FIST

Fast Interconnect Simulation Techniques

Michael K. Papamichael, Eric S. Chung,
James C. Hoe, Babak Falsafi, Ken Mai

papamix@cs.cmu.edu, {echung, jhoe, babak, kenmai}@ece.cmu.edu

CALCM

PROTOFLEX

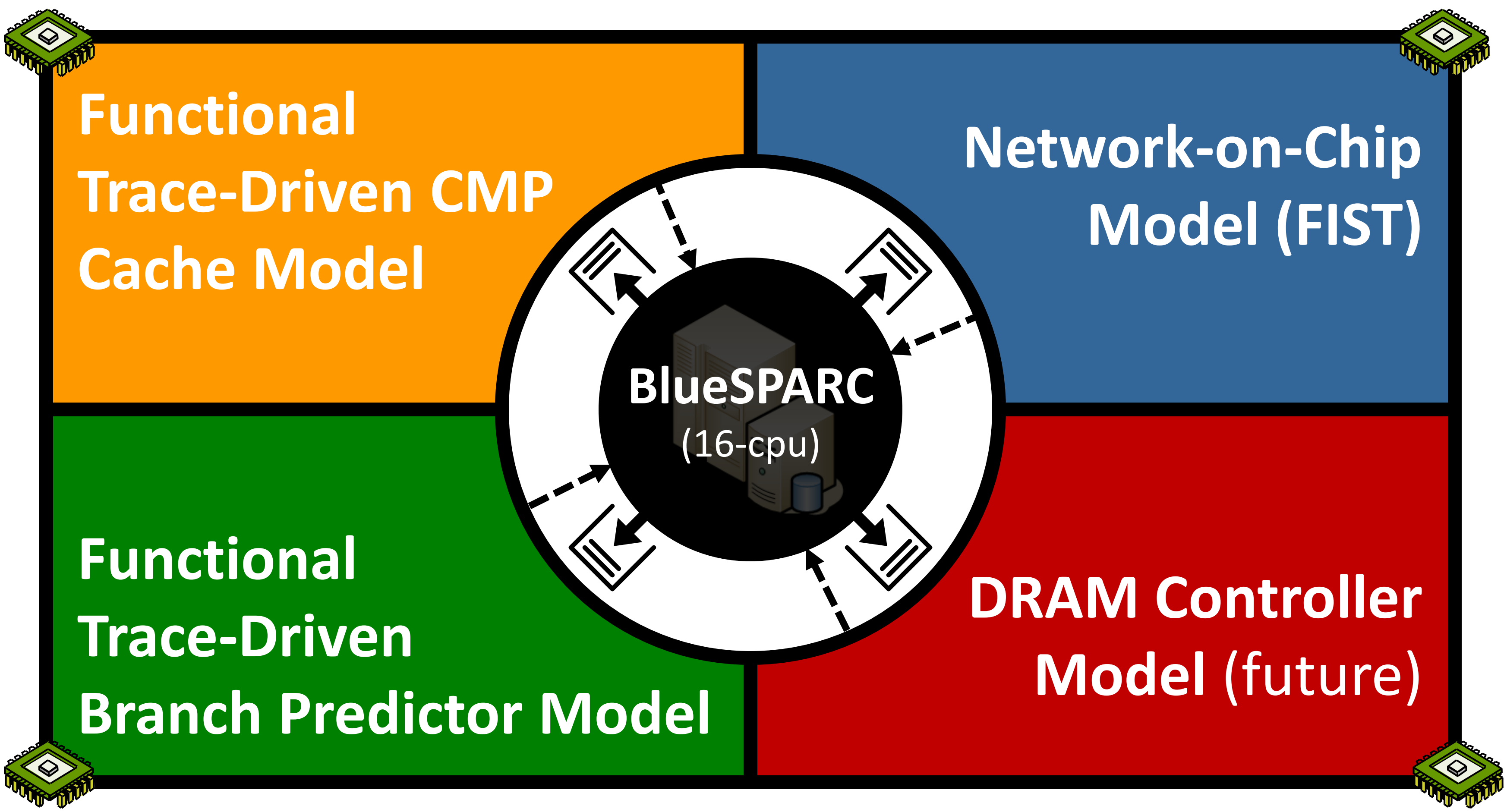
Computer Architecture Lab at
Carnegie Mellon

Our work in this area has been supported in part by NSF, IBM, Intel, and Xilinx.

