

Architecture Design Space Exploration Using RISC-V

Donato Kava, Sahan Bandara, Michel A. Kinsy
Adaptive and Secure Computing Systems (ASCS) Laboratory
Boston University, Boston, MA

Additional Project Participants: Alan Ehret, Mihailo Isakov, Nikola Hardi

Presentation Outline

- Introduction
 - Motivation
 - Background
- Processors
- Memory subsystem
- Network-on-Chip
- BRISC-V ecosystem
 - Hermes: A secure heterogeneous multicore architecture
 - Janus: An uncertain cache architecture
 - Sphinx-V: A software-hardware obfuscation architecture
- Conclusion

BRISC-V Vision & Motivation

- The design of multi/many-core systems requires tuning of a large number of parameters: core computation power, memory hierarchy, topology, routers, routing algorithms, area, power, among others
- The two approaches for system micro-architecture exploration
 - Software simulation
 - Hardware emulation

Design Space Exploration

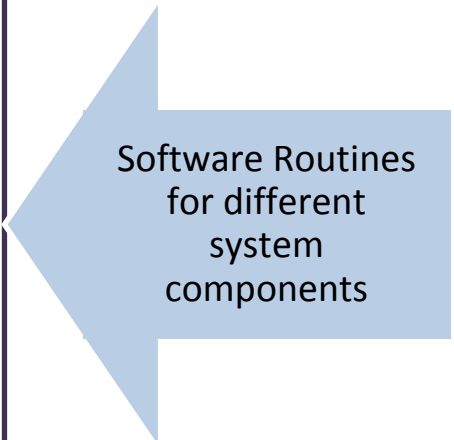
Advantages:

- Large programming tool support
- Internal states of Modules easily accessible
- Fast Compilation
- Less Constraints in terms of number of components to simulate

Disadvantages:

- Time prohibitive for many micro-architecture explorations
- Evaluation accuracy tradeoff with execution speed

Two Current Approaches



Software Routines
for different
system
components

Design Space Exploration

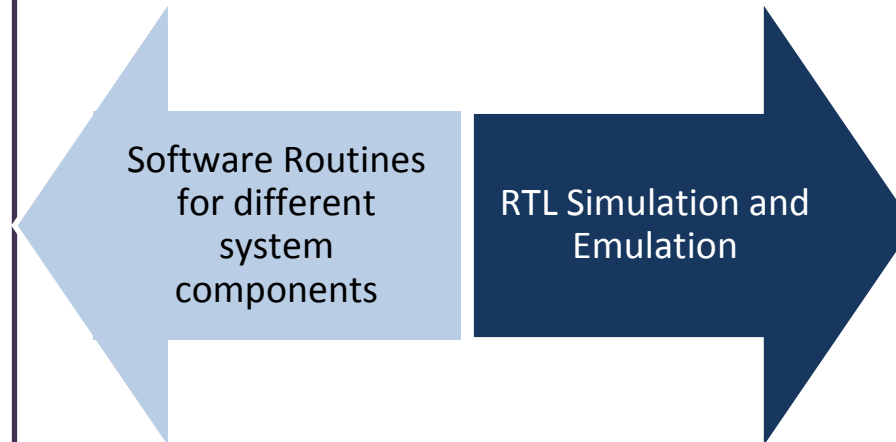
Advantages:

- Large programming tool support
- Internal states of Modules easily accessible
- Fast Compilation
- Less Constraints in terms of number of components to simulate

Disadvantages:

- Time prohibitive for many micro-architecture explorations
- Evaluation accuracy tradeoff with execution speed

Two Current Approaches



Advantages:

- Highly Accurate
- Less system mischaracterization
- Avoids late discovery of system performance bugs

Disadvantages:

- Simulation time increases with design time

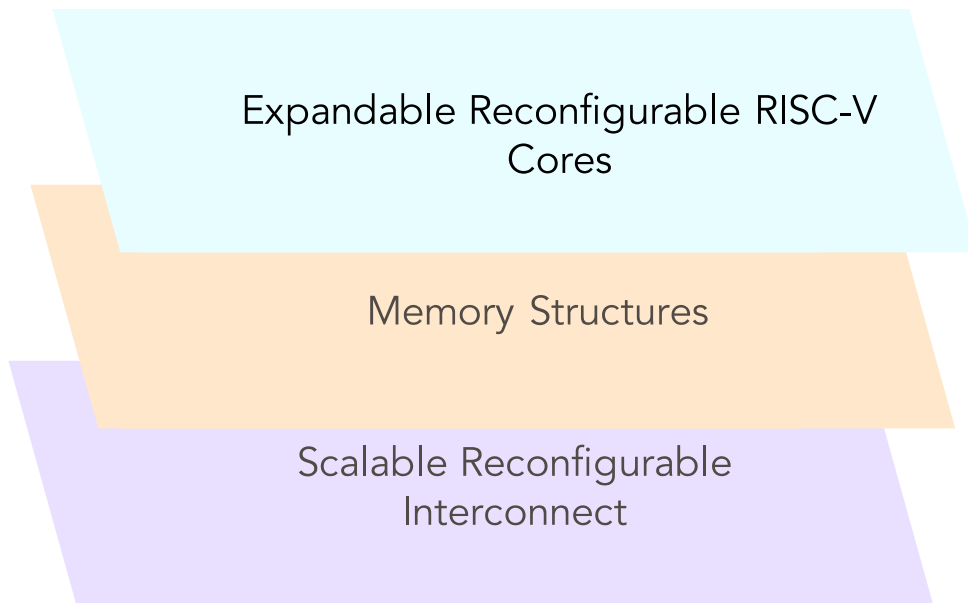
Solution:

- This can be circumvented by using synthesizable RTL and FPGAs to accelerate execution
- BRISC-V pre-designed, customizable, soft hardware modules

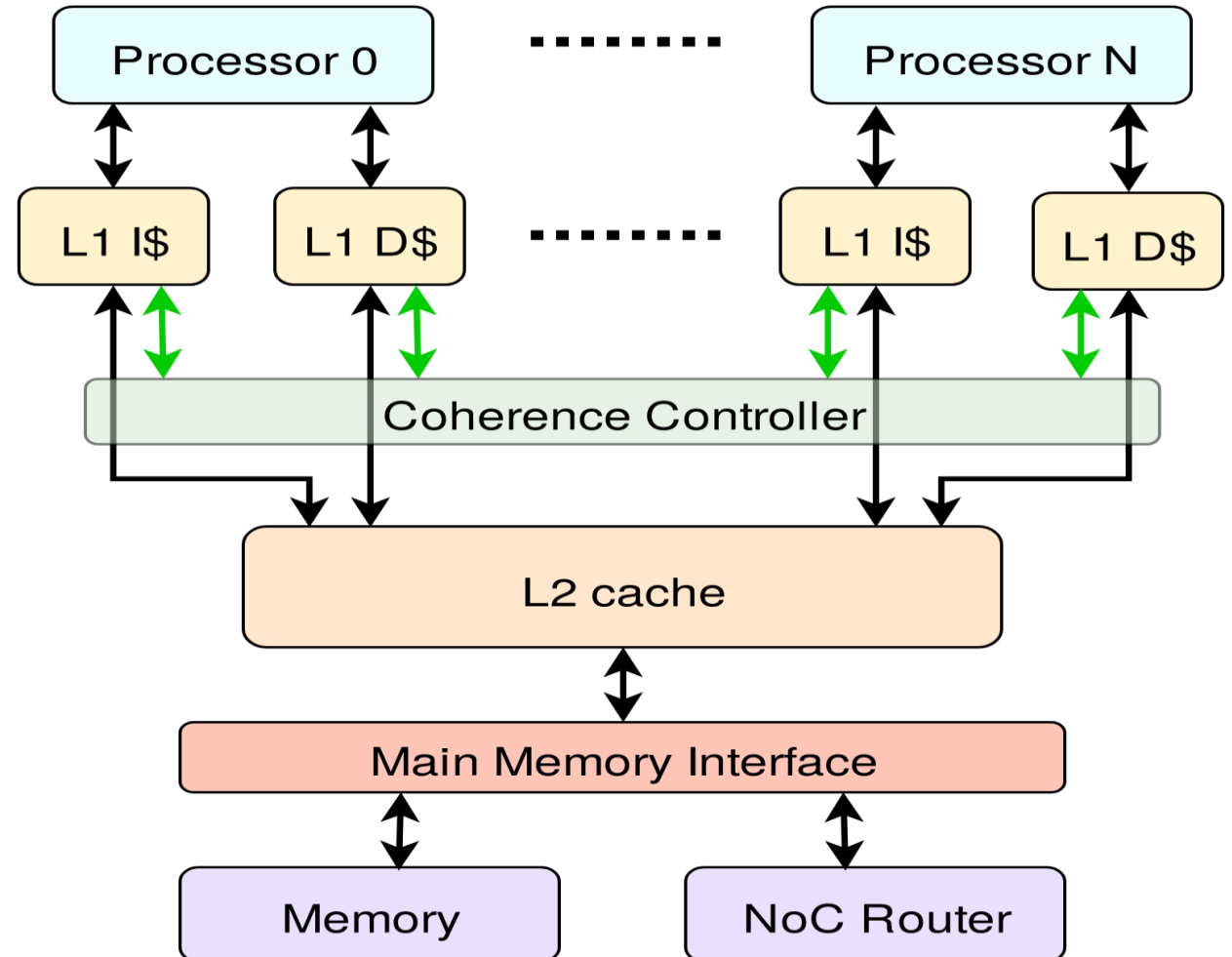
BRISC-V Vision & Motivation

- Multicore architectures and SoCs design exploration platform with RTL level correctness
- Provide a set of design templates with clear interfaces on which, a designer can build upon
- Different levels of complexity for different requirements
- Configurability provides higher flexibility in terms of design trade-offs
- Synthesizable RTL for faster design verification
- A readily available design base
 - Not meant as a final design but a starting point for exploration.

