Memory: Technology and Patterns

Memory, the 10,000 ft view. Latency, from Steve Wozniak to the power wall.

How DRAM works. Memory design when low cost per bit is the priority.

Break

How SRAM works. The memory technology available on logic dies.

Memory design patterns. Ways to use SRAM in your project designs.
40% of this ARM CPU is devoted to SRAM cache.

But the role of cache in computer design has varied widely over time.
1977: DRAM faster than microprocessors

CPU: 1000 ns
DRAM: 400 ns

Steve Jobs
Steve Wozniak
Q. How do architects address this gap?
A. Put smaller, faster “cache” memories between CPU and DRAM. Create a “memory hierarchy”.

The power wall

CPU
60% per yr
2X in 1.5 yrs

DRAM
9% per yr
2X in 10 yrs

Gap grew 50% per year

Performance
(1/latency)

Year
19 9 0 0 0 0
10000 1000 100 10 1
Caches: Variable-latency memory ports

Data in upper memory returned with lower latency.

Data in lower level returned with higher latency.

From CPU

To CPU

From CPU

To CPU

Data in upper memory returned with lower latency.

Data in lower level returned with higher latency.

From CPU

To CPU

Data in upper memory returned with lower latency.

Data in lower level returned with higher latency.

From CPU

To CPU

Data in upper memory returned with lower latency.

Data in lower level returned with higher latency.

From CPU

To CPU

Data in upper memory returned with lower latency.

Data in lower level returned with higher latency.
Programs with locality cache well ...

The caching algorithm in one slide

Temporal locality: Keep most recently accessed data closer to processor.

Spatial locality: Move contiguous blocks in the address space to upper levels.
Goal: Illusion of large, fast, cheap memory

Let programs address a memory space that scales to the disk size, at a speed that is usually as fast as register access
90 nm, 58 M transistors
L1 (64K Instruction) ↓ ↓ ↓ ↓
L1 (32K Data) ↑ ↑ ↑ ↑
512K L2

Registers (1K)
PowerPC 970 FX
## Latency: A closer look

Read latency: Time to return first byte of a random access

<table>
<thead>
<tr>
<th>Size</th>
<th>Reg</th>
<th>L1 Inst</th>
<th>L1 Data</th>
<th>L2</th>
<th>DRAM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>1</td>
<td>64K</td>
<td>32K</td>
<td>512K</td>
<td>256M</td>
<td>80G</td>
</tr>
<tr>
<td>64K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Latency (cycles) | 1 | 3 | 3 | 11 | 160 | 1E+07 |
| Latency (sec)   | 0.6n | 1.9n | 1.9n | 6.9n | 100n | 12.5m |
| Hz              | 1.6G | 533M | 533M | 145M | 10M  | 80    |

### Architect’s latency toolkit:

1. **Parallelism.** Request data from $N$ 1-bit-wide memories at the same time. Overlaps latency cost for all $N$ bits. Provides $N$ times the bandwidth. Requests to $N$ memory banks (interleaving) have potential of $N$ times the bandwidth.

2. **Pipeline memory.** If memory has $N$ cycles of latency, issue a request each cycle, receive it $N$ cycles later.
Capacitance and memory
State is coded as the amount of **energy** stored by a device.

State is read by sensing the amount of energy

Problems: **noise** changes $Q$ (up or down), **parasitics** leak or source $Q$. Fortunately, $Q$ cannot change **instantaneously**, but that only gets us in the ballpark.
How do we fight noise and win?

Store more energy than we expect from the noise.

\[ Q = CV. \] To store more charge, use a bigger \( V \) or make a bigger \( C \).

Cost: Power, chip size.

Represent state as charge in ways that are robust to noise.

Example: 1 bit per capacitor.

Write 1.5 volts on \( C \).

To read \( C \), measure \( V \).

\( V > 0.75 \) volts is a “1”.

\( V < 0.75 \) volts is a “0”.

Cost: Could have stored many bits on that capacitor.

Correct small state errors that are introduced by noise.

