CSAIL

Heracles:
A Tool for Fast RTL-Based Design Space Exploration of Multicore Processors

Michel Kinsy, Michael Pellauer, and Srinivas Devadas

http://projects.csail.mit.edu/heracles

Massachusetts Institute of Technology

Outline

- Systems Overview
  - Motivation
  - Design Flow
- Processing Units
  - Injector & Single Hardware-Threaded MIPS Cores
  - Two-way & Hardware Context Migration MIPS Cores
- Memory System Organization
  - Main Memory Configuration
  - Caches & Cache Coherence Protocols
- Network-on-Chip (NoC)
  - Flow Control & Routing Algorithms
  - Network Topology Re-configuration
- Programming Models & Toolchain
  - Sequential & Parallel Programming Models
  - Graphical User Interface & Programs Compilation
- Conclusion
  - Summary and Future Work

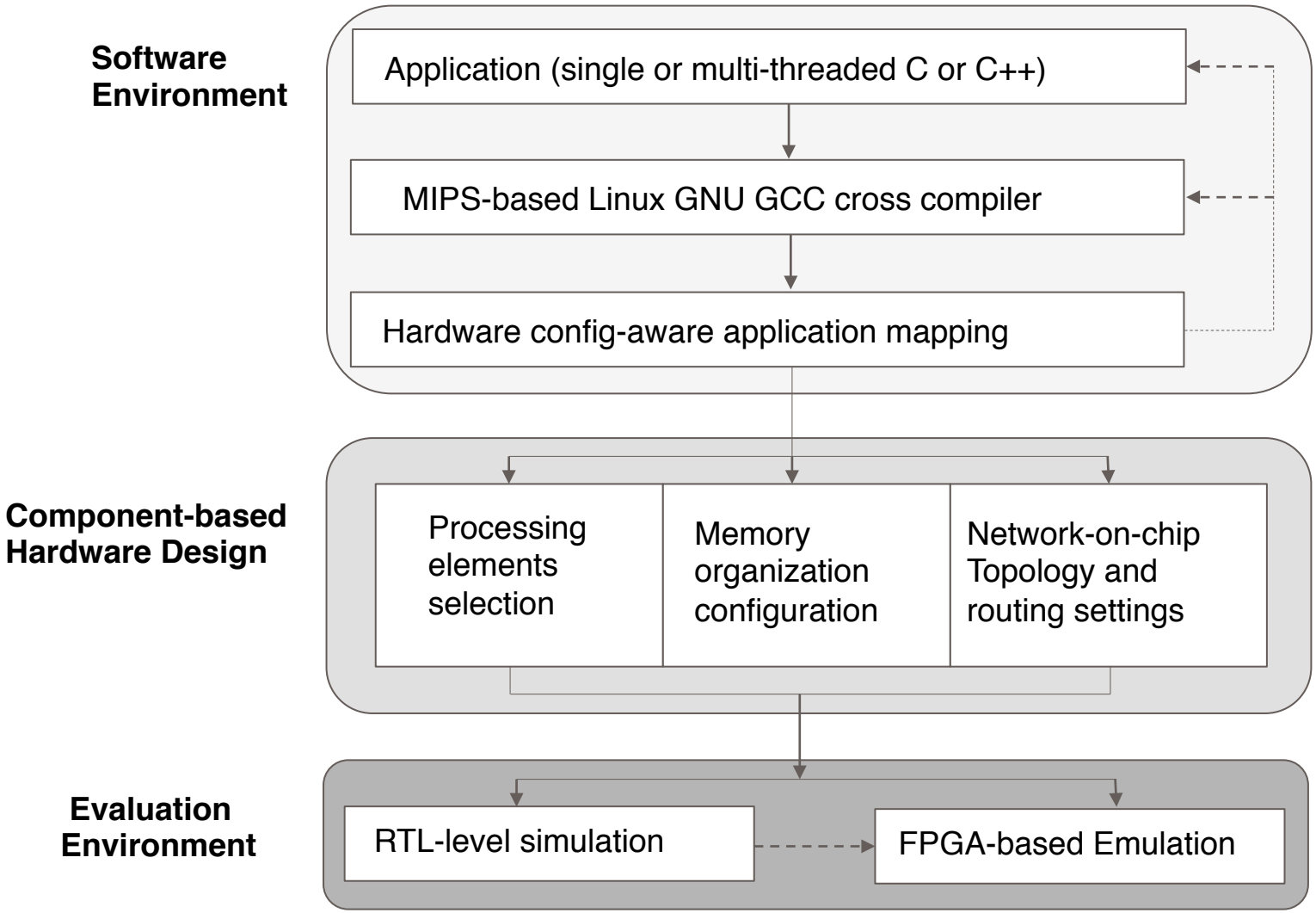
System Overview: Motivation

- 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.
- Software systems were too slow: DARSIM, HORNET, GRAPHITE, etc.
- Loose software models can lead to inaccurate or misleading system characterization.
- RTL simulation or emulation considerably reduces system behavior mis-characterization and helps avoid late discovery of system performance problems.

System Overview: Motivation

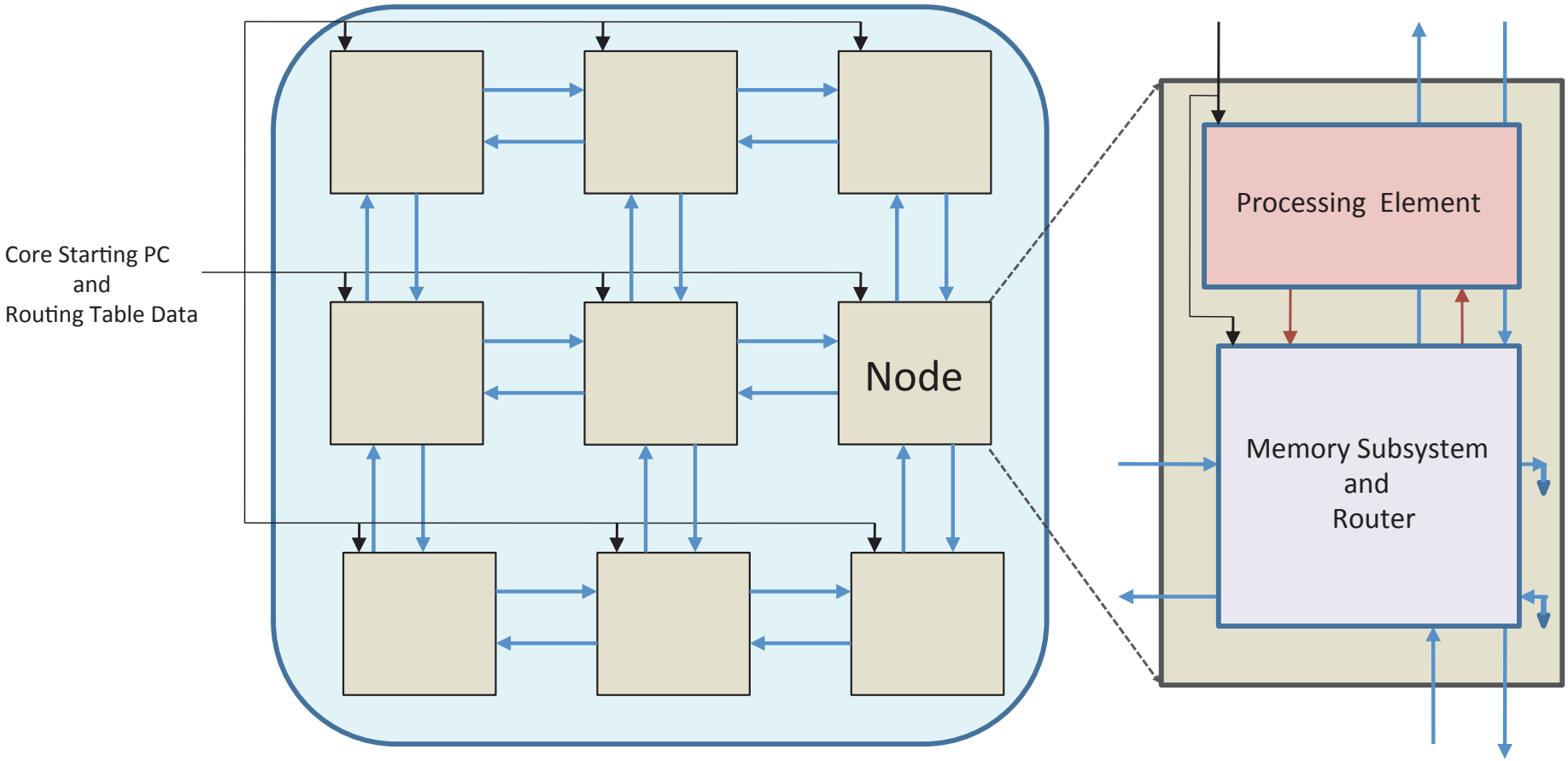
- But RTL simulation/emulation time increases with design size.
- Adopting synthesizable RTL running on FPGA boards can help mitigate the simulation time.
- Current hardware systems were too inflexible: RAMP, LEON, S-Scale, etc…
- Our Solution is Heracles, a component-based, synthesizable, design environment.

