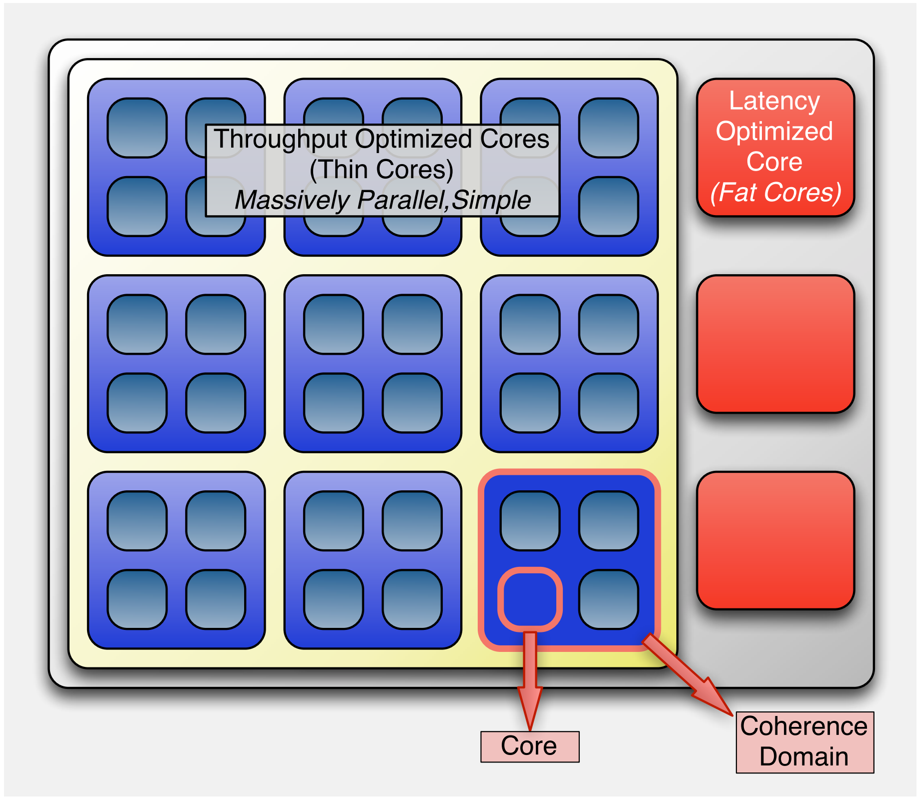
OpenSoC Fabric: On-Chip Network Generator

Using Chisel to Generate a Parameterizable
On-Chip Interconnect Fabric

Farzad Fatollahi-Fard, David Donofrio, George Michelogiannakis, John Shalf

MODSIM 2014 Presentation – 2014 Aug 13

Abstract Machine Model For Emerging Node Architectures



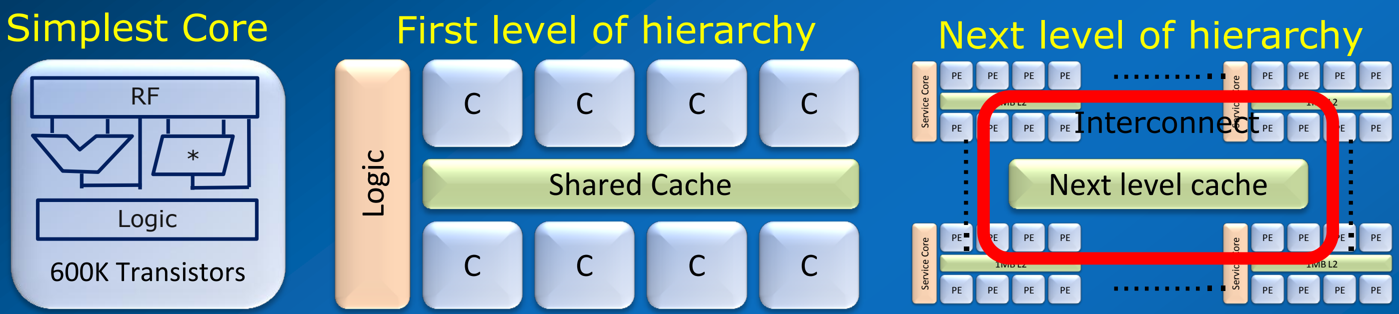
- The number of cores on a chip will be on the order of 1000s
- Maintaining cache coherence is NOT scalable
  - Expect coherence domains
- Flat and infinitely fast on-chip interconnect is NO longer practical
  - Expect complex NOCs
- Processing elements within a node are NOT equidistant.
  - Expect non-uniformity