29-Jan-2010

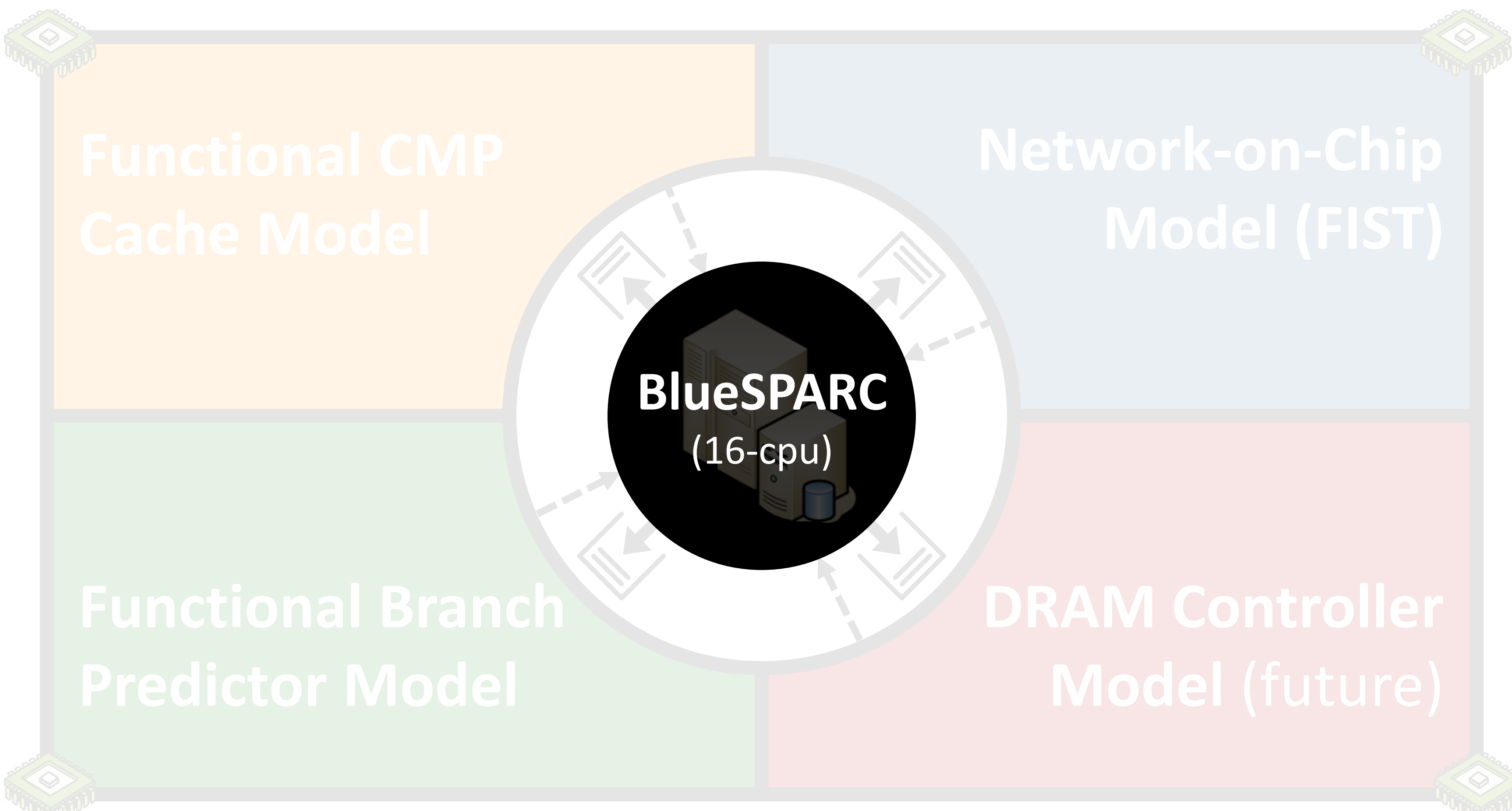
Fast Instrumentation using FPGAs

- Accelerate sampling-based timing simulations
- Uncore modeling
- Add timing through backpressure



Outline

- **BlueSPARC (1-slide overview)**
- Network Modeling (FIST)

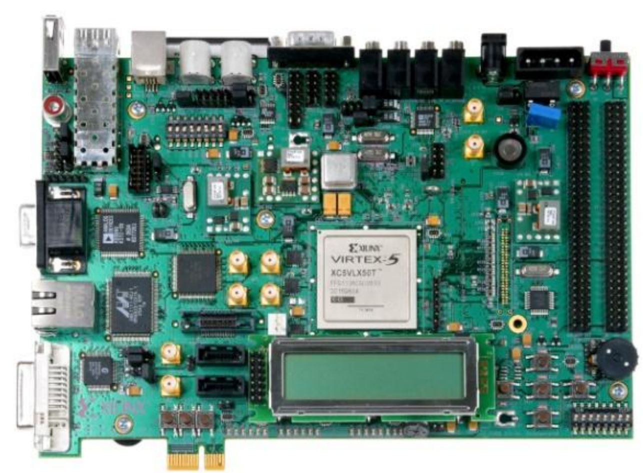


BlueSPARC Simulator

- **Open-source HW-based Full-system Functional Simulator**
  - Models 16-cpu UltraSPARC III server
  - Can boot OS, run commercial apps
  - Publicly released under GNU GPL v2
  - Visit **www.ece.cmu.edu/~protoflex**

- **Interesting BlueSPARC Facts**
  - Implemented on BEE2 and XUPv5 FPGA platforms

BEE2

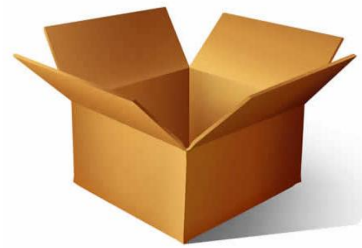


XUPv5

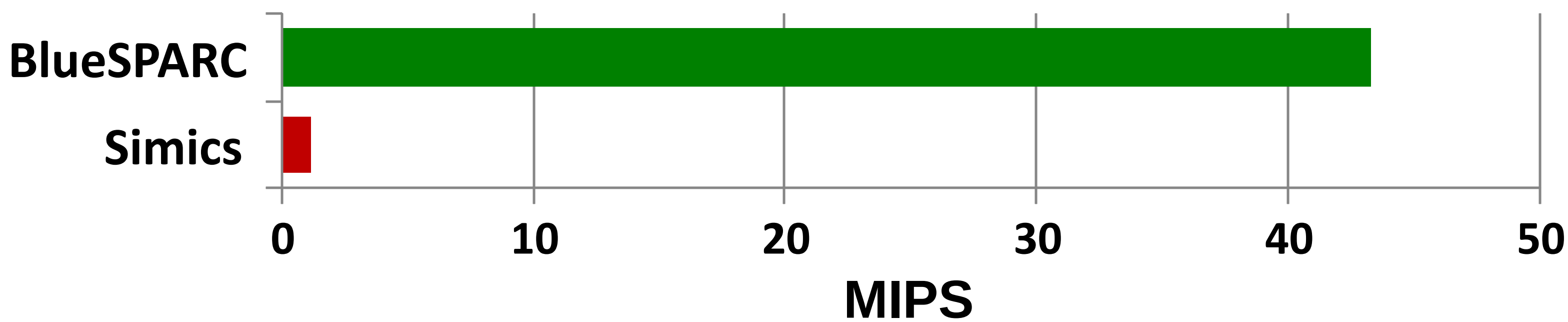
OVERVIEW

BlueSPARC Simulator

- **Open-source HW-based Full-system Functional Simulator**
  – Models 16-cpu UltraSPARC III server
  – Can boot OS, run commercial apps
  – Publicly released under GNU GPL v2
  – Visit **www.ece.cmu.edu/~protoflex**