System Overview: Design Flow



Processing Units

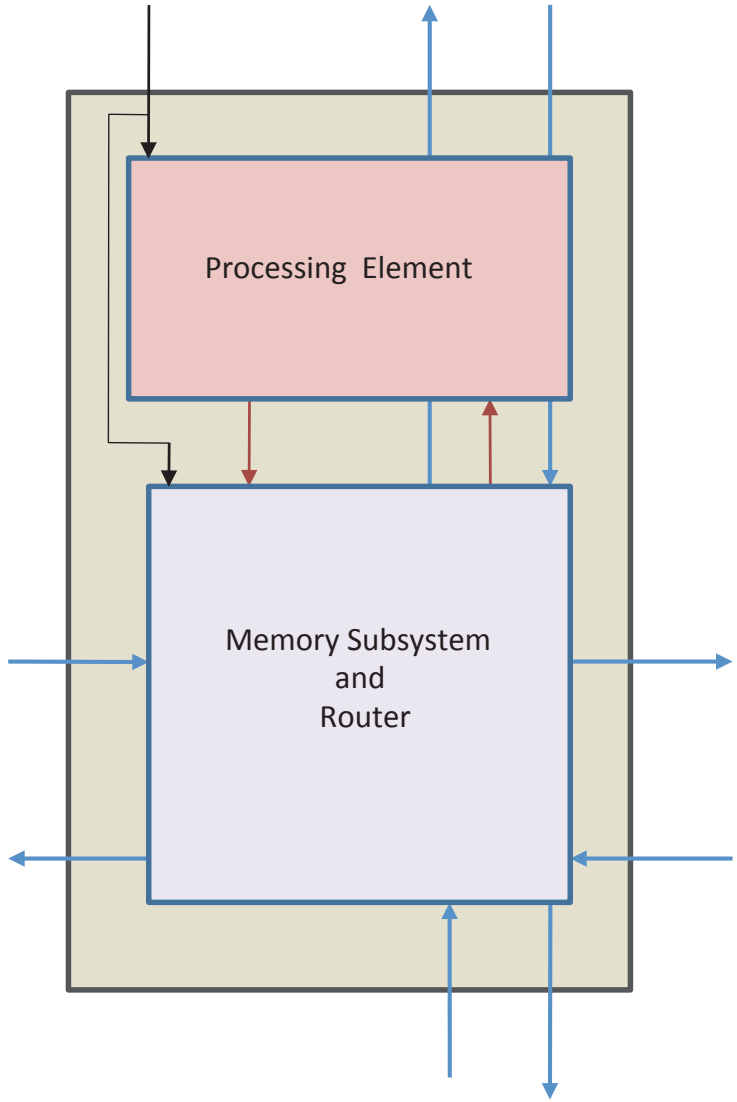
Heracles in a mesh configuration



- The processing element is oblivious to the memory hierarchy and the network topology.

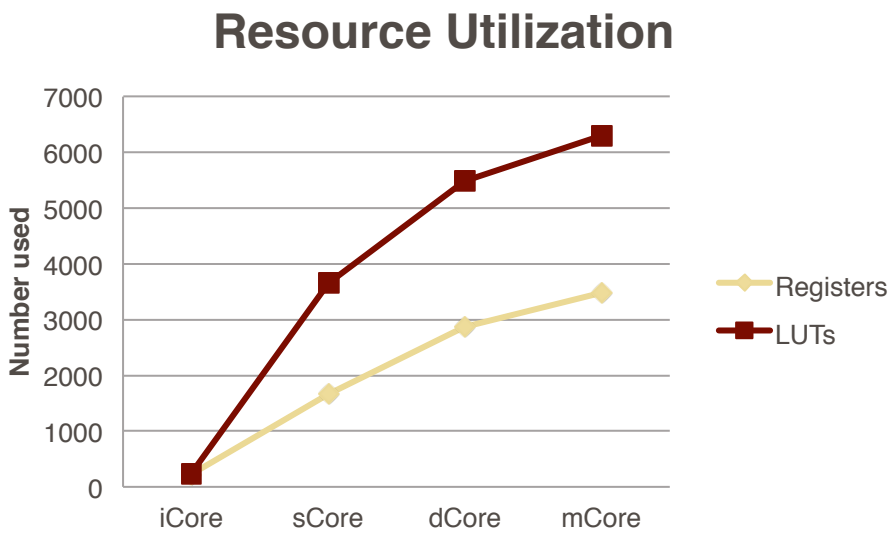
Processing Units: Types

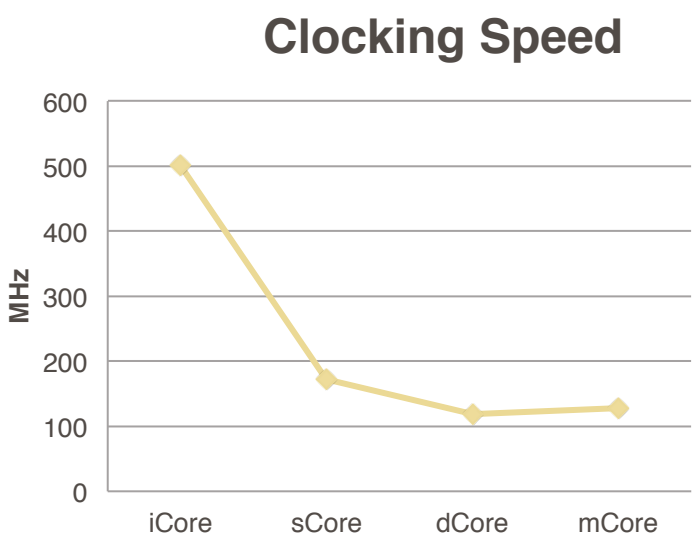
- Injector core
- Single threaded MIPS core
- Two-way hardware threaded MIPS core
- Two-way hardware threaded MIPS core with migration
- All cores have the same interface
- The processing element can be replaced by any core with no change to the rest of the platform.



Processing Units: Comparative Sizes

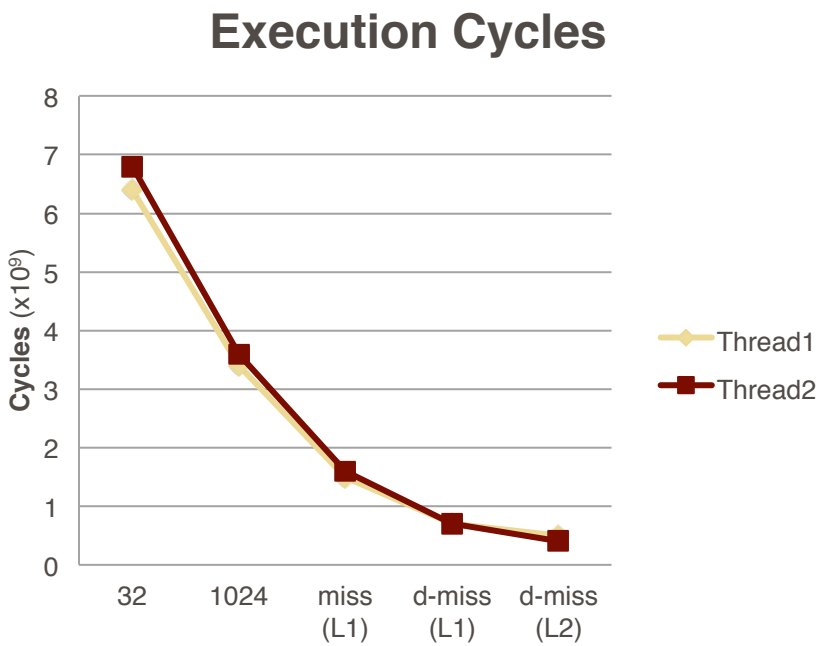
- Synthesis is done using Xilinx ISE Design Suite 11.5, with Virtex-6 LX550T package ff1760 speed -2, as the targeted FPGA board.
- A hardware thread context consists of a program counter (PC), a set of 32 data registers, and one 32-bit state register.
- One hardware thread is active on any given cycle, and pipeline stages are drained between context switches to avoid state corruption.





Processing Units: Multi-Threading

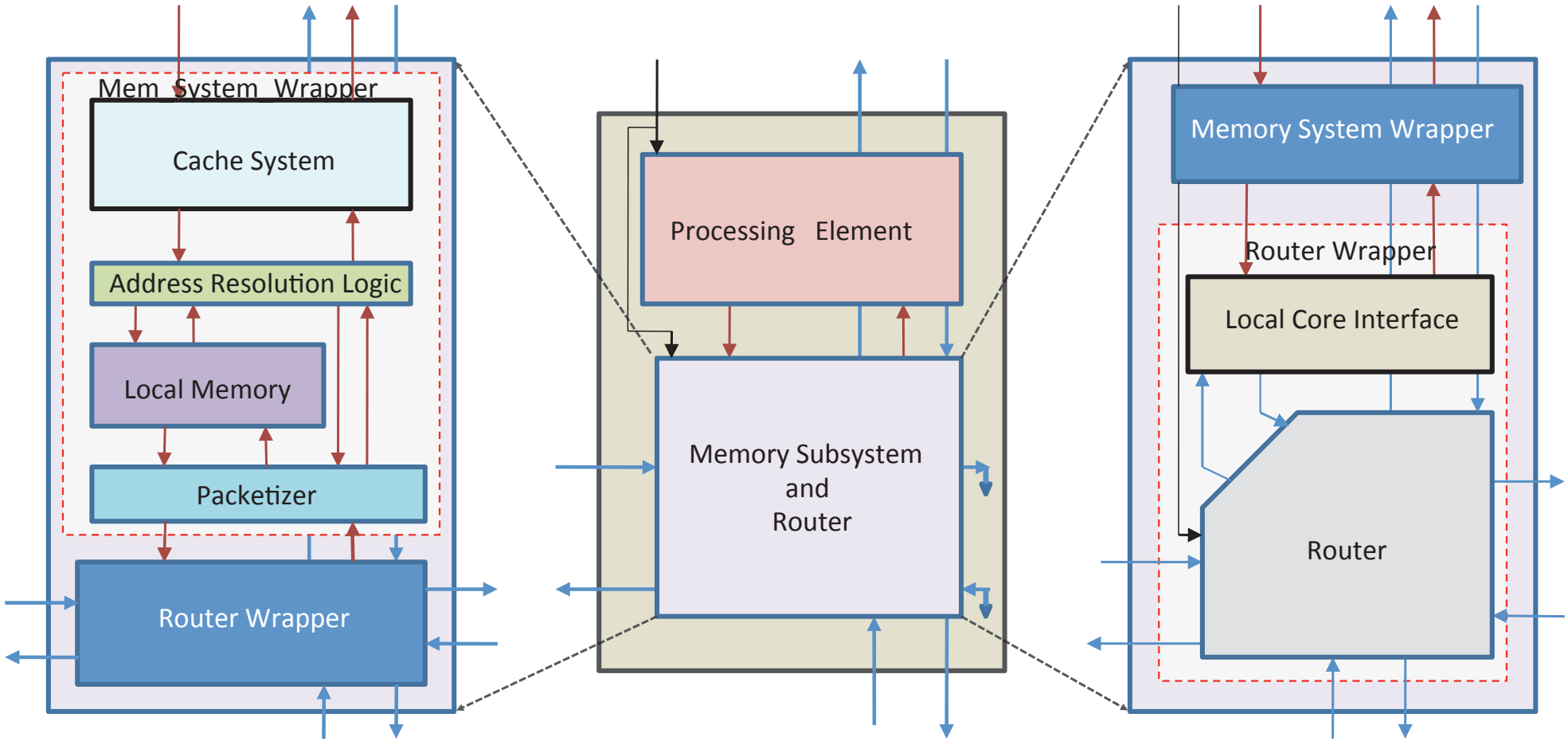
- The effect of hardware multithreading (HMT) on system performance.
- The 197.parser application from the SPEC CINT2000 benchmarks on a single node with the dCore as processing unit.
- Five different execution interleaving policies.
- No network contention (only one core running)



