Scalable Bayesian Network Discovery with Reconfigurable Hardware

Christopher W. Fletcher
Greg Gibeling
Dan Burke
John Wawrzynek

UC Berkeley

Narges B. Asadi
Eric Glass
Wing Wong
Teresa Meng
Garry Nolan

Stanford

Outline

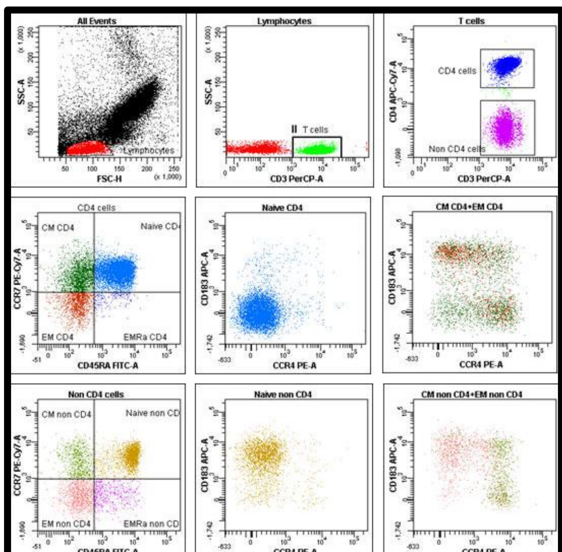
- Biological Perspective
  - The Motivation: Learning the structure of cell signaling networks
  - The Algorithm: Computational complexity & MCMC
  - Algorithmic approach
- Reconfigurable Computing Perspective
  - Hardware approach
  - FPGA implementation
  - Design scalability
- Results
- Future Work
- Conclusion and Summary

Cell Signaling Networks

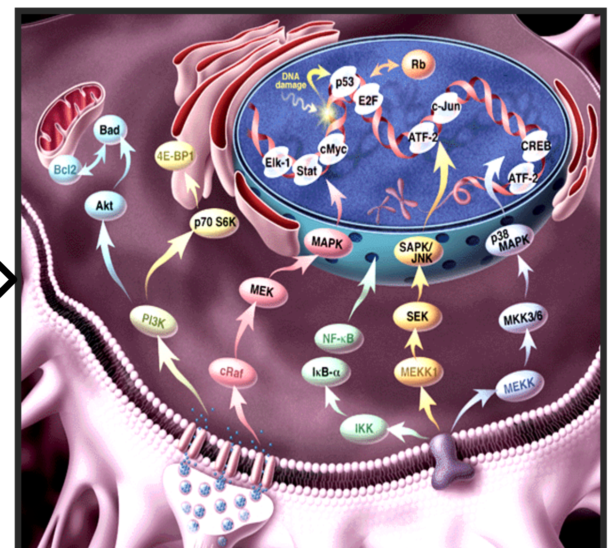
Goal: Given flow cytometry data, learn the structure of cell signaling networks

- Flow Cytometry
  - Data in the form of “raw” quantitative observations
  - Measurement of proteins & other components inside cells

- Cell Signaling Networks
  - Structures that model protein signaling pathways
  - Modeling perturbations to a network can help uncover the cause of human disease



This talk

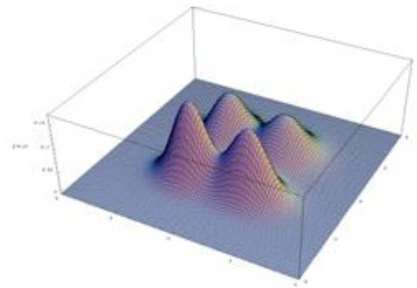


Problem: Kernel is NP-Hard

Goal: Determine which network best explains the data

- Algorithm Bottlenecks
  - Search space grows super-exponentially with the graph's node count
  - Multiple local optima, encoding best-solutions, may exist