BRISC-V Toolbox

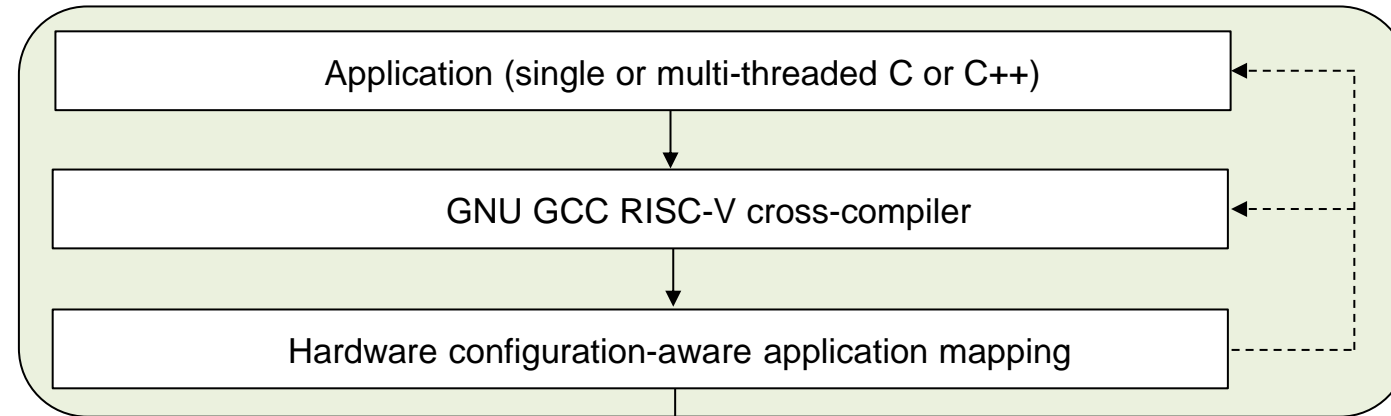


- It is a full system design exploration and development toolbox

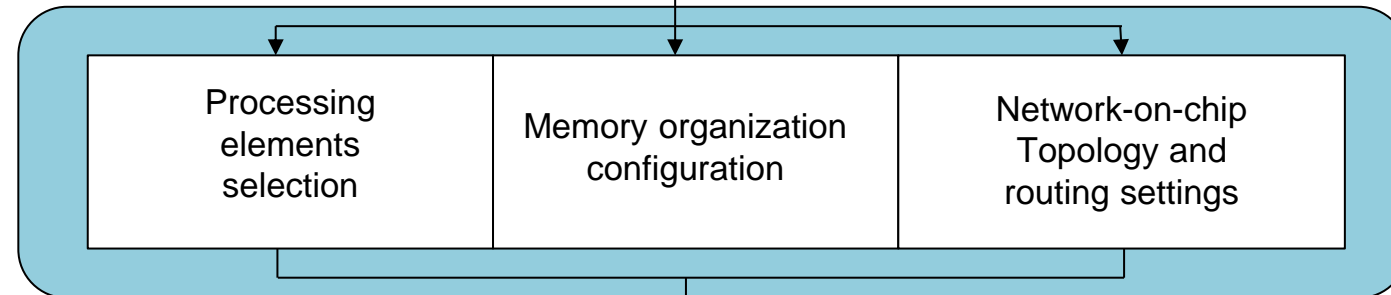


BRISC-V Design Infrastructure

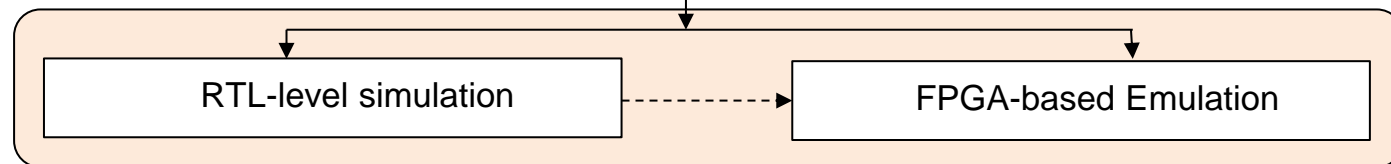
Software Environment



Component-based Hardware Design

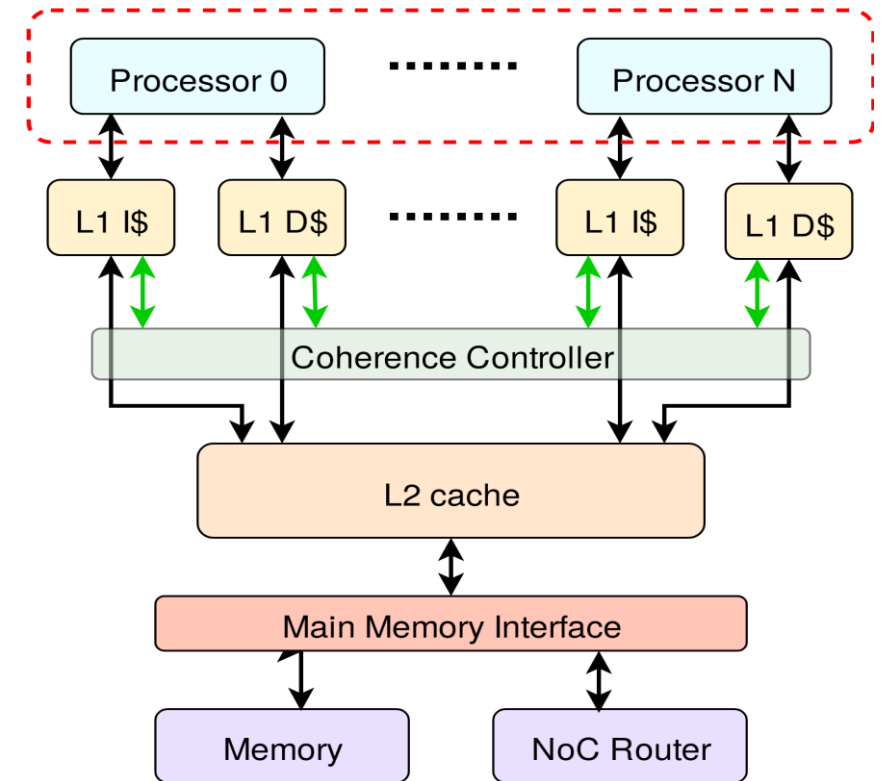


Evaluation Environment



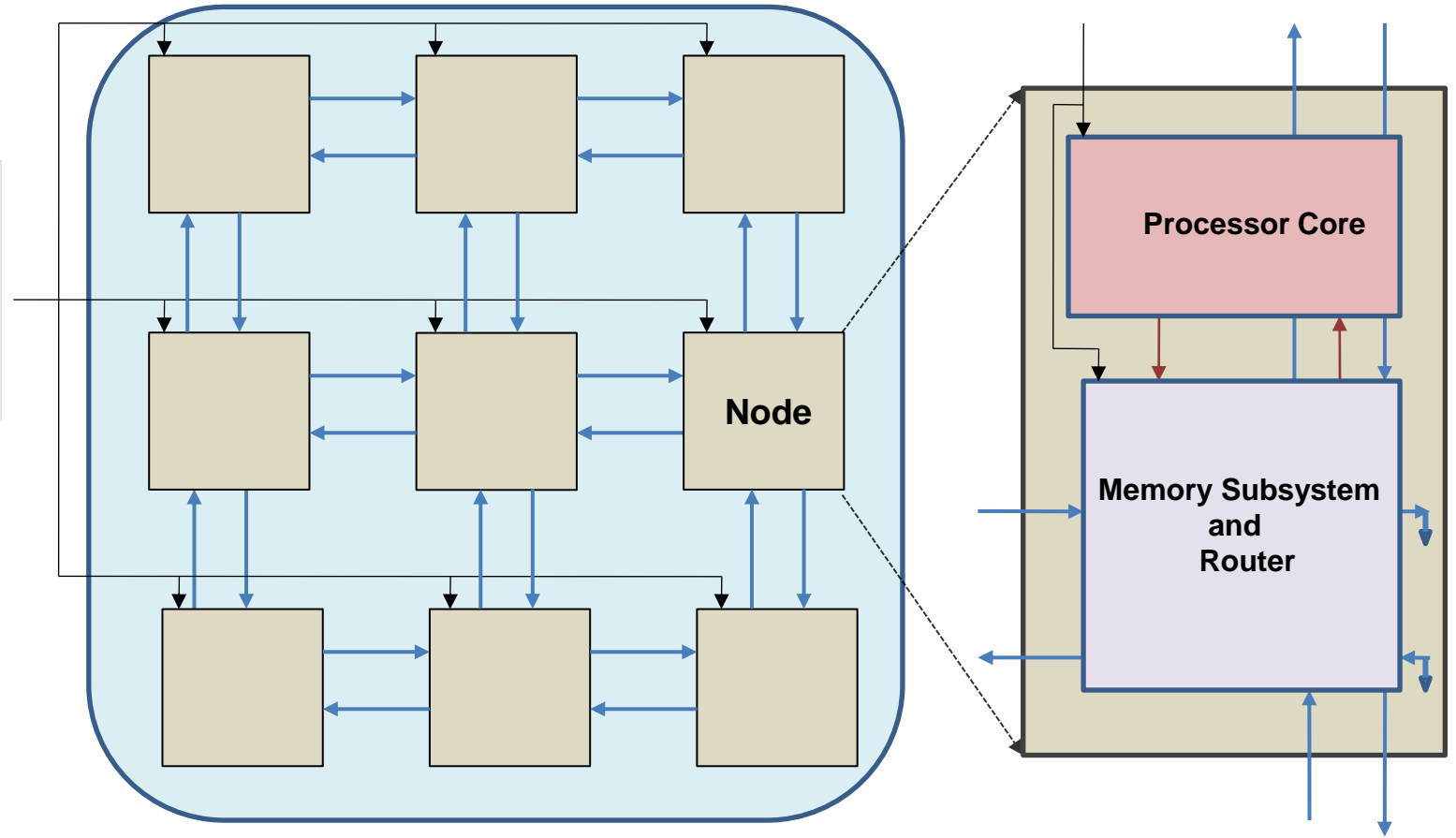
Presentation Outline

- Introduction
- Processors
 - Single Cycle RISC-V Core
 - 5 and 7 – Cycle RISC-V Cores
 - Out-of-Order RISC-V Core
 - Hardware-Threaded RISC-V Cores
- Memory subsystem
- Network-on-Chip
- BRISC-V ecosystem
- Conclusion



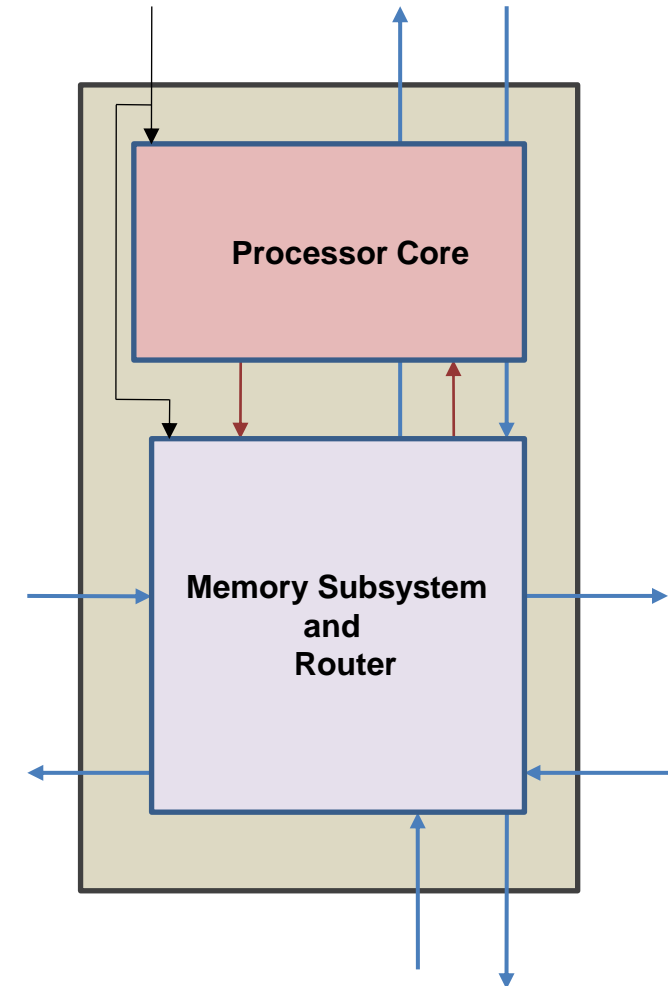
System Mesh Architecture Instance

- Binaries
- Routing table data
- Starting PC per core



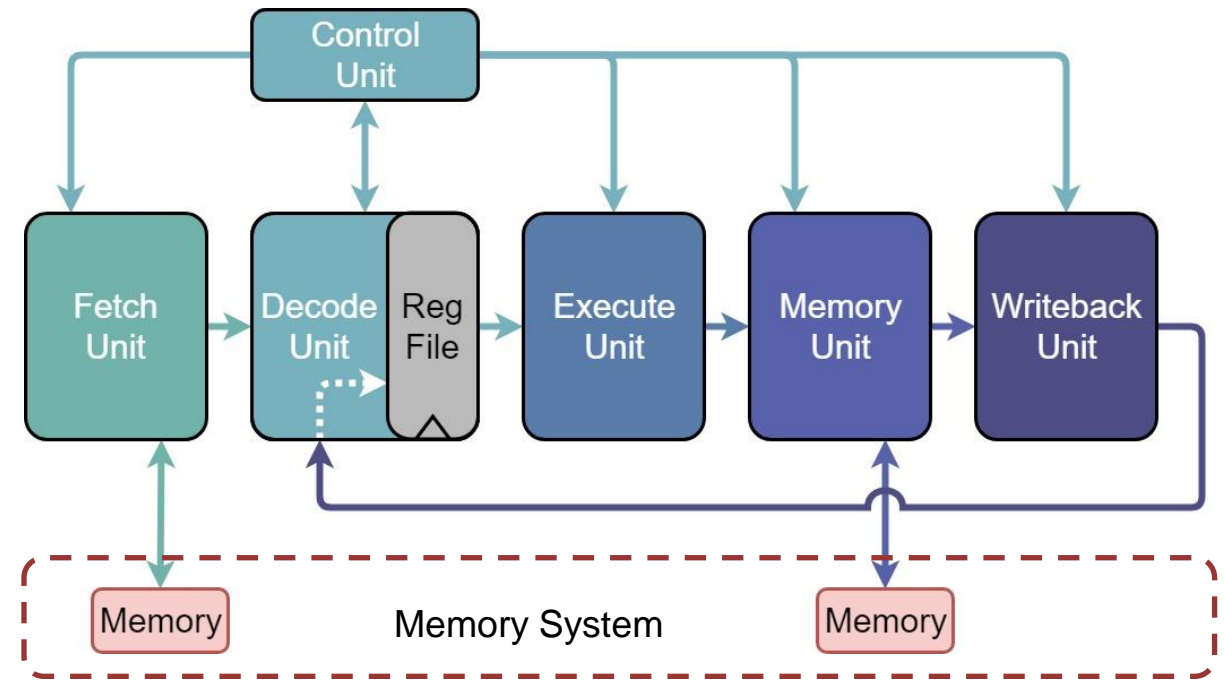
BRISC-V Processor Cores

- The processor cores are oblivious to the memory hierarchy and the network topology
- Different cores with the same interface - all base RV32I
 - Single, Five and Seven cycle(s)/ stage(s)
 - Out-of-Order
 - Parameterized for easy configuration changes
- A cores can be replaced with no change to the rest of the platform



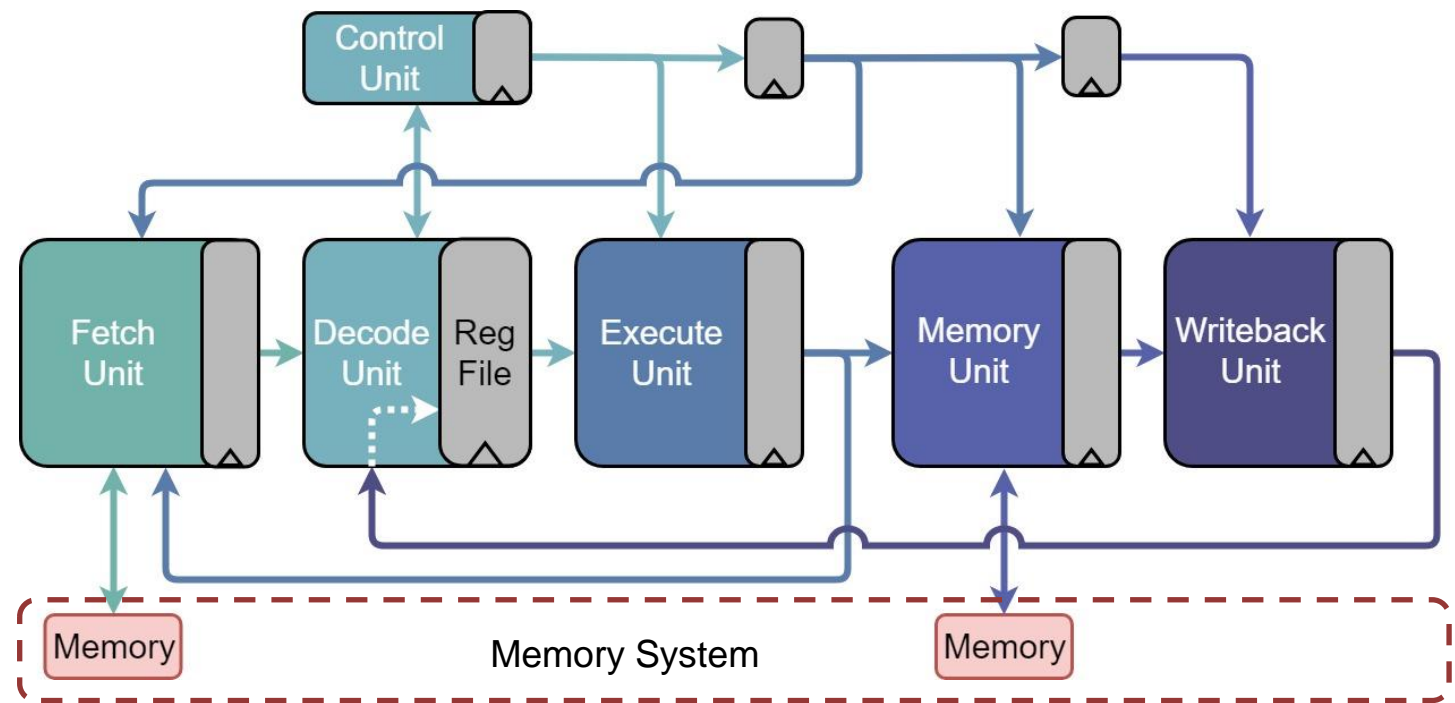
Single Cycle Processor

- Smallest, lowest complexity
- Single cycle – therefore limit the use of block ram implementation
 - Address space limited by target FPGA
- Execution steps in the single cycle
 - IF + ID + EX + DM + WB



Five Cycle Processor

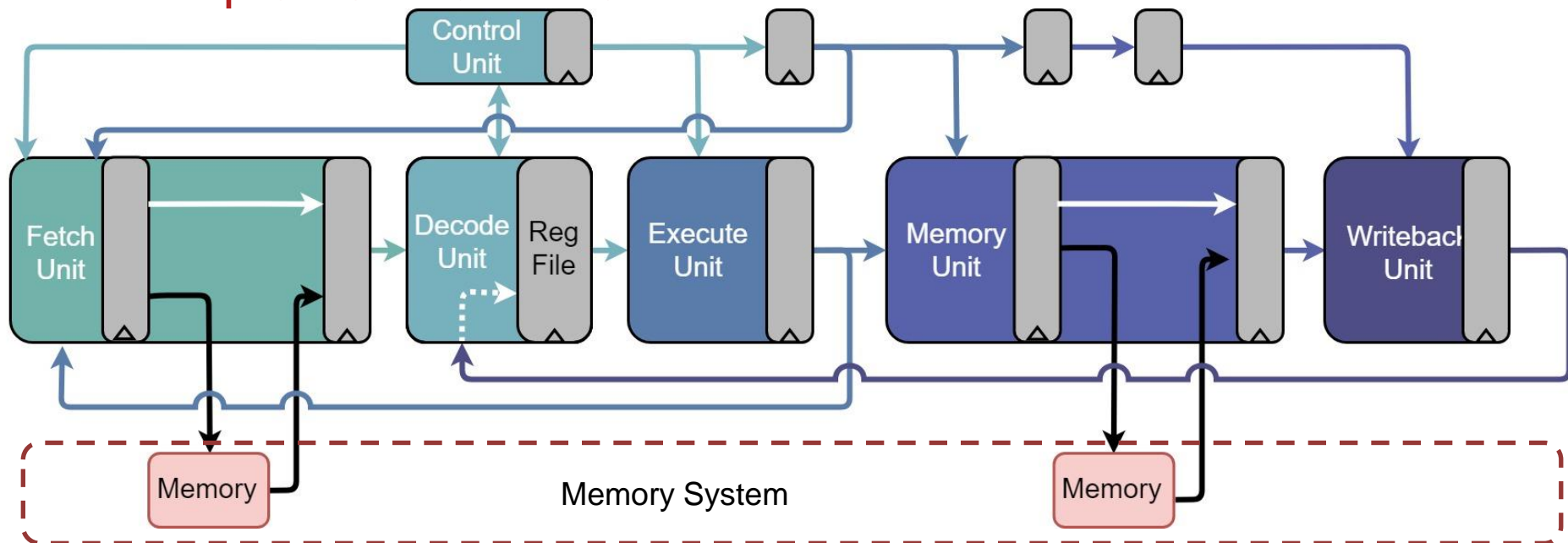
- Five stage pipelining
- Expanded control logic
 - Register forwarding and pipeline stall
- Execution step in the single cycle
 - IF | ID | EX | DM | WB
 - FPGA implementation is still limited



