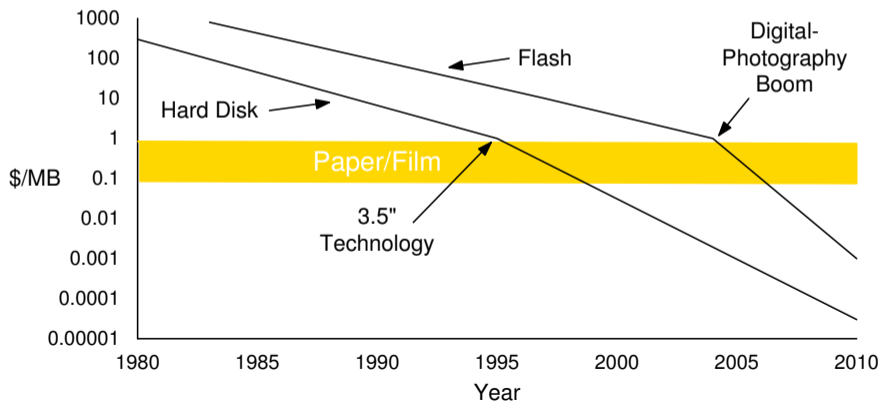# CS 137: File Systems

Persistent Solid-State Storage

Technology Change is Coming

- Disks are cheaper than any solid-state memory
- Likely to be true for many years
- But SSDs are now *cheap enough* for some purposes

ROM

- *ROM* (Read-Only Memory) chips were programmed in the factory
  - Array of transistors
  - Trivial to leave out a wire to make one "defective"
  - Result was array of ones and zeros
- Most of chip predesigned; only one mask layer changed
- Still fairly expensive for that mask
- Ultra-low cost in large volumes

PROM

- ▶ PROM (Programmable ROM) is field-programmable
  - ▶ Array of fuses (literally!)
  - ▶ Blow a fuse to generate a zero
  - ▶ Special high-voltage circuitry to select fuse
- ▶ Much more expensive per chip than ROM
- ▶ But low startup cost made cheaper in low volumes
- ▶ One-time use meant lots of chips thrown away

EPROM

- ► EPROM (Erasable PROM) used floating-gate technology
  - ► Direct predecessor to flash
  - ► Electrons in floating gate (see later slide) store data
  - ► UV light used to drive out electrons and erase
- ► 15 minutes to erase
- ► Expensive, but reusability reduced effective cost


All images from Wikipedia

EEPROM

- ▶ EEPROM (Electrically Erasable PROM) used thinner oxide layer
- ▶ Introduced ca. 1983
- ▶ High voltage could erase without UV
- ▶ Basically flash memory where entire chip erased at once

The Flash Cell
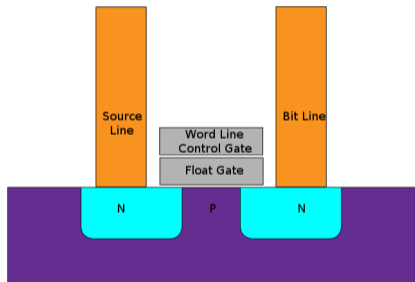
- ▶ Source line provides voltage, bit line senses
- ▶ Current flows between "N" regions, through "P"
- ▶ Voltage on control gate restricts current flow in "P"
- ▶ Charge on floating gate "screens" control gate
  - ▶ Allows sensing whether charge is present

Programming NOR Flash

Programming Via Hot Electron Injection
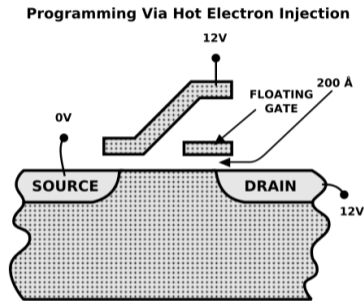
- ▶ Default state is 1 (current can flow)
- ▶ Apply high voltage to control gate
- ▶ Run current through channel
- ▶ "Hot" electrons jump through insulation to floating gate

Erasing NOR Flash
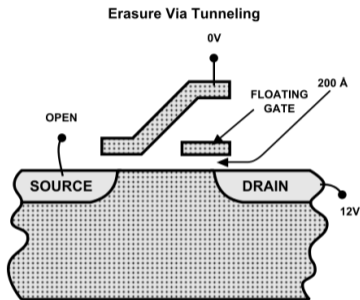
- Apply reverse voltage to control gate
- Disconnect source
- Electrons will now tunnel off floating gate into drain

