

JackHMMer: Exploiting Coarse Grained Parallelism to Accelerate Protein Motif Finding with a Network Processor

Ben Wun, Jeremy Buhler, Patrick Crowley
Washington University

2005 Intel IXA University Summit

Introduction

- Network Processors
 - CMPs designed for network processing tasks
 - CMPs adopted in mainstream processors
 - Are they good for other tasks?
- Scientific computing apps that:
 - Display high level of thread parallelism
 - Have predictable access memory patterns
 - Take a really long time to run

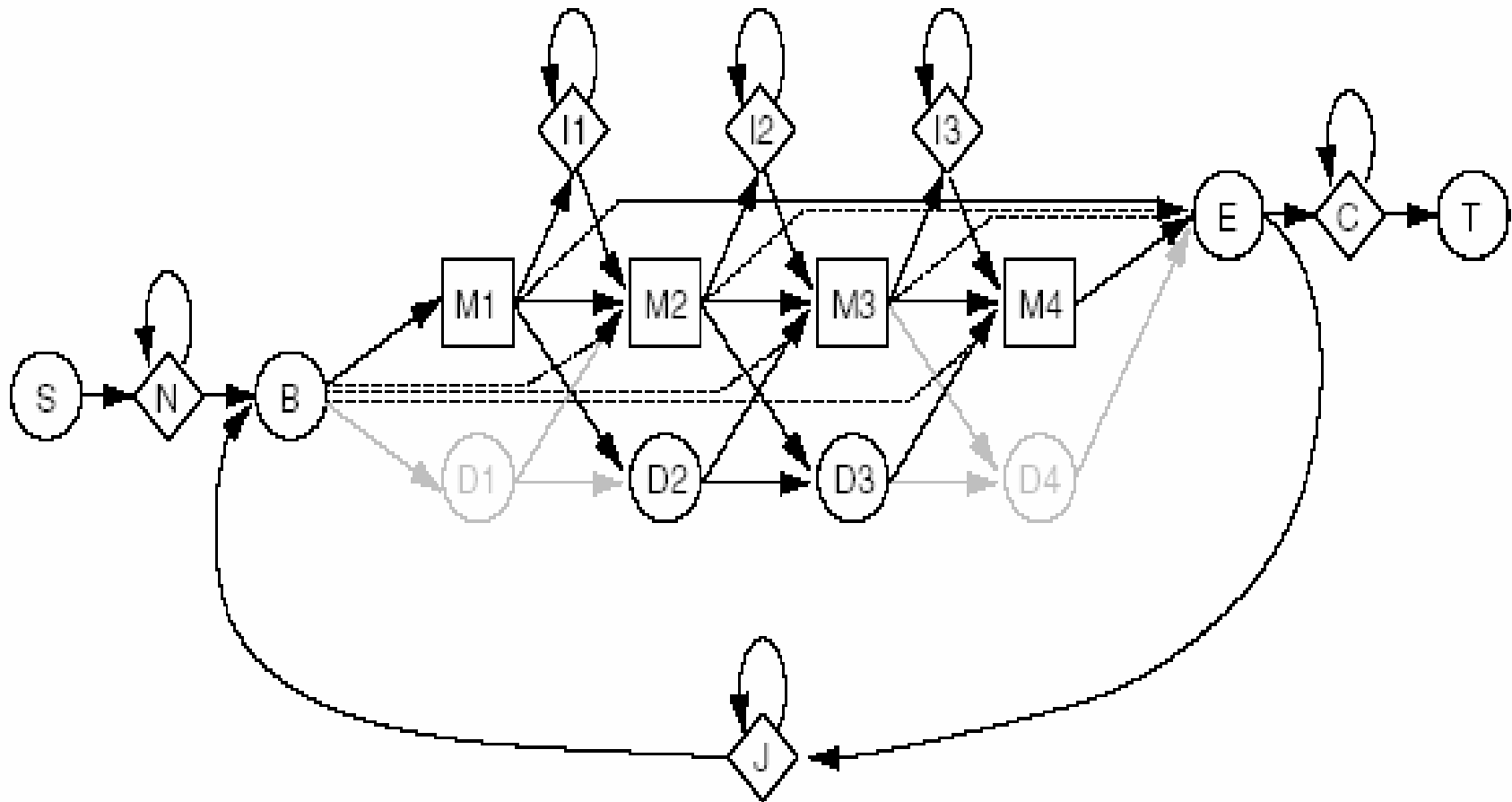
Outline

- HMMer
- Intel IXP
- HMMer on the P4
- JackHMMer

Motif Example

| Protein | Motif Sequence |
|------------|---|
| RA25_SCHPO | RENSVYLAKLAEQAERYEEMVENMKKVACSND . . KLSVE |
| BMH1_YEAST | REDSVYLAKLAEQAERYEEMVENMKTVAS SGQ . . ELSVE |
| 1434_LYCES | REENVYLAKLAEQAERYEEM IEFMEKVAKTADVEELTVE |
| 143T_HUMAN | KTEL I QKAKLAEQAERYDDMATCMKAVTEQGA . . ELSNE |
| 1433_XENLA | AKLSEQAERYDDMAASMKAVTELGA . . ELSNE |

HMM



HMMer

- Main tool used by biologists to identify motifs in protein sequences
- Compares protein to database of motifs (represented as HMMs) to identify function
- Outputs a score of an optimum path through the HMM
 - Score is how well protein matches HMM

Viterbi Algorithm

- Dynamic programming
 - Finds the optimal parse of the sequence through the HMM.
- Well known algorithm with many uses
 - Speech recognition
 - Image processing
 - Time series analysis of data
- Implemented as a doubly nested loop
 - Outer loop over sequence
 - Inner loop over models states

Related Work

- TimeLogic DeCypher engine
 - FPGAs with a Sun Sparc host
 - Great performance
 - Loss of accuracy
- Bos and Huang- sequence to sequence comparison in an IXP 1200
 - Achieved parity with 1.8 GHz P4