Outline

- Systems Overview
  - Motivation
  - Design Flow
- Processing Units
  - Injector & Single Hardware-Threaded MIPS Cores
  - Two-way & Hardware Context Migration MIPS Cores
- Memory System Organization
  - Main Memory Configuration
  - Caches & Cache Coherence Protocols
- Network-on-Chip (NoC)
  - Flow Control & Routing Algorithms
  - Network Topology Re-configuration
- Programming Models & Toolchain
  - Sequential & Parallel Programming Models
  - Graphical User Interface & Programs Compilation
- Conclusion
  - Summary and Future Work

Memory System Organization



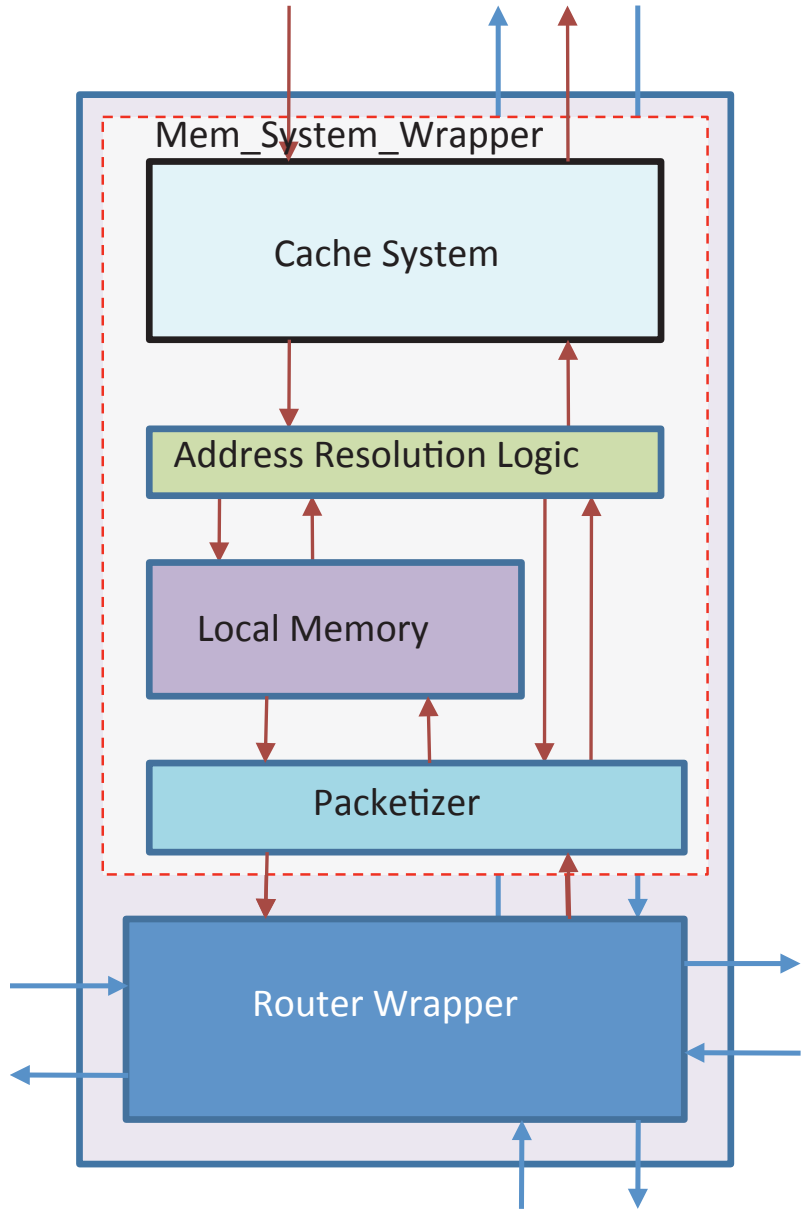
(a) Expanded View of Memory Hierarchy

(b) Expanded View of Router Wrapper

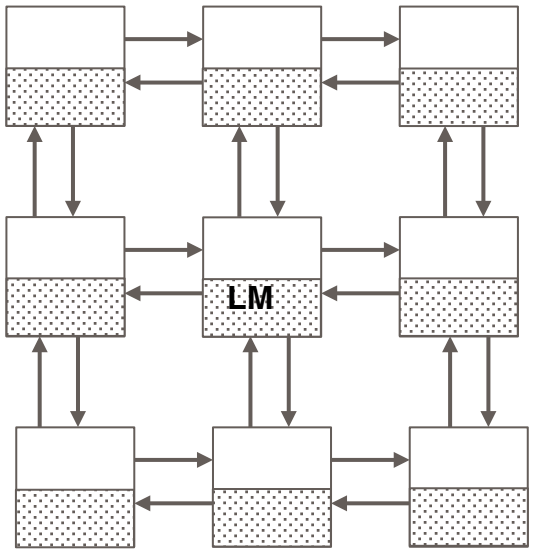
- Memory System Wrapper and Router Wrapper

Memory System Organization

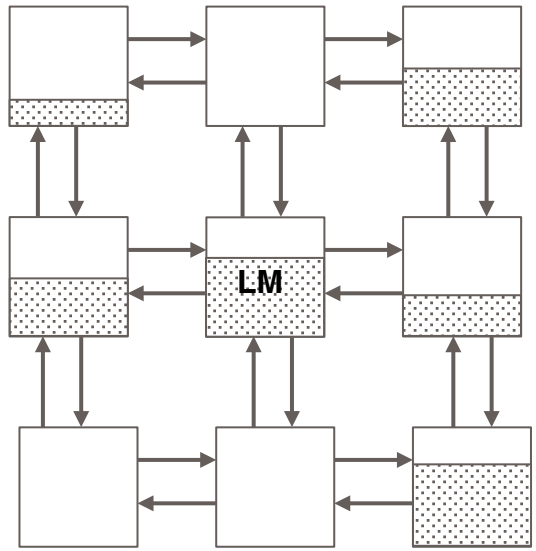
- **Cache system**: data cache & instruction cache can be separate or unified. INDEX_BITS & OFFSET_BITS parameters control the number of cache blocks and their size.
- **Address resolution logic**: determines if a request can be served at the local memory, or if the request needs to be sent over the network.
- **Local block memory**: the LOCAL ADDR BITS parameter is used to set the size of the local memory.