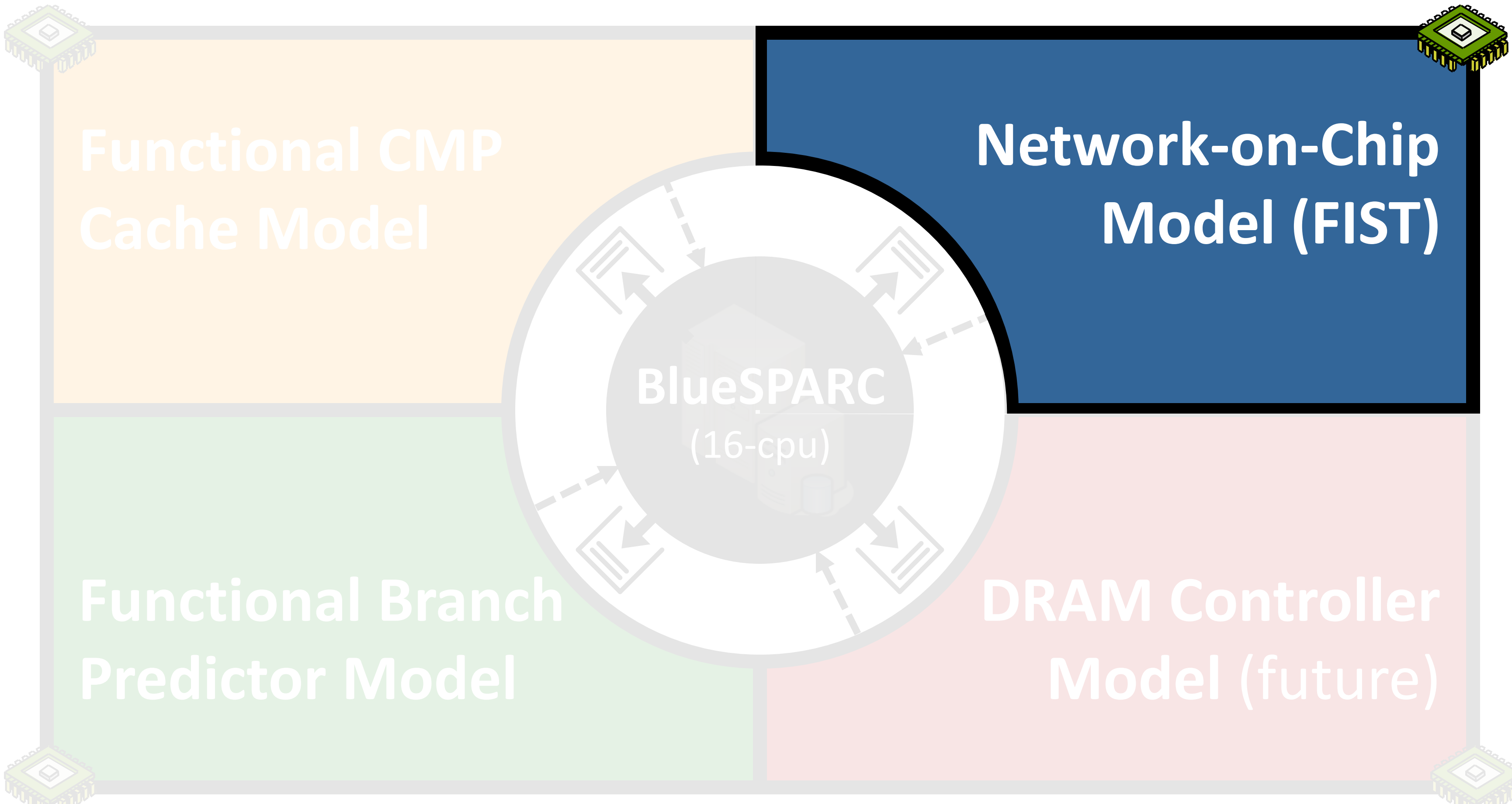
- **Interesting BlueSPARC Facts**
  – Implemented on BEE2 and XUPv5 FPGA platforms
  – Written in Bluespec HDL
  – Enables fast instrumentation (**37x** speedup over SW on avg.)



OVERVIEW

Outline

- BlueSPARC (1-slide overview)
- **Network Modeling (FIST)**



Network Simulation Goals

- **Avoid building actual Network-on-Chip in FPGA**
  - Buffered NoC w/ multiple VCs very **complex**
  - **Limited size** of simulated network

Datapath Width	3x3	4x4	5x5	6x6	7x7	8x8
32-bit	43%	76%	120%	172%	234%	306%
64-bit	63%	112%	175%	253%	344%	449%
128-bit	101%	180%	282%	405%	552%	721%
256-bit	177%	315%	493%	709%	965%	1261%

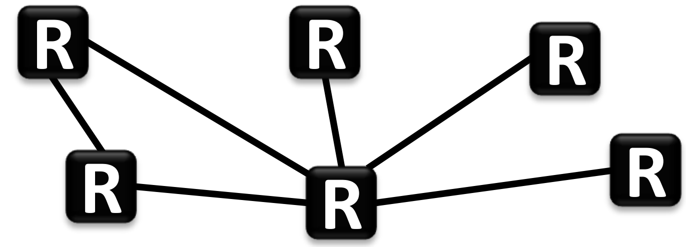
Mesh NoC LUT Usage on Xilinx LX110T*

- **Stay within acceptable error margin**
- **Trade-off complexity vs. fidelity**
- **Simulate variety of network topologies**
- **Be fast to keep up with BlueSPARC**

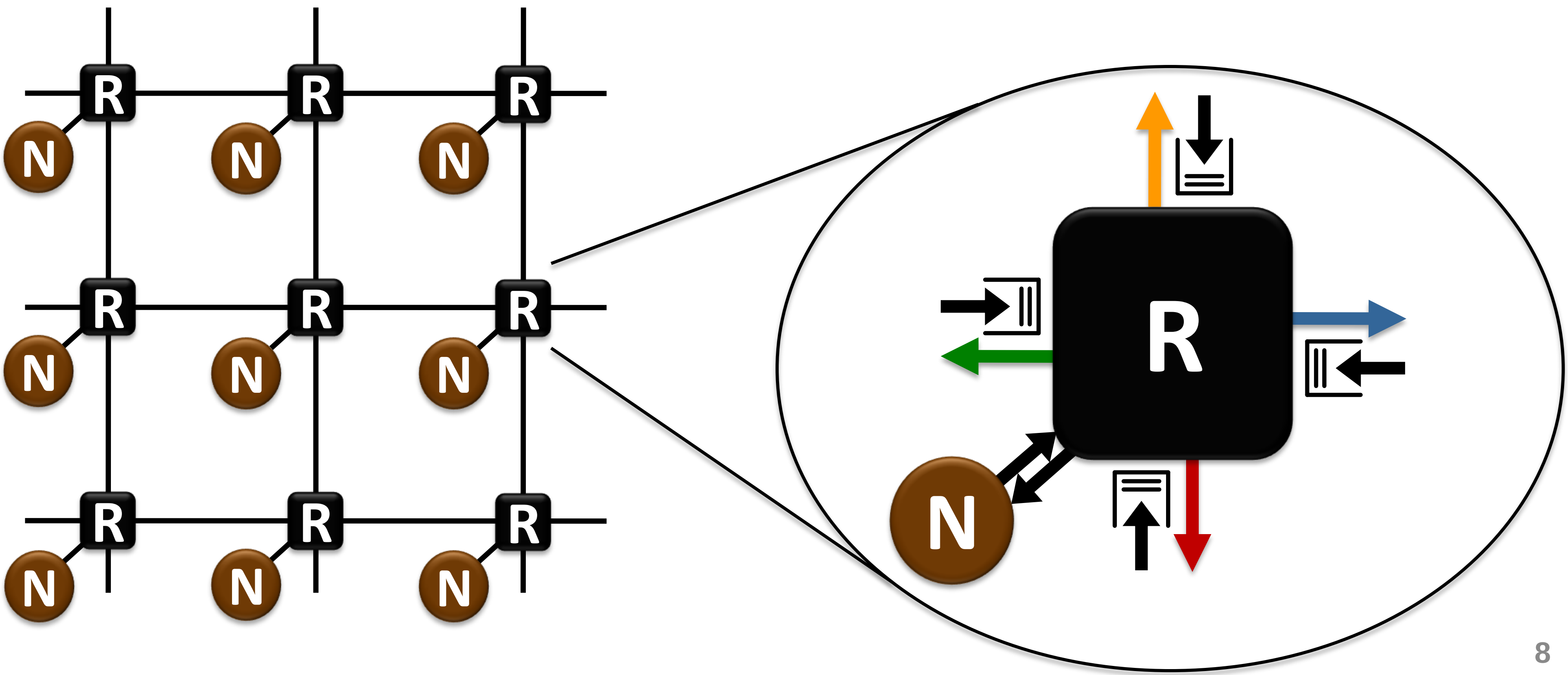
*Results obtained through synthesis of open source NoC Router RTL found at http://nocs.stanford.edu/router.html

What is a NoC?