Ex: read \( C \) every 1 ms.

Is \( V > 0.75 \) volts?

Write back 1.5V (yes) or 0V (no).

Cost: Complexity.
Dynamic Memory Cells
DRAM cell: 1 transistor, 1 capacitor

Word Line and Vdd run on "z-axis"

Why $V_{\text{cap}}$ values start out at ground.

Diode leakage current.
A 4 x 4 DRAM array (16 bits) ....
Invented after SRAM, by Robert Dennard

United States Patent Office

Patented June 4, 1968

3,378,286

FIELD-EFFECT TRANSISTOR MEMORY
Robert H. Dennard, Croton-on-Hudson, N.Y., assignor to
International Business Machines Corporation, Armonk,
N.Y., a corporation of New York
Filed July 14, 1967, Ser. No. 653,415
21 Claims. (Cl. 340—173)

FIG. 1

FIG. 2

ponent in disclosing various concepts and structures which have been developed in the application of field-effect transistors to different types of memory applications, the primary thrust up to this time in conventional read-write random access memories has been to connect a plurality of field-effect transistors in each cell in a latch configuration. Memories of this type require a large number of active devices in each cell and therefore each cell re-
DRAM Circuit Challenge #1: Writing

Vdd - Vth. Bad, we store less charge. Why do we not get Vdd?

\[ I_{ds} = k \left[V_{gs} - V_{th}\right]^2, \]
but "turns off" when \( V_{gs} \leq V_{th} \)!

Vgs = Vdd - Vc. When Vdd - Vc = Vth, charging effectively stops!
DRAM Challenge #2: Destructive Reads

Raising the word line removes the charge from every cell it connects to!

DRAMs write back after each read.
DRAM Circuit Challenge #3a: Sensing

Assume Ccell = 1 fF

Bit line may have 2000 nFet drains, assume bit line C of 100 fF, or 100*Ccell.

Ccell holds \( Q = Ccell \cdot (Vdd - Vth) \)

When we dump this charge onto the bit line, what voltage do we see?

\[
dV = \frac{[Ccell \cdot (Vdd - Vth)]}{[100 \cdot Ccell]} \\
dV = \frac{(Vdd - Vth)}{100} \approx \text{tens of millivolts!}
\]

In practice, scale array to get a 60mV signal.
DRAM Circuit Challenge #3b: Sensing

How do we reliably sense a 60mV signal?

Compare the bit line against the voltage on a "dummy" bit line.

"Dummy" bit line. Cells hold no charge.

Bit line to sense

Dummy bit line

"sense amp"
DRAM Challenge #4: Leakage ...

Parasitic currents leak away charge.

Solution: “Refresh”, by rewriting cells at regular intervals (tens of milliseconds).

Diode leakage...
Cell capacitor holds 25,000 electrons (or less). Cosmic rays that constantly bombard us can release the charge!

Solution: Store extra bits to detect and correct random bit flips (ECC).
DRAM Challenge #6: Yield

If one bit is bad, do we throw chip away?

Solution: add extra bit lines (i.e. 80 when you only need 64). During testing, find the bad bit lines, and use high current to burn away "fuses" put on chip to remove them.

Extra bit lines. Used for "sparing".
DRAM Challenge #7: Scaling

Each generation of IC technology, we shrink width and length of cell.

**Problem 1:** If Ccell and drain capacitances scale together, number of bits per bit line stays constant.

\[ dV = 60 \text{ mV} = \frac{[C_{\text{cell}}(V_{\text{dd}} - V_{\text{th}})]}{[100 \times C_{\text{cell}}]} \]

**Problem 2:** Vdd may need to scale down too! Number of electrons per cell shrinks.

Solution: Constant Innovation of Cell Capacitors!
Poly-diffusion C-cell is ancient history

Word Line and Vdd run on "z-axis"
Early replacement: “Trench” capacitors

Figure 4

SEM photomicrograph of 0.25-μm trench DRAM cell suitable for scaling to 0.15 μm and below. Reprinted with permission from [17]; © 1995 IEEE.
The companies that kept scaling trench capacitors for commodity DRAM chips went out of business.
Samsung 90nm stacked capacitor bitcell.

