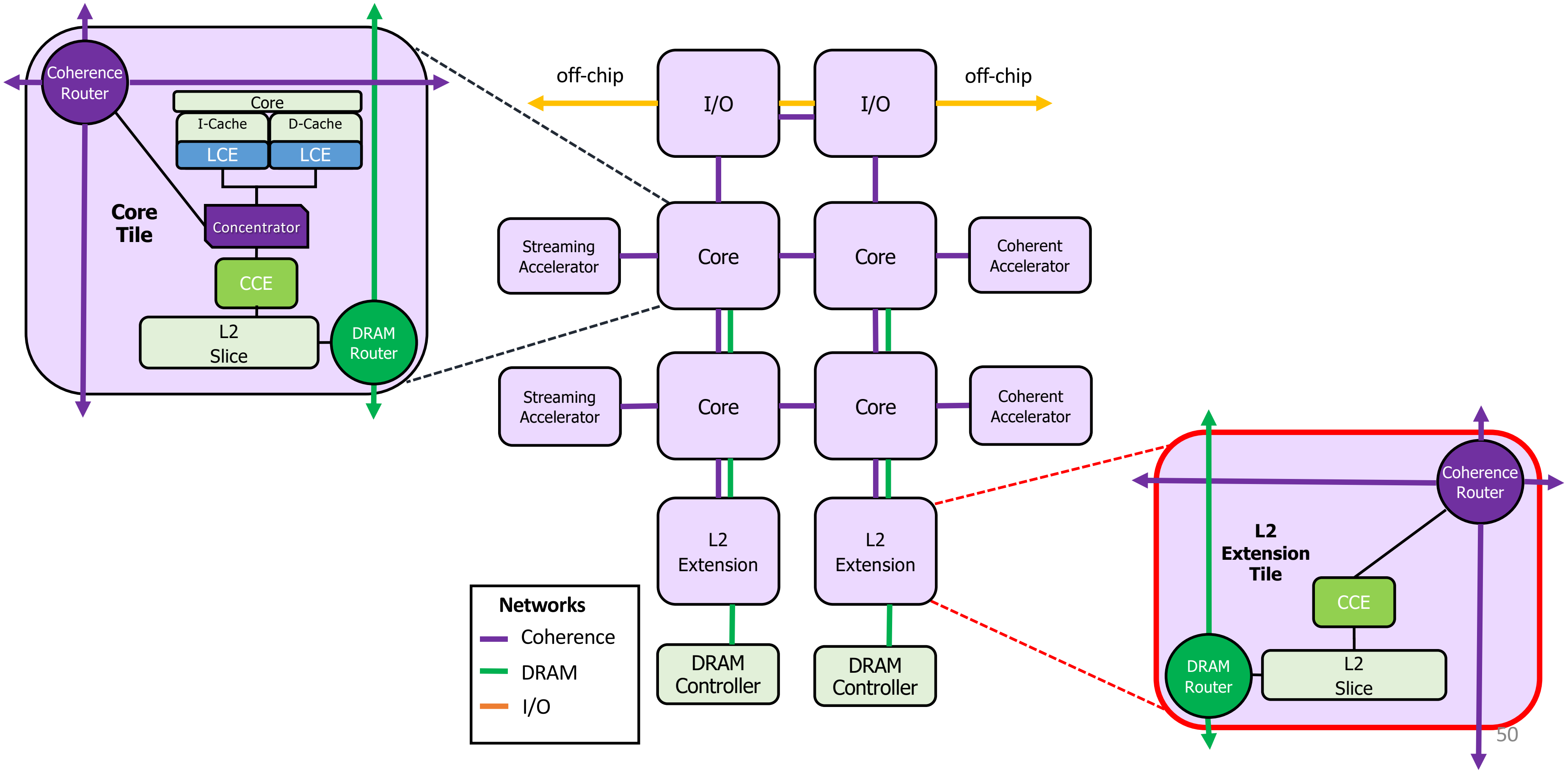
BaseJump STL L2 Cache

BaseJump STL Wormhole Router

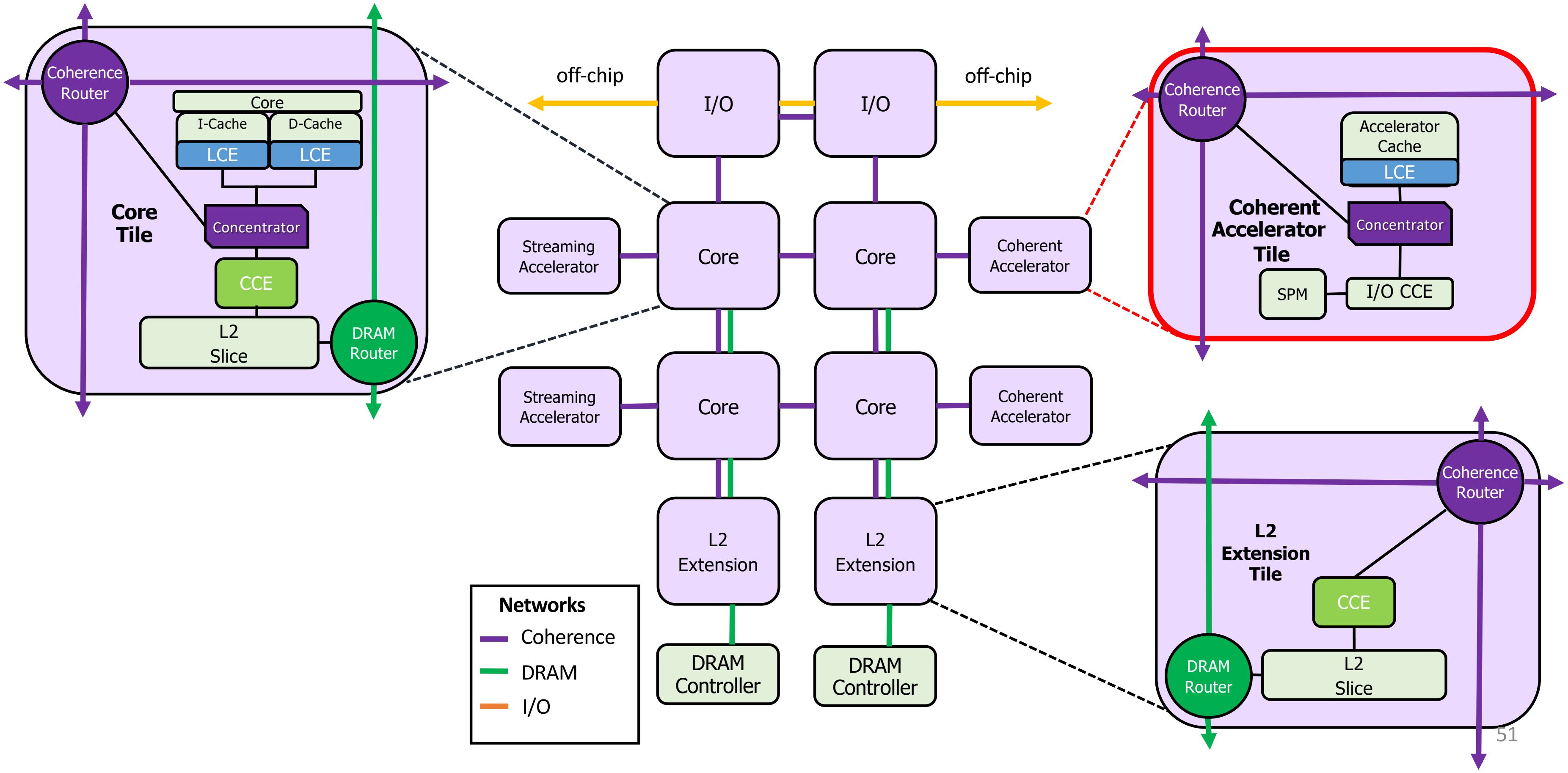
BlackParrot Supports Diverse Tile Types



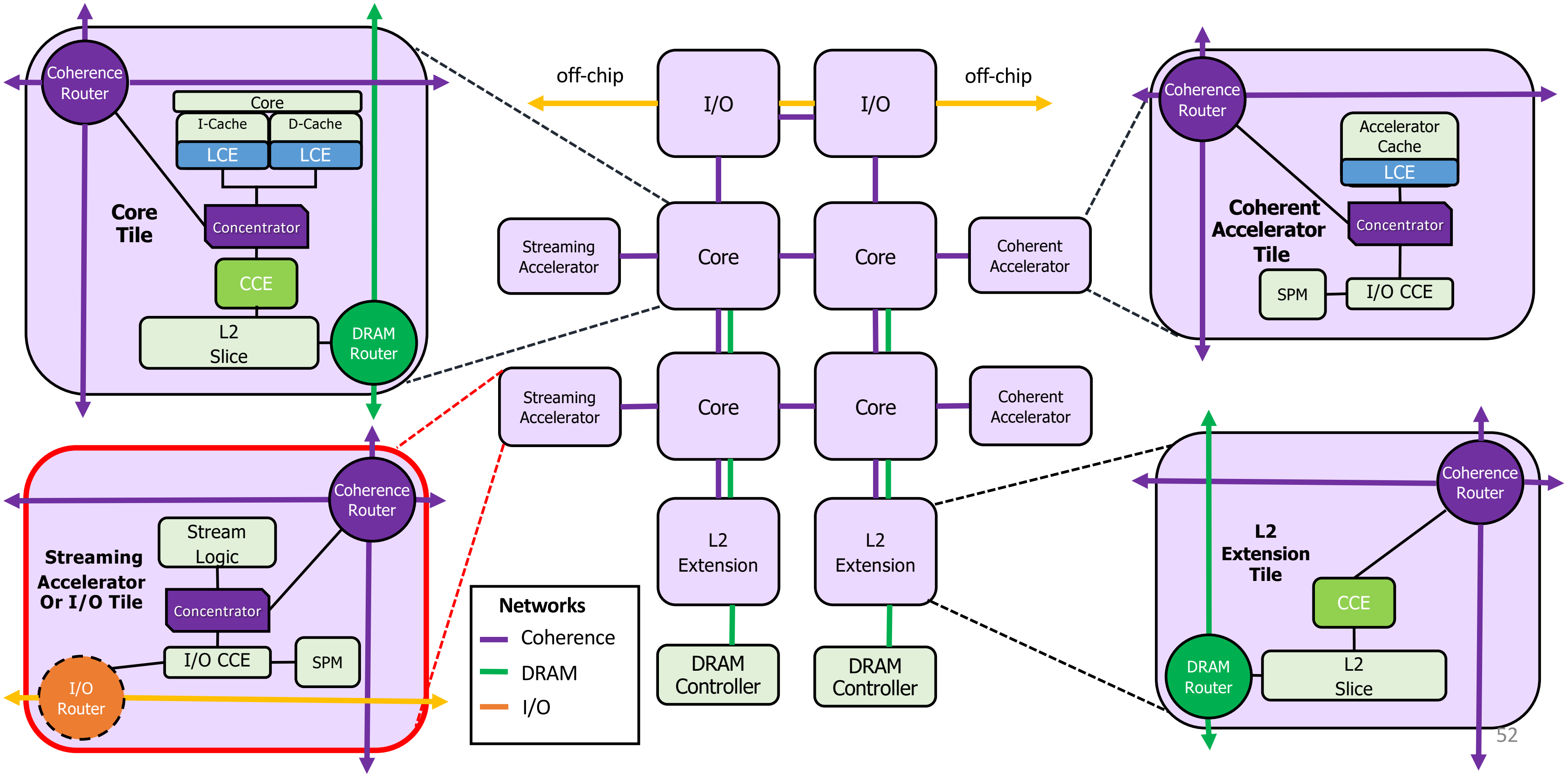
BlackParrot Supports Diverse Tile Types



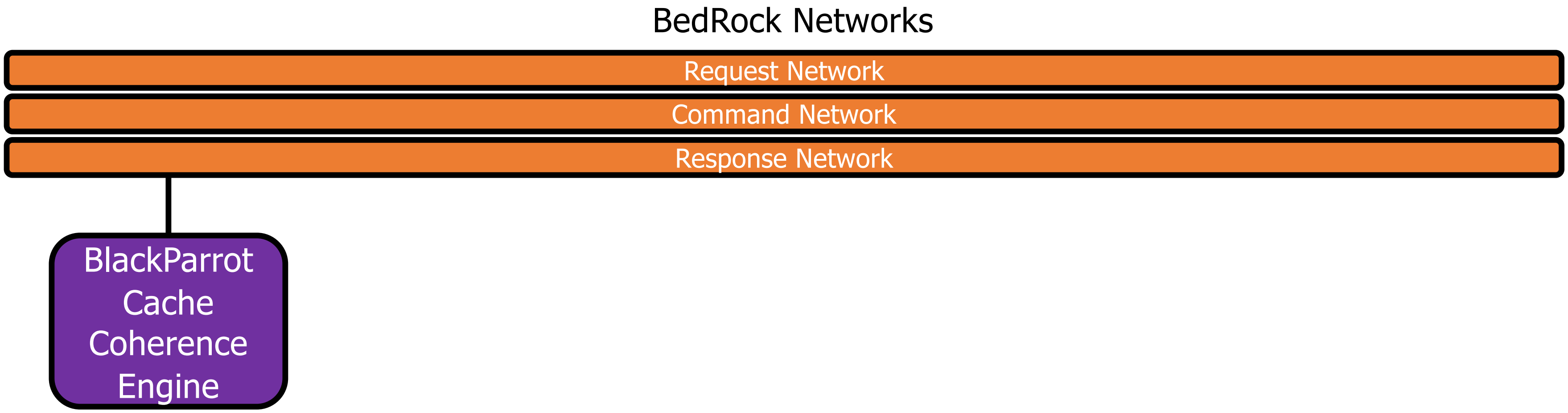