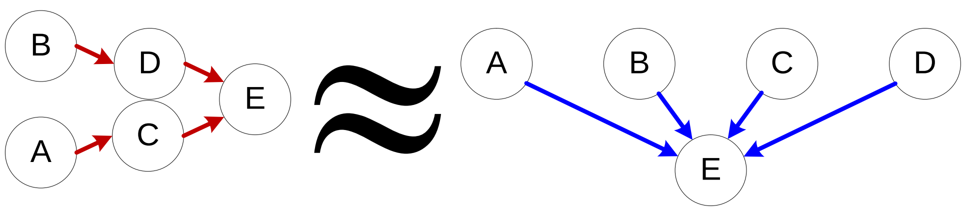
Nodes	Graphs
4	453
5	29281
10	4.7×10^{17}
20	2.34×10^{72}

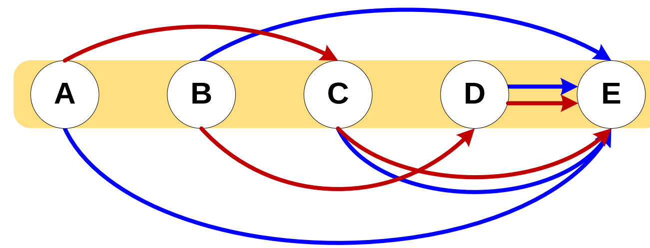


- Alternative Approach: "MCMC Sampling"
  - Markov Chain Monte Carlo
  - Slower than search methods
  - More reliable and less prone to get stuck in local optima (higher "QoR")

Algorithmic Approach

- Graph vs. Order Space
  - The “order space” is much smaller than the “graph space”
  - Swapping nodes in the order space results in a larger move





- Computational Strategy
  (1) Calculate local scores per parent set
  (2) “Order Sampler”: Determine the likely orders (**algorithm kernel**)
  (3) “Graph Sampler”: Extract graphs from probable orders
- Idea: Implement the Order Sampler in Hardware
  - Minimize the time it takes to score an **order**
  - Reduce the computational complexity to score an order

Hardware Approach

- Scoring an order is "embarrassingly" parallel
  - Divide computation by node

Score for Order $\prec$ Local score of node V_i

Comparisons, Multiplications, and Subtractions

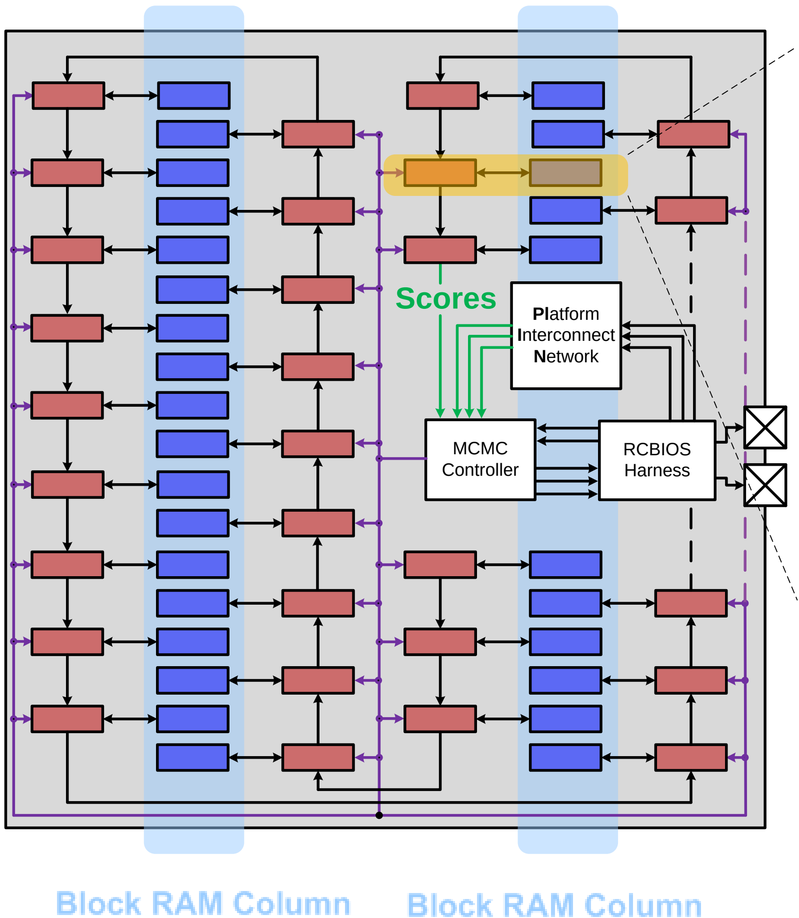
$$\boxed{score(\prec)} = \sum_{i=1}^{N} \boxed{score(V_i)}$$

