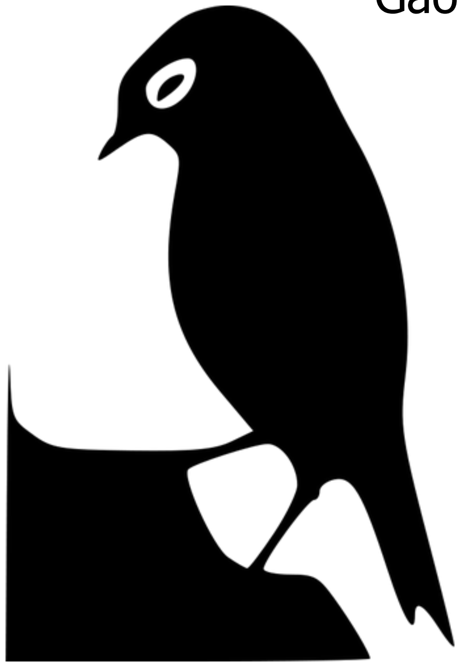
BlackParrot: An Agile Open Source RISC-V Multicore for Accelerator SoCs

Daniel Petrisko[1], Farzam Gilani[1], Mark Wyse[1], Tommy Jung[1], Scott Davidson[1], Paul Gao[1], Chun Zhao[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



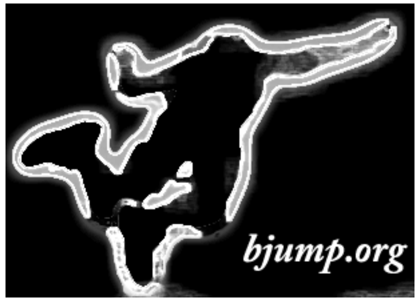
BOSTON UNIVERSITY

Hardware Development is Rapidly Opening Up

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



OpenROAD

FuseSoC

RISC-V®

VERILATOR

CHIPS ALLIANCE

Yosys Open SYnthesis Suite

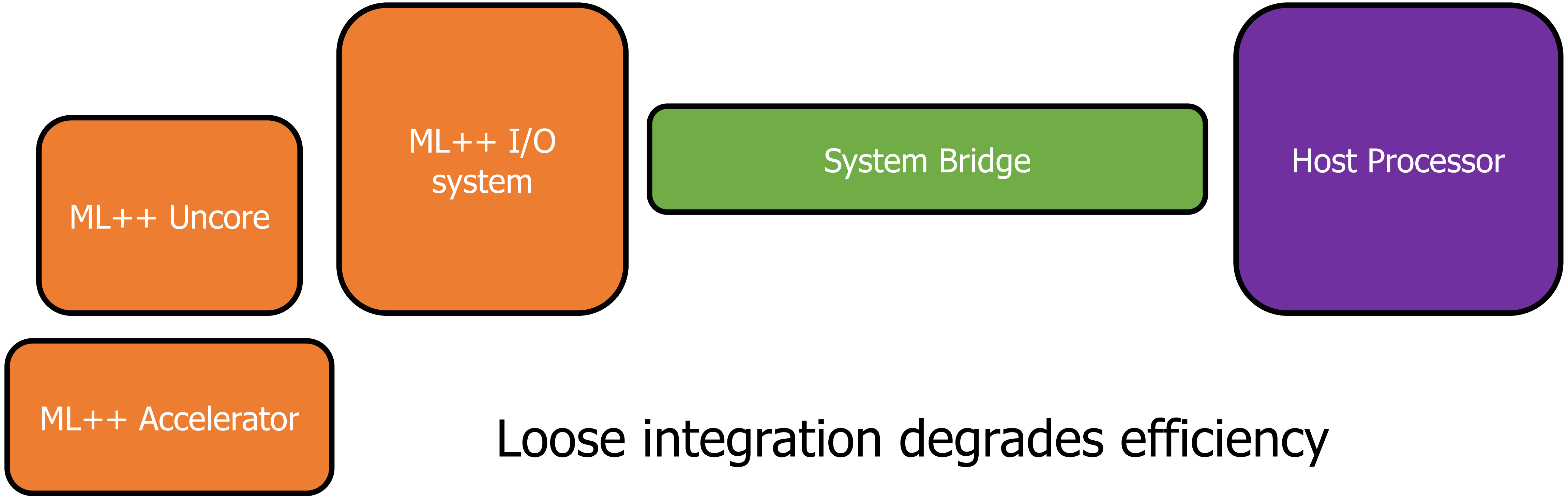
Diverse Workloads Demand Custom Silicon

Accelerators provide performance and energy efficiency benefits

**“MLPlusPlus:
Accelerated Machine
Learning for the Cloud”**

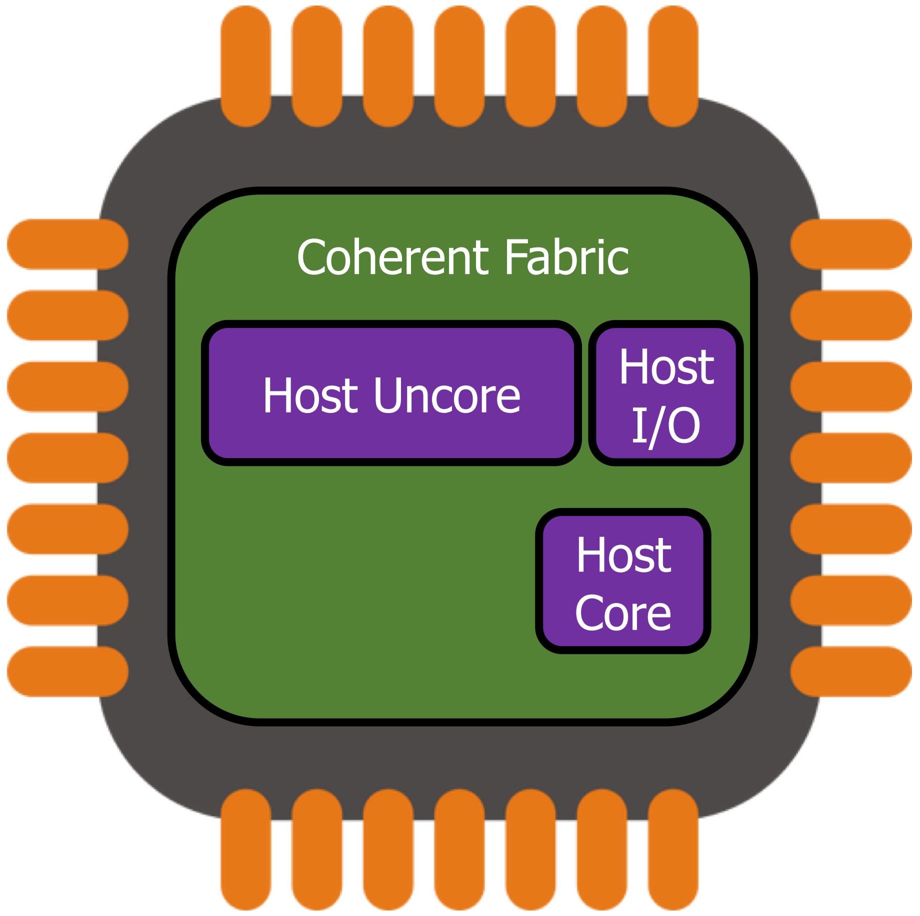
But the Accelerator Is a Small Part of the System

Need to develop an infrastructure around each accelerator



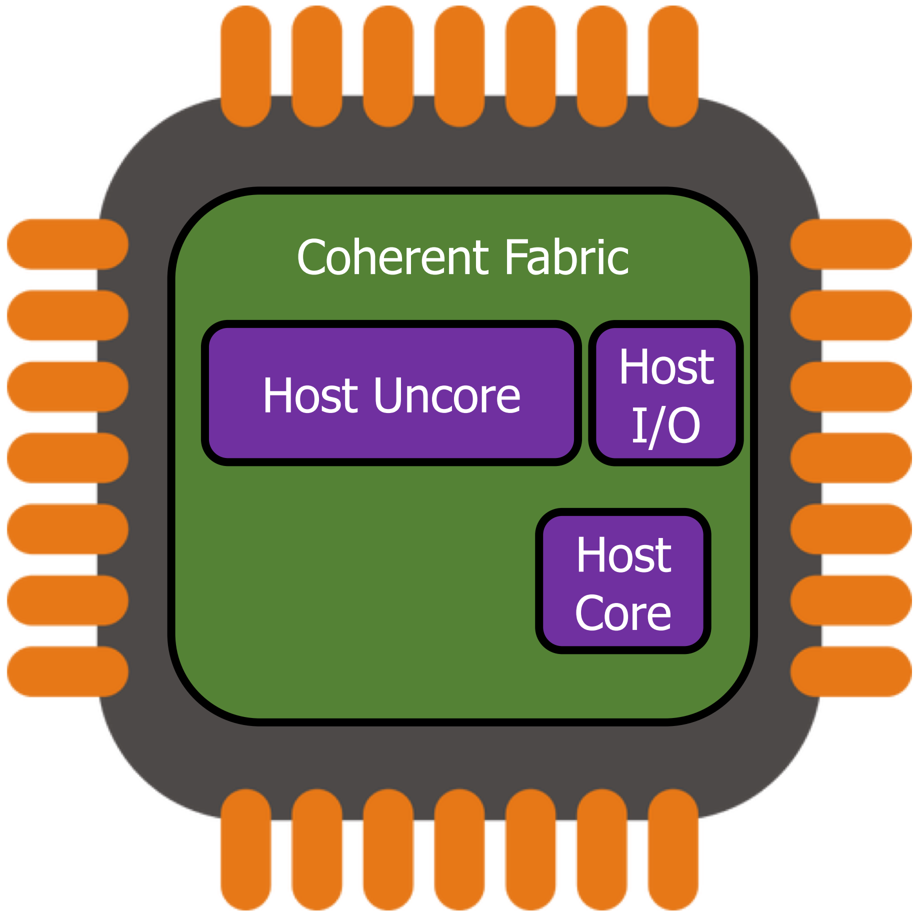
Loose integration degrades efficiency

An Ideal Host Processor Would Be:



An Ideal Host Processor Would Be:

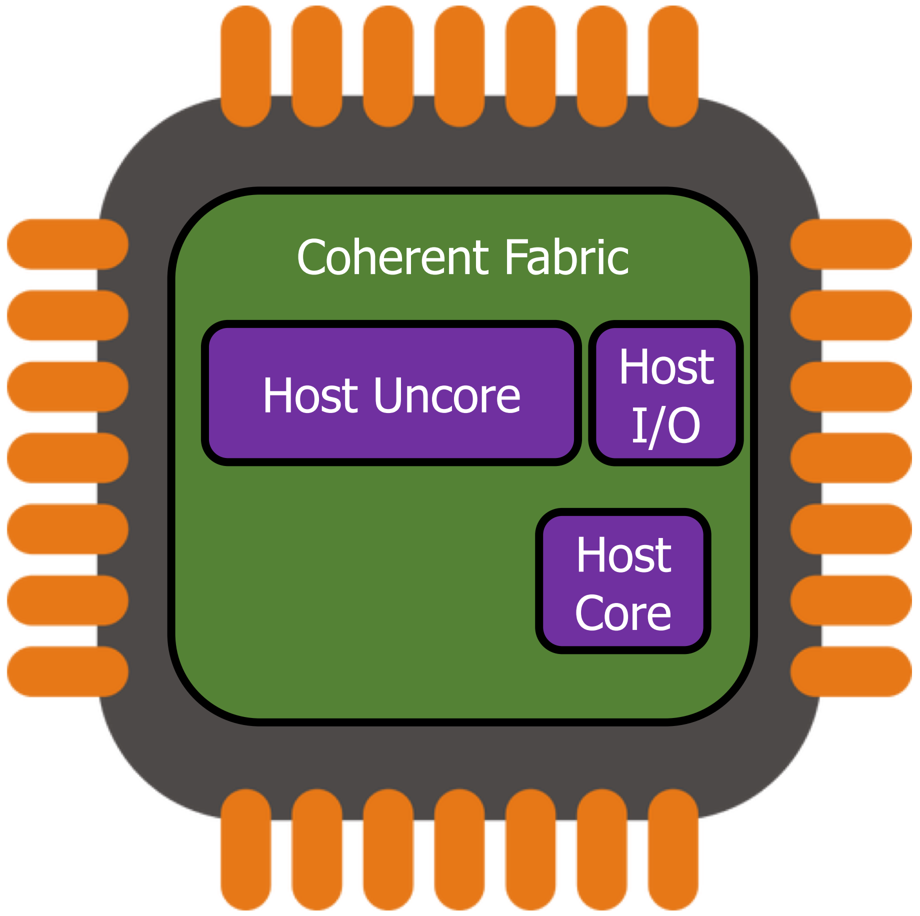
Efficient, performant and reliable



An Ideal Host Processor Would Be:

Efficient, performant and reliable

Easy to integrate with a variety of architectures

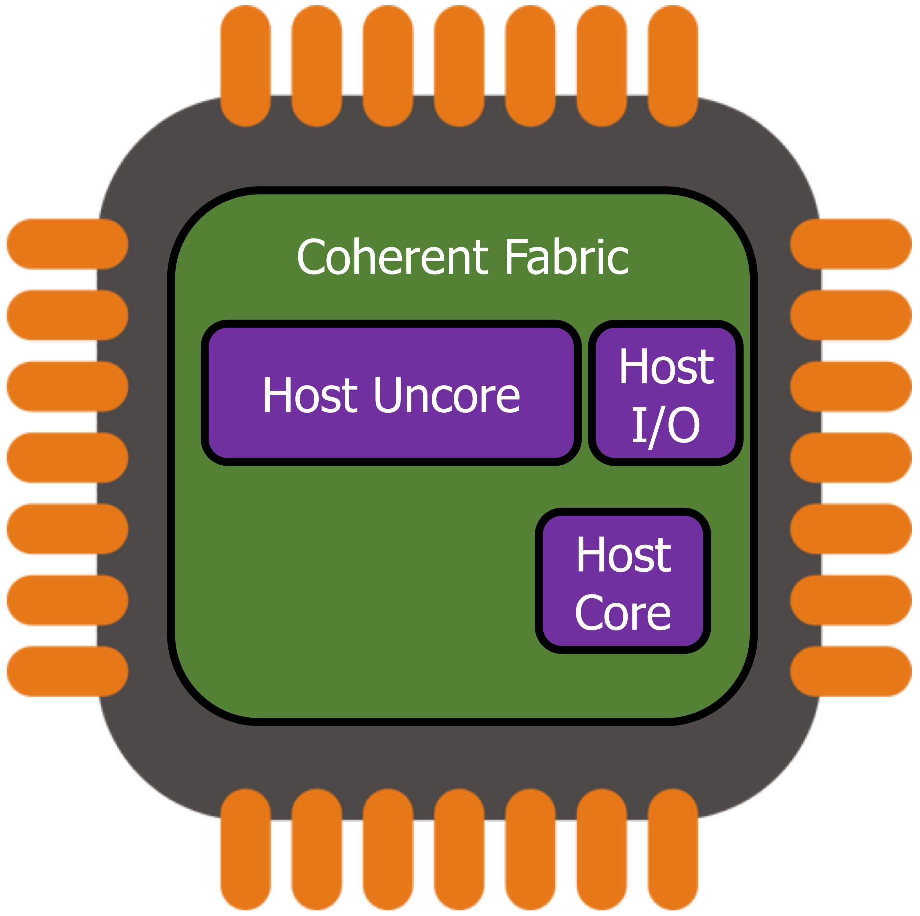


An Ideal Host Processor Would Be:

Efficient, performant and reliable

Easy to integrate with a variety of architectures

Thoroughly verified and silicon-validated



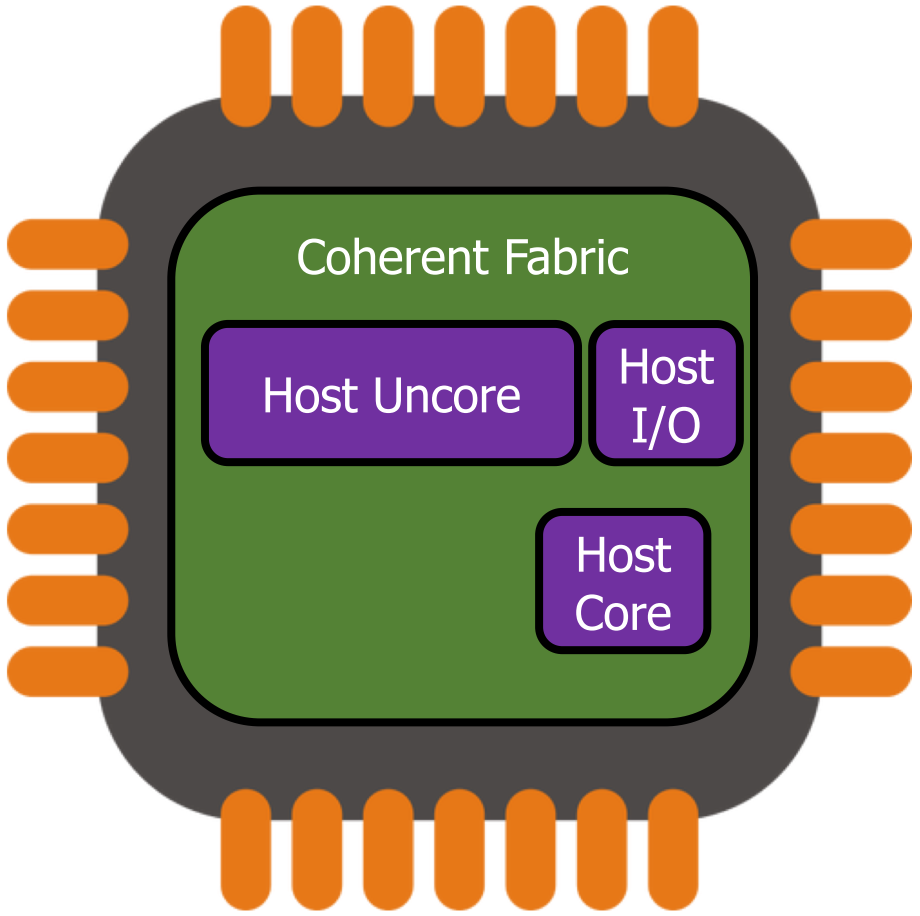
An Ideal Host Processor Would Be:

Efficient, performant and reliable

Easy to integrate with a variety of architectures

Thoroughly verified and silicon-validated

Free and open-source!