BlackParrot Supports Diverse Tile Types



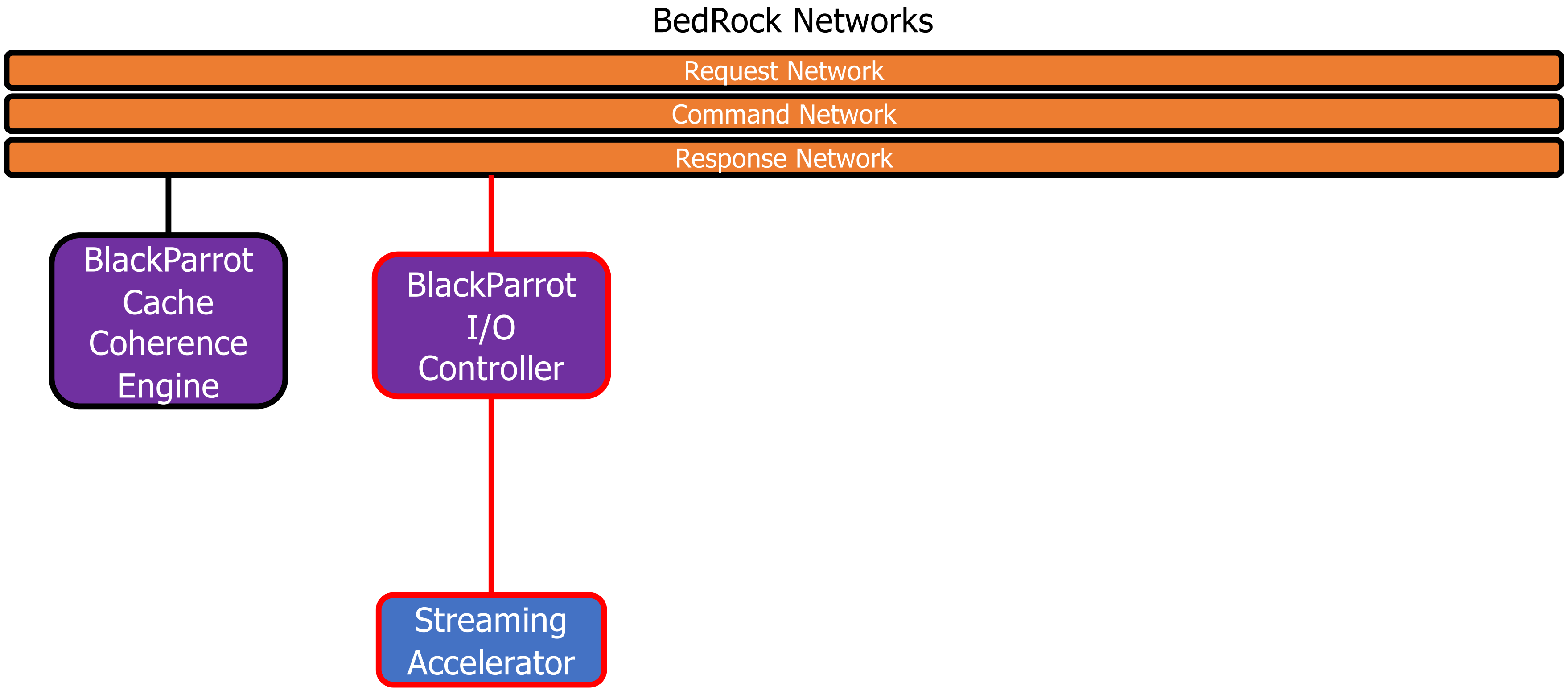
BlackParrot Supports Diverse Tile Types



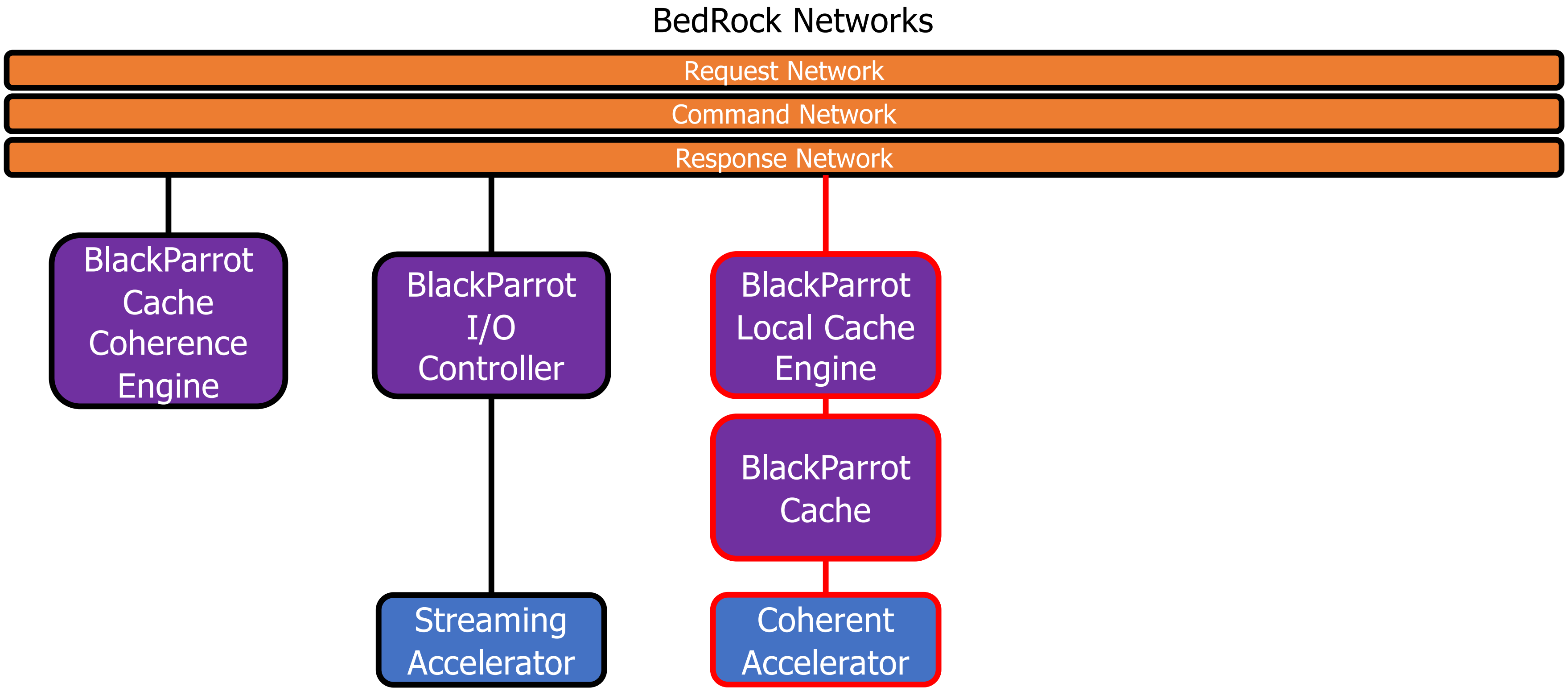
BP Supports Several Accelerator Integration Strategies



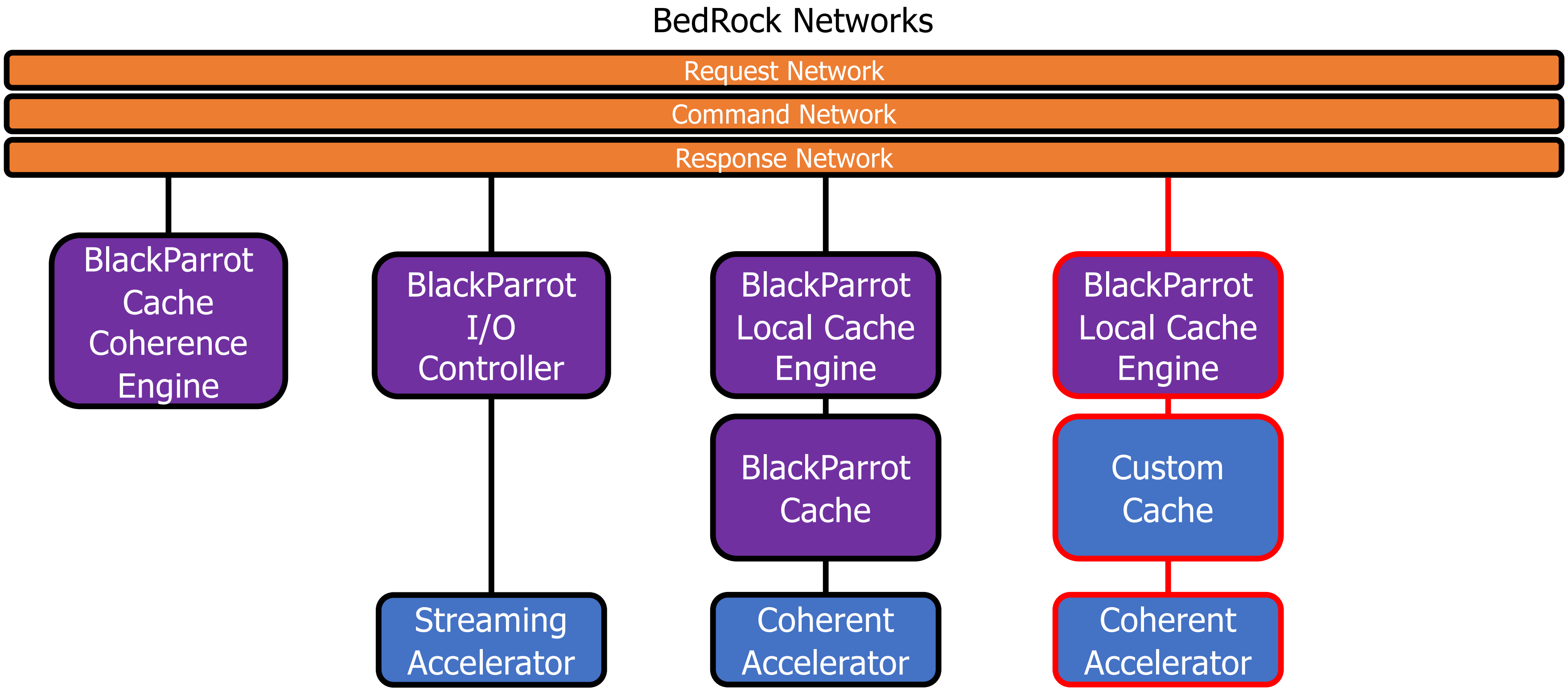
BP Supports Several Accelerator Integration Strategies



