FPGA-Based Prototype of a PRAM-On-Chip Processor

Xingzhi Wen and Uzi Vishkin

A. JAMES CLARK SCHOOL *of* ENGINEERING

Motivation

Bird-eye's view of commodity computer systems

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

Chapter 2 2004--: Parallel. #"cores": ~d^{y-2003} Clock frequency: flat.

➔Prime time is ready for parallel computing. But is parallel computing ready for prime time? We are yet to see a general-purpose parallel computer that:

(i) is easy to program;

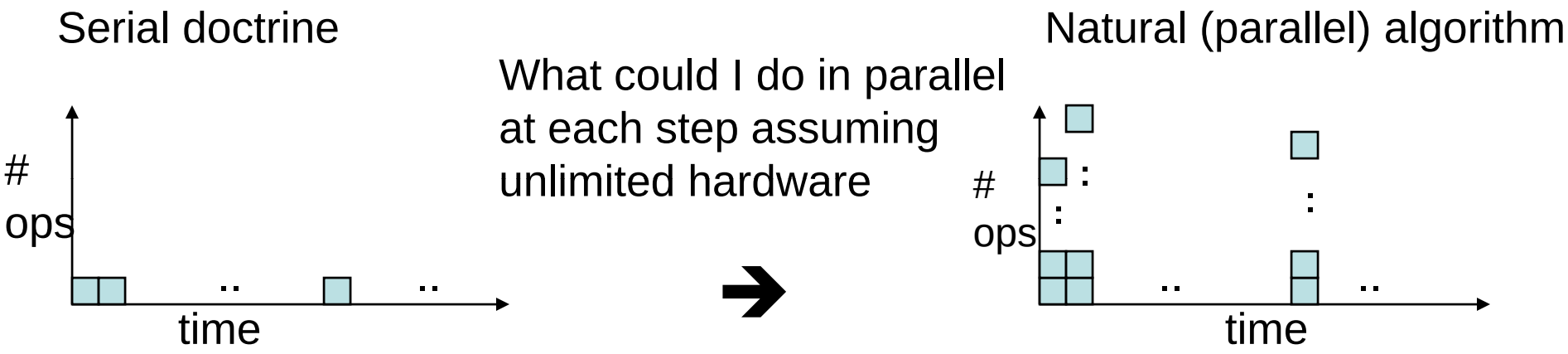
(ii) gives good performance with any amount of parallelism provided by the algorithm; namely, up- and down-scalability including backwards compatibility on serial code; and

(iii) fits current chip technology and scales with it.

This talk: PRAM on its way to address (i)-(iii).

Parallel Random-Access Machine/Model (PRAM)

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



time = #ops

time << #ops

Note: Natural extension to serial

1979- : THEORY figure out how to think algorithmically in parallel

"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

Compare with

Build-first figure-out-how-to-program-later. Yet to prove itself.

Clear lesson of decades of parallel computing research: parallel programming must be properly resolved.

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”

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 Dream

UP Won  the battle of ideas on parallel algorithmic thinking.

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

DOWN FCRC'93: "PRAM is not feasible".

UP Dream coming true? eXplicit-multi-threaded (XMT) computer; realize PRAM-On-Chip vision

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.

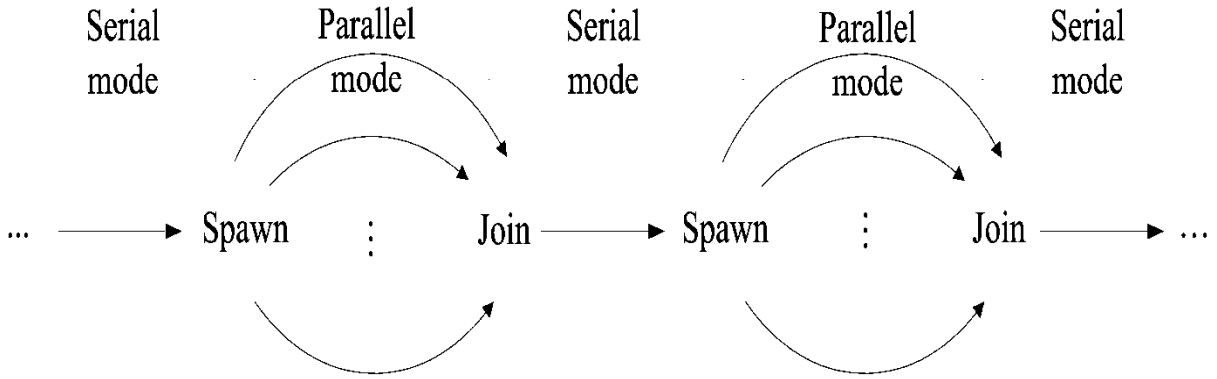
XMT: enough bandwidth for on-chip interconnection network. [Balkan,Horak,Qu,V-HotInterconnects'07: 9mmX5mm, 90nm ASIC tape-out—"Layout-accurate"]

