

Layout-Accurate Design and Implementation of a High-Throughput Interconnection Network for Single-Chip Parallel Processing

Aydin O. Balkan **2nd** speaker

Michael N. Horak

Gang Qu

Uzi Vishkin **1st** speaker

Hot Interconnects 2007



A. JAMES CLARK SCHOOL *of* ENGINEERING

Commodity computer systems

Chapter 1 1946—2003: Serial. Clock frequency: $\sim a^{y-1945}$

Chapter 2 2004--: Parallel. #”cores”: $\sim d^{y-2003}$ Clock freq: flat.

Programmer’s IQ? Flat..

Need A general-purpose parallel computer framework that:

- (i) is **easy to program**;
- (ii) gives good performance with **any amount of parallelism** provided by the algorithm; namely, up- and down-scalability including **backwards compatibility** on **serial** code;
- (iii) supports application programming (VHDL/Verilog, OpenGL, MATLAB) and performance programming; and
- (iv) fits **current** chip **technology** and **scales** with it.

PRAM-On-Chip@UMD is addressing (i)-(iv).

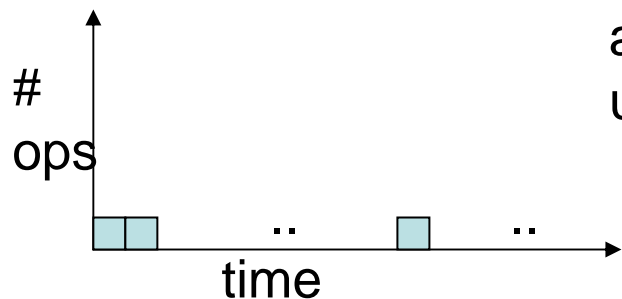
Rep speed-up [Gu-V, JEC 12/06]: 100x for VHDL benchmark.

Parallel Random-Access Machine/Model (PRAM)

Serial RAM Step: 1 op (memory/etc).

PRAM Step: many ops.

Serial doctrine

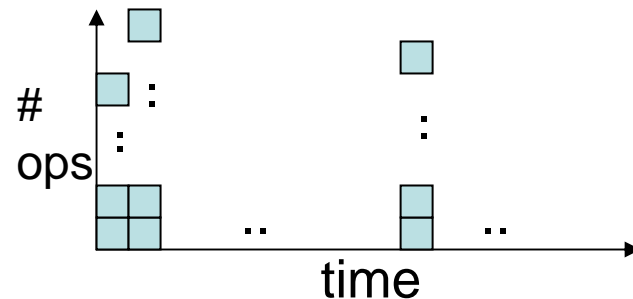


time = #ops

What could I do in parallel
at each step assuming
unlimited hardware



Natural (parallel) algorithm



time \ll #ops

1979- : THEORY figure out **how to think algorithmically in parallel**
(Also, ICS07 Tutorial)

“In theory there is no difference between theory and practice but
in practice there is” →

1997- : PRAM-On-Chip@UMD: derive **specs for architecture;**
design and build

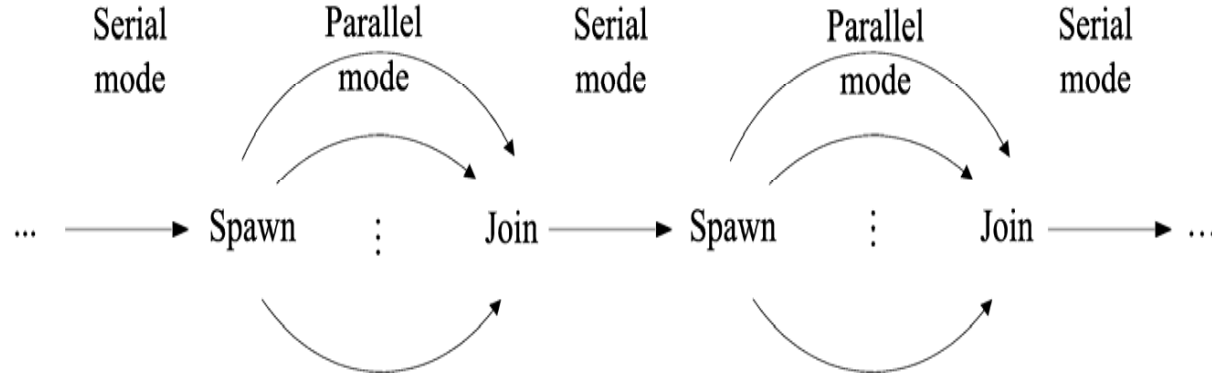
Snapshot: XMT High-level language

XMTC: Single-program multiple-data (SPMD) extension of standard C.

Arbitrary CRCW PRAM-like programs.

Includes Spawn and PS - a multi-operand instruction. Short (not OS) threads.

To express architecture desirables present PRAM algorithms as:
[ideally: compiler in similar XMT assembly; e.g., locality, prefetch]



Cartoon Spawn creates threads; a thread progresses at its own speed and expires at its Join.