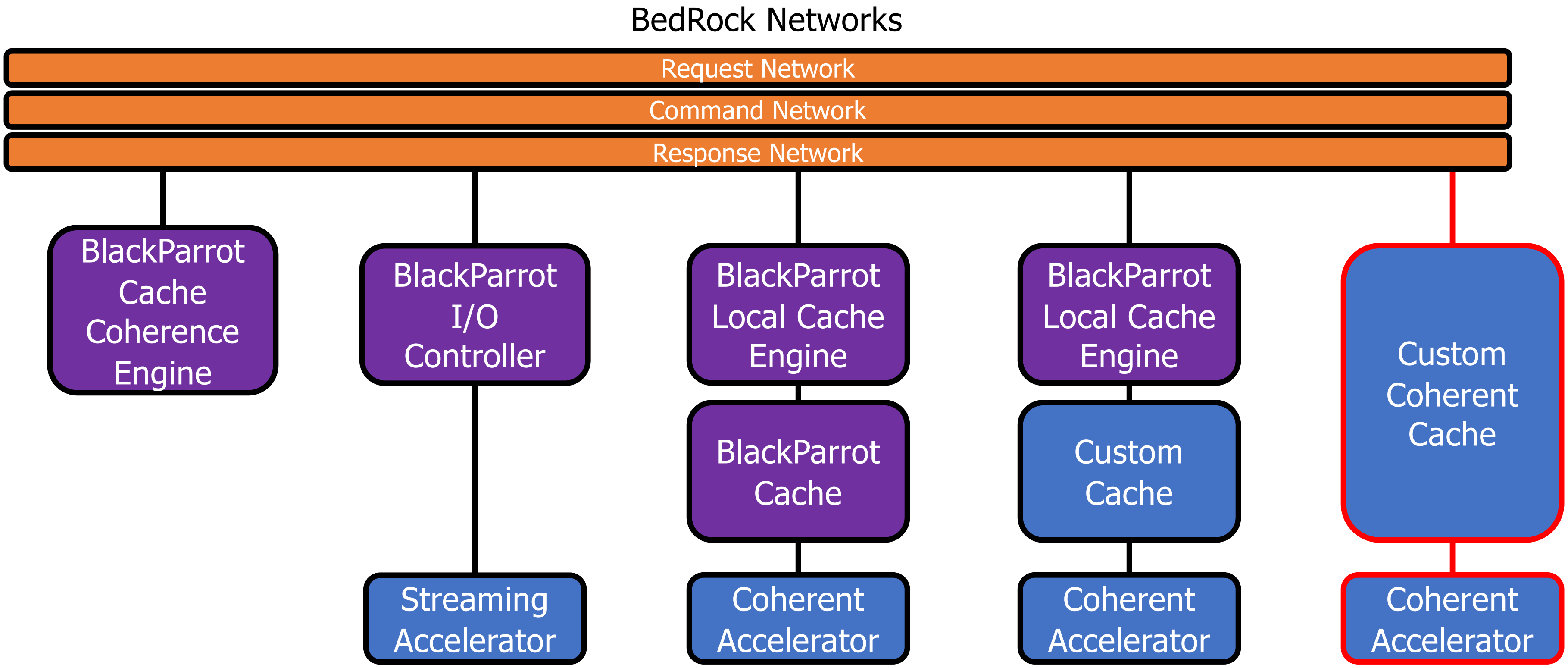
BP Supports Several Accelerator Integration Strategies



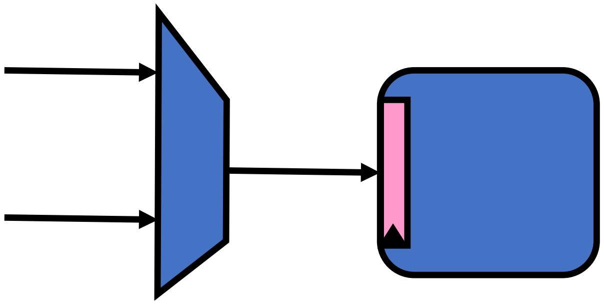
BP Supports Several Accelerator Integration Strategies



BP Supports Several Accelerator Integration Strategies

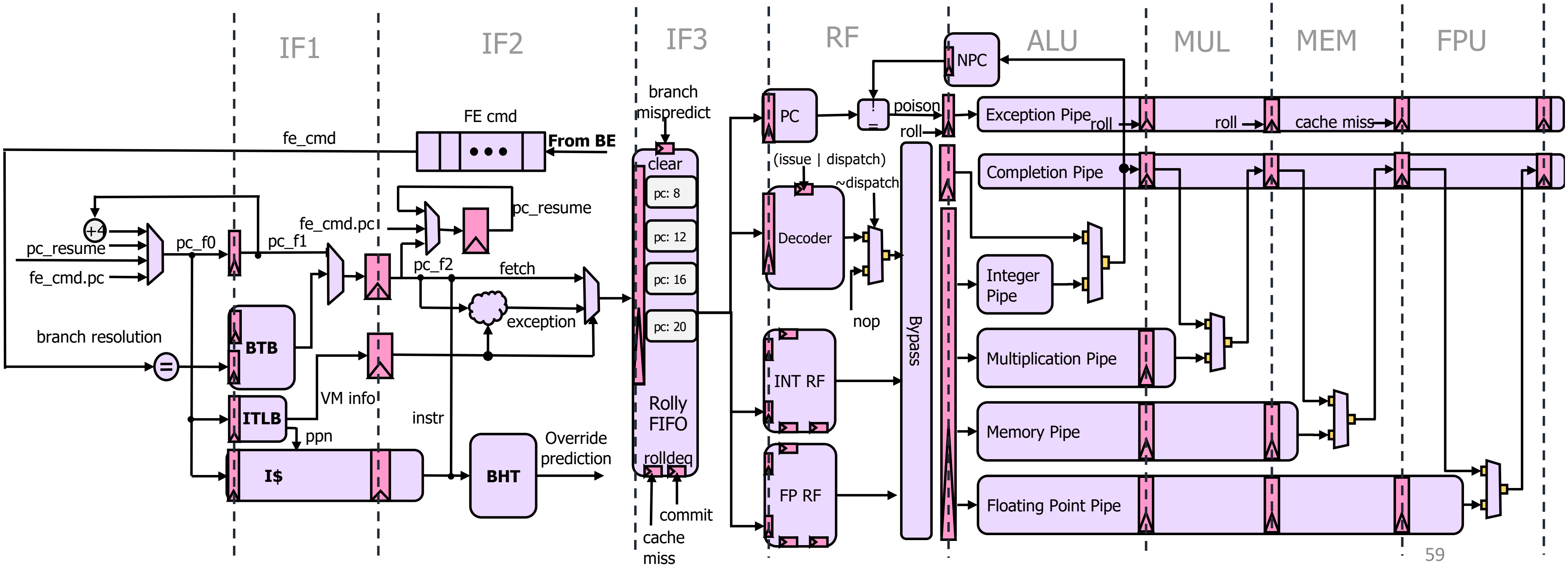


Core Microarchitecture

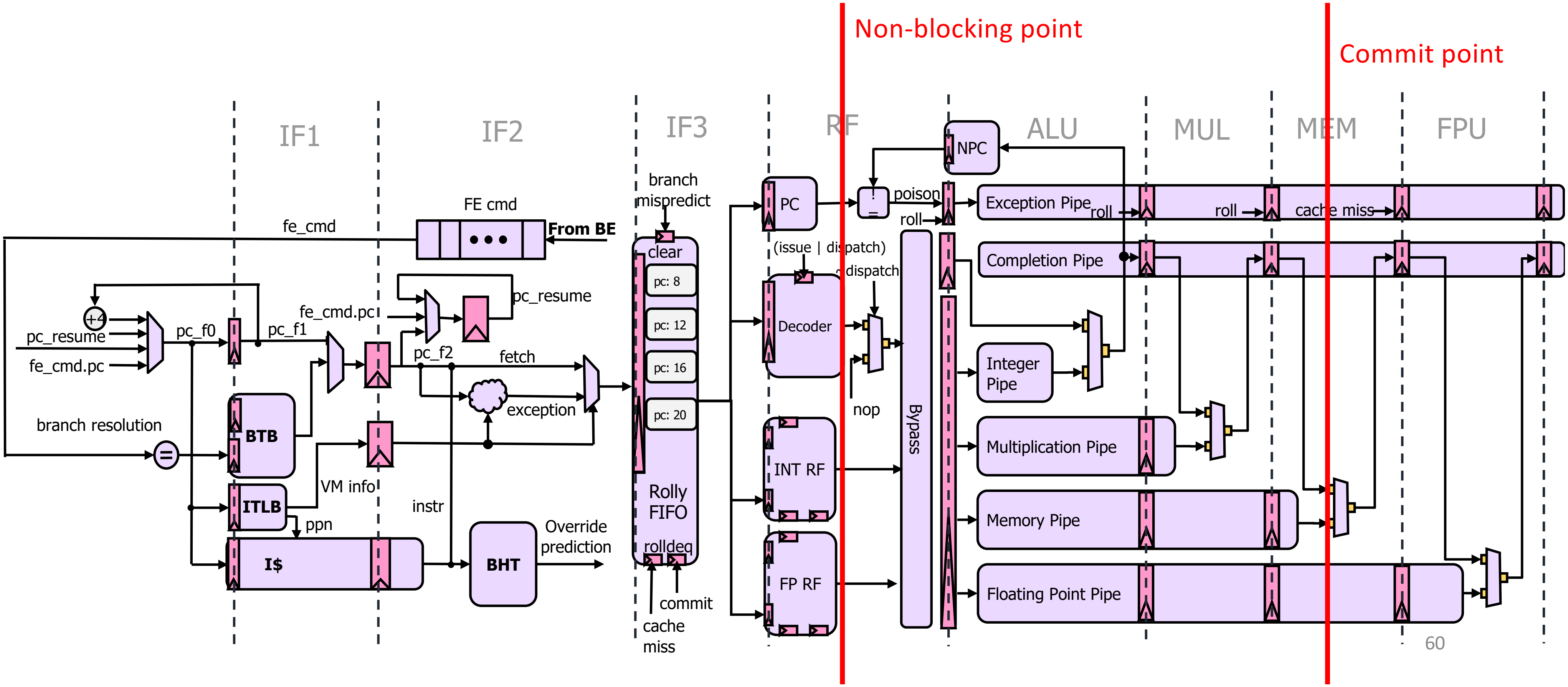


Ultra Efficient, In-order Core Pipeline

Extending the Science of In-Order Core Design



Ultra Efficient, In-order Core Pipeline

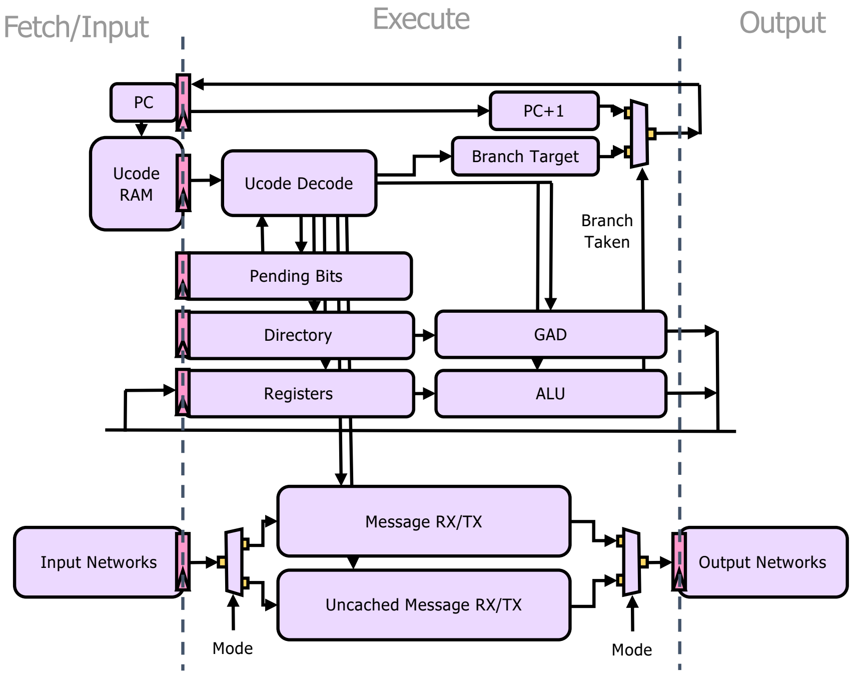


BedRock: A Programmable Cache Coherence Engine

Custom RISC ISA with specialized coherence protocol operations

Coherence logic is programmed at boot time, able to change policies on the fly

Flexibility to add security, debug, or performance monitoring functionality

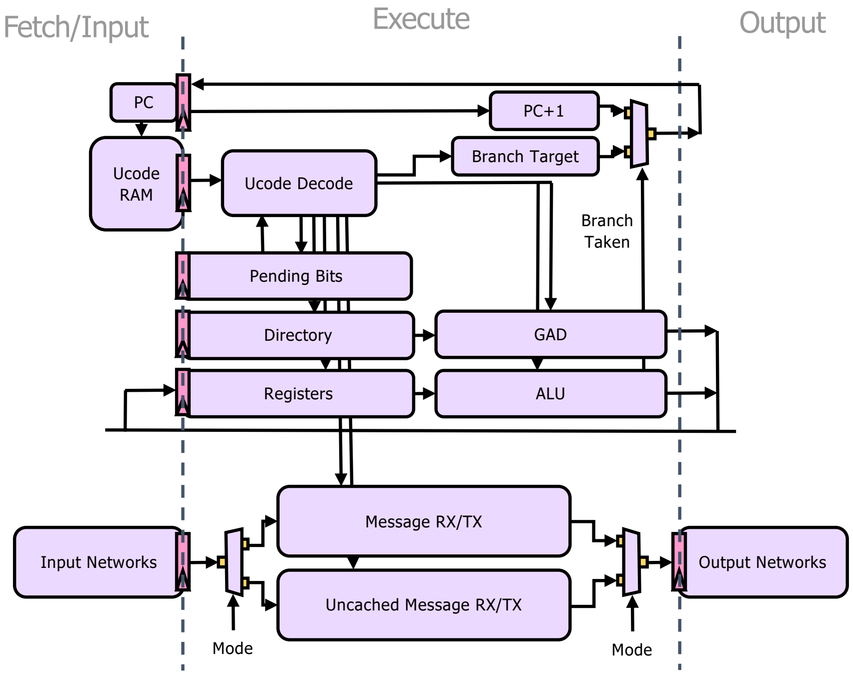


BedRock: A Programmable Cache Coherence Engine

Custom RISC ISA with specialized coherence protocol operations

Coherence logic is programmed at boot time, able to change policies on the fly

Flexibility to add security, debug, or performance monitoring functionality **...post-tapeout!**



