Perspectives on the Memory Wall

"It's the Memory, Stupid!"
– Richard Sites

Salishan 2003
High-Speed Computing Conference

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering

CORNELL

First, My Philosophy…

Use existing resources more wisely

Add minimal hardware support, isolate complexity

Modify software (OS/compiler/libs/apps) to exploit that hardware

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

The Game Plan

What's the problem?

- Numbers
- Pictures
- Details, details

What are we going to do about it?

- Good news
- Bad news
- Silver Bullet?

(Selective, Subjective) Chronology

Summary

1994 Wulf, McKee

Hitting the Memory Wall: Implications of the Obvious

"... although the disparity between processor and memory speed is already an issue, downstream someplace it will be a much bigger one. How big and how soon?"

1995 McCalpin

Memory Bandwidth and Machine Balance in Current High Performance Computers

"steps need to be taken soon to **increase the memory bandwidth** more rapidly ... merely increasing bandwidth and decreasing latency will not be adequate"

1996 Sites

It's the Memory, Stupid!

"... over the coming decade, **memory system design will be the only important design issue** for microprocessors"

1996 Burger, Goodman, Kägi (ISCA)

Memory Bandwidth Limitations of Future Microprocessors

"... future microprocessors ... **limited in performance by memory bandwidth**"

Plus ça change ...

Problem is NOT the one most people's ...

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

The Memory Wall

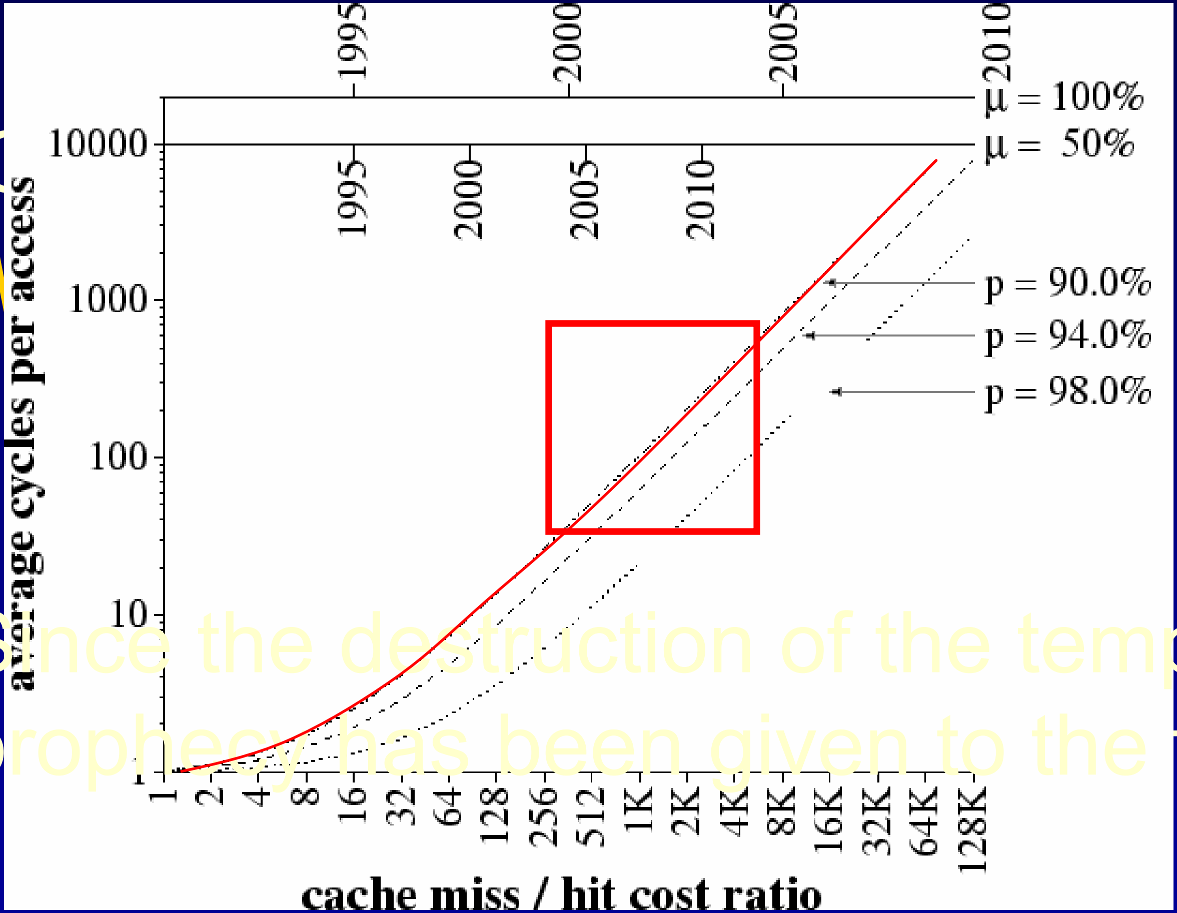
Made simplifying assumptions

- $t_{avg} = p \times t_{cache} + (1-p) \times t_{memory}$
- Every 5th instruction references memory
- CPU speeds increase 50-100% / year
- DRAM speeds increase 7% / year

How long before **ALWAYS** waiting for memory?

