

# NAND-Flash Storage for High-Performance Computing

Craig Ulmer  
cdulmer@sandia.gov



September 25, 2007

Craig Ulmer  
Maya Gokhale  
Greg Diamos  
Michael Rewak

SNL/CA,  
LLNL  
Georgia Institute of Technology  
University of Florida

# Overview

- Storage-Intensive Super Computing (SISC) project at LLNL
  - Investigate architectures for data-driven applications
  - Exploit new storage and computing technologies
- Near-Term: NAND Flash Memory
  - Consumer demand has made solid-state storage feasible
  - Vendors beginning to produce flash drives with mixed results
- Research questions
  - What are the operational characteristics of flash?
  - How can scientific users take advantage?





# Outline

- Motivation: Storage performance
- NAND-Flash Fundamentals
- Early experiences with commercial hardware
- Concluding remarks



# Motivation: Storage Performance

# Computing Improvements 1987-2007

- Significant gains in computing over the last 20 years
  - Storage performance lags
- How much does this affect scientific computing?

| Year | CPU            | MFLOPS | Memory | I/O Bus | Network | Storage  |           |        |
|------|----------------|--------|--------|---------|---------|----------|-----------|--------|
|      |                |        |        |         |         | Capacity | Bandwidth | Access |
| 1987 | 386/387        | 0.1    | 1 MB   | 5 MB/s  | 10 Mb/s | 50 MB    | 5 MB/s    | 30 ms  |
| 2007 | Dual Quad-Core | 20,000 | 4 GB+  | 8 GB/s  | 10 Gb/s | 1 TB     | 110 MB/s  | 3 ms   |

> 1,000x

< 25x

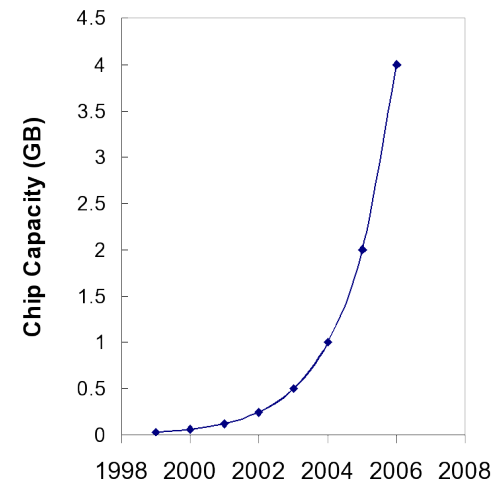


# Storage Opportunities in Science

- **Informatics**
  - Simulations produce massive datasets
  - Data discovery algorithms to find regions of interest
  - Examples: Post processing, Intelligence Community
- **Checkpointing**
  - Reliability is a serious problem in petascale systems
  - Disk arrays have trouble keeping up with network
- **Deployable Science**
  - Sensors capture large amounts of data, but network is slow
  - Embed mass storage at node, analyze data at source
  - Examples: Wireless Sensor Networks, Satellite architectures

# NAND-Flash

- NAND-Flash as non-volatile storage
  - Capacities and performance between DRAM and hard disk
  - Capacity doubling every year
  - Access time: *microseconds* instead of milliseconds
- Multiple NAND-Flash vendors
  - Samsung (45%), Toshiba, Hynix, Micron, Intel
  - >\$3B in sales last quarter
- Commercial flash solid state drives
  - 32-128 GB Flash SATA drives at \$50/GB
  - Read/Write: 100/80 MB/s
  - Access Time: 0.1 ms (>30x improvement)

