

Computer Architecture Lab at Carnegie Mellon

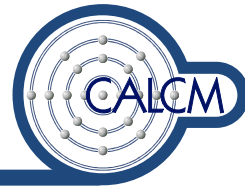
Fast Scalable FPGA-Based Network-on-Chip Simulation Models

Michael K. Papamichael

papamix@cs.cmu.edu

Cambridge, UK, July 2011

This Year's MEMOCODE Contest



- **Objective**

- Build the fastest simulator for a class of Networks-on-Chip
- Replicate cycle-by-cycle behavior of SW reference simulator

- **Simulator takes two inputs**

- 1. Network configuration**

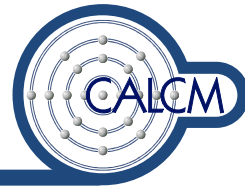
- number and input/output configuration of routers
- network topology
- number of virtual channels
- credit delay cycles

- 2. Routing info and traffic pattern**

- routing information for each network router
- number and type of packets to send

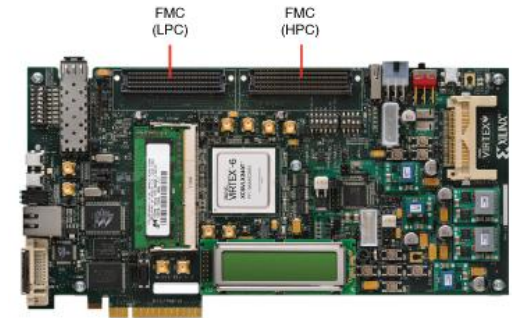
Many parameters → very large design space!

Dual-Engine NoC Simulator



- **FPGA-based solution**

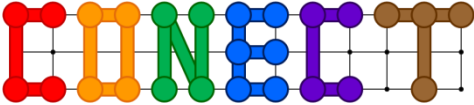
- Developed in Bluespec System Verilog
- Targets the Xilinx ML605 board
- Consists of two NoC simulation engines



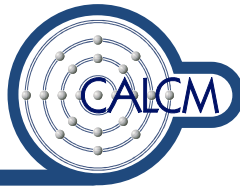
- **NoC simulation engines**


- High-performance **direct-mapped** engine
 - Supports up to moderately sized networks (~100 routers)
 - Provides **500x-1000x speedup**
- Highly scalable **virtualized** time-multiplexed engine
 - Supports all possible networks in the design space
 - Provides **5x-50x speedup**

Dual-engine approach effectively covers entire design space

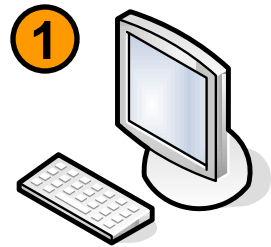
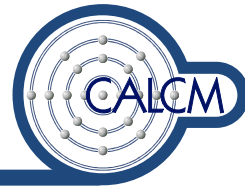
- Introduction
- **FPGA-based NoC Simulator**
 - FPGA Simulation Platform
 - Direct-Mapped NoC Simulation Engine
 - Virtualized NoC Simulation Engine
- **Results**
- **Discussion**
-  + Demo

Outline



- Introduction
- **FPGA-based NoC Simulator**
 - FPGA Simulation Platform
 - Direct-Mapped NoC Simulation Engine
 - Virtualized NoC Simulation Engine
- Results
- Discussion
-  + Demo