Download the CAL AMM doc from http://www.cal-design.org/

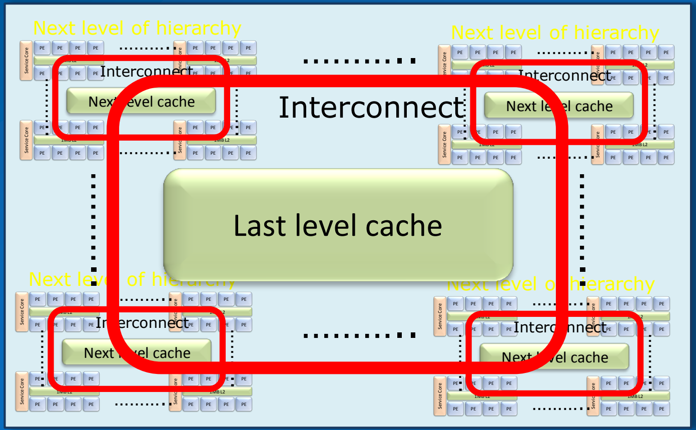
Motivation

Straw-man Exascale Processor

Shekhar Borkar, IPDPS 2013



Processor

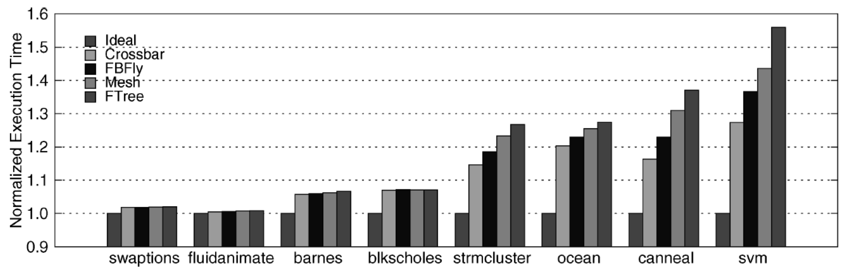


Technology	7nm, 2018
Die area	500 mm2
Cores	2048
Frequency	4.2 GHz
TFLOPs	17.2
Power	600 Watts
E Efficiency	34 pJ/Flop

Computations alone consume 34 MW for Exascale

Impact of NoCs

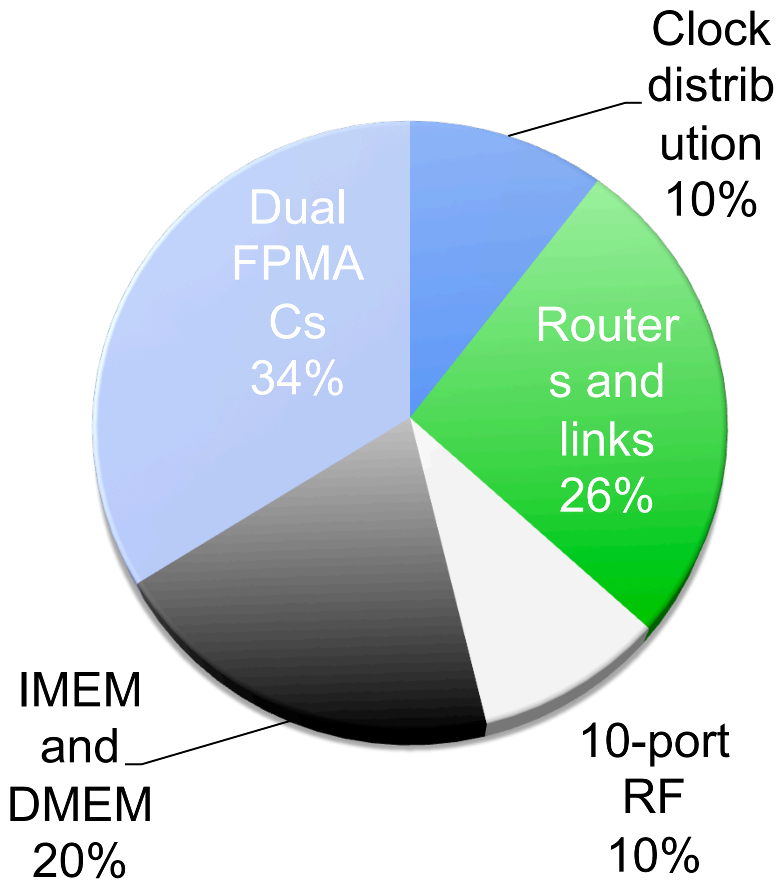
Application Performance



An analysis of on-chip interconnection networks for large-scale chip multiprocessors

ACM Transactions on computer architecture and code optimization (TACO), April 2010