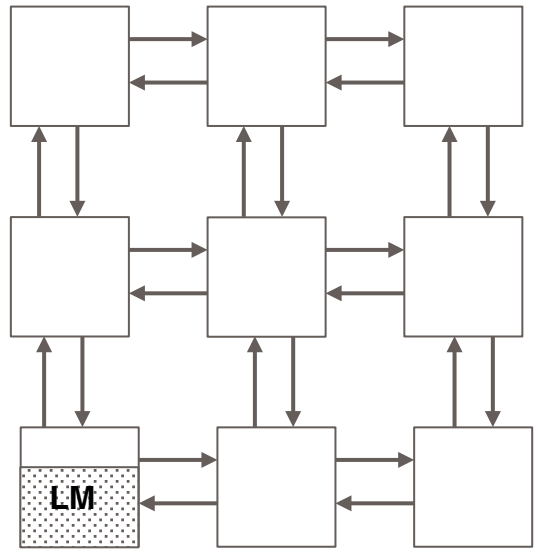
Memory System Organization



Uniform distributed memory



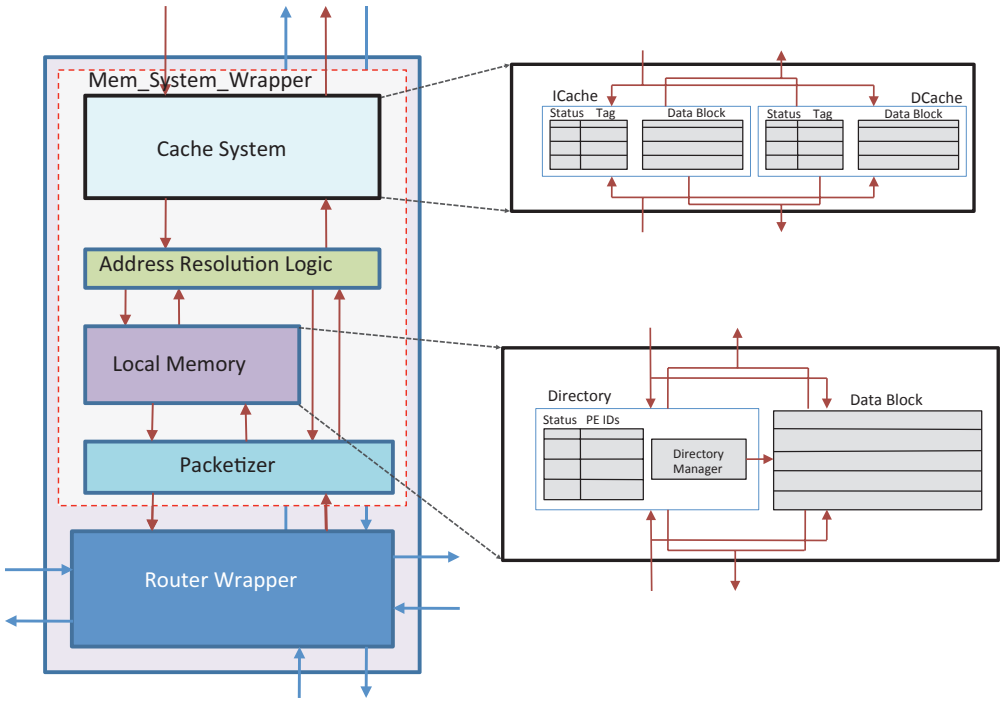
Non-uniform distributed memory

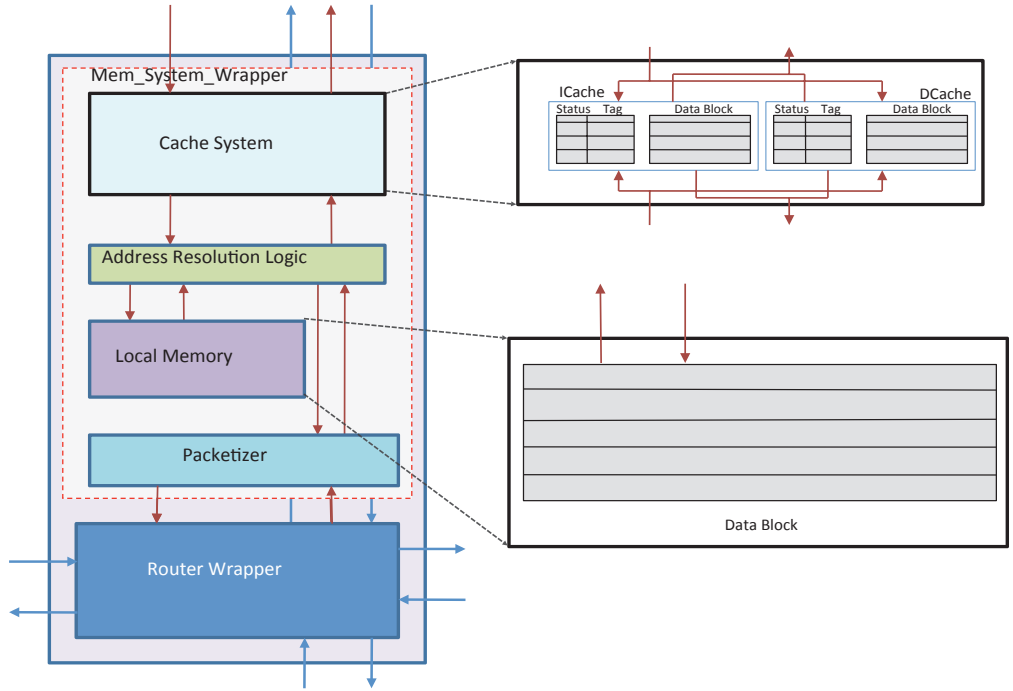


Non-distributed memory

- The main memory is constructed to allow different memory space configurations.
- For Centralized Shared Memory (CSM) implementation, all processors share a single large main memory block.
- In Distributed Shared Memory (DSM), where each processing element has a local memory: size can be changed on a per core-basis for uniform and non-uniform distributed memory.

Caches & Cache Coherence

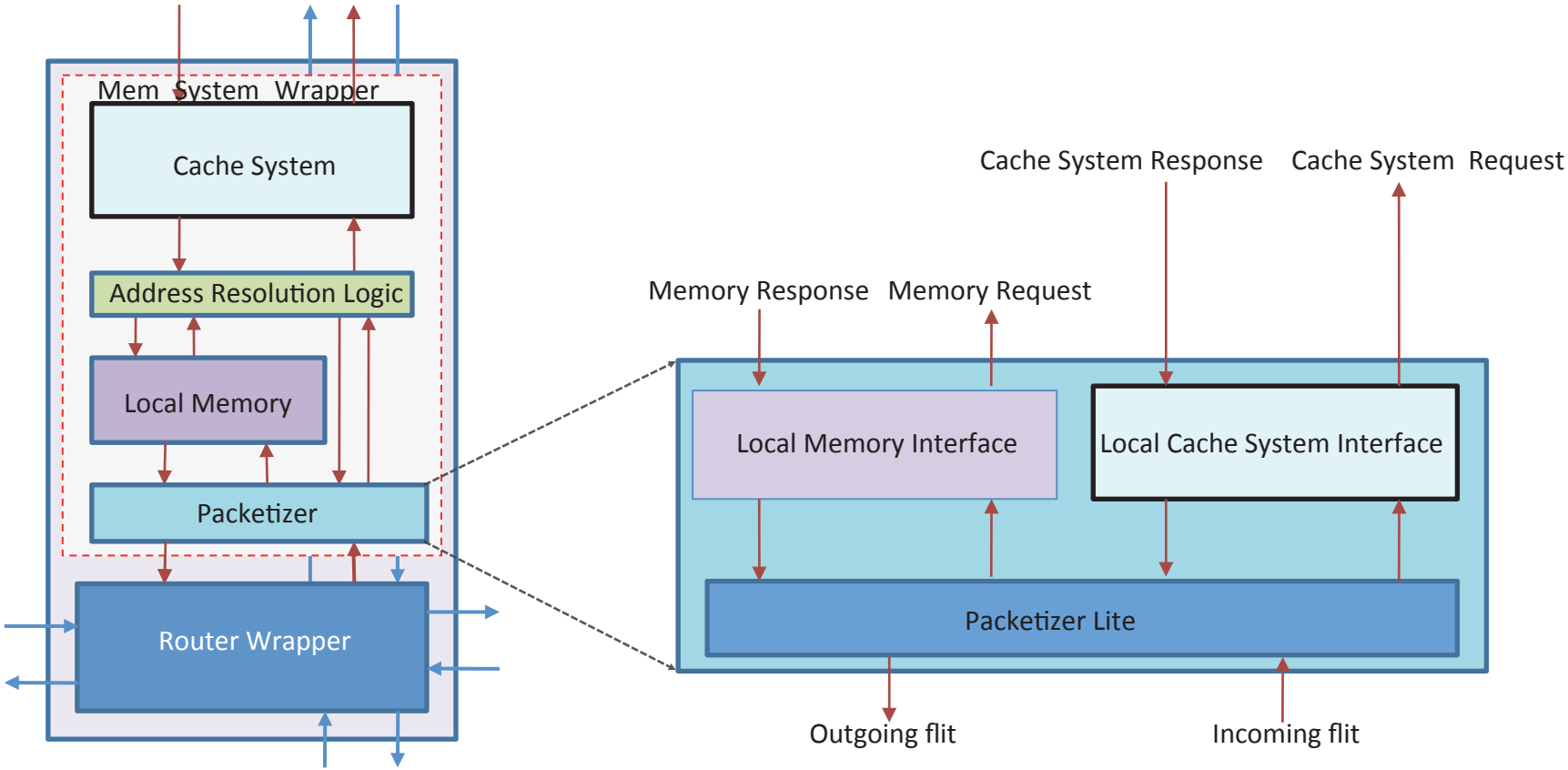




- A directory can be attached to each local memory and the MESI protocol is implemented as the directory-based default coherence mechanism.
- FPGA registers used: 11482

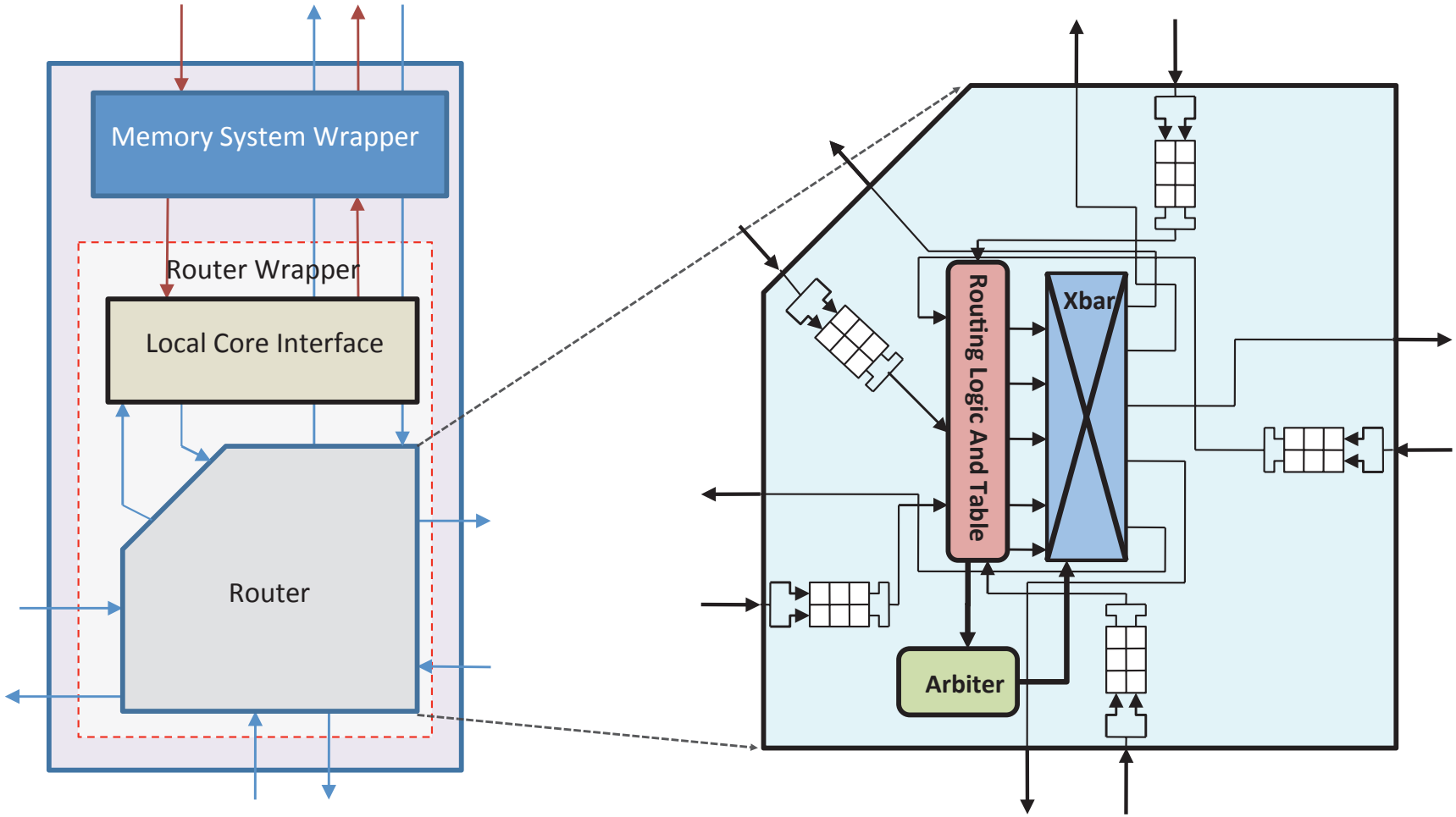
- Cache coherence can also be done via remote access (RA).
- FPGA registers used: 2424

