The XMT FPGA Prototype/Cycle-Accurate-Emulator Hybrid

Xingzhi Wen, Uzi Vishkin
Institute for Advanced Computer Studies (UMIACS)
Department of Electrical and Computer Engineering
University of Maryland

June 22, 2008

Outline

Outline

Motivation

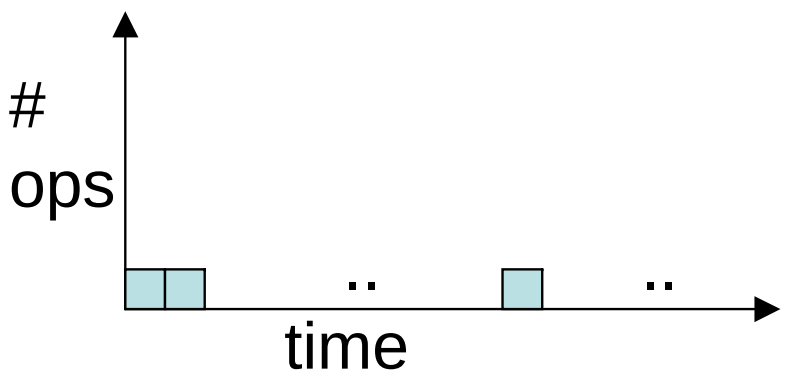
- End of exponential performance improvements in sequential computer.
  - Clock rate unlikely to increase as used to
  - Diminishing return stage for ILP.
- No choice but parallel computing - chip multiprocessor
- What is the challenge?
  Building a parallel computer is not a trivial task. But researchers have built quite a few parallel computers. Building a parallel computer that is *easy to program* is much more humbling. We are yet to see one following decades of effort.

PRAM and Sequential Programing

PRAM - Parallel Random Access Model/Machine
Objective of the XMT: Efficiently execute PRAM-like algorithms
PRAM is a natural extension to sequential programming model

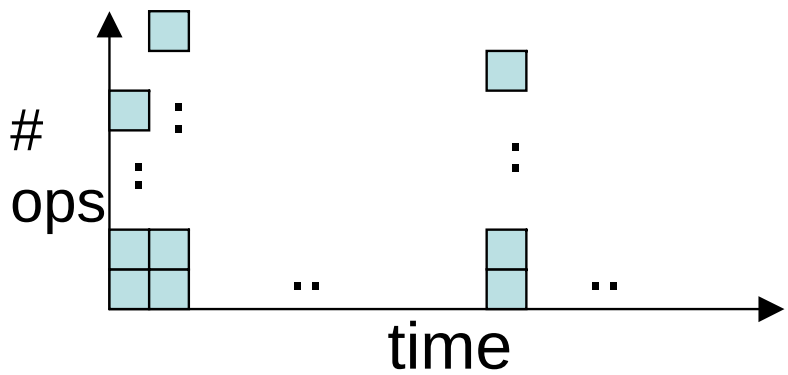
- Sequential RAM: One operation at a time. Sequential between operations.
- PRAM: Many concurrent operations at one round. Sequential between rounds.

Serial doctrine



time = #ops

Natural (parallel) algorithm

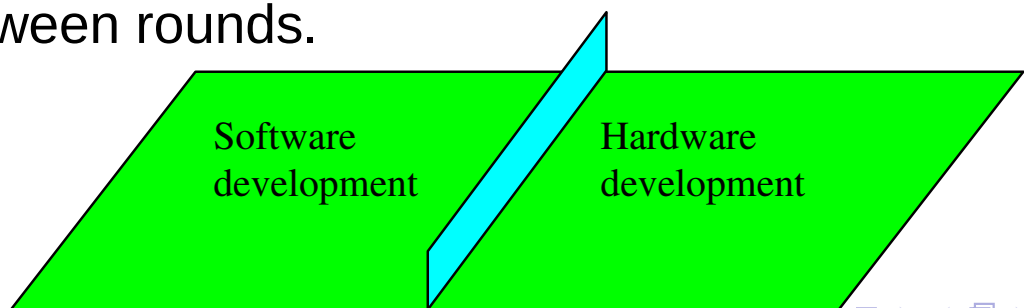


time << #ops

Decoupling SW and HW Development

Current solution: Like assembly programming [Report to President, June 2005]

- Sequential RAM:
  - Software: Minimize total number of operations required for a program assuming sequential execution.
  - Hardware: Maximize the number of instructions executed in unit time, while keeping the sequential semantics.
- PRAM:
  - Software: Minimize total number of operations and number of rounds.
  - Hardware: (1) Maximize the number of “concurent” instructions executed in unit time, preserving parallel semantics. And (2) maximize the number of rounds executed in unit time, preserving sequential semantics between rounds.