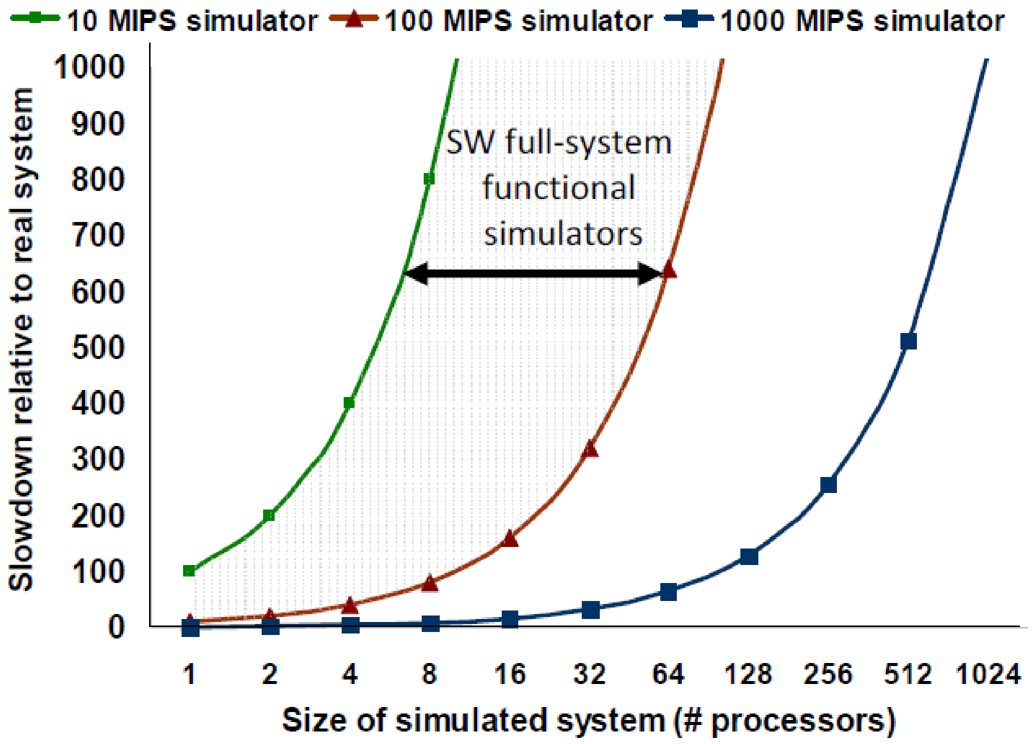
Power



A 5-GHz Mesh Interconnect for a Teraflops Processor. *IEEE Micro*. 2007

Evaluation Infrastructure

- Software simulation slow for large-scale chips and systems
  - Also does not evaluate cycle time
- Hardware emulation requires a large development effort
  - Offers limited internal visibility for statistics



A complexity-effective architecture for accelerating full-system multiprocessor simulations using FPGAs. FPGA 2008

Building an SoC from IP Logic Blocks

It's Legos with a some extra integration and verification cost

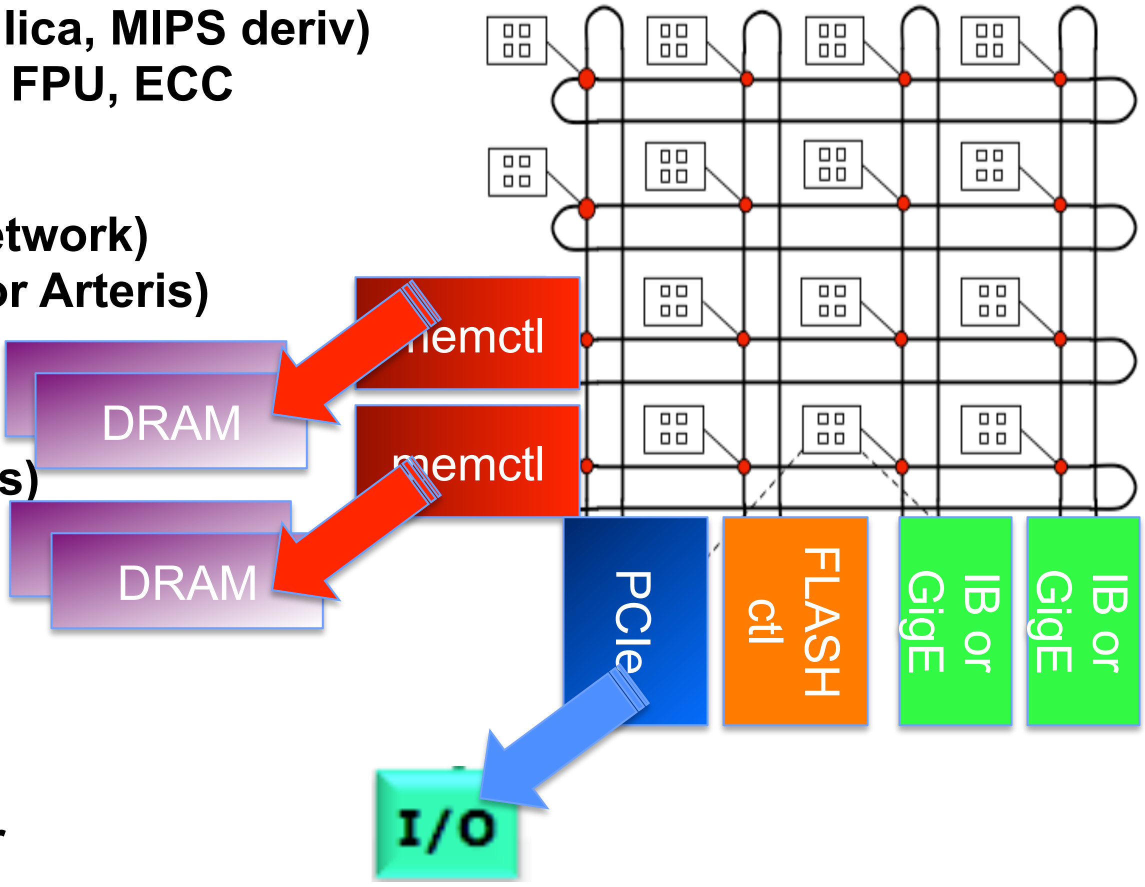
Processor Core (ARM, Tensilica, MIPS deriv)
With extra "options" like DP FPU, ECC