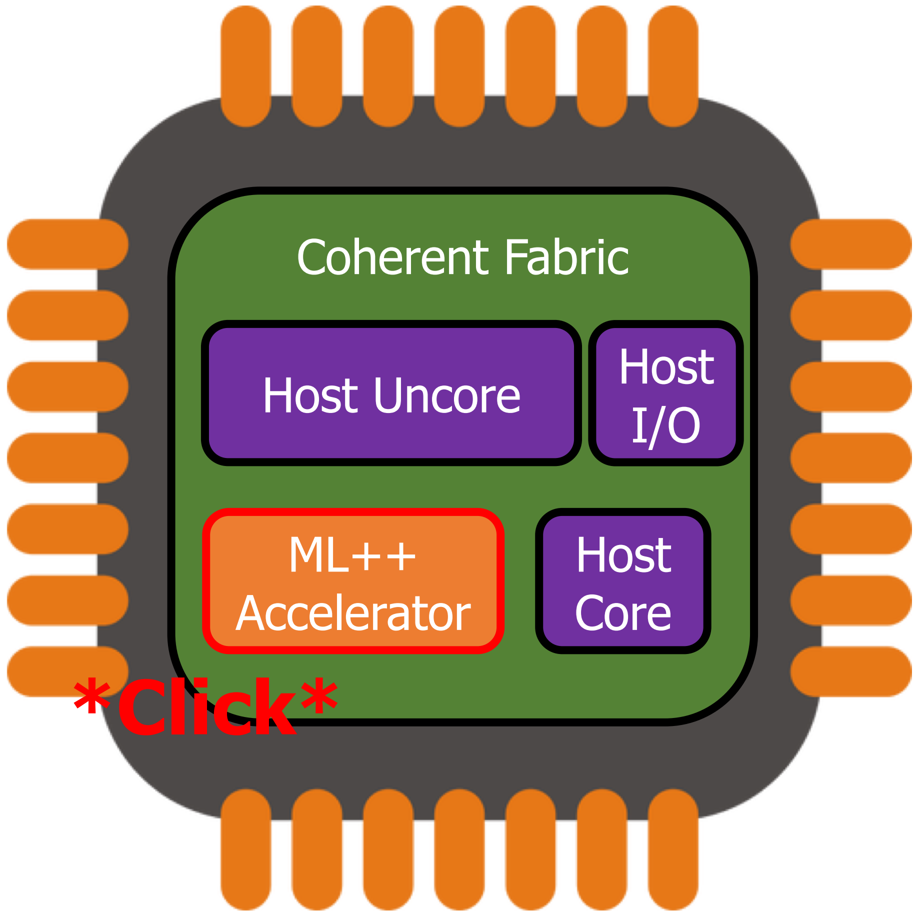
An Ideal Host Processor Would Be:

Efficient, performant and reliable

Easy to integrate with a variety of architectures

Thoroughly verified and silicon-validated

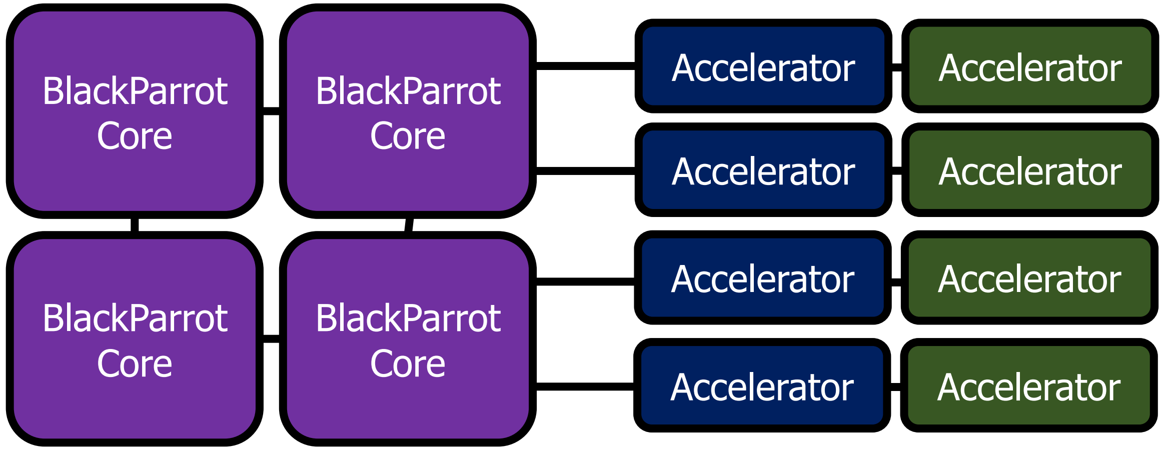
Free and open-source!



BlackParrot: A “Base Class” for Accelerator SoCs

Goal:
To be the default Linux-capable host multicore used 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

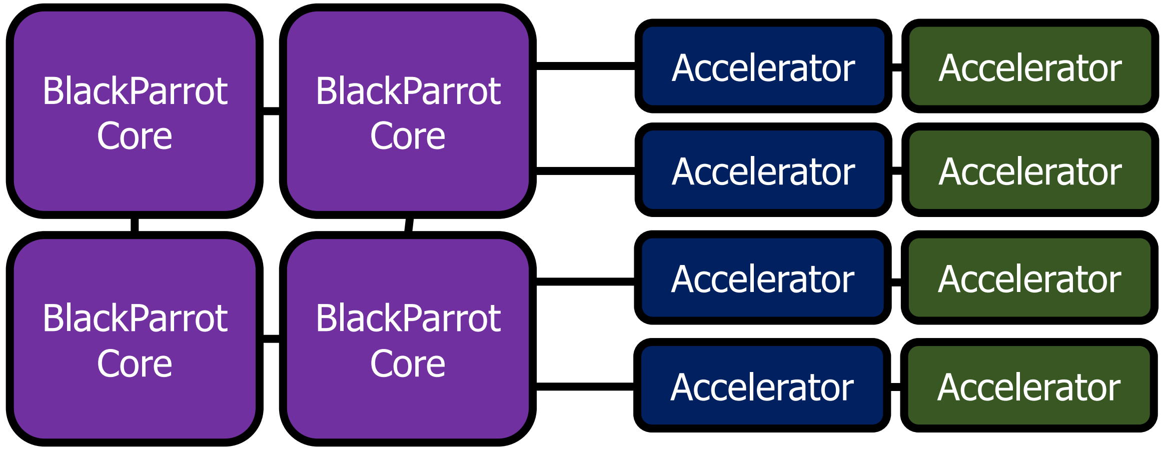


BlackParrot: A “Base Class” for Accelerator SoCs

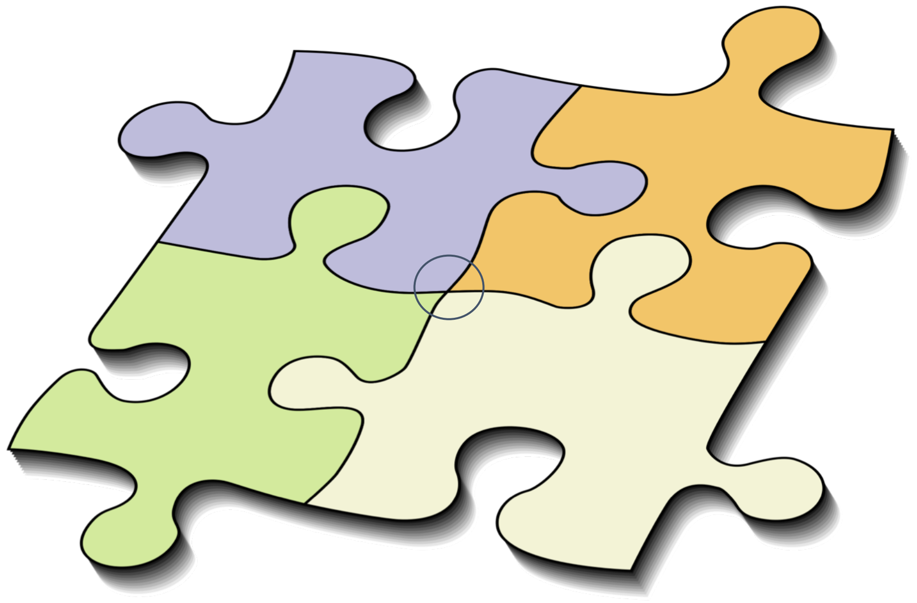
Goal:
To be the default Linux-capable host multicore used by the world

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

- Anti-target: Intel Xeon competitor



BlackParrot: Four Success Metrics



BlackParrot: Four Success Metrics

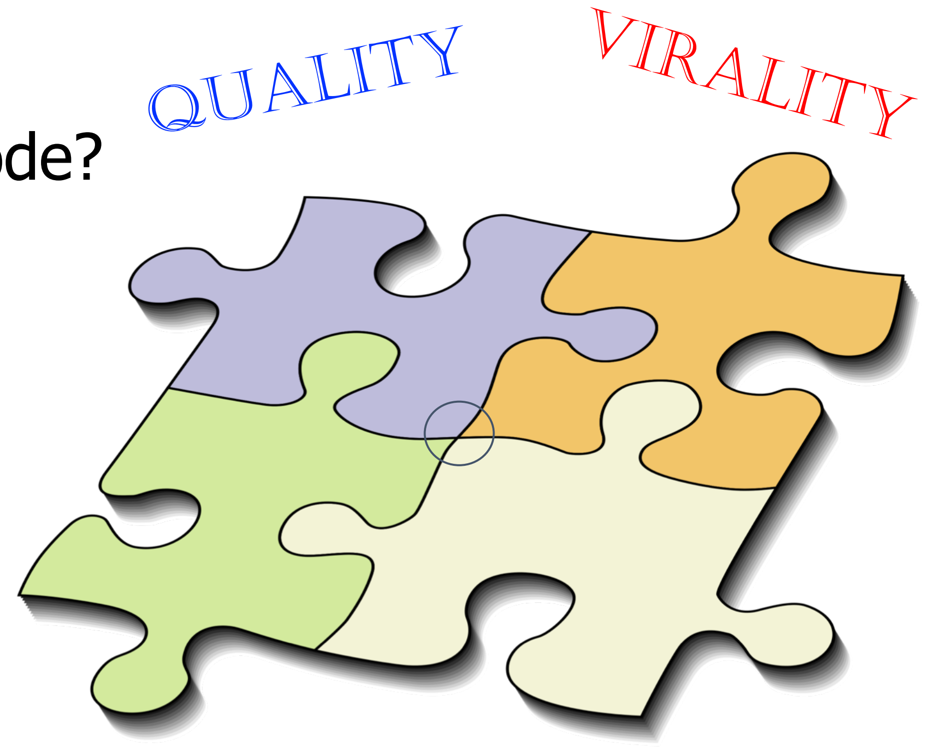
Will people trust our code?

- 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?



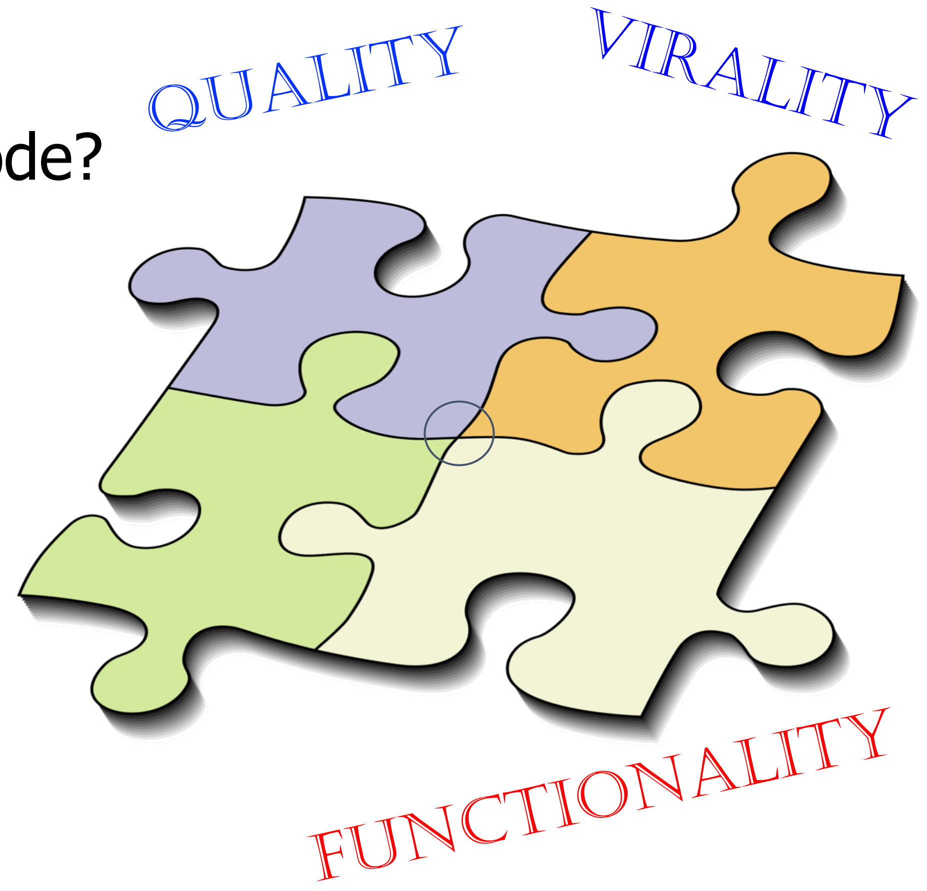
Will people use our code?

- 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?

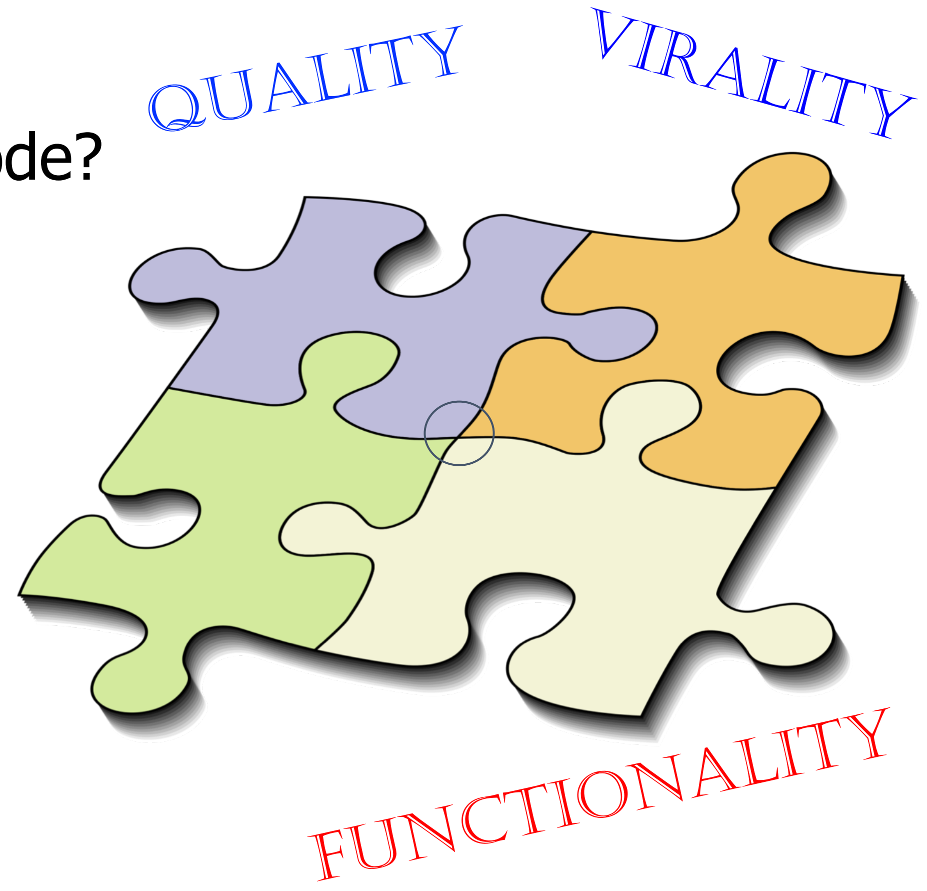


Will people use our code?

- 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?

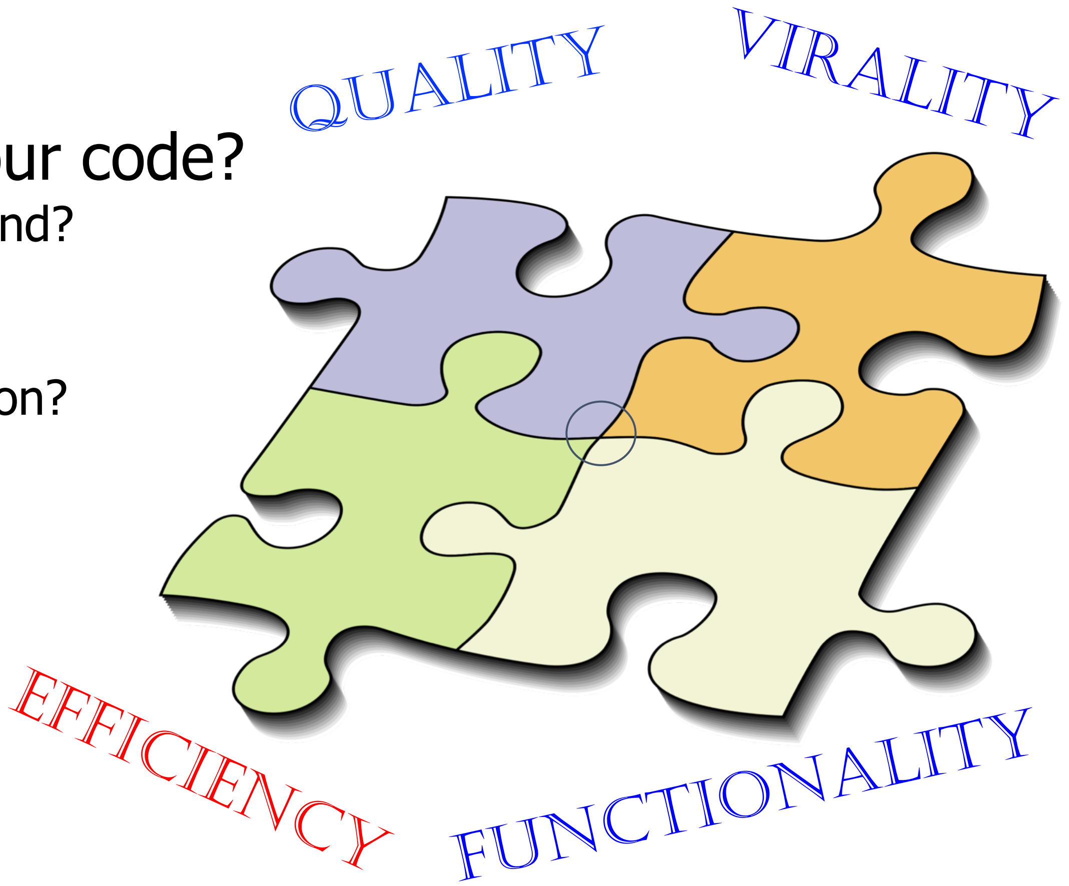
Will people use our code?

- 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



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?