Synchronization: only at the Joins.

So, virtual threads avoid busy-waits by expiring.

New: Independence of order semantics (IOS).

Unique First parallelism. Then decomposition

[ideally: given XMTC program, compiler provides decomposition]

Compare with

Build-first figure-out-how-to-program-later architectures.

J. Hennessy 2007: “Many of the early ideas were motivated by observations of what was easy to implement in the hardware rather than what was easy to use”

No proper programming model: **poor programmability**.


Painful to program decomposition-first step in other parallel programming approaches.

Culler-Singh 1999: “Breakthrough can come from architecture if we can somehow...truly design a machine that can look to the programmer like a PRAM”

The PRAM Rollercoaster ride



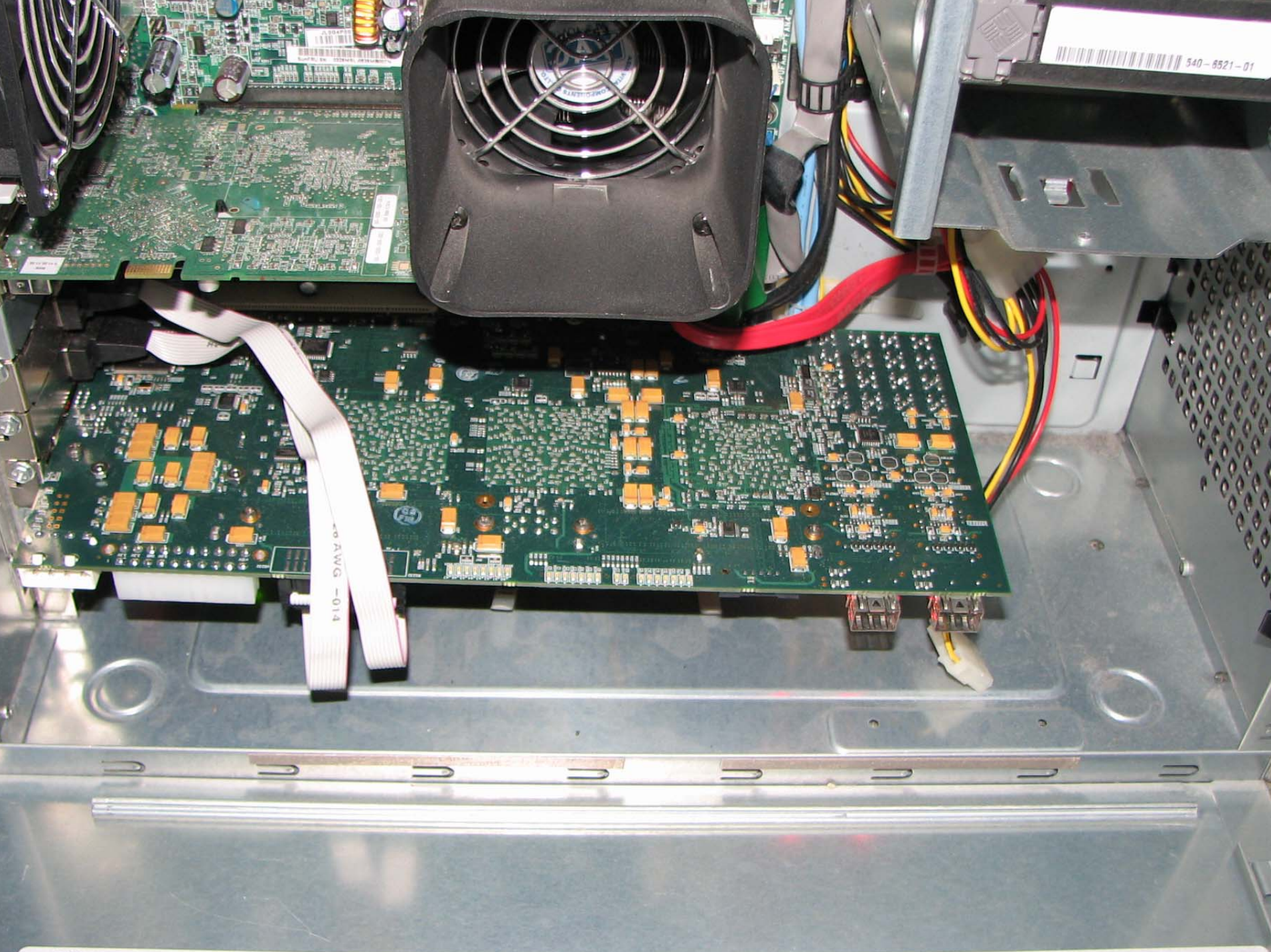
Late 1970's Theory work began

UP Won  the battle of ideas on parallel algorithmic thinking. **No silver or bronze!**

Model of choice in all theory/algorithms communities.
1988-90: Big chapters in standard algorithms textbooks.