Network-on-Chip: Network Interface



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

Virtual Channel Router Architecture



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

Network-on-Chip: Routing

- The routing operation has 4 phases:
  1. **Route computation** (RC), it determines the next hop for the packet .
  2. **Virtual-channel allocation** (VA), it allocates a virtual channel in the next hop.
  3. **Switch allocation** (SA), the flit competes for a switch port.
  4. **Switch traversal** (ST), it moves flit onto the output port link.

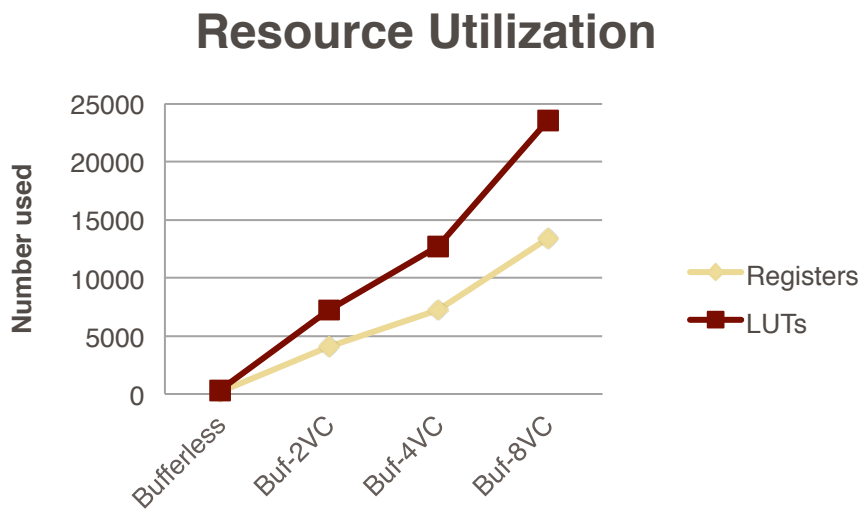
- In current version each phase corresponds to a router pipeline stage.

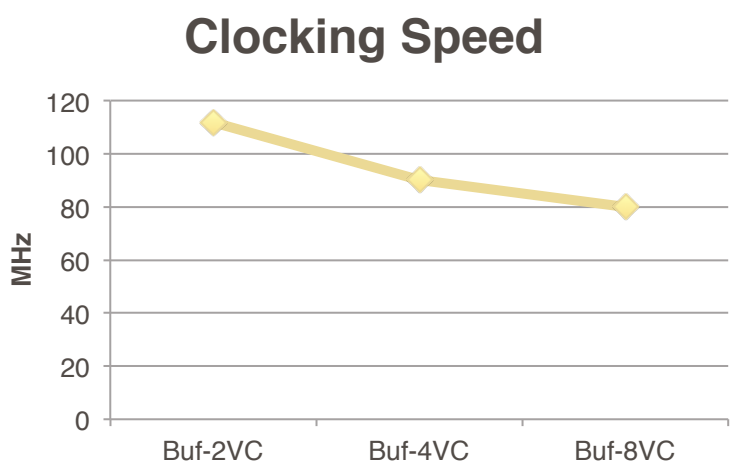
- The switch allocation (SA) stage is the critical stage in this design, due to the logical complexity of the arbiter.

- It supports both fixed logic (e.g, DOR) and table routing (e.g., BSOR) based oblivious routing algorithms.

Network-on-Chip: VC Effects

- The RT_ALG parameter is used to select the proper routing algorithm for a given application and topology.
- Its supports for both static and dynamic virtual channel allocation.
- The key take-away is that larger number of VCs at the router increases both the router resource utilization and the critical path.





Network Topology Configuration

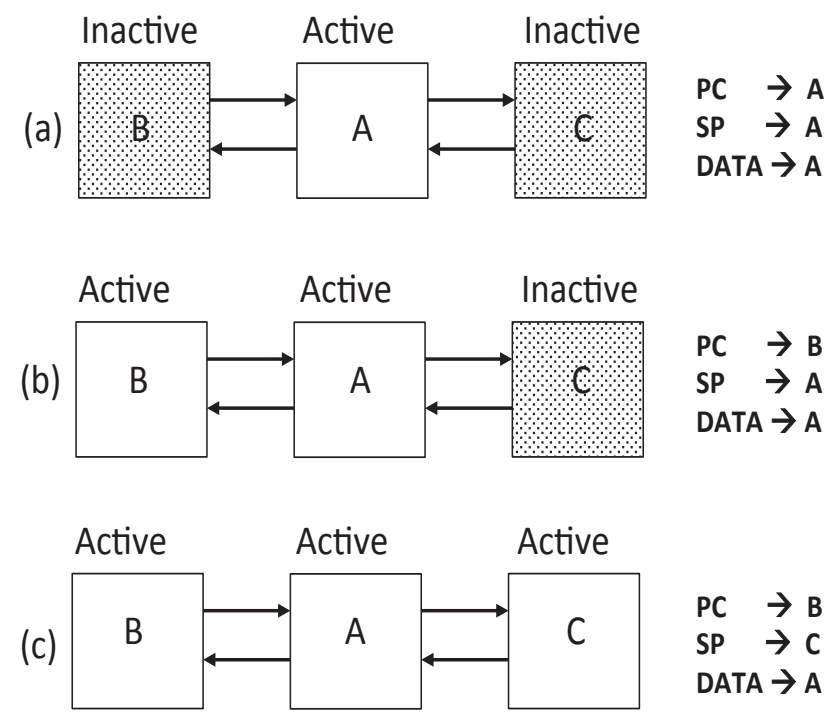
- The parameterization of the number of input ports and output ports gives the platform the flexibility to metamorphose into different network topologies.
- For example, k-ary n-cube, 2D-mesh, 3D-mesh, hypercube, ring, or tree.
- A new topology is constructed by changing the IN_PORTS, OUT_PORTS, and SWITCH_TO_SWITCH parameters and reconnecting the routers.
- In the case of the 3D-mesh, IN_PORTS and OUT_PORTS parameters are set to 2 or 3, one to connect the router to the local core and the rest to the third dimension.

Outline

- Systems Overview
  - Motivation
  - Design Flow
- Processing Units
  - Injector & Single Hardware-Threaded MIPS Cores
  - Two-way & Hardware Context Migration MIPS Cores
- Memory System Organization
  - Main Memory Configuration
  - Caches & Cache Coherence Protocols
- Network-on-Chip (NoC)
  - Flow Control & Routing Algorithms
  - Network Topology Re-configuration
- Programming Models & Toolchain
  - Sequential & Parallel Programming Models
  - Graphical User Interface & Programs Compilation
- Conclusion
  - Summary and Future Work

Programming Models

- In the sequential programming model: single entry point (starting program counter-PC) and single execution thread.
- Local memory of executing core has instruction binary, stack frame, and application data (a).
- Instruction binary resides at another core's local memory, stack frame and application data are local (b).
- Instruction binary and application data reside at other cores.
- Local memory is completely uninvolved in application execution.



Programming Models: Threading

```
...
int HHThread1 (int *array, int item, int size) {
    sort(array, size);
int output = search(array, item, size);
return output;
}

int HHThread2 (int input,int src,int aux,int dest) {
int output = Hanoi(input, src, aux, dest);
return output;
}
...
```

- The keywords HHThread1 and HHThread2 are provided to specify the part of the program to execute on each hardware thread.

```
@e2        // <HHThread1>
27bdffd8   // 00000388 addiu sp,sp,-40
...
8fc60030   // 000003bc lw a2,48(s8)
0c00004c   // 000003c0 jal 130 <search>
...
00000000   // 000003e4 nop

@fa        // <HHThread2>
27bdffd8   // 000003e8 addiu sp,sp,-40
...
8fc70034   // 00000414 lw a3,52(s8)
0c0000ab   // 00000418 jal 2ac <Hanoi>
...
03e00008   // 00000438 jr ra
00000000   // 0000043c nop

@110 // <HHThread1_Dispatcher>
```