Seven Cycle Processor

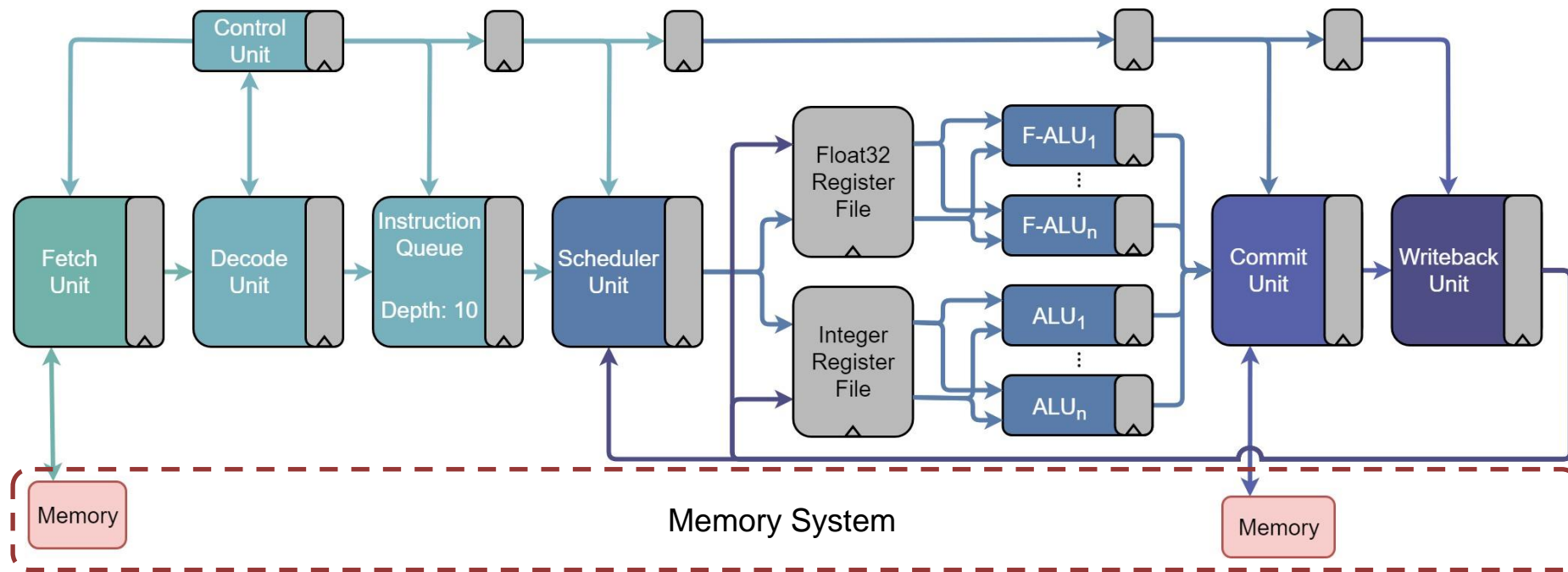
■ Complex memory subsystem support

- Two-cycle memory stage - IF_i | IF_r | ID | EX | DM_i | DM_r | WB
- Cache integration support
- One can now implement full Block-RAM



Out-of-Order Processor

- Out of order execute, in order commit
- Expanded modules
 - Instruction queue, scheduler, commit

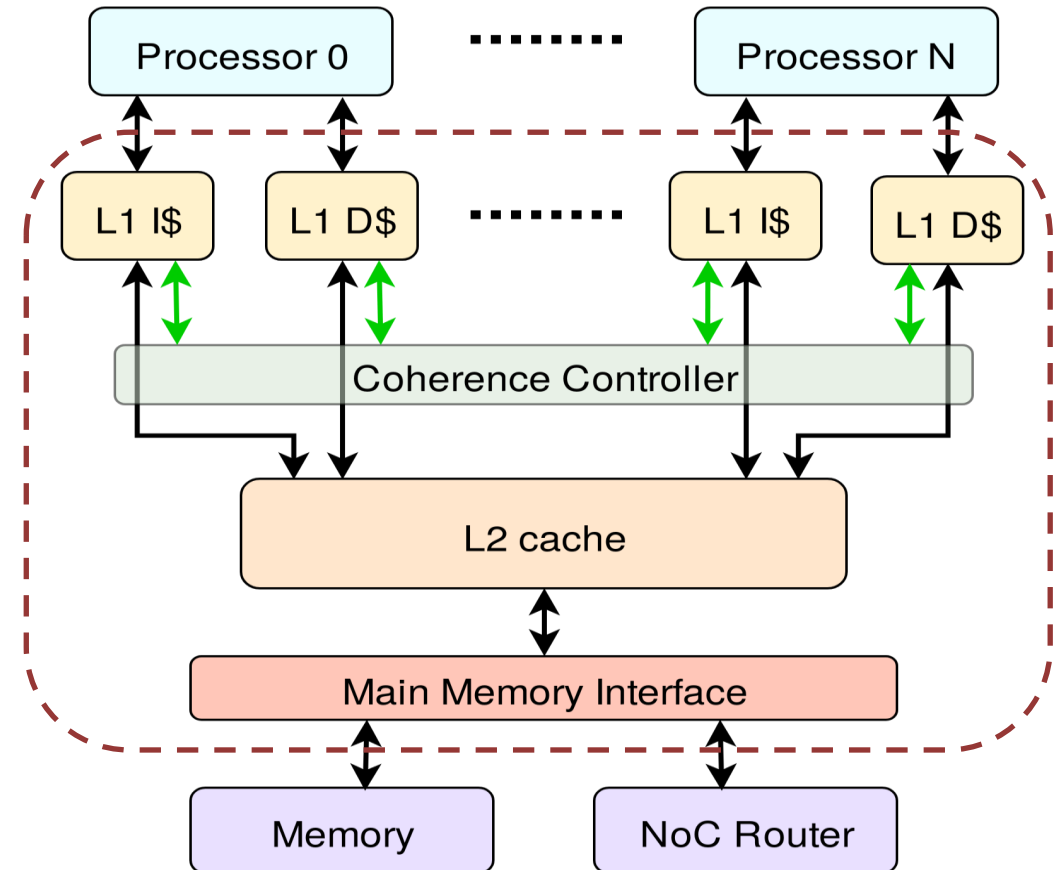


Cores Synthesis Results

Core Type	Address Bits	Logic Elements	Total BRAM bits	Worst case Fmax
Single Cycle	10	53,448	0	29.0 MHz
Five stage stalled	12	3,160	262,144	64.8 MHz
Five stage forwarded	12	3,406	262,144	61.5 MHz
Seven cycle forwarded	12	3,497	262,144	68.6 MHz

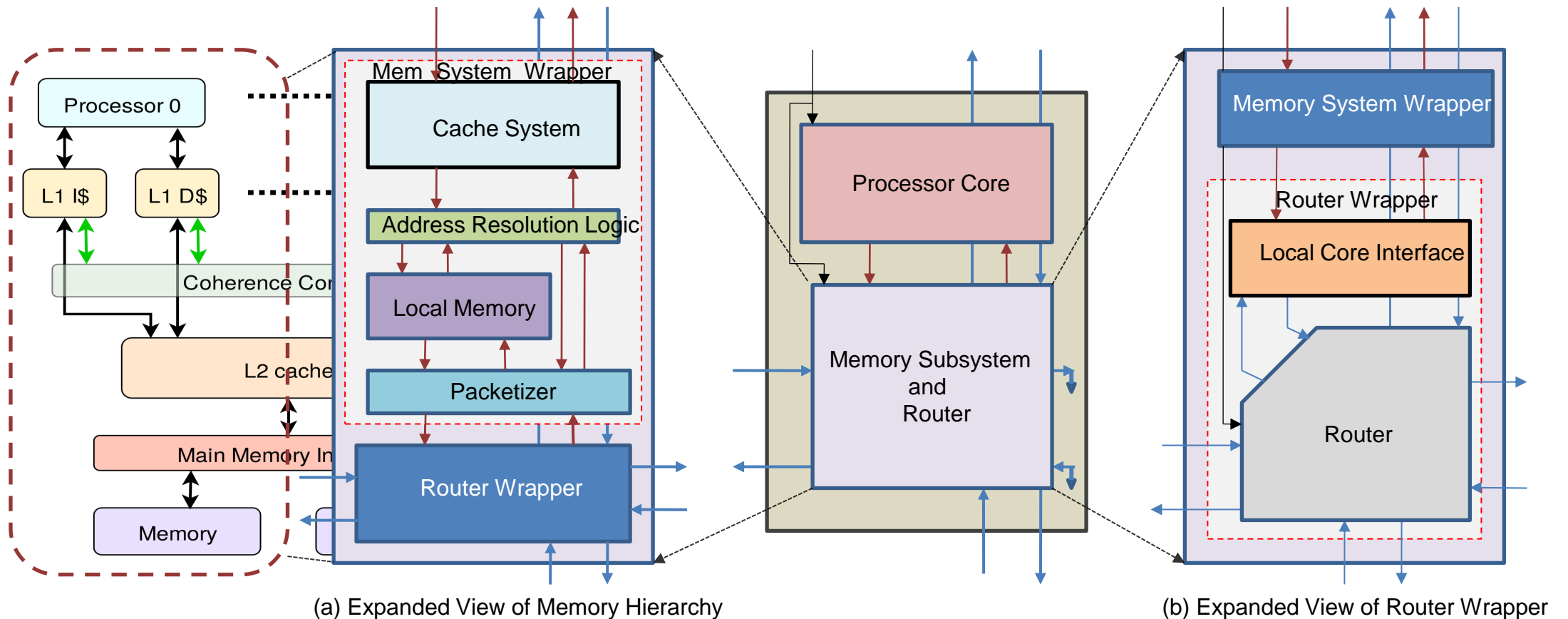
Presentation Outline

- Introduction
- Processors
- Memory subsystem
 - Caches structure
 - Reconfigurable hierarchies
- Network-on-Chip
- BRISC-V ecosystem
- Conclusion