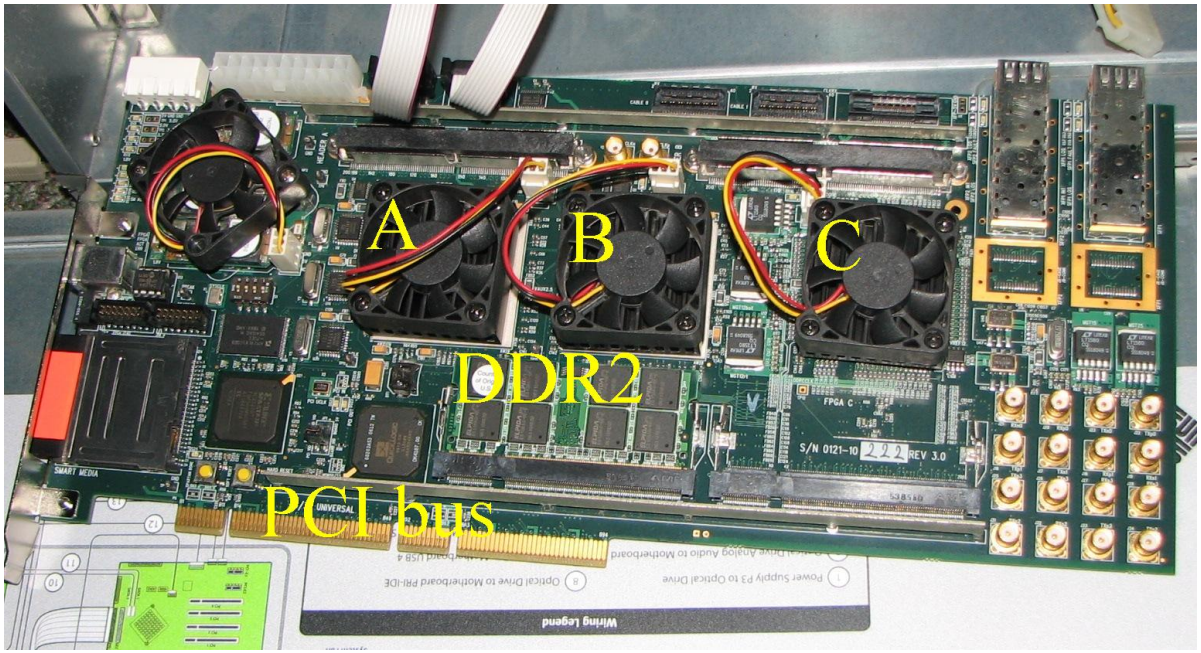


Historical Perspective on PRAM

- PROs:
  - Only approach that was ever endorsed by the theory of computer science
  - Chapter in standard algorithms & data-structure textbooks by 1990
  - (Later) XMT programming successfully taught to high school students.
- CONs:
  - PRAM is oversimplified.
  - “impossible in reality”
- What is new?
  - On-chip multiprocessor architecture is possible
  - Sufficient on-chip communication bandwidth is available.