DOWN FCRC'93: "PRAM is not feasible". ['93+ despair
→ **no** proper **alternative!** Puzzled: *where do vendors expect good alternatives to come from in 2007?*]

UP eXplicit-multi-threaded (XMT) FPGA-prototype computer (not simulator), SPAA'07; **towards realizing PRAM-On-Chip vision:**



PRAM-On-Chip

Specs and aspirations

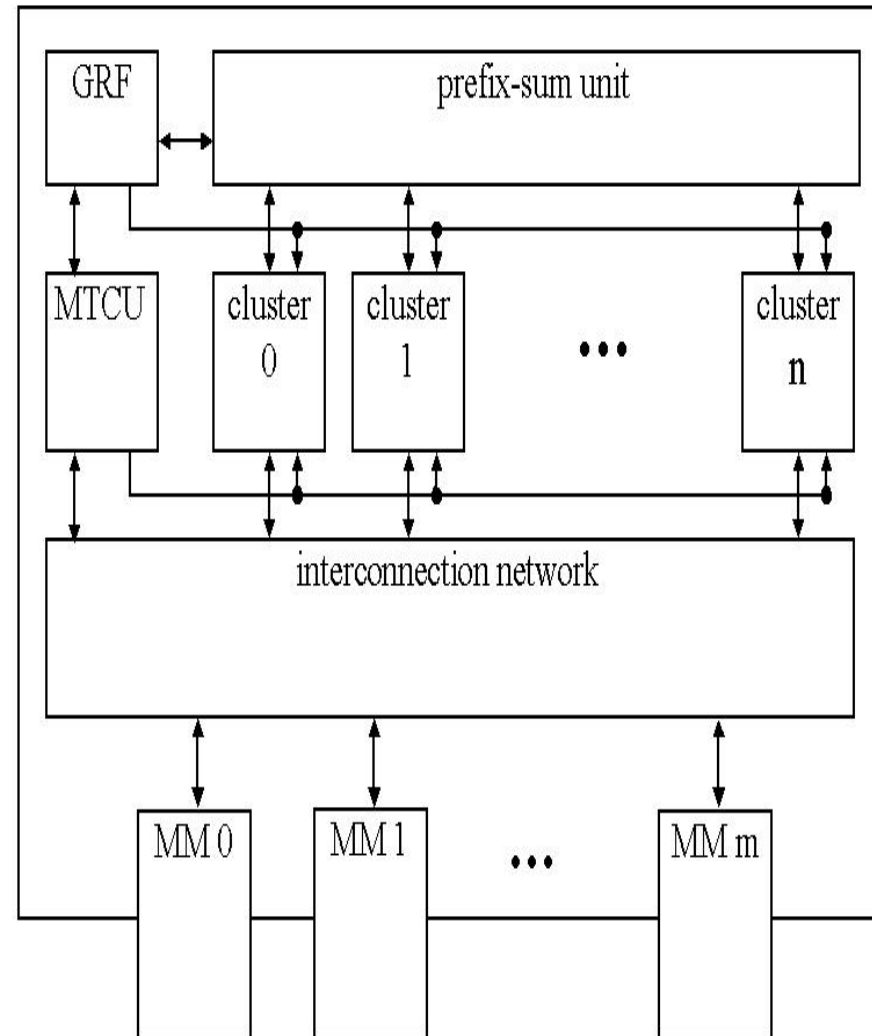
n=m	64
# TCUs	1024

- Multi GHz clock rate
- Get it to **scale to cutting edge technology**
- Proposed **answer to the many-core era: “successor to the Pentium”?**

FPGA Prototype built n=4,
#TCUs=64, m=8, 75MHz.

- Cache coherence defined away: Local cache only at master thread control unit (MTCU)
- Prefix-sum functional unit (F&A like) with global register file (GRF)
- Reduced global synchrony
- Overall design idea: no-busy-wait FSMs

Block diagram of XMT



What is different this time around?

crash course on parallel computing

– How much processors-to-memories **bandwidth**?

Enough

Limited

Ideal Programming Model: PRAM

Programming **difficulties**

In the past bandwidth was an issue.