- **Abstract View of NoC**
  - Set of links connected by routers
  - Buffers may exist at inputs/outputs

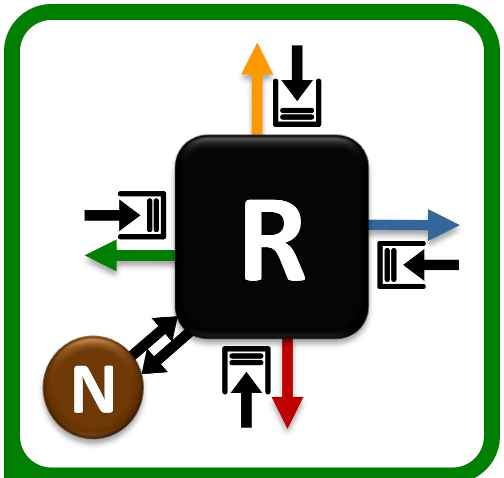


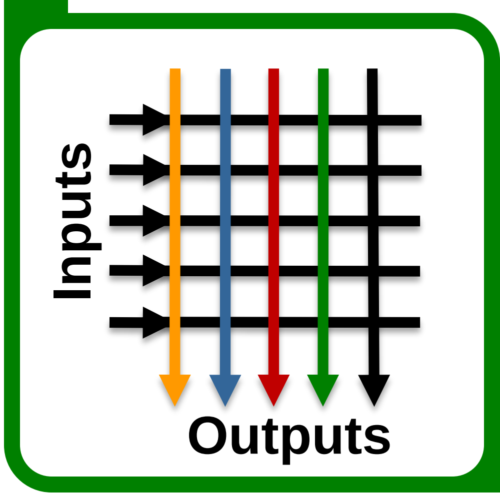
- **Mesh Example** (Input-Queued)

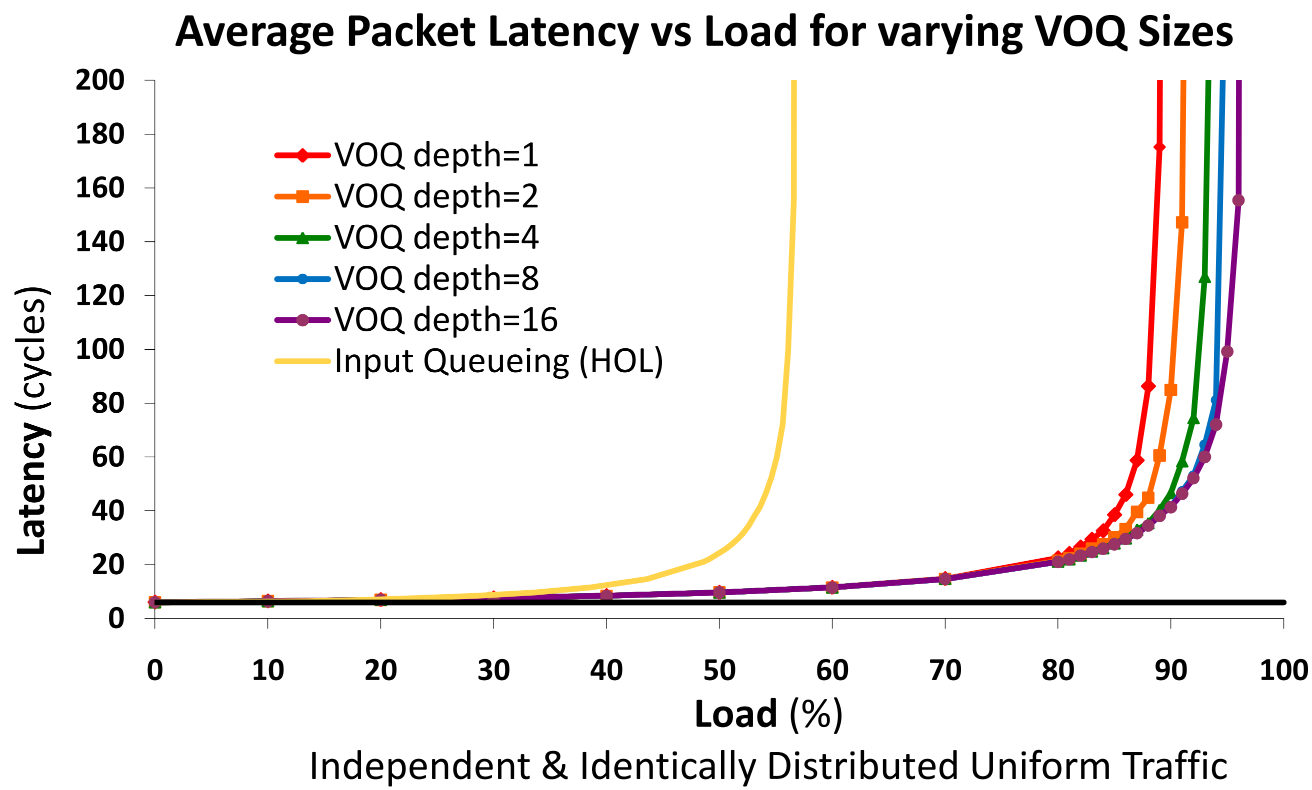


Key Idea

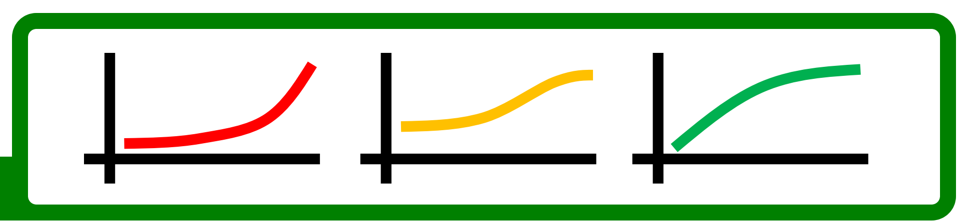
- Router equiv. to crossbar w/ **characteristic delay-load curves**
- Known curves for given **configuration** & **traffic pattern**