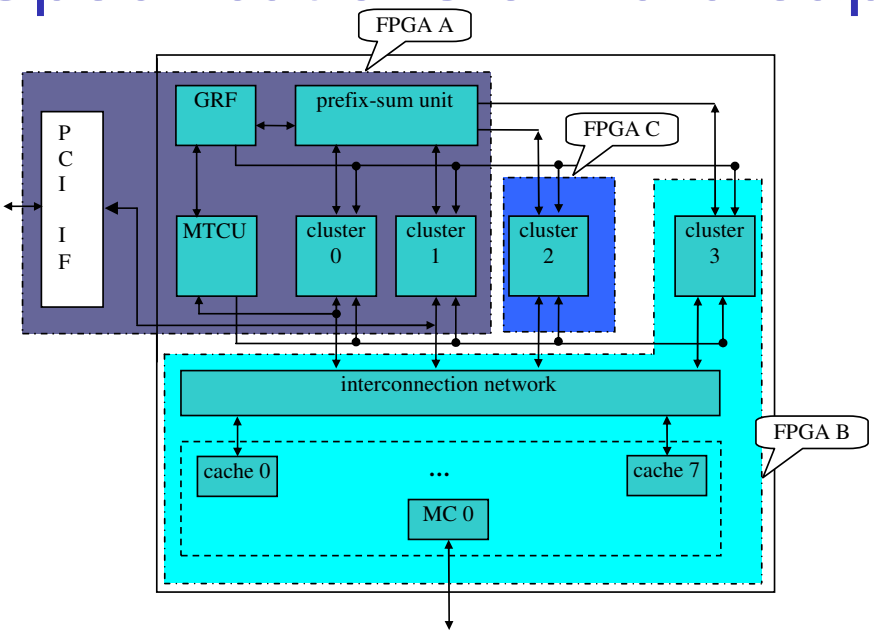
Outline

The Paraleap



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

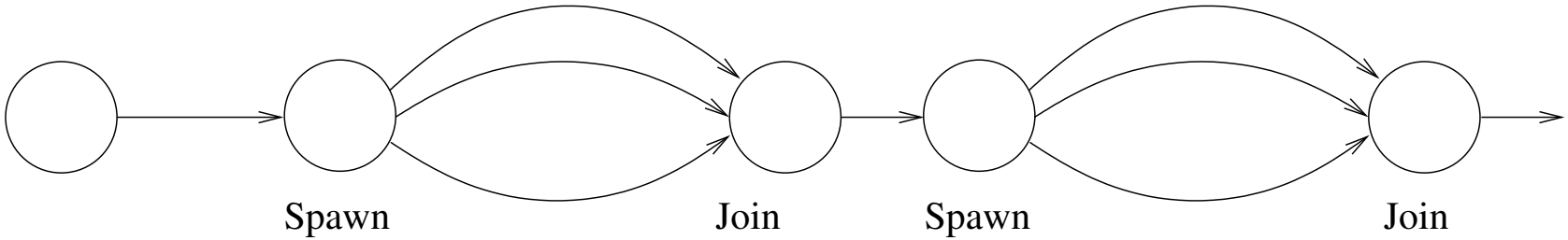
Specifications of Paraleap



- 64-TCU, 75MHz
- 1GB DDR2 DRAM
- PCI bus to host PC

number of clusters	4
number of TCUs per cluster	16
DRAM data rate	2.4GB/s
number of shared cache modules	8
size of a shared cache module	32KB
MTCU local cache	8KB
MTCU cache access latencies	1:25:51
TCU cache access latencies	2:9:30:56

Introduction to XMT architecture



- SPMD (Single Program Multiple Data)
- Arbitrary CRCW (Concurrent Read, Concurrent Write)
- IOS (Independence of Order Semantics)
- Dynamic load balancing

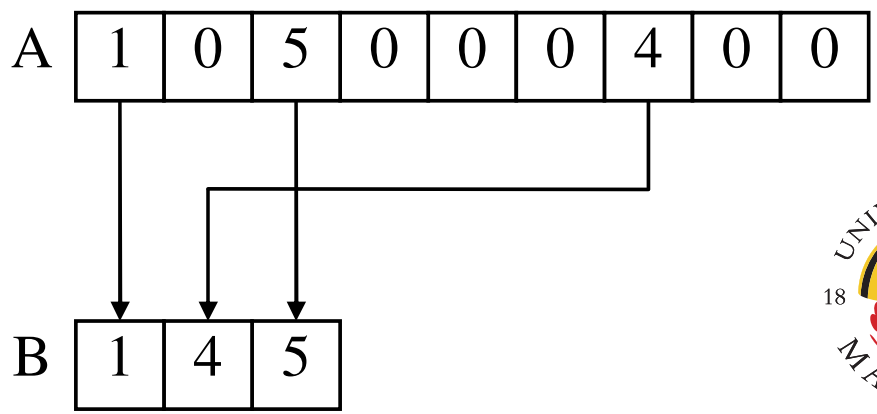
Example

Array compaction (artificial) problem:

Input: A: Array of n integer elements

Map in some order all A(i) not equal to 0 to array B

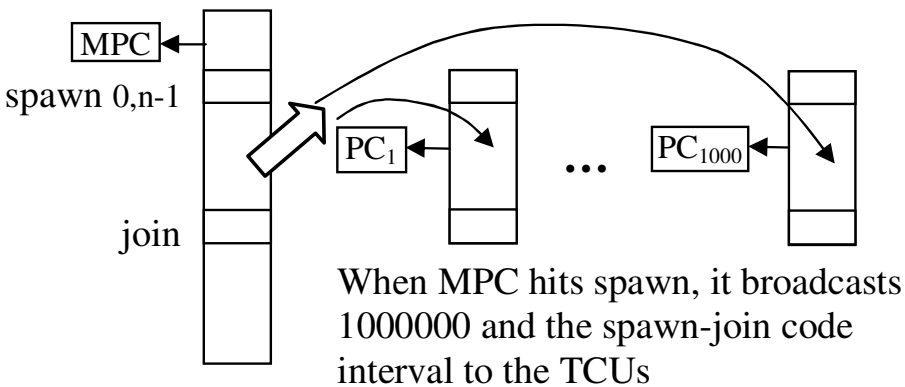
```
psBaseReg x=0;
spawn(0,n-1){//assume n=1000000
   int e=1;
   if(A[$]) != 0){
      ps(e,x);
      B[e] = A[$];
   }
}//implicit join
```



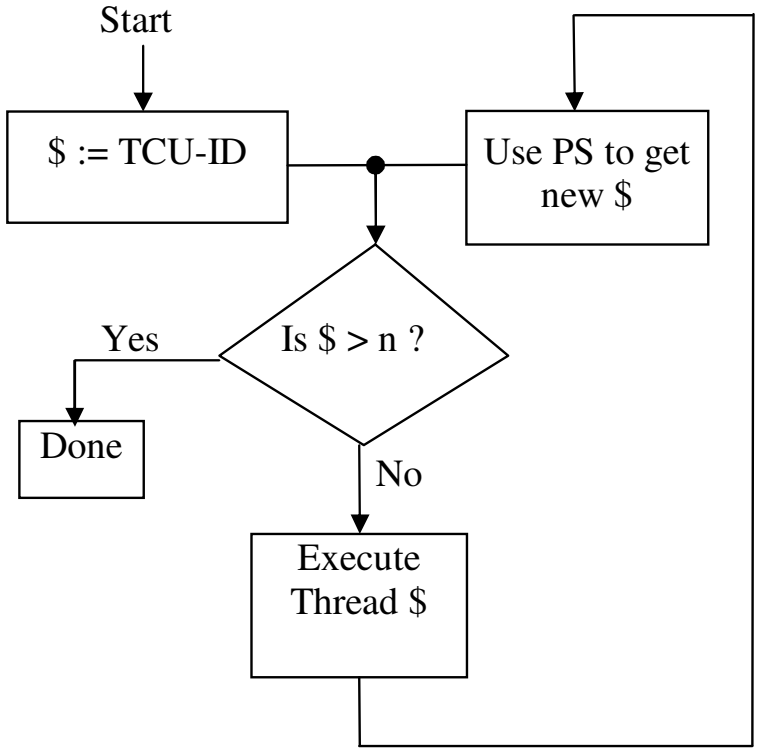
How XMT Processor Works?

```
psBaseReg x=0;
spawn(0,n-1){//assume n=1000000
    int e = 1;
    if(A[$]) != 0){
        ps(e,x);
        B[e] = A[$];
    }
}//implicit join
```

(a) XMTC program (repeated)