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

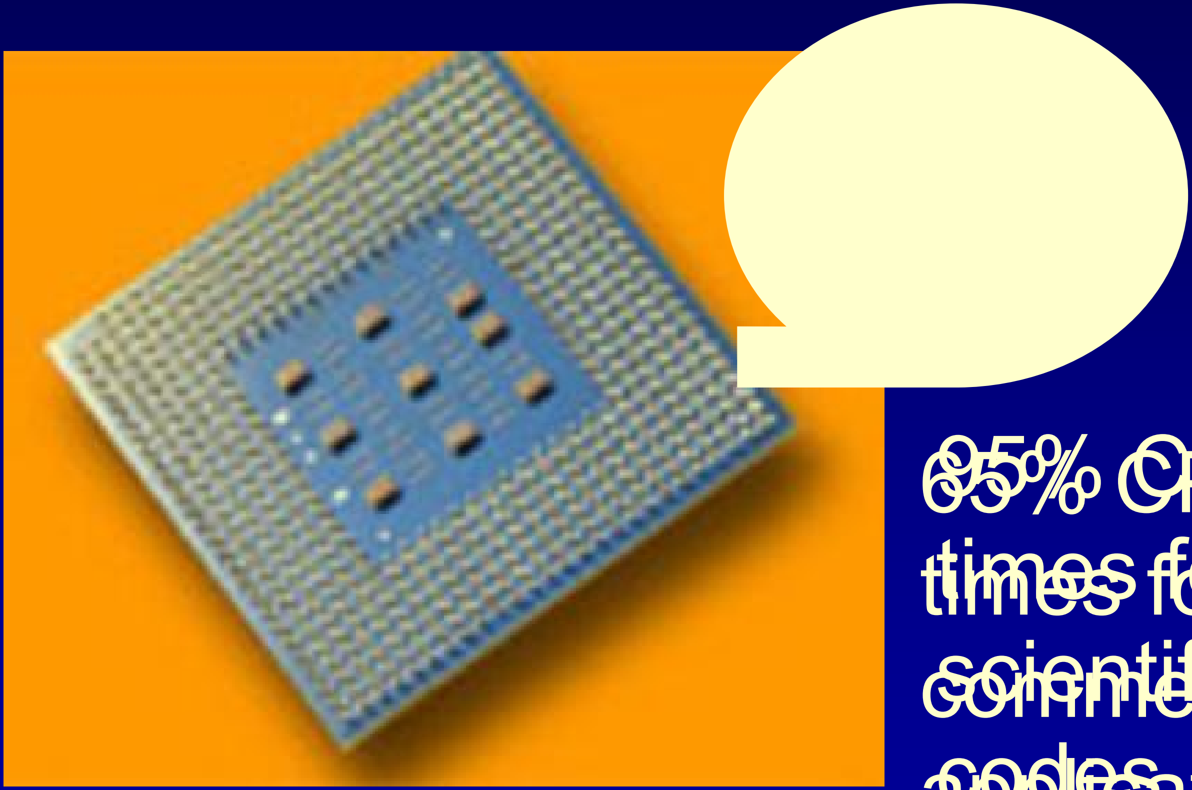
The Original Prediction



Since the destruction of the temple, prophecy has been given to the fools

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

A Picture's Worth



95% CPU idle times for HPC scientific codes

65% CPU idle times for commercial TP applications

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Why?

Lack of reference locality

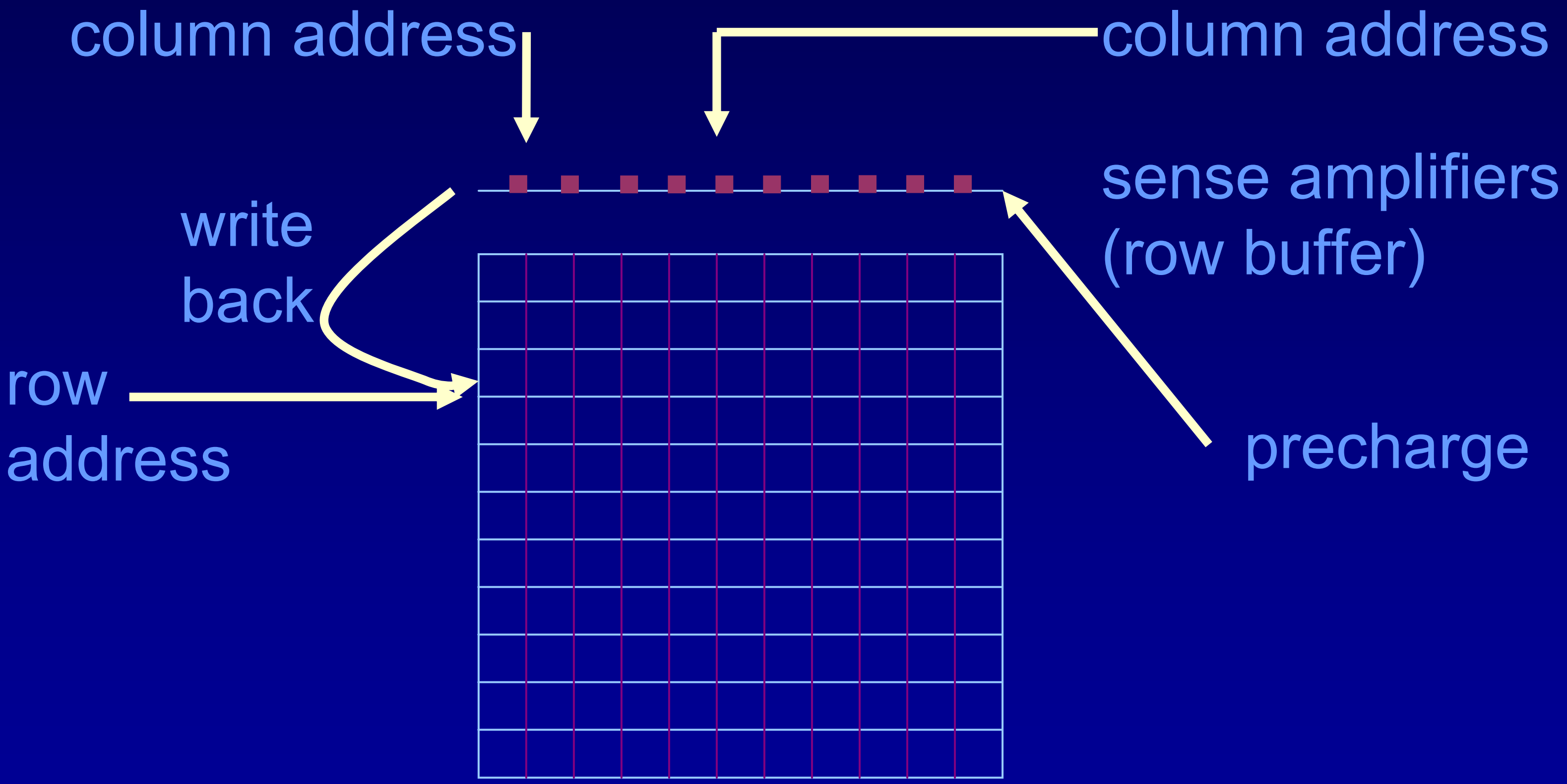
- Registers
- Cache lines (∀ caches)
- TLB entries (btw, TLB == cache)
- VM pages (yup, VM == cache)
- DRAM pages (caching here, too)

Contention for resources

almost dual of locality optimizations

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Non-uniform DRAM Access



Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Possible Approaches

Use bigger, deeper cache hierarchies

Add more/better latency-tolerating features

- Non-blocking caches
- Out-of-order instruction pipelines
- More speculation
- Multithreading

Migrate intelligence $\leftrightarrow$ DRAMs

Isolates complexity within one component $\swarrow$

Create smarter memory subsystems

Make software control how cache is managed

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Smarter How?

