ECE 571 – Advanced Microprocessor-Based Design Lecture 23

Vince Weaver

https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

1 November 2024

Announcements

- HW7 was due
- Midterm not graded yet
- Useful readings:
  - "A performance and power comparison of modern high-speed DRAM arch" from MEMSYS 2018
  - "DRAM Refresh Mechanisms, Penalties, and Trade-offs" Bhati, Chang, Chishti, Lu and Jacob. IEEE transactions on Computers, 2016.

HW#6 Review

HW#6 – Intel Prefetch Background

- Intel/x86 sw prefetch
  - prefetcht0 – prefetch all levels
  - prefetcht1 – prefetch 2nd and 3rd level
  - prefetcht2 – same as prefetcht1
  - prefetchnta – non-temporal (when would you use this)?
  - prefetch – (3dnow) L1
  - prefetchw – (3dnow) prefetch, plan to write (MESI)
  - movnti – non-temporal move (don't cache)
- Intel hw prefetch

- Can disable all individually (chicken switch?)
- DCU + IP – fetch to L1
- Spatial (next line) – L2
- Stream – L3 (and L2 if L2 not too busy)

HW#6 – SW Prefetch Numbers

- `objdump --disassemble-all ./bzip2 | grep prefetch | wc -l`
  ```
   0 of them
  objdump --disassemble-all ./bzip2.swprefetch | grep prefetch
  ./bzip2.swprefetch:     file format elf64-x86-64
      2fa3: 0f 18 0a              prefetcht0 (%rdx)
      3f73: 0f 18 0a              prefetcht0 (%rdx)
     10be3: 0f 18 0a              prefetcht0 (%rdx)
     118d4: 0f 18 0f              prefetcht0 (%rdi)
      1c32: 0f 0d 7b 00           prefetch 0x0(%rbx)
      1c4c: 0f 0d 7b 00           prefetch 0x0(%rbx)
  ```

- `objdump --disassemble-all ./equake_l | grep prefetc`
  ```
  (nothing)
  objdump --disassemble-all ./equake_l.swprefetch | grep prefetch
      1d99: 0f 18 0e              prefetcht0 (%rsi)
  ```

```
1fd2: 0f 18 4d 00            prefetcht0 0x0(%rbp)
49d5: 41 0f 18 0c 24         prefetcht0 (%r12)
4bb5: 41 0f 18 0c 24         prefetcht0 (%r12)
50d7: 0f 18 0b               prefetcht0 (%rbx)
50da: 0f 18 0f               prefetcht0 (%rdi)
```

- C library has 293 sw prefetch instructions (most likely in inline assembly for memcpy and the like?)
- Automated by C compiler was all prefetcht0
- Glibc hand-optimized had all cominations, including prefetchnta (for memset or similar?)
- Can use `addr2line` to try to find out where the compiler

is inserting SW prefetch, though for equake didn't seem to work

HW#6 – Results

- BZIP

		l2-cache-misses	prefetches	time
1a:	bzip2:	34.1%	166M	3.8s
2a:	SW prefetch:	33.6%	170M	3.7s
5a:	HWdisable	43%	76k	4.8s
5a:	HWdisable+Sw	43%	76k	3.9s

- Equake

		l2-cache-misses	prefetches	time
3a:	equake_l:	20%	50B	29.5s
4a:	equale_l swpref	20%	50B	29.3s
5a:	hwdisable	66%	9.2M	62s
5a:	hwdisable swpref	66%	9.1M	68s

- Summary: disabling prefetch hurt, dramatically so on equake.
  Unclear what exactly the prefetch perf counter is measuring
  Enabling SW prefetch does not seem to do much, even with HW prefetch disabled.

- Why? Lots of possible reasons. compiler bug. hardware bug. hardware engineers not enable SW prefetch (is it incorrect to ignore?) other.

Static RAM (SRAM) Review

- Used on chip: caches, registers, etc. Made in same process as CPU
- 6 transistors (or 4 plus hard-to make resistors with high static power)
- Cross-coupled inverters
  - For read, precharge both bitlines. Raise wordline.
  - Lots of capacitance so hard to swing whole way, so sense amp which amplified the small voltage shift
  - For write, set bitline and not-bitline, set wordline.

Overpowers inverters

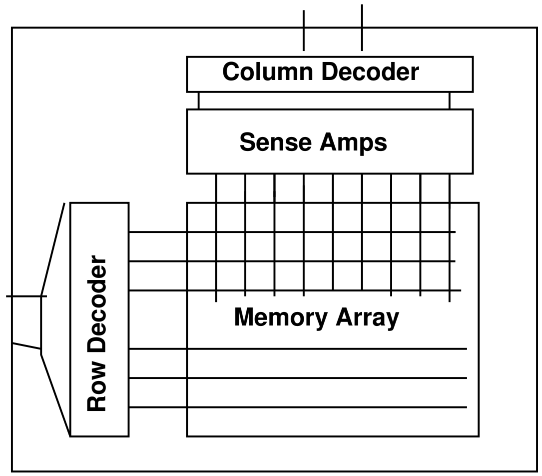
- Clocked or no, clocked saves power (synchronous vs asynchronous. synchronous can be pipelined and only operate sense amp when needed)
- Bitlines might be braided to avoid noise