XMT: enough bandwidth for **on-chip interconnection network**. **This paper!**

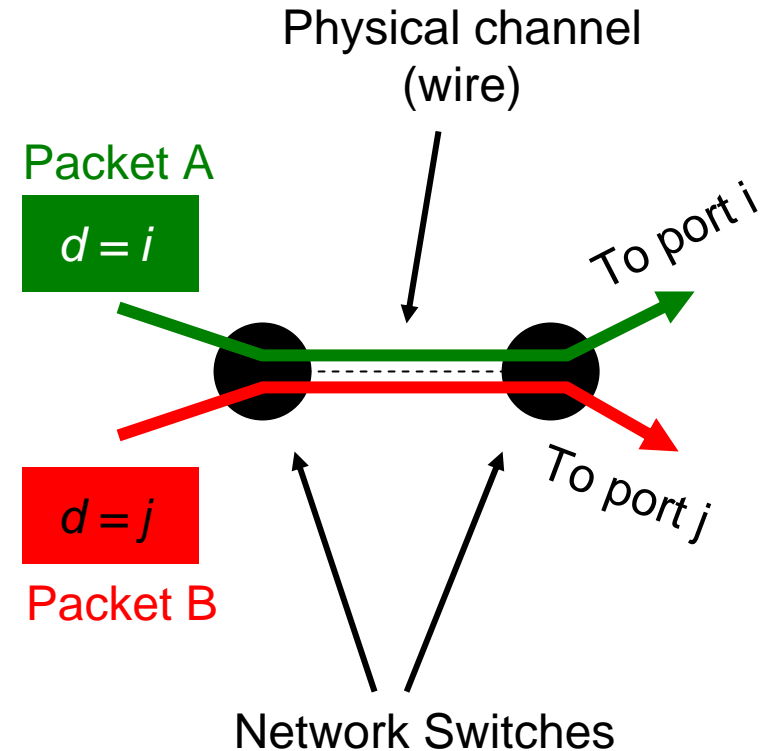
Glad to fail Einstein's test for insanity "do the same thing, yet expect different results".

One of several basic differences relative to "PRAM realization comrades": NYU Ultracomputer, IBM RP3, SB-PRAM and MTA.

PRAM was just ahead of its time, and we are getting there...

A Common Problem of Networks

- Interference
 - Path of **A** interferes with path of **B**
 - Packets are interleaved
 - Throughput is reduced to both directions



Mesh-of-Trees Topology

- Traditional Topology

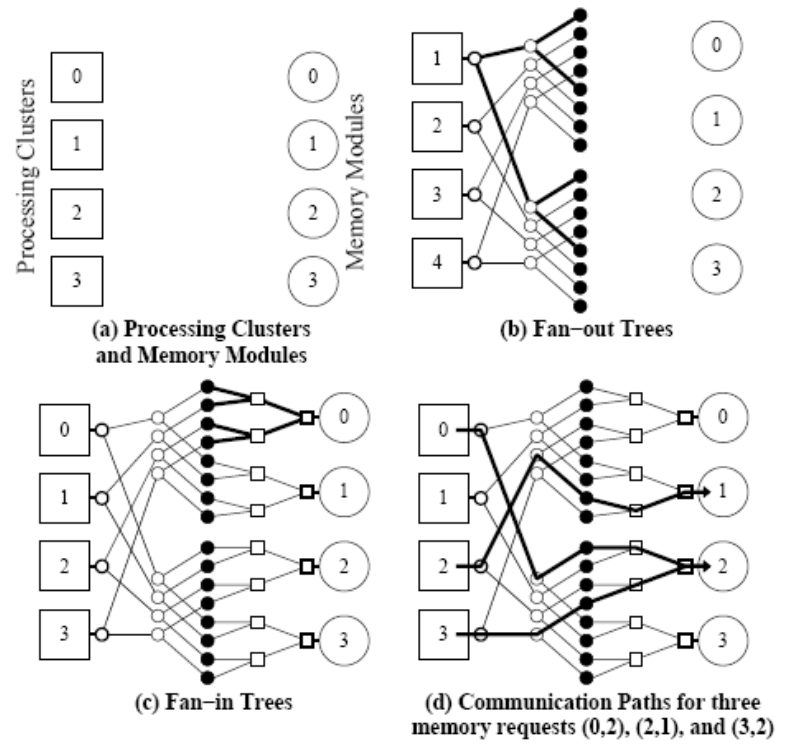
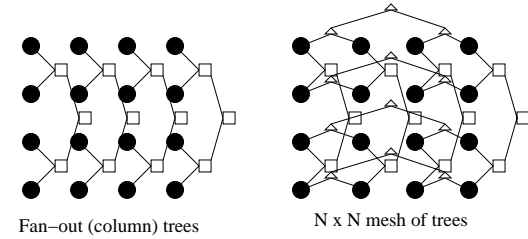
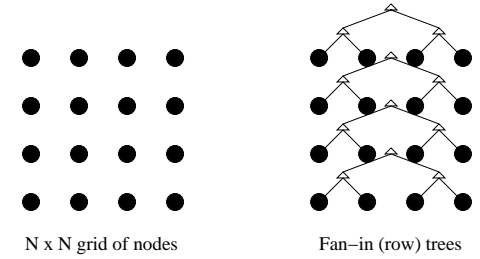
- From leaves to leaves via roots
- Interference possible