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

The BlackParrot Manifesto

Be TINY

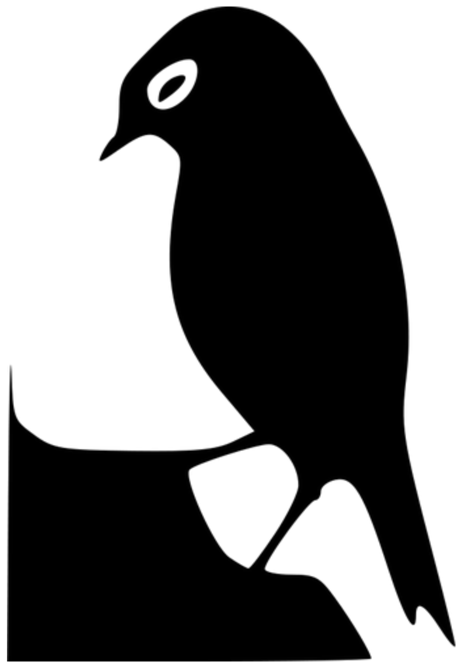
- Place a premium on small, understandable code base
- Minimize unused features or configurations that increase complexity

Be Modular

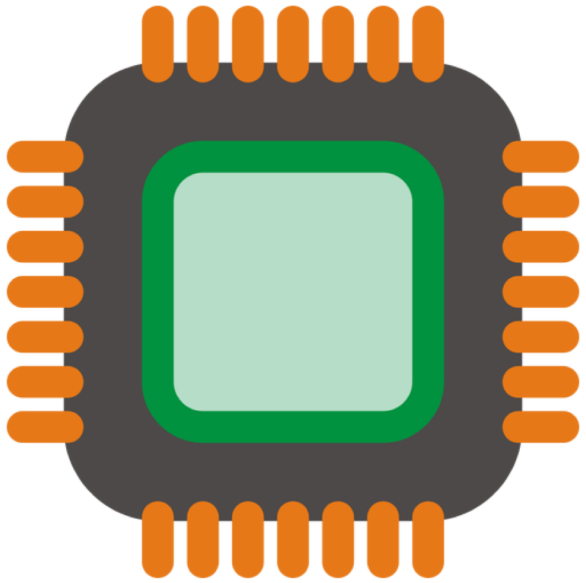
- Use well-defined interfaces to enable scalable, global participation
- Maintain modular testability and continuous integration

Be Friendly

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

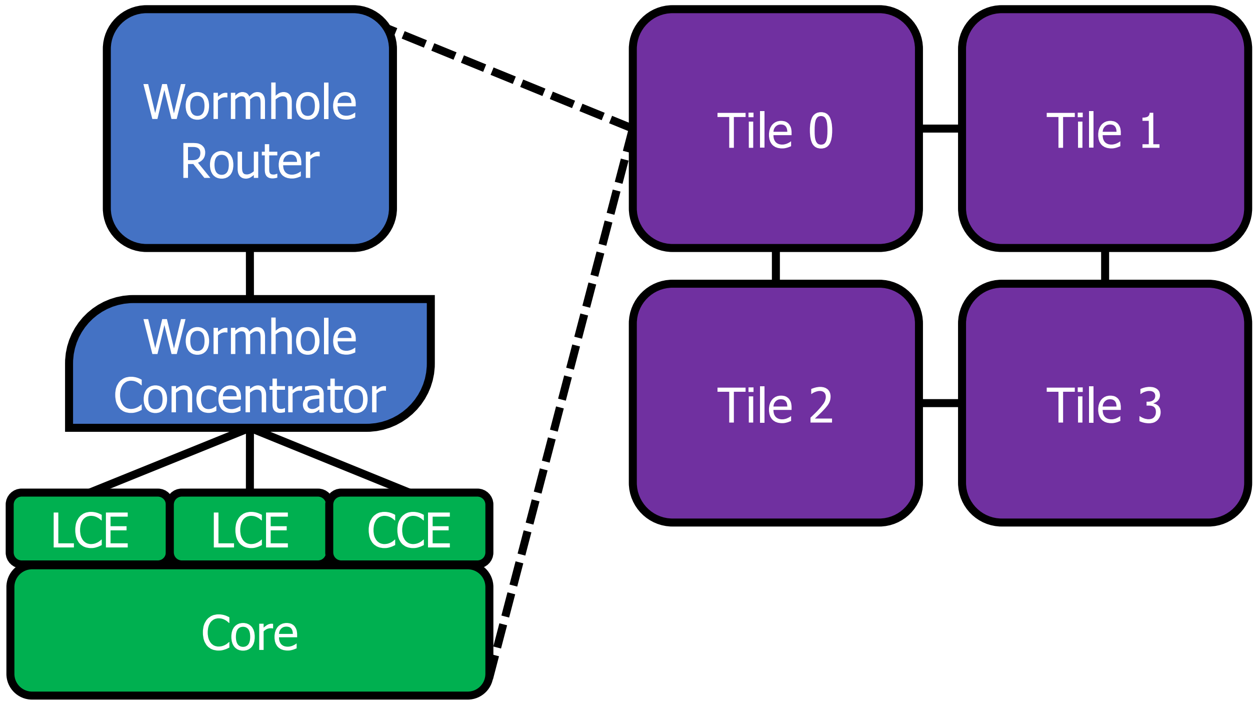


BlackParrot System Architecture



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



BedRock: A Race-Free Directory-Based Coherence Protocol

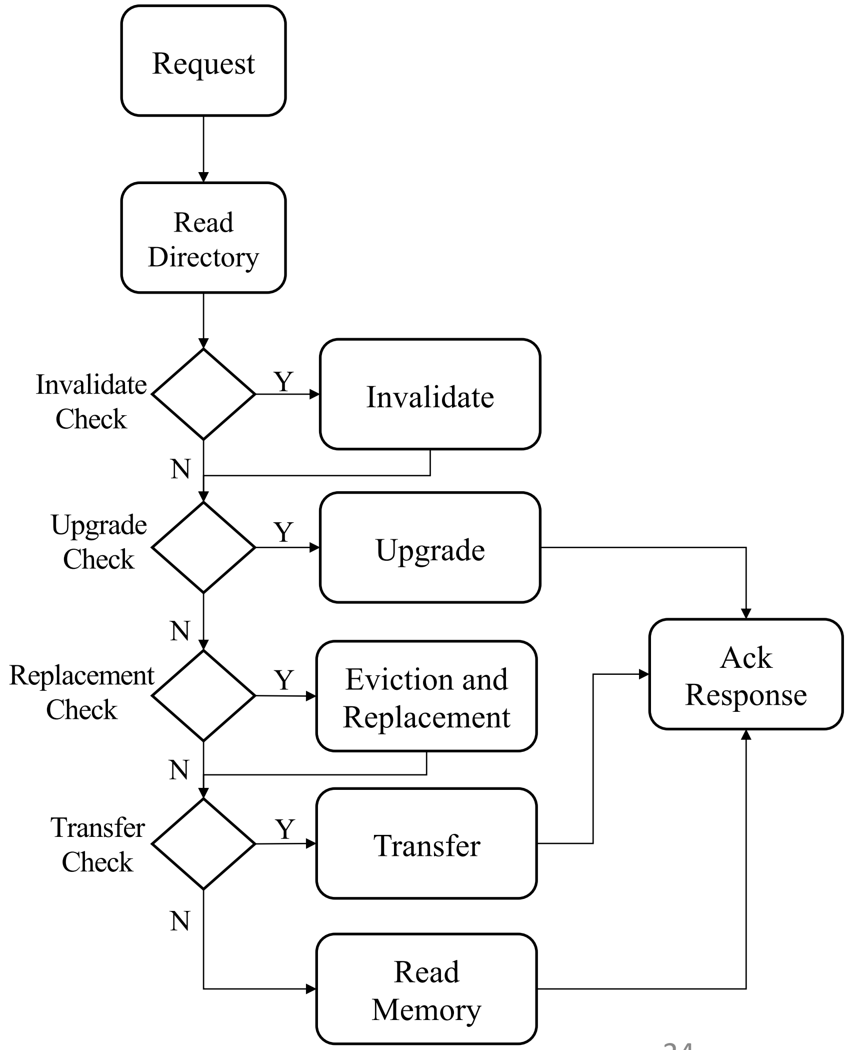
Separate into two components: Local Cache Engines (LCEs) and Cache Coherence Engines (CCEs)

LCEs are "dumb", all control logic is in CCE

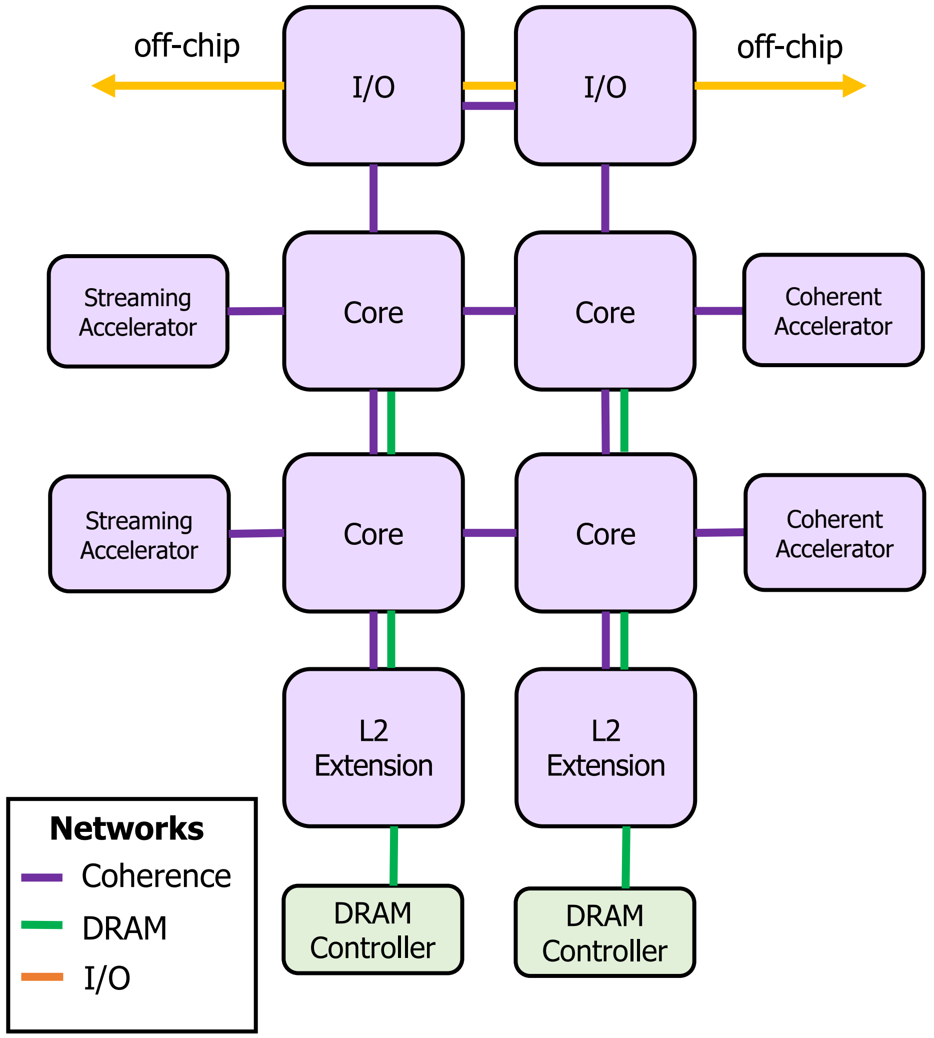
Atomic transactions, zero transient states in protocol!

Significantly faster to verify correctness compared to traditional MESI

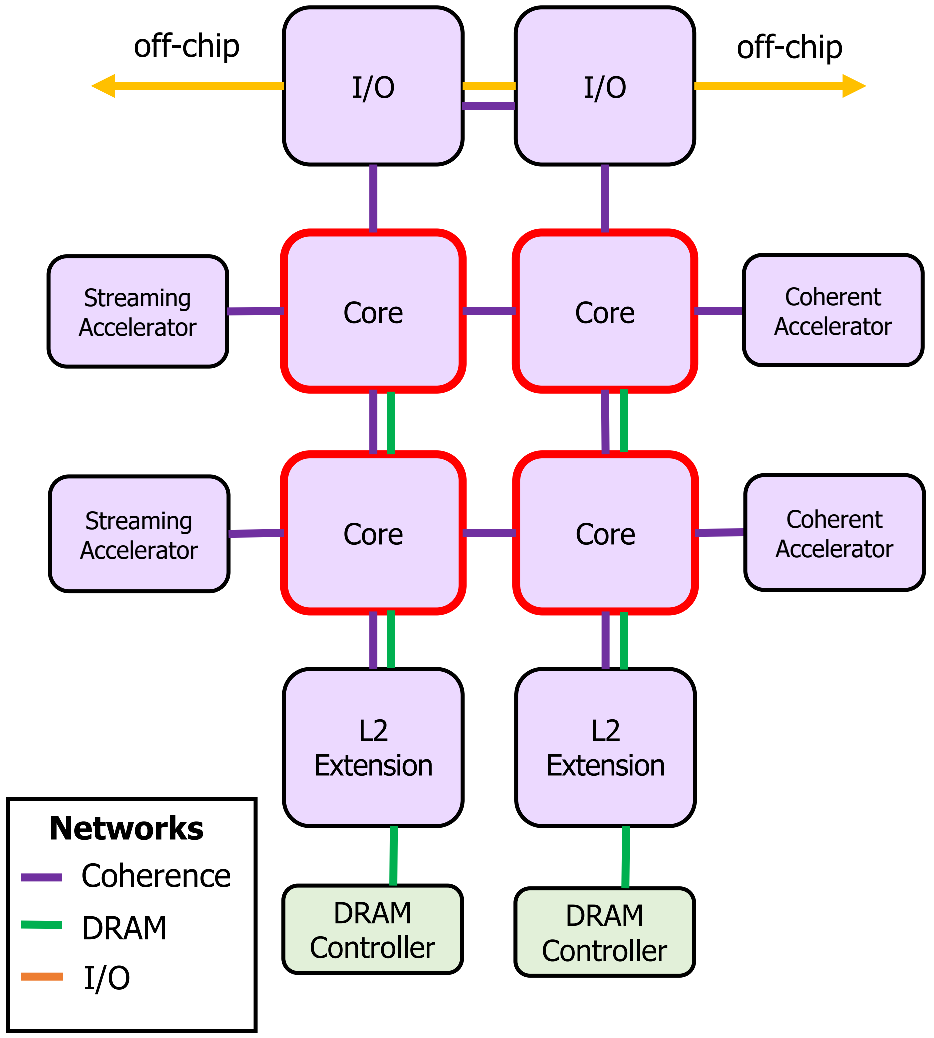
- 26.9x faster for a 6-cache system
- Verified correct in 3.5 hours for 7-cache system while traditional MESI protocol did not finish verification within 72 hours