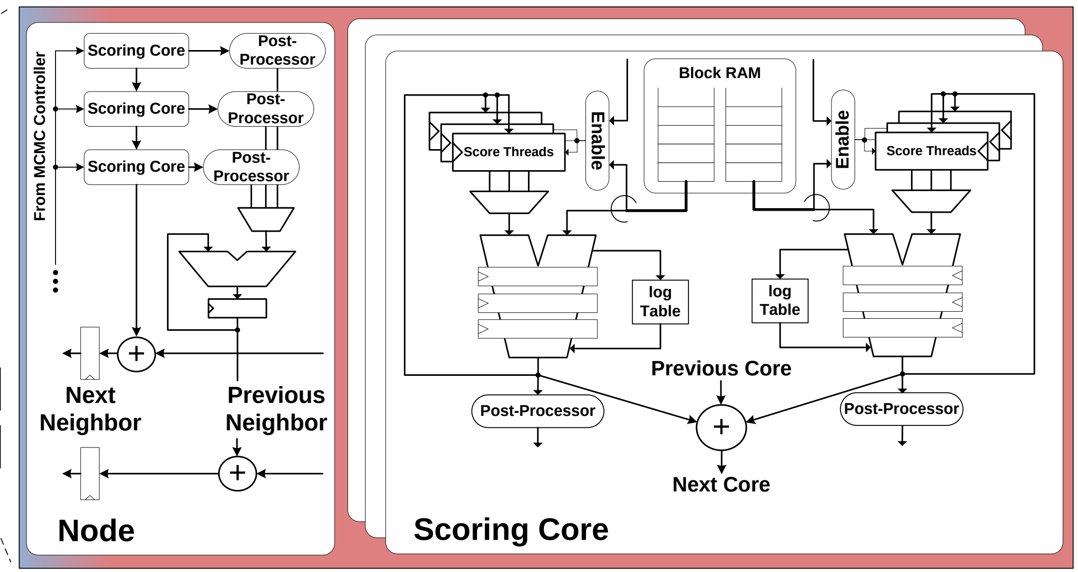
$$= \underbrace{score(V_1) + score(V_2) + \ldots + \overbrace{score(V_N)}^{\text{Comparisons, Multiplications, and Subtractions}}}_{\text{Built as separate parallel units in hardware}}$$

Built as separate parallel units in hardware

  - Partition parent sets into block RAMs
- Perform (3) – the "Graph Sampler" step – alongside the Order Sampler
- Map computations to **`log`** space
  - Bulk of computations are on probabilities (small values)
  - Multiplications → Additions