DRAM: the field for material and process innovation

Arabinda Das
From JSSC, and Arabinda Das
A 31 ns Random Cycle VCAT-Based 4F² DRAM
With Manufacturability and Enhanced Cell Efficiency

Ki-Whan Song, Jin-Young Kim, Jae-Man Yoon, Sua Kim, Huijung Kim, Hyun-Woo Chung, Hyungi Kim, Kanguk Kim, Hwan-Wook Park, Hyun Chul Kang, Nam-Kyun Tak, Dukha Park, Woo-Seop Kim, Member, IEEE, Yeong-Taek Lee, Yong Chul Oh, Gyo-Young Jin, Jeiwhan Yoo, Donggun Park, Senior Member, IEEE, Kyungseok Oh, Changhyun Kim, Senior Member, IEEE, and Young-Hyun Jun
Memory Arrays
People buy DRAM for the bits. "Edge" circuits are overhead.

So, we amortize the edge circuits over big arrays.
A “bank” of 128 Mb (512Mb chip -> 4 banks)

In reality, 16384 columns are divided into 64 smaller arrays.

13-bit row address input

1 of 8192 decoder

16384 columns

8192 rows

134 217 728 usable bits
(tester found good bits in bigger array)

16384 bits delivered by sense amps

Select requested bits, send off the chip
Recall DRAM Challenge #3b: Sensing

How do we reliably sense a 60mV signal?

Compare the bit line against the voltage on a “dummy” bit line.

“Dummy” bit line. Cells hold no charge.

Bit line to sense

Dummy bit line

“sense amp”
“Sensing” is row read into sense amps

Slow! This 2.5ns period DRAM (400 MT/s) can do row reads at only 55 ns (18 MHz).

DRAM has high latency to first bit out. A fact of life.

13-bit row address input

1 of 8192 decoder

8192 rows 134 217 728 usable bits (tester found good bits in bigger array)

16384 columns

16384 bits delivered by sense amps

Select requested bits, send off the chip
An ill-timed refresh may add to latency.

Parasitic currents leak away charge.

Solution: “Refresh”, by rewriting cells at regular intervals (tens of milliseconds).
Latency versus bandwidth

What if we want all of the 16384 bits?
In row access time (55 ns) we can do 22 transfers at 400 MT/s.
16-bit chip bus -> 22 x 16 = 352 bits << 16384
Now the row access time looks fast!

Thus, push to faster DRAM interfaces

13-bit row address input

16384 columns

134 217 728 usable bits (tester found good bits in bigger array)

16384 bits delivered by sense amps

Select requested bits, send off the chip
DRAM latency/bandwidth chip features

Columns: Design the right interface for CPUs to request the subset of a column of data it wishes:

- 16384 bits delivered by sense amps
- Select requested bits, send off the chip

Interleaving: Design the right interface to the 4 memory banks on the chip, so several row requests run in parallel.

- Bank 1
- Bank 2
- Bank 3
- Bank 4
Off-chip interface for the Micron part ...

A clocked bus: 200 MHz clock, data transfers on both edges (DDR).

Note! This example is best-case!
To access a new row, a slow ACTIVE command must run before the READ.

DRAM is controlled via commands (READ, WRITE, REFRESH, ...)

Synchronous data output.
Opening a row before reading ...

Auto-Precharge READ

55 ns between row opens.
However, we can read columns quickly

![Diagram showing memory operation](image)

**Note:** This is a “normal read” (not Auto-Precharge). Both READs are to the same bank, but different columns.
Why can we read columns quickly?

13-bit row address input

8192 rows

Column reads select from the 16384 bits here

16384 columns

134 217 728 usable bits (tester found good bits in bigger array)

16384 bits delivered by sense amps

Select requested bits, send off the chip
Interleave: Access all 4 banks in parallel

Interleaving: Design the right interface to the 4 memory banks on the chip, so several row requests run in parallel.