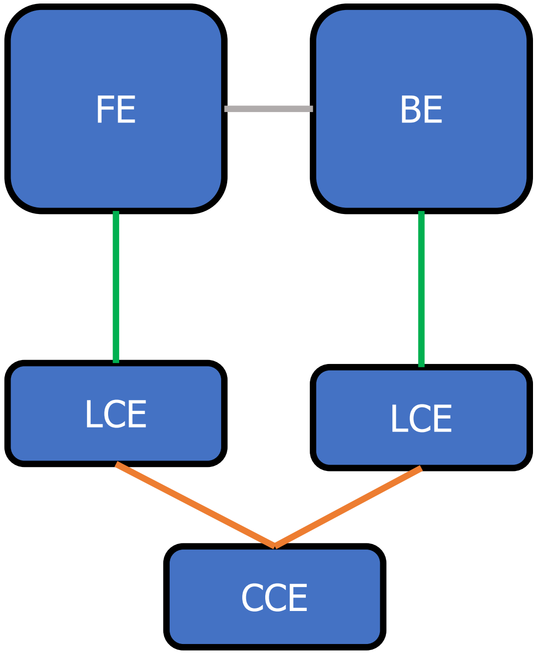
Standardized, Flexible, Latency-Insensitive Interfaces

Borrowing from software, we focus on defining clean and narrow interfaces

Then microarchitectural change only needs to be verified at the module level

- E.g. Adding a new branch predictor only affects the Front End, not the Back End
- Timing paths end at module boundaries

Flexible enough to support various levels of sophistication in implementation without incurring hardware overhead



Case Study: HammerParrot

The HammerBlade Supercomputer For ML/Graphs

"Dense" Machine Learning

"Sparse" ML Graph Embedding

Graph Analytics & Processing

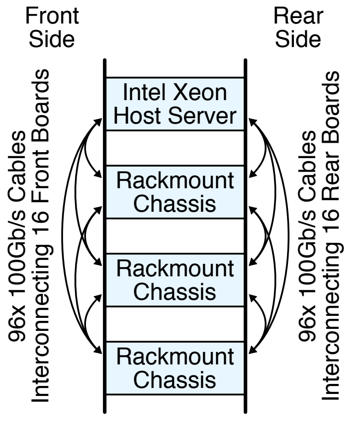
PyTorch

tvm

GraphIt

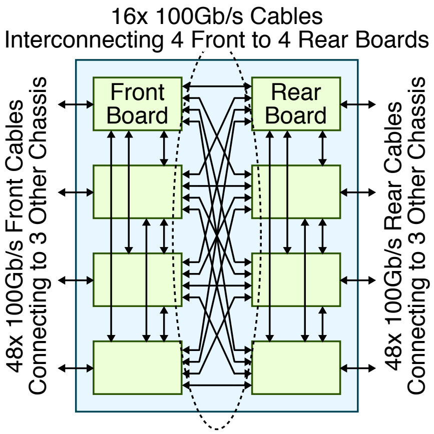
CUDA-Lite and HammerBlade IR

Open Source!



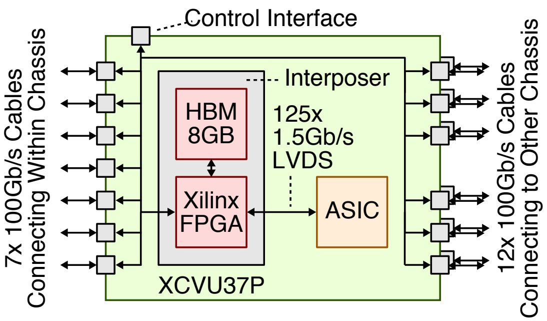
Front Side
Rear Side
Intel Xeon Host Server
Rackmount Chassis
Rackmount Chassis
Rackmount Chassis
96x 100Gb/s Cables Interconnecting 16 Front Boards
96x 100Gb/s Cables Interconnecting 16 Rear Boards

HB Cabinet



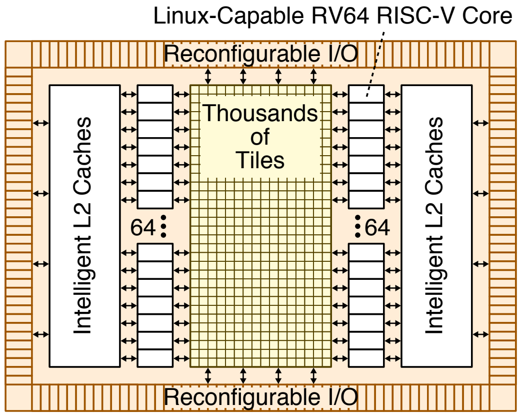
16x 100Gb/s Cables Interconnecting 4 Front to 4 Rear Boards
Front Board
Rear Board
48x 100Gb/s Front Cables Connecting to 3 Other Chassis
48x 100Gb/s Rear Cables Connecting to 3 Other Chassis

HB Chassis



Control Interface
Interposer
HBM 8GB
125x 1.5Gb/s LVDS
Xilinx FPGA
ASIC
XCVU37P
7x 100Gb/s Cables Connecting Within Chassis
12x 100Gb/s Cables Connecting to Other Chassis

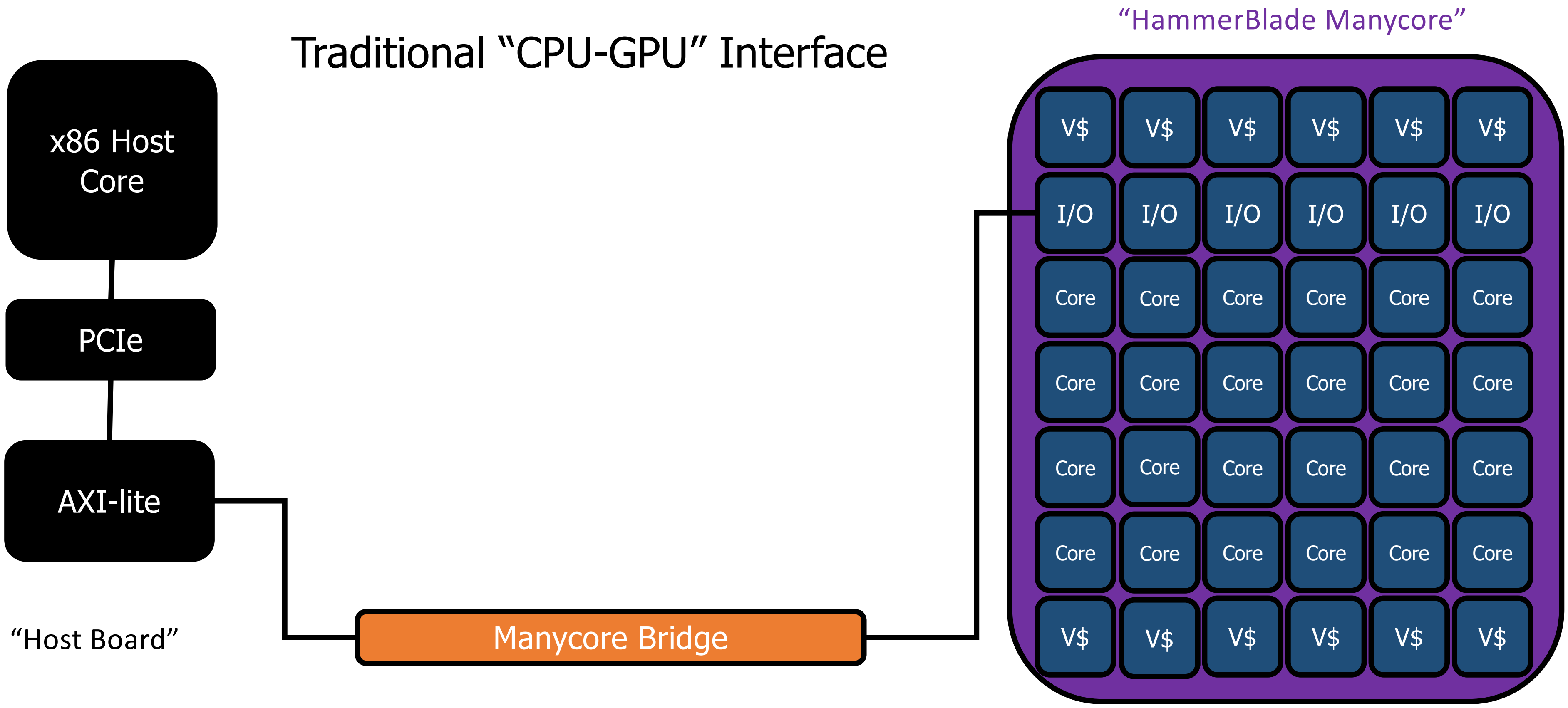
HB Board



HB ASIC

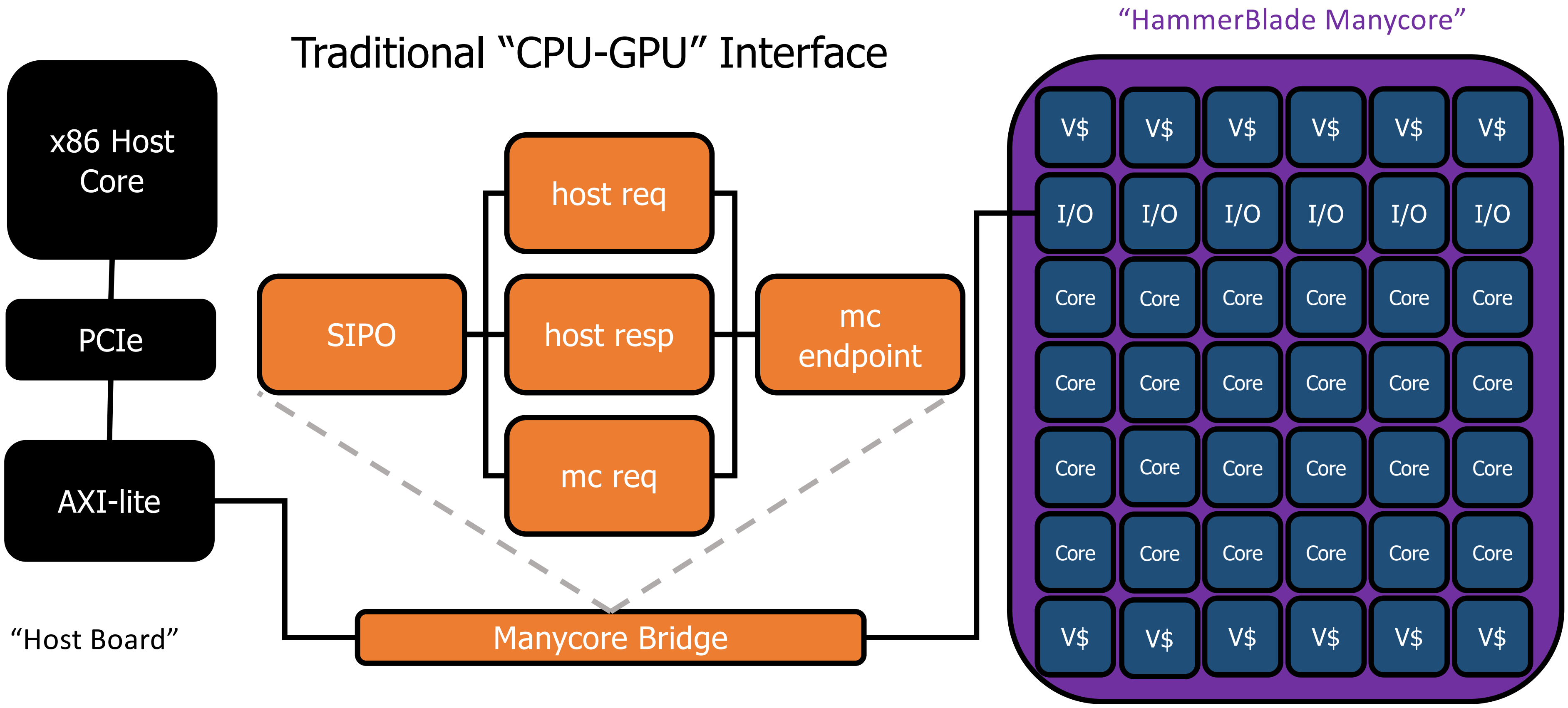
Prototyping Realities Inform Design Decisions