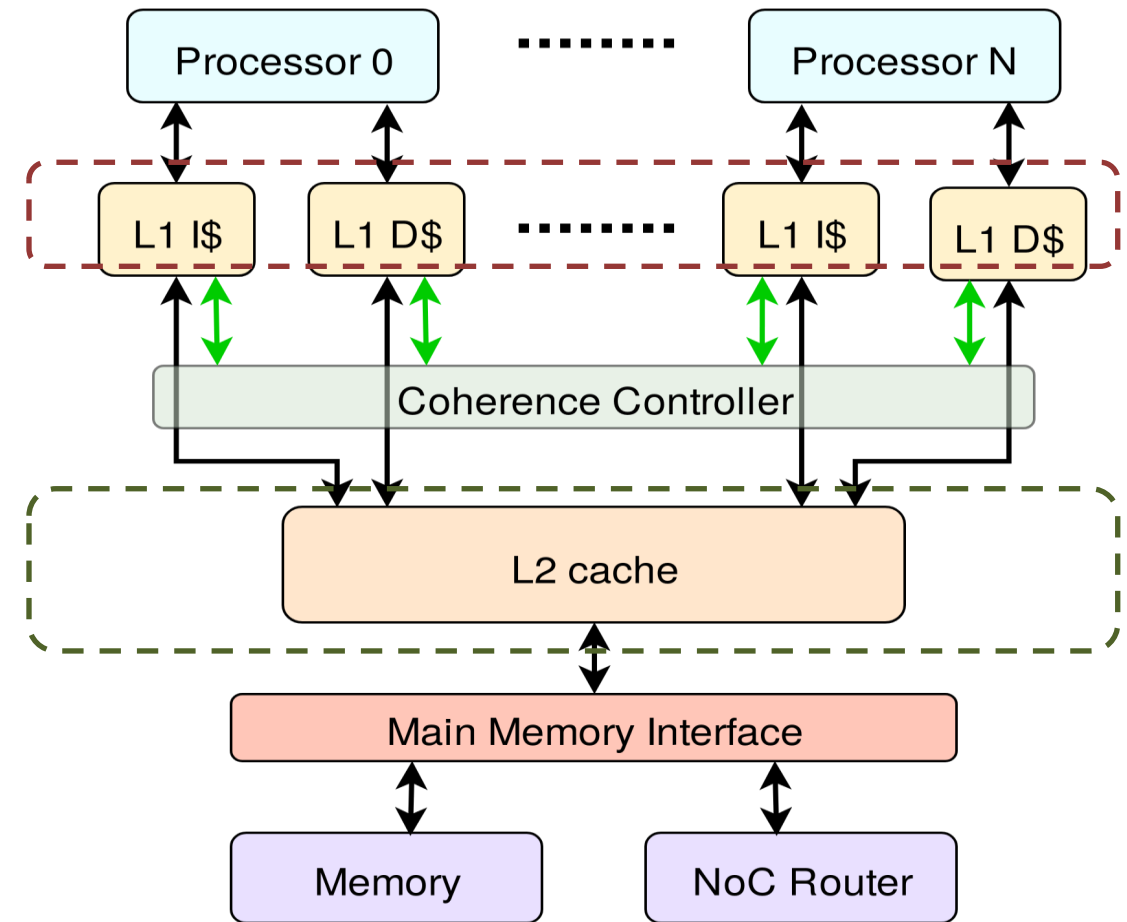
Memory System Organization

- Single processor full node



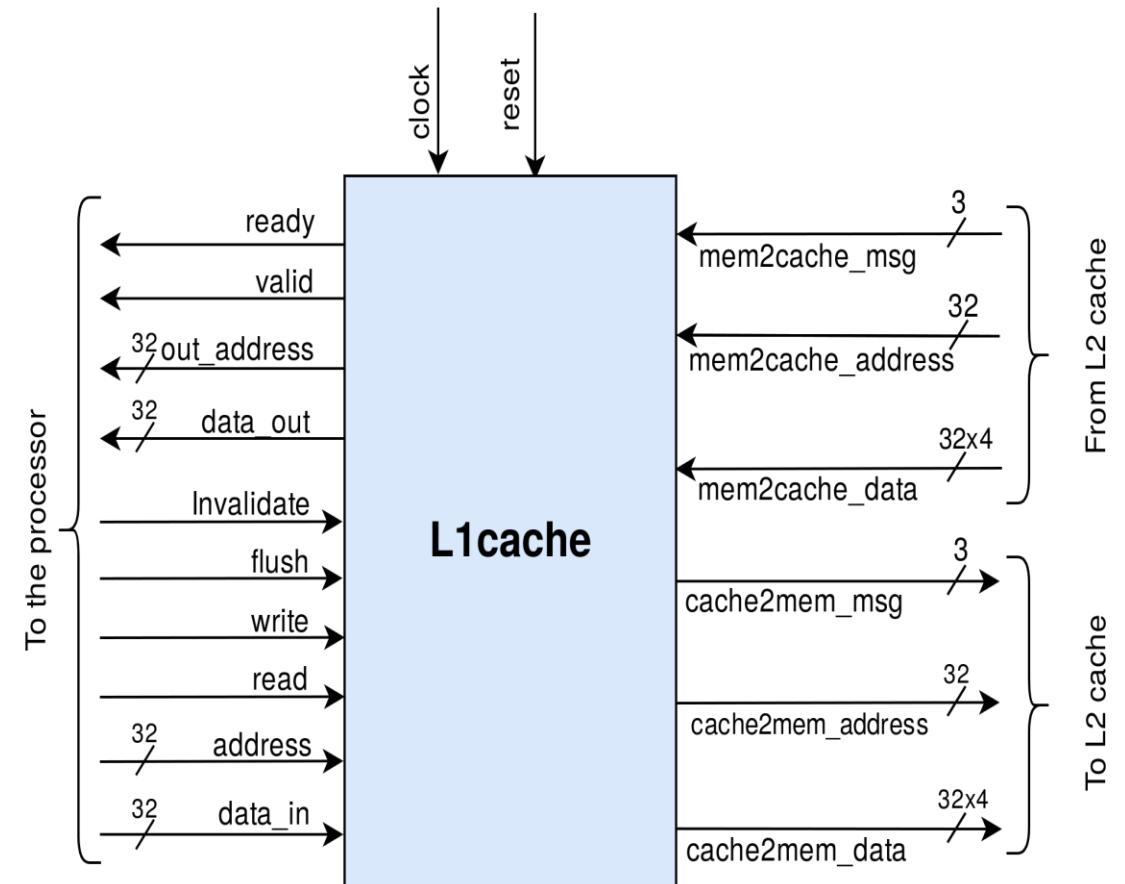
Memory Subsystem

- Multi-level cache hierarchy
 - Two cache modules
 - Primary caches : L1cache
 - Secondary caches : Lxcache



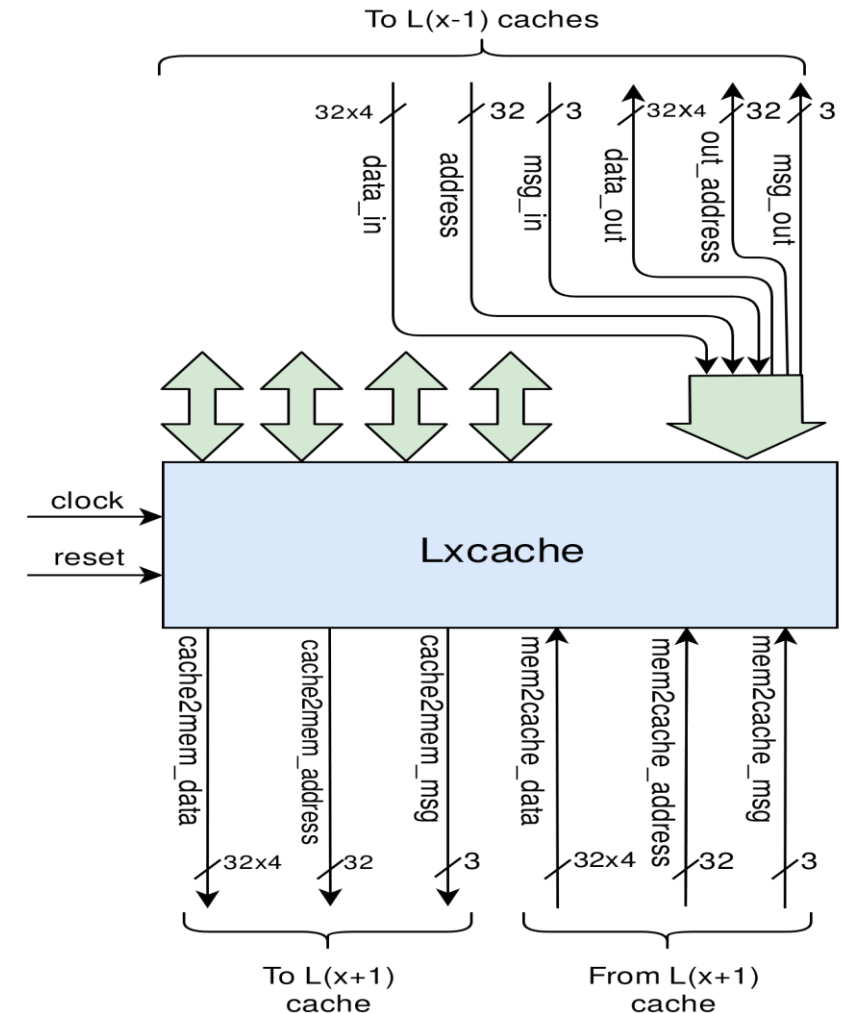
Memory Subsystem

- Multi-level cache hierarchy
 - Two cache modules
 - Primary caches : L1cache
 - Two cycle access time
 - Pipelined access
 - Blocking
 - Use dual port RAMs to allow coherence operations in parallel with cache accesses
 - Secondary caches : Lxcache



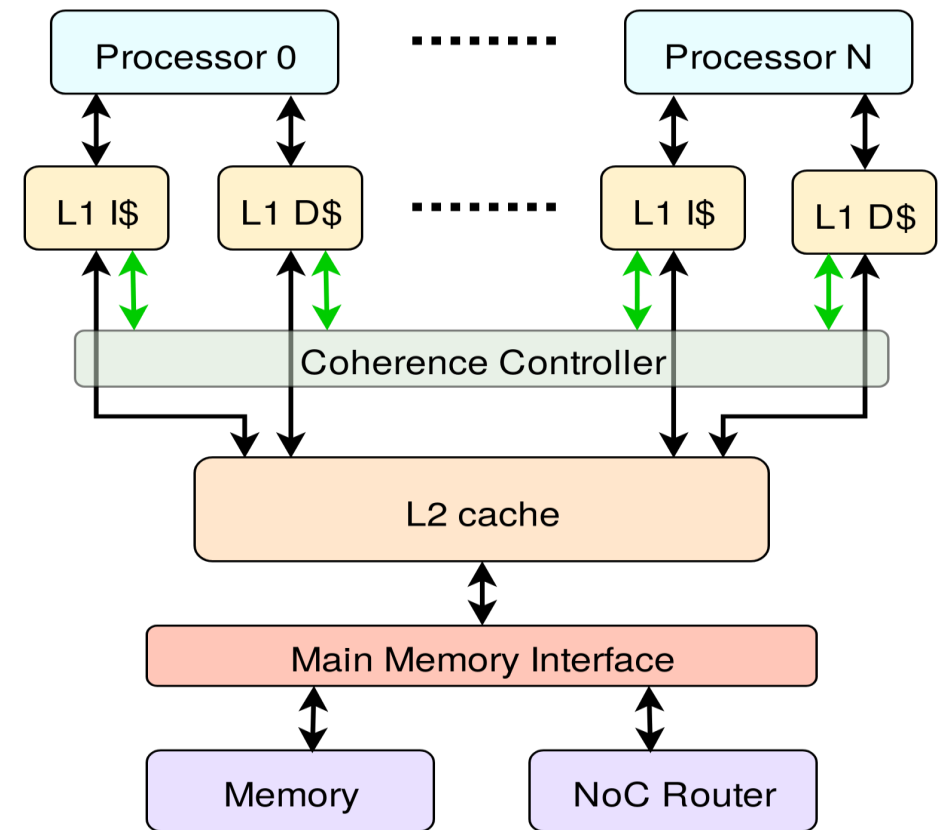
Memory Subsystem

- Multi-level cache hierarchy
 - Two cache modules
 - Primary caches : L1cache
 - Secondary caches : Lxcache
 - Could be used at L2 or L3
 - Parameterized number of ports
 - Round robin arbitration



Memory Subsystem

- Multi-level cache hierarchy
 - Two cache modules
 - Primary caches : L1cache
 - Secondary caches : Lxcache
- Key features
 - Inclusive caches
 - Write-back with write-allocate
 - True LRU replacement
 - MESI cache coherence



Memory Subsystem

■ Configurable parameters

- Number of cache sets
- Cache line width
 - Impacts the routing overhead as the buses between cache levels are as wide as a cache line.

L1 cache :		Cache size = 4kB		Associativity = 4	
Line width (Bytes)	Cache sets	Logic elements	Register usage	BRAM usage	Worst case Fmax
4	256	1,663	612	63,488	82.48 MHz
8	128	2,227	835	48,128	81.02 MHz
16	64	3,198	1282	40,448	82.67 MHz
32	32	5,039	2177	36,608	81.07 MHz
64	16	8,888	3968	34,688	76.28 MHz

Lxcache :		Cache size = 16kB		Associativity = 4	
Line width (Bytes)	Cache sets	Logic elements	Register usage	BRAM usage	Worst case Fmax
4	1024	1,627	696	249,856	70.31 MHz
8	512	2,259	1014	190,464	71.93 MHz
16	256	3,180	1652	160,768	73.82 MHz
32	128	5,241	2930	145,920	71.02 MHz
64	64	9,468	5488	138,496	71.1 MHz

Memory Subsystem

■ Configurable parameters

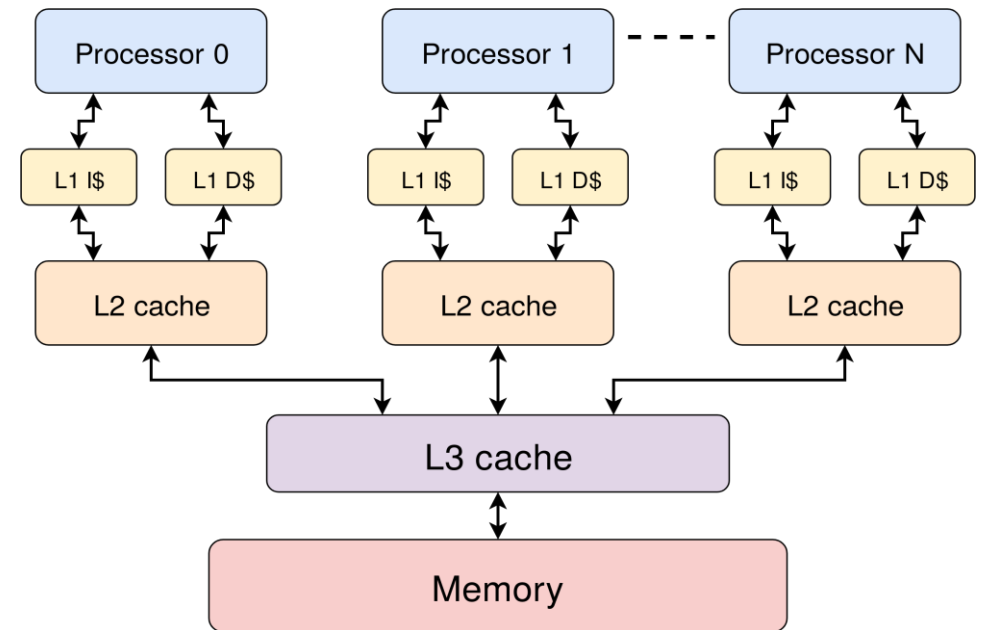
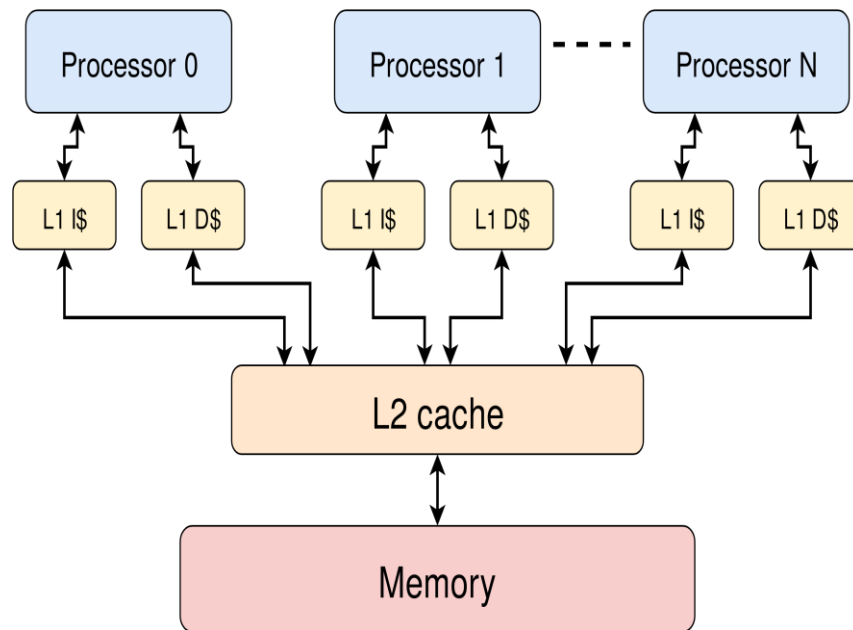
- Number of cache sets
- Cache line width
- Associativity

L1 cache :		Cache size = 4kB		Line width = 16 Bytes	
Associativity	Cache sets	Logic elements	Register usage	BRAM usage	worst case Fmax
1	256	2,387	1272	39,424	104.98 MHz
2	128	2,736	1277	39,936	97.12 MHz
4	64	3,198	1282	40,448	82.67 MHz
8	32	4,534	1287	40,960	66.76 MHz
16	16	6,546	1292	41,472	49.22 MHz

Lxcache :		Cache size = 16kB		Line width = 16 Bytes	
Associativity	Cache sets	Logic elements	Register usage	BRAM usage	Worst case Fmax
1	1024	1,918	1161	156,672	99.17 MHz
2	512	2,507	1331	158,720	79.66 MHz
4	256	3,180	1652	160,768	73.82 MHz
8	128	4,583	2301	162,816	58.51 MHz
16	64	7,370	3614	164,864	50.13 MHz

Memory Subsystem

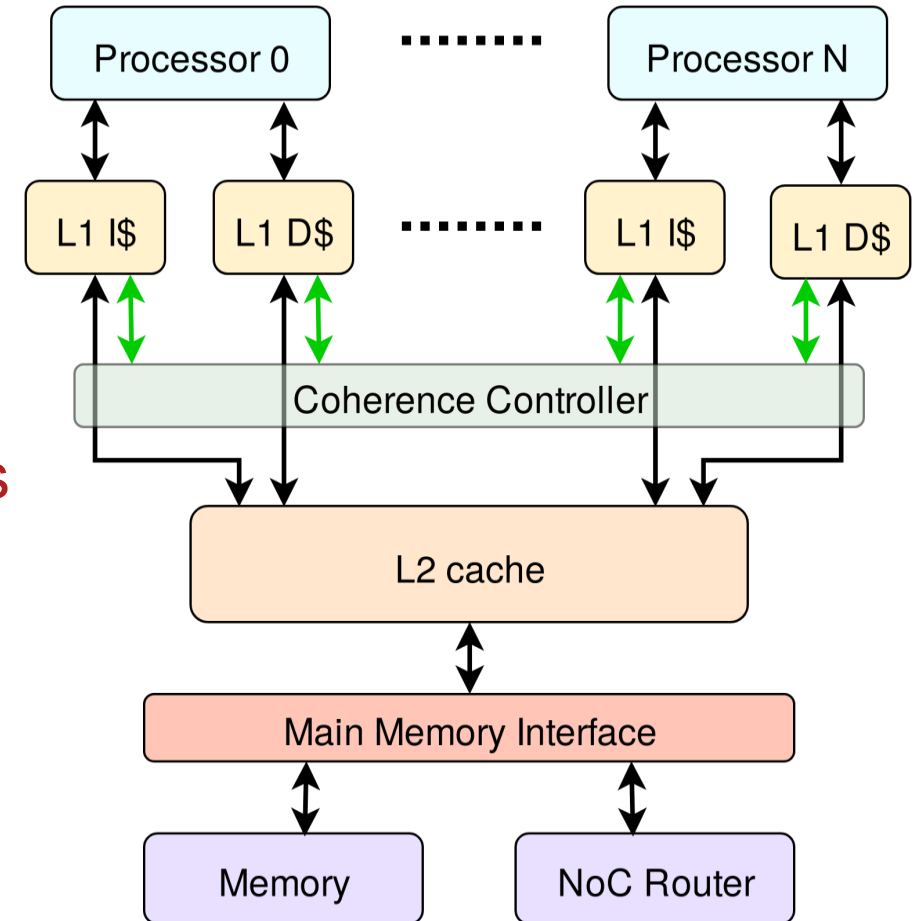
- Configurable parameters
 - Number of levels in the hierarchy



Memory Subsystem

■ Modular design

- Clear interfaces
- Consistent across levels
- Replace any module
 - Including processors
- Substitute modules for different functions
 - Ex: replacement controller, arbitrator
- Rest of the functionality clearly separated into procedural blocks.
 - Ex: Coherence operations and normal cache operations.