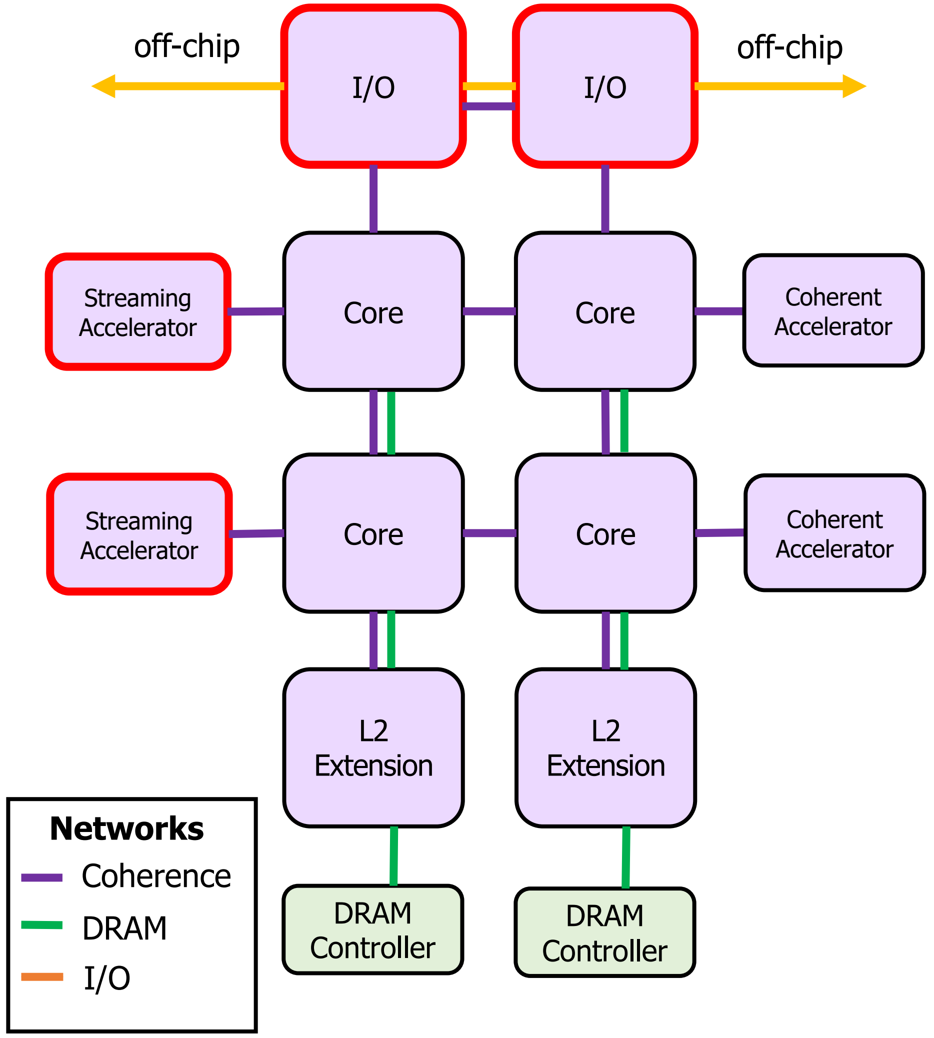
BlackParrot Supports Diverse Tile Types



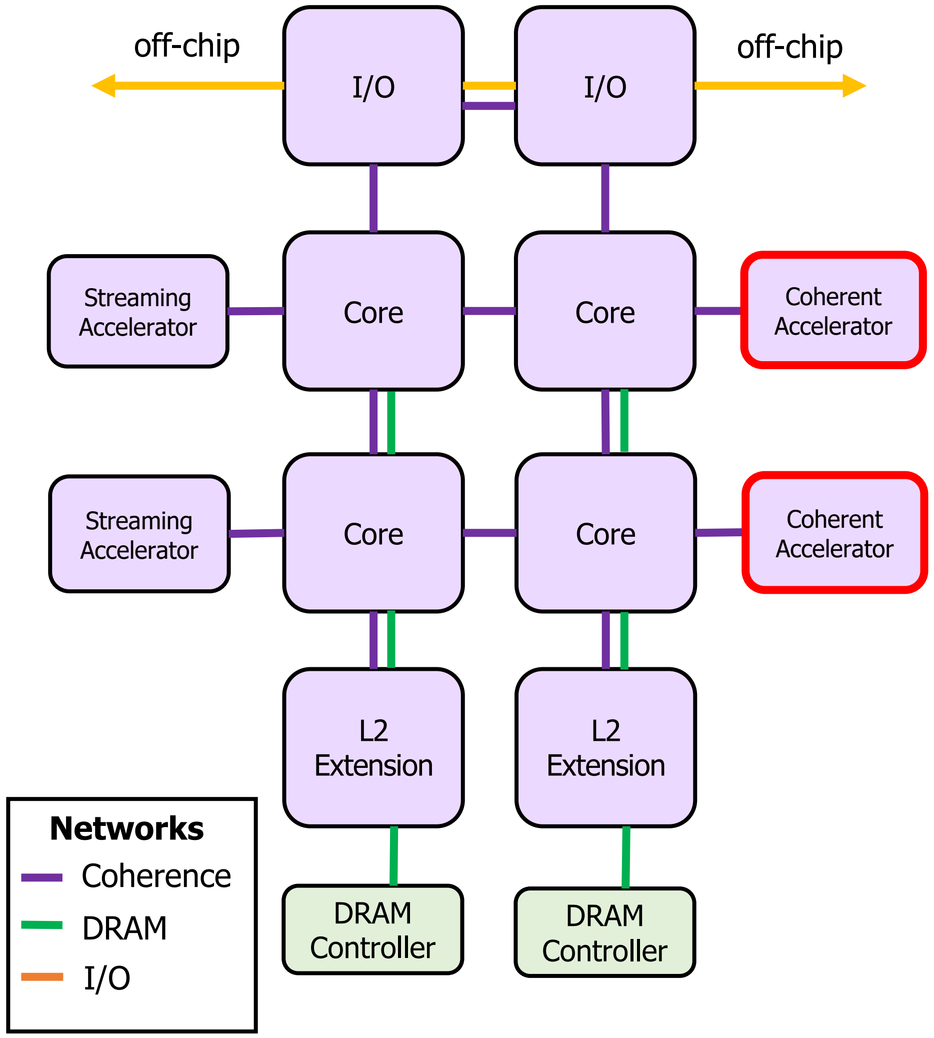
BlackParrot Supports Diverse Tile Types



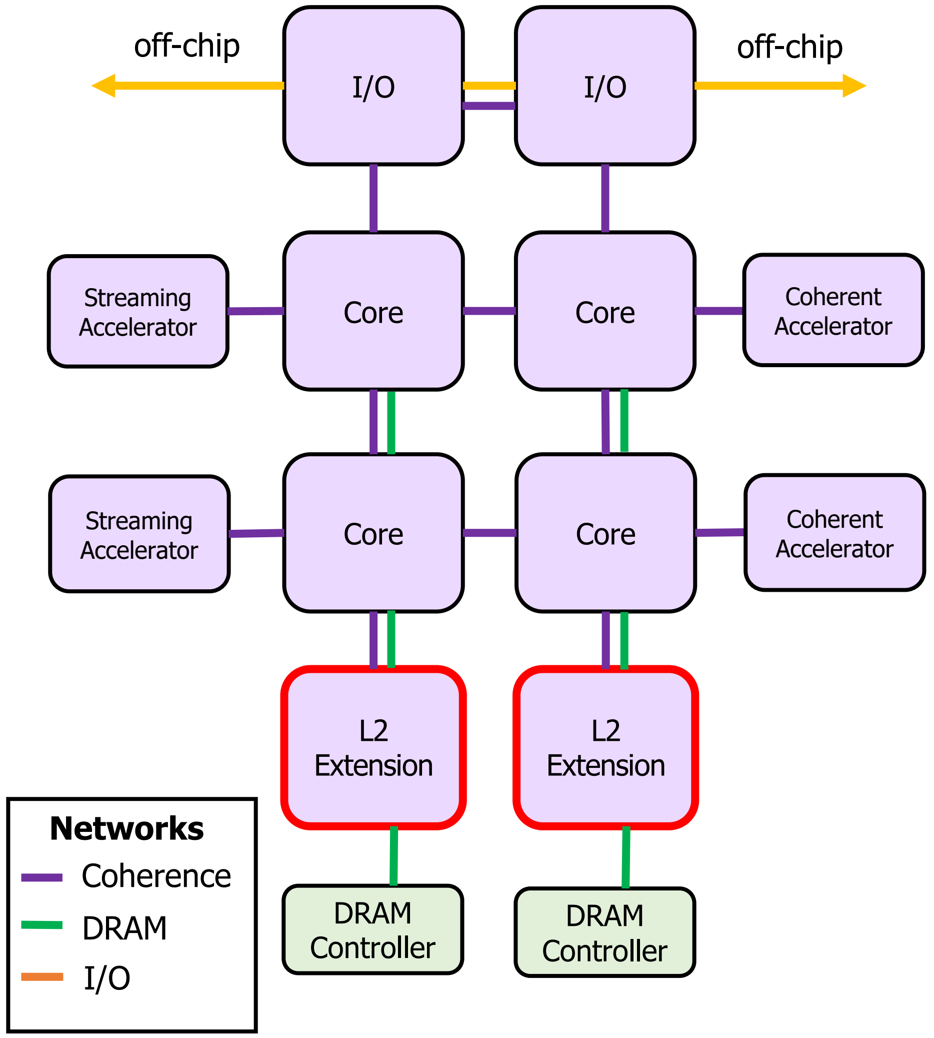
BlackParrot Supports Diverse Tile Types



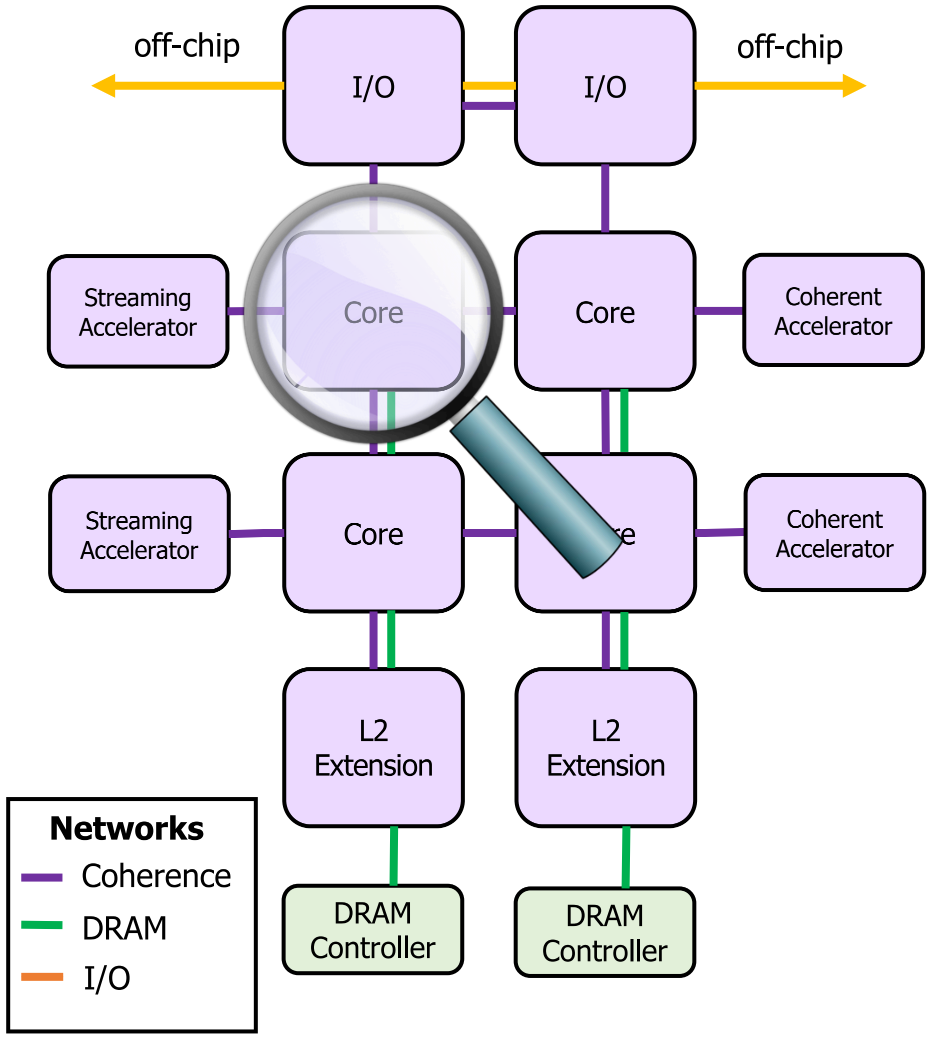
BlackParrot Supports Diverse Tile Types



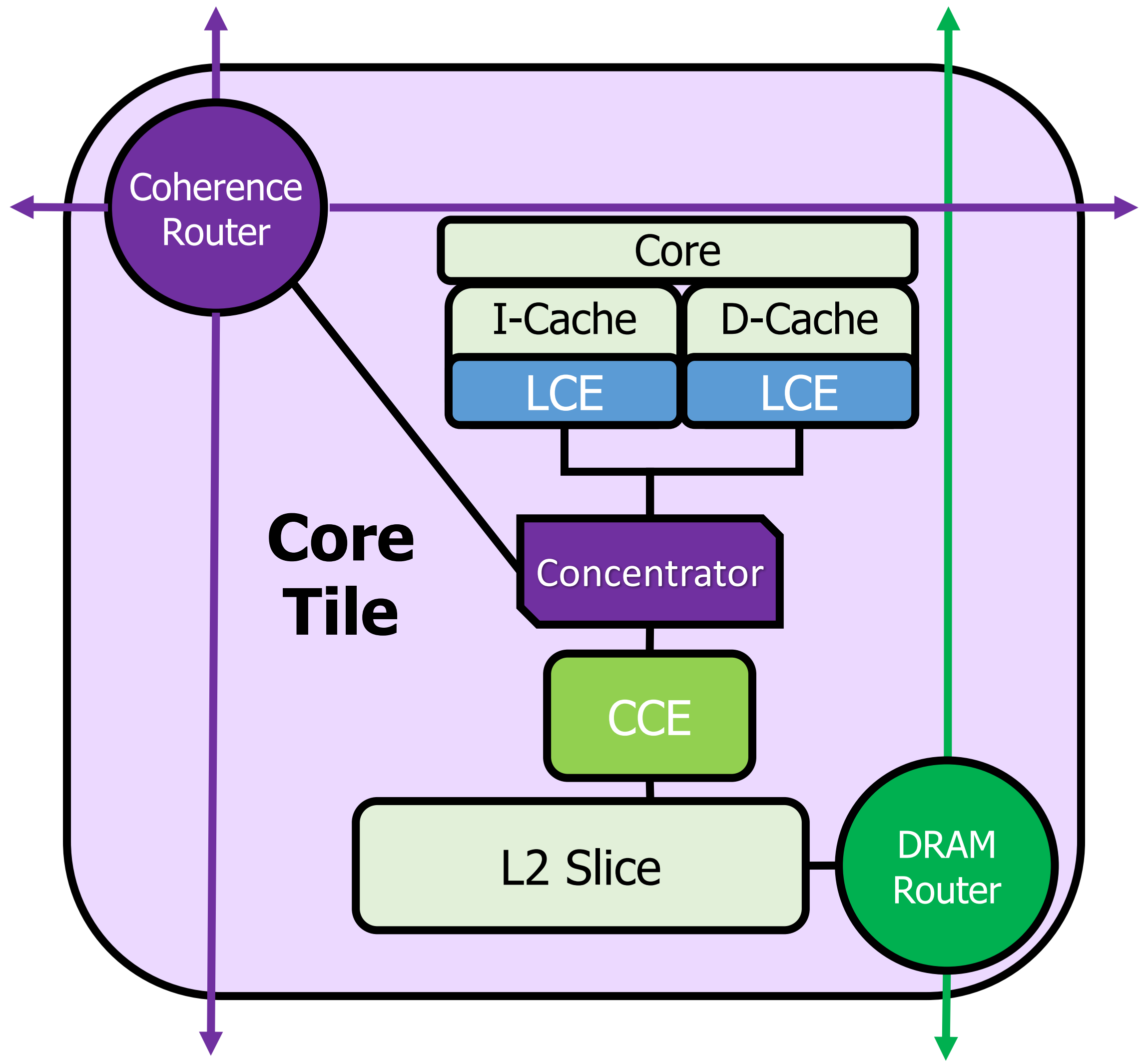
BlackParrot Supports Diverse Tile Types



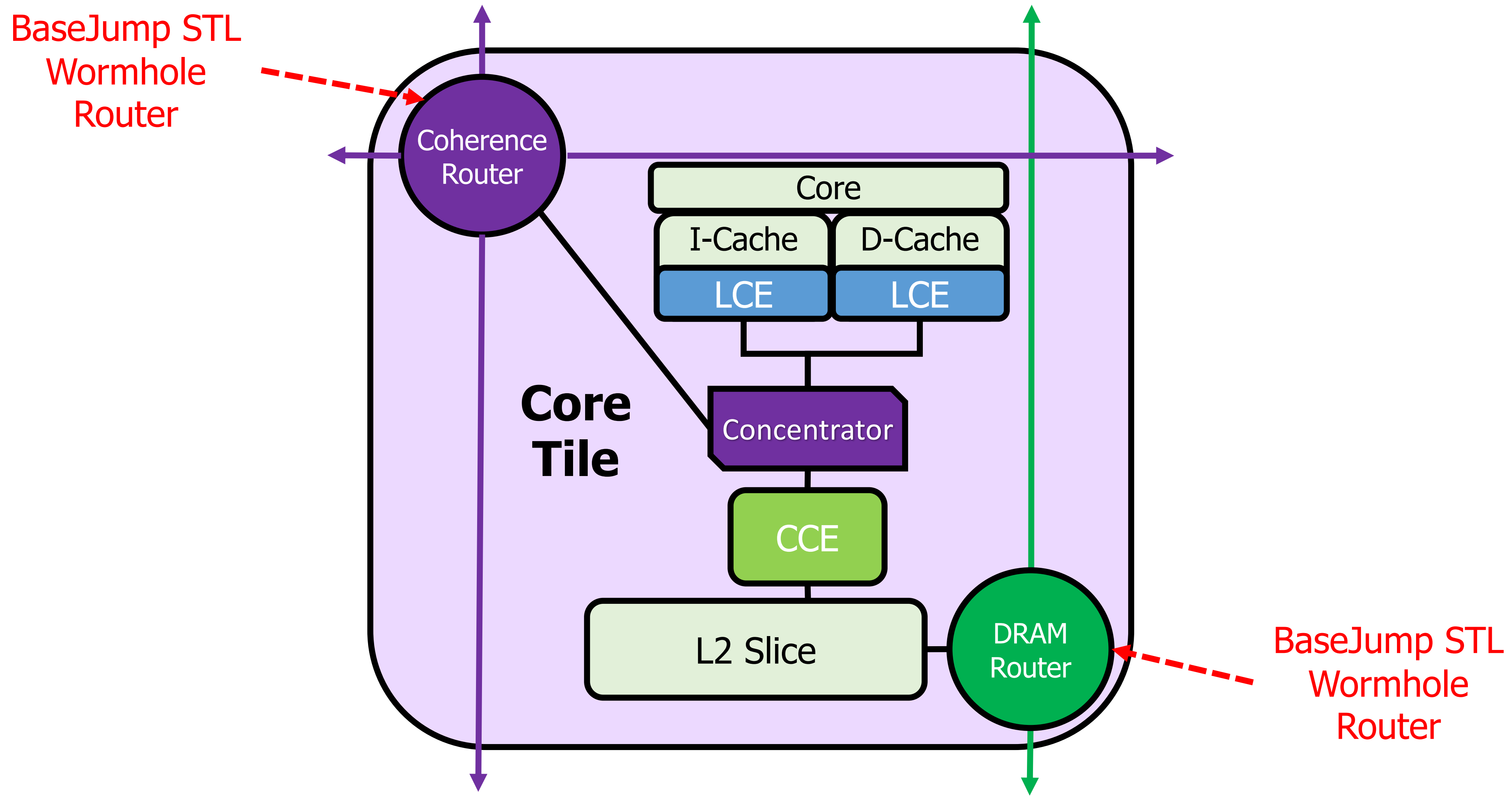
BlackParrot Supports Diverse Tile Types



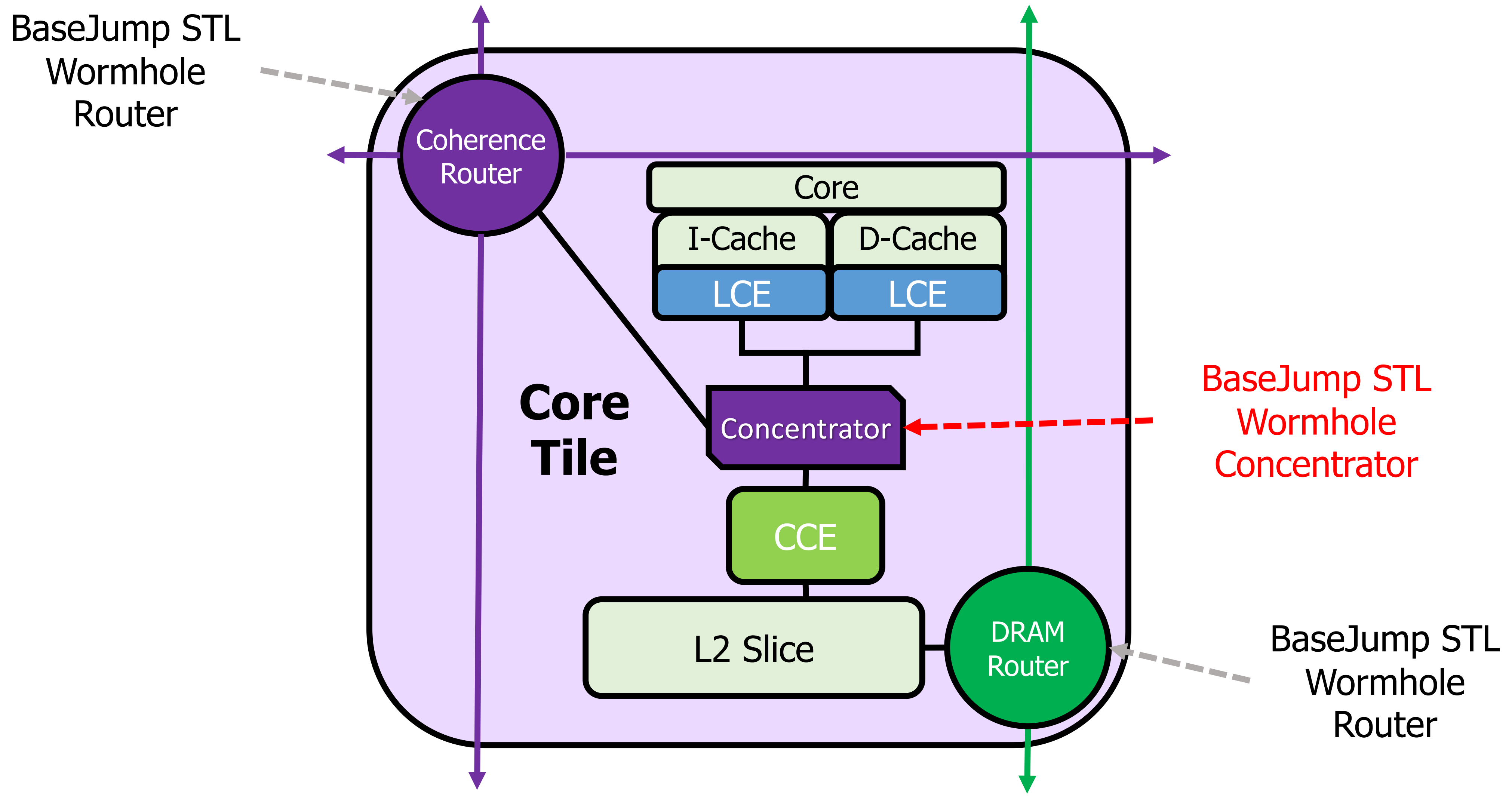
Assemble Tiles Out of Reusable Components