Traditional "CPU-GPU" Interface



V$	V$	V$	V$	V$	V$
I/O	I/O	I/O	I/O	I/O	I/O
Core	Core	Core	Core	Core	Core
Core	Core	Core	Core	Core	Core
Core	Core	Core	Core	Core	Core
Core	Core	Core	Core	Core	Core
V$	V$	V$	V$	V$	V$

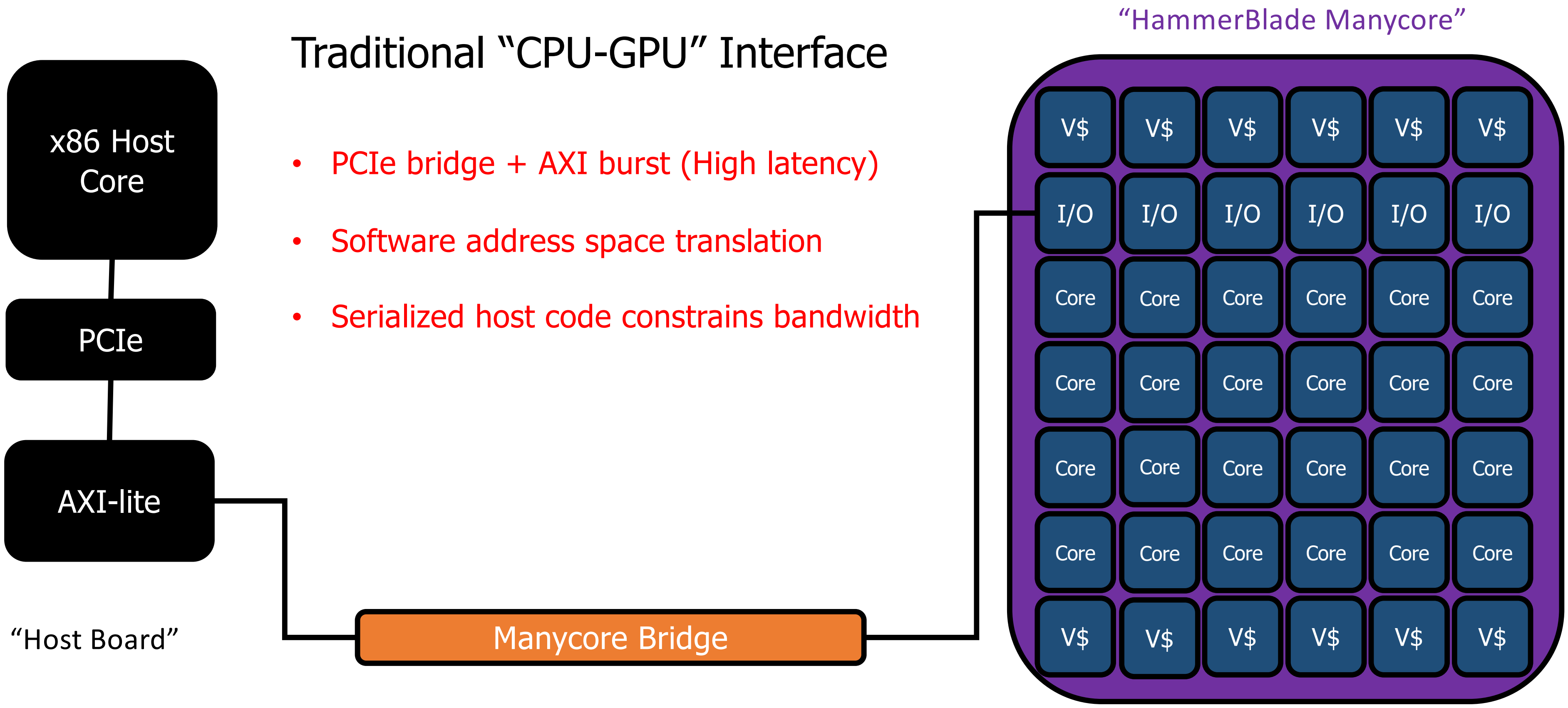
Prototyping Realities Inform Design Decisions



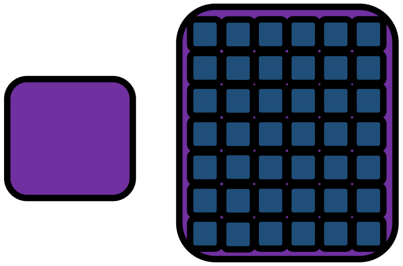
Prototyping Realities Inform Design Decisions

Traditional "CPU-GPU" Interface

- PCIe bridge + AXI burst (High latency)
- Software address space translation
- Serialized host code constrains bandwidth

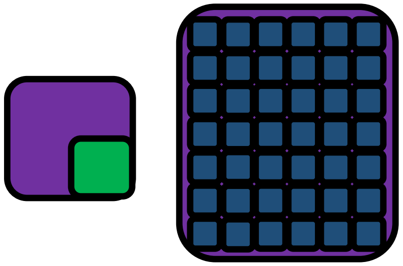


BlackParrot Enables Flexible Integration



BlackParrot Enables Flexible Integration

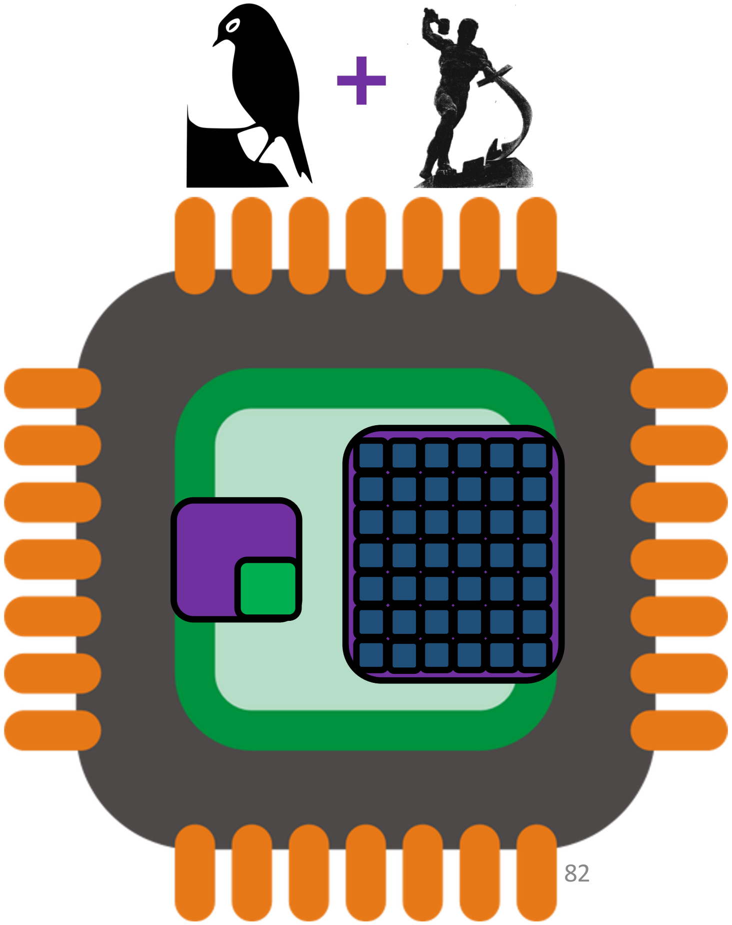
Open-source design, fully customizable core



BlackParrot Enables Flexible Integration

Open-source design, fully customizable core

Shares die area with manycore

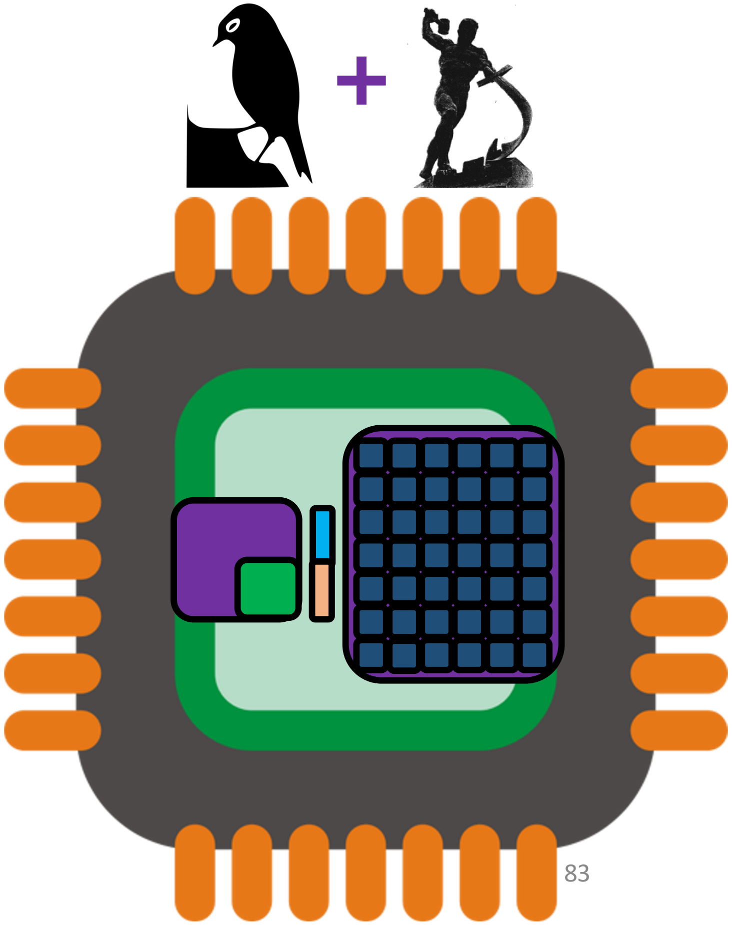


BlackParrot Enables Flexible Integration

Open-source design, fully customizable core

Shares die area with manycore

Can add hardware blocks close to core



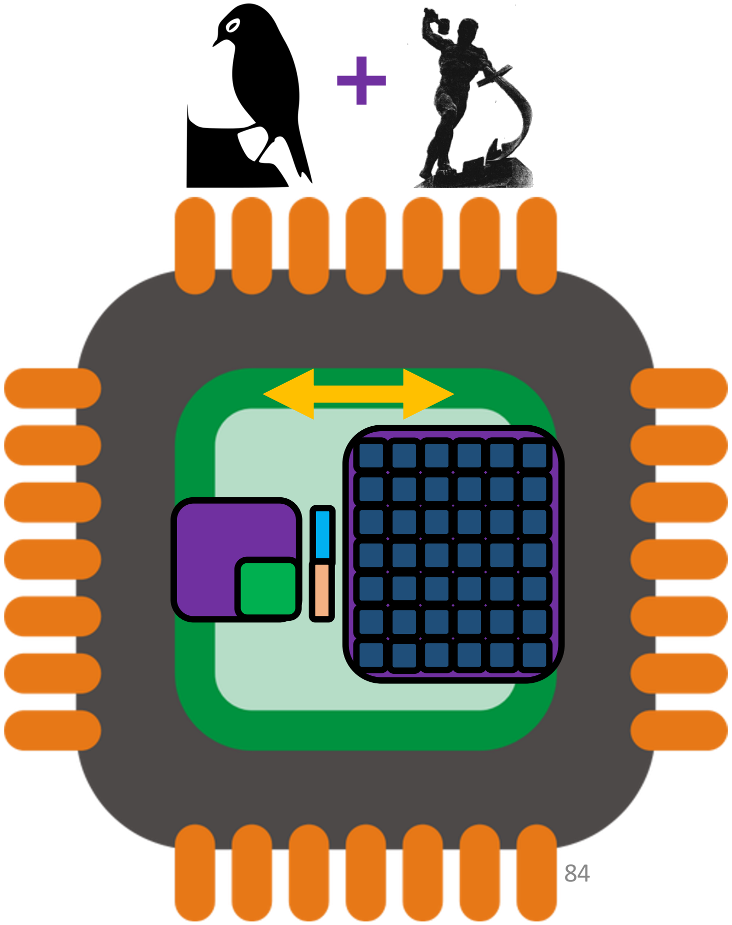
BlackParrot Enables Flexible Integration

Open-source design, fully customizable core

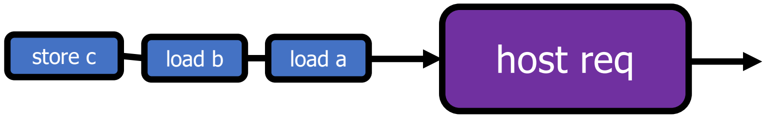
Shares die area with manycore

Can add hardware blocks close to core

Result: Tiny, efficient BlackParrot cores can coordinate peak system performance!