Memory Subsystem

■ Memory System Wrapper

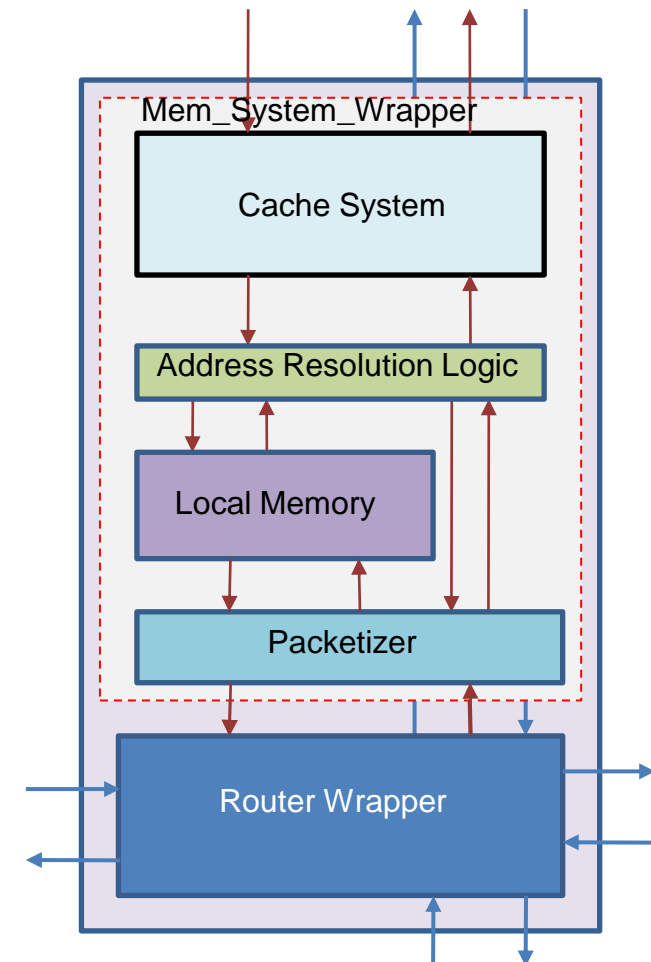
● Cache system

■ Address resolution logic

- Determines if a request can be served at the local memory, or if the request needs to be sent over the network

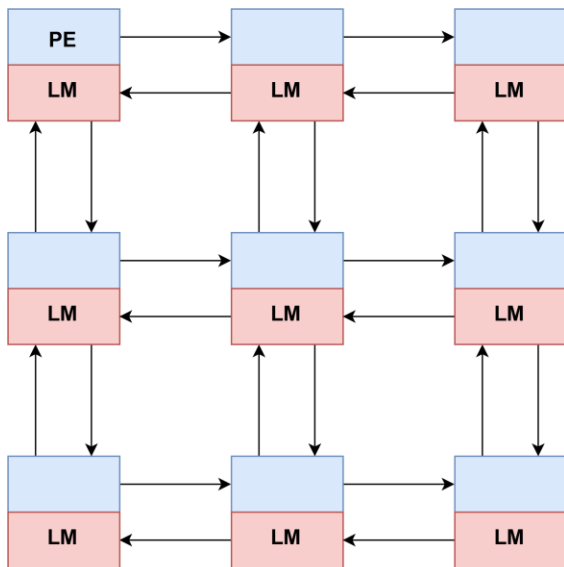
■ Local block memory

- The LOCAL_ADDR_BITS parameter is used to set the size of the local memory

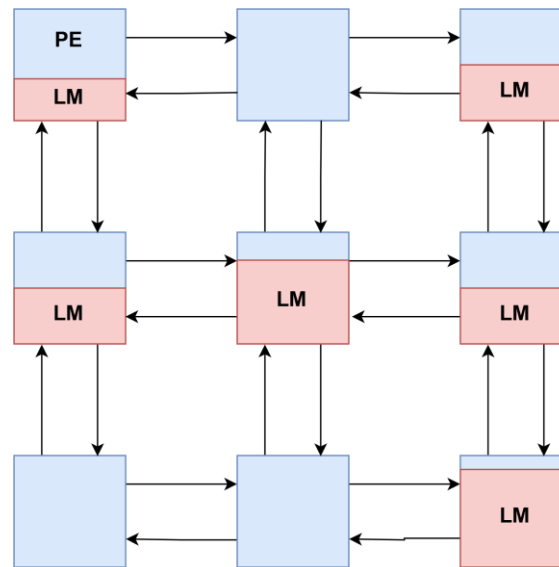


Main memory

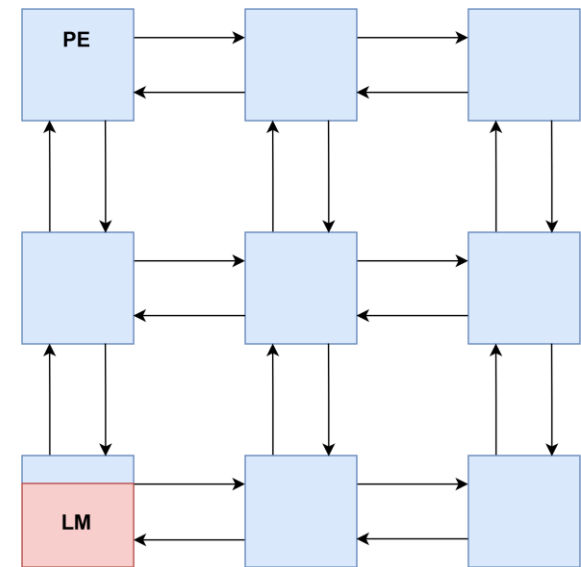
- Constructed to allow different physical memory configurations
 - Centralized Shared Memory (CSM)
 - Distributed Shared Memory (DSM)
 - Uniform and non-uniform distributed memory



Uniform distributed memory



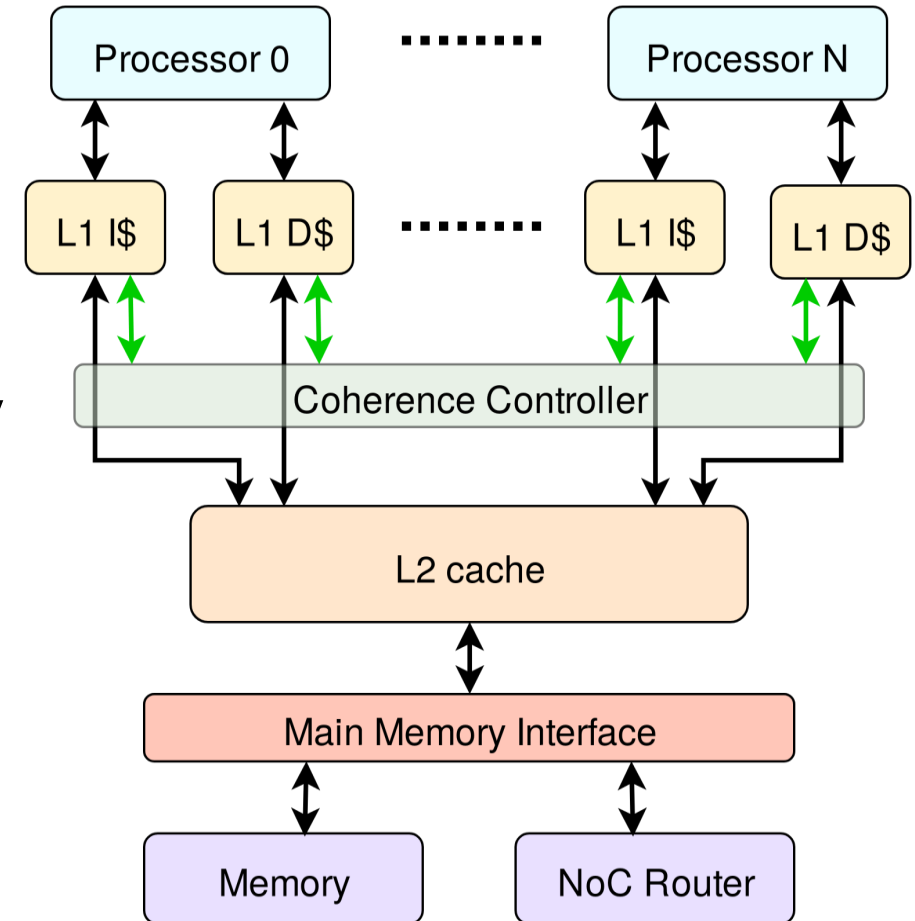
Non-uniform distributed memory



Non-distributed memory

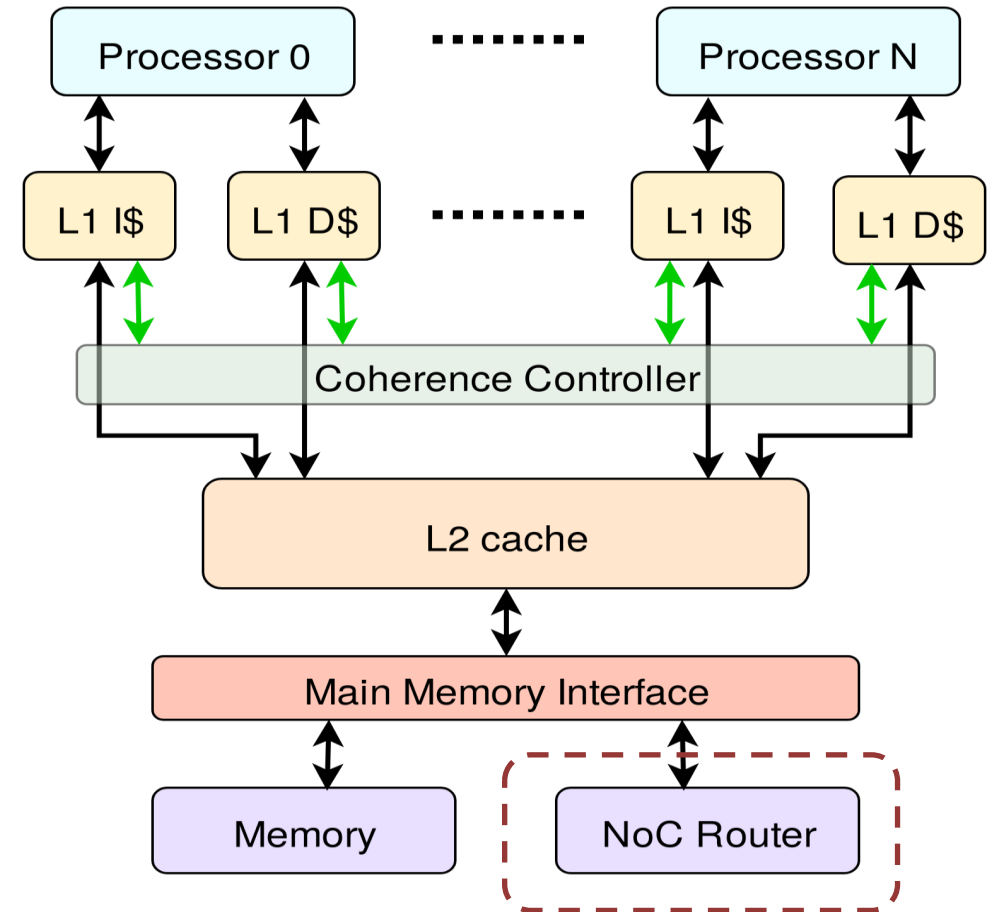
Main memory

- Constructed to allow different configurations
 - Centralized Shared Memory (CSM)
 - Distributed Shared Memory (DSM)
 - Uniform and non-uniform distributed memory
- Main memory interface
 - Handles address resolution in a DSM system
 - Bridges bandwidth gap between main memory and cache subsystem



Presentation Outline

- Introduction
- Processors
- Memory subsystem
- Network-on-Chip
 - Router Micro-Architecture
 - Routing
- BRISC-V ecosystem
- Conclusion