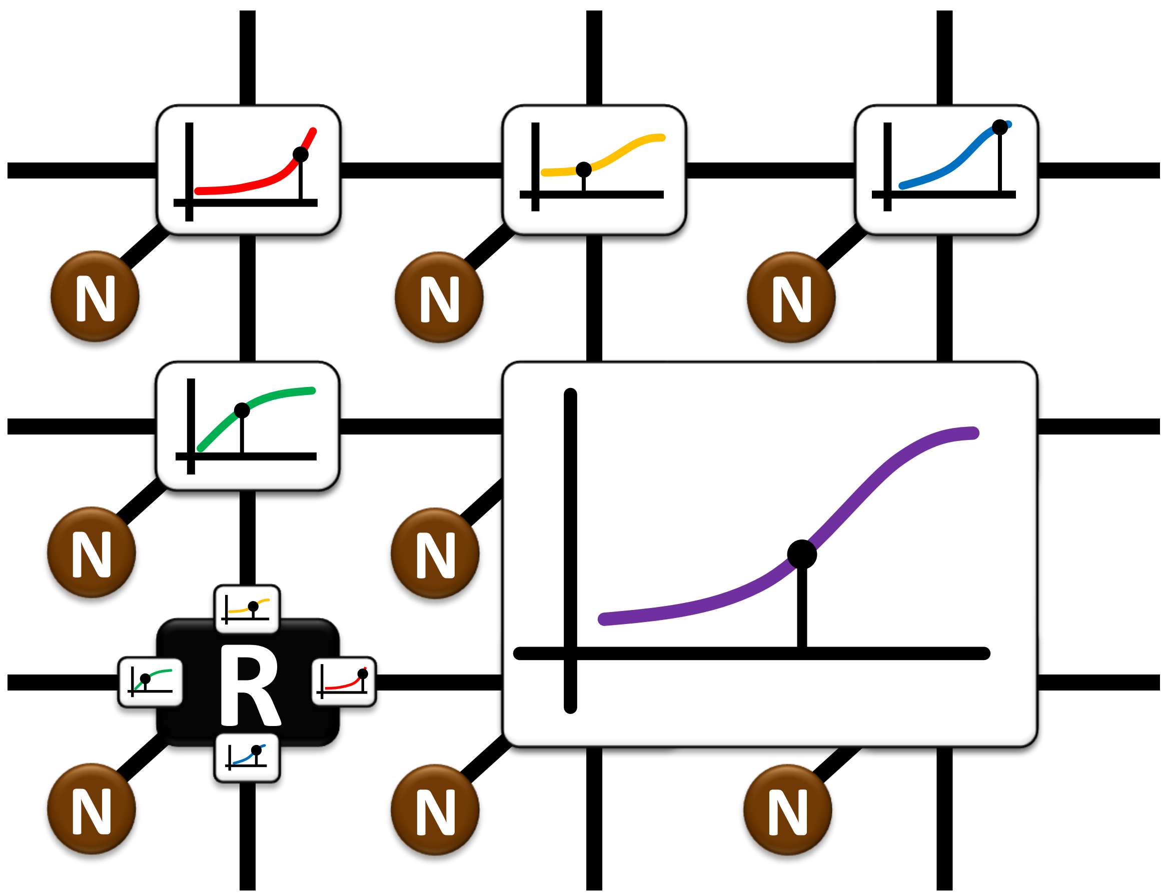


Delay-Load Curve Example for Buffered Crossbar



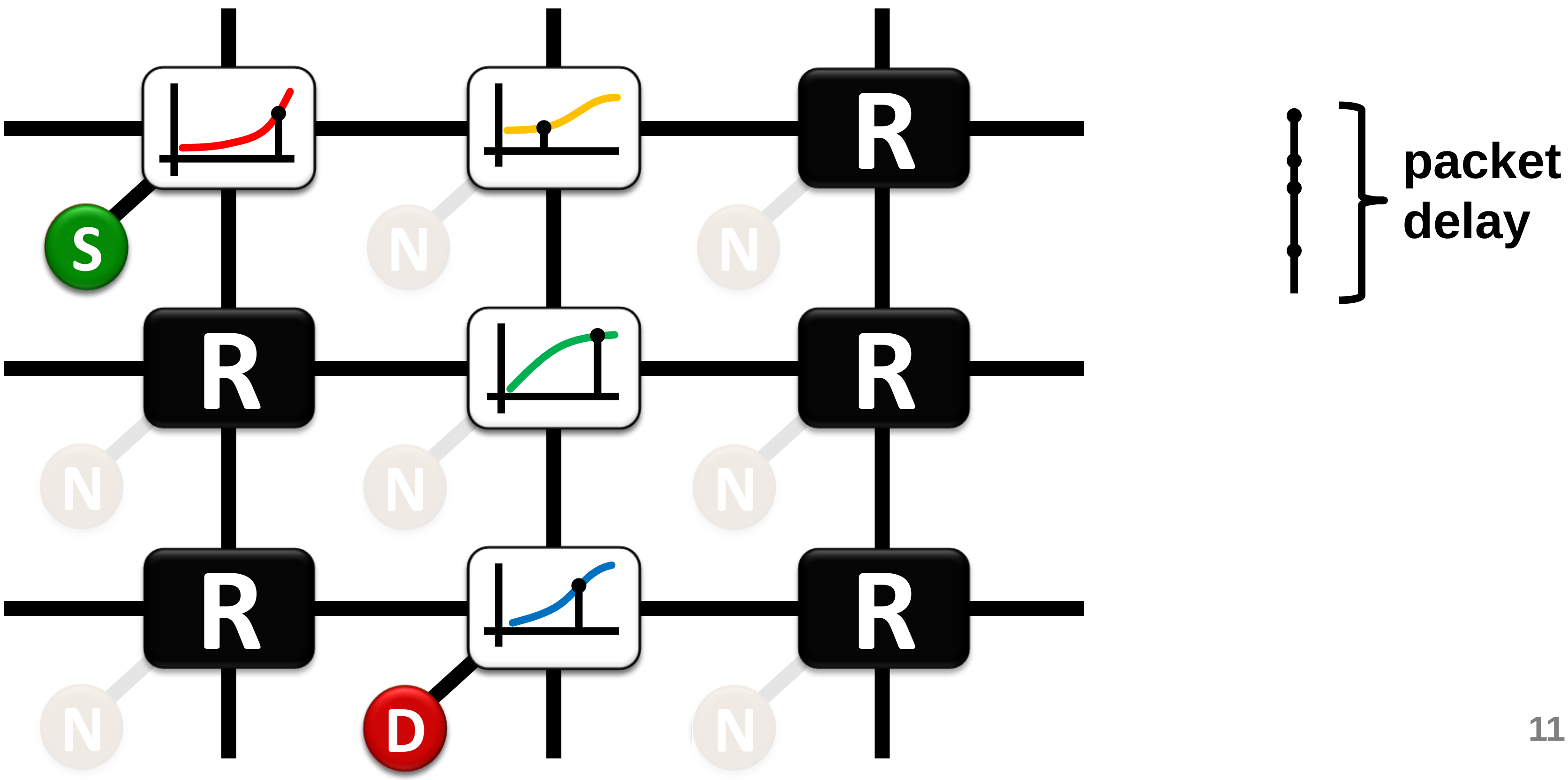
FIST Approach

- Treat each hop as a delay vs. load curve
  - Trade-off between model complexity and fidelity
- Keep track of load at each node