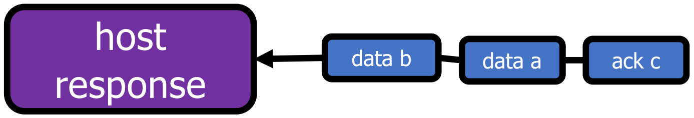


Hardware Accelerated Software API



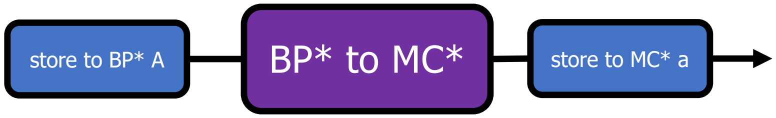
Request type 1: Load/Store Packet

- Initiated by non-blocking store from BP->MC
- Embedded in a 64-bit packet sent over BedRock Network



Request type 2: Load Response

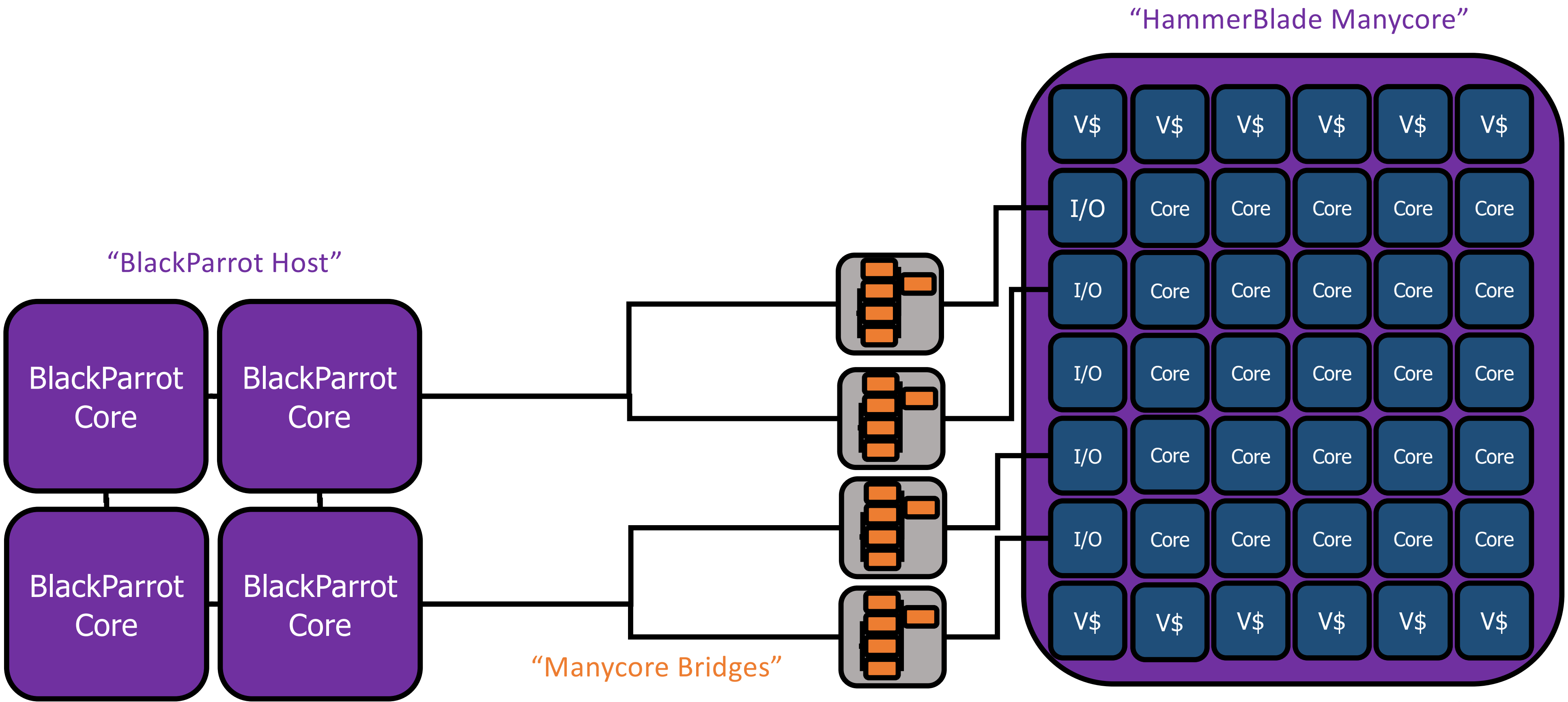
- Initiated by blocking load from BP->MC
- Comprises two 64-bit loads sent over BedRock Network



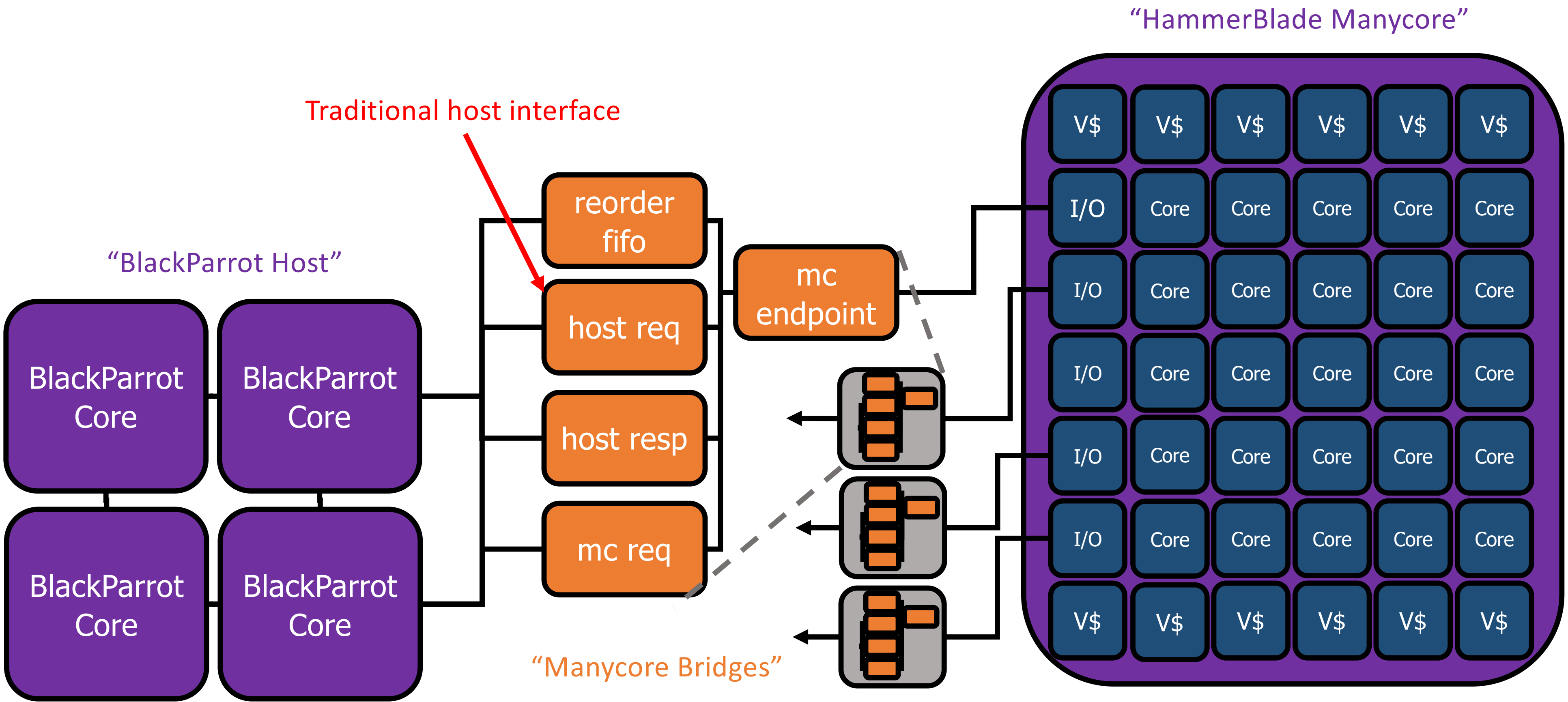
Request type 3: Direct MMIO

- Memory-mapped uncached range
- BP loads/stores directly converted to manycore addresses
- 32-bit address embedded in 40-bit BP physical address space

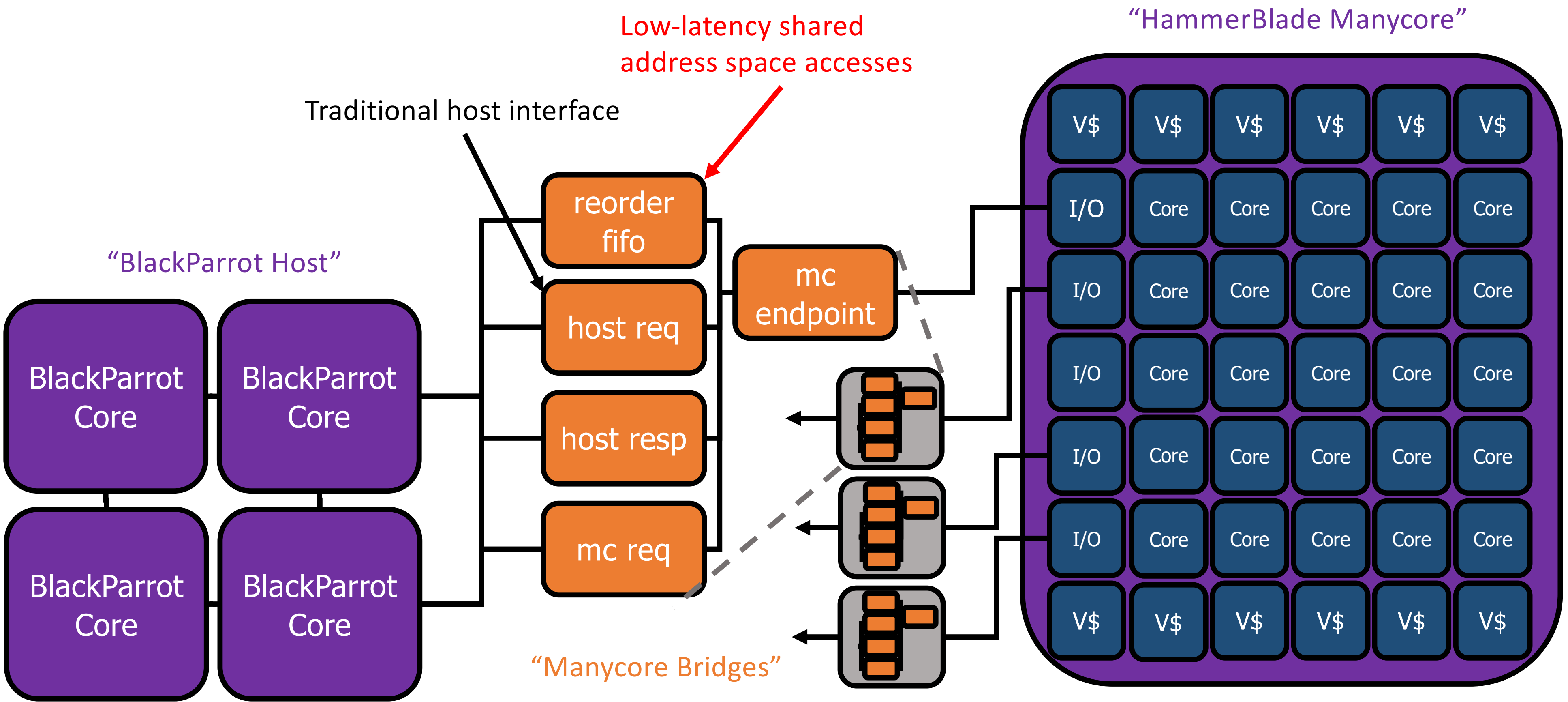
HammerParrot: Scalable Multi-Manycore Communication