OpenSoC Fabric (on-chip network)
(currently proprietary ARM or Arteris)

DDR memory controller
(Denali/Cadence, SiCreations)
+ Phy & Programmable PLL

PCIe Gen3 Root complex

Integrated FLASH Controller

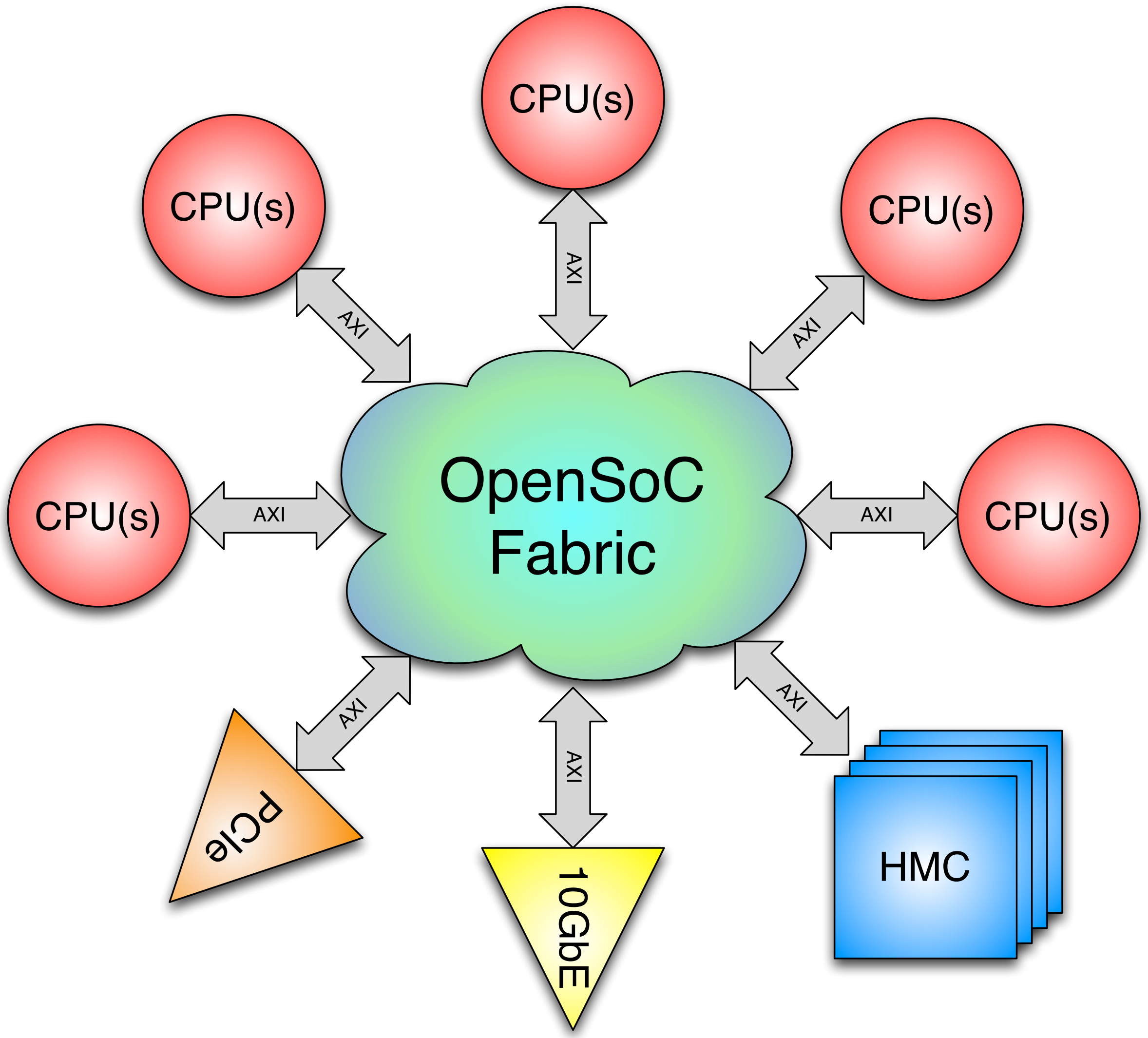


10GigE or IB DDR 4x Channel

Design Priorities

- Primary goal is re-configurability
  - Provide a powerful collection of parameters
  - Make creating new modules and replacing existing ones easy
- Fast verification of hardware and software models
- Provide standardized connection interface
- Invite community to participate and contribute