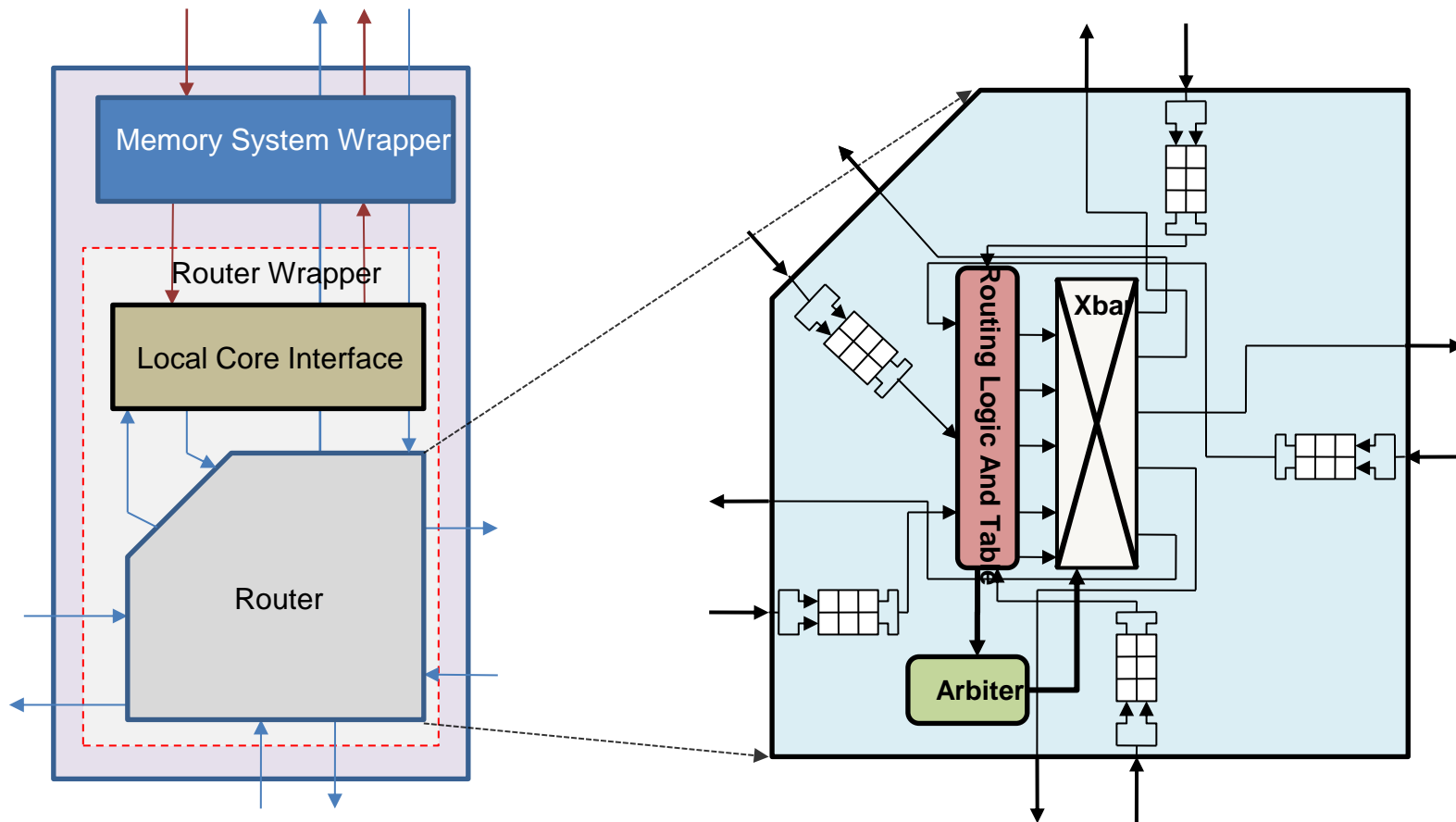


Router Micro-Architecture Design

- Conventional virtual-channel router
 - Route Computation (RC)
 - Virtual-channel Allocation (VA)
 - Switch Allocation (SA)
 - Switch Traversal (ST)
- Each phase corresponds to a pipeline stage for the pipeline router
- Supports both fixed logic and table based oblivious routing
- Parameterized
 - No. of input/output ports
 - Virtual channels per port
 - Virtual channel depth

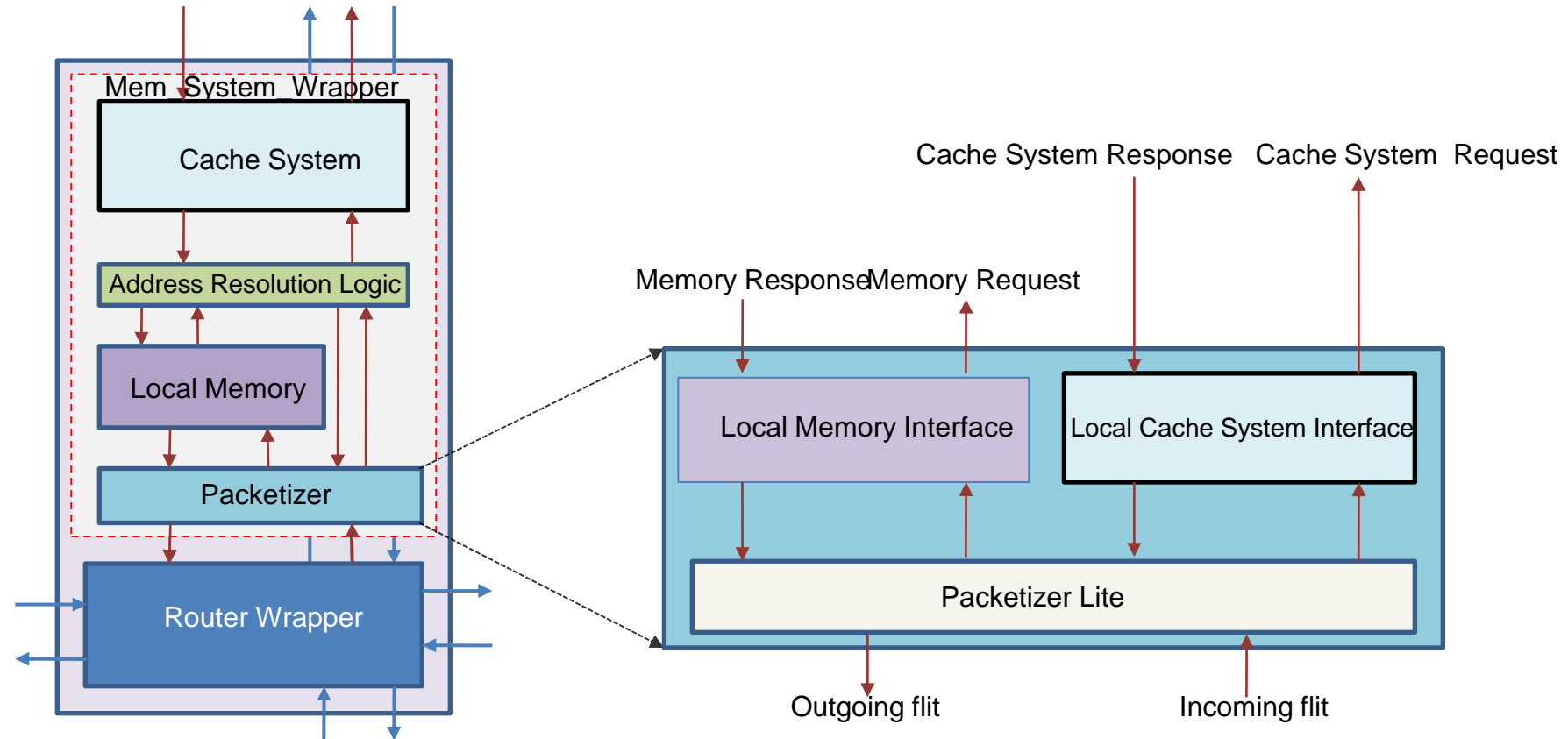
Router Micro-Architecture Design

- IN_PORTS**, **OUT_PORTS**, **VC_PER_PORT** and **VC_DEPTH** parameters allow user controlled router size.



Router Interface Design

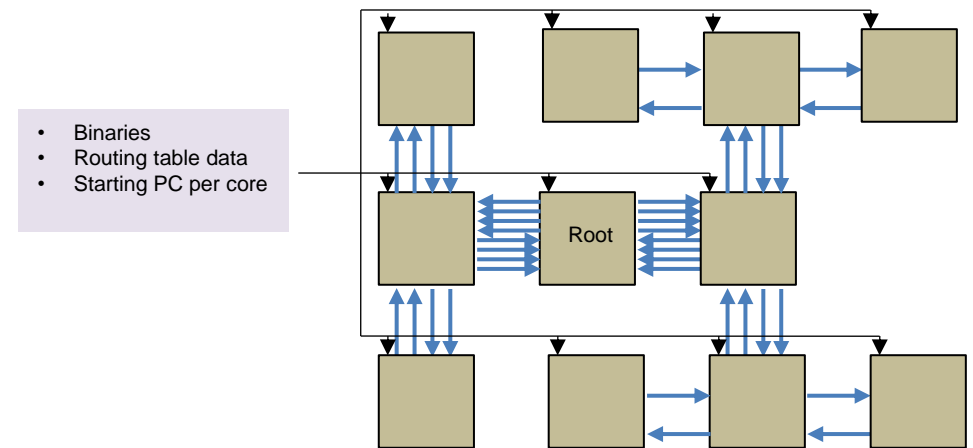
- Packetizer Lite:** is responsible for converting data traffic into packets that can be routed inside the on-chip network.



Network-on-Chip Architecture

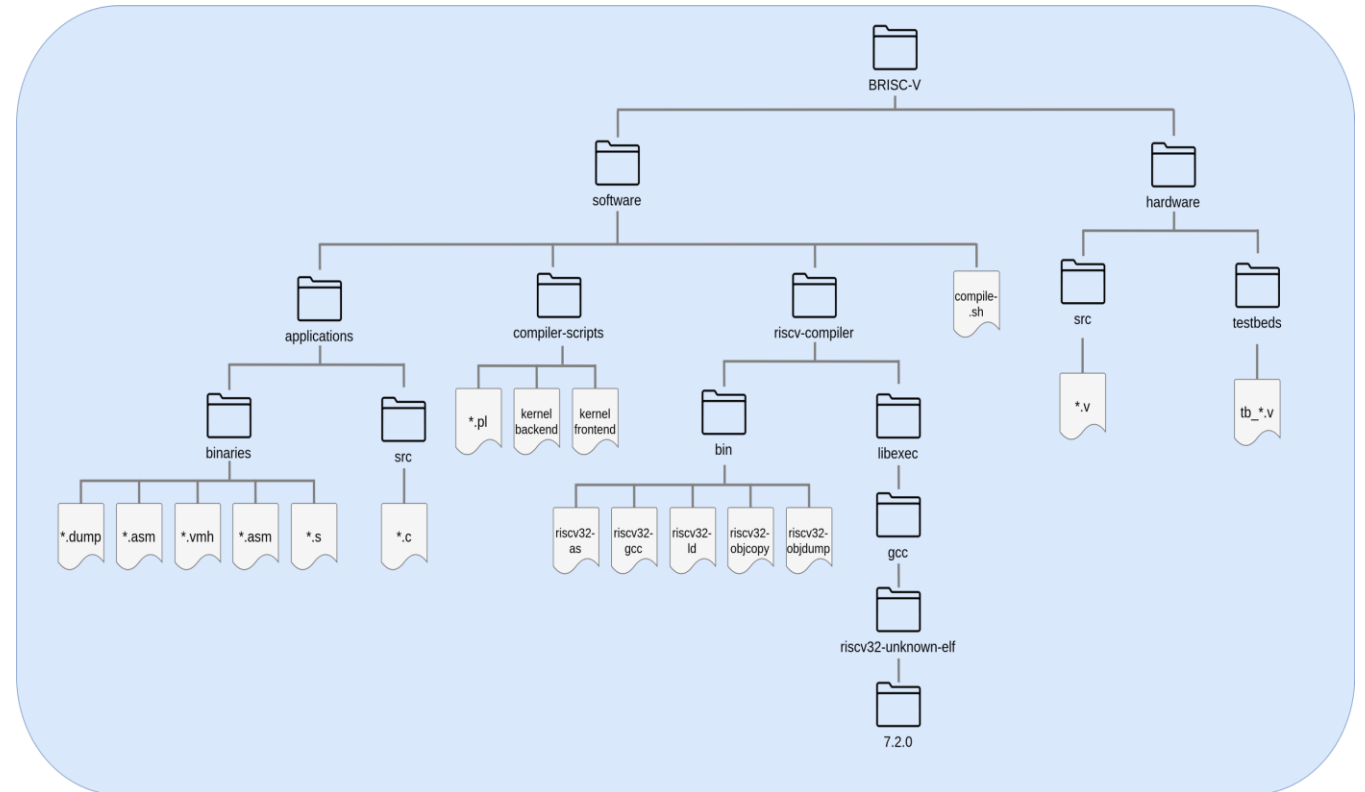
- Flow control
 - Bufferless or buffered routers
- Routing algorithm
 - Oblivious routing using either fixed logic or routing tables
 - Fixed logic for Dimension Order Routing by default
 - Statically configured routing tables
- Network topology
 - Configure number of ports on the routers and routing tables

Unbalanced Fat-Tree Topology



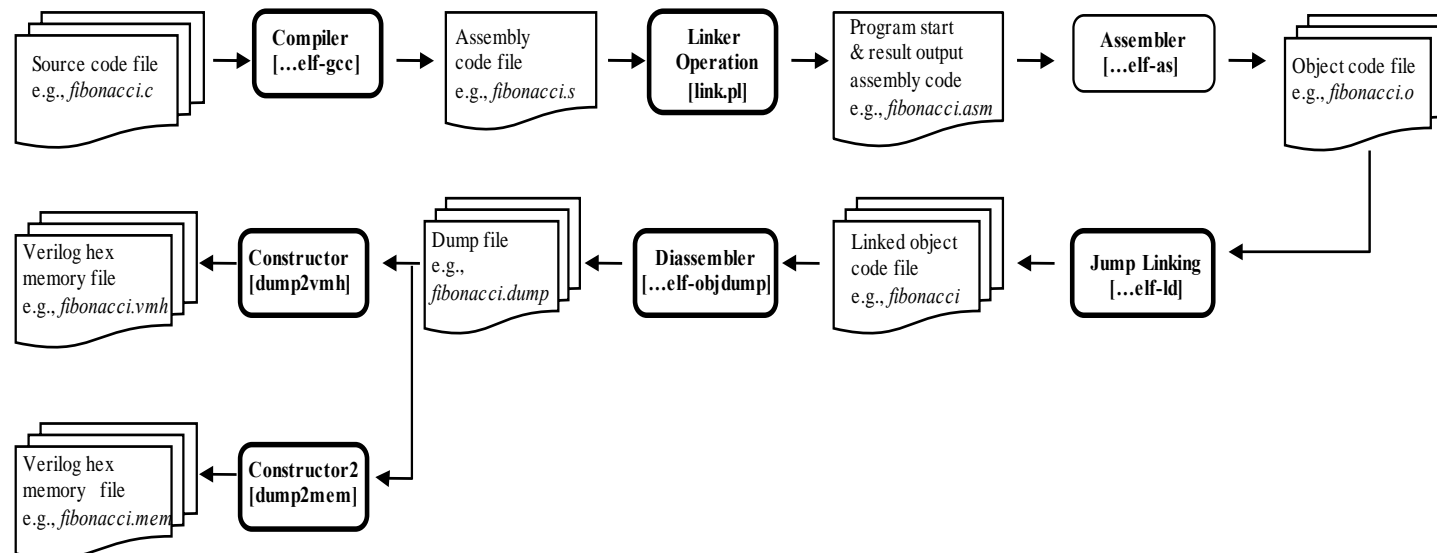
Presentation Outline

- Introduction
- Processors
- Memory subsystem
- Network-on-Chip
- **BRISC-V ecosystem**
- Conclusion



BRISC-V Ecosystem

- Program compilation flow
 - To assist in developing software for the BRISC-V processor, it is accompanied with a GCC RISC-V cross-compiler
 - The software flow for compiling a C program into the compatible BRISC-V instruction code that can be executed on the processor



BRISC-V Ecosystem

- We are maintaining a website for updates, news and new module releases



Adaptive and Secure Computing
Systems Lab • Boston University

Home | Downloads | Members | Contact Us

Release Notes | BRISC-V Explorer | BRISC-V Emulator

BRISC-V Toolbox

The BRISC-V toolbox is the Boston University RISC-V architecture design exploration suite. BRISC-V is comprised of a number of different processor architectures, an emulator, and a visual verilog file generation tool, for education and research projects.