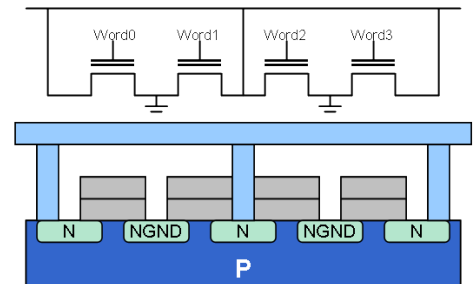




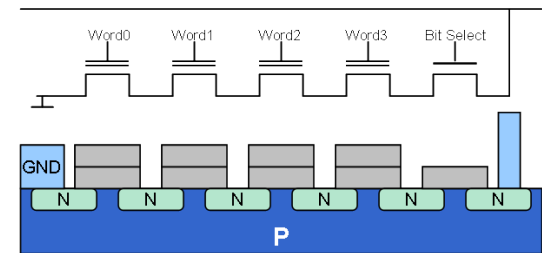
# NAND-Flash Fundamentals

# NAND-Flash Overview

- Flash memory invented in 1984 by Toshiba
  - Based on floating-gate transistors
  - Erase stores charge in floating-gate '1'
  - Write of '0' clears charge, '1' no change
  - Cell can be reprogrammed 100,000 times
- Two gate layout strategies for flash
  - NOR: Slow erase/program, but random access
  - NAND: Fast, higher capacity, but page based
- Majority of flash is NAND based
  - 32-Gbit parts available
- Example: Micron's 16-Gbit Part (MT29F16G)