# Wear-Out

- Some electrons get stuck in oxide during programming
- Add to electric field from floating gate (even if no charge present)
- Eventually becomes impossible to erase effectively

Multilevel Cells (MLC)

- ► Classic flash stores charge or not: zero or one
- ► Possible to store different charge quantities
  - ► Sense varying current levels
  - ► Can translate back into multiple bits
  - ► Current limit is eight levels $\equiv$ three bits
- ► Obvious density improvement
- ► Slower to read and write
- ► Poorer reliability
- ► Modern chips often combine single-level cells (SLC) for speed with MLC for density

NOR Flash
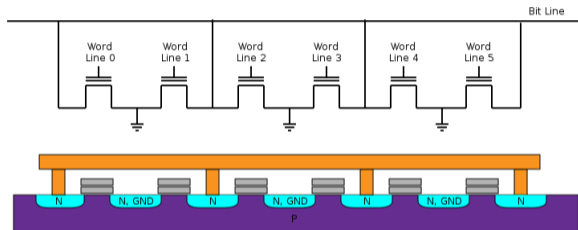
- ▶ All bit lines tied together
- ▶ Readout voltage placed on exactly one word line
- ▶ If "0" stored, nobody conducts
- ▶ If "1" stored, bit line is shorted to ground
  - ▶ Works like NOR of word lines

NAND Flash
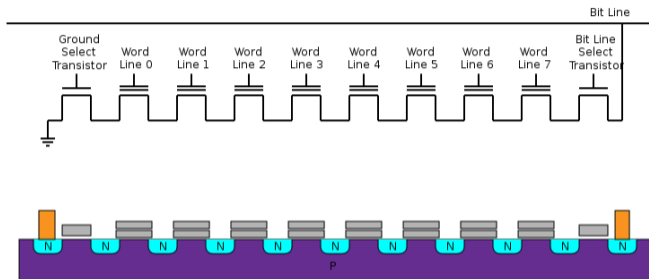
- ▶ Extra-high voltage placed on all but one word line
  - ▶ All will conduct
- ▶ Remaining line gets "just barely" voltage
  - ▶ If programmed, will conduct
- ▶ Lower number of bit & ground lines means better density
- ▶ Programming via tunnel injection, erase via tunnel release

Comparison of NOR and NAND

NOR flash:

- ► Lower density
- ► Usually wired for true random read access
- ► Wired to allow writing of individual cells
- ► Erase in blocks of 64-256 KB

NAND flash:

- ► Cells take about 60% of NOR space
- ► More space saved by block-read wiring
- ► Writing ("programming") is in page-sized chunks of 0.5-4 KB
- ► Erase in blocks of 16-512 kB
- ► Extra bits (more individually accessible) to provide ECC and per-page metadata
- ► OK to have bad blocks

A Sample NAND Chip

Samsung K9F8G08U0M (1G$\times$8)

- ▶ Each page is 4K bytes + 128 extra
- ▶ One block is 64 pages
- ▶ Entire device is 8448 Mbits
- ▶ 5-cycle access: CAS1, CAS2, RAS1, RAS2, RAS3
  - ▶ Eight address bits per cycle
  - ▶ CAS is 13 bits + 3 for future
  - ▶ RAS is 18 + 6 for future
  - ▶ Spare bits mean can later put bigger device into same circuit design
- ▶ On RAS3, loads 4K + 128 into *Page Register*

Chip Commands

Samsung K9F8G08U0M accepts 16-bit commands, such as:

- ▶ Reset
- ▶ Read
- ▶ Block Erase
- ▶ Page Program
- ▶ Read Status
- ▶ Read for Copy Back
- ▶ Copy-Back Program

"Two-plane" commands available for overlapped speedup

Random programming prohibited—but can go back and change metadata

Chip Timing

For Samsung K9F8G08U0M:

- ► Block erase: 2ms (probably not accurate to μs level)
- ► Program: 700μs
- ► Read page to buffer: 25μs
- ► Read bytes: 25ns per byte

Bottom line:

- ► $25\mu s + 4096 \times .025 = 25 + 102.4 = 127.4\mu s$ to read a page
  - $= 32.15$ MB/s data rate
- ► $102.4\mu s + 700 = 802.4\mu s$ to write page if already erased
  - ► Otherwise extra $31.25\mu s$ (amortized) to erase
  - ► Writing is $\approx 6.3 - 6.5\times$ slower than reading