FPGA-based Simulation Platform

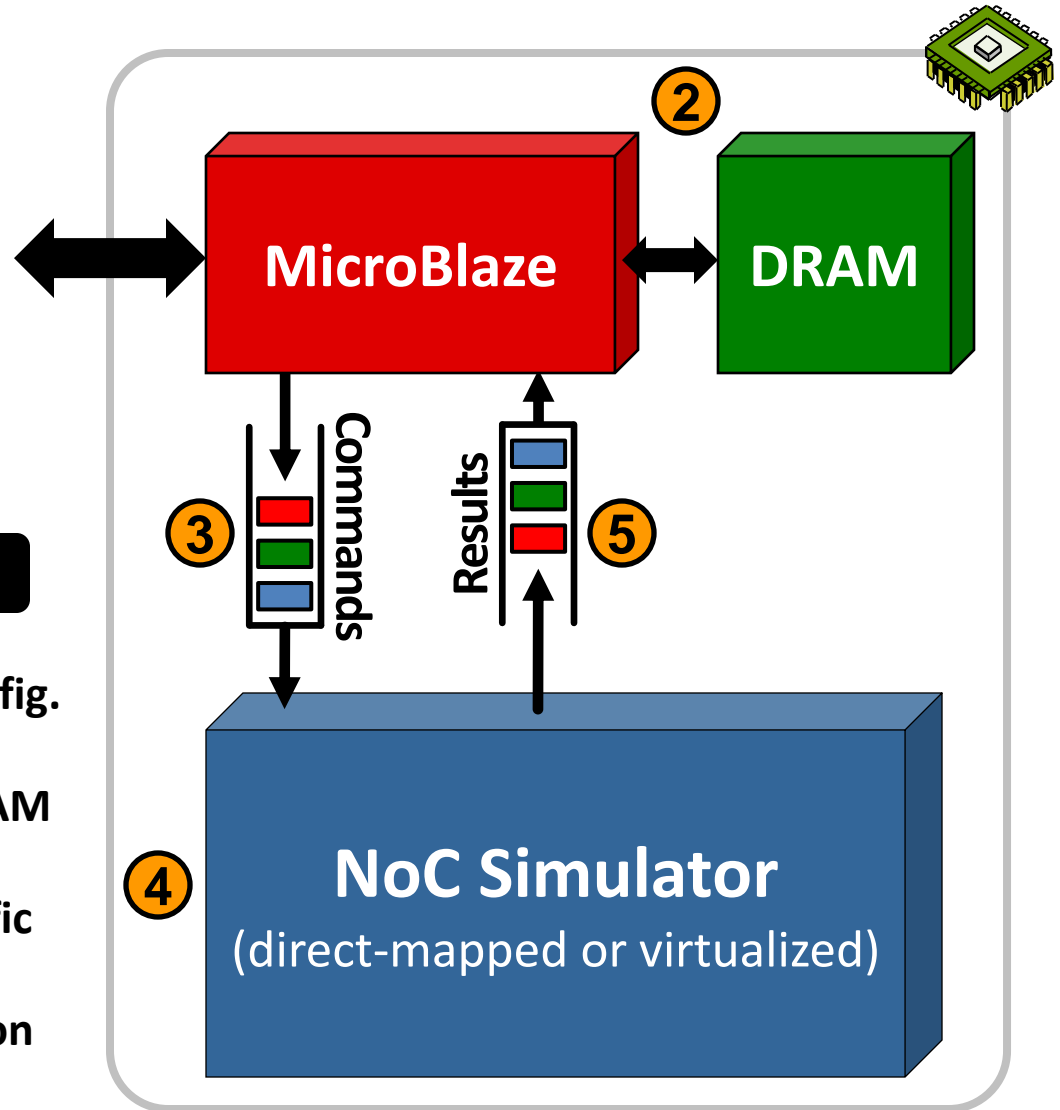


Host PC



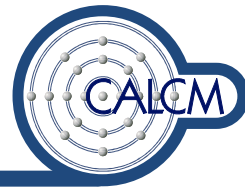
Steps to run a simulation

- ① Generate FPGA design for NoC config.
- ② Configure FPGA, MicroBlaze & DRAM
- ③ Initialize NoC sim. w/ routing+traffic
- ④ Run sim. until termination condition
- ⑤ Extract simulation results