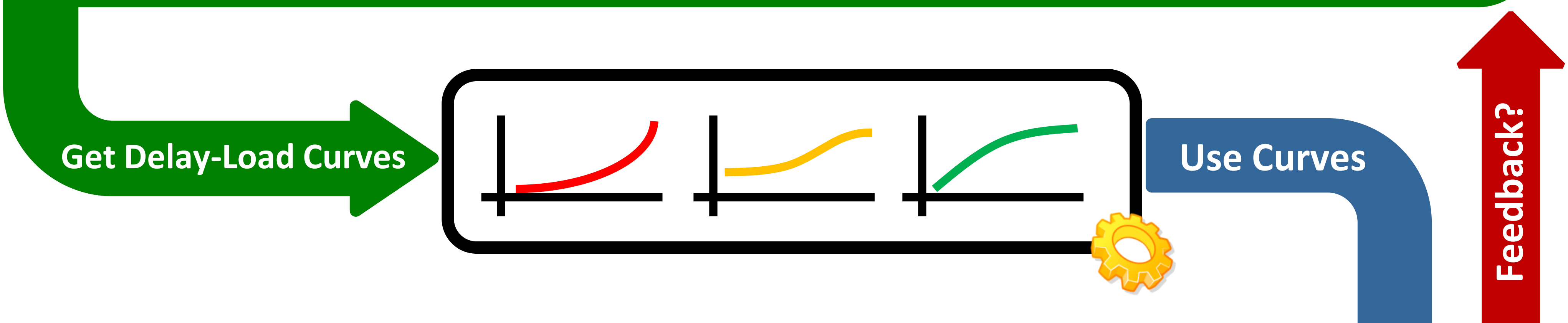
FIST in Action

- Route packet from source to destination
  - **Deterministic routing** (e.g. dimensional routing): **easy**
  - **Non-deterministic routing:** (e.g. adaptive routing): **harder**
- Add up the delays for each traversed router
  - **Index delay-load curves using current load at each router**



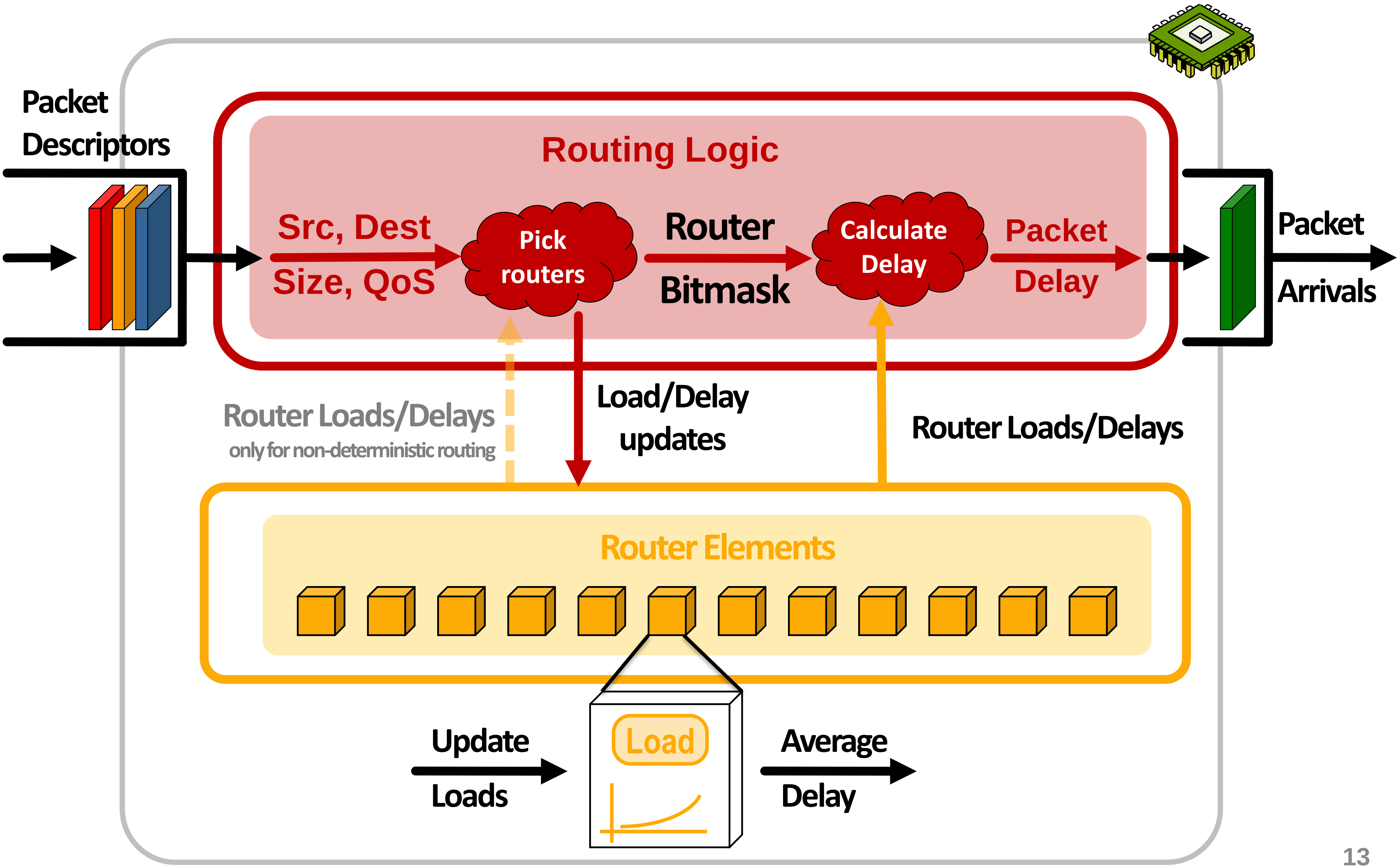
Network Sim. Usage Scenarios

- **Synthetic/Independent Network Studies** (e.g. BookSim)
  - Characterize network (e.g. load-latency, saturation throughput)
  - Study asymptotic behavior network
  - Use synthetic traffic patterns & “absolute virtual time”



- **Full-system studies w/ networks** (e.g. GEMS+Garnet)
  - Model network within a broader simulated system
  - Assign delay to each packet traversing the network
  - Traffic gen. by real workloads (often self-throttling)