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

Can PRAM-On-Chip address (i)-(iii) in Motivation slide?
Yes ➔ PRAM was just ahead of its time

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).

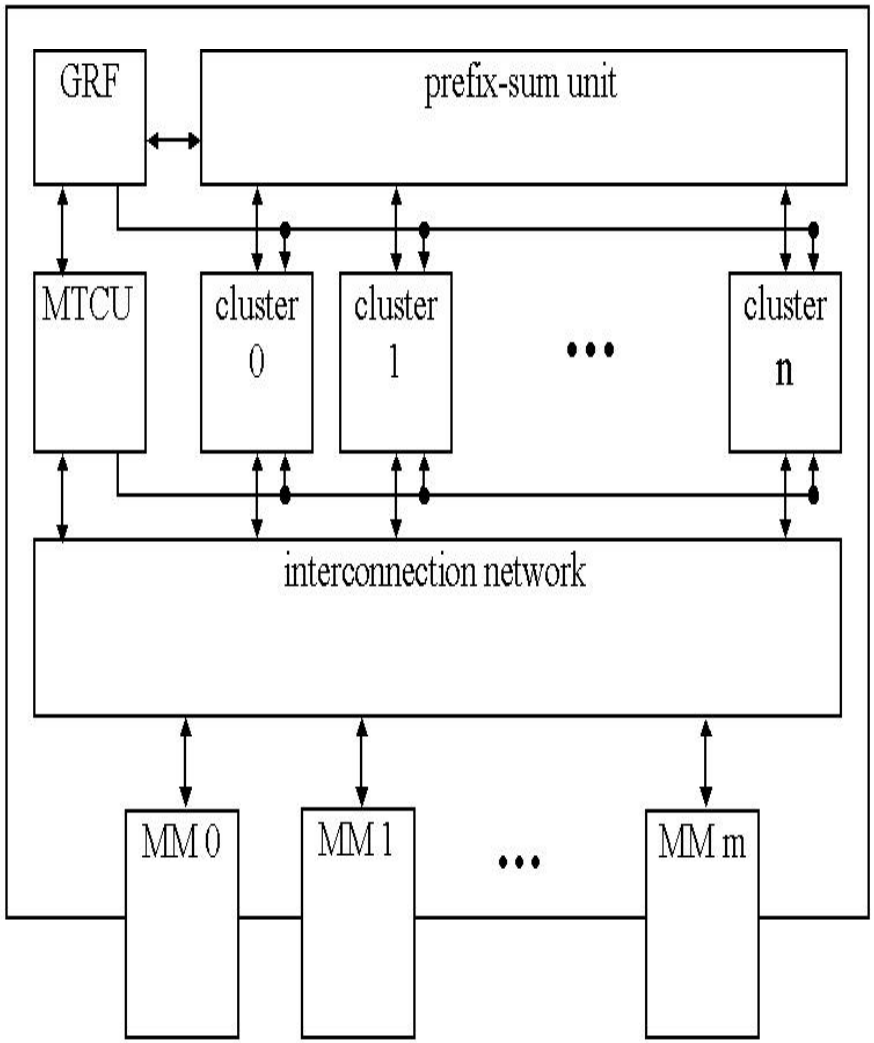
PRAM-On-Chip

Specs and aspirations

n=m	64
# TCUs	1024

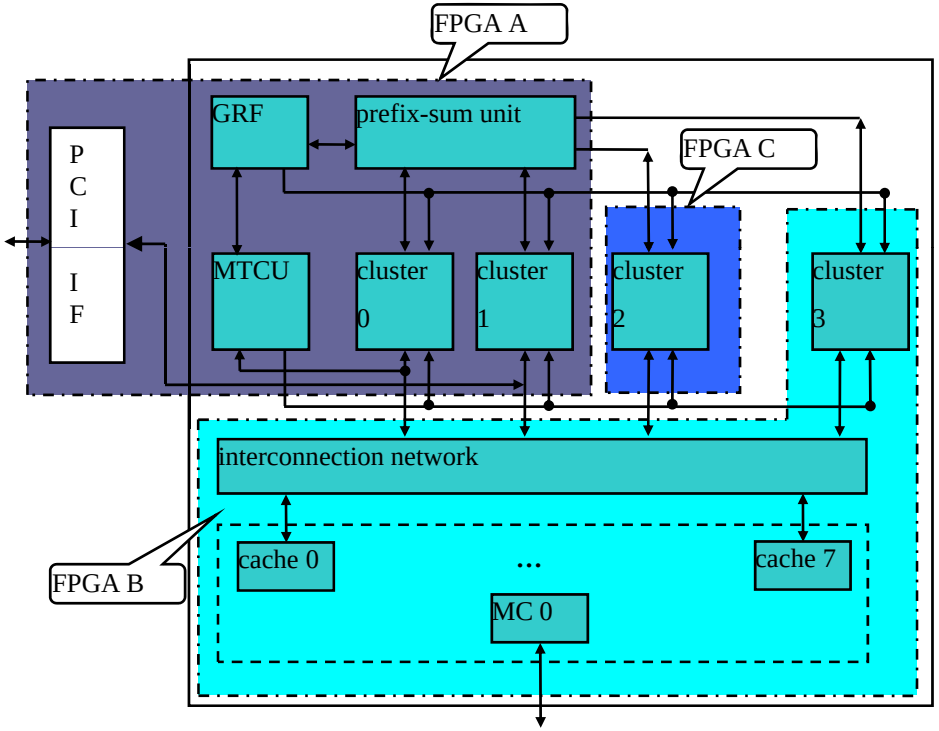
- Multi GHz clock rate