I know how to build an efficient, cost-effective memory subsystem that does scatter/gather REALLY well, makes good use of DRAM resources, and makes better use of on-chip cache resources

Overlap CPU and memory activity

- Prefetch read data
- Buffer write data

Remap addresses

- Use cache capacity better
- Use bus bandwidth wisely

Schedule backend (DRAM) accesses better

- Optimize use of memory bus
- Exploit parallelism among DRAM banks
- Exploit locality in page buffers (hot rows)

Won’t slow down normal accesses

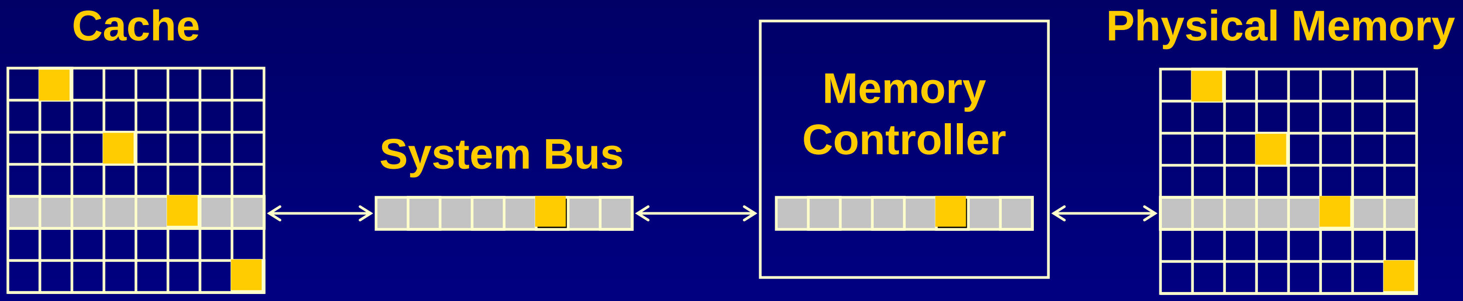
Won’t change CPU

Will perform like SRAM

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Motivating Example

```
for (i = 0; i < n; i++)
    sum += A[i][i];
```



Wasted bus bandwidth

Low cache utilization

Low cache hit rate

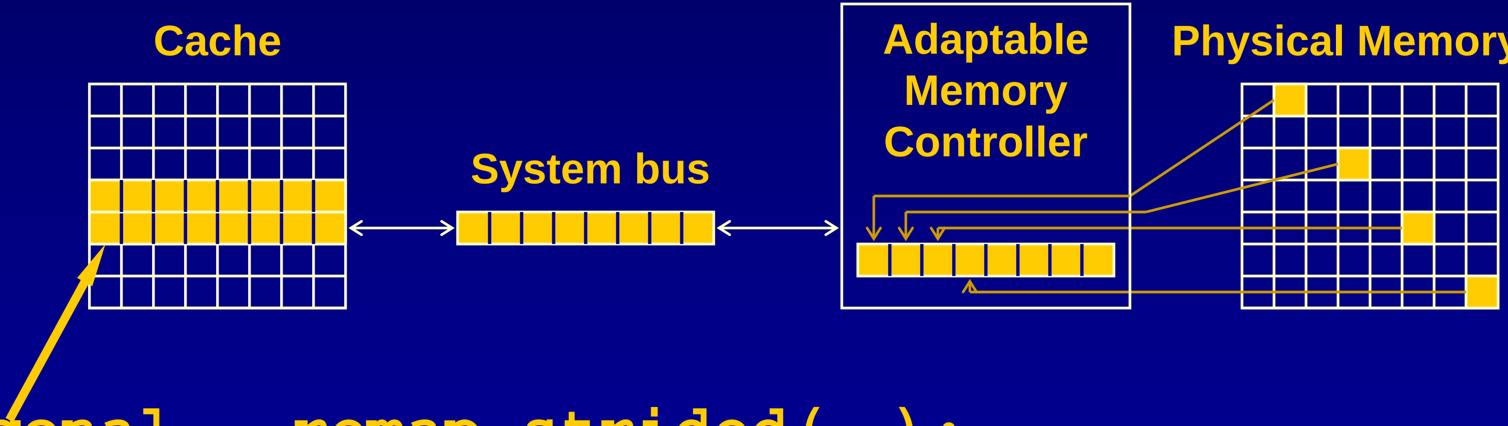
Low TLB hit rate

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Gathering within the Impulse MC

Load only data needed by processor

Gather sparse data to dense cache lines



```
diagonal = remap_strided(...);
for (i = 0; i < n; i++)
    sum += diagonal[i];
```