FPGA Implementation





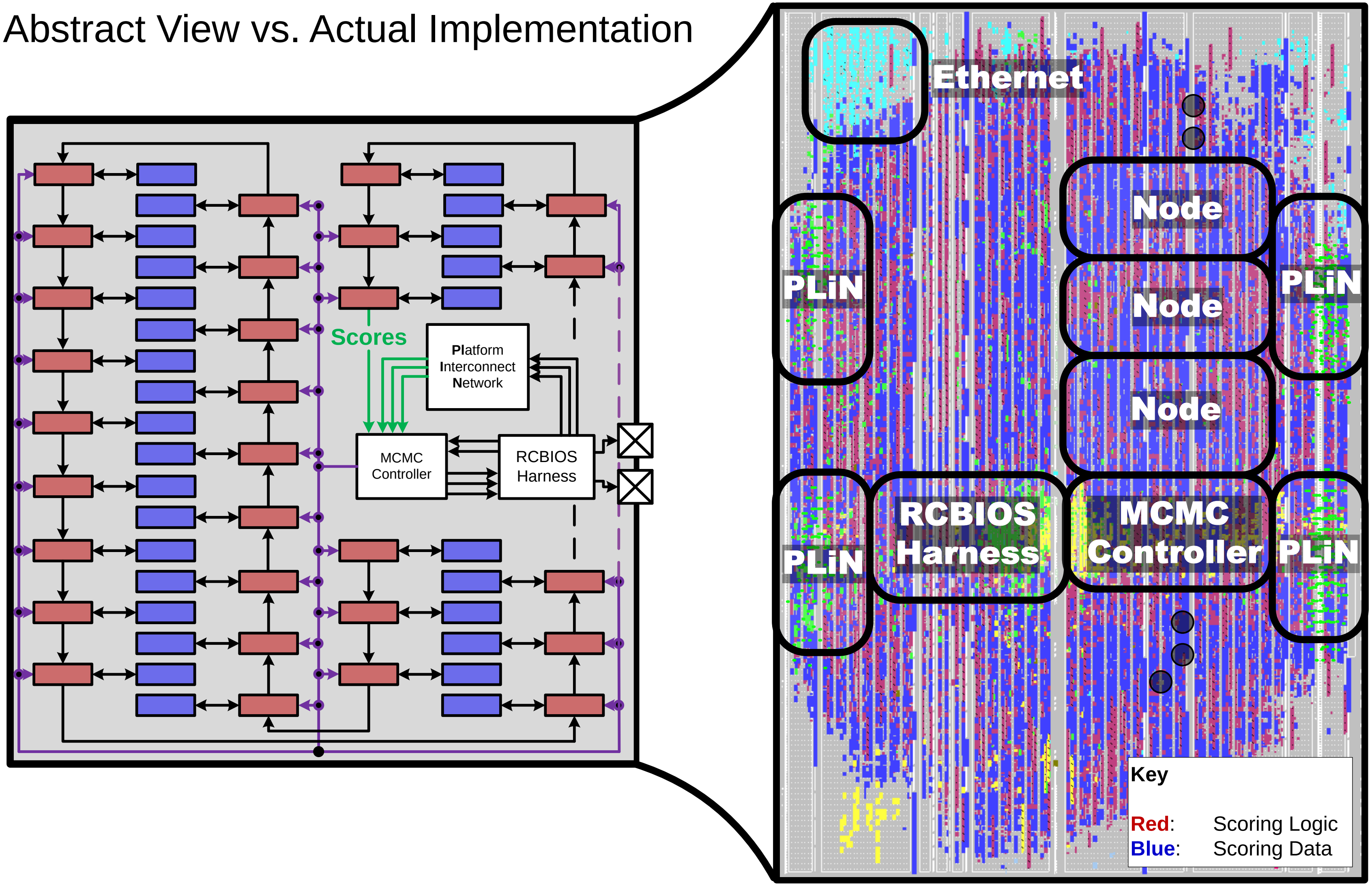
Xilinx Virtex-5
LX155T FPGA

29 node system
3 scoring cores per node

Key
- Scoring Data, Scoring Logic: One node
- Point where Proposed Score is produced
- Proposed Order