Chip Timing

For Samsung K9F8G08U0M:

- ▶ Block erase: 2ms (probably not accurate to μs level)
- ▶ Program: 700μs
- ▶ Read page to buffer: 25μs
- ▶ Read bytes: 25ns per byte

Bottom line:

- ▶ $25\mu s + 4096 \times .025 = 25 + 102.4 = 127.4\mu s$ to read a page
  - = 32.15 MB/s data rate
- ▶ $102.4\mu s + 700 = 802.4\mu s$ to write page if already erased
  - ▶ Otherwise extra 31.25μs (amortized) to erase
  - ▶ Writing is $\approx 6.3 - 6.5\times$ slower than reading

BUT 2ms latency if nothing currently erased.

Comparison to Disk Timing

For 3-TB Seagate Barracuda XT (3.5-inch):

- ▶ Average latency: 4.16 ms (7200 RPM)
- ▶ Average seek time: 8.5 ms (read), 9.5 ms (write)
- ⇒ 12.66 ms to read one random page
- ▶ Sustained transfer rate: 149 MB/s = 27.5μs per 4K bytes

Bottom line: 12.66 ms to read one random page (ouch!)

- ▶ $99.4\times$ slower!
- ▶ But sequential reads $4.66\times$ faster than flash chip
- ▶ Sequential writes are $\approx 30\times$ faster

Comparison to Disk Timing

For 3-TB Seagate Barracuda XT (3.5-inch):

- ▶ Average latency: 4.16 ms (7200 RPM)
- ▶ Average seek time: 8.5 ms (read), 9.5 ms (write)
- ⇒ 12.66 ms to read one random page
- ▶ Sustained transfer rate: 149 MB/s = 27.5μs per 4K bytes

Bottom line: 12.66 ms to read one random page (ouch!)

- ▶ $99.4\times$ slower!
- ▶ But sequential reads $4.66\times$ faster than flash chip
- ▶ Sequential writes are $\approx 30\times$ faster
- ▶ **But** can wire flash chips in parallel to increase bandwidth

Issues in Using Flash for Storage

- ▶ Pre-erasing blocks
- ▶ Wear leveling
- ▶ Clustering blocks for group writing
- ▶ Efficient updates
- ▶ ECC and bad-block mapping

Issues in Simulating a Disk

- ► Can't tell what pages are live
- ► Expected to allow random updates
- ► Some blocks (e.g., FAT, inode table) much hotter than others

General Solution: Flash Translation Layer

- All flash "drives" have embedded μprocessor (usually 8051 series)
- Give block-numbered interface to outside world
- Hold back some memory (e.g., 5GB drive pretends to be 4GB)
- Map externally visible blocks to internal physical ones
- Use metadata to track what's live, bad, etc.

Problems in FTLs

- ▶ Wear leveling (what if most blocks are read-only?)
  - ▶ Solution: must sometimes move RO data
- ▶ File system wants to rewrite randomly
  - ▶ Solution: group newly written blocks together regardless of logical address
  - ▶ Called "Log-Structured File System" (LFS)
    - ▶ (We'll read that paper later...)
- ▶ Unused block might or might not be live
  - ▶ Solution: only reclaim block when overwritten
  - ▶ Solution: know that it's FAT and reverse-engineer data as it's written

A Better Way

- ▶ Pretending to be a disk is just plain dumb
- ▶ When disks came out, we didn't make them look like punched cards
  - ▶ Well... mostly
- ▶ If filesystem designed for flash, don't need FTL
  - ▶ Problem: need entirely new interface
  - ▶ So far, manufacturers reluctant (chicken and egg)
  - ▶ Apple has now done it in MacBook Air (advantage of making both hardware and software)
- ▶ Some filesystems designed just for flash: YAFFS, JFFS2, TrueFFS, etc.
- ▶ We can expect continued development

The Bad News

- Feature-size limit is around 20 nm
- We're hitting that just about now!
- Some density improvement from MLC, maybe 3-D stacking
- This limit might kill flash as a disk replacement

Other Options

Flash isn't the only choice:

- ▶ Phase-change memory (PRAM or PCRAM)
- ▶ Magnetic RAM (MRAM)
- ▶ ???

New technologies offer:

- ▶ Read/write times slightly slower than DRAM
- ▶ Slower (or no) wear-out
- ▶ Longer storage life without refresh
- ▶ Byte addressability
  - ▶ What happens when filesystems are just like memory?
  - ▶ Active current research area