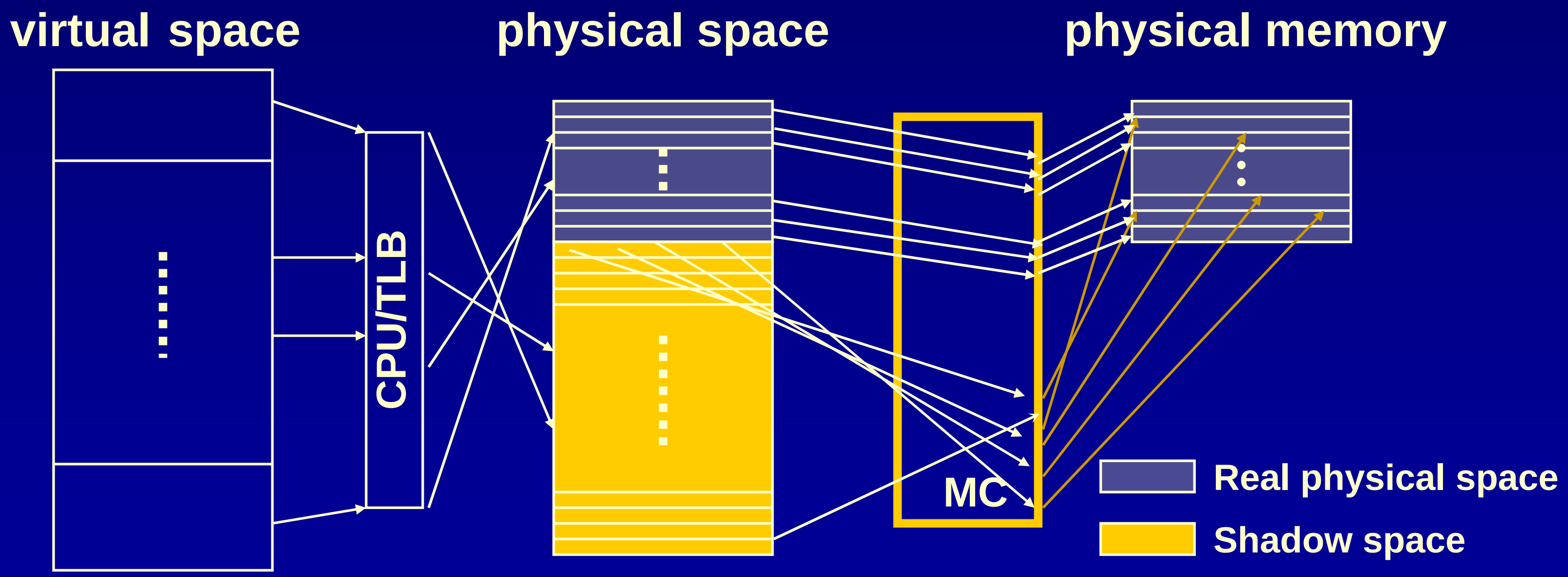
20x speedup for 4K×4K array

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Impulse Remapping

Exploit unused physical (shadow) addresses

Remap at fine or coarse granularity



Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Indirection Vector Remapping

```
                                    aA = remap_indirect(..)
for (i=0; i<n; i++)                 for (i=0; i<n; i++)
   ...A[iv[i]]...;                     ...aA[i]...;
```

Memory controller maps aA[i] $\Rightarrow$ A[iv[i]]

Indirect accesses replaced by sequential

Accesses to iv moved to MC

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Dynamic Indirection Vectors

Don't know entire iv ahead?

```
for (i=0; i<N; i++)
   sum += A[random()];
```

Stripmine loop:

```
aA = remap_DIV(A, &iv, 32, …);
for (i=0; i<N/32; i++) {
   for (k=0; k<32; k++)
      iv[k] = random();
   flush_to_MC(iv);
   for (k=0; k<32; k++)
      sum += aA[k];
   purge_from_cache(aA);
}
```

} Analogous to get/put

The Impulse "Big Picture"