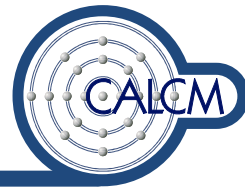
Xilinx ML605 Board

Direct-Mapped NoC Simulator

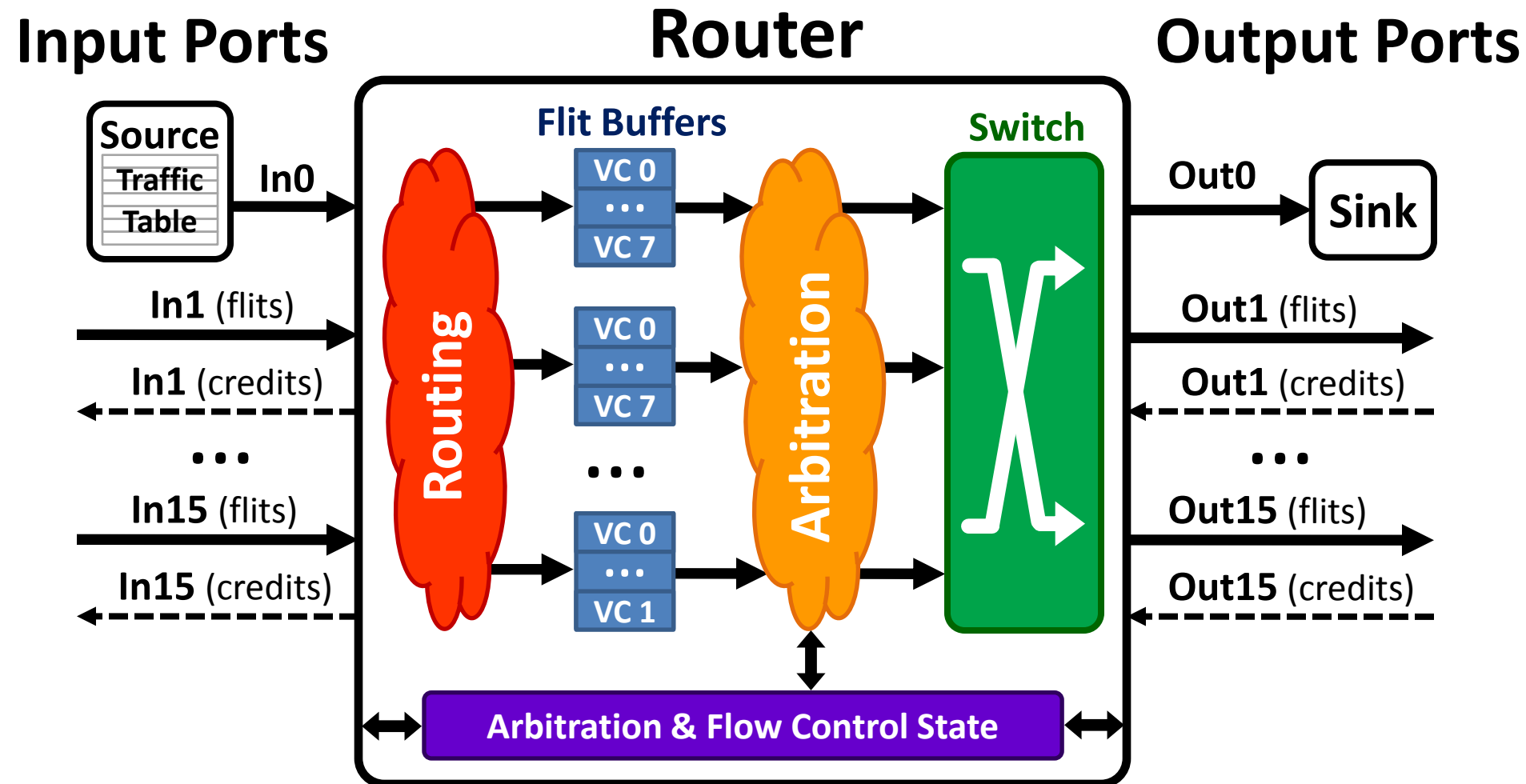


- **Why simulate... when you can prototype!**
- **Direct implementation of target NoC on the FPGA**
 - Instantiates all routers, links, traffic sources, etc
 - Collection of routers connected according to NoC configuration
 - Additional logic required to detect termination conditions
- **High performance at the cost of limited scalability**
 - Achieves **500x-1000x speedup** over software reference design
 - ML605 can fit up to ~100 routers of moderate complexity
 - 5-input/output, 4VC router occupies ~1% of LX240T FPGA
 - Need a more scalable solution for remaining design space