- Associated binary

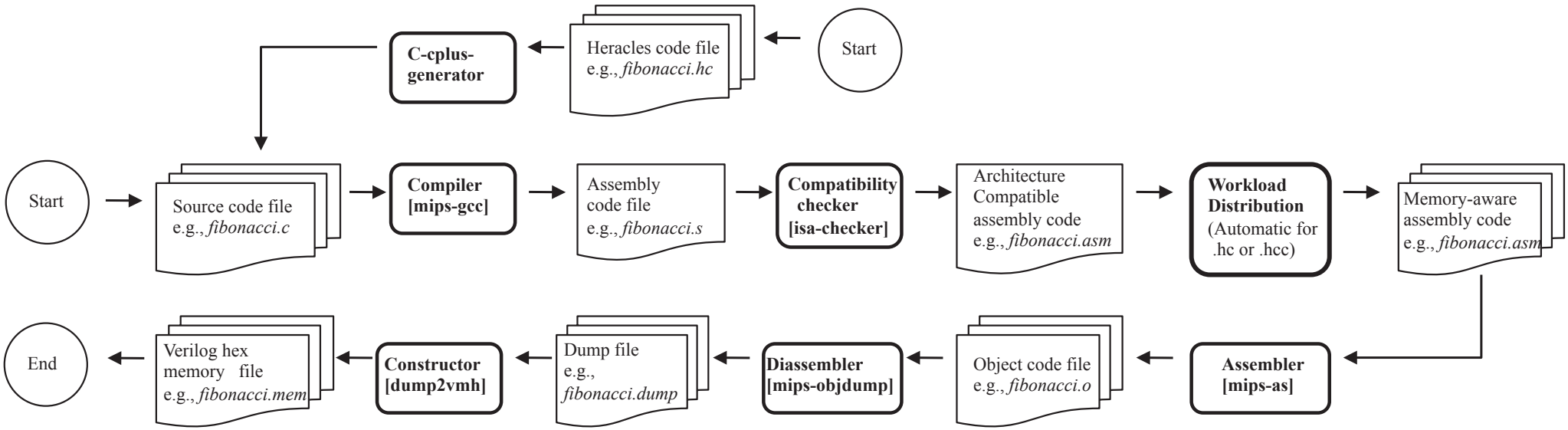
Programming Models: Mapping

```
...
#pragma Heracles core 0 {
   // Synchronizers
   HLock lock1,  lock2;
   HBarrier bar1, bar2;

   // Variables
   HGlobal int arg1, arg2, arg3;
   HGlobal int Arr[4][4];

   // Workers
   #pragma Heracles core 1 {
      start_check(50);
      check_lock(&lock1, 1000);
      matrix_add(Arr, Arr0, Arr1, arg1);
      clear_barrier(&bar1);
   }

   #pragma Heracles core 2 {
      start_check(50);
      check_lock(&lock1, 1000);
      matrix_add(Arr, Arr0, Arr1, arg2);
      clear_barrier(&bar2);
   }
}
 ...
```

```
 ...
int core_0_lock1, core_0_lock2;
int core_0_bar1, core_0_bar2;
int core_0_arg1, core_0_arg2, core_0_arg3;
int core_0_Arr[4][4];
...
void  core_0_work (void)
{
   // Synchronizers
   // Variables
   // Workers
   // Main function
```

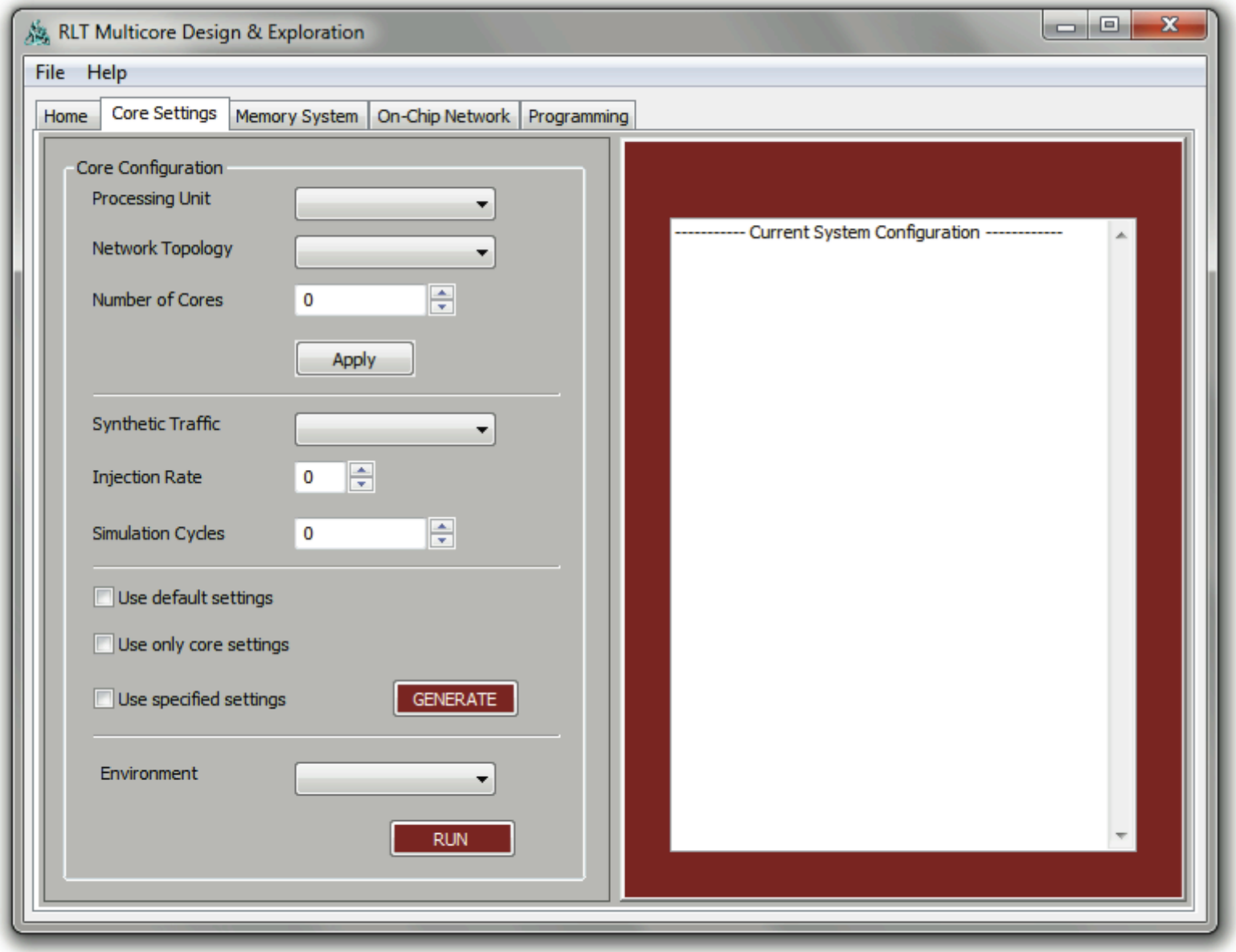
- Heracles uses OpenMP style pragmas to allow users to directly compile multi-threaded programs onto cores.
- Users specify programs or parts of a program that can be executed in parallel and on which cores.

Software Toolchain

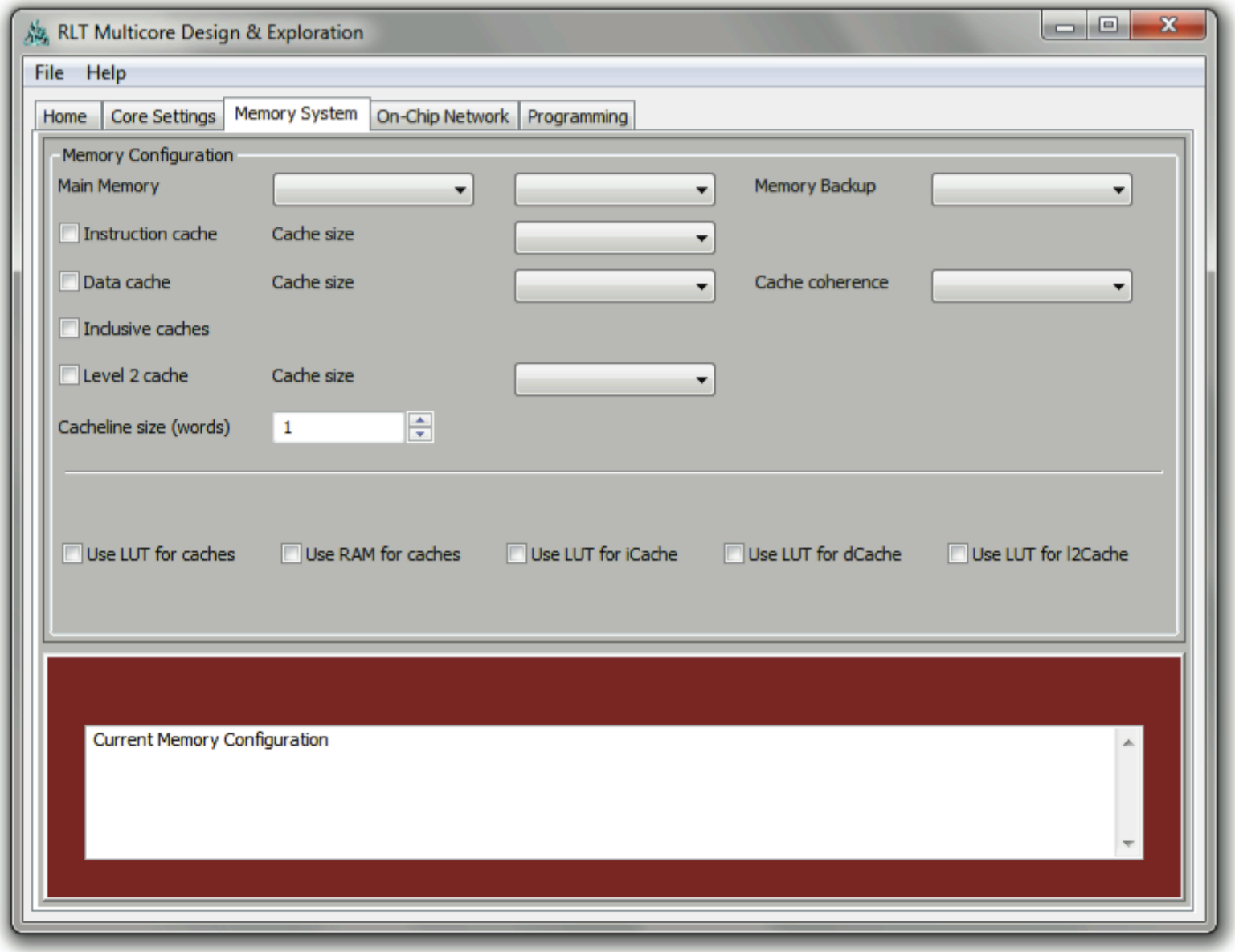


- The Heracles environment has an open-source compiler toolchain to assist in developing software for different system configurations. The current version of the toolchain is built around the GCC MIPS cross-compiler using GNU C version 3.2.1.