- Get it to scale to cutting edge technology
- Proposed answer to the many-core era: "successor to the Pentium"?

Block diagram of XMT



- 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 the prototype

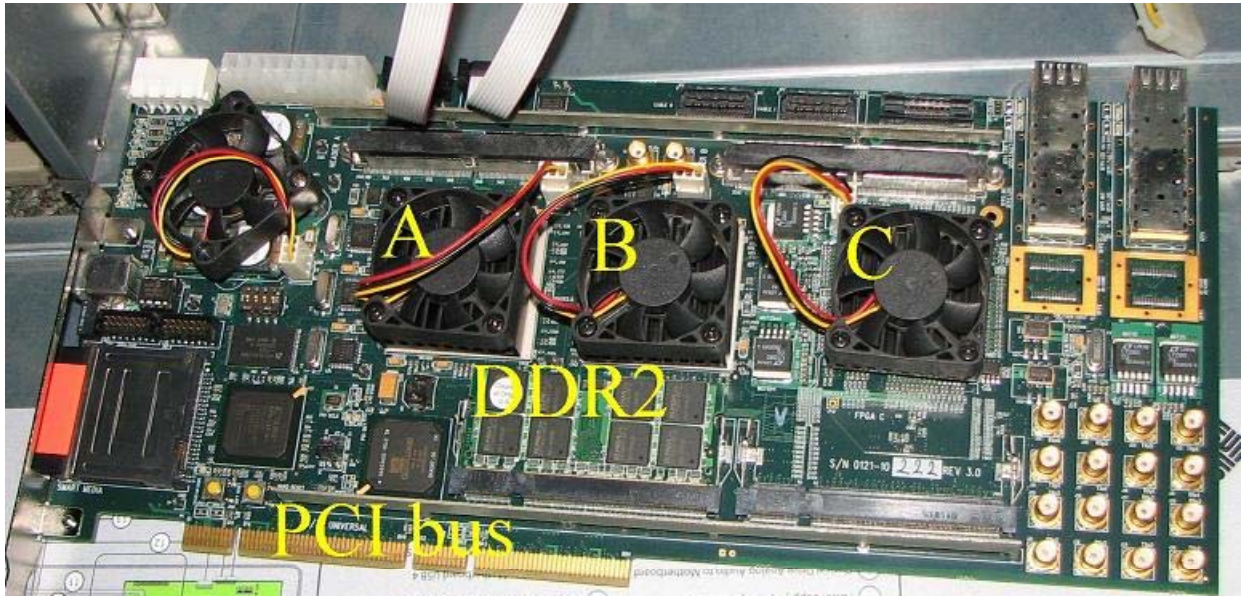


FPGA A,B Virtex-4 LX200
FPGA C Virtex-4 FX100 (smaller than LX200)*
*Constrained by the availability of the development board.

