Impact of CMP Design on High-Performance Embedded Computing

Patrick Crowley
pcrowley@wustl.edu

Joint with M. Franklin, J. Buhler, R. Chamberlain
HPEC 2006
MIT Lincoln Laboratory
Overview

- For the next several years, number of processor cores per chip will likely double

- Strong implications for HPEC

- Illustrate some aspects with our work in
  » Processor design
  » Application development
Intel IXP Network Processor

- Chip multiprocessor for networking @ 1.4GHz

- Multiple threads per Micro-Engine (ME)

- Special Instructions
  - Bit operations
  - Rings
  - CRC calculations

- XScale control processor

- Special Functional Units
  - e.g. Hash Unit, Cryptography unit
MicroEngines

- Small, simple processors

- No caches

- Multiple Thread Contexts

- Registers + Local Memory
  » 256 GPRs
  » Next Neighbor, Transfer

- Signals for inter-thread communication and asynchronous I/O ops
MEv2 Block Diagram

- Prev Neighbor 8 thds*16
- GPRs 8 thds*32
- Transfer IN Regs 8 thds*32
- Local Memory
- Pseudo-Random
- CRC
- Timers
- 32-b ALU
- To registers, local mem, and units
- Command Outlet
- Transfer OUT Regs 8 thds*32
- Next Neighbor
- Control Store
Impact on Processor Design

- Heterogeneity amongst processor cores
  - We created a CMP model with two core types, one simple and one sophisticated
  - Each implemented the same ISA
  - On a batch SPEC2000 workload
    - Heterogeneity at best yields 80% better performance
    - Clever thread migration can yield a further 40% gain

- Novel emphasis on performance/area efficiency
  - We have proposed micro-caches for instruction delivery in cluster CMP processors
  - Explored Icache sizes in Tensilica Xtensa cores between 64 and 256 bytes
  - Can yield 25% increase in performance or area
HMMER v.2.3.2 is a suite of tools from Washington U. Medical School for protein motif search.

Core is the Viterbi algorithm:
- Use an input sequence to traverse an HMM representation of a protein or nucleotide.

Outputs score and optimum path through the HMM:
- These help to determine how well the sequence and HMM match.
HMMer
Data Dependencies of HMMer 2.x P7Viterbi() Recurrence

score depends only on $\delta$ (i.e. on transition scores but not on input)

score depends on both $\delta$ and $\sigma$; requires input char $\pi(i)$

dependency is present for all $k$ in 1..m
Jack HMMer

- Implement the Viterbi algorithm in an Intel IXP 2850
- Runs multiple viterbi calculations in parallel

Why a Network Processor?

» Available commercial CMP
» Amenable to dense computing
» Substantial memory bandwidth
HMMer on the P4

- We optimized the P4 version of HMMer
  - hand optimized the x86 assembly code to provide a base case to test against.

- Our P4 version achieved 2x speedup

- Experiments run on a 2.6 GHz P4 with HT
  - Two concurrent HMMer threads
Characterization

- Trace cache misses < 1%
- L1 data cache misses ~ 7%
- L2 data cache misses < 1%
- Branch mispredictions negligible
- Dependencies within and between loop iterations limit ILP
Relative Performance

Runtime vs. Sequence Length

- Original P4
- Optimized P4
- IXP 2850

Seconds vs. Sequence Length
Optimizations (1)

- Loop unrolling
  » Inner loop unrolled twice
  » Makes writing to DRAM easier (word boundary issue)

- Pipelined asynchronous reads
  » Issue reads for data needed in iteration k+1

- Reorder of HMM in memory
  » Faster/fewer SRAM memory accesses

- Reduce SRAM queue contention by aggregating writes
  » Read/Write data for several iterations at once
  » Use ME local memory as buffer
Problems (1)

- Contention for SRAM
  - Command queue constantly backlogged

- Scales poorly
  - Too many threads contending for limited resources
  - No major speedup gained going from 12 to 16 threads
Multi-core Scalability

IXP 2850 Throughput vs. MEs

Cells/usec

MEs
Local-Memory JackHammer

- Keep all data in local memory
  » Eliminates SRAM queue bottleneck

- Faster and scales better

- Newest chips have sufficient local memory
Performance

Throughput vs. MEs

- Localmem
- Normal w/localmem dataset

Cells/Sec vs. MEs
Future Projections

- P4- expected speedup 5-10x
  - Based on multicores (2-4x), increased clock speed (to 4Ghz), x86-64 (10%), increased threading (38%)

- IXP- 10x
  - More Local Memory (2.26x), More MEs (2x), Clock to 3 Ghz (2.14x)
Conclusion

- Multi-core trend will have impact on HPEC

- Consequences will span
  - Application development
  - Processor and system design
  - Design goals

- Consequences of parallel programming will keep us all busy