Bank a  Bank b  Bank c  Bank d

Can also do other commands on banks concurrently.
Only part of a bigger story ...
DRAM controllers: reorder requests

(A) Without access scheduling (56 DRAM Cycles)

(B) With access scheduling (19 DRAM Cycles)

DRAM Operations:

P: bank precharge (3 cycle occupancy)
A: row activation (3 cycle occupancy)
C: column access (1 cycle occupancy)
Memory Packaging
From DRAM chip to DIMM module ...

Each RAM chip responsible for 8 lines of the 64 bit data bus (U5 holds the check bits).

Commands sent to all 9 chips, qualified by per-chip select lines.
MacBook Air ... too thin to use DIMMs
Macbook Air

Core i5: CPU + DRAM controller

4GB DRAM soldered to the main board
Original iPad (2010)  

128MB SDRAM dies (2)  

Apple A4 SoC  

“Package-in-Package”  

Cut-away side view  

Dies connect using bond wires and solder balls...
### 3-D memory stack

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM die size</td>
<td>10.7 mm x 13.3 mm</td>
</tr>
<tr>
<td>DRAM die thickness</td>
<td>50 μm</td>
</tr>
<tr>
<td>TSV count in DRAM</td>
<td>1,560</td>
</tr>
<tr>
<td>DRAM capacity</td>
<td>512 Mbit/die x 2 strata</td>
</tr>
<tr>
<td>CMOS logic die size</td>
<td>17.5 mm x 17.5 mm</td>
</tr>
<tr>
<td>CMOS logic die thickness</td>
<td>200 μm</td>
</tr>
<tr>
<td>CMOS logic bump count</td>
<td>3,497</td>
</tr>
<tr>
<td>CMOS logic process</td>
<td>0.18 μm CMOS</td>
</tr>
<tr>
<td>DRAM-logic FTI via pitch</td>
<td>50 μm</td>
</tr>
<tr>
<td>Package size</td>
<td>33 mm x 33 mm</td>
</tr>
<tr>
<td>BGA terminal</td>
<td>520 pin / 1 mm pitch</td>
</tr>
</tbody>
</table>

### Figure

*1 Gbit stacked DRAM with TSV (512 Mbit x 2 strata)*

- Molded resin
- Silicon lid
- FTI
- CMOS logic
- BGA

**A 3D Stacked Memory Integrated on a Logic Device Using SMAFTI Technology**

Yoshio Kurita, Satoshi Matsui, Nobuaki Takahashi, Koji Serajima, Masahiro Komuro, Makoto Itou, Chika Kakegawa, Masaya Kawano, Yoshimi Egawa, Yoshinari Sada, Hidaka Nakatsuki, Osamu Kato, Azusa Yamasaki, Toshio Mitahashi, Masakazu Ishino, Kayoko Shibata, Shiro Ushiyama, Junji Yamada, and Hiroaki Ikeda

NEC Electronics, Oki Electric Industry, and Elpida Memory

1120 Shimokuzawa, Sagamihara, Kanagawa 229-1198, Japan

y.kurita@necel.com
Break
**Static Memory Circuits**

**Dynamic Memory:** Circuit remembers for a fraction of a second.

**Static Memory:** Circuit remembers as long as the power is on.

**Non-volatile Memory:** Circuit remembers for many years, even if power is off.
Recall DRAM cell: 1 T + 1 C

Word Line

Row

Column

Bit Line
Idea: Store each bit with its complement

We can use the redundant representation to compensate for noise and leakage.
Case #1: $y = \text{Gnd}$, $\overline{y} = \text{Vdd}$ ...
Case #2: \( y = V_{dd}, \overline{y} = G_{nd} \ldots \)
Combine both cases to complete circuit

"Cross-coupled inverters"
SRAM Challenge #1: It’s so big!

SRAM area is 6X-10X DRAM area, same generation ...

Cell has both transistor types

Capacitors are usually "parasitic" capacitance of wires and transistors.