Improve memory locality via remapping

- Improve system bus utilization
- Increase cache efficiency

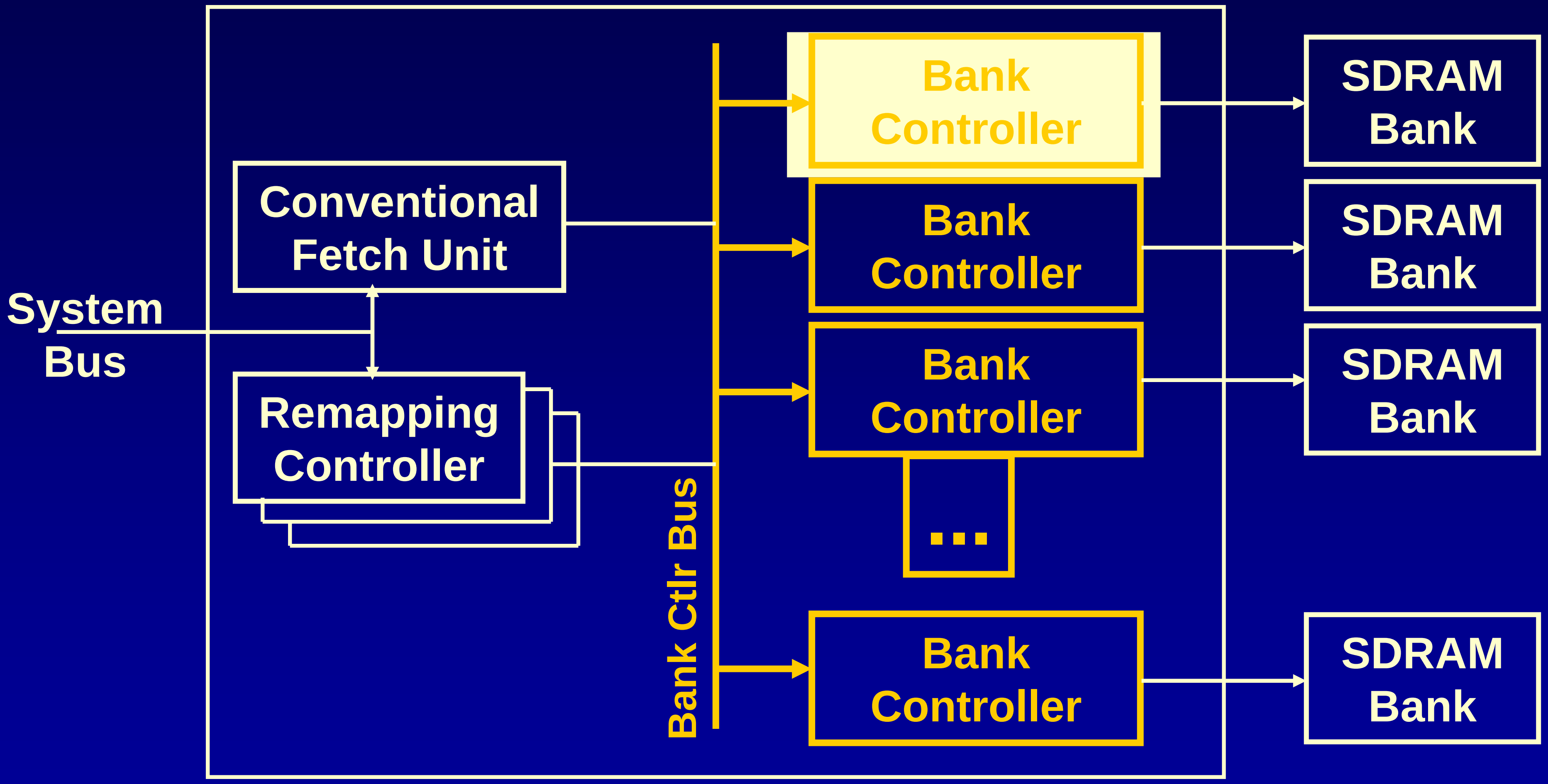
Increase throughput with parallelism

- Overlap CPU/memory activity
- Exploit parallel SDRAM banks

Exploit SDRAM's NUMA characteristics

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Conceptual Organization



Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Parallel Vector Access Backend

Remapping controllers issue special vector ops

- Base-stride: issue (first address, stride, length) tuple
- Vector-indirect: issue four indices per cycle (tentative design)

Bank controllers make independent decisions

- Am I involved in this vector read?
- What elements must I fetch?
- How can I fetch them most efficiently?

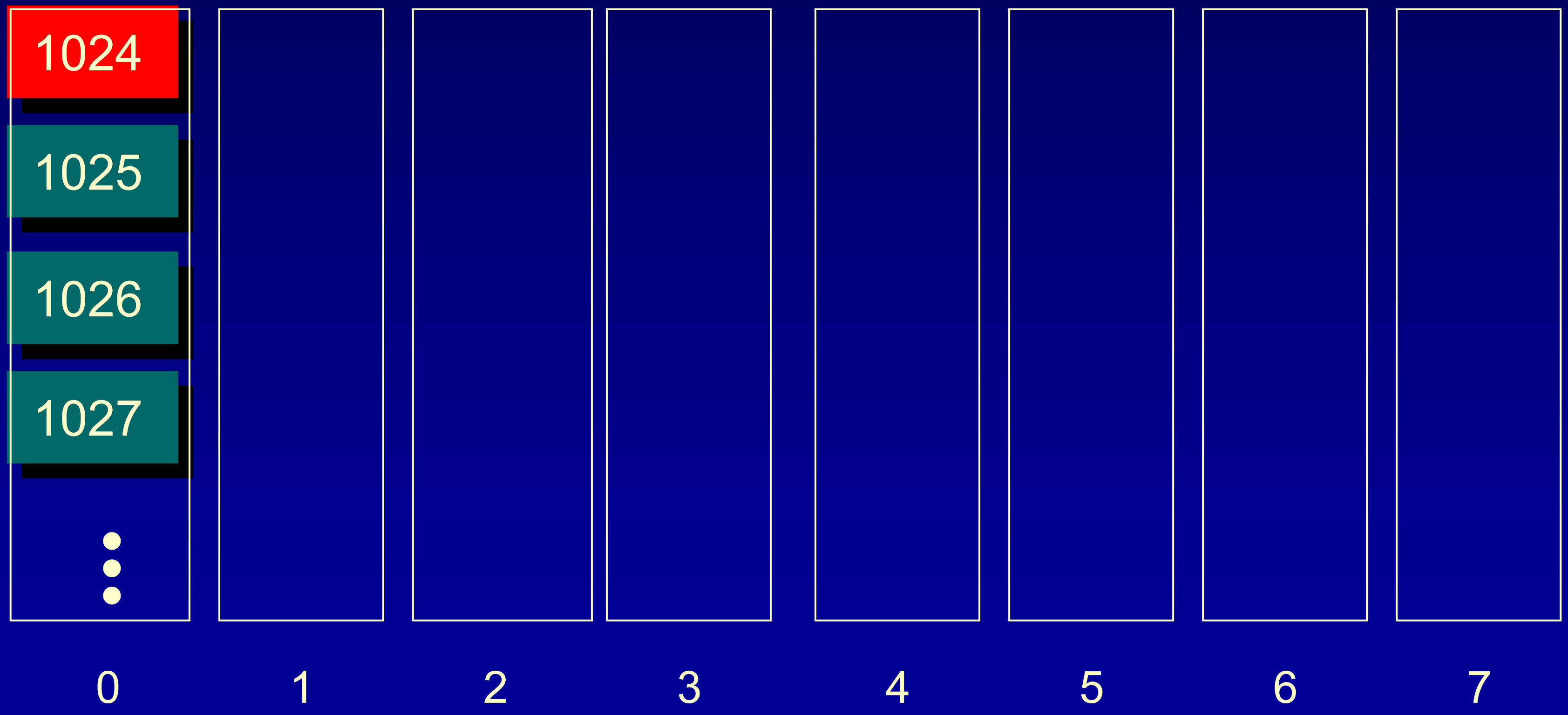
When all elements fetched on a read …

- Control lines indicate completion of vector read
- Coalescing done via wired-OR operations
- Bank controller bus speed matches system bus

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Cache-line Interleaved, Serial Vector Gathers