FPGA-based FIST Architecture Draft



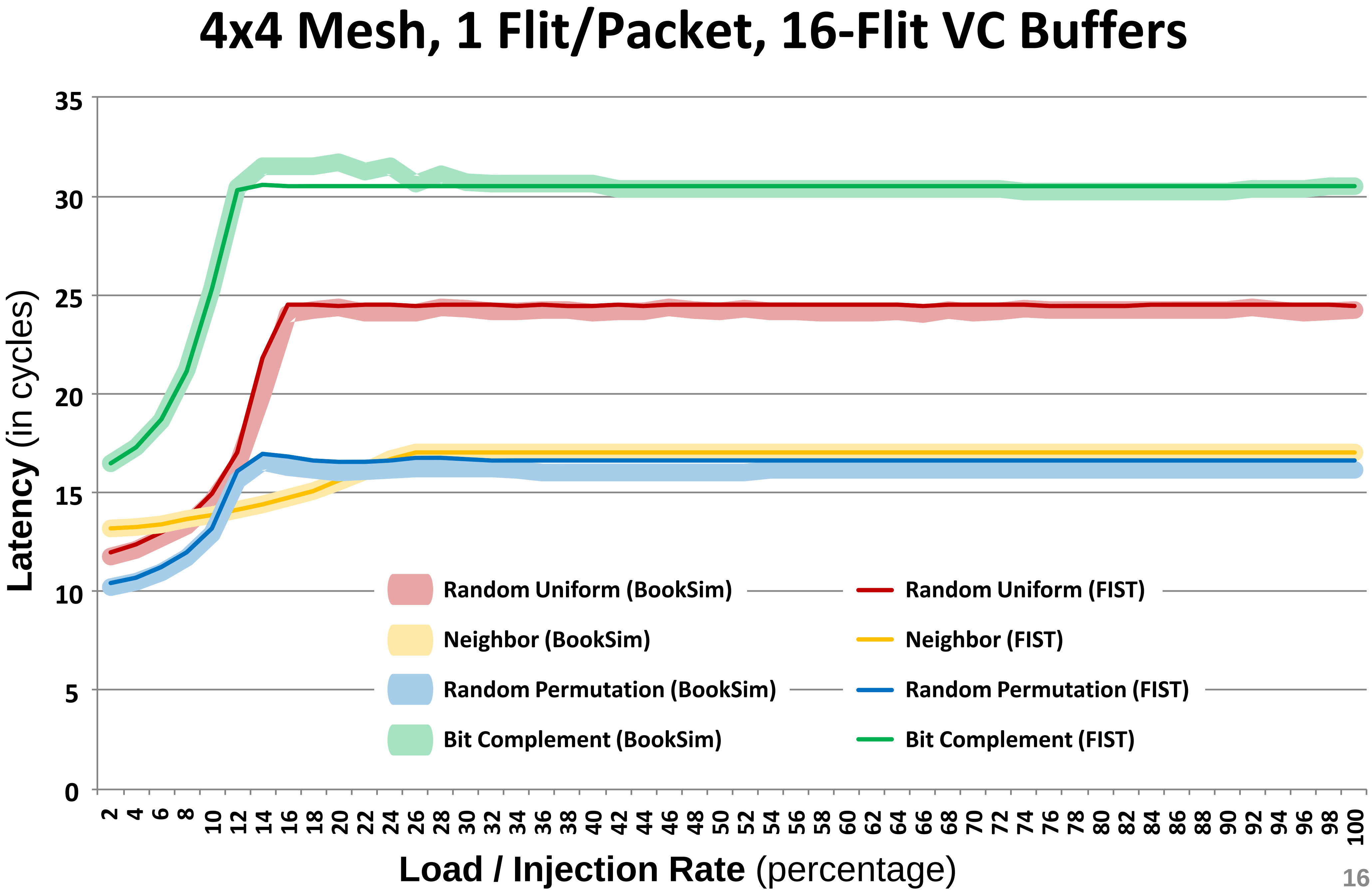
Implementation Issues

- **How do you obtain delay-load router curves?**
  - Precompute using cycle-accurate SW-based simulator
- **How many curves?**
  - One curve per router
  - One per traffic pattern
  - One per router input and/or output
- **Are curves dynamically updated? How often?**
  - Profile traffic and use same curves throughout entire workload
  - Detect traffic pattern and pick out of existing set of curves
  - Run SW-based simulator on the side and update periodically
- **How do you keep track of load at each router?**
  - Average injection rate at each router over window of N cycles
  - Amount of buffering in the network determines window size (N)