Vdd AND Gnd

More contacts, more devices, two bit lines ...
Challenge #2: Writing is a “fight”

When word line goes high, bitlines “fight” with cell inverters to “flip the bit” -- must win quickly!

Solution: tune W/L of cell & driver transistors

Initial state $V_{dd}$

Initial state $Gnd$

Bitline drives $Gnd$

Bitline drives $V_{dd}$
Challenge #3: Preserving state on read

When word line goes high on read, cell inverters must drive large bitline capacitance quickly, to preserve state on its small cell capacitances.

Cell state Vdd

Cell state Gnd

Bitline a big capacitor
Adding More Ports

Differential Read or Write ports

Optional Single-ended Read port
SRAM array: like DRAM, but non-destructive

Architects specify number of rows and columns. Word and bit lines slow down as array grows larger!

Parallel Data I/O Lines

How could we pipeline this memory?
Building Larger Memories

- Large arrays constructed by tiling multiple leaf arrays, sharing decoders and I/O circuitry
  - e.g., sense amp attached to arrays above and below
- Leaf array limited in size to 128-256 bits in row/column due to RC delay of wordlines and bitlines
- Also to reduce power by only activating selected sub-bank
- In larger memories, delay and energy dominated by I/O wiring

<table>
<thead>
<tr>
<th>Bit cells</th>
<th>Dec</th>
<th>Bit cells</th>
<th>Dec</th>
<th>Bit cells</th>
<th>Dec</th>
<th>Bit cells</th>
<th>Dec</th>
<th>Bit cells</th>
<th>Dec</th>
<th>Bit cells</th>
<th>Dec</th>
<th>Bit cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
</tr>
<tr>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
</tr>
<tr>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
<td></td>
<td>I/O</td>
</tr>
<tr>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
<td>Dec</td>
<td>Bit cells</td>
</tr>
</tbody>
</table>
SRAM vs DRAM, pros and cons

DRAM has a 6-10X density advantage at the same technology generation.

**SRAM advantages**

- SRAM has deterministic latency: its cells do not need to be refreshed.
- SRAM is much faster: transistors drive bitlines on reads.
- SRAM is easy to design in logic fabrication process (and premium logic processes have SRAM add-ons).
Flip Flops Revisited
Recall: Static RAM cell (6 Transistors)
Recall: Positive edge-triggered flip-flop

A flip-flop “samples” right before the edge, and then “holds” value.

Sampling circuit

Holds value

16 Transistors: Makes an SRAM look compact!

What do we get for the 10 extra transistors? Clocked logic semantics.
Small Memories from Stdcell Latches

- Add additional ports by replicating read and write port logic (multiple write ports need mux in front of latch)
- Expensive to add many ports

Diagram:
- Write Address Decoder
- Write Address
- Write Data
- Clk
- Data held in transparent-low latches
- Write by clocking latch
- Read Address Decoder
- Read Address
- Optional read output latch
- Combinational logic for read port (synthesized)
- Clock (Clk)
Synthesized, custom, and SRAM-based register files, 40nm

For small register files, logic synthesis is competitive.

Not clear if the SRAM data points include area for register control, etc.

Figure 3: Using the raw area data, the physical implementation team can get a more accurate area estimation early in the RTL development stage for floorplanning purposes. This shows an example of this graph for a 1-port, 32-bit-wide SRAM.

Bhupesh Dasila
Memory Design Patterns
When register files get big, they get slow.

Even worse: adding ports slows down as $O(N^2)$ ...