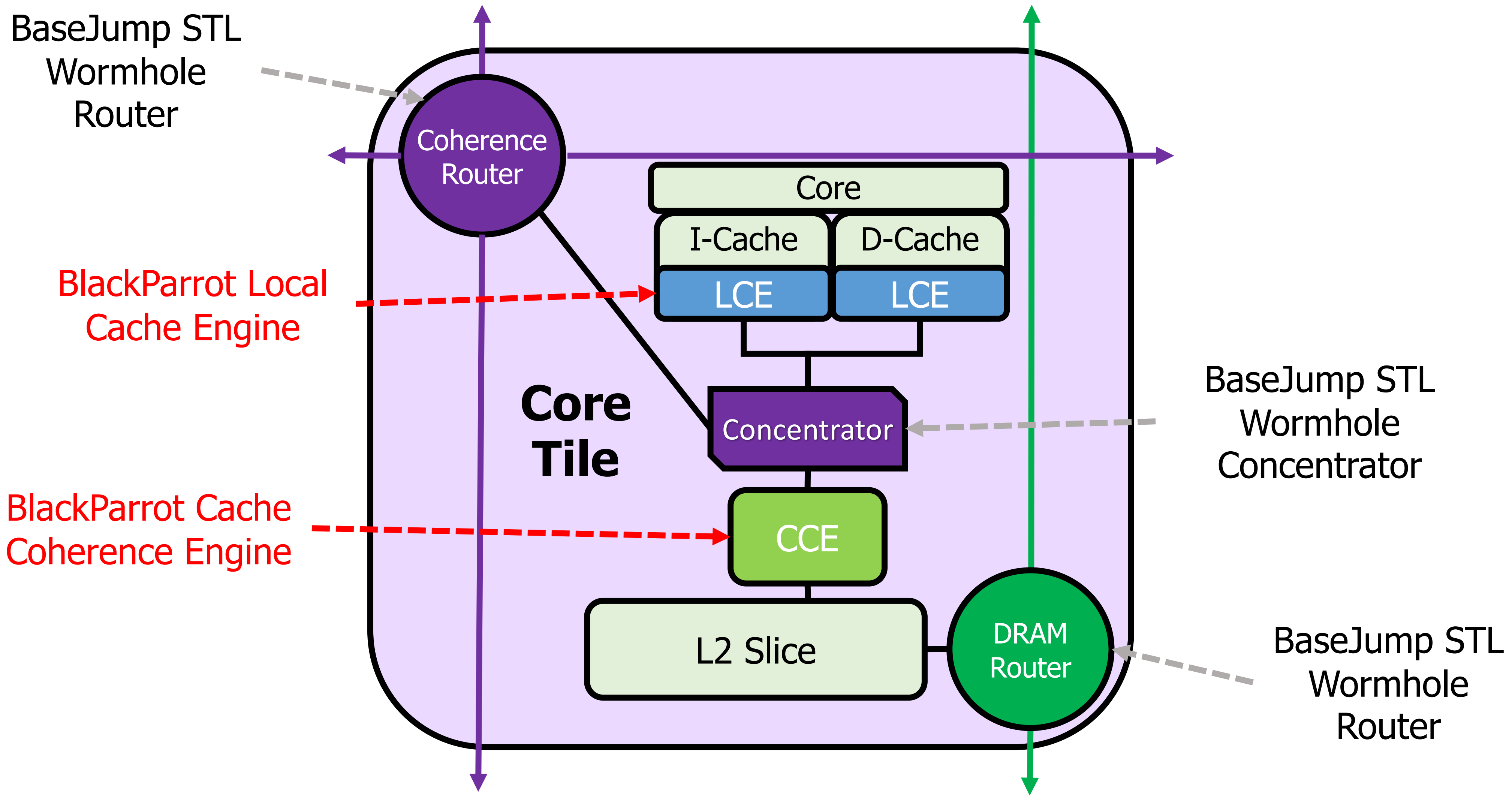
Assemble Tiles Out of Reusable Components



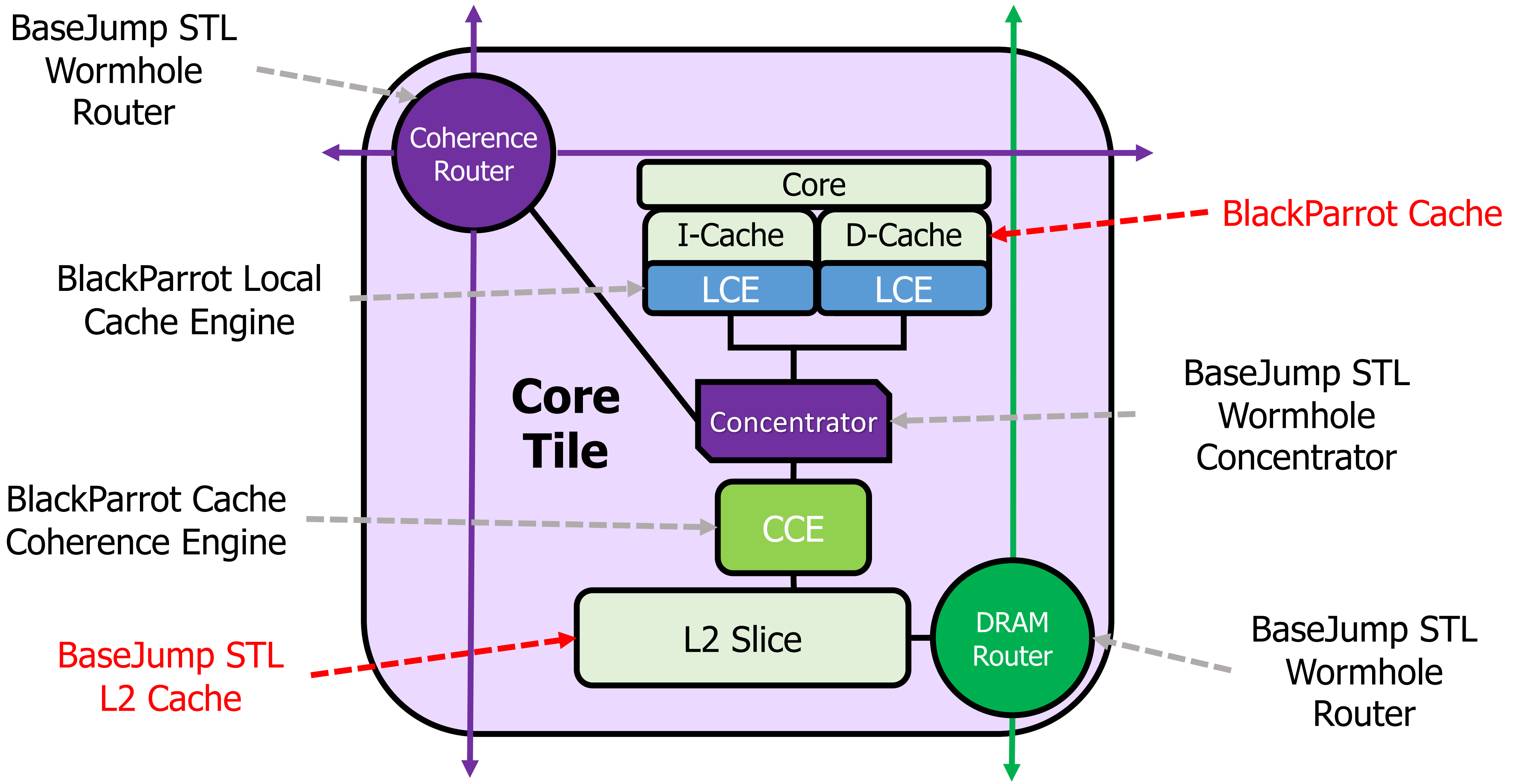
Assemble Tiles Out of Reusable Components


BaseJump STL Wormhole Router
Coherence Router
Core
I-Cache
D-Cache
LCE
LCE
Core Tile
Concentrator
BaseJump STL Wormhole Concentrator
CCE
L2 Slice
DRAM Router
BaseJump STL Wormhole Router

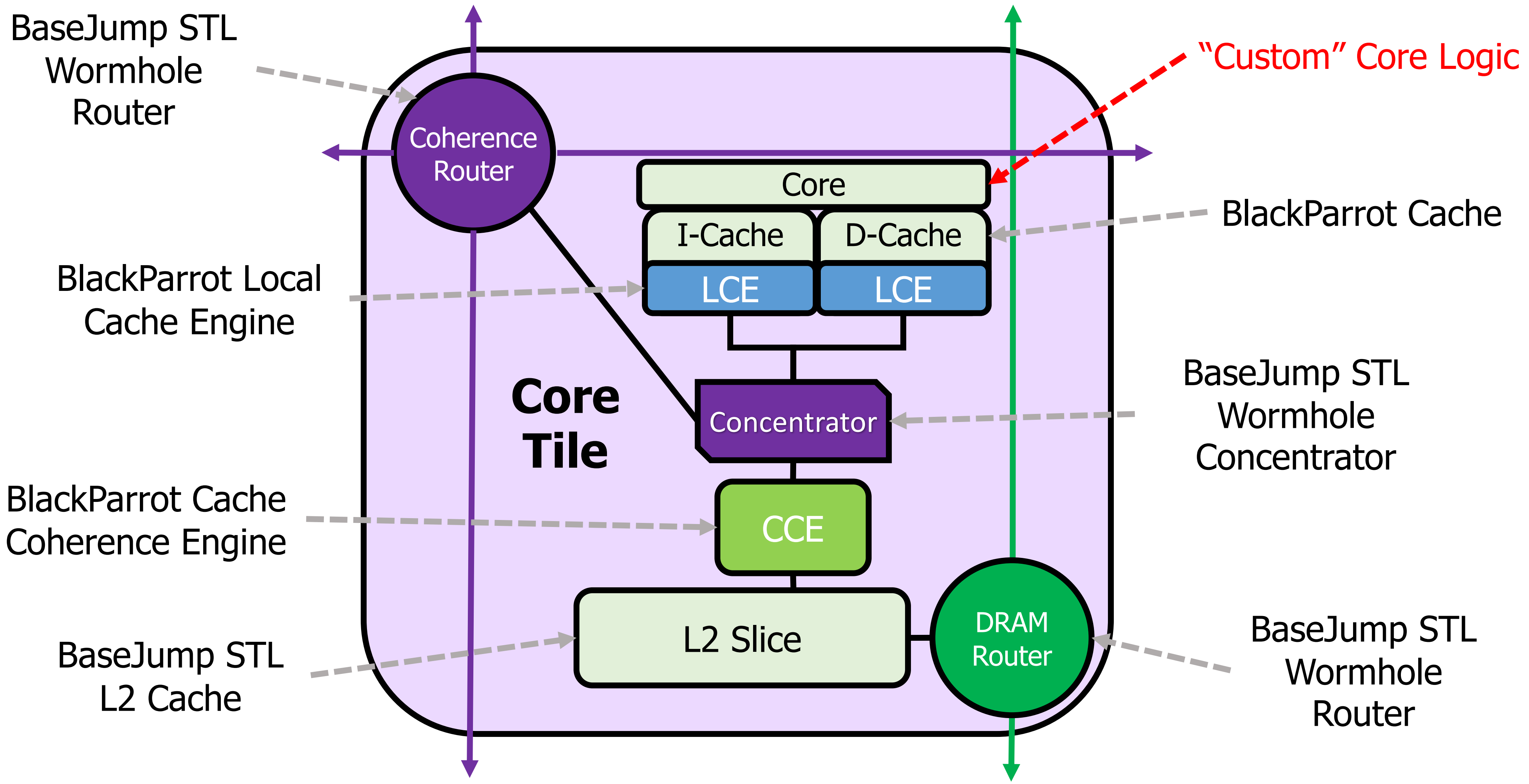
Assemble Tiles Out of Reusable Components


BaseJump STL Wormhole Router
Coherence Router
Core
I-Cache
D-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 Wormhole Router

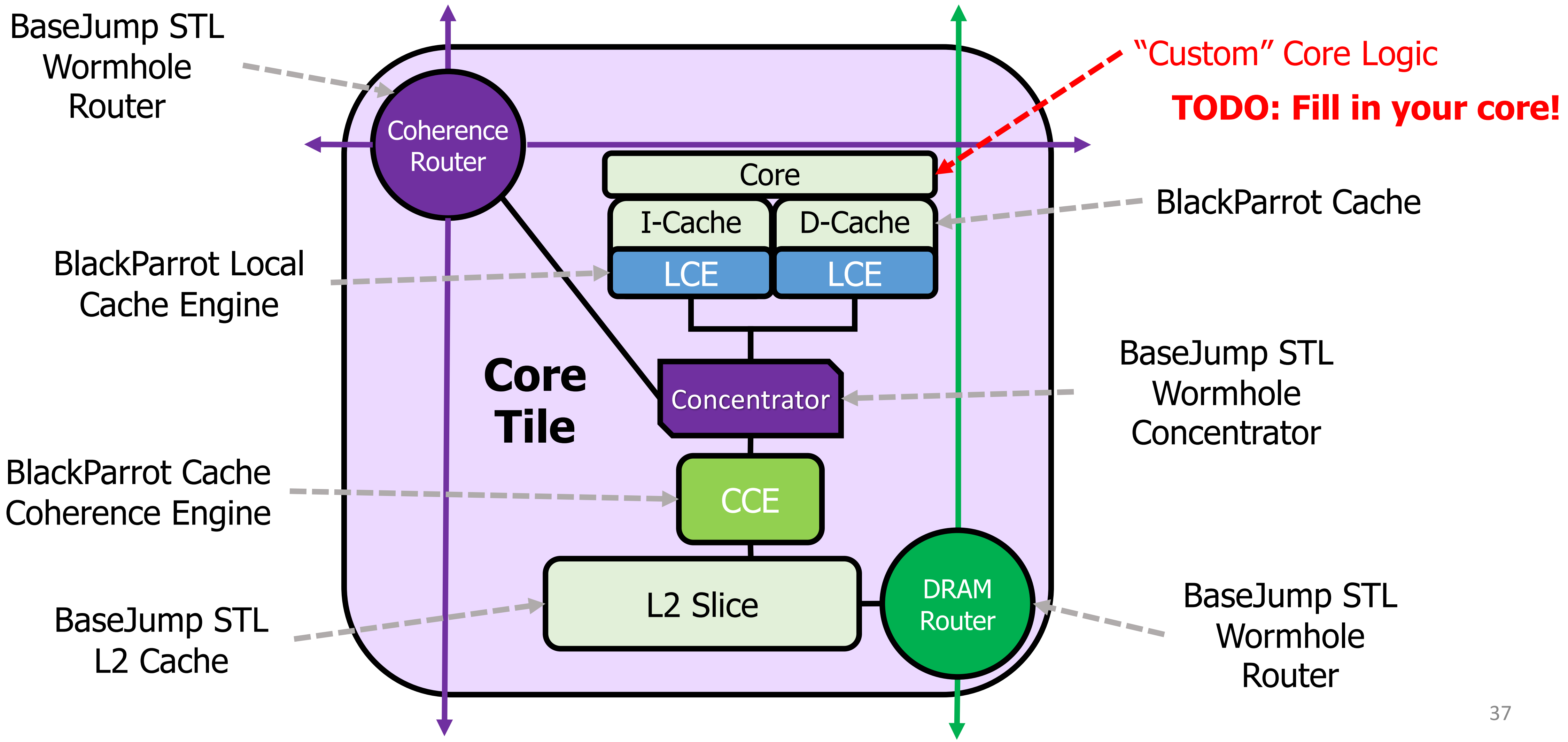
Assemble Tiles Out of Reusable Components


BaseJump STL Wormhole Router
Coherence Router
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

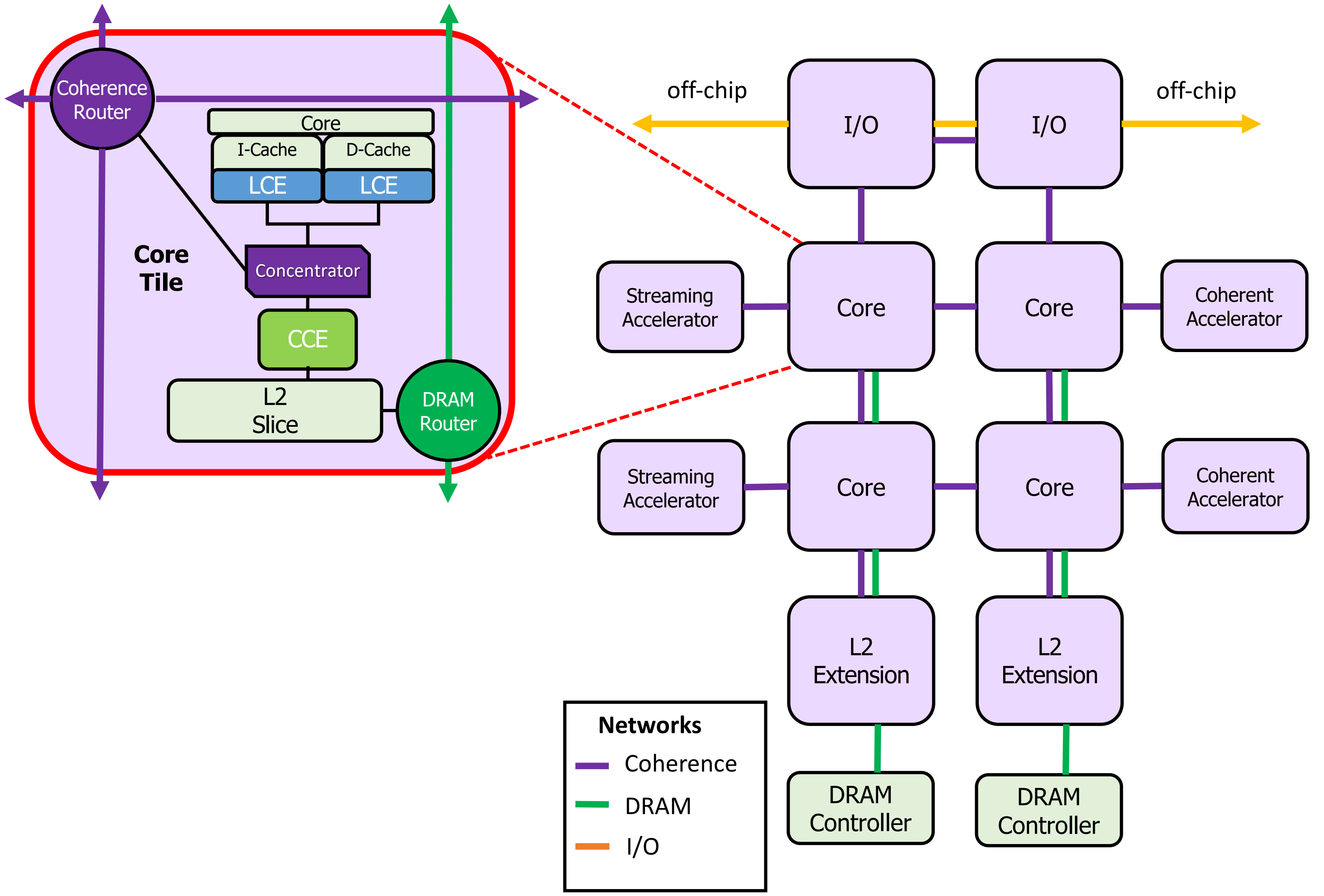
Assemble Tiles Out of Reusable Components


BaseJump STL Wormhole Router
"Custom" Core Logic
Coherence Router
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