- Our Topology

- From roots to roots via common leaves
- Avoid interference

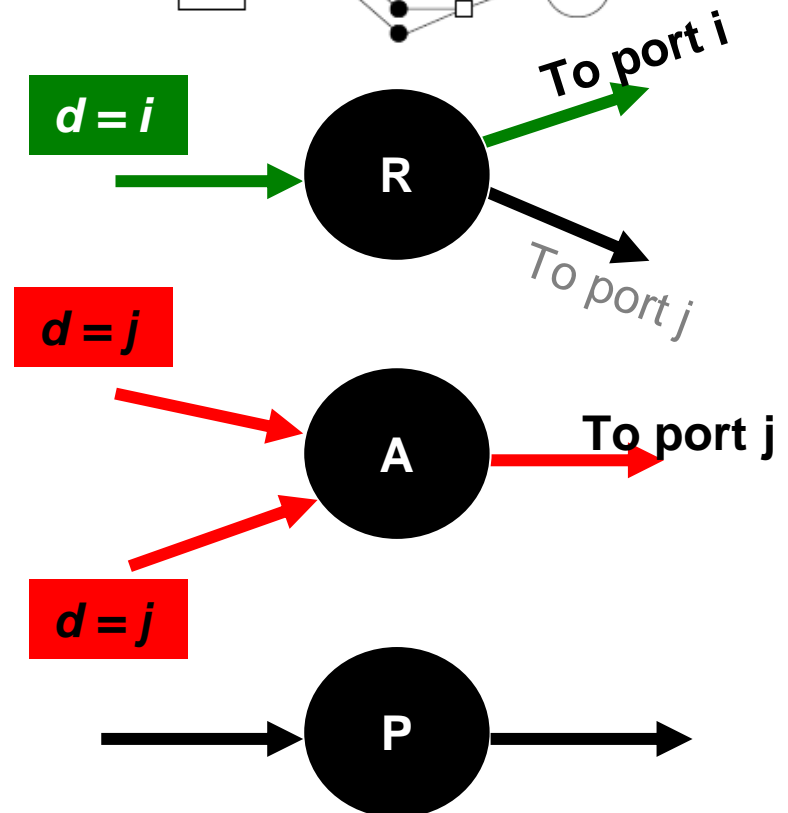
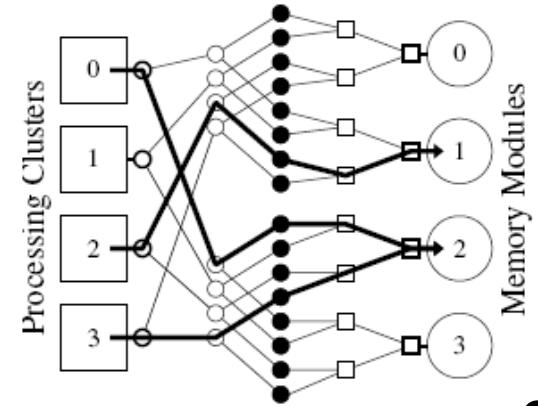
- Characteristics

- Switch degree : 2
- Levels : $2 \log N$
 - Fan-out (route) : $\log N$
 - Fan-in (arbitrate) : $\log N$
- Average hop count: $2 \log N$
- Min Bisection BW : N flits/cycle
- Ideal Throughput : N flits/cycle (1 flit/cycle per port)



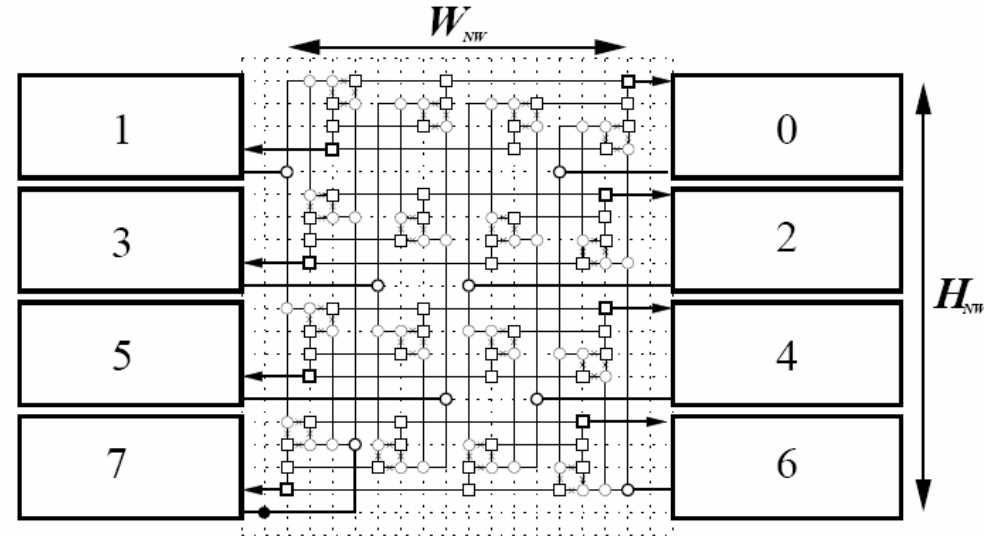
Flow Control

- Three primitive circuits
 - Route (Fan-out tree)
 - Arbitrate (Fan-in tree)
 - Pipeline (if needed)
- Simple control logic
 - Increase clock rate
- Localized decisions
 - Previous node
 - This node
 - Next node
- 2 data registers per input port