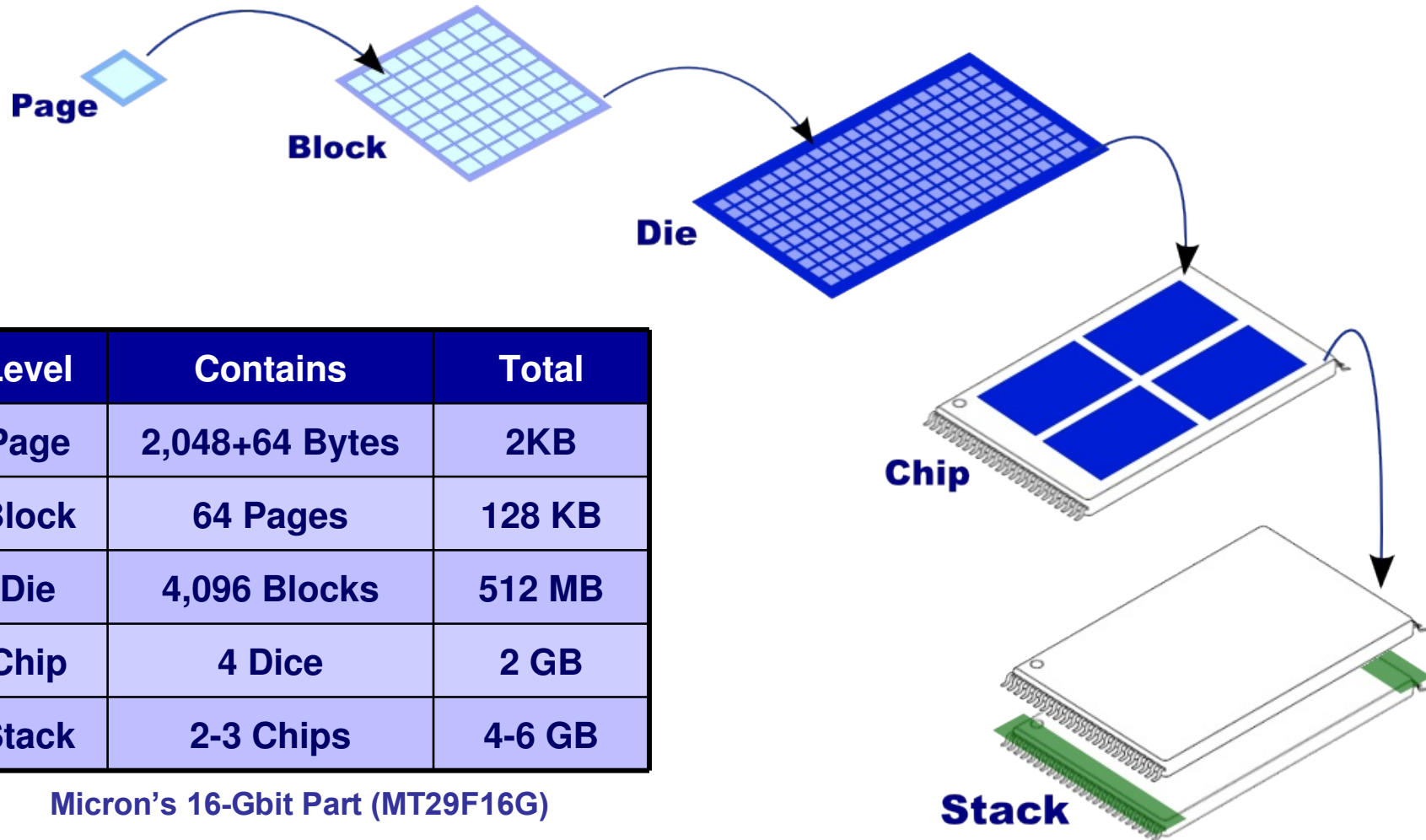


NOR Flash



NAND Flash

# Storage Hierarchy

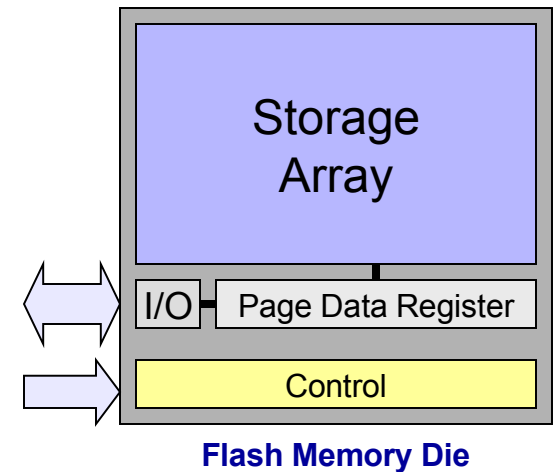


| Level | Contains       | Total  |
|-------|----------------|--------|
| Page  | 2,048+64 Bytes | 2KB    |
| Block | 64 Pages       | 128 KB |
| Die   | 4,096 Blocks   | 512 MB |
| Chip  | 4 Dice         | 2 GB   |
| Stack | 2-3 Chips      | 4-6 GB |

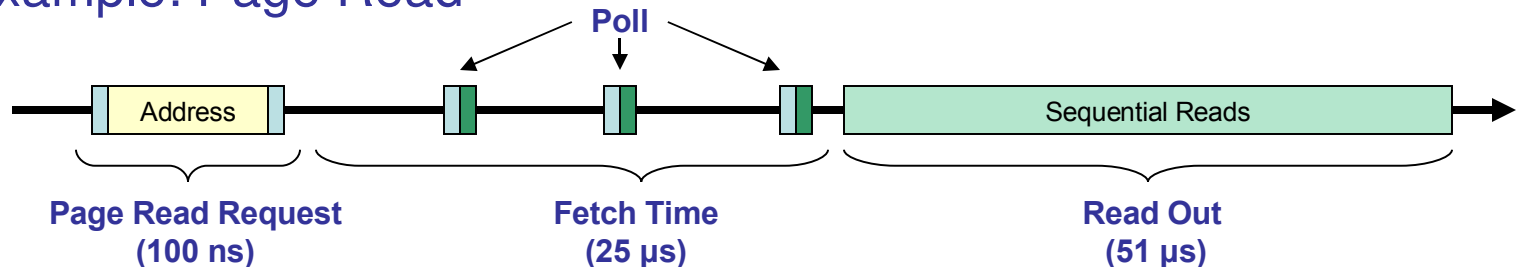
Micron's 16-Gbit Part (MT29F16G)

# Basic Interface to a Single Chip

- Small number of pins (15)
  - 8 pins bidirectional
- Page-level access
  - Die holds active page in data register
  - Random seeks possible in page



- Example: Page Read



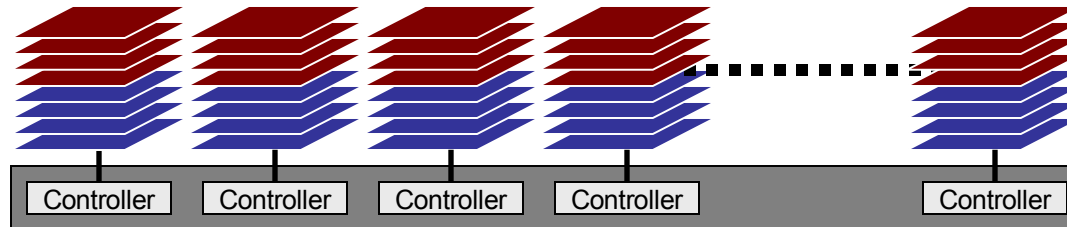


# Architecture Observations: Hardships

- Delete/Modifies can be problematic
  - Pages are WORM-like
  - Can only erase blocks, not pages
  - Erases have high overhead (2 ms)
- Reliability
  - Cells have a limited lifetime: 100,000 erase/program cycles
  - Products must do error correction and wear leveling
- Transfer Performance
  - Narrow bus limits data transfer bandwidth to 40 MB/s
  - Decent for a single chip...
    - ...but still need to be clever how we access data