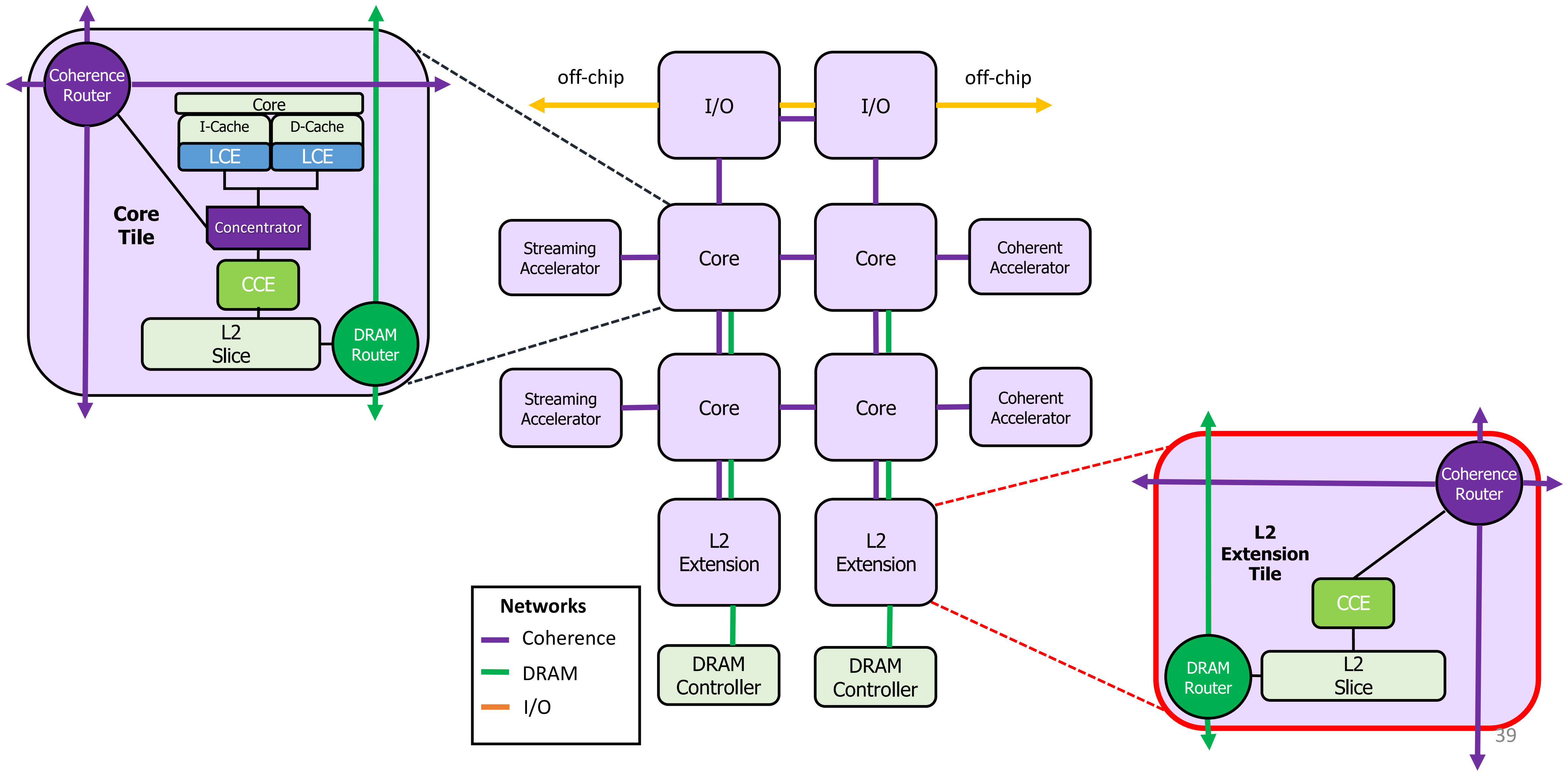
Assemble Tiles Out of Reusable Components



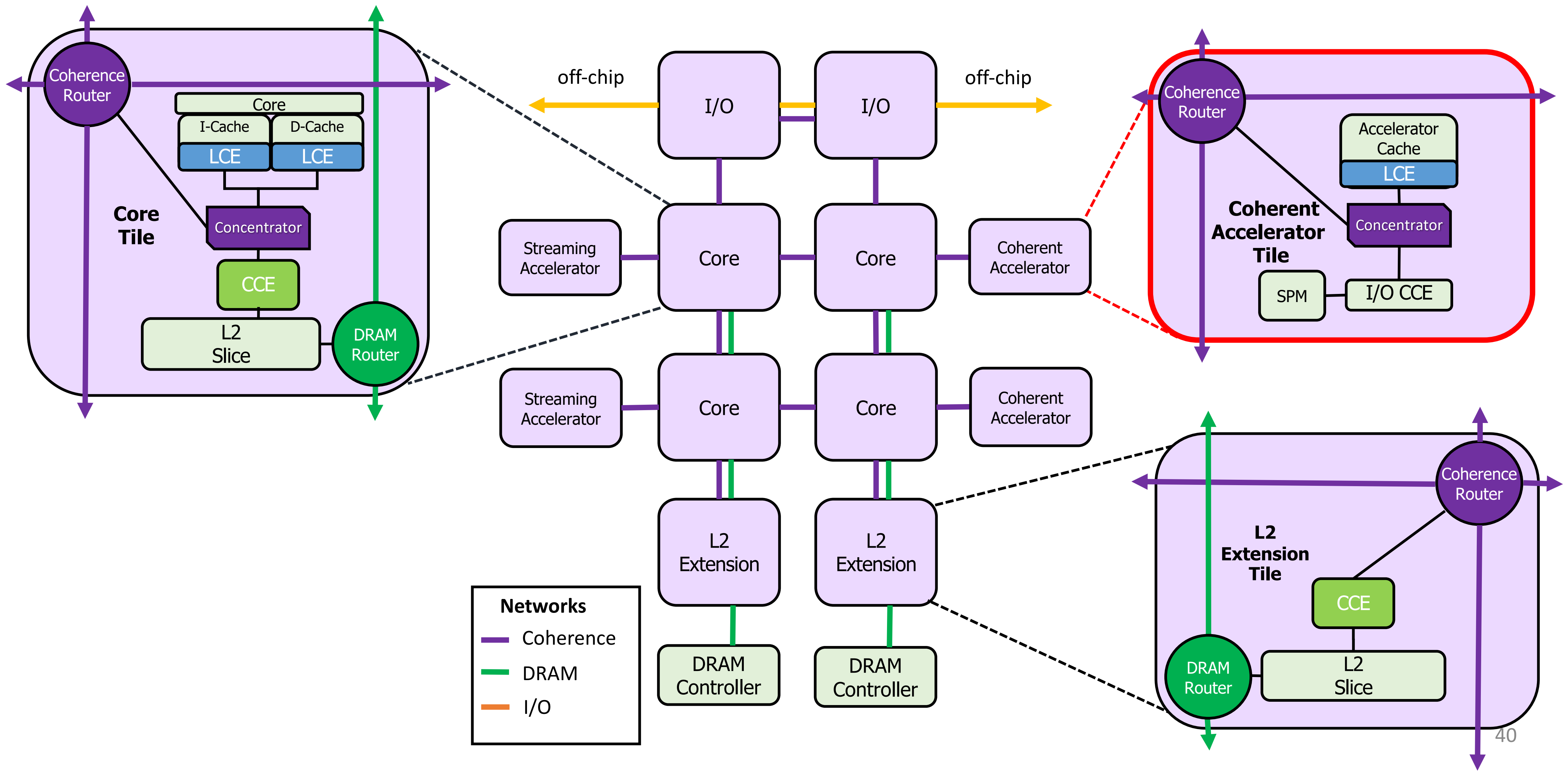
BlackParrot Supports Diverse Tile Types



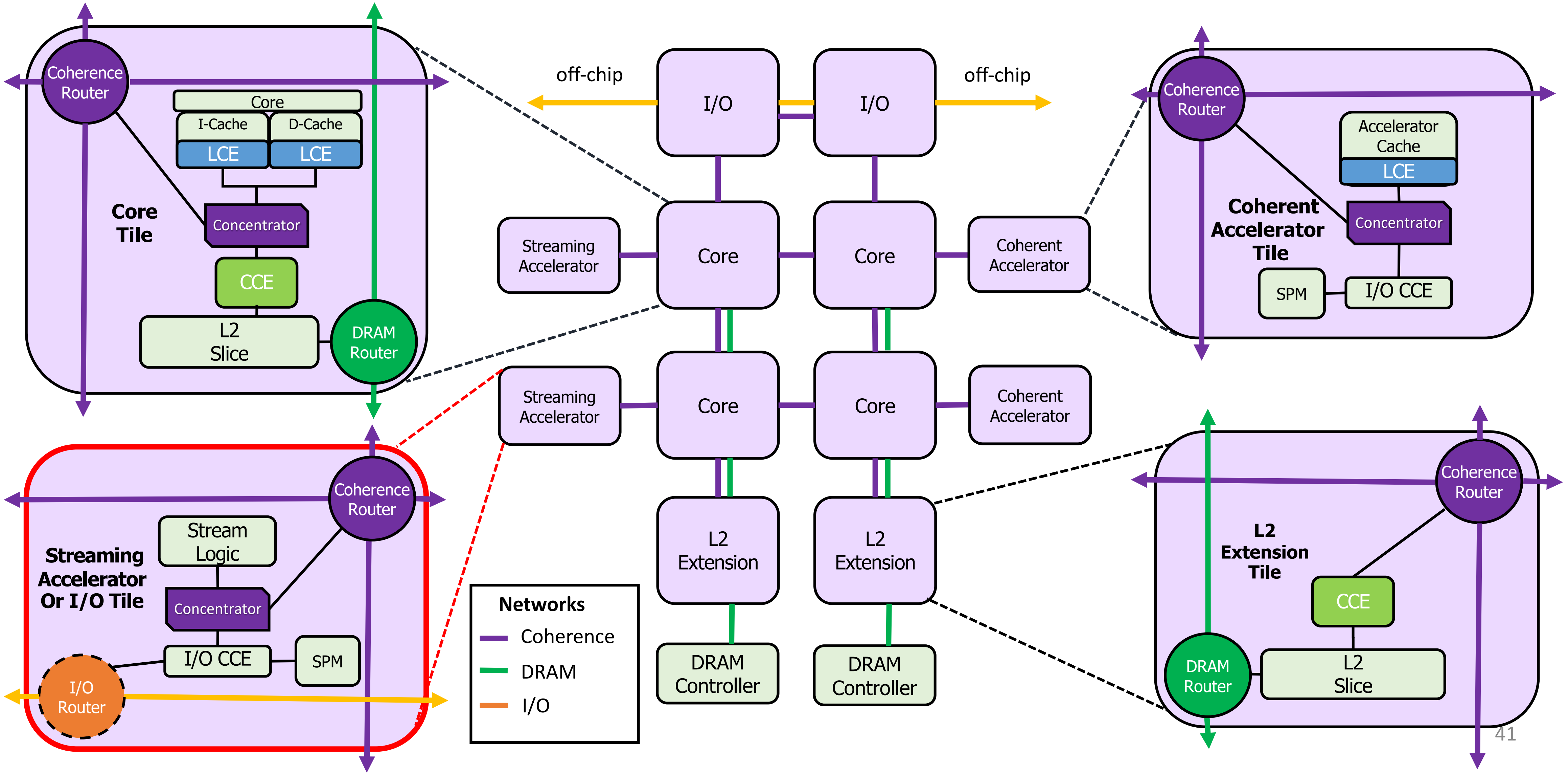
BlackParrot Supports Diverse Tile Types



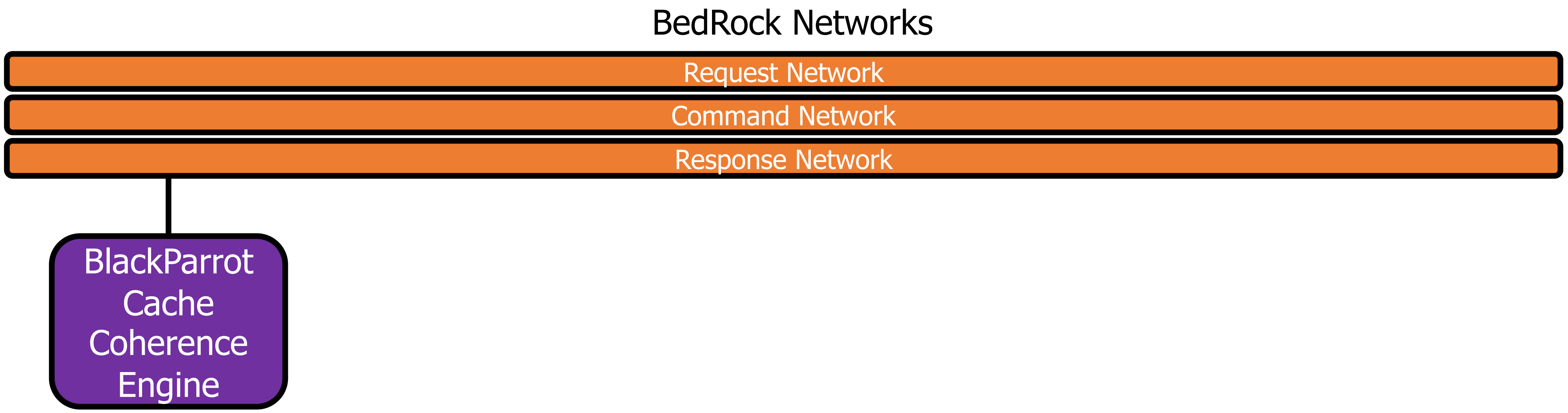
BlackParrot Supports Diverse Tile Types



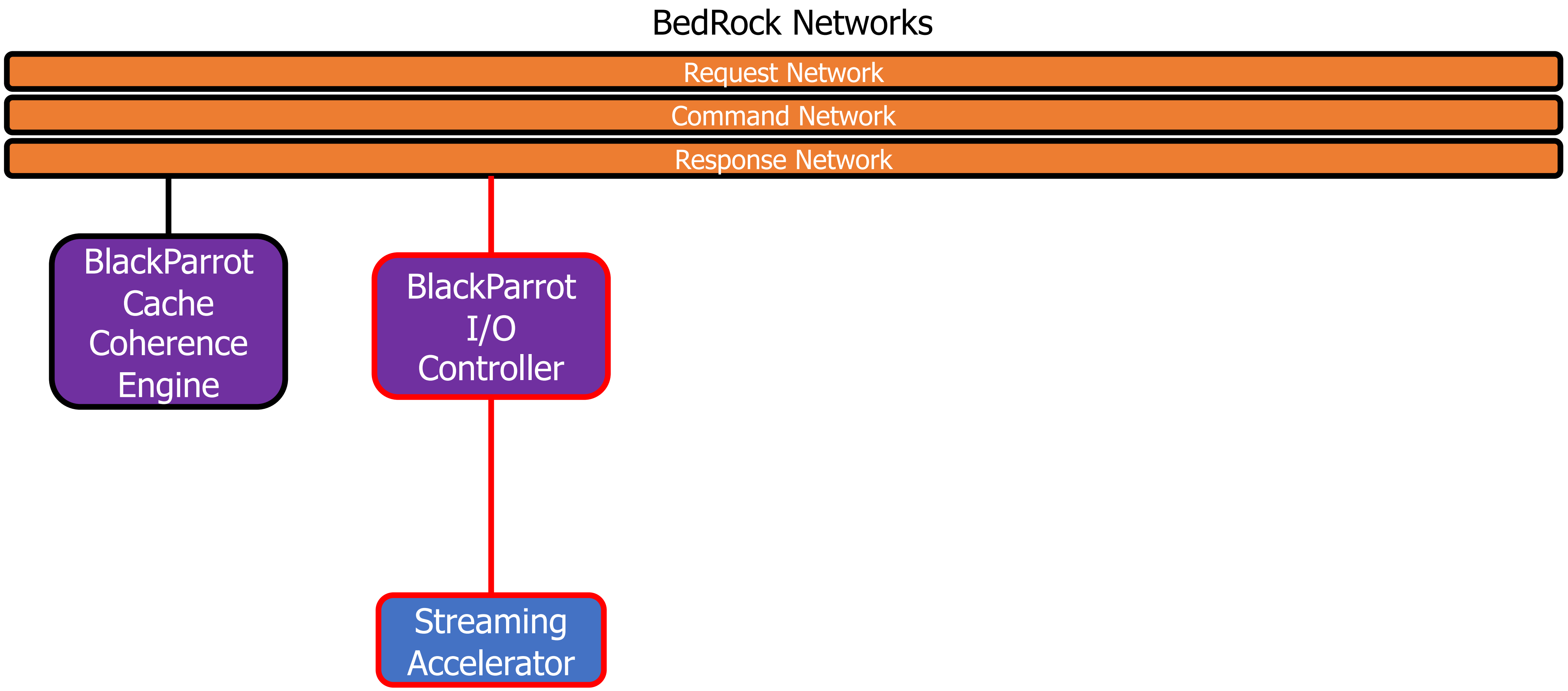
BlackParrot Supports Diverse Tile Types



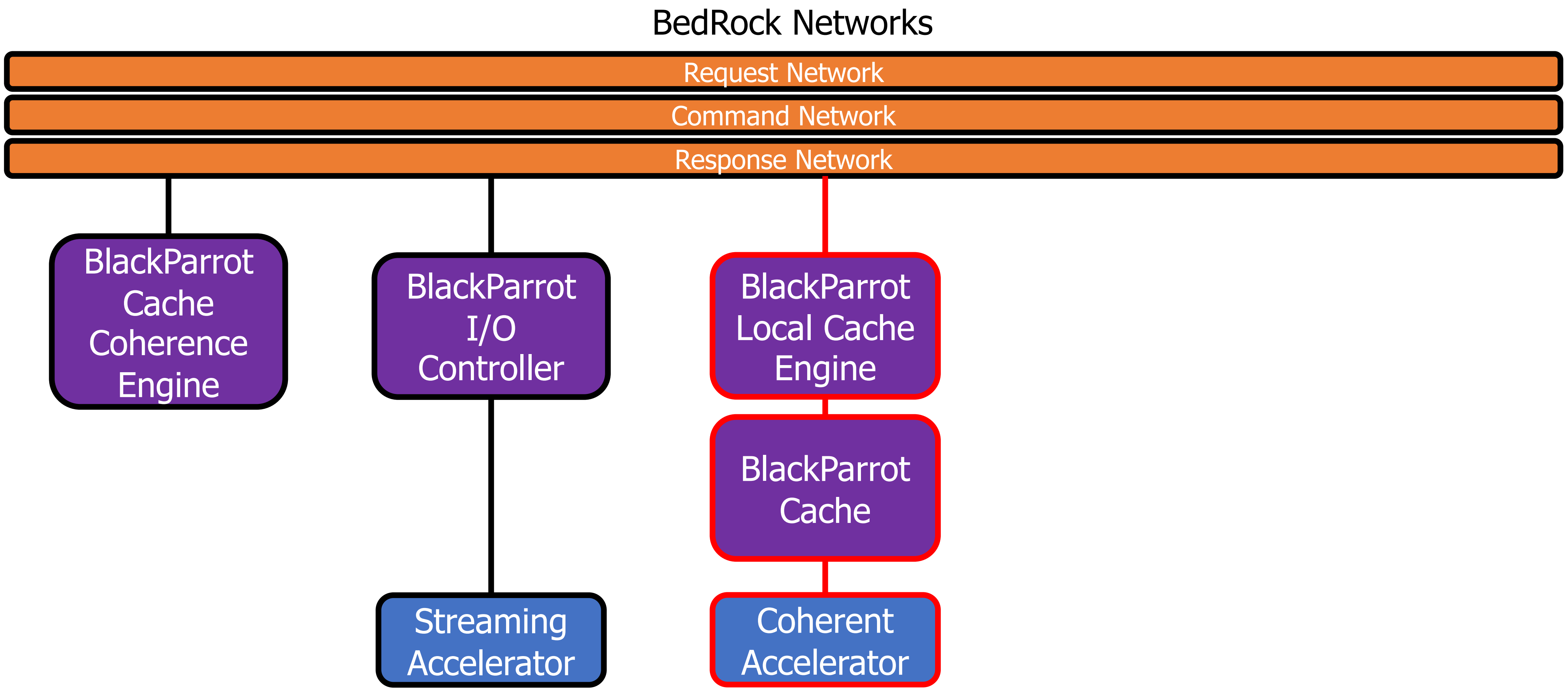
BP Supports Several Accelerator Integration Strategies