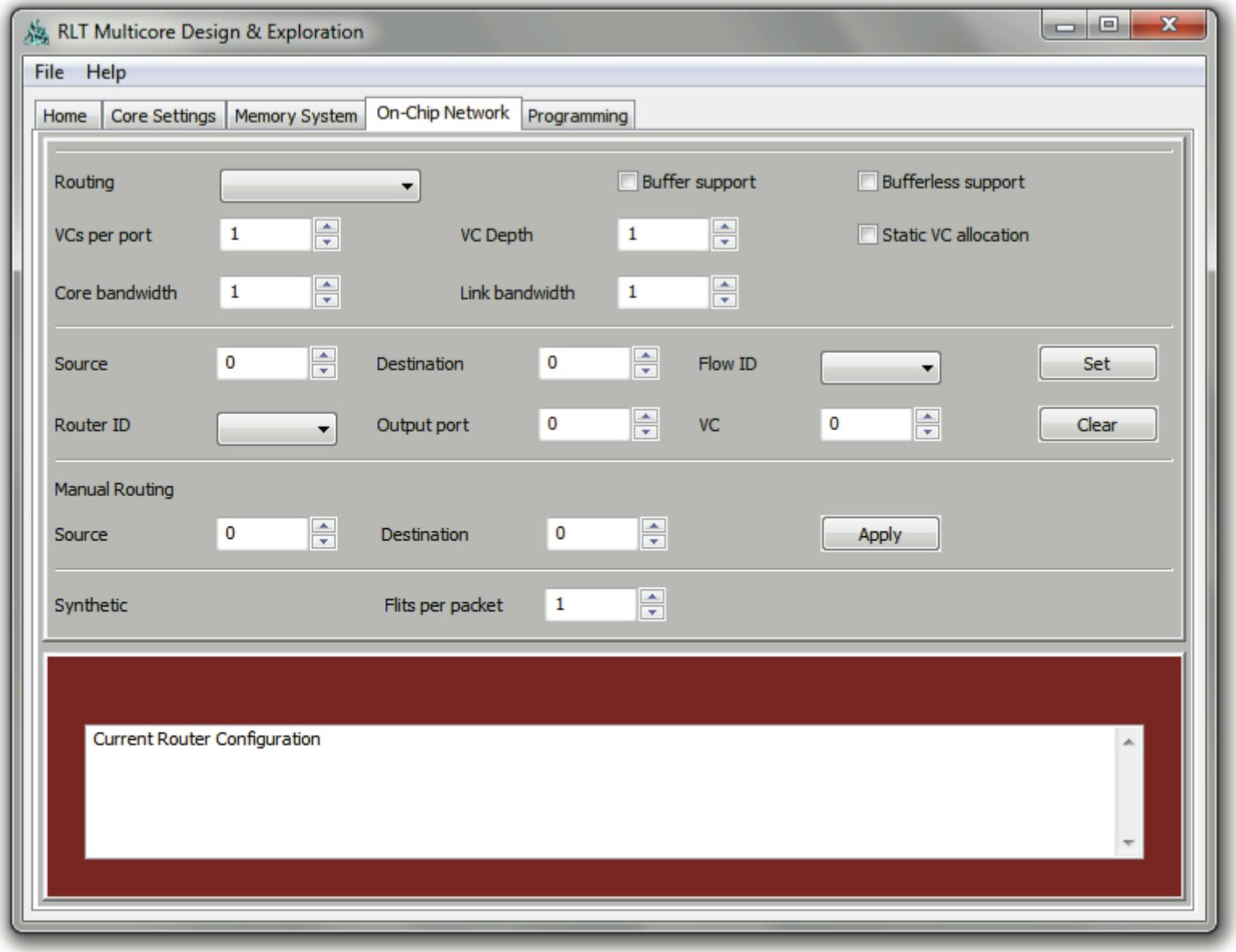
Graphical User Interface



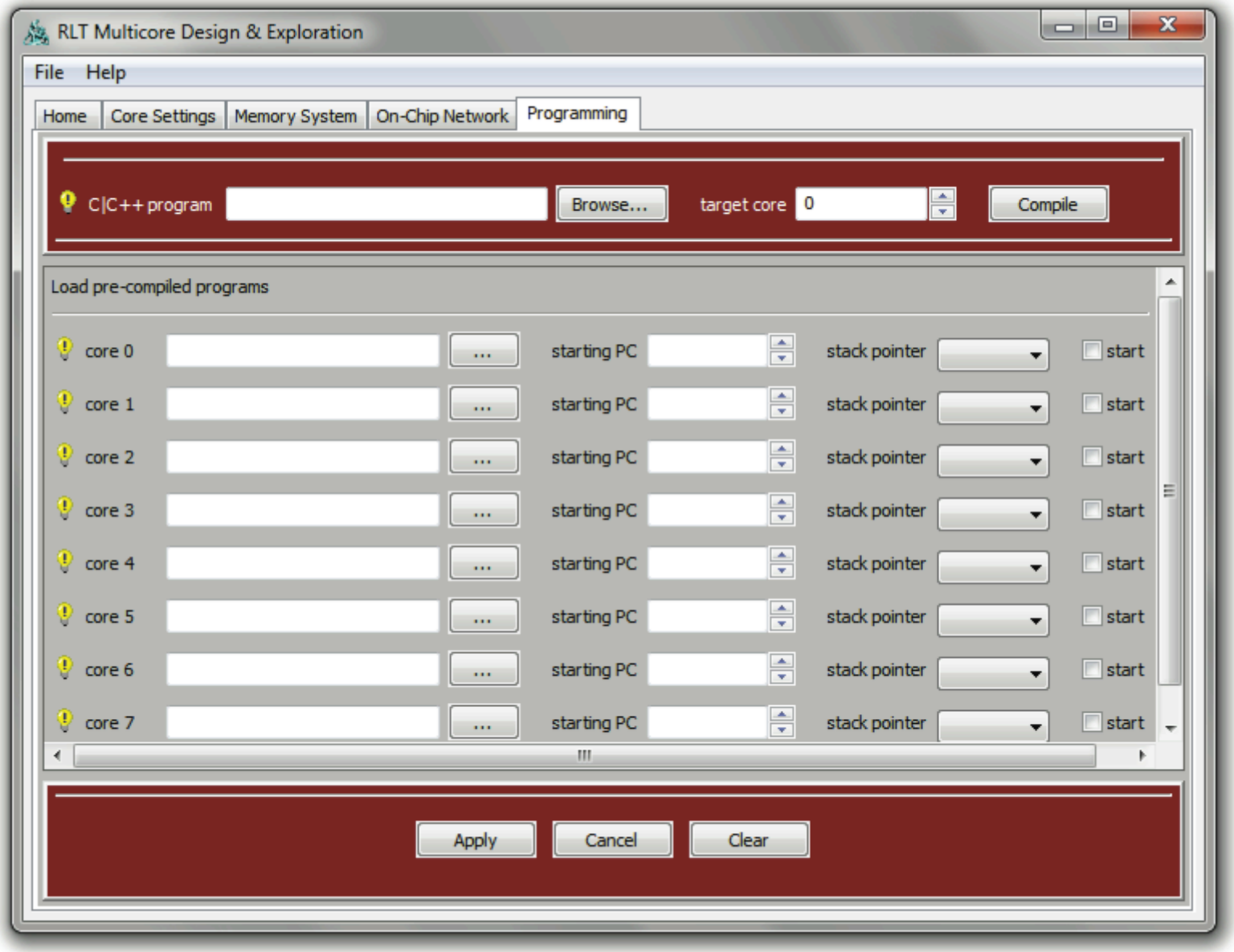
Graphical User Interface



Graphical User Interface



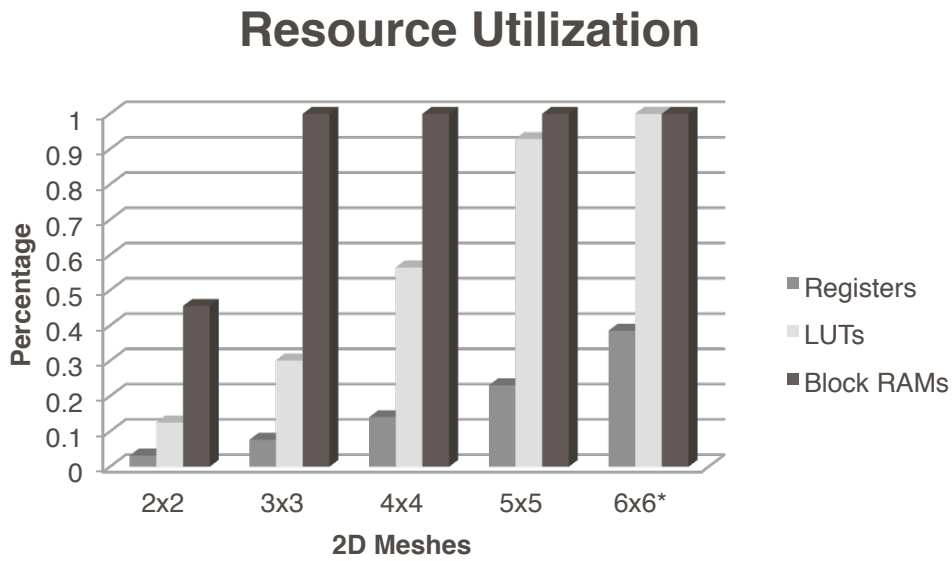
Graphical User Interface



Examples of Full System Evaluation

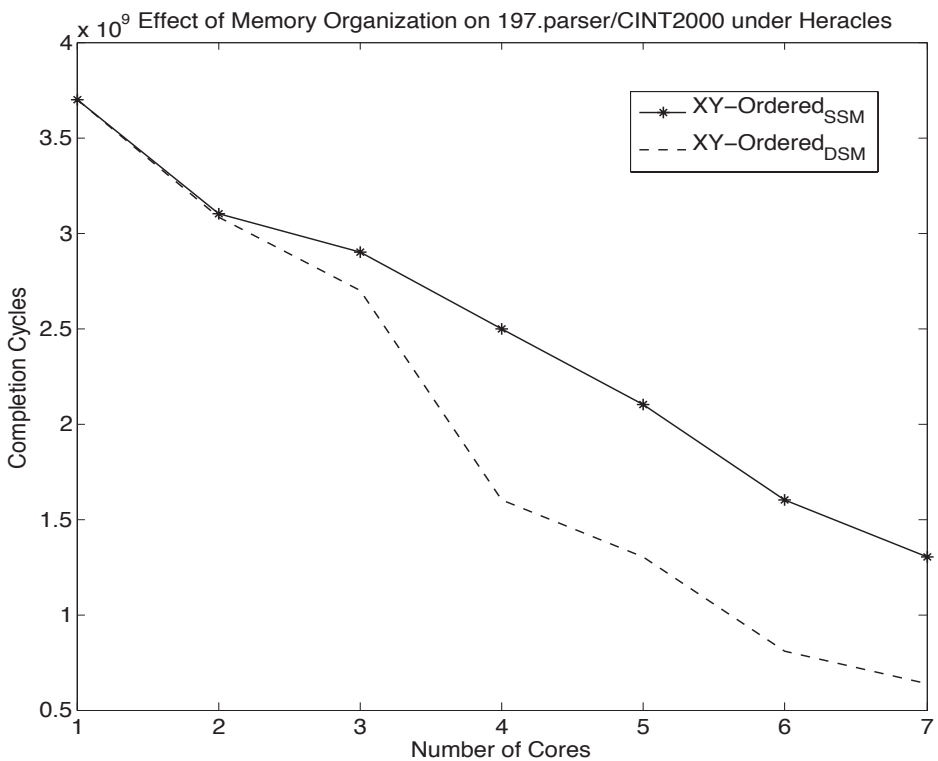
	Heracles
Core	
ISA	32-Bit MIPS
Multiply/Divide	Software
Floating Point	Software
Pipeline Stages	7
Bypassing	Full
Branch policy	Always non-Taken
Outstanding memory requests	1
Address Translation	None
Level 1 Instruction/Data Caches	
Associativity	Direct
Size	16KB
Outstanding Misses	1
On-Chip Network	
Topology	2D-Mesh
Routing Policy	DOR and Table-based
Virtual Channels	2
Buffers per channel	8

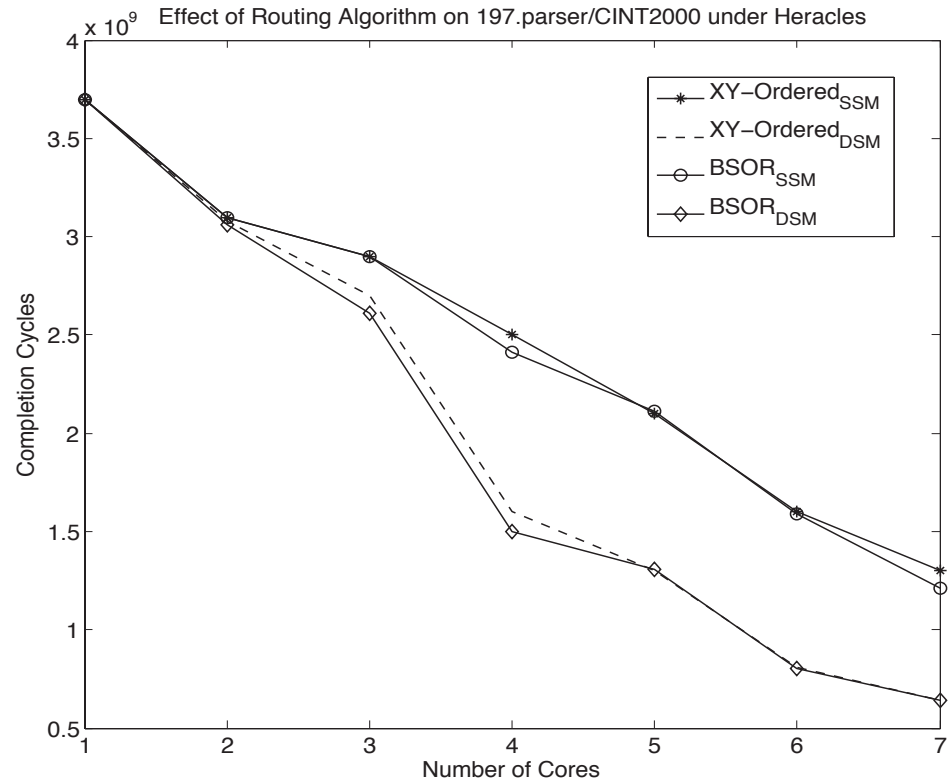
2D-MESH Heracles MULTICORE ARCHITECTURE



- In the 2x2 configuration, the local shared memory is set to 260KB per core, In the 3x3 configuration the size is reduced to 64KB per core, and in the 4x4 configuration it is set to 32KB.

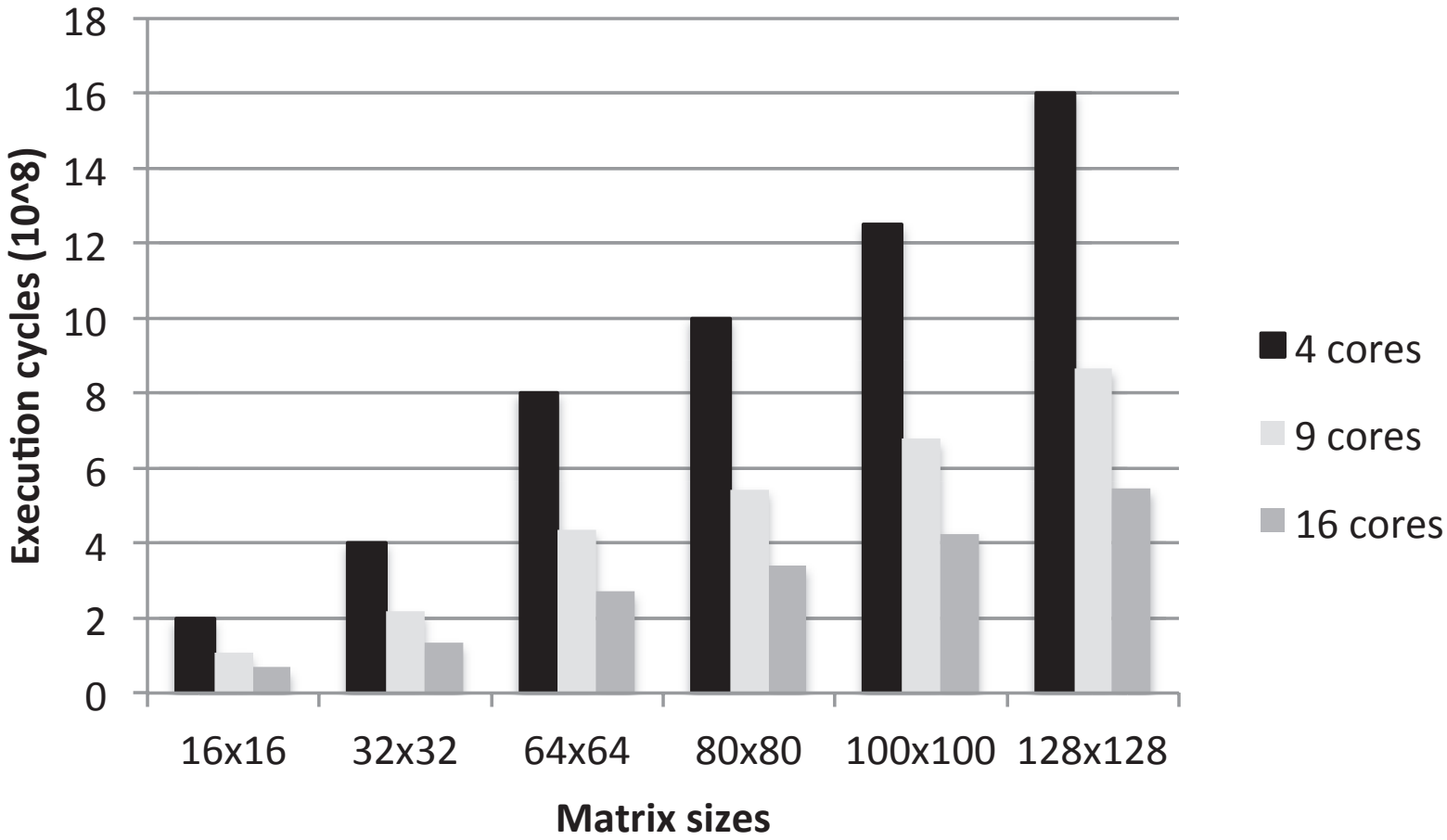
Examples of Full System Evaluation





Full C/C++ applications can be executed on Heracles, and various system performance aspects can be evaluated.

Examples of Full System Evaluation



Matrix multiplication acceleration tests.

Conclusion & Future Work

CSAIL

- *Heracles* allows to build complete, realistic, fully parameterized, synthesizable, modular, multi/many-core architectures.
- It uses a component-based design approach, where the processing element or core, the router and the network-on-chip, and the memory subsystem are independent building blocks, and can be used in other designs.
- Hardware modules are FPGA-type independent.
- Future work will involve adding a small kernel binary code to each core on start up for handling exceptions and proper interrupts for peripheral communications.
- Multi-threading and dynamic runtime workload management among the cores, via thread migration, will be explored.

Q & A Section

CSAIL

Thank You !

More Information at:

http://projects.csail.mit.edu/heracles

We thank Joel Emer, Li-Shiuan Peh, Omer Khan, Myong Hyon Cho,
and Noah Keegan for interesting discussions throughout the course of this work.