Abstract Block Diagram



Example Instantiation

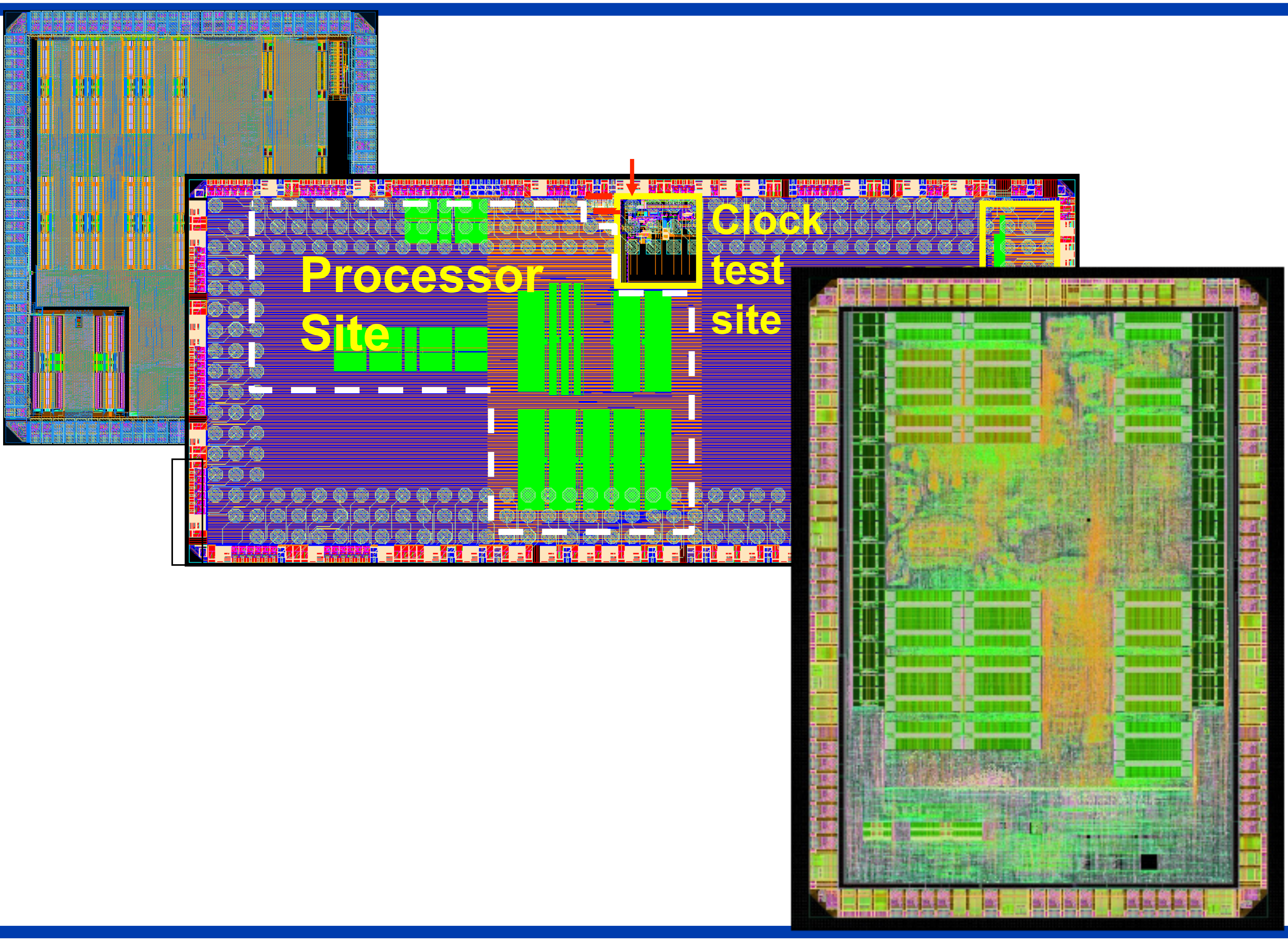
```
class MyOpenSOC extends Module {
  val myCPUs = Vec.fill(5) {Module(new CPU())}
  val myHMC = Module(new HMC())
  val myTenGbE = Module(new TenGbE())
  val myPCIe = Module(new PCIe())
  val NumNodes = 8

  val OpenSoCFabric = Module(new MyOpenSoCFabric(NumNodes))

  for (i <- 0 until 4) {
    myCPUs(i).io.AXIPort <> MyOpenSoCFabric.io.AXI(i)
  }
  myHMC.io.AXIPort <> MyOpenSoCFabric.io.AXI(5)
  myTenGbE.io.AXIPort <> MyOpenSoCFabric.io.AXI(6)
  myPCIe.io.AXIPort <> MyOpenSoCFabric.io.AXI(7)
}
```