Area Complexity

- Wire : $O(N^2 \log^2 N)$
- Switch : $O(N^2)$
- IBM 90nm (9SF)
8 levels of metal
standard cell based design
 - Wire area < Switch area
- Projection to 32nm
 - $N=128$, 80 bits per flit
 - Wire area : 12.5 mm^2
 - Switch area : 27.5 mm^2



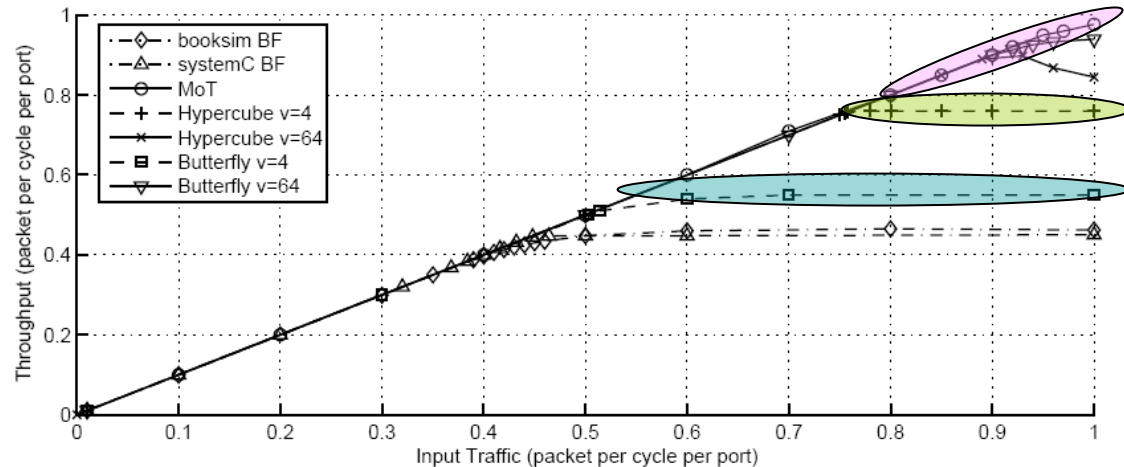
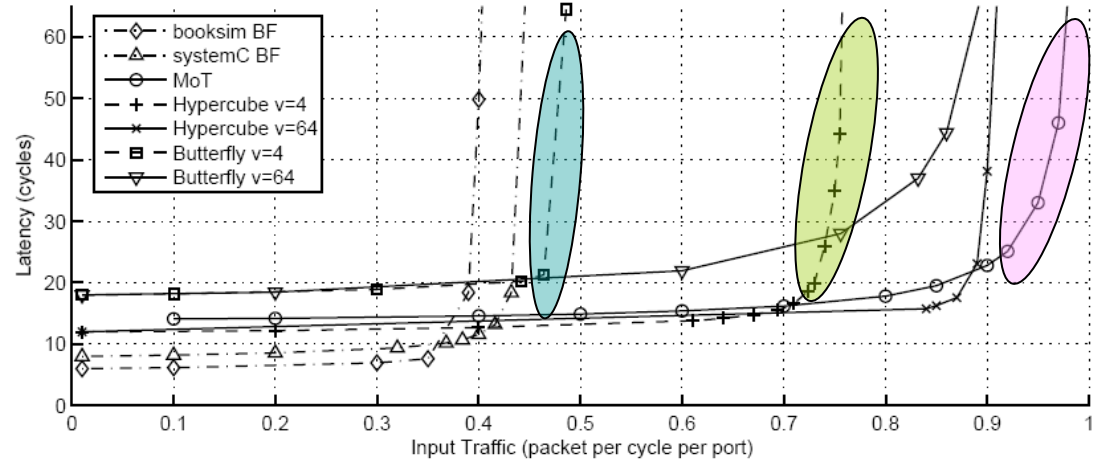
Terminals	4	8	16	32	64
Bits per flit	26	28	30	32	34
Cell Area	0.064	0.314	1.419	6.166	26.289
Wire Area	0.003	0.020	0.135	0.863	5.197

IBM 90nm process Cell vs Wire Area (mm^2)

Latency and Throughput

- Earlier results with software simulation [1]
- Hypercube
- Butterfly
- MoT
- MoT network has higher throughput and lower latency