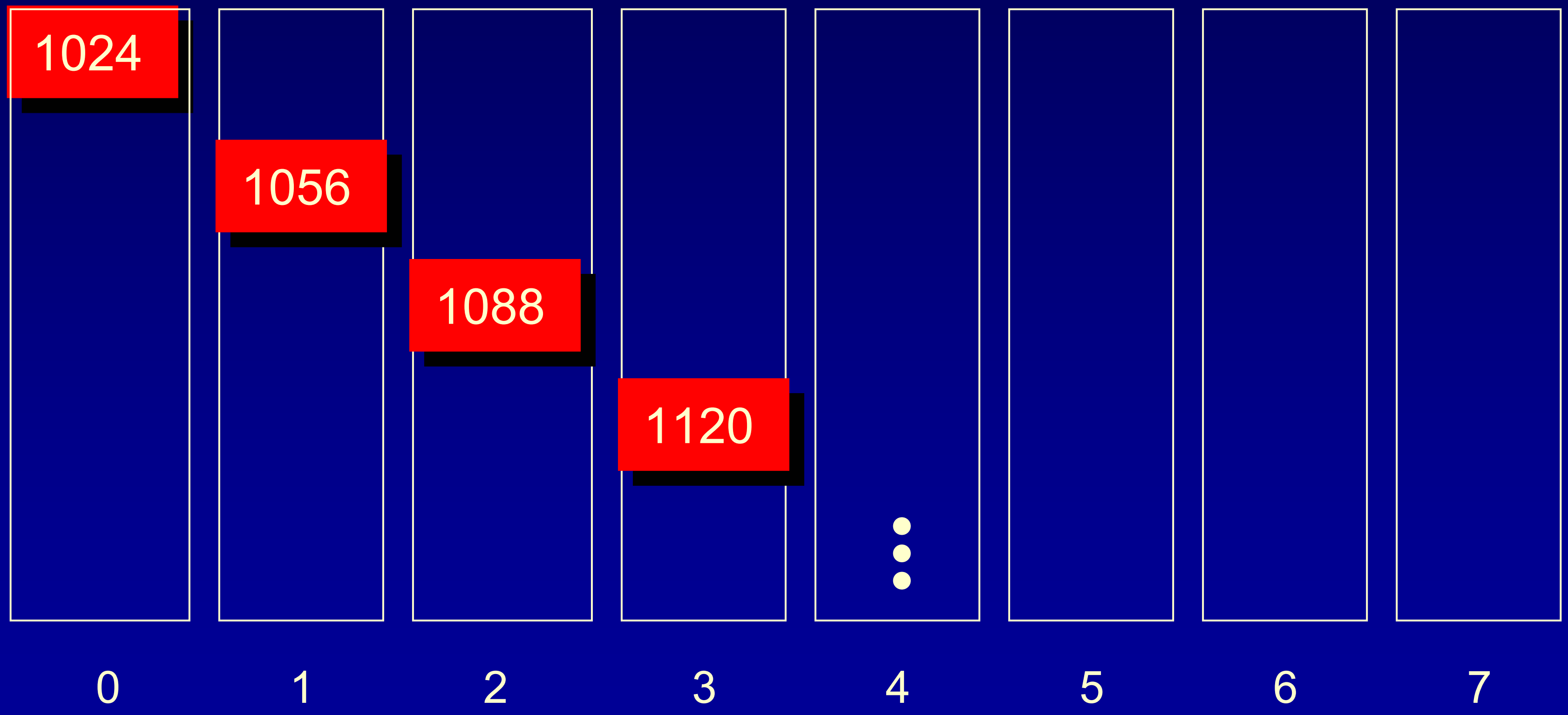
V = < 1024, 1, 16 > (same as cache-line fill)



Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Cache-line Interleaved, Serial Strided Vector Gathers

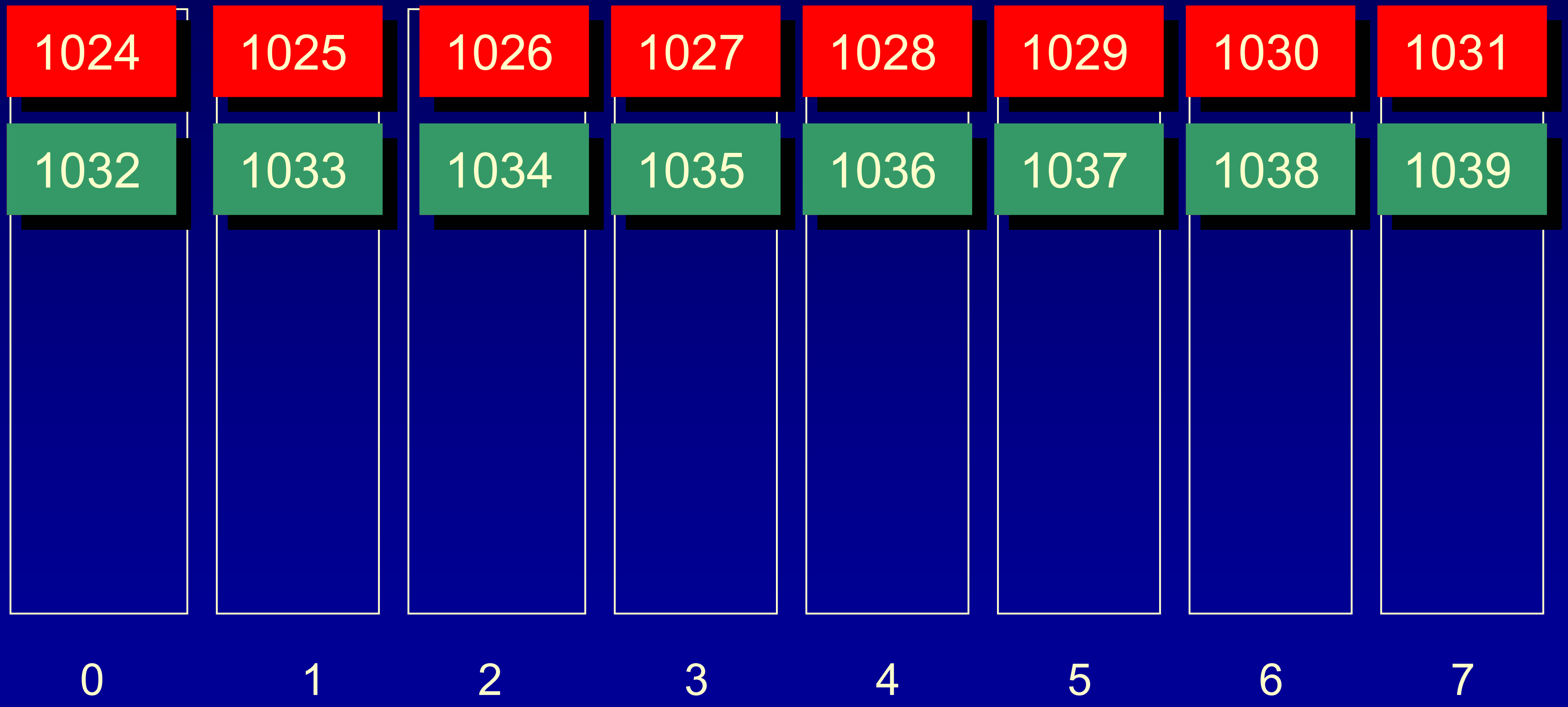
V = < 1024, 32, 16 > (vector gather)



Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

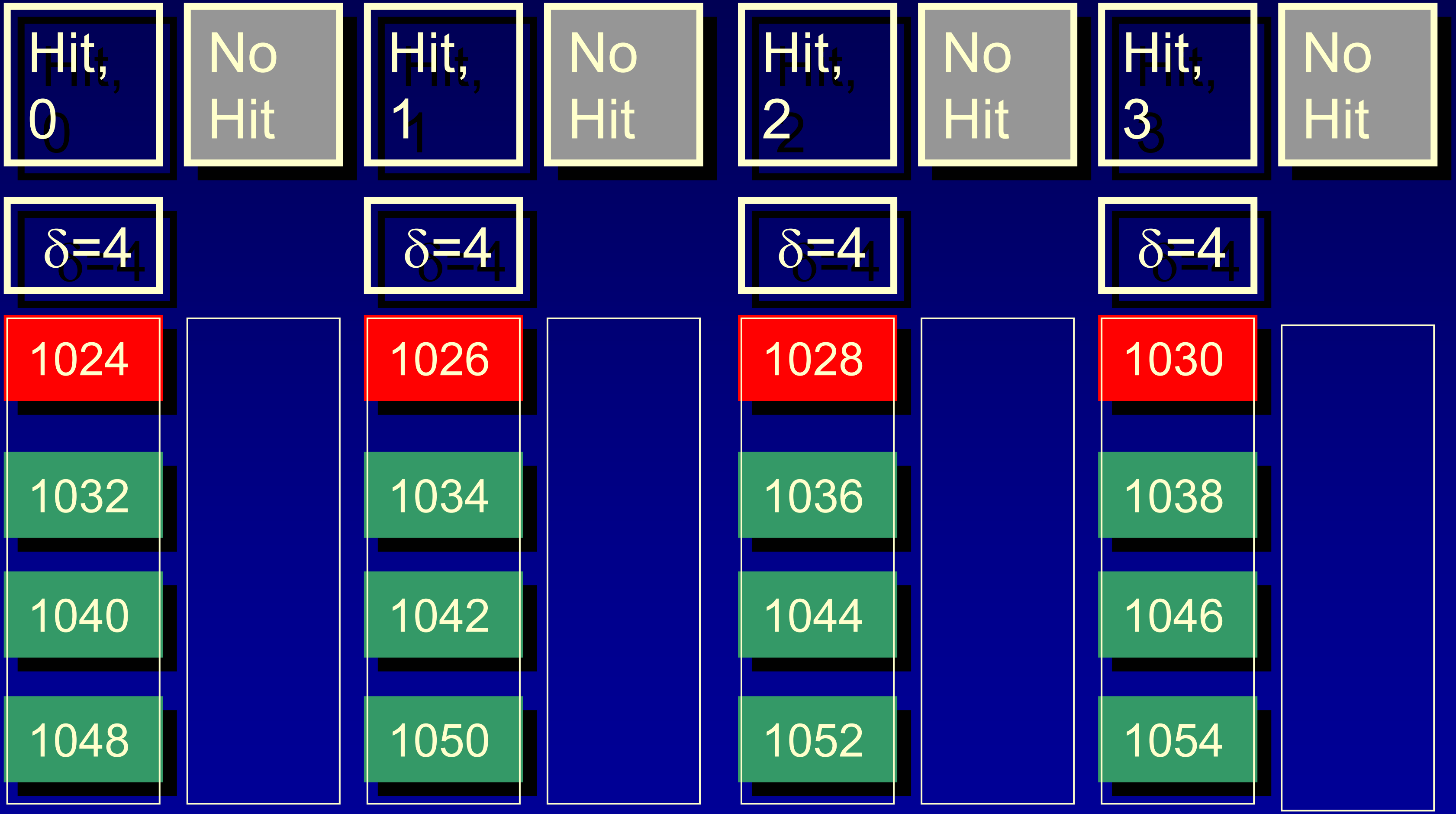
Word Interleaved, Serial Vector Gathers

V = < 1024, 1, 16 > (cache-line fill)