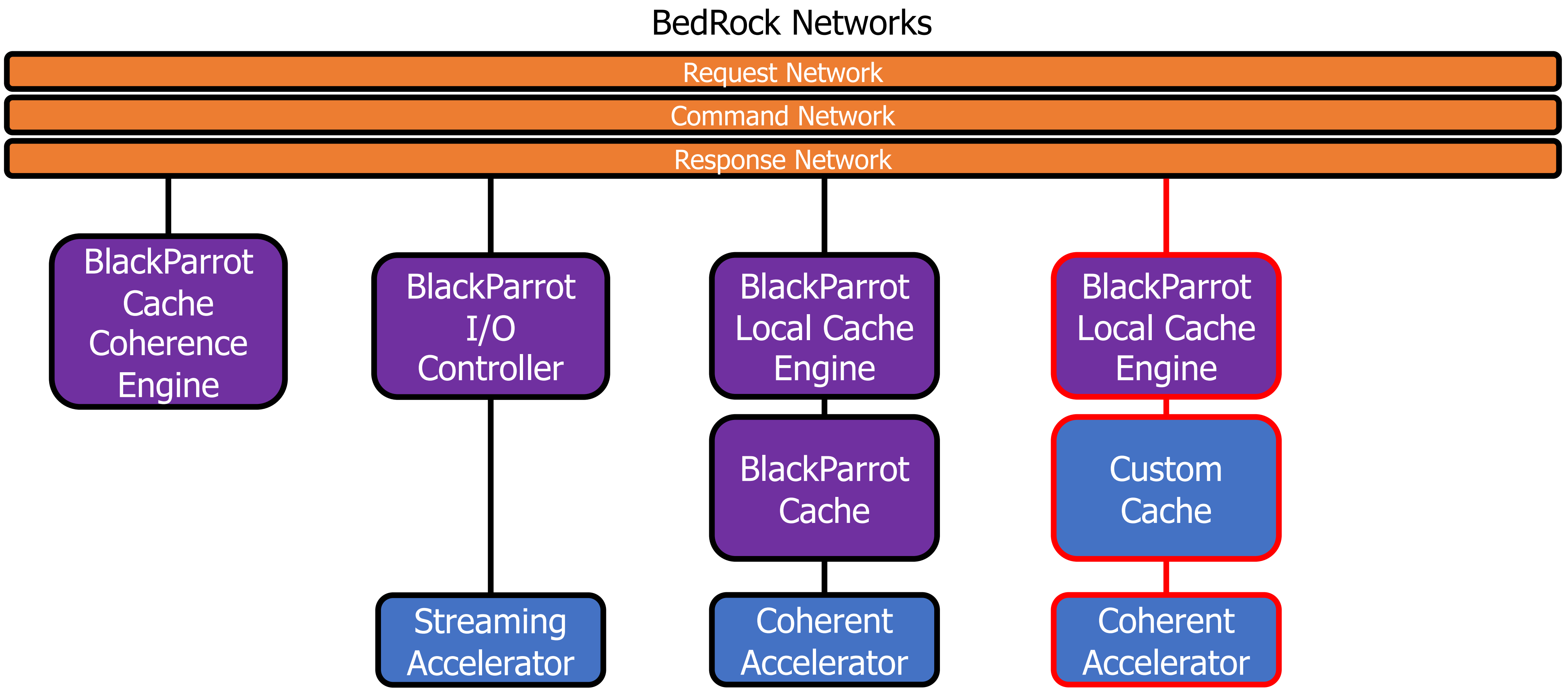
BP Supports Several Accelerator Integration Strategies



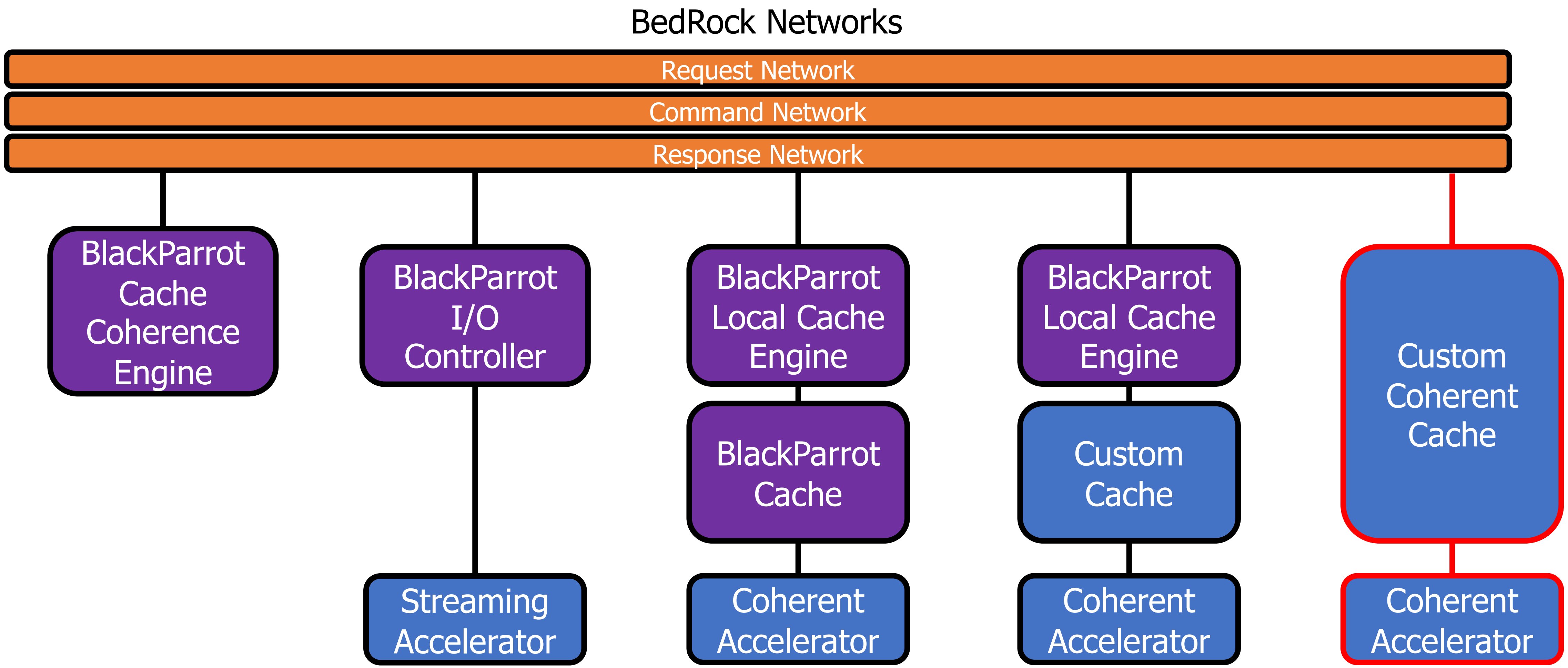
BP Supports Several Accelerator Integration Strategies



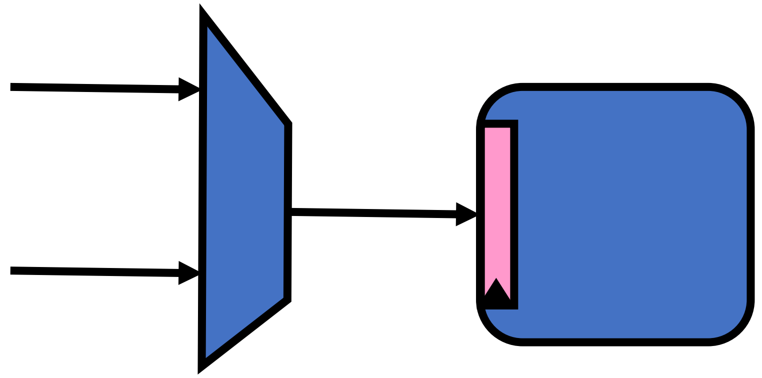
BP Supports Several Accelerator Integration Strategies



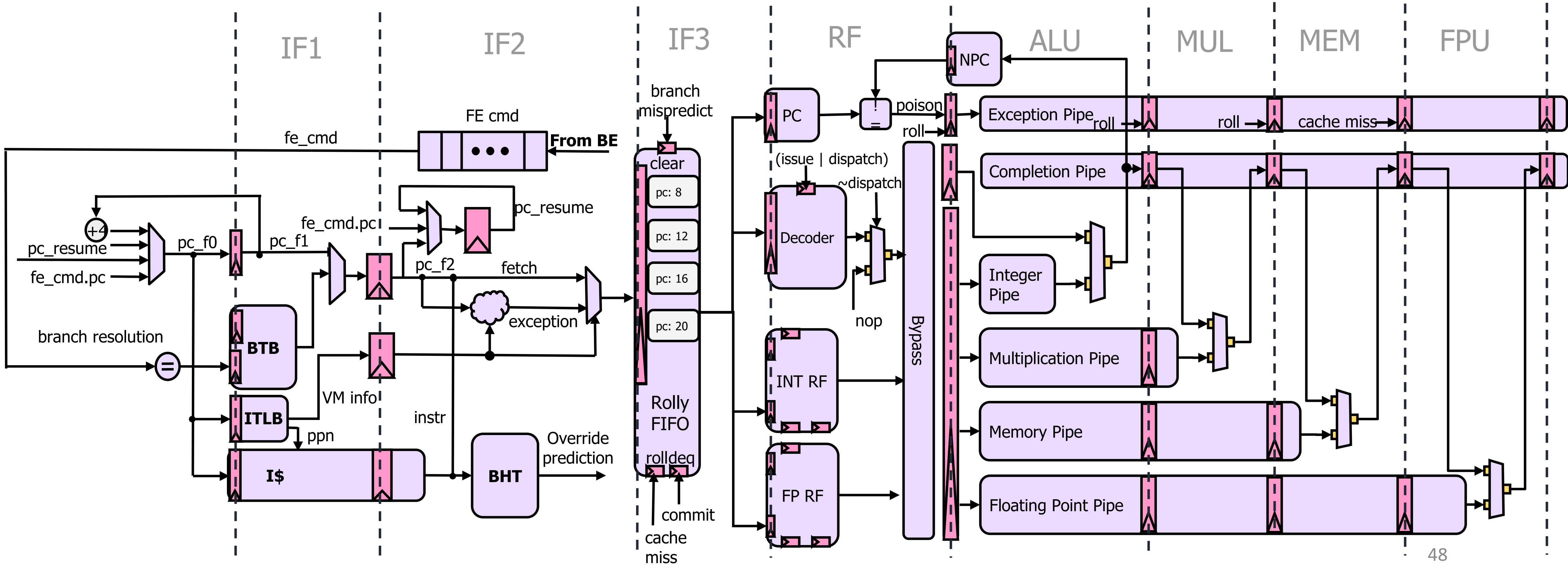
BP Supports Several Accelerator Integration Strategies



Core Microarchitecture



Efficient, In-order Core Pipeline

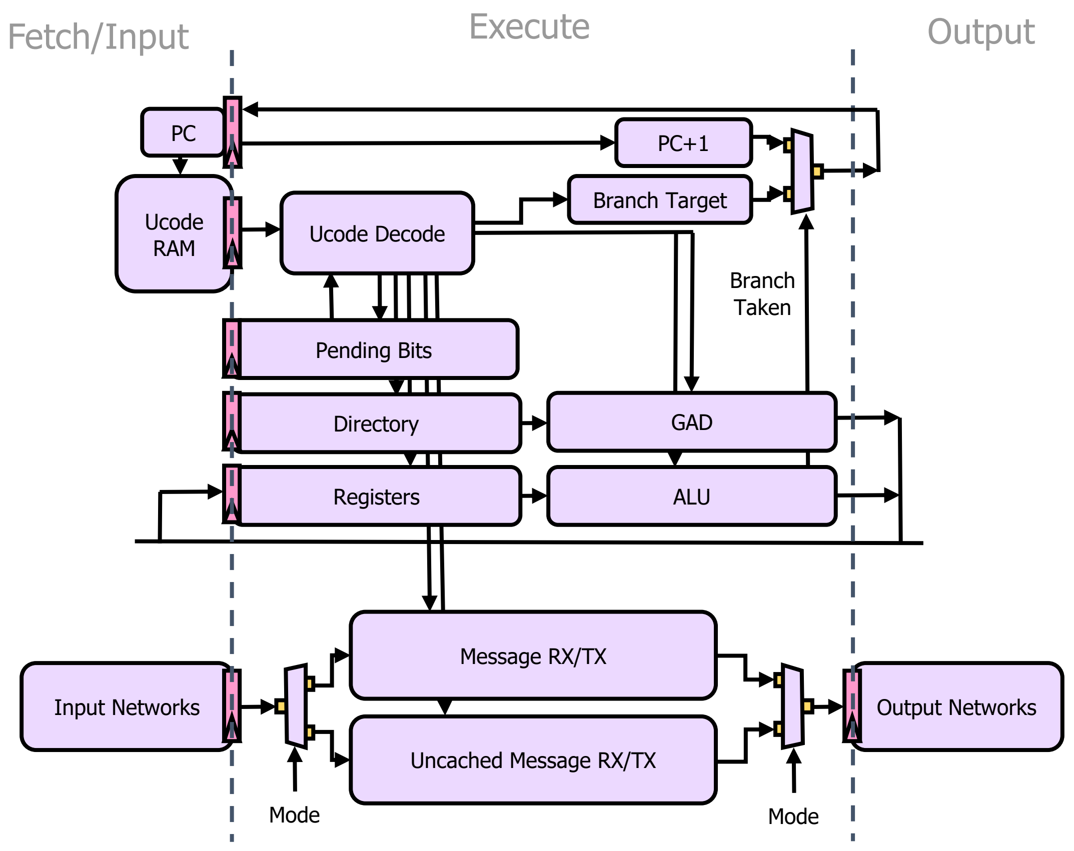


BedRock: A Programmable Cache Coherence Engine

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

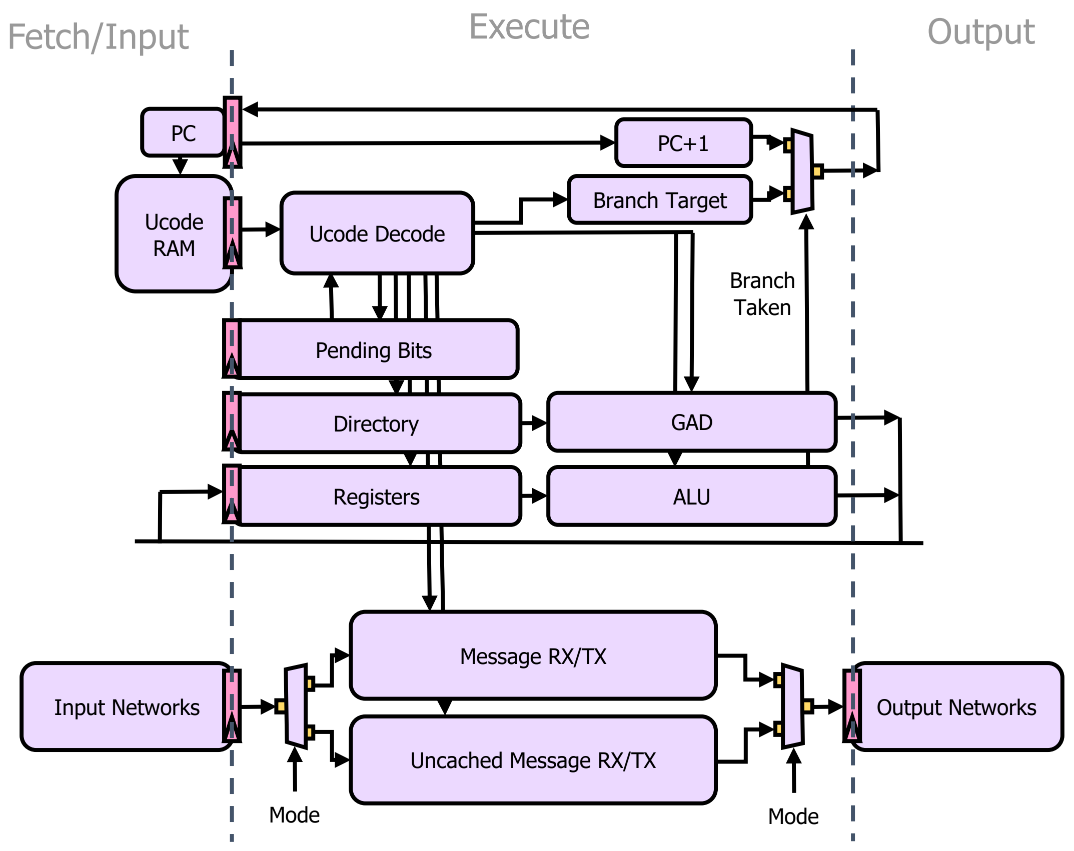


BedRock: A Programmable Cache Coherence Engine

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!