Direct-Mapped Router Architecture

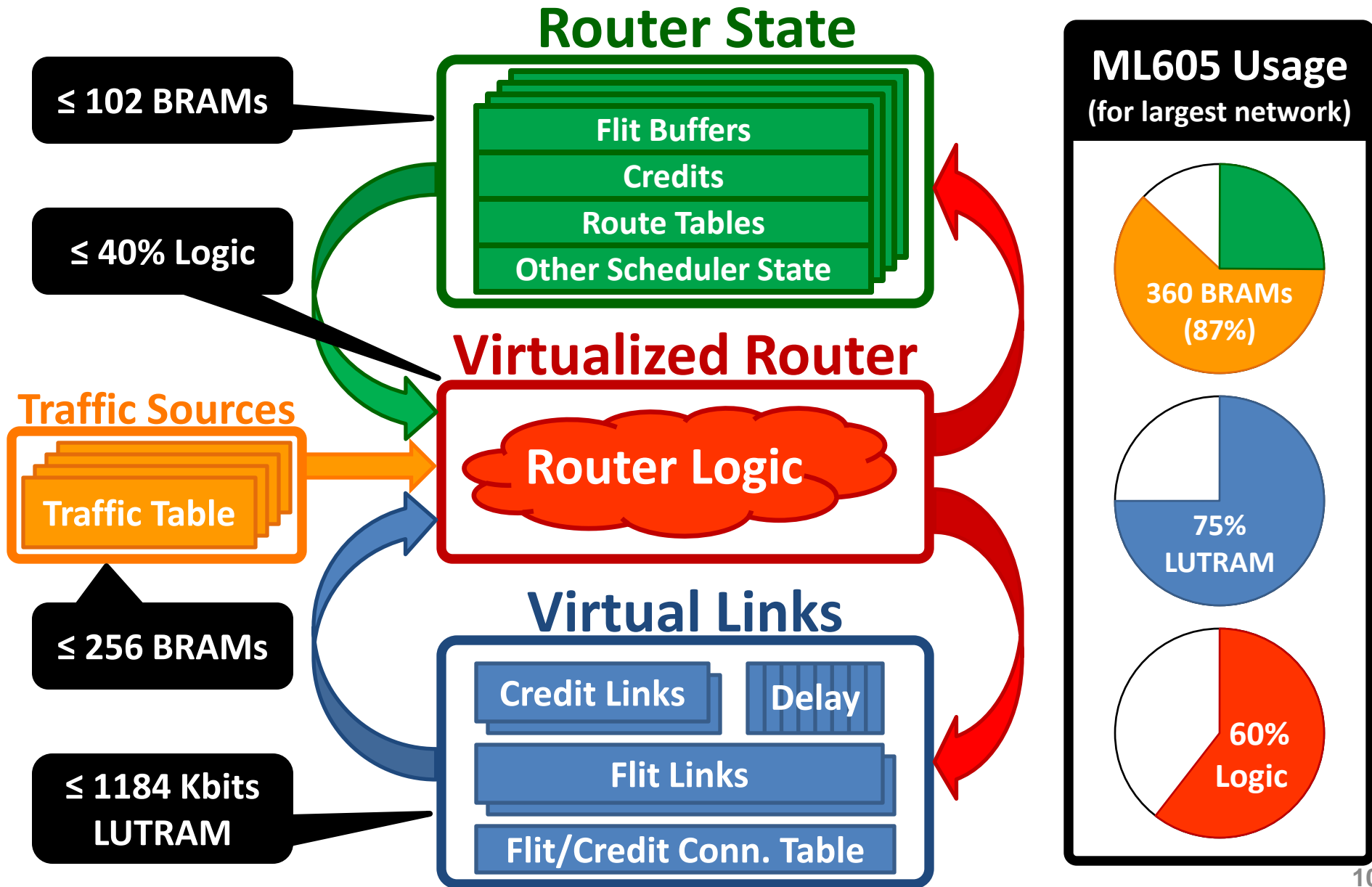
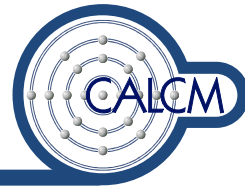