- 64-TCU, 75MHz XMT processor
  4cluster x 16 TCU/cluster
- DDR2 DRAM controller
  1GB DRAM, 2.4GB/s
- PCI bus to the host computer (PC) – Interface (IF) shown

- Prefetch buffers per TCU
- Read-only buffer per cluster
- Instruction/value broadcasting

The prototype



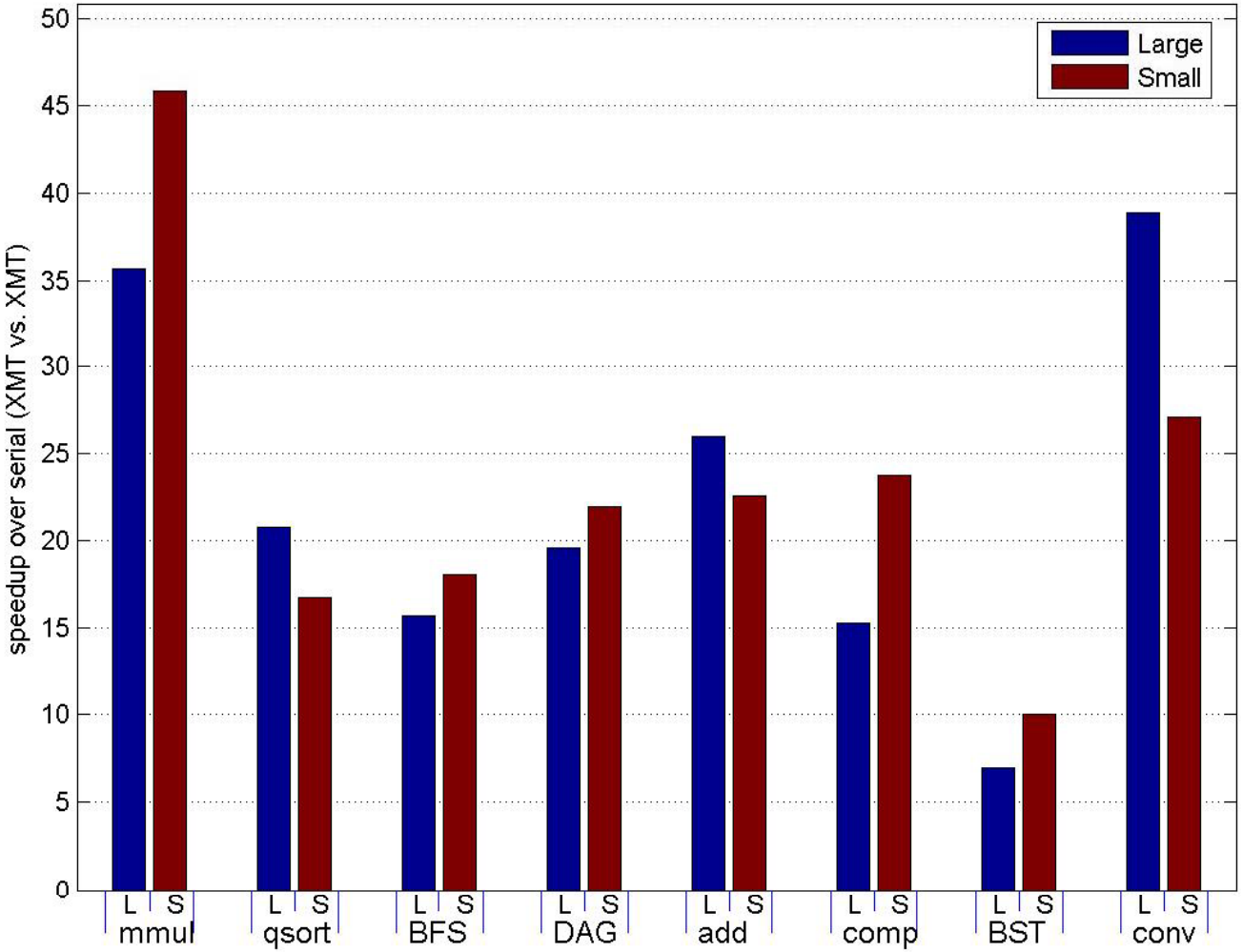
Item	FPGA A		FPGA B		Capacity of Virtex 4
	Used	%	Used	%	
Slices	84479	94	89086	99	89088
BRAMs	321	95	324	96	336

Notes: 80% of FPGA C was used. BRAM: FPGA-builtin SRAM.