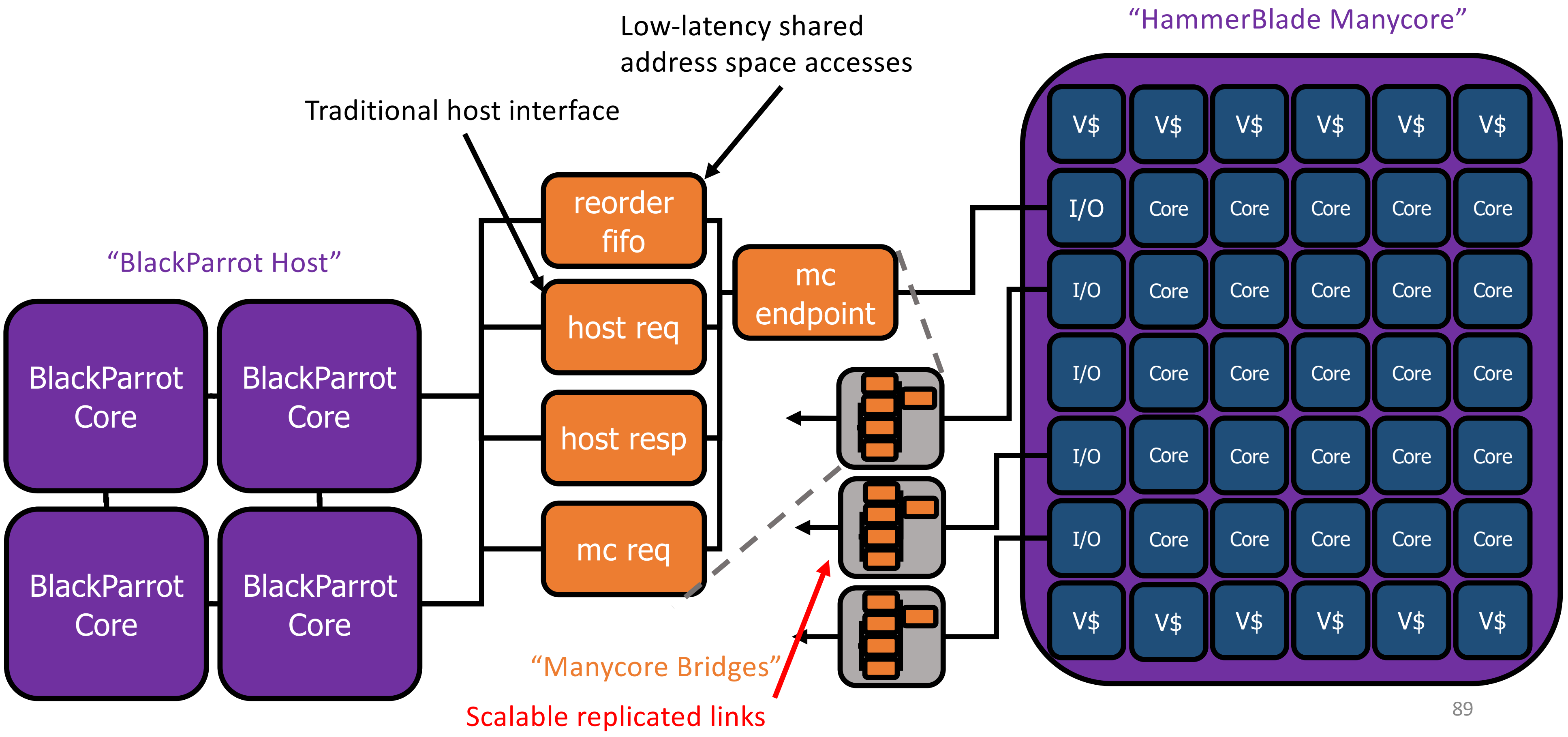
HammerParrot: Scalable Multi-Manycore Communication



HammerParrot: Scalable Multi-Manycore Communication



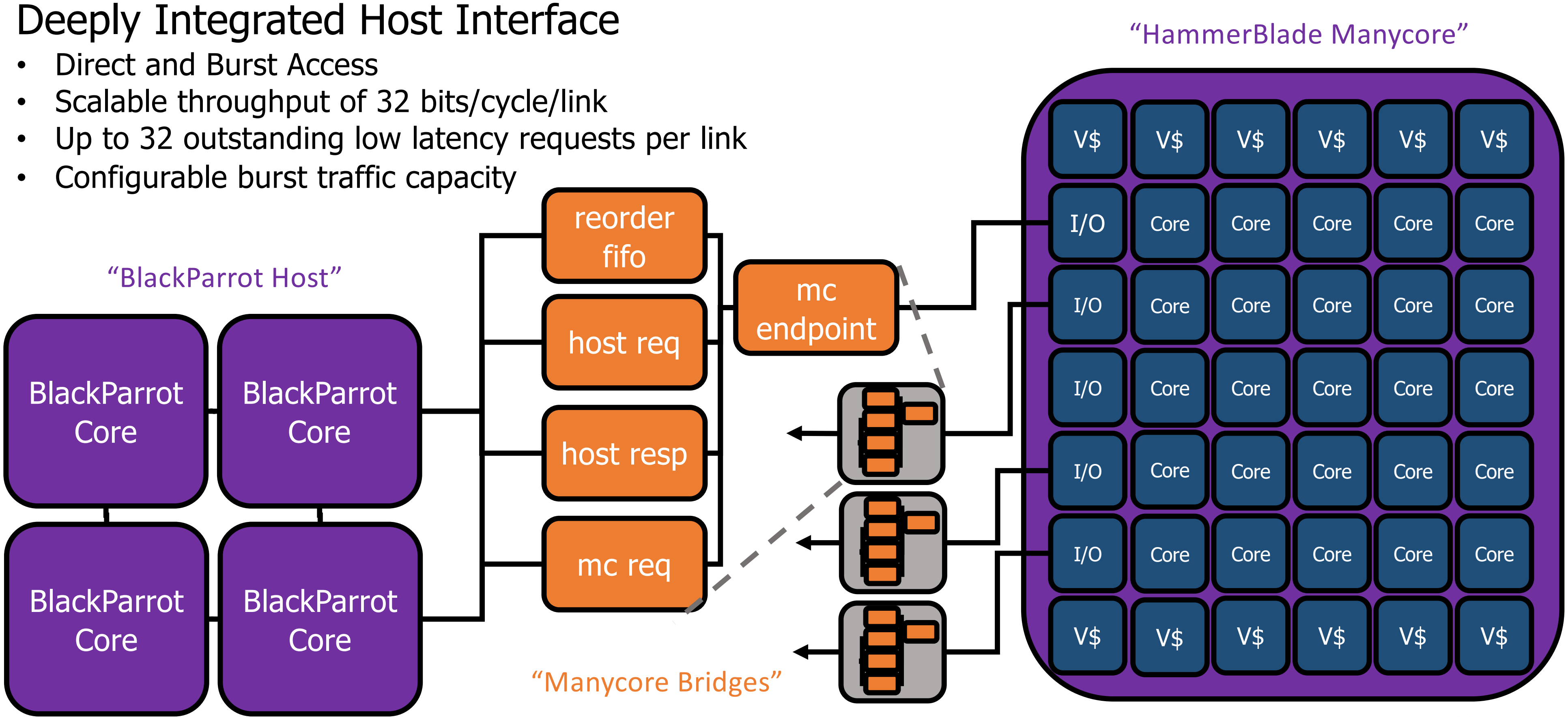
HammerParrot: Scalable Multi-Manycore Communication



HammerParrot: Scalable Multi-Manycore Communication

Deeply Integrated Host Interface

- Direct and Burst Access
- Scalable throughput of 32 bits/cycle/link
- Up to 32 outstanding low latency requests per link
- Configurable burst traffic capacity



BlackParrot Community

BlackParrot: Community Driven Microarchitecture

Encourage users to become developers

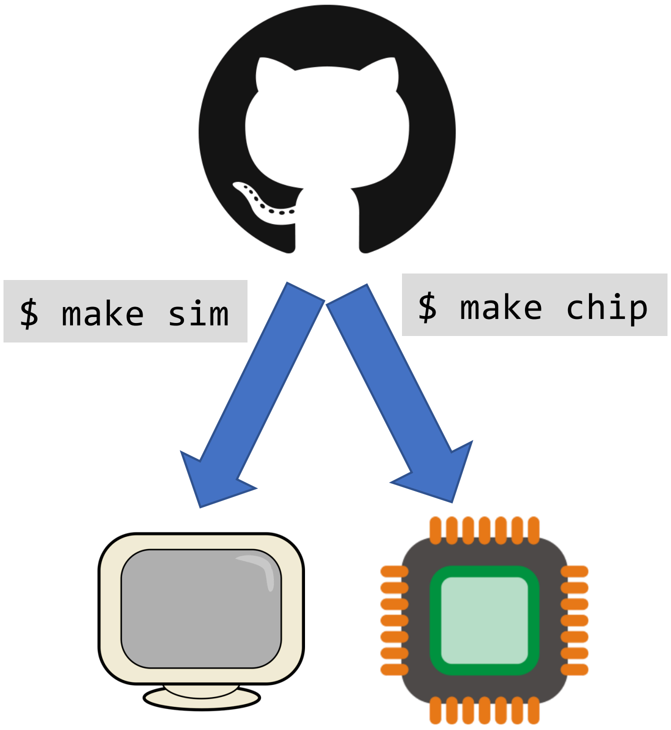
- Able to extend the system with cursory understanding of architecture
- Prioritize clarity over optimization

Build infrastructure with community in mind

- Open-source toolchain (Verilator, OpenROAD) support
- Strive for infrastructure agnosticism

Focus on out-of-box experience

- Bootstrap environments with minimal dependencies
- GitHub->Simulation->FPGA/ASIC in a handful of commands



```
$ make sim
```

```
$ make chip
```

BlackParrot is FPGA-Validated

BlackParrot has been synthesized and tested on Xilinx Artix-7, and Kintex Ultrascale+ and Virtex Ultrascale+ FPGAs

BlackParrot runs Linux on the LiteX open-source FPGA environment

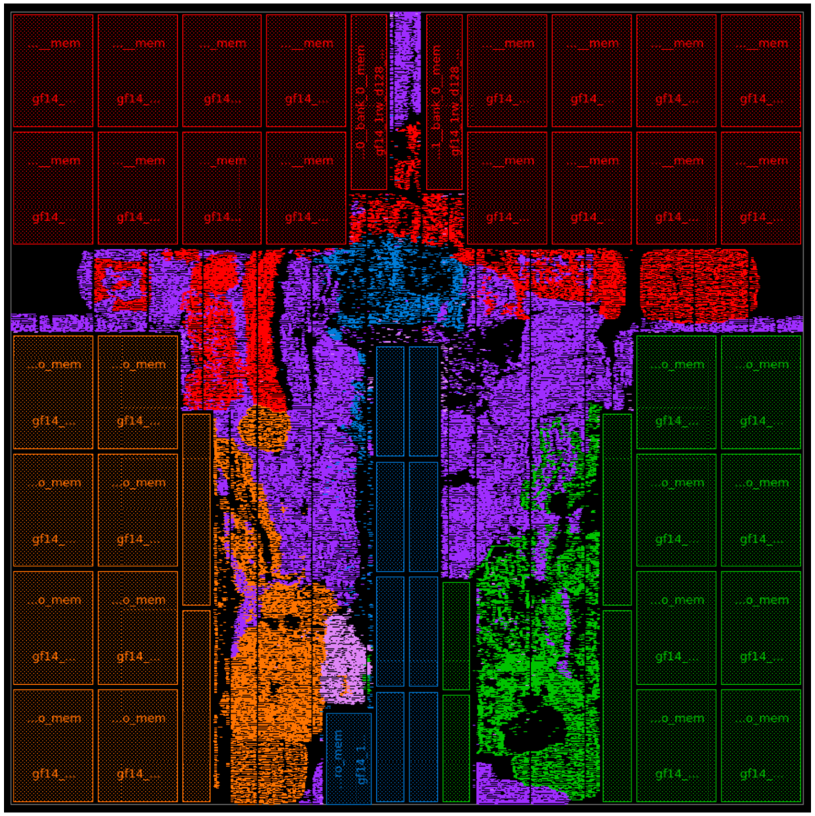
Build your hardware, easily!

BlackParrot is Silicon-Validated

A 4-core BlackParrot was taped out in GlobalFoundries 12nm in July'19

Silicon is back and running in lab since April '20!