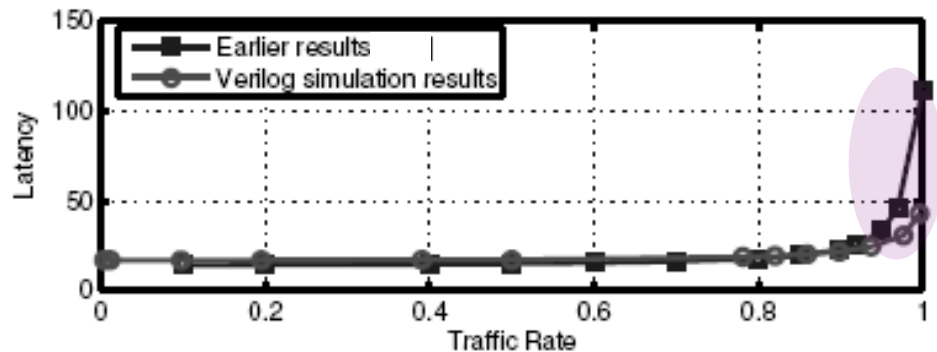
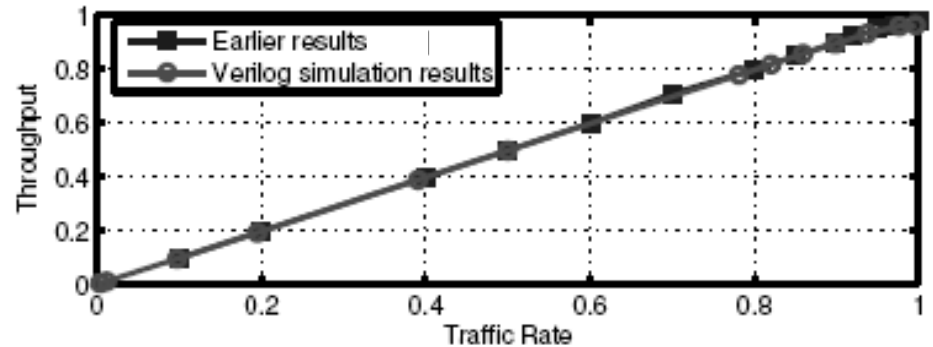
[1] A.O. Balkan, G. Qu, U. Vishkin, A Mesh-of-Trees Interconnection Network for Single-Chip Parallel Processing, Proceedings of ASAP 2006



Simulations with N=64 terminal network

Cycle-Accurate Validation

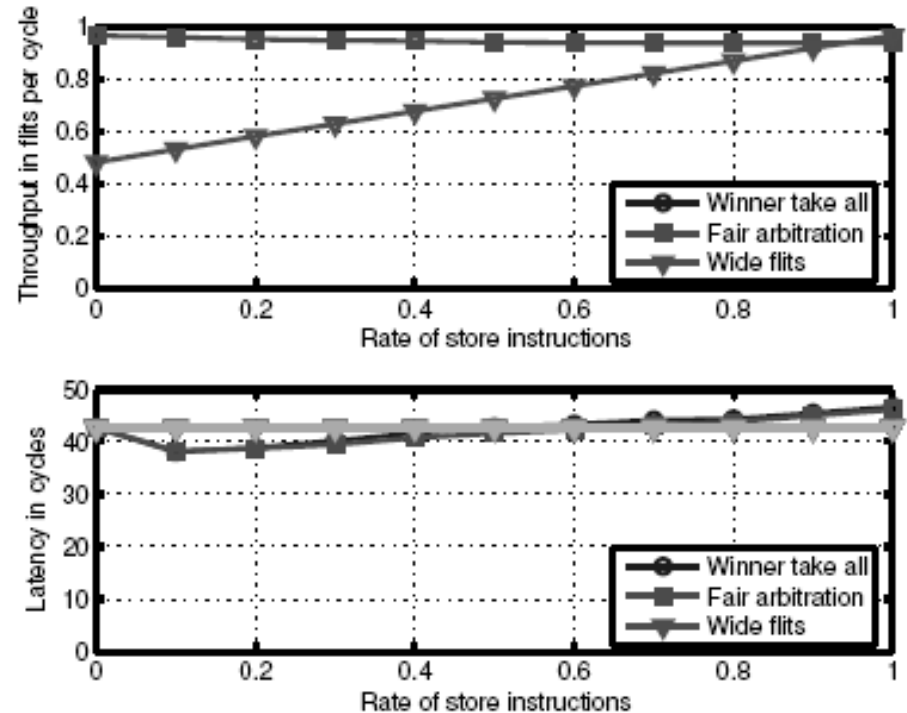
- Earlier results with in-house simulator
- Verified with verilog RTL and netlist simulations
- Uniform traffic
- 1 flit per packet
- Measured after warm-up
- 64-terminal network
 - Avg Tput 0.96 flits/cycle
 - Lat (10% traffic) 16.9 cycles
 - Lat (90% traffic) 21.6 cycles



Simulations with N=64 terminal network

Arbitration Options

- Network packets:
 - Load : 1 flit/packet
 - Store: 2 flits/packet
- Options
 - **Wide flits**
 - **Separate store flits**
 - Fair arbitration
 - Extra buffers and logic outside of network
 - **Linked store flits**
 - Winner-take-all
 - Extra logic in network, no additional buffer