Outline

- HMMer
- *Intel IXP*
- HMMer on the P4
- JackHMMer

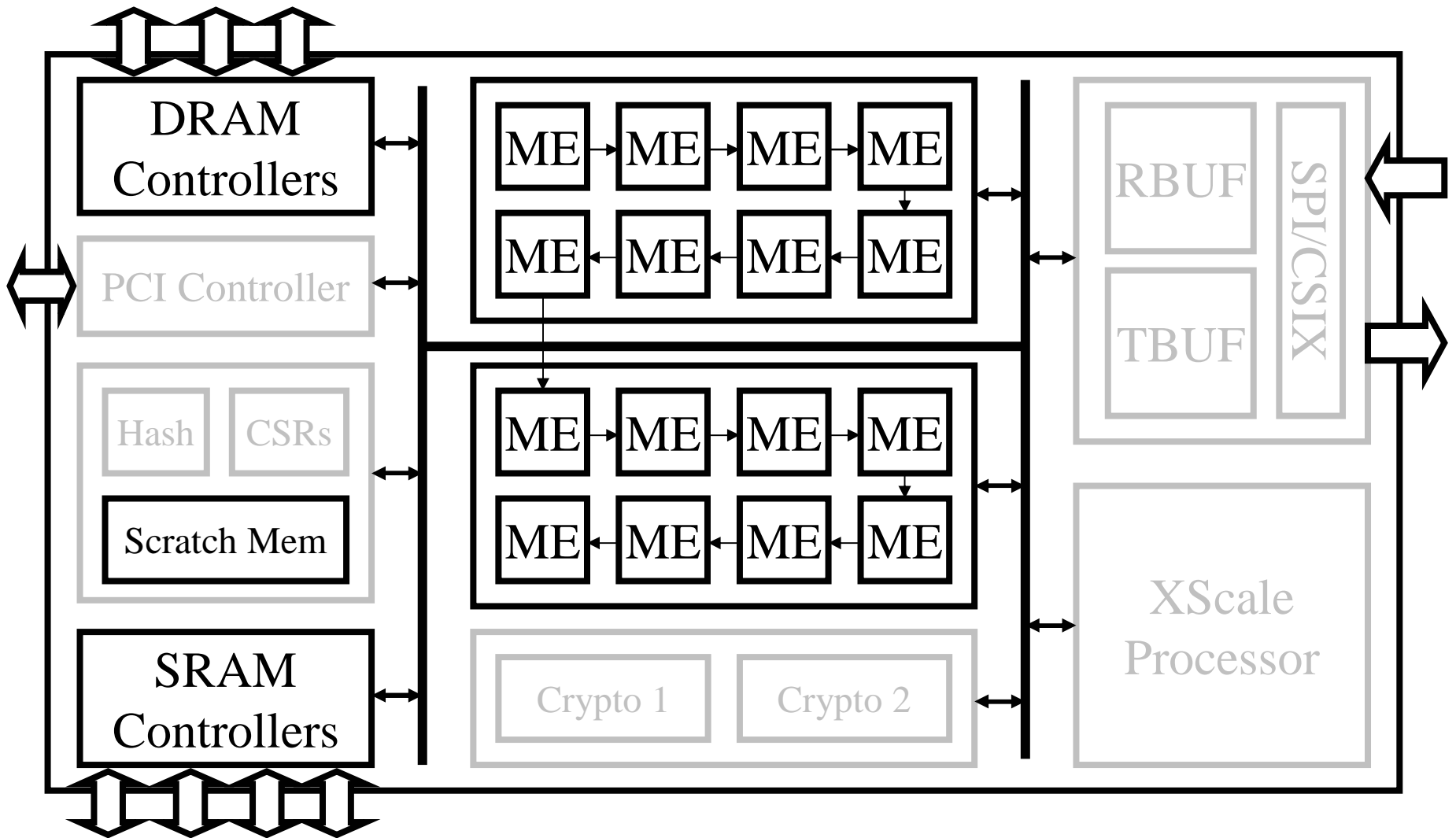
Intel IXP Network Processor

- Chip Multiprocessor for high-speed networking
- Multiple MicroEngines operate in parallel
 - 16 MEs on the 2850
- Heterogeneous memory space
- Special Functional Units

Why Network Processors?

- Chip multiprocessor affords a high amount of parallelism relatively cheaply
- IXP's ISA gives programmer more explicit control
- HMMer contains a lot of coarse grained parallelism and predictable memory access patterns