# Architecture Observations: Opportunities

- Hierarchy provides vertical parallelism
  - Simultaneous requests to each die to hide access time
  - Bottleneck becomes data transfer
- Low pin count provides horizontal parallelism
  - Stripe data across multiple chips
  - Manage independently
- Logical to implement controllers in FPGA
  - Room for application-specific hardware





# Early Experiments with Fusion-io Hardware

# Fusion-io Hardware

- Fusion-io
  - Officially announced this week, products early next year
- Fusion-io: PCIe NAND-Flash Storage Device
  - Compact board populated with an FPGA and 40 flash chips
  - IP makes board appear as standard block device
- Evaluation at SNL/CA and LLNL
  - Beta board with 16 chips (32GB)
  - Host-level performance evaluation
  - FPGA experiments

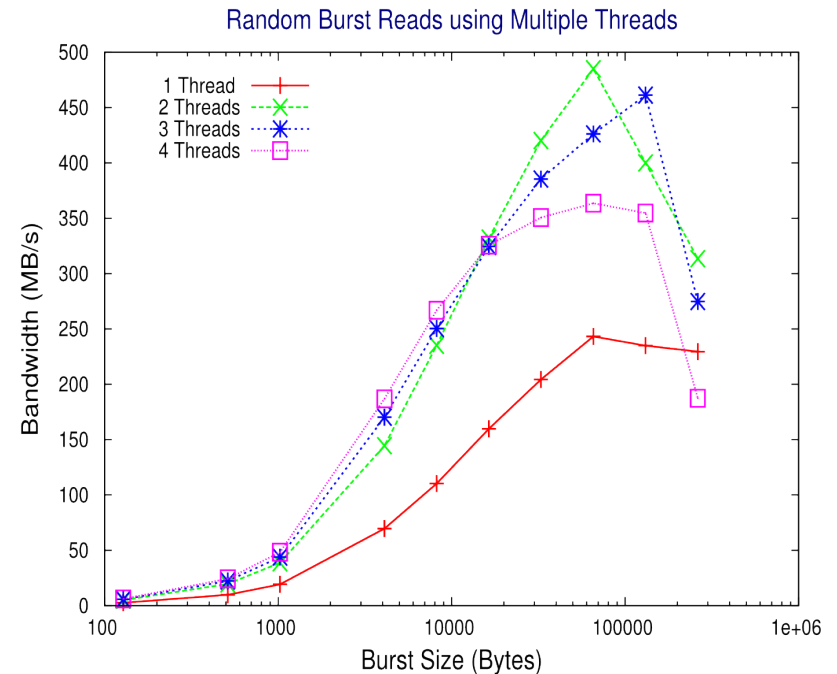


# Host-Level Testing

- Simple benchmarks from host applications
  - Quad-Core system w/ 2 SATA drives in RAID0
- Streaming transfers