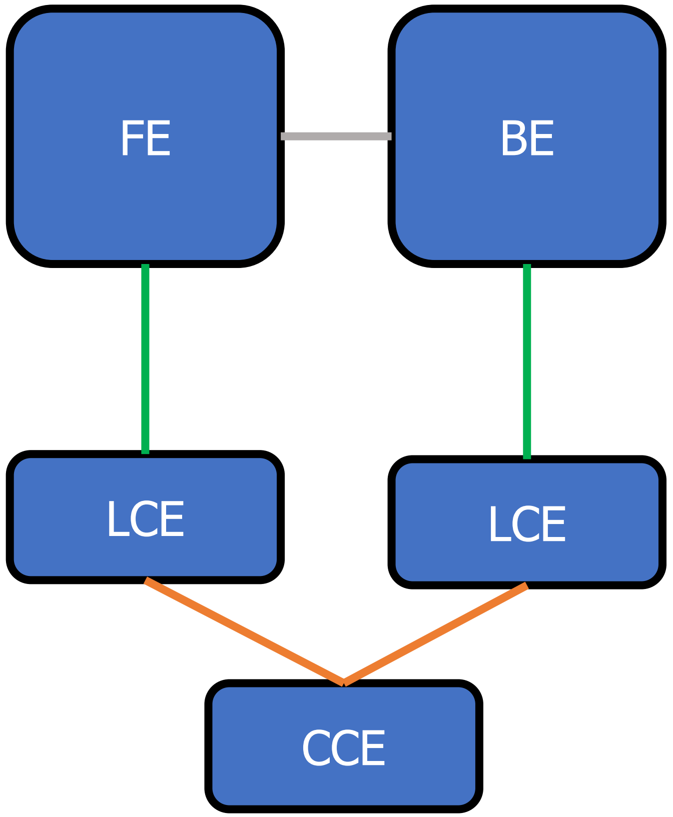
Standardized, Flexible, Latency-Insensitive Interfaces

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



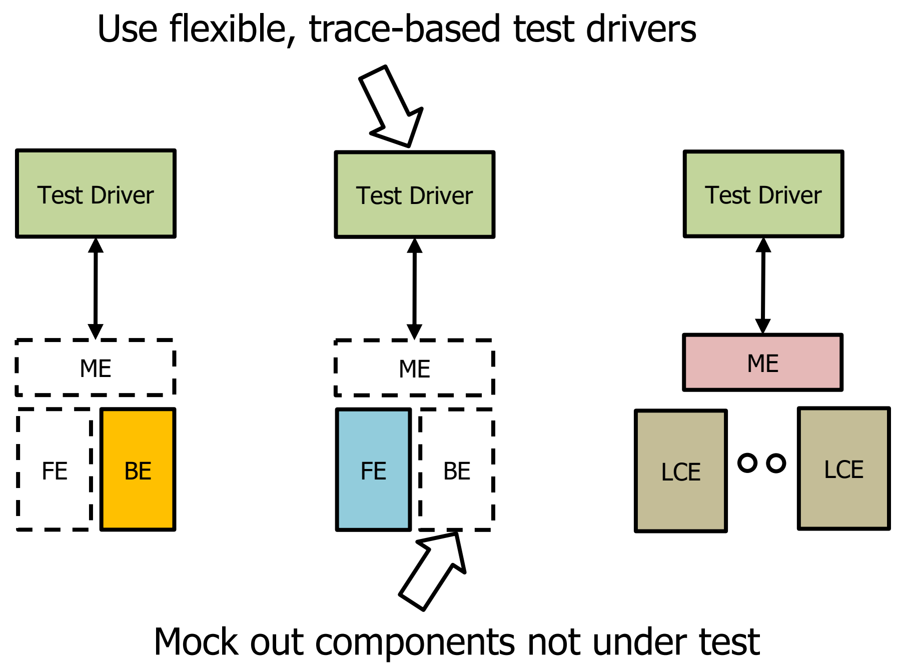
BlackParrot Community

A Modular Hardware Compliance Test Suite

Testbenches are expensive, but tests are invaluable

Maintain a small number of high value, flexible testbenches using mocks

Users can enhance any component of BlackParrot and quickly validate their changes


Use flexible, trace-based test drivers
Test Driver
Test Driver
Test Driver
ME
ME
ME
FE
BE
FE
BE
LCE
LCE
Mock out components not under test

BlackParrot: Community Driven Microarchitecture

Encourage users to become developers

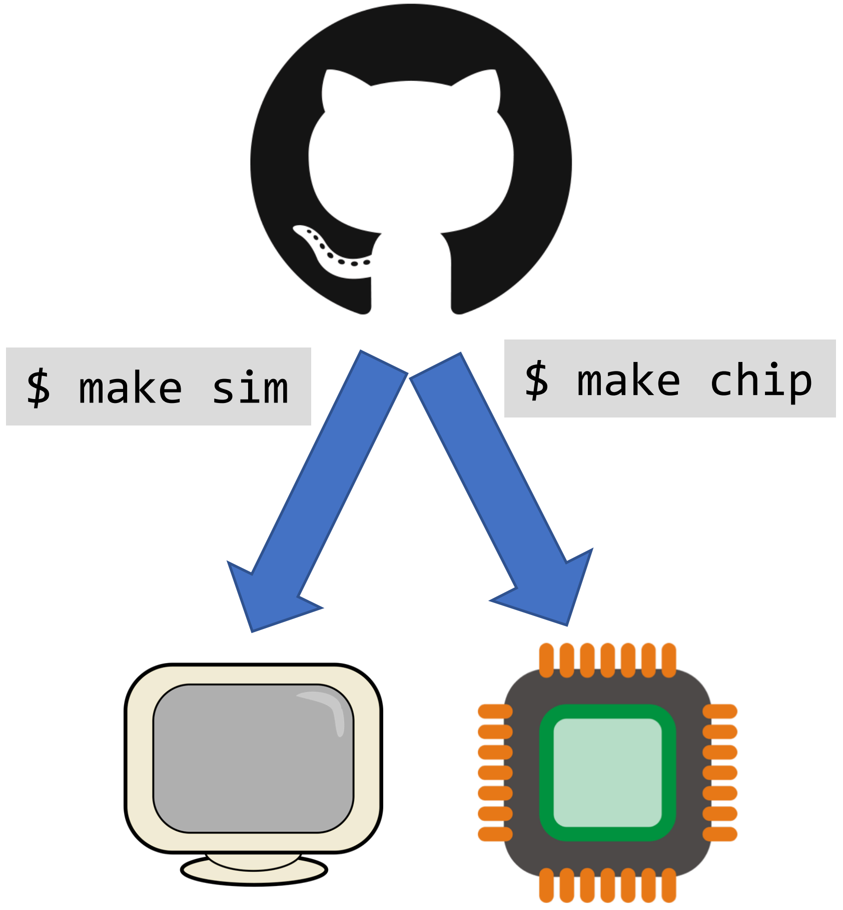
- Able to extend the system with cursory understanding of architecture
- Prioritize clarity over optimization

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



BlackParrot is FPGA-Validated

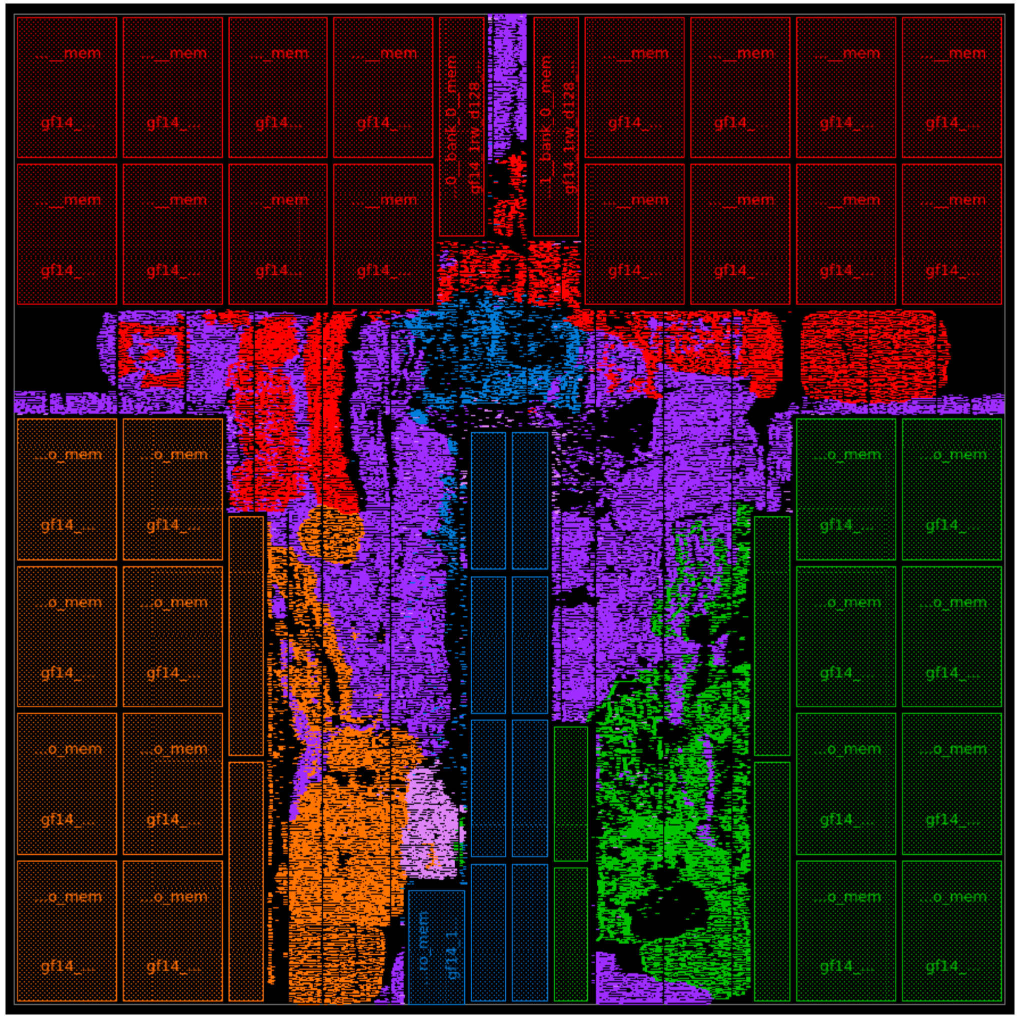
BlackParrot has been synthesized and tested on Xilinx Artix-7 FPGAs

Work is underway to integrate BlackParrot in the LiteX open-source FPGA environment

BlackParrot is Silicon-Validated

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

BlackParrot has also been ported to TSMC40 and FreePDK45 nodes



Fostering Community Around BlackParrot Backend Flows

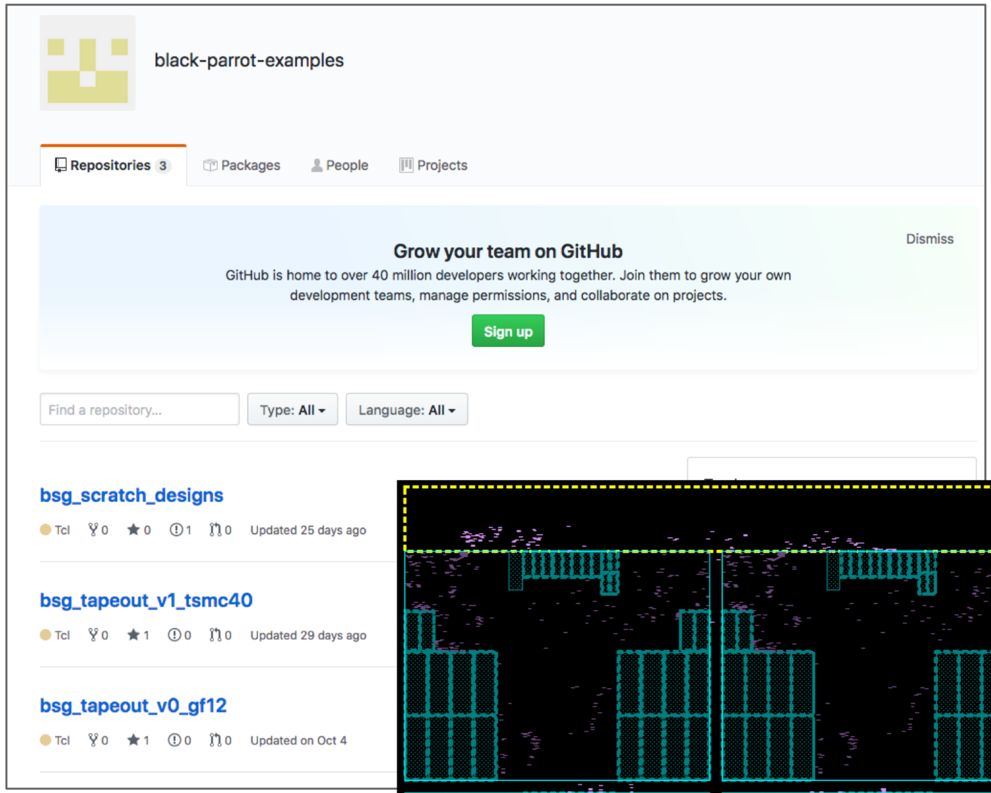
Open-source Tapeout Directories

- 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

Repositories 3 | Packages | People | Projects

Grow your team on GitHub

Dismiss

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

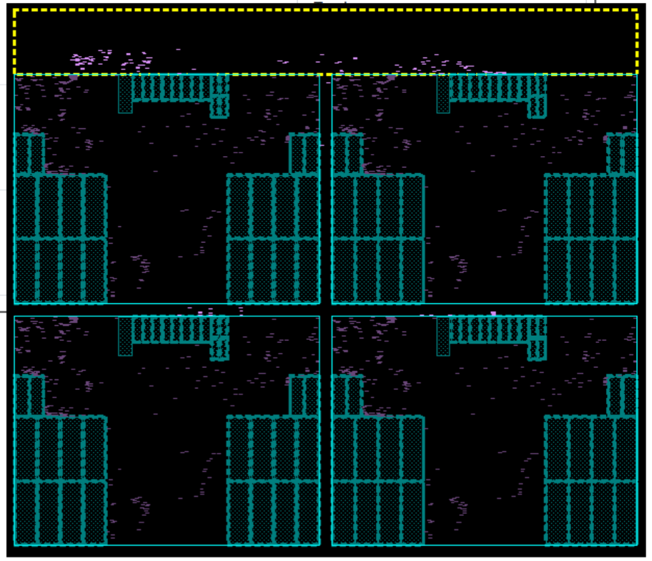
Sign up

Find a repository... | Type: All | Language: All

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

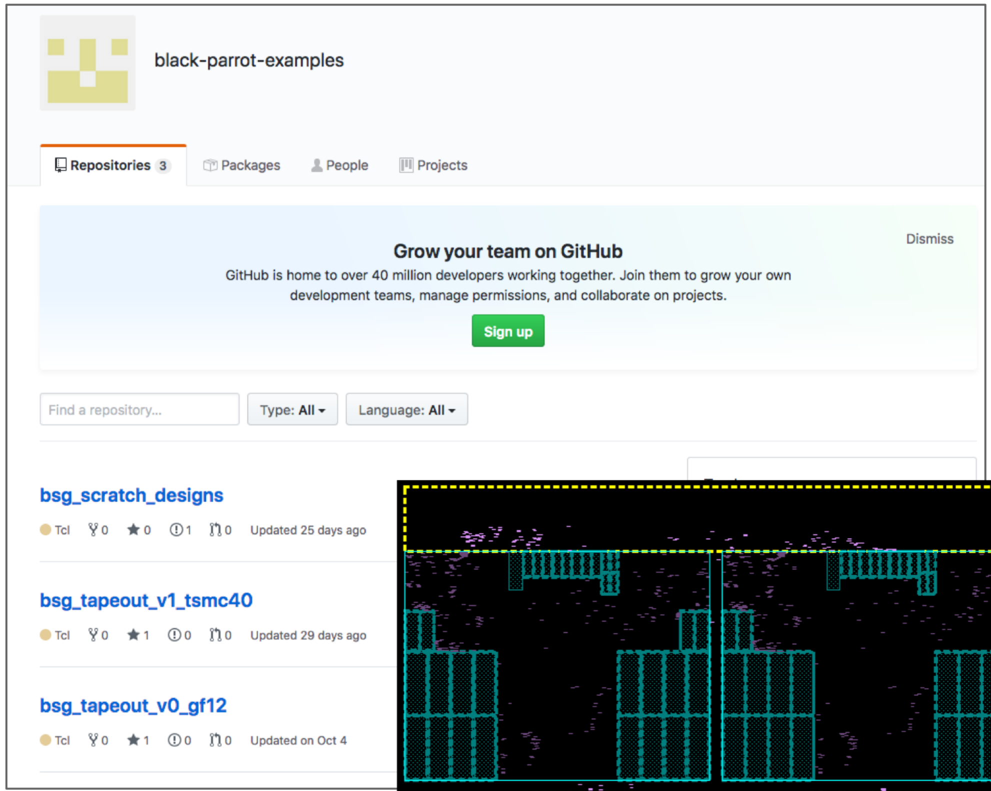
Open-source Tapeout Directories

- 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

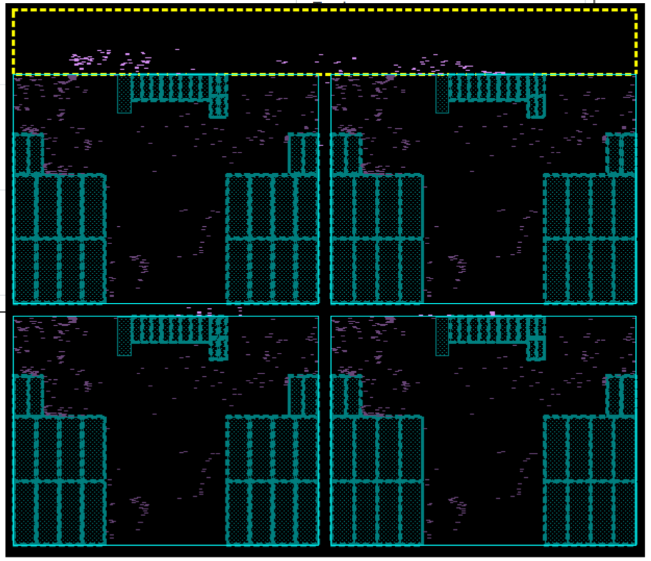
Grow your team on GitHub

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



BlackParrot "Genesis Release" Team

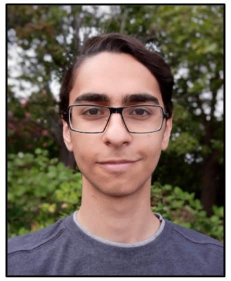
Prof. Michael Taylor

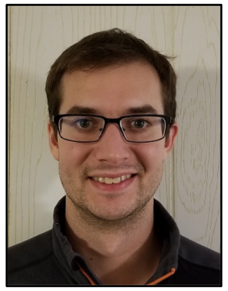
BOSTON UNIVERSITY

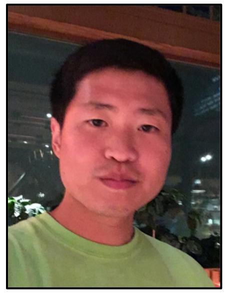
Prof. Ajay Joshi

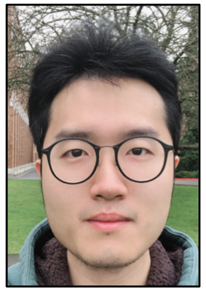
Prof. Mark Oskin

Dan Petrisko


Farzam Gilani


Mark Wyse


Tommy Jung

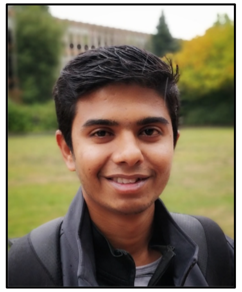

Paul Gao

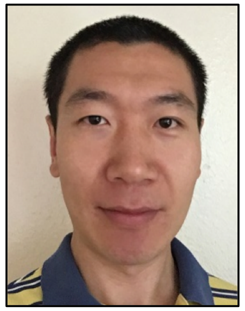
Sadullah Canakci

Zahra Azad

Scott Davidson

Yongqin Wang


Bandhav Veluri


Chun Zhao

Tavio Guarino

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!

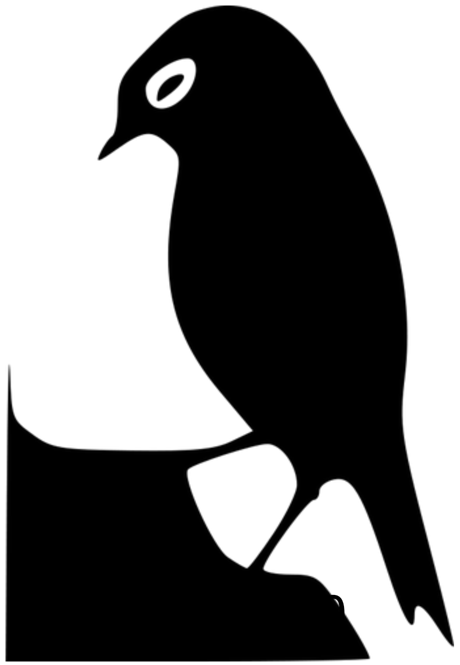
Please explore, use, break things and let us know your experience!

We salute you!

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.