Layout-Accurate Area and Performance

- Layout
 - Area, Clock Rate
 - Netlist
 - Simulated switching activity
→ Power
- Pipelining
 - Recovers performance
 - High level heuristic
 - Higher cell area
 - Higher power consumption
- One terminal can serve 16 light-weight processors [2]
 - 8 terminal → 128 processors
 - 32 terminal → 512 processors

Config.	4	8	16	32	16 p	32 p
Clock Rate	970	890	680	578	748	764
Bits per flit	26	28	30	32	30	32
Peak Tput	101	199	326	592	359	782
Avg Tput	88.6	180	302	563	334	747
Low trf lat	8.64	10.8	12.8	14.8	13.5	17.8
High trf lat	18.0	16.9	17.9	19.3	18.7	22.6
Cell area	0.08	0.41	1.89	6.5	1.88	7.3
BBbox area	0.16	0.74	3.21	13.4	3.21	13.4
Power	72	268	794	N/A*	967	N/A*

Clock Rate in **MHz**

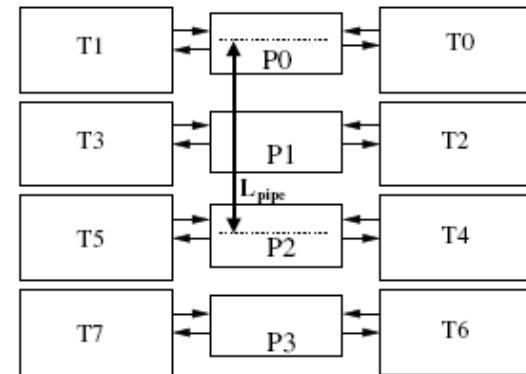
Latency in **cycles**

Low trf: 10% High trf 90%

Throughput in **Gbps**

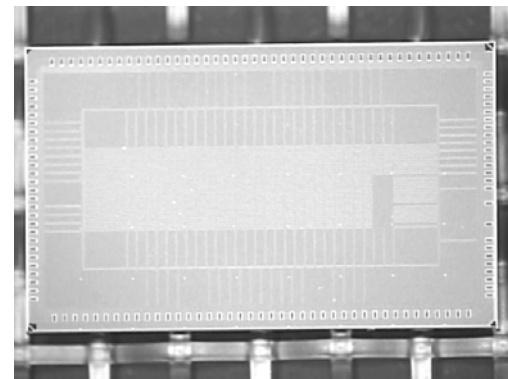
Area in **mm²**

Power in **mW**



Summary of Technical Part

- Mesh-of-Trees network
 - Provides high performance
 - High throughput allows multiple processors per terminal (e.g. 16)
 - High wire complexity, but cell area dominates for several future technology generations
- Performance verified with verilog simulations
- Arbitration options evaluated
 - Linked address and data flits in store instructions
- Layout generated for 4 to 32 terminal networks
 - Clock Rate → Throughput in Gbps
 - Area
 - Power
- 8-terminal network chip fabricated (August 2007)



Bare die photo of 8-terminal chip
IBM 90nm process, 9mm x 5mm

Conclusion

Badly needed: **HOT Alg. & Programming Models.**

Just think: How to teach algorithms & programming to students in HS & College & other programmers?

Multi-decade evidence of commercialization problems in parallel computing due to poor programmability.

Currently, only PRAM provides strong-enough theory

[Hot Interconnects, Hot Chips, compilers, etc, are crucial for bridging theory and practice]

IOHO: (i) Competition to PRAM unlikely

(ii) It is only a matter of time & money for us to complete a basis for ubiquitous general-purpose parallel computing

Experience with new FPGA computer

Included: basic compiler [Tzannes, Caragea, Barua, V].

New computer used: to validate past speedup results.

Zooming on Spring'07 parallel algorithms class @UMD

- **Standard PRAM class**. 30 minute review of XMT-C.
- Reviewed the architecture **only in the last week**.
- 6(!) significant programming projects (in a theory course).
- FPGA+compiler operated nearly flawlessly.

Sample speedups over best serial by students Selection: 13X.

Sample sort: 10X. BFS: 23X. Connected components: 9X.

Students' feedback: "XMT programming is easy" (many), "The XMT computer made the class the gem that it is", "I am excited about one day having an XMT myself! "

12,000X relative to cycle-accurate simulator in S'06. Over an hour
→ sub-second. (Year → 46 minutes.)

More “keep it simple” examples

Algorithmic thinking and programming

- PRAM model itself; and the following plans:
- Work with motivated **high-school** students, Fall’07.
- **1st semester programming course**. Recruitment tool: “CS&E is where the action is”. Spring’08.
- Undergrad parallel algorithms course. Spring’08

XMT architecture and ease of implementing it

Single (hard working) student (X. Wen) completed synthesizable Verilog description AND the new FPGA-based XMT computer (+ board) in slightly more than two years. No prior design experience.