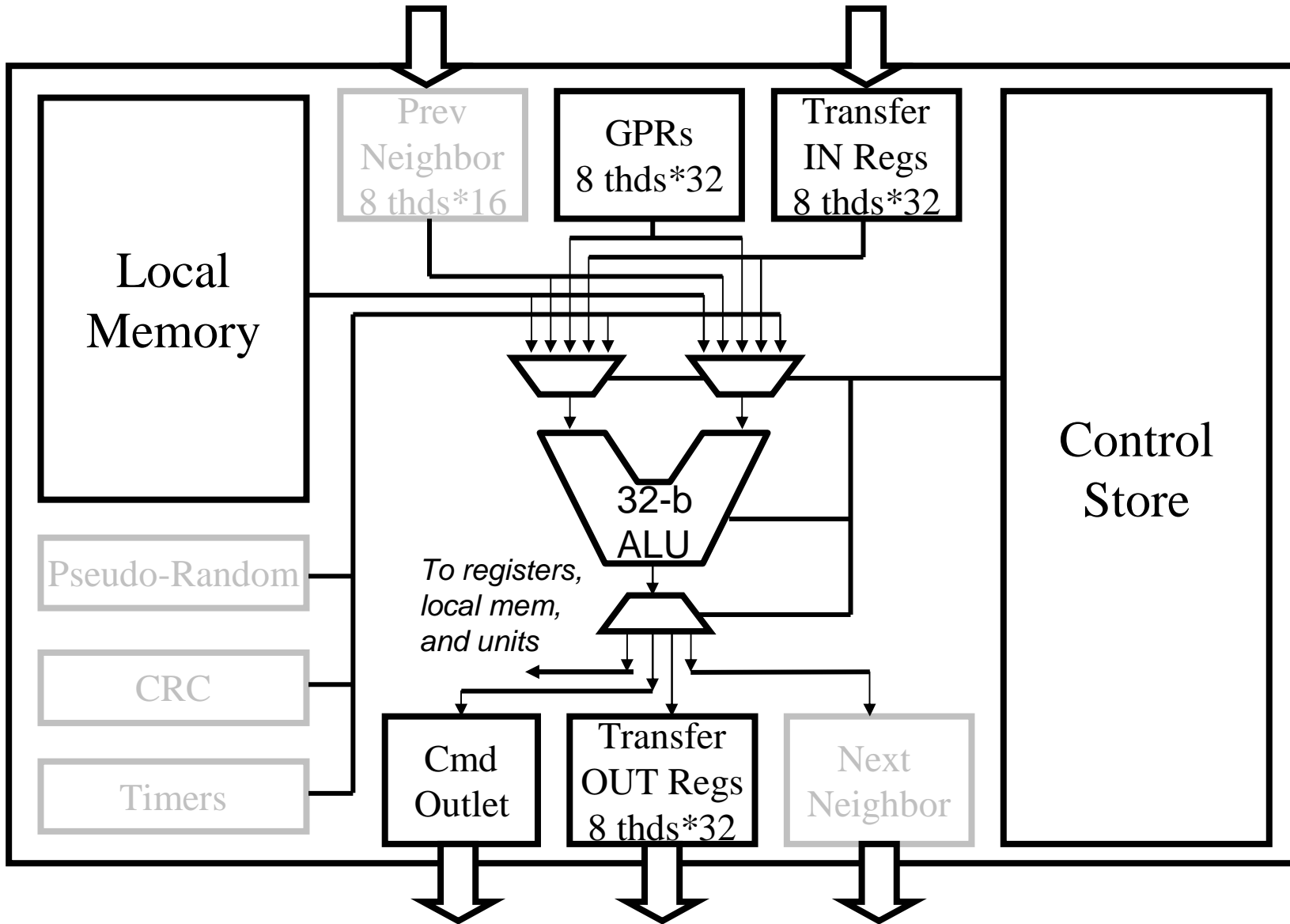
IXP 2850 Block Diagram



MicroEngines

- Small, simple processors
- Local memory, No cache
- Signals for inter-thread communication and asynchronous I/O ops
- 256 GPRs

MEv2 Block Diagram



Difference to Programmer

- Asynchronous I/O operations
 - Ex:
dram[read, \$buffer, src, 0, 5], sig_done[read_sig]
...
ctx_arb[read_sig]
- Heterogeneous memory interface
- Most coding in assembly, not HLL
 - Register Allocator

Outline

- HMMer
- Intel IXP
- *HMMer on the P4*
- JackHMMer

HMMer on the P4

- We optimized the P4 version of HMMer, including hand editing the x86 assembly code to provide a base case to test against.
- Achieved 2x speedup
- Tests run on a 2.6 GHz P4 with HT
 - Cut database in half and used 2 threads, 1 for each half

Performance Characterization

- Vtune shows that:
 - Trace cache misses < 1%
 - L1 data cache misses ~ 7%
 - L2 data cache misses < 1%
 - Branch mispredictions negligible
- Limited ILP hinders performance
 - Dependencies within and between loop iterations limit ILP
 - Vtune shows only 2.16 IPC
- Simplescalar simulation with very large instruction window (128) still only achieves 3.6 IPC