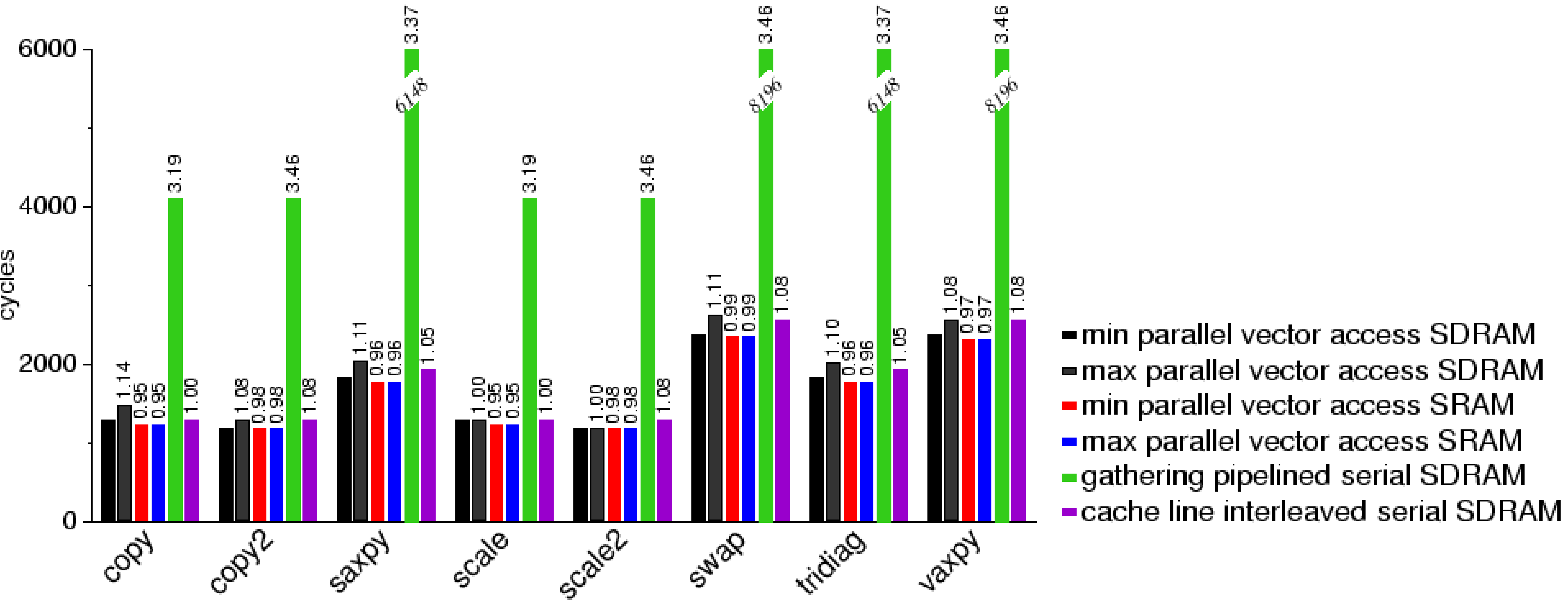


Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

PVA Stride-2 Gather



Stride-1 Vectors (Cache Line Fills)

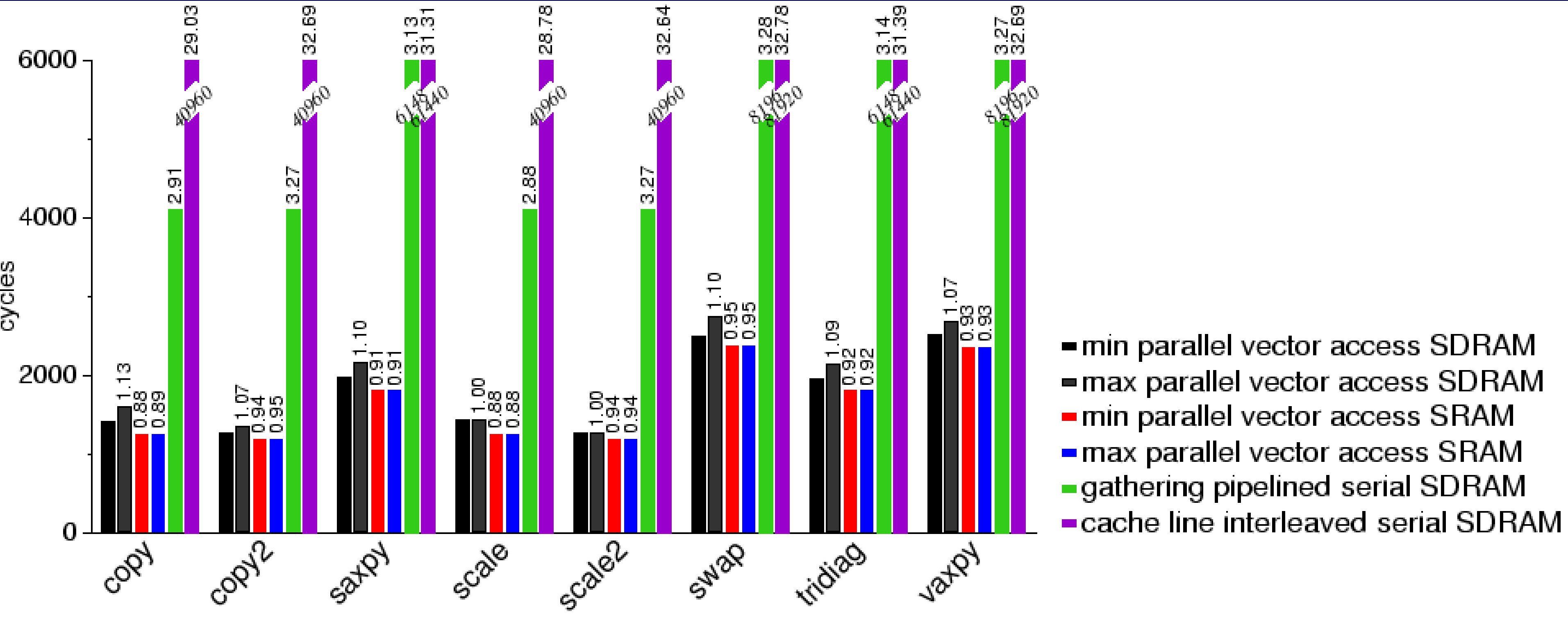


SDRAM PVA takes about same time as SRAM system

PVA takes about same time as cache-line optimized controller

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Stride-19 Vectors (Diagonal Example)



PVA takes about same time as SRAM memory system

PVA takes about same time as for stride-1 vector

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

PVA Results Summary

FPGA Synthesis:

- 3600 lines of Verilog
- 10K logic elements and 2K on-chip RAM
- FirstHit() requires 2 cycles (under 20nsec at 100MHz)
- NextHit() requires 1 cycle
- Minimal increase in hardware complexity

Highlights of performance:

- Stride 1: PVA fast as usual cacheline-optimized serial unit (99%-108%)
- Stride 4: PVA 3x faster than pipelined serial gather unit
- Stride 19: PVA up to 33x faster than cacheline-optimized serial unit
- Specific gains depend on relative skew of the various vectors
- 2-5x faster than similar proposed designs

The Bad News

These are uniprocessor solutions

- Working on SMP adaptations
- Require hardware/software changes
- Complexity still isolated

Have to restructure code

- Compiler can do much of the work
- Can semi-automate the rest?
- Need better tools

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Tools Wish List

Memory performance monitoring

- Better metrics
- Automatic identification of bottlenecks

Visualization

Interactive performance tuning

- Let compiler do what it can
- Exploit user's knowledge of application
- Exploit temporal locality better

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

So What Do We Do?

We're stuck with DRAM

- Economics
- Lack of viable alternatives

Everything we can

- Change hardware (where possible)
- Restructure code (at least recompile)
- Build better tools

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering
CORNELL

Questions?

www.cs.utah.edu/impulse

www.csl.cornell.edu/~sam/papers

sam@csl.cornell.edu

Sally A. McKee
Computer Systems Laboratory
Electrical and Computer Engineering