- High-level block diagram of parameterized router

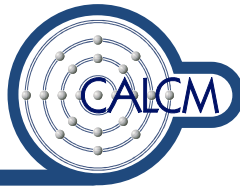



- **If resources start getting scarce, virtualize!**
- **Time-multiplexed implementation**
 - Routers are simulated one at a time in successive clock cycles
 - Router, traffic source and link state stored in on-chip memory
 - Special care to retain proper ordering of events [Pellauer '11]
 - Aggressive prefetching to maintain high simulation throughput
- **Scales to very large networks with complex routers**
 - Can cover entire design space on ML605
 - 256-router network occupies ~85% of LX240T
 - Only used when direct-mapped approach will not fit

Virtualized NoC Simulator Details

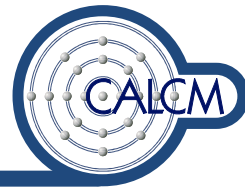


Outline

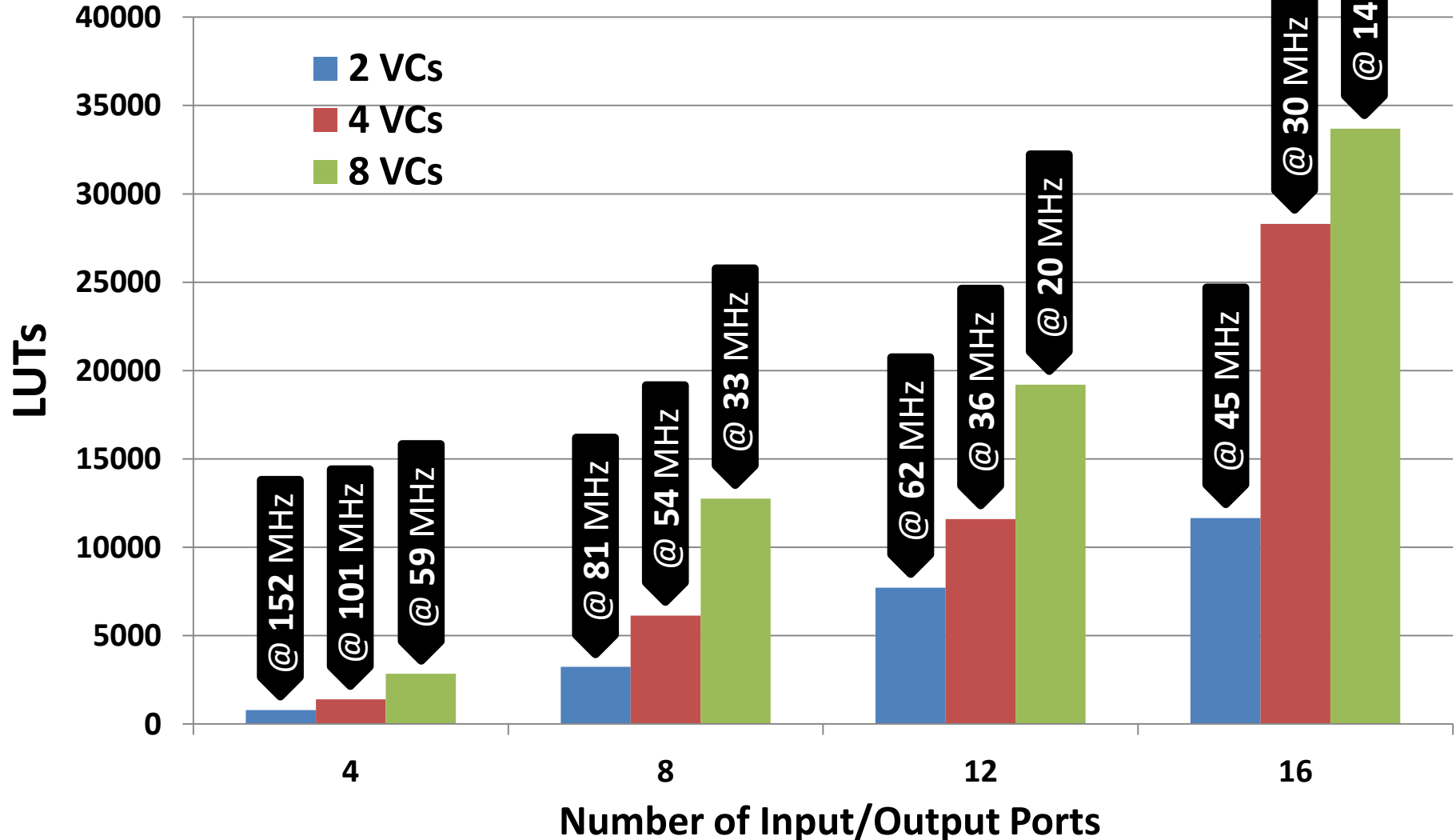


- Introduction
- FPGA-based NoC Simulator
 - FPGA Simulation Platform
 - Direct-Mapped NoC Simulation Engine
 - Virtualized NoC Simulation Engine
- **Results**
- Discussion
-  + Demo

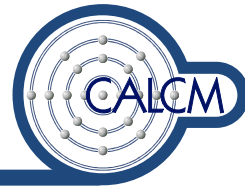
Direct-Mapped Implementation Results



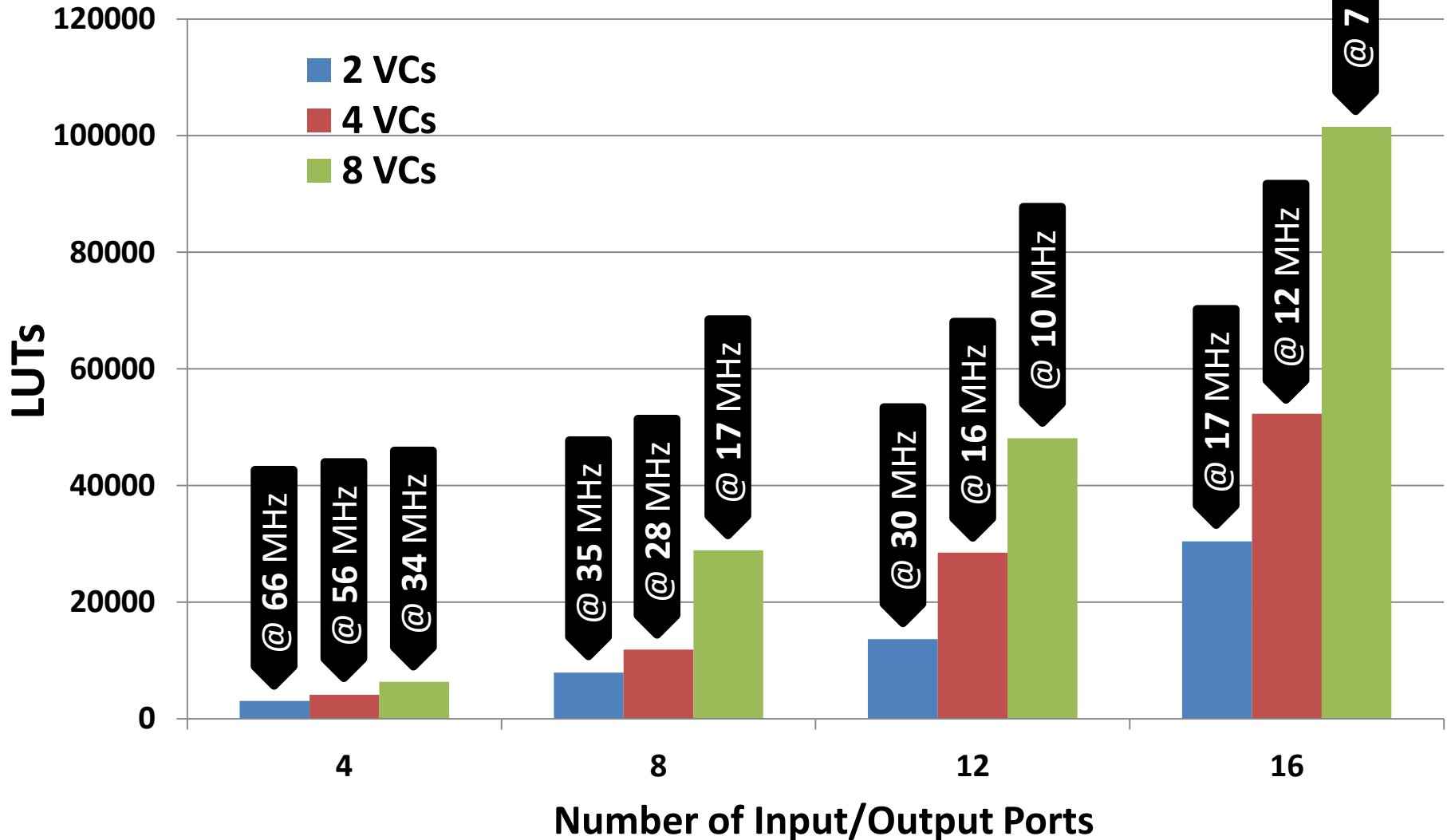
● LUT usage and frequency for single router



