CS 135: 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

![Graph showing the cost of storage media from 1980 to 2010. The graph compares the cost of Paper/Film, Hard Disk, Flash, and Digital Photography Boom. The Y-axis represents the cost in dollars per megabyte ($/MB), and the X-axis represents the year from 1980 to 2010. The cost of Paper/Film remains relatively constant, while the costs of Hard Disk, Flash, and Digital Photography Boom decrease over time.]
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 line let fuse be selected
- 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 (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
- Reusability dropped cost
EEPROM

- EEPROM (Electrically Erasable PROM) used thinner oxide layer
- Introduced ca. 1983
- High voltage could erase without UV
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 charge
Programming NOR Flash

- 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
Wear-Out

- Some electrons get stuck in oxide during programming
- Add to electric field
- Eventually becomes impossible to erase effectively
Multilevel Cells

- 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
  - Typically four levels, two bits
- Obvious density improvement
- Slower to read and write
- Poorer reliability
NOR Flash

- All bit lines tied together
- Readout voltage placed on exactly one word line
- If “1” stored, nobody conducts
- If “0” 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 to provide ECC and per-page metadata
- OK to have bad blocks
Samsung K9F8G08U0M (1G×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
- RAS 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 $\mu$s level)
- Program: 700$\mu$s
- Read page to buffer: 25$\mu$s
- Read bytes: 25ns per byte

Bottom line: $25 + 4096 \times 0.025 = 25 + 102.4 = 127.4\mu$s to read a page, $102.4 + 700 = 802.4$ to write—if already erased; otherwise extra $31.25\mu$s (amortized) to erase
For Samsung K9F8G08U0M:

- Block erase: 2ms (probably not accurate to $\mu$s level)
- Program: 700$\mu$s
- Read page to buffer: 25$\mu$s
- Read bytes: 25ns per byte

Bottom line: $25 + 4096 \times 0.025 = 25 + 102.4 = 127.4 \mu s$ to read a page,
$102.4 + 700 = 802.4$ to write—if already erased; otherwise extra
$31.25 \mu s$ (amortized) to erase

**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 = 27.5\(\mu\)s per 4K bytes

Bottom line: 12.66 ms to read one random page (ouch!), but sequential reads 5X faster than SSD, and sequential writes 30X faster
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 blocks are live
- Expected to allow random updates
- Some blocks (e.g., FAT, inode table) much hotter than others
All flash “drives” have embedded µprocessor (usually 8051 series)
Give block-numbered interface to outside world
Hold back some memory (e.g., 5G drive pretends to be 4G)
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)
- 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)
- Some filesystems designed just for flash: YAFFS, JFFS2, TrueFFS, etc.
- Can expect further 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 might kill flash as a disk replacement