Outline

- HMMer
- Intel IXP
- HMMer on the P4
- *JackHMMer*

JackHMMer

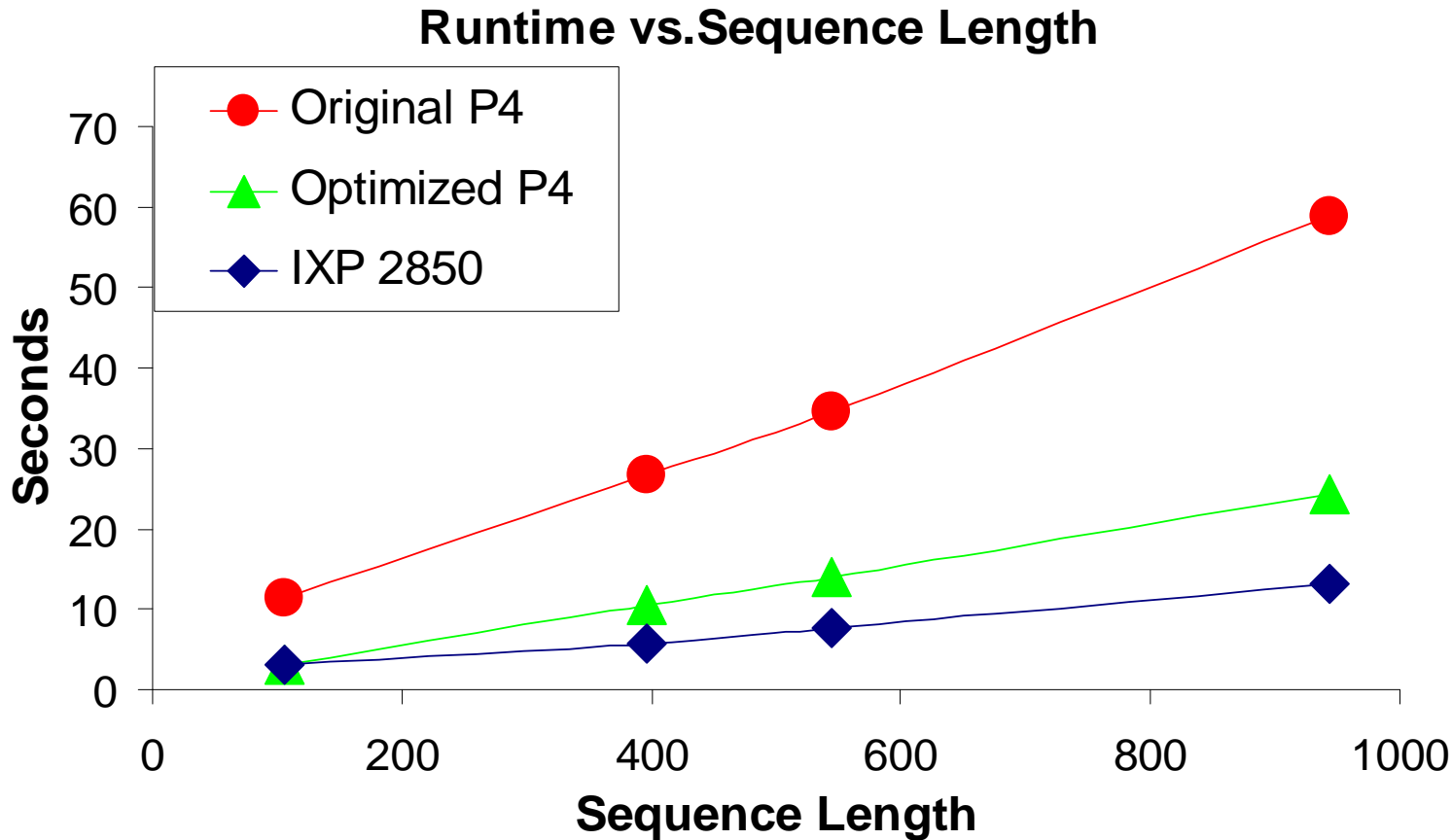
- Implements Viterbi algorithm in IXP 2850
- Runs multiple Viterbi calculations in parallel
 - Each ME compares a different motif model to common sequence

JackHMMer Implementation

- Pipelined asynchronous reads
 - Issue reads for data needed in iteration $k+1$
- Loop unrolling
 - Inner loop unrolled twice
 - Makes writing to DRAM easier (word boundary issue)
- 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