Virtualized Implementation Results



● LUT usage and frequency for 256-router network



Results for Contest Networks



- Five network and router configurations

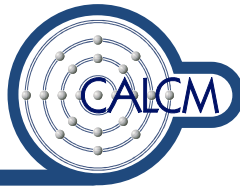
| Network Name | Routers | Ports/router | VCs | Credit Delay |
|--------------|---------|--------------|-----|--------------|
| butterfly | 112 | 3 | 8 | 1 |
| highradix | 16 | 16 | 8 | 15 |
| mesh | 253 | 5 | 4 | 3 |
| torus | 252 | 7 | 5 | 2 |
| hypercube | 256 | 9 | 1 | 1 |

All configurations lie at “edge” of design space, i.e. max-out at least one parameter

- Implementation results for contest networks

| Network | Xilinx Virtex-6 LX240T | | | Xilinx Virtex-6 LX760T | | |
|-----------|------------------------|--------|---------|------------------------|--------|---------|
| | Sim. Type | % LUTs | Speedup | Sim. Type | % LUTs | Speedup |
| butterfly | Direct Map | 86% | 1511x | Direct Map | 27% | 2330x |
| highradix | Virtualized | 63% | 6x | Direct Map | 93% | 421x |
| mesh | Virtualized | 3% | 28x | Direct Map | 96% | 4281x |
| torus | Virtualized | 8% | 786x | Virtualized | 2% | 789x |
| hypercube | Virtualized | 8% | 21x | Virtualized | 2% | 33x |

Outline



- Introduction
- FPGA-based NoC Simulator
 - FPGA Simulation Platform
 - Direct-Mapped NoC Simulation Engine
 - Virtualized NoC Simulation Engine
- Results
- Discussion
-  + Demo

- **Correctness First**

- First only focused on correctness and verify design
- Then optimize entire system and individual components

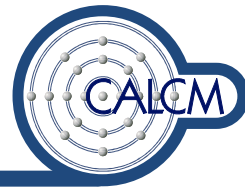
- **Parameterization and Modularity**

- Built parameterized versions of all submodules
- Utilize Bluespec's powerful parameterization mechanisms

- **Harnessing the power of Bluespec**

- **Static elaboration** for easy parameterization
- Define **clean interfaces** for modularity
- Rely on **type checking** for early bug detection
- Maintain **high-level of abstraction** to harness complexity

Bluespec code sample for virtual design



// Main simulation rule for virtualized design. Simulates network one router at a time.
rule simulate_vcycle (init_done);