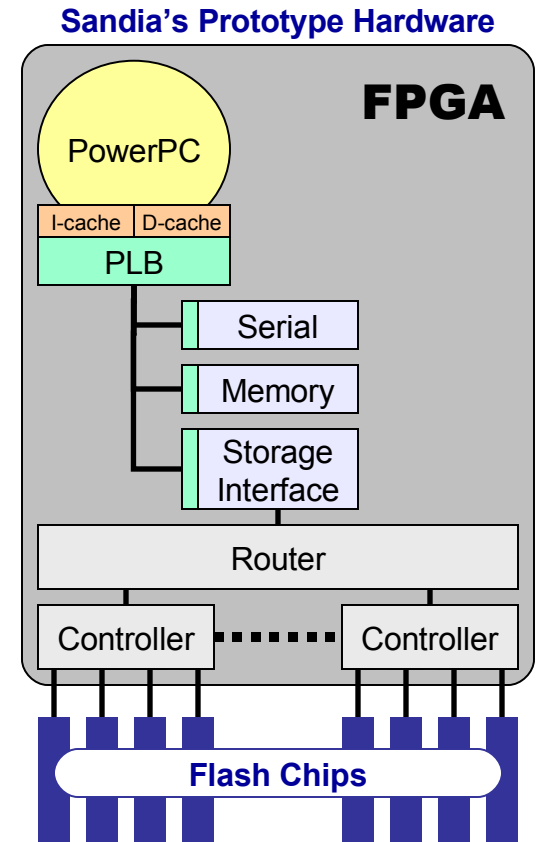
| Drive     | Read     | Write    |
|-----------|----------|----------|
| RAID0     | 120 MB/s | 96 MB/s  |
| Fusion-io | 446 MB/s | 378 MB/s |

- Random burst read test
  - Multiple-threads
  - RAID0: 10 MB/s
  - Fusion-io: 460 MB/s



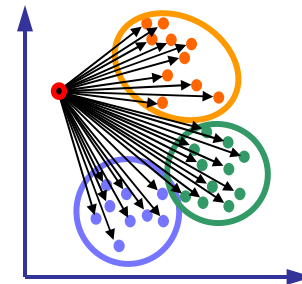
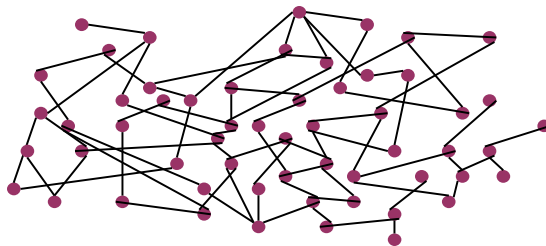
# FPGA Experiments

- Reprogram FPGA with our own hardware
  - Vehicle for low-level flash experiments
  - Prototype application engines
- Constructed a development platform
  - Four Flash memory controllers
  - Interface into on-chip PowerPC
  - Allows hardware/software prototypes
  - Currently no off-card communication



# FPGA Experiments

- Access time measurements
  - Expected 25  $\mu$ s, but observed 11  $\mu$ s reads, 18  $\mu$ s writes
- Read reliability
  - Filled a chip and then reread constantly for 3 weeks without errors
- Work-in-Progress: Application-Specific handlers
  - Goal: Minimize amount of data transferred over pins
  - Embed simple readers at flash controller
  - Examples: graph algorithms and k-nearest neighbors



# Summary

- Flash memory advancing at rapid rate
  - Speed much better than disk, capacity improving
  - Products are beginning to become available
  - Vendors must be careful about how they transfer data
- Disruptive technology for data-driven applications
  - Plug-in replacement can give performance speedups
  - Worthwhile to rethink storage strategies (threading, WORM)
- Technology allowing us to think beyond current constraints
  - Revisit intelligent storage concepts when it makes sense

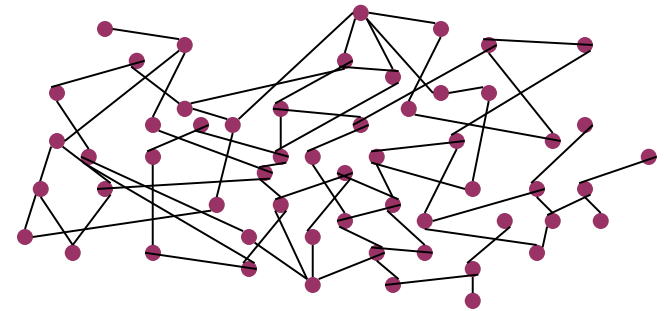




# Backup Slides

# Work-in-Progress: FPGA Applications

- Sparse Graphs: Breadth-First Search
  - Millions of nodes, few neighbors
  - Strategy:
    - Pack multiple nodes into page
    - Controller extracts only relevant info
  - Advantage: extracts only relevant info



- Data Analysis: K-Nearest Neighbors
  - Compare vector to all training vectors
  - Strategy
    - Stream through linked list of vectors
    - Stop comparison as soon as crosses threshold
  - Advantage:
    - Early termination of reads
    - Fine-grained control of access