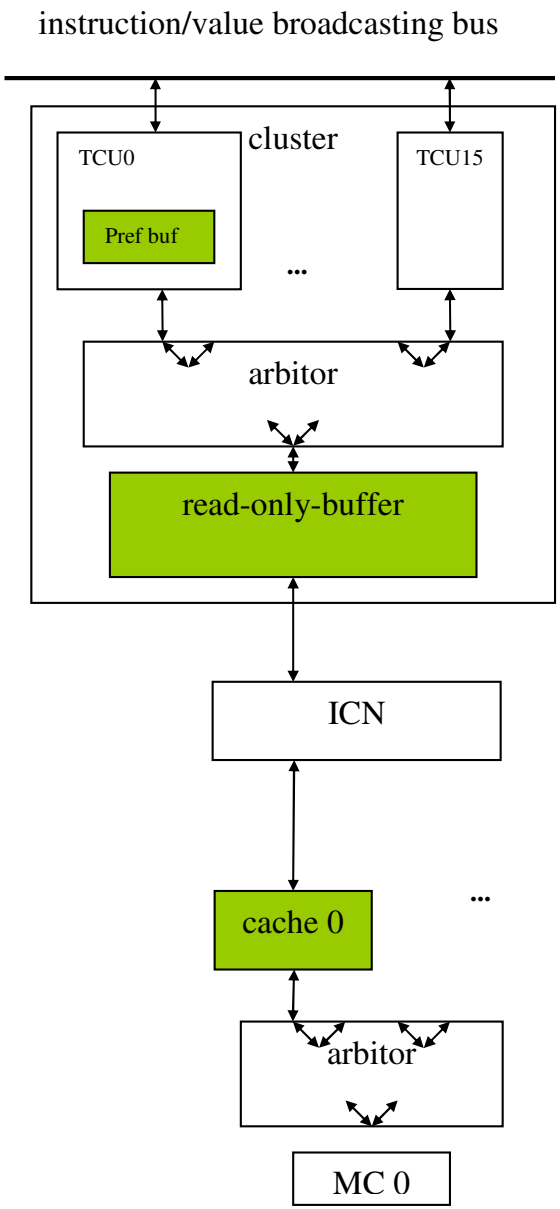


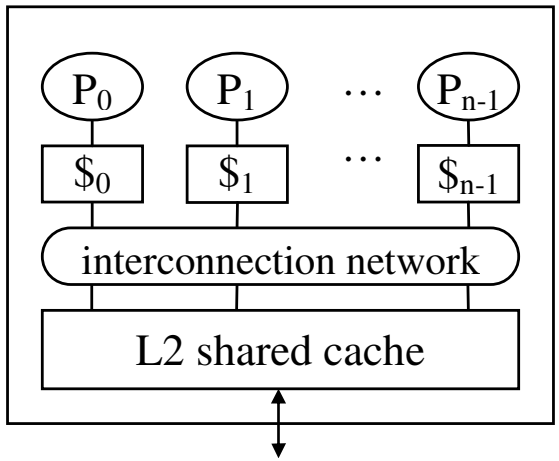
(b) Program counter + stored program



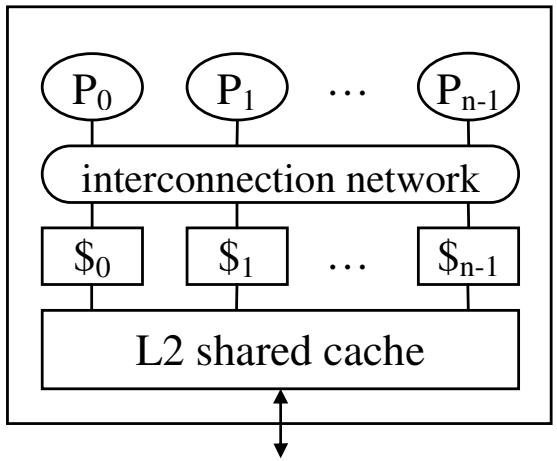
(c) TCU execution flow

Memory Hierarchy in XMT





(a) with private cache



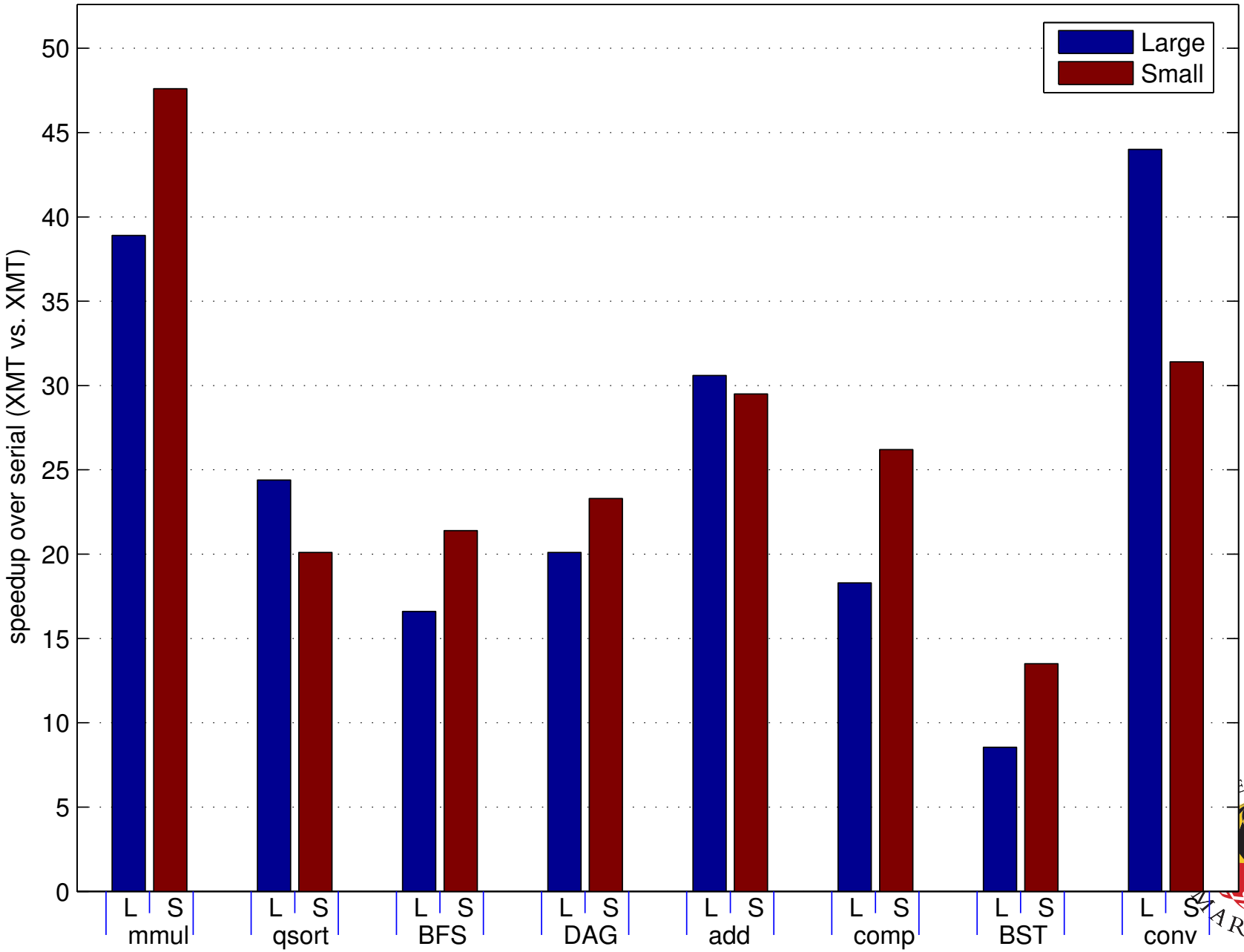
(b) shared cache only (XMT)

- Shared L1 cache, no private caches
- Prefetch buffers per TCU
- Read-only buffer per cluster, shared by all TCUs within the same cluster.
- Instruction/value broadcasting

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 of parallel Vs. serial in Paraleap



Experiences with Paraleap

- Students in algorithms classes
  - Spring 2007 graduate students
  - Fall 2007 informal course to 12 high-school students
  - Spring 2008 1st year course, non-CS UMD Honors course
  - Spring 2008 undergraduate CS students
  - Spring 2008 23 high scholl students, by self-taught HS teacher

