A. JAMES CLARK SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Performance Potential of an Easy-to-Program PRAM-On-Chip Prototype Versus State-of-the-Art Processor

UMIACS University of Maryland Institute for Advanced Computer Studies

George C. Caragea, A. Beliz Saybasili, Xingzhi Wen, Alexandros Tzannes. Prof: Uzi Vishkin

Many-Cores are Programmability Limited

~2003 Wall Street traded companies gave up the safety of the only paradigm that worked for them—serial computing based on using a one-processor computer--for parallel computing

The Challenge Reproduce the success of the serial paradigm for many-core computing, where obtaining strong, but not the absolutely best performance is relatively easy.

Positive News Vendors open up to ~40 years of parallel computing. Also to SW that matches vendors' HW (2009 acquisitions). But, did they pick the right part for adoption?

Never Easy-to-program, fast general-purpose parallel computer for single task completion time. Less politically correct Current parallel architectures: never really worked for programmability.

1991: "parallel software crisis"

2003: "as intimidating and time consuming as programming in assembly language"--NSF Blue Ribbon Committee

Why drag the whole field to a recognized disaster area?

The Business Food Chain

SW developers serve the customers. The software spiral (HW improvements ➔ SW improvements ➔ HW …) is broken. The customer will benefit from HW improvements only if SW uses them. If HW developers will not get used to idea of serving SW developers by making HW more programmable, guess what will happen to customers of their HW.

Questions that application SW vendors may ask Develop in 2009 for 2010s many-cores, or wait? Will 2009 code be supported in 2010s? Development-hours in 2009 vs 2010s? Maintenance in 2010s? Performance in 2010s vs programming in 2010s for 2010s HW improvement?

➔Many-cores are programmability limited.

Is there any really good news?

Many-core programming is too constrained.

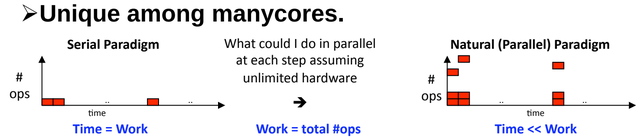
If only, we could "set the programmer free".

XMT Programmer's Model

- Manycores are coming. But 40yrs of parallel computing:
  - Never a successful general-purpose parallel computer (easy to program, good speedups, up&down scalable).
  - IF you could program it → great speedups.
  - XMT: Fix the IF
- XMT: Designed from the ground up to address that for on-chip parallelism
- Unlike matching current HW
- Tested HW & SW prototypes
- Builds on PRAM algorithmics. Only really successful parallel algorithmic theory. Latent, though not widespread, knowledgebase
- **Our work: ~10X speedup XMT Prototype relative to Intel Core 2 Duo**

XMT Programmer's Model

- Design for work-depth:1 operation now. **Any** # ops next time unit
  - **Unique among manycores.**



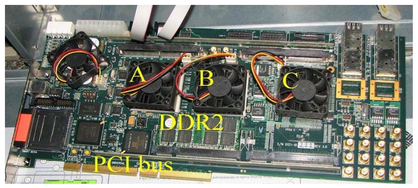
- Reason about correctness & complexity in synchronous model
- Single Program, Multiple Data (SPMD) reduced synchrony
  - Main construct: spawn-join block. Note: can start any number of processes **at once**

  - Threads advance at own speed, not lockstep
  - Prefix-sum (ps). Independence of order semantics (IOS).

Ease of Programming

- Benchmark: can any CS major program your manycore?
  - Necessary condition for success.
- Teachability demonstrated so far for XMT:
  - To freshman class with 11 non-CS students. Some prog. assignments: merge-sort, integer-sort & samples-sort.
  - Magnet HS teacher. Downloaded simulator, assignments, class notes, from XMT page. Self-taught. Recommends: Teach XMT first. Easiest to set up (simulator), program, analyze: ability to anticipate performance (as in serial). Not just for embarrassingly parallel. Teaches also OpenMP, MPI, CUDA.
  - High school & Middle School (some 10 year olds) students from underrepresented groups by HS Math teacher.

Paraleap: XMT PRAM-on-chip silicon



Clock rate	75 MHz
DRAM size	1GB
DRAM channels	1
Mem. data rate	0.6GB/s
No. cores (TCUs)	64
Clusters	8
Cache modules	8
Shared cache	256KB

- Built FPGA prototype
- Announced in SPAA'07
- Built using 3 FPGA chips
  - 2 Virtex-4 LX200
  - 1 Virtex-4 FX100

X. Wen completed synthesizable Verilog description AND the new FPGA-based XMT computer in slightly more than two years.

Software Release

- Allows to use **your own computer** for programming on an XMT environment and experimenting with it, including:
  - Cycle-accurate simulator of the XMT machine
  - Compiler from XMTC to that machine
- Extensive material for teaching or self-studying parallelism, including
  - Tutorial + manual for XMTC (150 pages)
  - Classnotes on parallel algorithms (100 pages)
  - Video recording of 9/15/07 HS tutorial (300 minutes)
  - Video recording of grad Parallel Algorithms lectures (30+hours)
- Download: www.umiacs.umd.edu/users/vishkin/XMT/sw-release.html

Benchmarks

- Sparse Matrix – Vector Multiplication (SpMV)
  - Matrix stored in Compact Sparse Row (CSR) format
  - Serial version: iterate through rows
  - Parallel version: one thread per row
- 1-D FFT
  - Fixed-point arithmetic implementation
  - Serial version: Radix-2 Butterfly Algorithm
  - Parallel version: Parallelized bit traversal and parallelized butterfly stages
- Quicksort
  - Serial version: standard textbook implementation
  - Parallel version: two phases
    - Phase 1: For large sub-arrays, parallelize partitioning operation using atomic prefix-sum
    - Phase 2: Process all partitions in parallel using serial partitioning algorithm

XMT vs. Intel Clock-Cycle Speedup

- Large dataset: realistic input sizes, suggested by Intel engineer
- Small dataset: fits in both CPUs caches, more fair comparison

	Core 2 – ICC		Core 2 - GCC	
Program	small	large	small	large
SpMV	6.7x	3.3x	6.3x	3.26x
FFT	9.51x	2.51x	8.76x	2.71x
Quicksort	13.07x	7.75x	13.89x	8.18x

- Paraleap outperforms Intel Core 2 on all benchmarks
- Lower speed-ups for Large dataset because of smaller cache size
  - Will not be an issue for future implementations of XMT
- Silicon area of 64-TCU XMT roughly the **same as one core** of Intel Core 2 Duo