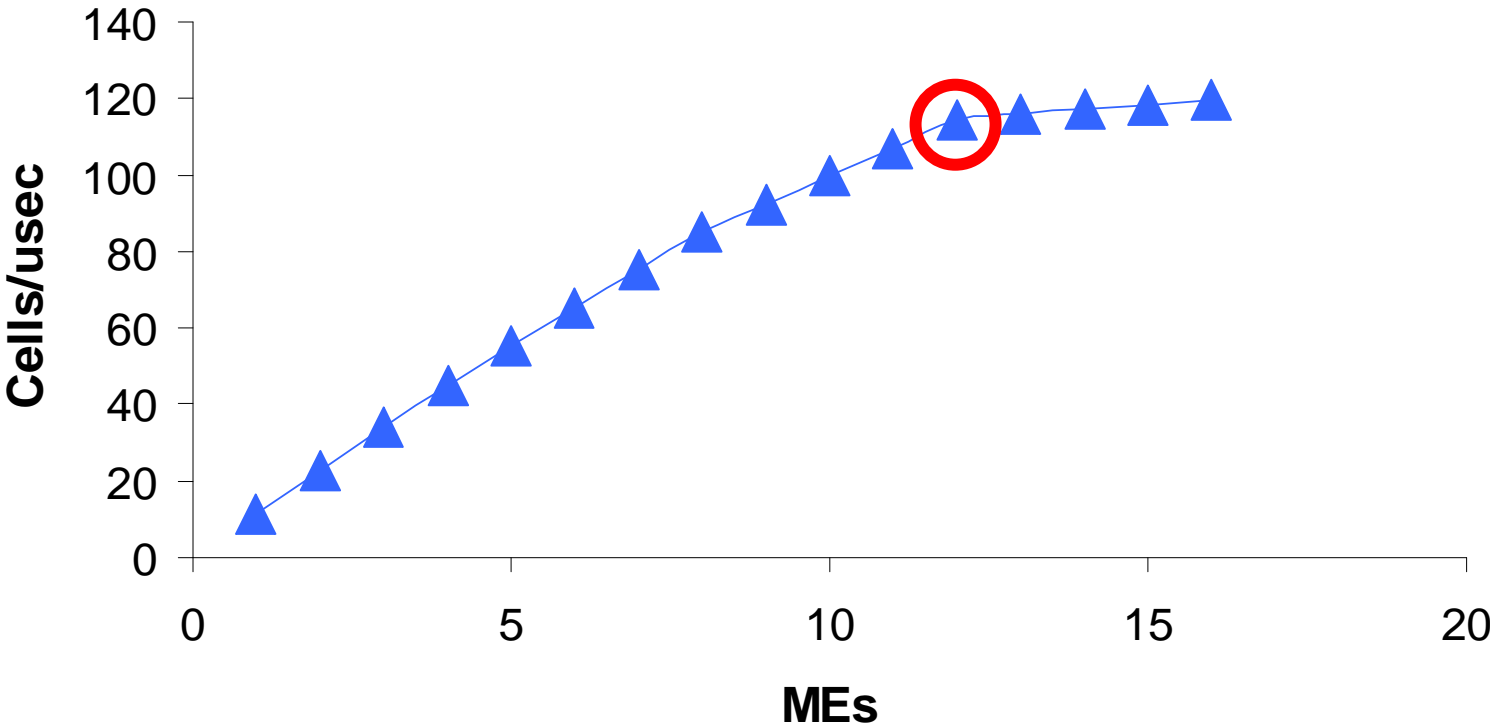
1.82x speedup over P4

Performance Comparison



Scalability

IXP 2850 Throughput vs. MEs

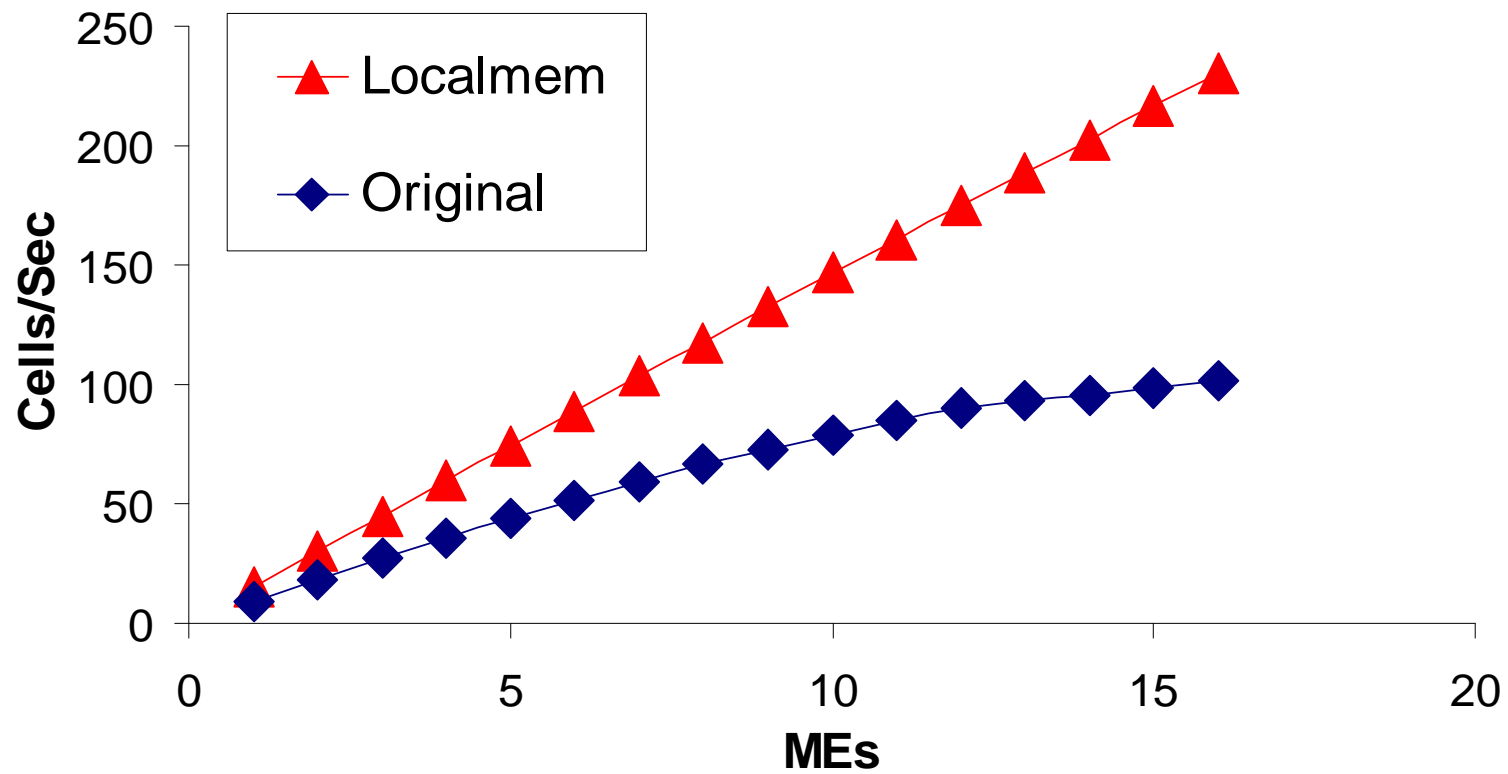


Solution: Local Memory JackHMMer

- Keep all data in local memory
 - Eliminates SRAM queue bottleneck
- Faster and scales better
- Proof of concept
 - Not enough memory *in this generation of IXP* to function over all HMMs in database

Benefits of Local Memory

Throughput vs. MEs



Relative Speedup

- Full utilization, IXP beats P4 by 3x
- Memory bottleneck reduces to 1.82x
- Local Memory increases to 3.5x
 - P4 not fully utilized

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)
- We expect JackHMMer to maintain its advantage
 - Supporting details and justification in paper

Conclusion

- Network processors can accelerate important classes of scientific applications
- HMMer/Viterbi is suited to the IXP
 - Significant, coarse-grained parallelism
 - Explicit, application-controlled mechanisms
 - Minor architectural changes could make it even better

Future Work

- Unroll J-state loopback
 - Lets us invert Viterbi loop order, reduces memory usage
- Different architecture
 - Caches instead of asynchronous memory access
- Other applications with similar structure
 - Speech processing, image processing, etc.