  Total of 15-20 minutes on programming

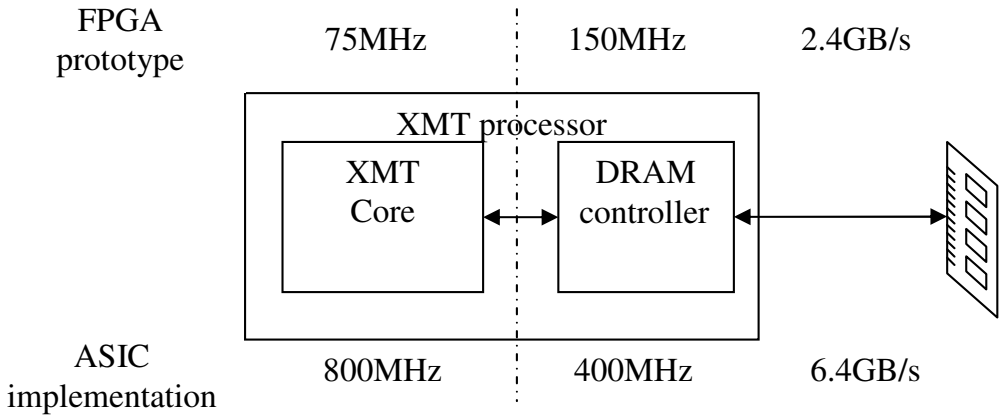
  Compare with: ”To make effective use of multicore hardware today, you need a PhD in computer science”, Chuck Moore, Senior Fellow, AMD.
- Performance monitoring for program fine tunning
  - Performance counters for each TCU, i.e. average read latency
  - Performance counters for each cache modules, i.e. cache hit rate, number of memory accesses
- Development of XMT system tools, i.e. XMTC compiler
- Development of PRAM algorithms
- Test different XMT architecture options

Other XMT Developments

- Hardware
  - Interconnection network
    Studied mesh of trees (MoT) network, Hybrid of MoT and butterfly network.
    Fabricated 8-terminal interconnection network using IBM 90nm process.
  - ASIC prototype
    Taped out XMT ASIC prototype using IBM 90nm process, will be fabricated by end of 2008
- XMTC compiler
  - Completed a basic, yet stable
  - Optimization ubder development: prefetching, read-only buffer, ...
- Applications

Outline

Performance Projection of 800MHz XMT



Assumption

- XMT core operate at 800MHz
- Using DDR2 800 (PC2-6400)

Challenge

- DRAM cannot scale up as XMT core. current 2.4GB/s $\longrightarrow$ DDR2 800 6.4GB/s only 2.67 times faster.
- Some DRAM timing constraints are in absolute time (ns), therefore, the latency in cycle will increase in 800MHz system.

Timing Constraints of DRAM

Symbol	Parameter	time (ns)	cycle in	
			75M	800M
t_{RAS}	ACT to PRE	45	4	36
t_{RCD}	ACT to RD/WR	12.5	1	10
t_{RC}	ACT to ACT(same bk)	55	6	44
t_{RRD}	ACT to ACT(diff. bk)	7.5	1	6
t_{RTP}	RD to PRE	7.5	1	6
t_{WTR}	WR to RD	7.5	1	6
t_{WR}	WR to PRE	15	2	12
t_{RP}	PRE period	12.5	1	10

Slowing Down DRAM

Item	75MHz	75MHz emulating 800MHz	800MHz emulated
Read latency [a] (cycle)	3.5 [b]	24.5	24.5
Maximum DRAM command per cycle	2	0.5	0.5
Peak bandwidth	2.4GB/s 32B/cycle	0.6GB/s 8B/cycle	6.4GB/s 8B/cycle

[a] Latency of DRAM access depends on many factors. We only noted latency (from ACTIVE to the last column of the data) of a read operation, under some DRAM assumptions. For those familiar with DRAMs and DRAM terminology, the assumptions are that the reading is done from a closed bank and there are no activities in other banks.

[b] The DRAM controller operates at 150MHz and one cycle in 150MHz is converted to half a cycle in 75MHz