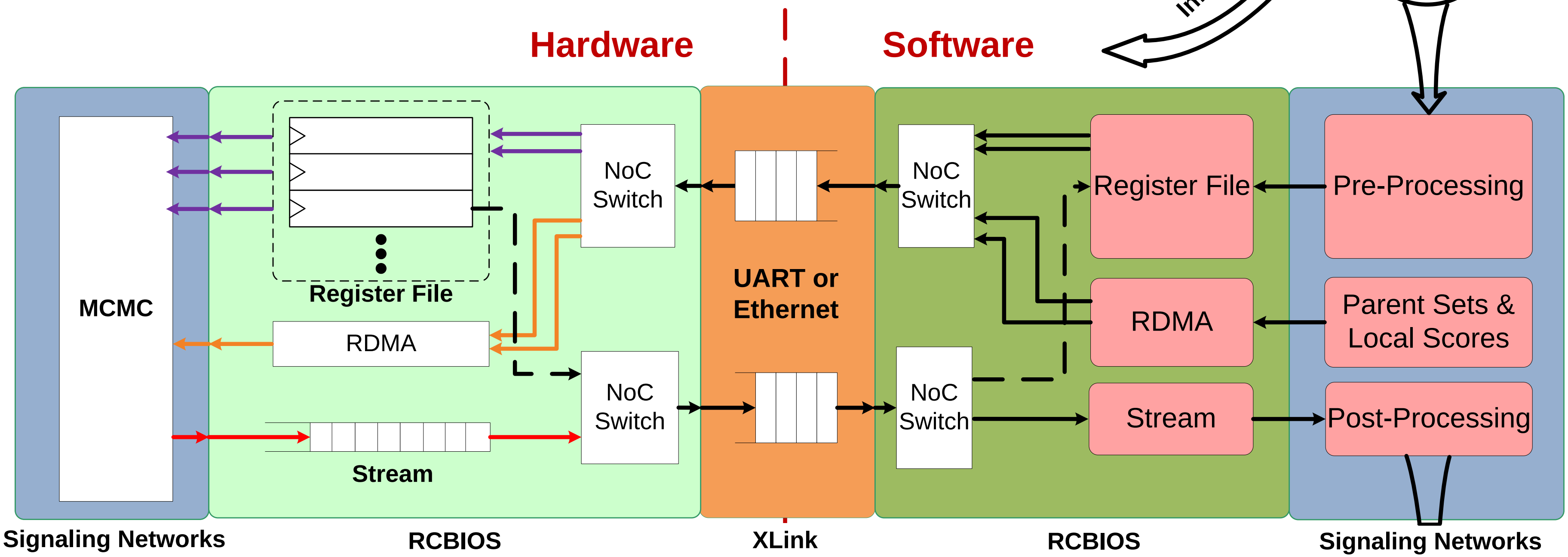
FPGA Floorplanner (LX155T)

Abstract View vs. Actual Implementation



FPGA Infrastructure

- RCBIOS – Part of GateLib
  - Scalable FPGA←→Software communication
  - Composed of Verilog, Java, and Apache ANT



  - NoC (as opposed to bus) based
  - XLink: physical link independent (UART/Ethernet/JTAG/VPI)

Design Scalability

“MCMC Mesh”