BlackParrot has also been ported to TSMC40 and FreePDK45 nodes

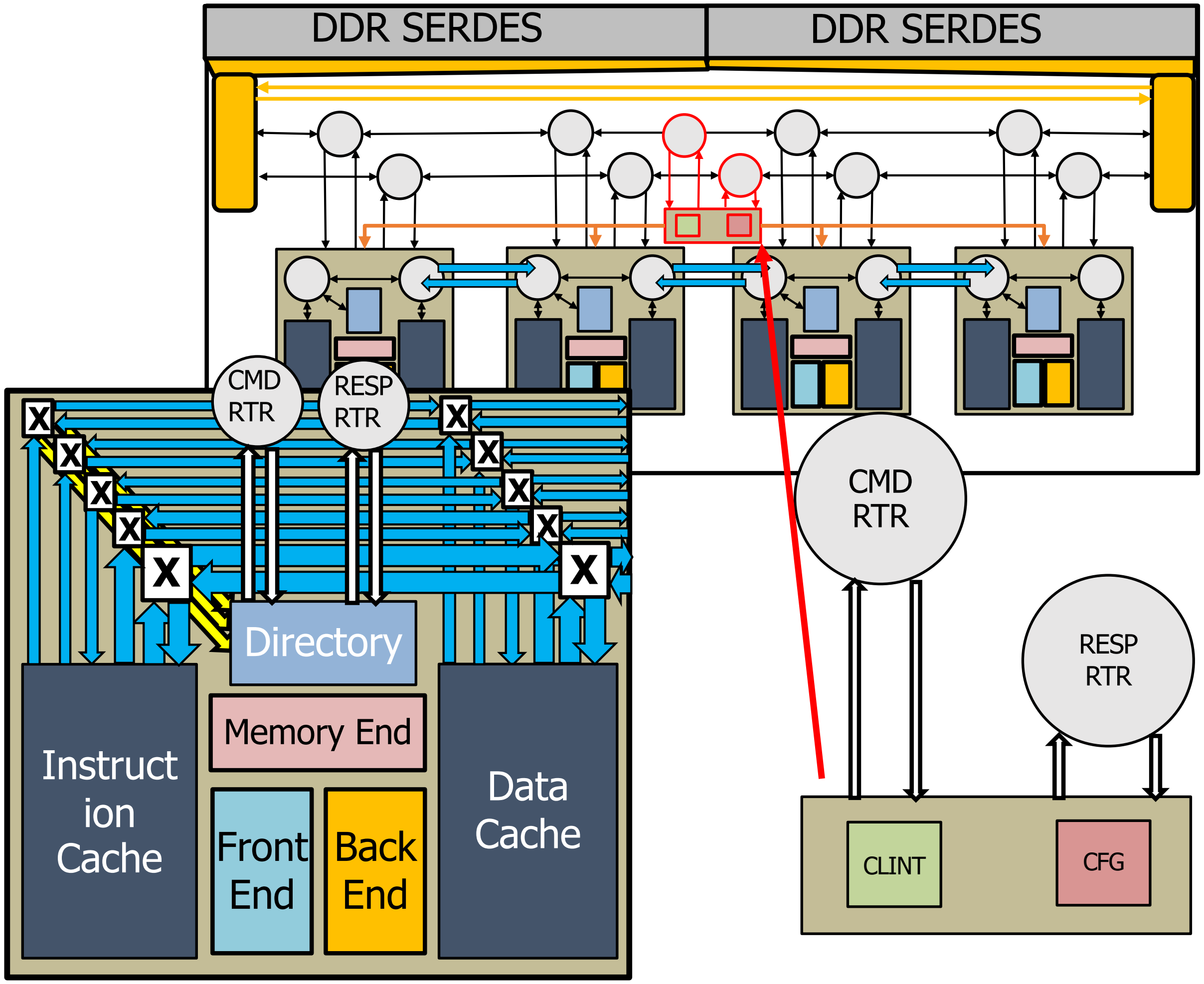


BlackParrot One ASIC

Taped out July 13, 2019!
GF 12nm Process Technology
3mm x 3mm die

4-core 64b RISC-V multicore
CLINT Interrupt Controller
Off-chip SERDES for DRAM and I/O
Chips can be chained

Each core:
- RV64IA with Virtual Memory
- Single-issue In-order
- 32K Data cache
- 32K Instruction cache
- 64-entry BTB
- 8-entry DTLB
- 8-entry ITLB



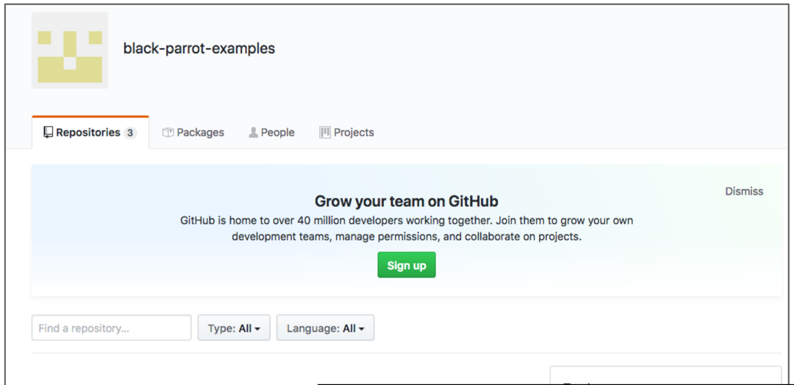
Fostering Community Around BlackParrot Backend Flows

Open-source Tapeout Directories

- All BlackParrot tapeouts/FPGA environments collected at https://github.com/black-parrot-examples
- Both BSG + external (with permission)
- Share common SoC/infrastructure modules

OpenROAD + bsg_fakeram + FreePDK45

- Brand new open-source push-button CAD flow
- Cacti-based predictive SRAM generator
- Predictive 45nm PDK with click-through license



black-parrot-examples

Repositories 3 | Packages | People | Projects

Grow your team on GitHub

GitHub is home to over 40 million developers working together. Join them to grow your own development teams, manage permissions, and collaborate on projects.

bsg_scratch_designs

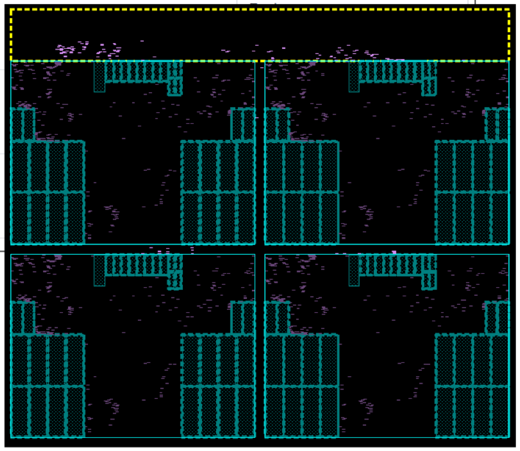
● Tcl ⑂0 ★0 ⓘ1 ⇅0 Updated 25 days ago

bsg_tapeout_v1_tsmc40

● Tcl ⑂0 ★1 ⓘ0 ⇅0 Updated 29 days ago

bsg_tapeout_v0_gf12

● Tcl ⑂0 ★1 ⓘ0 ⇅0 Updated on Oct 4



Fostering Community Around BlackParrot Backend Flows

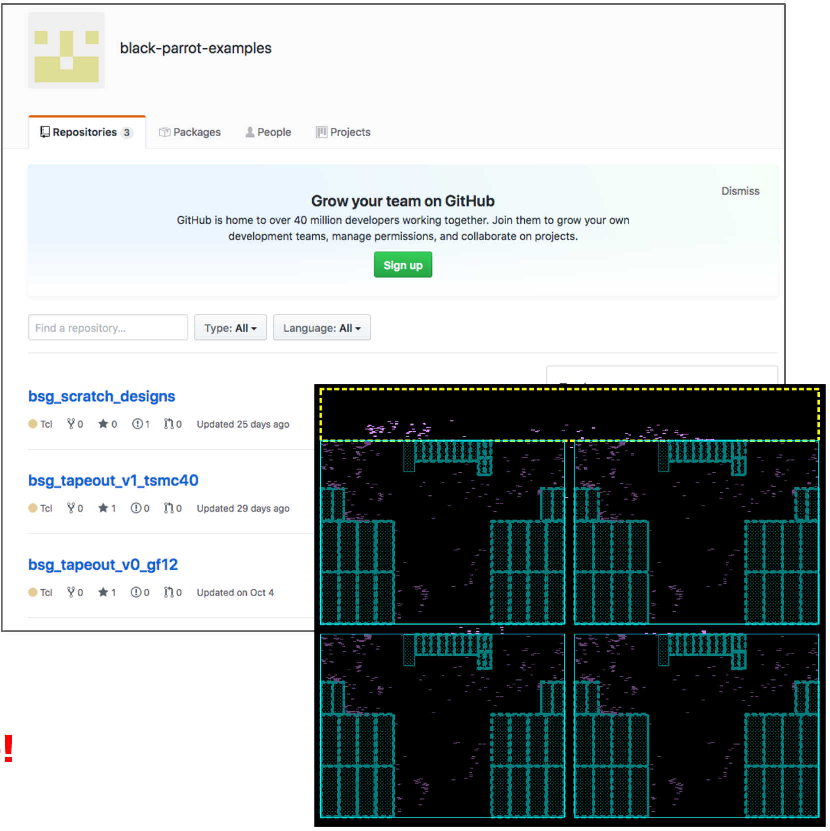
Open-source Tapeout Directories

- All BlackParrot tapeouts/FPGA environments collected at https://github.com/black-parrot-examples
- Both BSG + external (with permission)
- Share common SoC/infrastructure modules

OpenROAD + bsg_fakeram + FreePDK45

- Brand new open-source push-button CAD flow
- Cacti-based predictive SRAM generator
- Predictive 45nm PDK with click-through license

Download today and push through a CAD flow for free!



black-parrot-examples

Grow your team on GitHub

GitHub is home to over 40 million developers working together. Join them to grow your own development teams, manage permissions, and collaborate on projects.

bsg_scratch_designs

Tcl 0 ★ 0 1 0 Updated 25 days ago

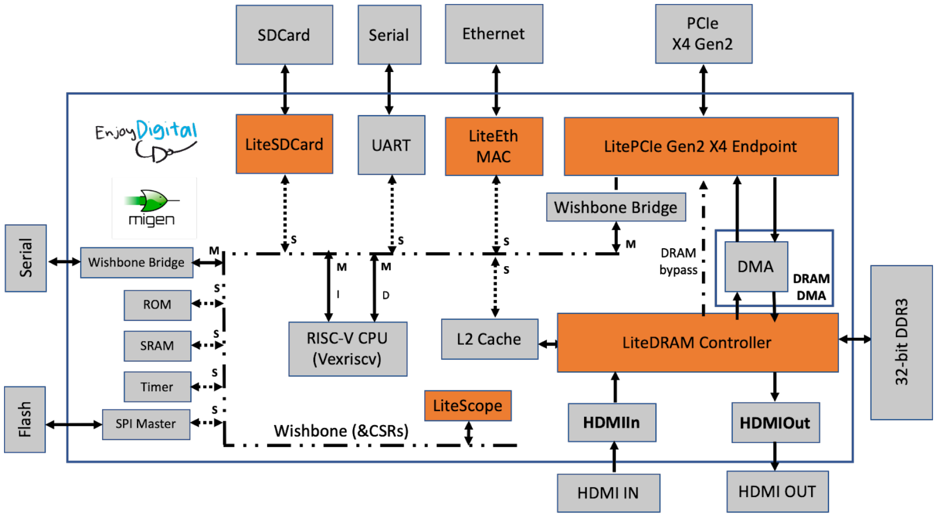
bsg_tapeout_v1_tsmc40

Tcl 0 ★ 1 0 0 Updated 29 days ago

bsg_tapeout_v0_gf12

Tcl 0 ★ 1 0 0 Updated on Oct 4

Virality



Hundreds of students are trained each year and have learned to Use, Modify and Contribute to BlackParrot:

EE 477: VLSI, Year Two
CSE 548: Grad Architecture
BU EC700: Advance Comp Arch

Released easy to use FPGA version (Litex)

Integrated with the OpenPiton BYOC infrastructure

Give tutorials & present at workshops
- ISCA: computer architecture researchers
- DAC: CAD tool researchers
- RISC-V Workshop
- Hotchips: record-busting metrics presented to industry
- FOSDEM: General open source community

Find niches and “must have” features for them

Seed research groups, encourage contributions back to main branch; student exchanges

Meetups, Maker communities

Give talks at companies and seek feedback & advice
Microsemi, AMD, VMWare, Microsoft, Google, Oracle

BlackParrot: A Base Class for Accelerator SoCs

BlackParrot is a Linux-capable RISC-V multicore, ideal as a lightweight host

BlackParrot is silicon-validated and ready to be included in your next project!

Please explore, use, break things and let us know your experience!

Thanks! Questions?

Dan Petrisko

petrisko@cs.washington.edu

https://github.com/black-parrot

This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7856. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.