What is Chisel?

- **C**onstructing **H**ardware **I**n a **S**cala **E**mbedded **L**anguage
- An open-source hardware construction language developed at the ASPIRE Lab at UC Berkeley
- Hierarchical + Object Oriented + Functional Construction
- Generates both software and hardware
  - Easy functional verification with software model (C++)
  - Use hardware (Verilog) in FPGAs or ASICs

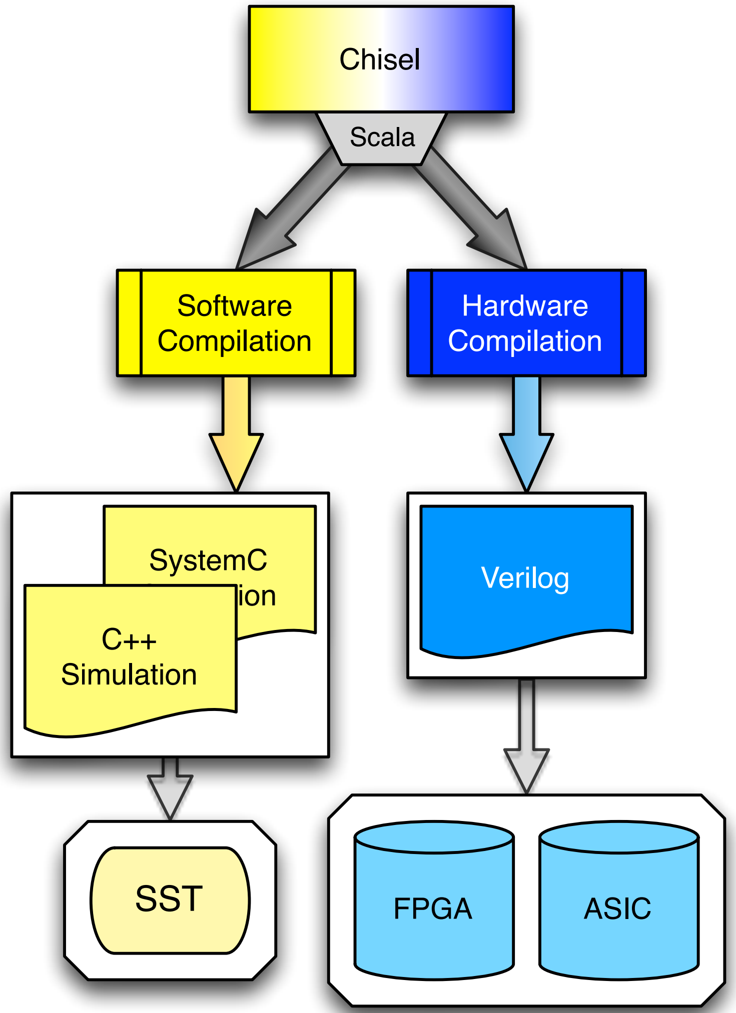
```
import Chisel._

class GCD extends Module {
  val io = new Bundle {
    val a  = UInt(INPUT,  16)
    val b  = UInt(INPUT,  16)
    val e  = Bool(INPUT)
    val z  = UInt(OUTPUT, 16)
    val v  = Bool(OUTPUT)
  }
  val x  = Reg(UInt())
  val y  = Reg(UInt())
  when   (x > y) { x := x - y }
  unless (x > y) { y := y - x }
  when (io.e) { x := io.a; y := io.b }
  io.z := x
  io.v := y === UInt(0)
}

object Example {
  def main(args: Array[String]): Unit = {
    chiselMain(args,
      () => Module(new GCD()))
  }
}
```