Why? Number of loads on each Q goes as $O(N)$, and the wire length to port mux goes as $O(N)$. 
True Multiport Example: Itanium-2 Regfile

- Intel Itanium-2 [Fetzer et al, IEEE JSSCC 2002]

B. Operand Bypass Datapath

The integer datapath bypassing is divided into four stages, to afford more timing critical inputs the least possible logic delay to the consuming ALUs. Critical L1 cache return data must flow through only one level of muxing before arriving at the ALU inputs, while DET and WRB data, available from staging latches, have the longest logic path to the ALUs. This allows the bypassing of operands from 34 possible results to occur in a half clock cycle, enabling a single-cycle cache access and instruction execution.
True Multiport Memory

**Problem:** Require simultaneous read and write access by multiple independent agents to a shared common memory.

**Solution:** Provide separate read and write ports to each bit cell for each requester.

**Applicability:** Where unpredictable access latency to the shared memory cannot be tolerated.

**Consequences:** High area, energy, and delay cost for large number of ports. Must define behavior when multiple writes on same cycle to same word (e.g., prohibit, provide priority, or combine writes).
Crossbar networks: many CPUs sharing cache banks

Sun Niagara II: 8 cores, 4MB L2, 4 DRAM channels

<table>
<thead>
<tr>
<th>SPC 0</th>
<th>L2$ bank 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPC 1</td>
<td>L2$ bank 1</td>
</tr>
<tr>
<td>SPC 2</td>
<td>L2$ bank 2</td>
</tr>
<tr>
<td>SPC 3</td>
<td>L2$ bank 3</td>
</tr>
<tr>
<td>SPC 4</td>
<td>L2$ bank 4</td>
</tr>
<tr>
<td>SPC 5</td>
<td>L2$ bank 5</td>
</tr>
<tr>
<td>SPC 6</td>
<td>L2$ bank 6</td>
</tr>
<tr>
<td>SPC 7</td>
<td>L2$ bank 7</td>
</tr>
</tbody>
</table>

Each DRAM channel: 50 GB/s Read, 25 GB/s Write BW.

Crossbar BW: 270 GB/s total (Read + Write).

(Also shared by an I/O port, not shown)
Banked Multiport Memory

**Problem:** Require simultaneous read and write access by multiple independent agents to a large shared common memory.

**Solution:** Divide memory capacity into smaller banks, each of which has fewer ports. Requests are distributed across banks using a fixed hashing scheme. Multiple requesters arbitrate for access to same bank/port.

**Applicability:** Requesters can tolerate variable latency for accesses. Accesses are distributed across address space so as to avoid “hotspots”.

**Consequences:** Requesters must wait arbitration delay to determine if request will complete. Have to provide interconnect between each requester and each bank/port. Can have greater, equal, or lesser number of banks*ports/bank compared to total number of external access ports.
Banked Multiport Memory

Port A

Bank 0

Bank 1

Bank 2

Bank 3

Port B

Arbitration and Crossbar
Cached Multiport Memory

**Problem:** Require simultaneous read and write access by multiple independent agents to a large shared common memory.

**Solution:** Provide each access port with a local cache of recently touched addresses from common memory, and use a cache coherence protocol to keep the cache contents in sync.

**Applicability:** Request streams have significant temporal locality, and limited communication between different ports.

**Consequences:** Requesters will experience variable delay depending on access pattern and operation of cache coherence protocol. Tag overhead in both area, delay, and energy/access. Complexity of cache coherence protocol.
Cached Multiport Memory

Port A

Cache A

Arbitration and Interconnect

Cache B

Port B

Common Memory
The arbiter and interconnect on the last slide is how the two caches on this chip share access to DRAM.
Stream-Buffered Multiport Memory

**Problem:** Require simultaneous read and write access by multiple independent agents to a large shared common memory, where each requester usually makes multiple sequential accesses.

**Solution:** Organize memory to have a single wide port. Provide each requester with an internal stream buffer that holds width of data returned/consumed by each memory access. Each requester can access own stream buffer without contention, but arbitrates with others to read/write stream buffer from memory.

**Applicability:** Requesters make mostly sequential requests and can tolerate variable latency for accesses.

**Consequences:** Requesters must wait arbitration delay to determine if request will complete. Have to provide stream buffers for each requester. Need sufficient access width to serve aggregate bandwidth demands of all requesters, but wide data access can be wasted if not all used by requester. Have to specify memory consistency model between ports (e.g., provide stream flush operations).
Stream-Buffered Multiport Memory

- Stream Buffer A
- Stream Buffer B
- Arbitration
- Wide Memory

Port A
Port B
Replicated-State Multiport Memory

**Problem:** Require simultaneous read and write access by multiple independent agents to a small shared common memory. Cannot tolerate variable latency of access.