The Boston University RISC-V Processor Set (BRISC-V) is a parameterized set of modules for design space exploration using RISC-V ISA based architectures. We call the full set of processors and tools to support them the tool box. Included with the BRISC-V Tool box are (i) numerous RISC-V cores with different levels of complexity (e.g., single-cycle core, multiple-cycle, and reconfigurable pipelined), (ii) a programmable memory system with reconfigurable multi-level cache subsystems, (iii) BRISC-V explorer which is GUI software support for selecting parameterized verilog and (iv) the BRISC-V emulator for software RISC-V instruction emulation. The toolbox provides an easy to use, open-source, parameterized, fully synthesizable, platform for students and researchers experimenting with the RISC-V ISA features to quickly bring up a complete and fully working architecture and start applying their own modifications.

The BRISC-V hardware is comprised of a number of different single-core and multi-core systems. Each hardware component of BRISC-V (the cores, and cache subsystem) is written in parameterized Verilog HDL modules, enabling architectural changes with parameter settings. The core types currently supported by BRISC-V include RV32I implementations of a small single cycle core, five and seven stage pipeline cores and a larger RV32I out-of-order core. The cache subsystem interface supports numerous memory hierarchy configurations. An arbitrary cache size, associativity and number of levels can be selected with module parameters.

Additionally the BRISC-V Explorer provides a user friendly way to choose parameters and visualize a BRISC-V system. The application runs in a browser allowing users to easily run it on Windows, Linux or Mac. In the BRISC-V Explorer, users can 1) select their desired core type, 2) enter parameters such as memory size for that core, and 3) configure cache parameters including block size and associativity. Once a user is sure of their processor and generates their design a verilog the explorer tool will generate a verilog design based around the users selections. From there its up to the users imagination on what to do with it.

Finally the BRISC-V Emulator visually shows the state of the processor at every instruction and allows for exploration of a compiled code behaving as expected. Being an in browser tool OS dependencies are avoided allowing for an easy, fast, and intuitive exploration. The register file, instruction break down, memory state and program list are all displayed as the program operates.

News

October 29th, 2018
Donato and Sahan will be introducing the platform at the RISC-V Summit.
[More...](#)

October 26th, 2018
Thanks for EC513 Spring 2018 students for their feedback on the development version.
[More...](#)

October 26th, 2018
BRISC-V Website is now a thing!
[More...](#)





ADAPTIVE & SECURE COMPUTING SYSTEMS
LABORATORY

Home | Downloads | BRISC-V © 2018

Downloads

Here is a brief overview of the architecture and system configuration [\[Manual\]](#).

For feedback, questions or contributions contact us at briscv.feedback@gmail.com

If you use BRISC-V in your research, we would appreciate the following citation in any publications to which it has contributed:

Sahan Bandara, Alan Ehret, Donato Kava and Michel A. Kinsy: "BRISC-V: Open Source Architectural Design Space Exploration Toolbox" Tech-Report 0V1, October 2018 [\[bib\]](#)

By clicking the download links, you acknowledge that you have read and agree to the Software Licensing Agreement below.

BRISC-V source code and test bench:



BRISC-V Application (e.g., gcd.c) Compiler:



Software Licensing Agreement


Copyright © 2018 BRISC-V (ASCS/ECE/BU). Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

BRISC-V Ecosystem

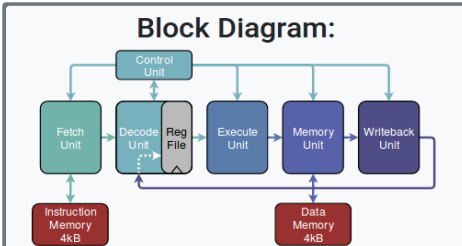
- GUI for quick launches
- Physical design inspection
- Assembly level debugging

BRISC-V Home
BRISC-V Explorer
Manual & Examples


BRISC-V Explorer
Adaptive and Secure Computing Systems Lab • Boston University

Directions
Core Selection
Processor Parameters
Cache Parameters
Generate


Directions:
 Select your BRISC-V Core type, enter the desired parameters and click the "Generate" button.

Block Diagram:


Output:
 Click the Generate button in Core Selection to output a configuration.

BRISC-V © 2018

RISC-V Emulator
ascslab.org/research/briscv/emulator/emulator.html
BRISC-V Home
BRISC-V Emulator
Manual & Examples


BRISC-V Emulator
Adaptive and Secure Computing Systems Lab • Boston University

Download
Play
Pause
Reset

Register	Value	Register	Value
zero (0)	0	ra (1)	37
sp (2)	1488	gp (3)	0
tp (4)	0	t0 (5)	0
t1 (6)	0	t2 (7)	0
s0/tp(8)	1536	s1 (9)	0
a0 (10)	0	a1 (11)	0
a2 (12)	0	a3 (13)	0
a4 (14)	0	a5 (15)	1
a6 (16)	0	a7 (17)	0
s2 (18)	0	s3 (19)	0
s4 (20)	0	s5 (21)	0
s6 (22)	0	s7 (23)	0
s8 (24)	0	s9 (25)	0
s10 (26)	0	s11 (27)	0
t3 (28)	0	t4 (29)	0
t5 (30)	0	t6 (31)	0

Instruction Breakdown:

31	25	24	20	19	15	14	12	11	7	6	0
imm	rs2	rs1	010	imm	opcode						
111111110	00000	01000	010	1100	0000011						

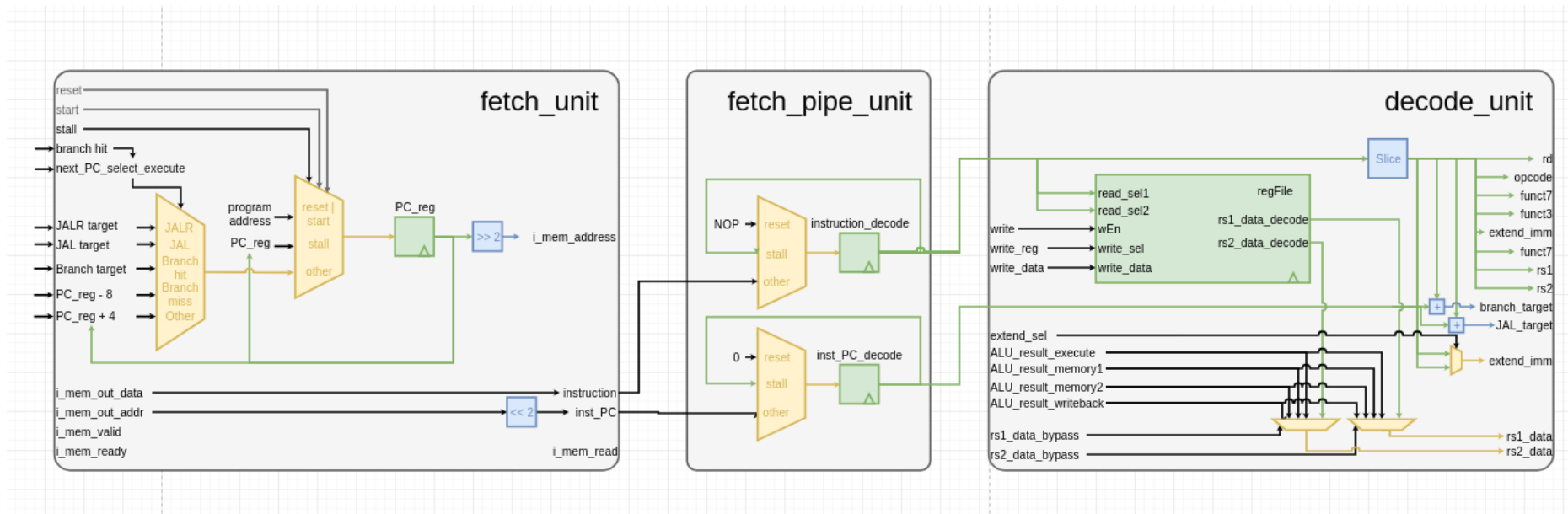
```

94      addi zero,zero,0
95      addi zero,zero,0
    .file "fibonacci_print.c"
    .option nopic
    .text
    .align 2
    .globl main
    .type main, @function
main:
    addi sp,sp,-48
    sw ra,44(sp)
    sw s0,40(sp)
    addi s0,sp,48
100     li a5,-1879948192
101     sw a5,-32(s0)
102     li a5,10
103     sw a5,-36(s0)
104     li a5,1
105     sw a5,-24(s0)
106     sw zero,-20(s0)
107     sw zero,-28(s0)
108     j .L2
.L2:
109     lw a4,-24(s0)
110     lw a5,-20(s0)
111     add a5,a4,a5
112     sw a5,-40(s0)
113     lw a5,-24(s0)
114     sw a5,-20(s0)
115     lw a5,-40(s0)
116     sw a5,-24(s0)
117     lw a0,-40(s0)
118     call tx_uint
119     lw a5,-28(s0)
120     addi a5,a5,1
121     sw a5,-28(s0)
                
```

STACK SEGMENT
 FREE SPACE
 0x000005FF: 0x00000000
 0x000005FE: 0x00000000
 0x000005FD: 0x00000000
 0x000005FC: 0x25
 0x000005FB: 0x00000000
 0x000005FA: 0x00000000
 0x000005F9: 0x00000000
 0x000005F8: 0x0
 0x000005F7: 0x00000000
 0x000005F6: 0x00000000
 0x000005F5: 0x00000000
 0x000005F4: 0x00000000
 0x000005F3: 0x00000000
 0x000005F2: 0x00000000
 0x000005F1: 0x00000000
 0x000005F0: 0x00000000
 0x000005EF: 0x00000000
 0x000005EE: 0x00000000
 0x000005ED: 0x00000000
 0x000005EC: 0x00000000
 0x000005EB: 0x00000000
 0x000005EA: 0x00000000
 0x000005E9: 0x00000000
 0x000005E8: 0x1
 0x000005E7: 0x00000000
 0x000005E6: 0x00000000
 0x000005E5: 0x00000000
 0x000005E4: 0x00000000
 0x000005E3: 0x00000000
 0x000005E2: 0x00000000
 0x000005E1: 0x00000000
 0x000005E0: 0x-70000000
 0x000005DF: 0x00000000
 0x000005DE: 0x00000000
 0x000005DD: 0x00000000
 0x000005DC: 0xA

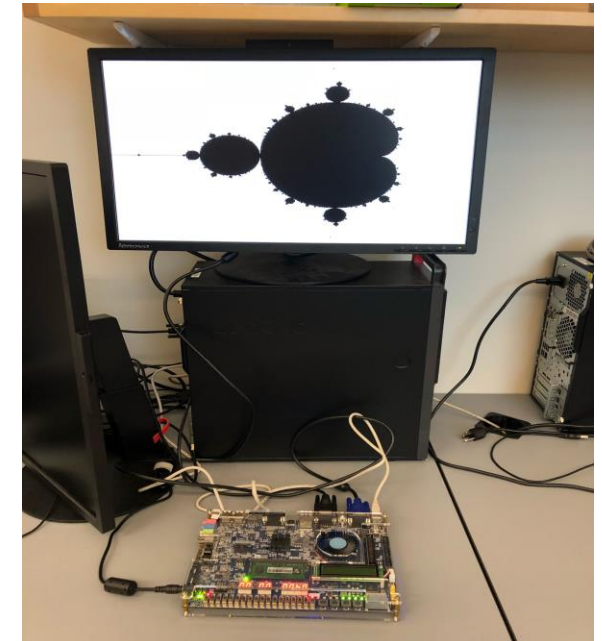
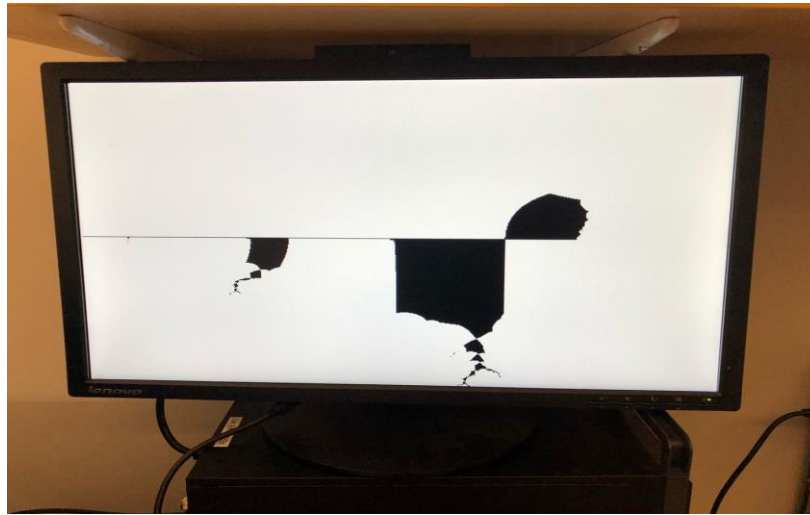
BRISC-V Ecosystem

- All provided modules and runtime generated code are in synthesizable, human readable and editable Verilog
- Designs come with module schematics for facilitate signal understanding



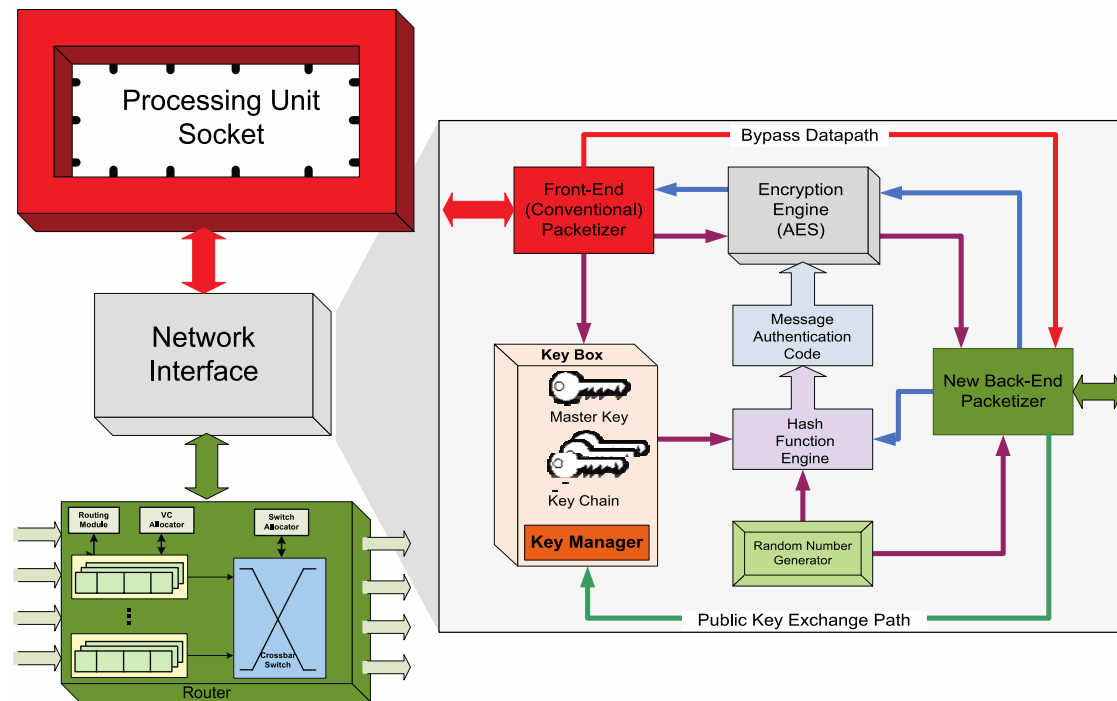
BRISC-V Ecosystem

- Faster design space exploration and closing the system mischaracterization gap
 - For example, how to measure and evaluate physical side-channel effects in software? Or emulation?