ASICs in Chisel
Raven 28nm

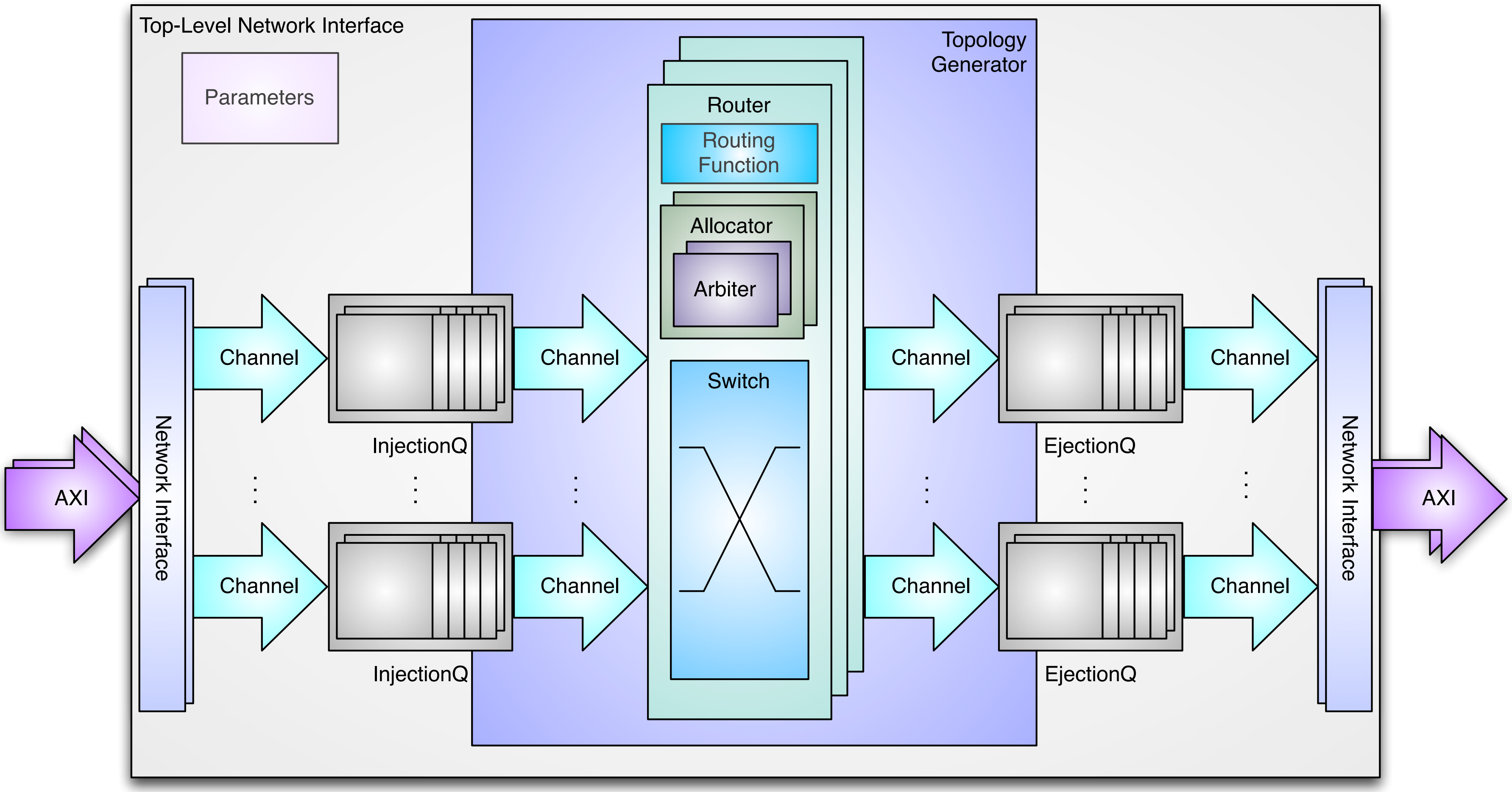


Why Chisel?