**Solution:** Replicate storage and divide read ports among replicas. Each replica has enough write ports to keep all replicas in sync.

**Applicability:** Many read ports required, and variable latency cannot be tolerated.

**Consequences:** Potential increase in latency between some writers and some readers.
Replicated-State Multiport Memory

Write Port 0  Write Port 1

Copy 0  Copy 1

Read Ports

Example: Alpha 21264 Regfile clusters
Intel Micron 8 GB NAND flash device, 2 bit per cell, 25 nm minimum feature, 16.5 mm by 10.1 mm.
NAND Flash Memory
The physics of non-volatile memory

Two gates. But the middle one is not connected.

1. Electrons “placed” on floating gate stay there for many years (ideally).

2. 10,000 electrons on floating gate shift transistor threshold by 2V.

3. In a memory array, shifted transistors hold “0”, unshifted hold “1”.

---

CS 250 L10: Memory
Moving electrons on/off floating gate

A high drain voltage injects “hot electrons” onto floating gate.

1. Hot electron injection and tunneling produce tiny currents, thus writes are slow.

2. High voltages damage the floating gate. Too many writes and a bit goes “bad”.

A high gate voltage “tunnels” electrons off of floating gate.
NAND Flash Memory
Flash: Disk Replacement

Chip “remembers” for 10 years.

Presents memory to the CPU as a set of pages.

Page format:

2048 Bytes + 64 Bytes

(2048 Bytes) (64 Bytes)

(user data) (meta data)

1GB Flash: 512K pages
2GB Flash: 1M pages
4GB Flash: 2M pages
Reading a Page ...

33 MB/s Read Bandwidth

Read Operation

CLE

CE

WE

ALE

RE

Page address in: 175 ns

00h Col. Add1 Col. Add2 Row Add1 Row Add2 Row Add3 30h

Column Address Row Address

I/Ox

Dout N Dout N+ Dout M

Busy

First byte out: 10,000 ns

Clock out page bytes: 52,800 ns

8-bit data or address (bi-directional)

Bus Control

Flash Memory

Samsung K9WAG08U1A
Where Time Goes

Figure 1. K9K8G08U0A Functional Block Diagram

Page address in: 175 ns

First byte out: 10,000 ns

Clock out page bytes: 52,800 ns
Writing a Page ...

A page lives in a block of 64 pages:
1GB Flash: 8K blocks
2GB Flash: 16K blocks
4GB Flash: 32K blocks

To write a page:

1. Erase all pages in the block (cannot erase just one page). Time: 1,500,000 ns

2. May program each page individually, exactly once. Time: 200,000 ns per page.

Block lifetime: 100,000 erase/program cycles.
Block Failure

Even when new, not all blocks work!

1GB: 8K blocks, 160 may be bad.
2GB: 16K blocks, 220 may be bad.
4GB: 32K blocks, 640 may be bad.

During factory testing, Samsung writes good/bad info for each block in the meta data bytes.

<table>
<thead>
<tr>
<th>Block 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 0</td>
</tr>
<tr>
<td>Page 1</td>
</tr>
<tr>
<td>Page 62</td>
</tr>
<tr>
<td>Page 63</td>
</tr>
</tbody>
</table>

2048 Bytes + 64 Bytes

(user data) (meta data)

After an erase/program, chip can say “write failed”, and block is now “bad”. OS must recover (migrate bad block data to a new block). Bits can also go bad “silently” (!!!).
Flash controllers: Chips or Verilog IP...

Flash memory controller manages write lifetime management, block failures, silent bit errors...

Software sees a "perfect" disk-like storage device.
Recall: iPod 2005 ...

Flash memory

Flash controller