Kernel benchmarks

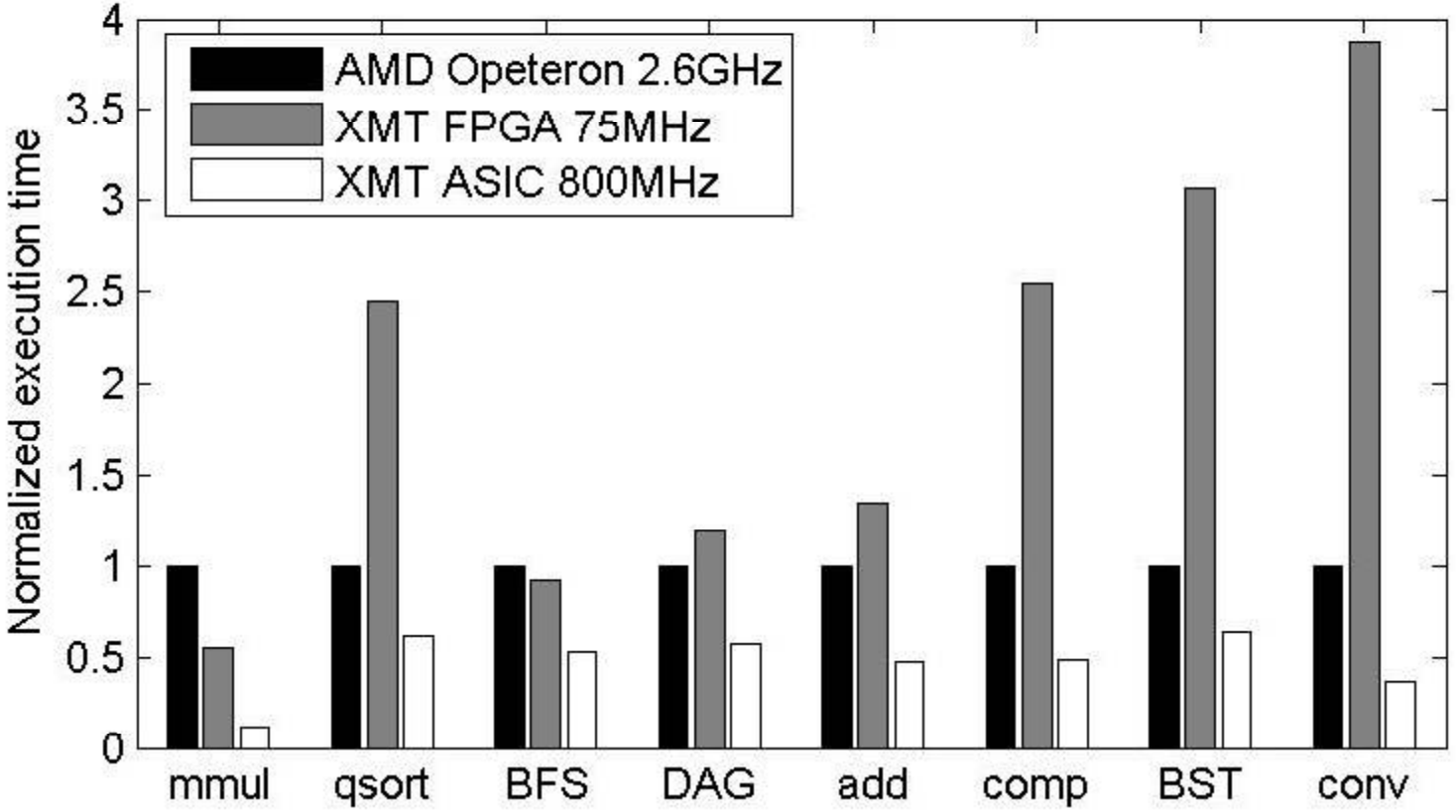
APP	Brief description	Large size	Small size
mmul	Matrix multiplication	2000x2000	128x128
qsort	Quick sort	20M	100K
BFS	Breadth first search	V=1M,E=10M	V=100K,E=1M
DAG	Finding longest path in a directed acyclic graph	V=1M,E=17M	V=50K,E=600K
add	Array summation	50M	3M
comp	Array compaction	20M	2M
BST	Binary search tree	16.8M nodes, 512K keys	2.1M nodes, 16K keys
conv	convolution	image: 1000x1000 filter:32x32	image: 200x200 filter:16x16

Speedups (Par. Vs. Ser. in XMT)



Cache size issues. Prefetch buffers and Read-only buffers were limited.