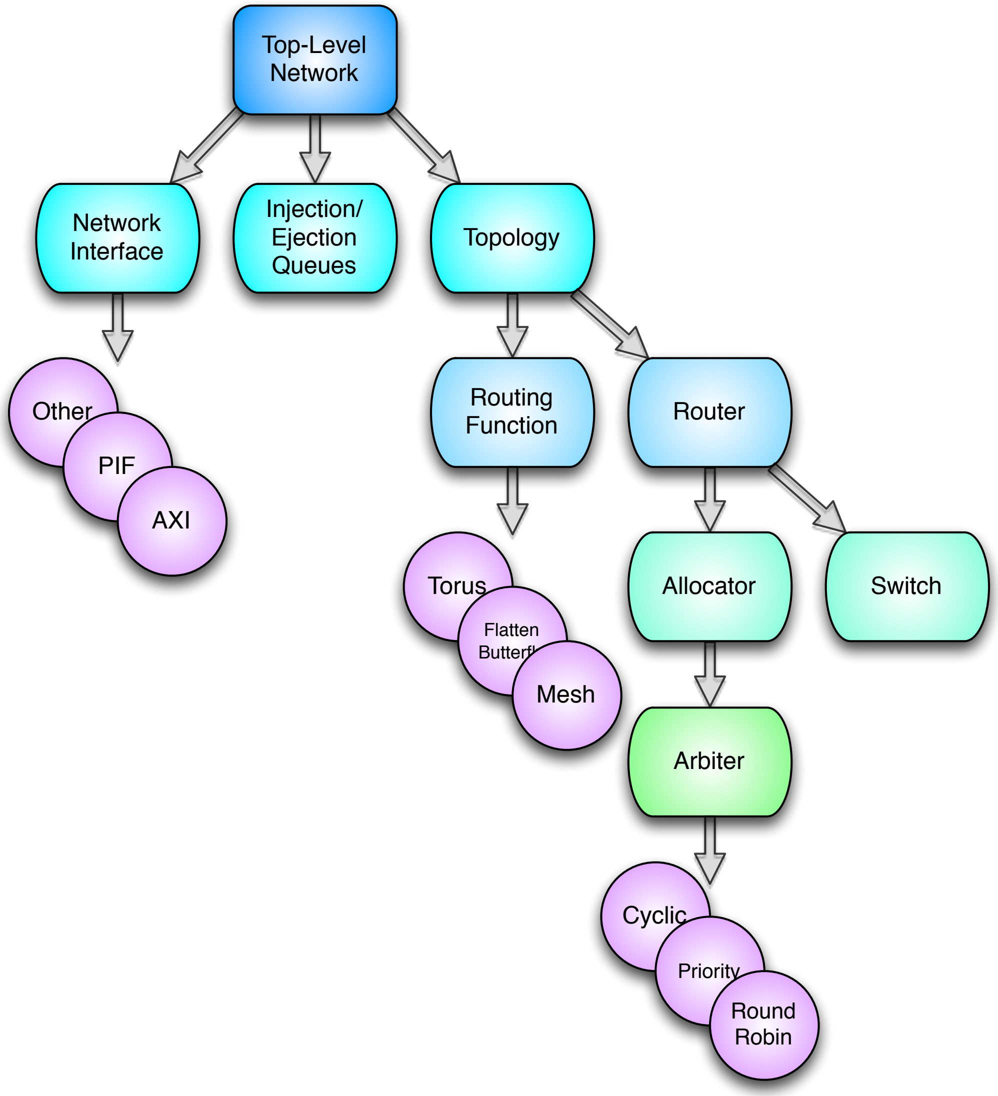
- For thousand-core chips, neither software or hardware models suffice alone
  - Software models too slow and do not regard hardware complexity
  - Hardware RTL too labor intensive and collecting internal statistics can be difficult
- Chisel provides both models from the same codebase
- Provides hierarchical design for easy parameterization and module replacement



Fabric Internal Block Diagram



Class Hierarchy Diagram
Flow of Instantiation



Existing NoC Generators

- A few examples such as:
  - Arteris (FlexNoC)
  - ARM (CoreLink and CoreSight)
  - Orchestra (SoC generator)
  - Academic tools (Verilog for FPGA)

- Our project will be open source
  - Functionality can be extended by users
  - Both software and hardware models

Current Status

- All major architectural components are complete and tested
  - Each with individual testers
- Initial network is simple mesh network
  - Dimension ordered routing
  - Wormhole buffered flow control
- Currently debugging network functionality and adding features
  - Implementing and testing Virtual Channels
  - Implementing and testing different topologies (Flattened Butterfly, Torus, etc.)
- Developing more sophisticated testbenches to ensure robustness

Future Work

- Continue to add network features
  - Adaptive routing
  - Additional topologies
  - And so much more!
- Expand to support circuit-switched networks
  - Enables optical networking studies
- Validate against Booksim
  - Booksim has been validated against RTL implementations
- Complete AXI Interface
  - Enable integration with ARM-based cores
- Integrate into multi-core SystemC based software models
  - Include HMC, DRAM, and NVRAM memory endpoints
- Use ASIC flow to incorporate power and area calculations into software model

Questions?

- For more information, please visit:
  - http://www.opensocfabric.org