BRISC-V Support for Secure RISC-V Designs

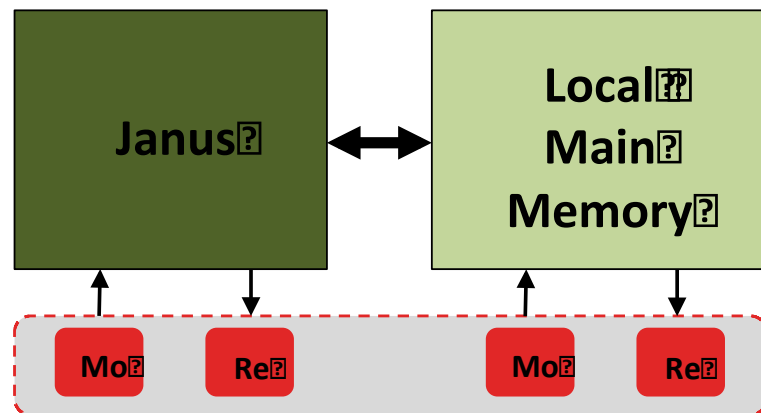
- We currently have a number of ongoing secure RISC-V designs
 - Hermes: A Secure SoC Design



M. Kinsy, S. Khadka, M. Isakov and A. Farrukh: "Hermes: Secure Heterogeneous Multicore Architecture Design". In the IEEE International Symposium on Hardware Oriented Security and Trust (HOST), May 2017

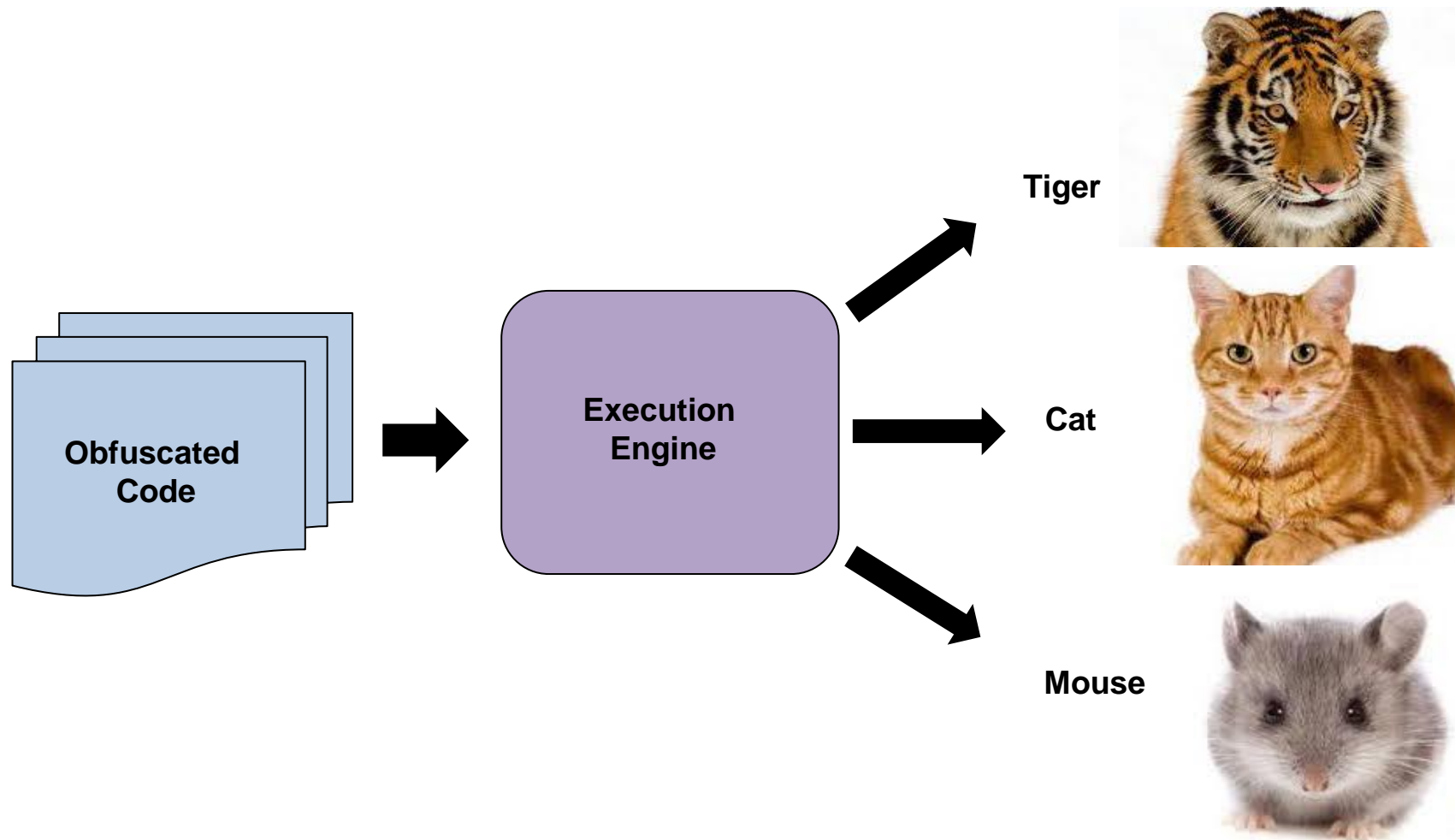
BRISC-V Support for Secure RISC-V Designs

- We currently have a number of ongoing secure RISC-V designs
 - **Janus: Cache Subsystem Obfuscation Architecture**
 - A hardware-support for cache-based side-channel attack prevention with low overheads
 - Uses runtime cache block on-off mechanism for obfuscation
 - Two new instructions to support runtime cache block ON/OFF operation
 - Intentionally injects cache misses to randomize power and timing behavior



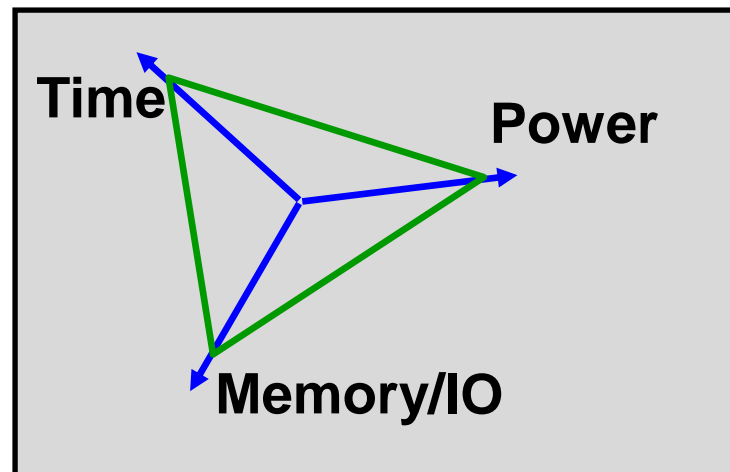
H. Hosseinzadeh, M. Isakov, M. Darabi, A. Patooghy, and M. Kinsy:
 "Janus: An uncertain cache architecture to cope with side channel
 attacks". In 2017 IEEE 60th International Midwest Symposium on
 Circuits and Systems (MWSCAS) Aug 2017. **The Myril B. Reed Best
 Paper Award.**

Sphinx-V: RISC-V Software-Hardware Obfuscation Architecture

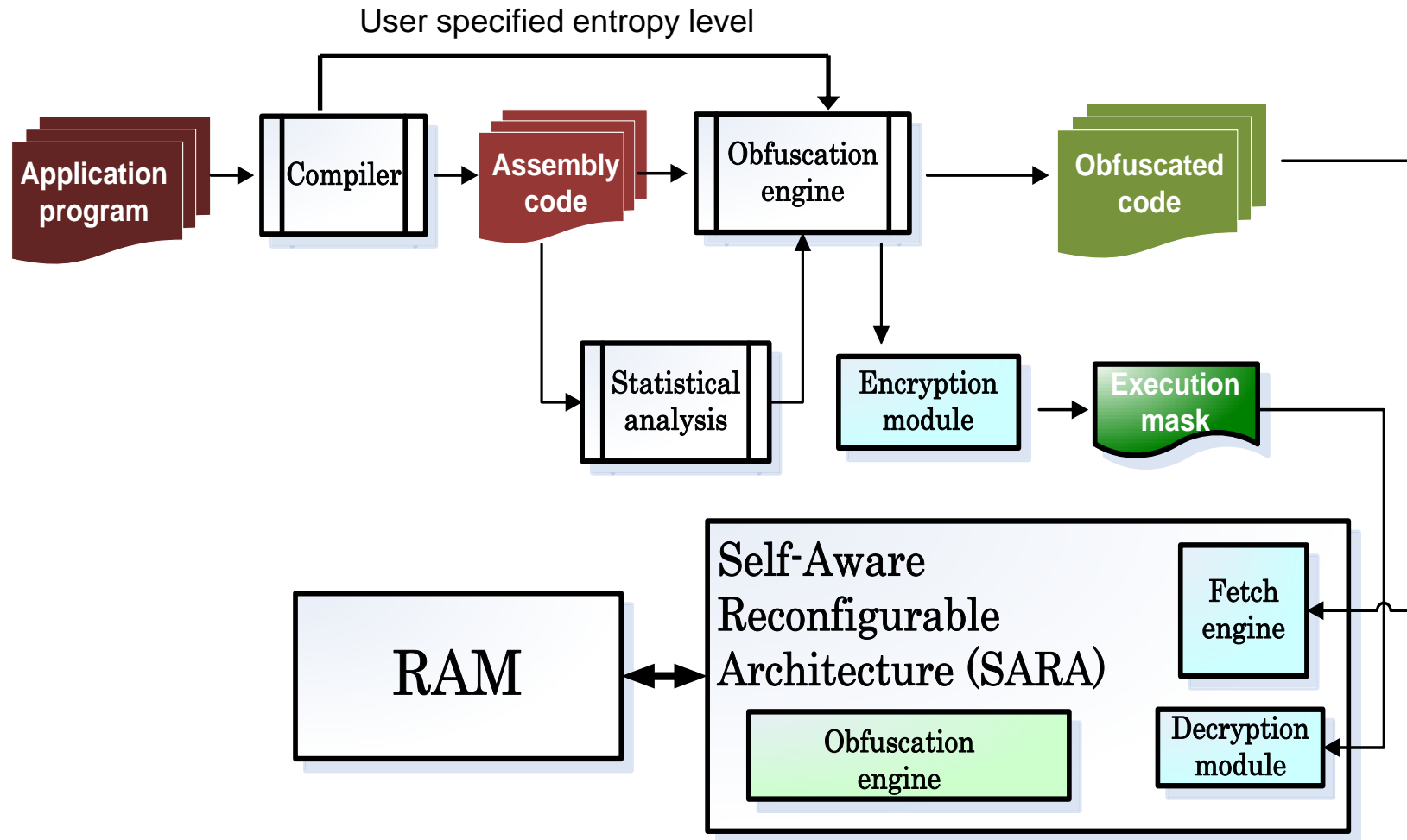


Sphinx-V: RISC-V Software-Hardware Obfuscation Architecture

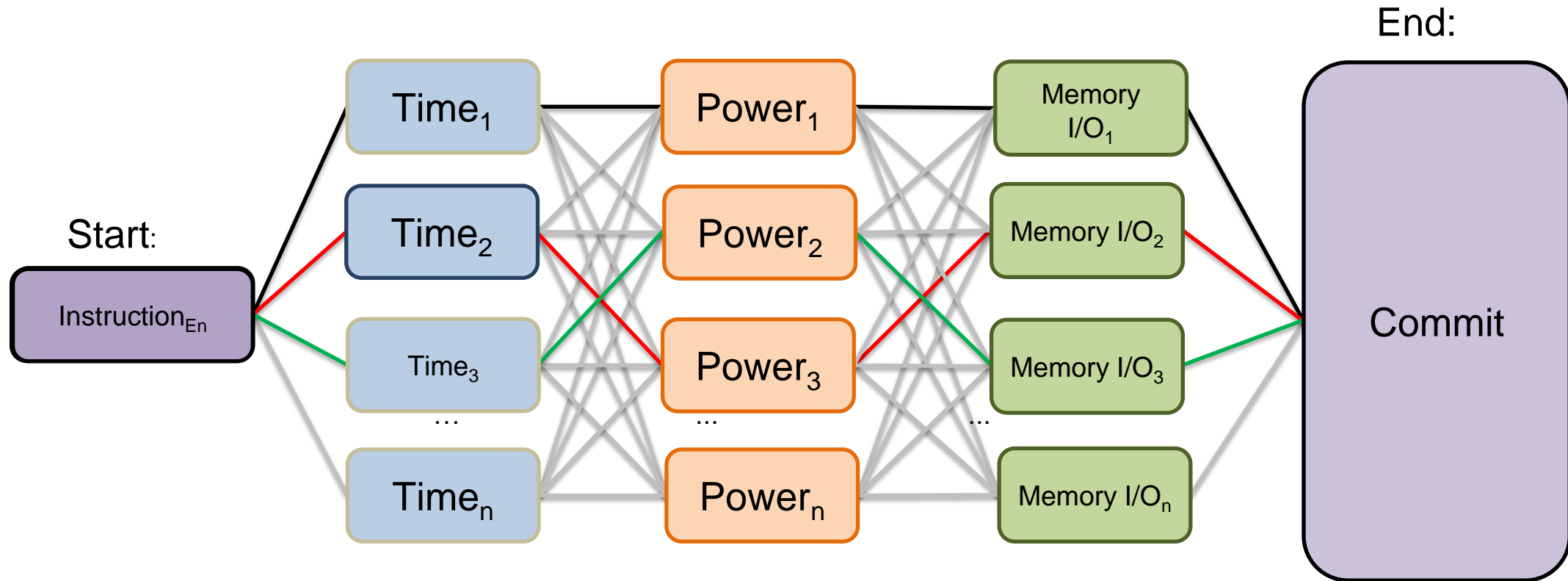
- Adaptive self-reconfigurable computing systems for moving target security
- An architecture capable of providing:
 - Performance/Timing-Awareness for timing obfuscation
 - Power-Awareness for power obfuscation
 - Self-Organized data storage for memory and I/O obfuscation



Sphinx-V: RISC-V Software-Hardware Obfuscation Architecture



Sphinx-V: RISC-V Software-Hardware Obfuscation Architecture



Conclusion and Future Work

- This is a growing RISC-V multicore design space exploration environment
- We welcome feedback, comments and collaborations, or even contributions
- Currently we are using the BRISC-V environment to design
 - A multicore, hardware multithreaded secure architecture
 - A 128-bit Extended Base Global Address Space (xBGAS) architecture for HPC and data center applications

Acknowledgements

- ASCS Lab Members, Boston University

More Information at
<http://ascslab.org/>