// Gather incoming flits/credits and router state

1 let source_flit <- virtualSources.getFlit(cur_router); **// Gather flits from source**
in_links.in_flits = gatherIncomingFlits(source_flit, flitConnections, flitLinks); **// Gather flits from routers**
in_links.in_credits = gatherIncomingCredits(creditConnections, creditLinks); **// Gather credits**
RouterState_t router_state_before = routerState.value(); **// Get router state**

// Simulate cycle for this router and write new router state

let router_simulation_result = virtualRouter.simulateCycle(router_state_before, in_links, ...);
2 RouterState_t router_state_after = tpl_1(router_simulation_result); **// extract new router state**
RouterOutLinks_t out_links = tpl_2(router_simulation_result); **// extract flits/credits to send**
routerState.write(cur_router, router_state_after); **// Write new router state**

// Send outgoing flits/credits

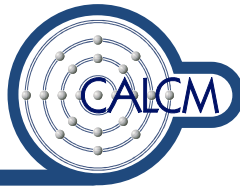
3 flitLinks.putFlits(cur_router, out_links.out_flits, tickVirtualClock); **// Send flits to other routers**
virtualSources.putCredits(cur_router, out_links.out_credits[0]); **// Send credits to traffic sources**
creditLinks.putCredits(cur_router, out_credits_to_routers, tickVirtualClock); **// Send credits**

// Advance to next router

4 cur_router <= next_router;

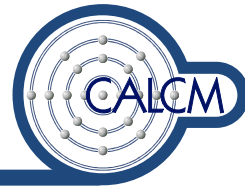
endrule

Outline



- Introduction
- FPGA-based NoC Simulator
 - FPGA Simulation Platform
 - Direct-Mapped NoC Simulation Engine
 - Virtualized NoC Simulation Engine
- Results
- Discussion
-  + Demo

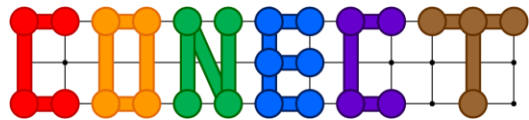
Taking it to the next level - CONECT



- **Direct-mapped approach implements an actual NoC**
 - Parameterized
 - FPGA-friendly
 - Supports arbitrary network topologies



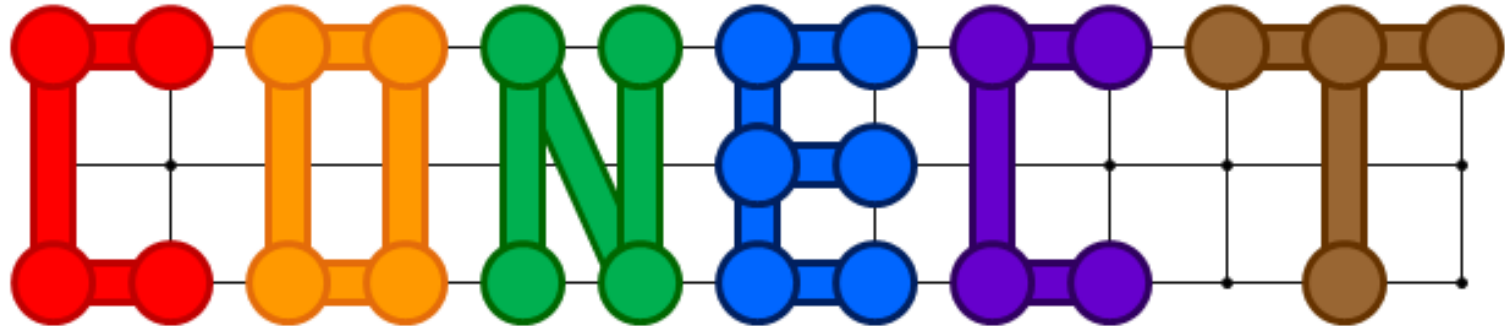
Build on top of this to create a useful tool!



: Configurable Network Creation Tool

- **Highly parameterized Network-on-Chip generation tool**
 - # routers, topology, routing, allocation, # VCs, buffer width/depth, etc
- **Back end developed in Bluespec System Verilog**
- **Python command-line interface and web interface (demo)**

CONNECT Web Interface Demo



Thanks!



Questions?