Validation of the Projection

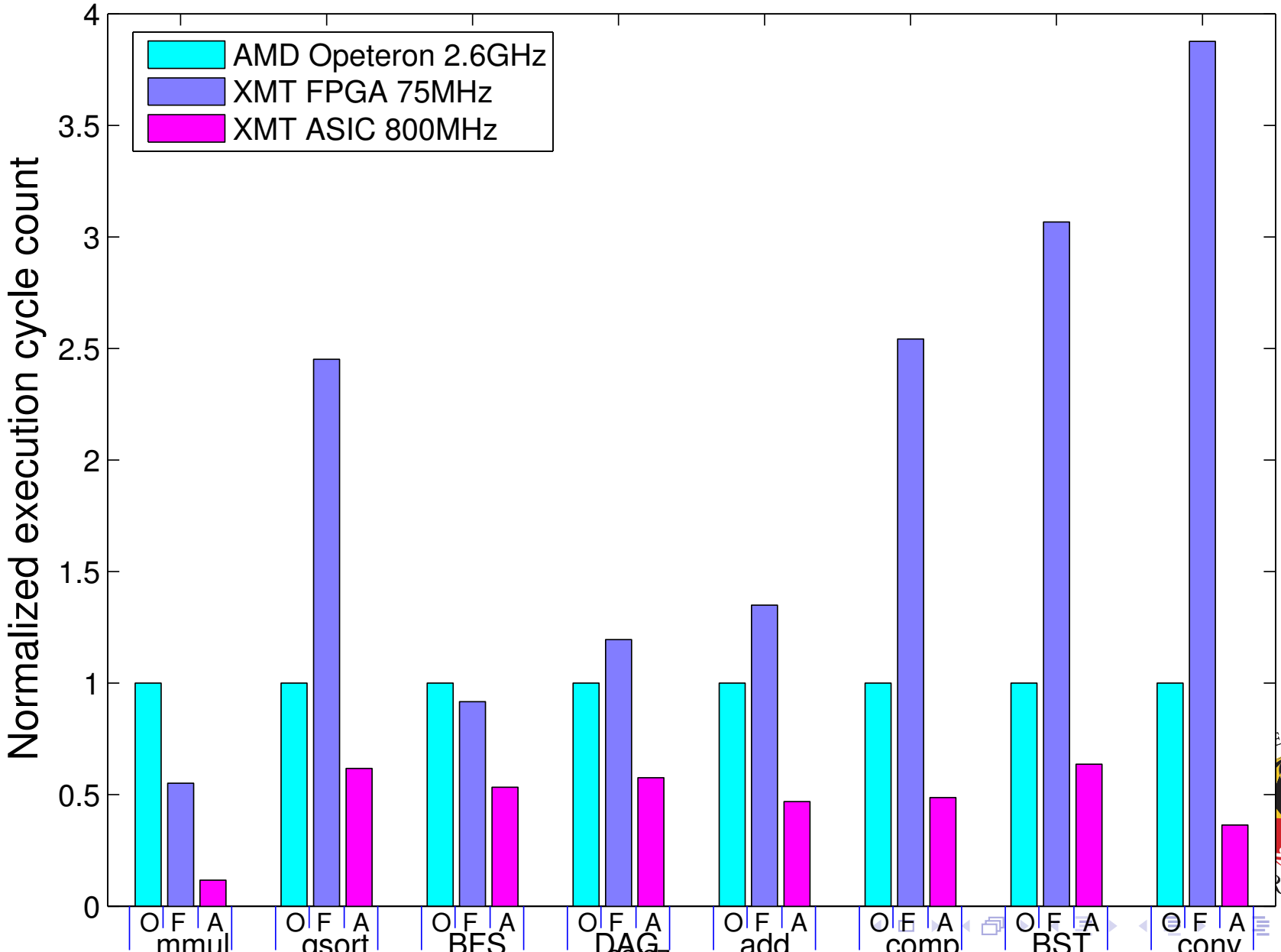
Table: cycle counts (million) in a random memory access program (different sizes)

test	1	2	3	4	5	6
accessed memory	4MB	8MB	16MB	32MB	64MB	128MB
simulated	4.782	10.88	23.23	48.10	97.80	197.5
projected	4.944	11.05	23.68	49.14	100.0	201.9
error	1.3%	1.5%	1.9%	2.2%	2.3%	2.2%

Table: cycle counts (million) in kernel benchmarks

Input	mmul	qsort	BFS	DAG	add	comp	BST	conv
sim.	1.111	2.311	13.03	18.92	2.043	5.410	4.690	5.328
proj.	1,126	2.271	13.20	19.12	1.988	5.470	4.750	5.334
error	1.3%	-1.8%	1.3%	1.1%	-2.7%	1.1%	1.3%	0.11%

Normalized Execution Time - Good and Promissing Performance



Outline

Conclusion - XMT is a Promissing Candidate for the Processor-of-the-Future

- Easy to program – based on PRAM
- Fine-grained parallelism
- Good scalability
- Promissing performance

Future work

- Floating point operations
- Interrupt
- Virtual memory
- Port an OS to the prototype
- Performance improvement (double clock rate for IN etc.)

Thanks

Questions?