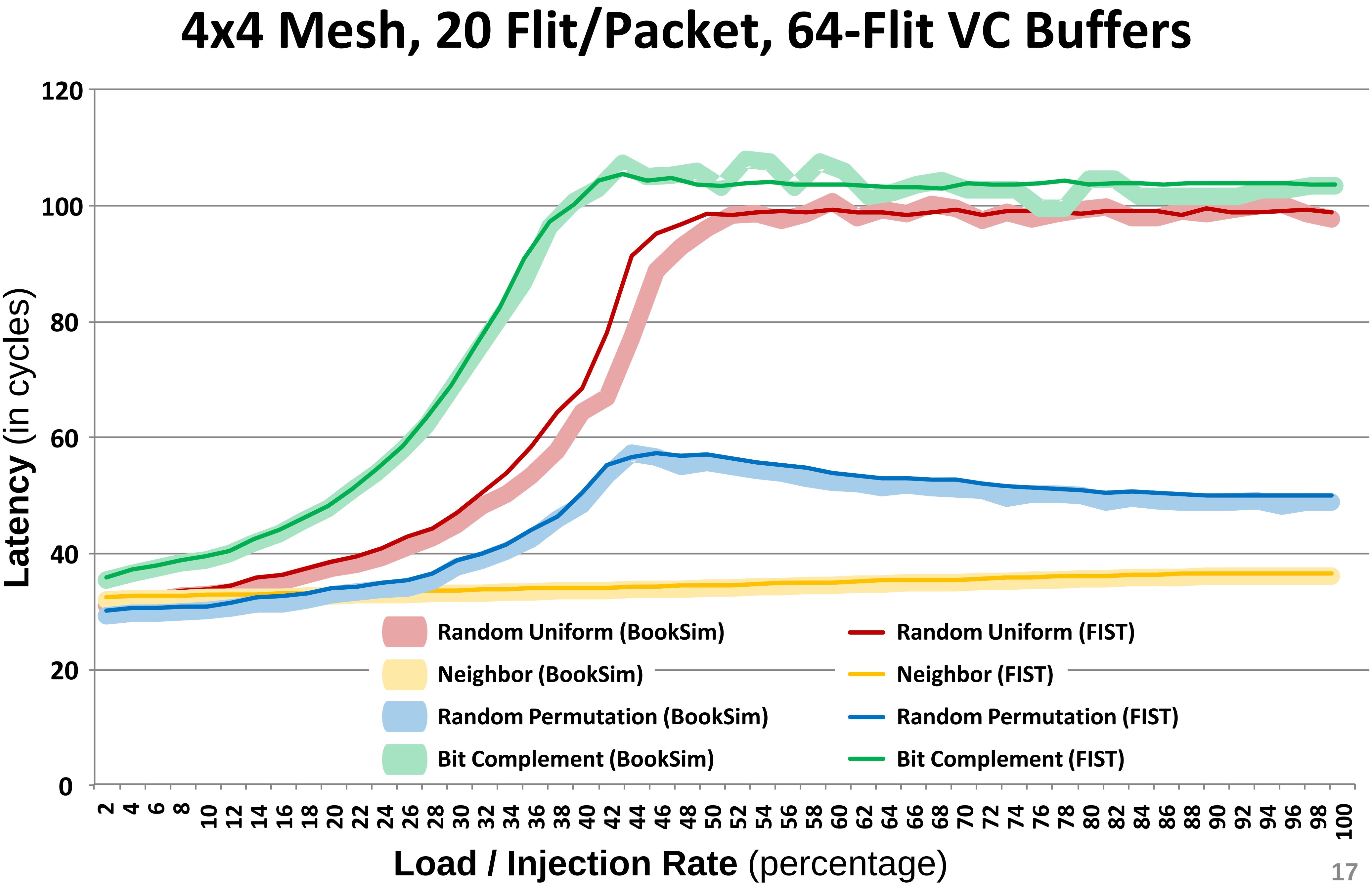
Software-based Prototype

- **C++ implementation of FIST scheme (~1000 L)**
  - Used for experimentation and validation
  - Can be easily ported to HW
- **Reference Network Simulator: BookSim**
  - Looked at variety of Mesh- and Torus-based networks
  - **Traffic Patterns**: Uniform Random, Transpose, Bit Complement, Bit Reverse, Shuffle, Tornado, Neighbor, Random Permutation
- **One Curve per Router**
  - 50 load-latency pairs per router
- **Usage / Experimental Methodology**
  1. Run BookSim experiment and get delay-load curves per router
  2. Plug curves into FIST & run experiment w same traffic
  3. Compare results & fine tune

Accuracy Results I

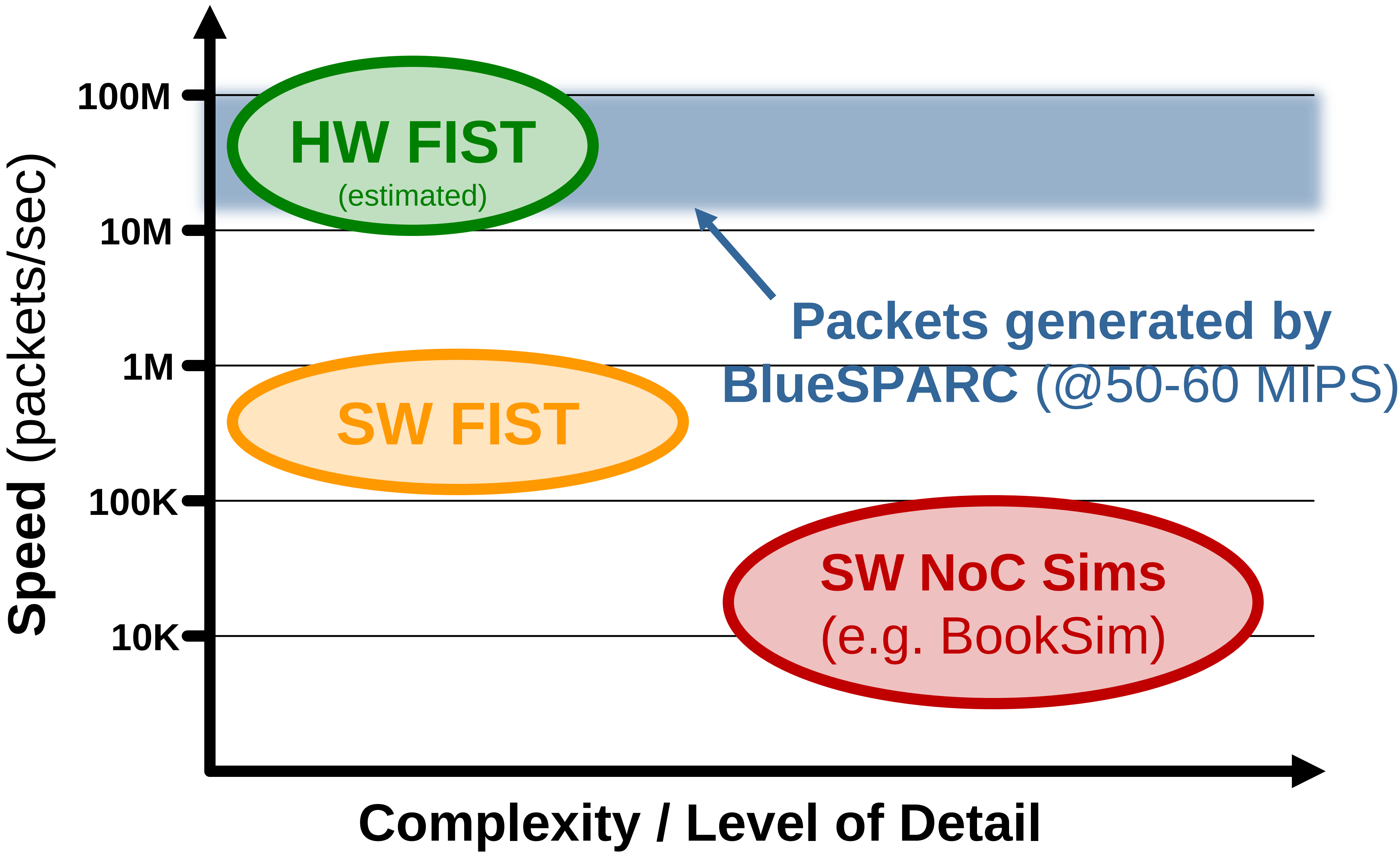


Accuracy Results II



Performance

- **Performance/Complexity Trade-Off**



Research Challenges

- **Traffic pattern not always known or well defined**
  - Maybe OK to assume some similar traffic pattern
  - Run SW sim on the side & dynamically update curves ("sampling")
  - Detect traffic and pick out of existing curves (machine learning?)
- **Finding path in non-deterministic routing**
  - Abstract away actual path. Only care about # hops.
- **Short-term transient effects hard to capture**
  - Cannot model fine-grain packet interactions
- **How good is good enough?**
  - For full-system simulations previous work settles with <10% error.
  - What is the acceptable error margin for the network component?

Conclusions & Future Directions

Conclusions

- **The devil is in the details**
  - Easy to get the trend right
  - Hard to get the details right

- **Majority of existing NoCs keep it simple**
  - E.g. no fancy routing, most stick with dimension ordering (DOR)
  - NoC shares power budget/die area with other components

Future Work & Directions

- **Deal with unknown/dynamic traffic patterns**
- **Non-deterministic routing** (e.g. adaptive routing)
- **Augment instrum. components with timing info**

Thanks! Any questions?

papamix@cs.cmu.edu

http://www.ece.cmu.edu/~protoflex