Idea: Split larger problems across multiple FPGAs

* While maintaining base design

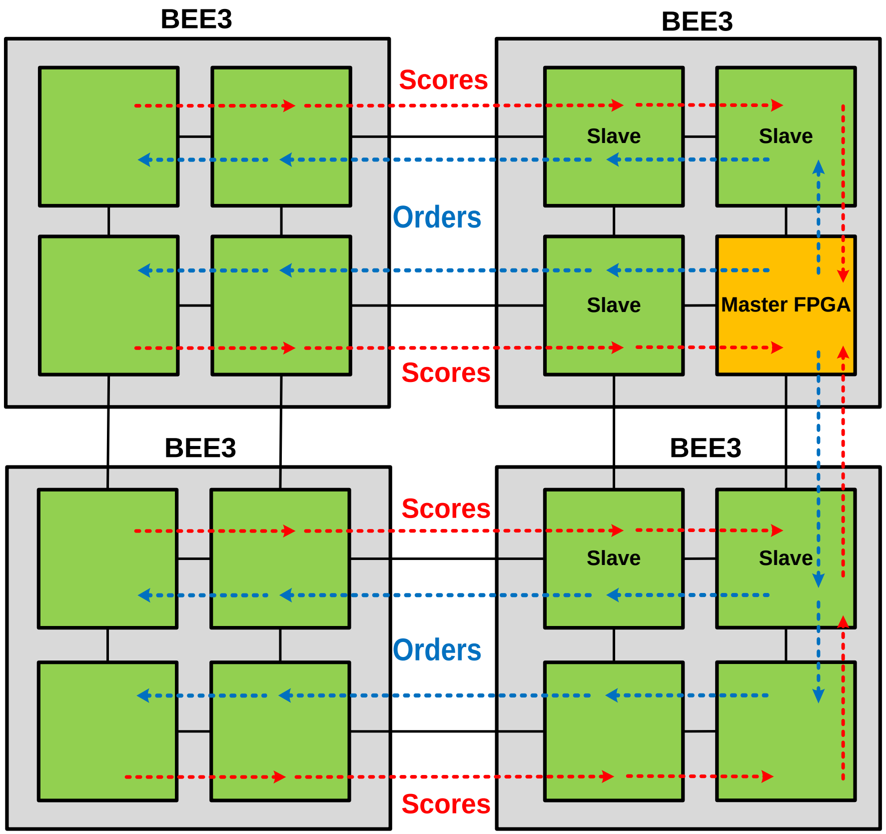
- Additional Infrastructure

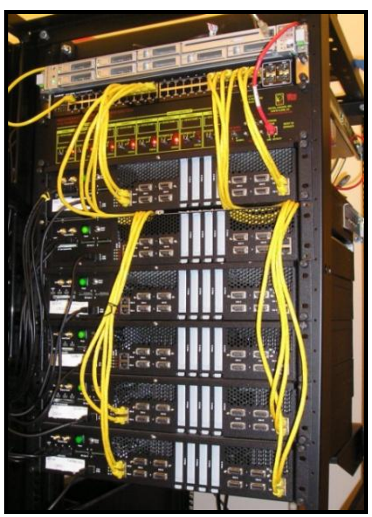
  (1) Inter-chip ring connections

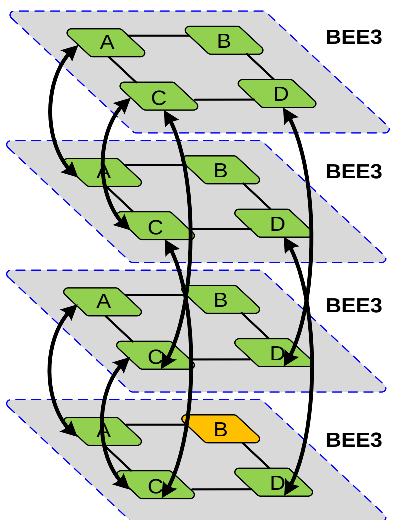
  (2) Inter-board Aurora high-speed links

  (3) **Pl**atform **I**nterconnect **N**etwork (PLiN)

  built on (1) and (2)