Normalized execution time



- The 800MHz projection based on conservative emulation
- The prototype is not fully optimized yet (generations of 100s of professional engineers on serial processor vs. single graduate student on serial+parallel; limited capacity of FPGA)

Experience with new FPGA computer

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

New computer used: to validate past speedup results.

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), "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.)

Experience with High School Students, Fall'07

Gave 1-day parallel algorithms tutorial to 12 HS students. Some managed 8 programming assignments, including 5 of the 6 in the grad course*. Only help: 1 office hour/week by undergrad TA. No school credit. Part of a computer club after 8 periods/day.

Underway (May-June 08): 23 HS students, by HS teacher, Alexandria, VA

Spring'08: Course to non-major Freshmen.

Recruitment tool: "CS&E is where the action is"

*The 6th was the Bilardi-Nicolau Bitonic Sort

Conclusion

Milestone toward getting PRAM ready for prime time.

Successful general-purpose approach must (also) answer: what will be taught in the algorithms class?

Not much choice beyond PRAM. Even the 1993+ despair did not produce proper alternative.

Impasse All vendors committed to multi-cores. Yet, their architecture and how to program them for single task completion time not clear (are they clueless?) ➔Avoid investment in long-term application SW development since: (i) architecture is about to change, and/or (ii) may bet on the wrong horse

Actionable agenda (for education) Will have to teach the basics of parallelism. The PRAM level of cognition is necessary, as it falls in the common denominator of other approaches. Just teach it…

This paper: It is also sufficient for a real architecture.

Instructors welcome to join

1. Teach parallel algorithmic thinking.
2. Give PRAM-like programming assignments.
3. Have your students' compile and run remotely on publically posted (by 6/2008) simulator.

How to Teach: www.umiacs.umd.edu/users/vishkin/XMT

Compare with (painful to program) decomposition step in other approaches.

Naming Contest for New Computer

➔Paraleap

chosen out of ~6000 submissions

Single (hard working) person (X. Wen) completed synthesizable Verilog description AND the new FPGA-based XMT computer in slightly more than two years. No prior design experience. Attests to: basic simplicity of the XMT architecture and ease of implementing it.

Back-up slides: Some experimental results

- AMD Opteron 2.6 GHz, RedHat Linux Enterprise 3, 64KB+64KB L1 Cache, 1MB L2 Cache (none in XMT), memory bandwidth 6.4 GB/s (X2.67 of XMT)
- M_Mult was 2000X2000 QSort was 20M
- XMT enhancements: Broadcast, prefetch + buffer, non-blocking store, non-blocking caches.

XMT Wall clock time (in seconds)

App.	XMT Basic	XMT	Opteron
M-Mult	179.14	63.7	113.83
QSort	16.71	6.59	2.61

Assume (arbitrary yet conservative)

ASIC XMT: 800MHz and 6.4GHz/s

Reduced bandwidth to .6GB/s and projected back by 800X/75

XMT Projected time (in seconds)

App.	XMT Basic	XMT	Opteron
M-Mult	23.53	12.46	113.83
QSort	1.97	1.42	2.61

Nature of XMT Enhancements

Question Can innovative algorithmic techniques exploit the opportunities and address the challenges of multi-core/TCU?

Ken Kennedy's answer: And can we teach compilers some of these techniques?

Namely: (i) identify/develop performance models compatible with PRAM; (ii) tune-up algorithms for them (can be quite creative); (iii) incorporate in compiler/architecture.

Back-up slide: Explanation of Qsort result

The execution time of Qsort is primarily determined by the actual (DRAM) memory bandwidth utilized. The total execution time is roughly = memory access time + Extra CPU time

6.4GB/s is the maximum bandwidth that memory system provides. However, the actual utilization rate depends on the system and application.

So, XMT seem to have achieved higher bandwidth utilization than AMD.

Envisioned XMT processor