Results

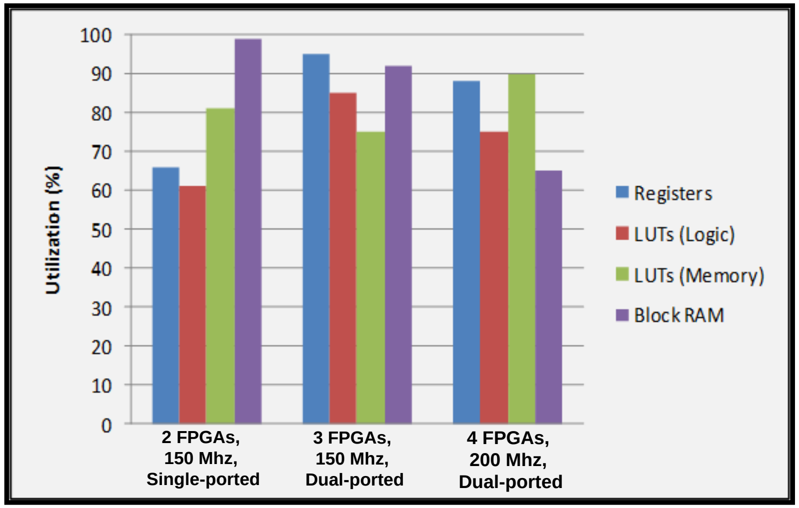
Problem Specification

22 nodes
7547 parent sets per node
100 random restarts
10,000 iterations per restart

Questions

- What's the deal with the 1x vs. 4x GPP?
- What is the "Caching Algorithm"?

Times (s):	Order		Graph
1x GPP*:	62.33	+	12.67
4x GPP*:	343.62	+	12.67
GPU:	98.42	+	12.67
2x FPGA:	8.13	+	0
3x FPGA:	5.11	+	0
4x FPGA:	4.42	+	0



GPP:	4-core Intel Xeon 3.00GHz (PowerEdge 1850), 7.71 GB RAM, 10.00 GB swap (Caching algorithm)
GPU:	1.3 GHz NVIDIA Tesla c1060 (Caching algorithm)
FPGA:	Xilinx Virtex-5 LX155T (-2)

Future Work

- Order caching

  Insight: A given order will always produce the same score
  - Optimization used by both GPU & GPP implementations
  - Can be made at an order or “local order” granularity
- Pre-processing on FPGA
  - (1) “Pre-processing” has become new bottleneck
  - Map “Local score” generation to each FPGA in network
  - Transport “observations” data to FPGA

  Insight: Observation files are small, score files are large
- Map Kernel to OpenRCL platform

Conclusion and Summary

This work coordinates clusters of FPGA accelerators

In order to learn protein network structure

- Reconfigurable Computing gives us the ability to…
  - Build each accelerator to best-fit different problems
  - Provide arbitrary design scaling with low overhead

Acknowledgements

For making this work possible, a special thanks to:

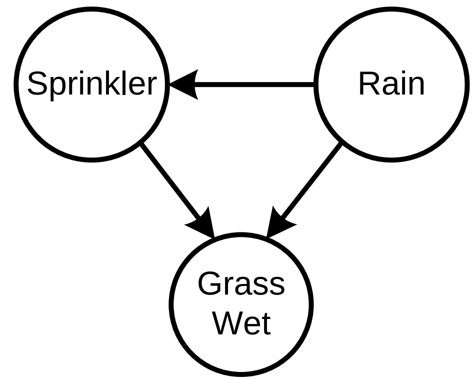
- Ilia Lebedev & Mingjie Lin for the **Pl**atform **I**nterconnect **N**etwork
- Dan Burke & Farzad Fard for developing the BEE3 EmCon
- All GateLib contributors
- NIH Grant #130826-02
- NSF Grants #0403427 & #0551739
- Berkeley Wireless Research Center (BWRC)
- Gigascale Systems Research Center (GSRC)

BACKUP SLIDES

Bayesian Networks

- "Belief Network"
  - Directed acyclic graph
  - Structure encodes…
    - Conditional independence
    - Causal relationships

	Sprinkler	
Rain	T	F
F	.4	.6
T	.01	.99



Rain	
T	F
.2	.8

Parent Set for node V_i

$$P(V_1,\ldots,V_N) = \prod_{i=1}^{N} P(V_i \mid \Pi_i)$$

		Grass Wet	
Sprinker	Rain	T	F
F	F	0	1
F	T	.8	.2
T	F	.9	.1
T	T	.99	.01

Courtesy of Tom Griffiths (U.C. Berkeley)

- Bayesian Score
  - A basis for comparing Bayesian Structures
  - Based on prior belief and observations

Experimental data

$$P(D,G) = \underbrace{P(G)}_{\text{Prior probability}} P(D \mid G)$$

Graph