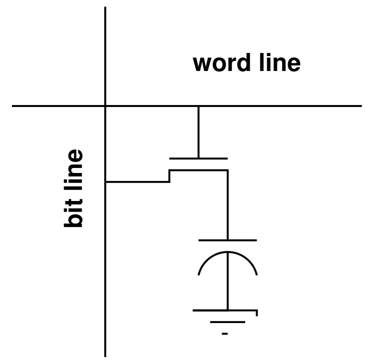
Why not have large SRAM

- Low power at low frequency, but more power at high freq
- It is harder to make large SRAMs with long wires
- It is a lot more expensive while less dense (Also DRAM benefits from the huge volume of chips made)
- Leakage for large data structures
- Price: (November 2022)
  - 16Mbit (2Mbyte) $18.70, $9000/GB
  - 8GB DDR4 DIMM, $54, $6/GB

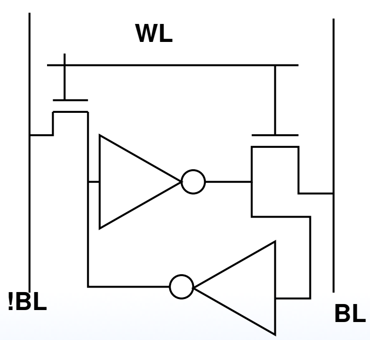
Diagrams

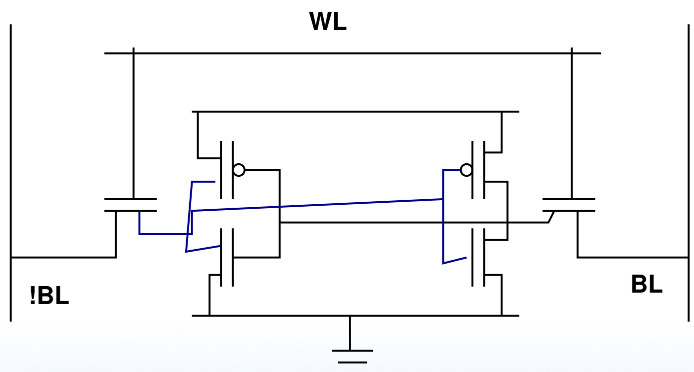
DRAM





SRAM





DRAM

- Single transistor/capacitor pair. (can improve behavior with more transistors, but then less dense)
- In 90nm process, 30fF capacitor, leakage in transistor 1fA. Can hold charge from milliseconds to seconds.
- DRAMs gradually lose charge, need to be refreshed.
- Need to be conservative.
  Refresh each row every 32 or 64ms
  (if 8192 rows, then 64ms/8192 is 7.8us)
- DRAM read is destructive, always have to write back

- Interesting article on history of 4116 16k RAM chip `https://www.righto.com/2020/11/reverse-engineer` `html`

DRAM Actions

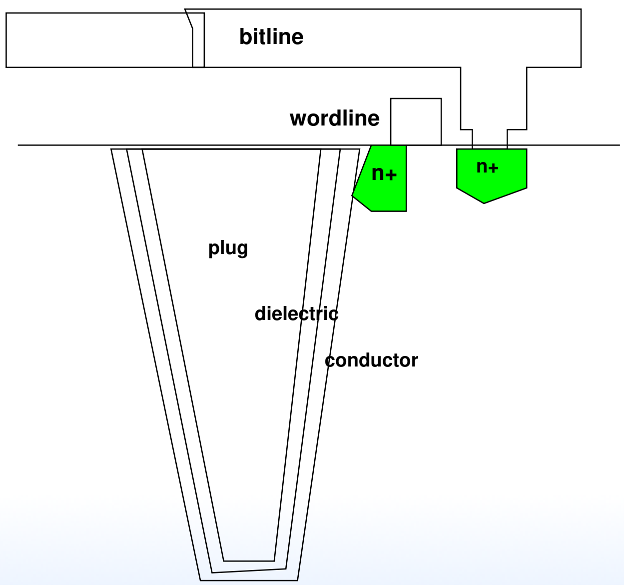
- *Precharging* the bitlines
- *Activating* entire row (discharge capacitors into the bitlines)
- *Sensing* the voltage change (sense amplifiers)
- *Reading/Writing*

DRAM Notes

- Two parts of modern DRAM, core bit-arrays and data I/O
- Core bit-arrays have not really increased in speed in years, always in 10s of nanoseconds
  issues are physical: resistance, capacitance
- Improvements come by parallelizing things, reading out lots of bits at once, buffering them, then clocking out
- Summary: hard to improve latency (random-access), easier to improve bandwidth

Low Level

- Planar (old)
- Stacked Capacitors (transistors are below)
- Trench Capacitors (transistors are above)



Memory Packaging

- DIMM – dual inline memory module
- Why dual? Replaced SIMMs
- SIMM had pins on both side but just duplicated signal
- SIMM also 32-bit, when modern systems moved to 64-bit bus (P5 pentium) you needed to have SIMMs in pairs
- DIMMs 64-bit memory bus and you only needed to add one module at a time

DIMMs (desktop/server)

- How many chips on DIMM? 8? 9?
  9 usually means ECC/parity
- Chips x1 x4 x8 bits, how many get output at a time. Grouped together called a "bank"
- Banks can mask latency, sort of like pipelining. If takes 10ns to respond, interleave the request.
- DIMM can have independent "ranks" (1 or 2 per DIMM), each with banks, each with arrays. (Rank is like a full additional 64-bit memory dimm enabled with chip-select,

but on same package)

- Layout, multiple mem controllers, each with multiple channels, each with ranks, banks, arrays
- Has SPD "serial presence detect" chip that holds RAM timings and info. Controlled by smbus (i2c)
- MCR DIMM – really big DIMMs for servers. Tall, 80 chips on them. 256GB modules, allow 3TB to 6TB on servers

Laptop Memory

- SODIMM – smaller form factor for laptops "small outline"
- DELL CAMM – controversial replacement for SODIMM CAMM2 is now a JEDEC standard?
- LPCAMM2 from Micron? LPDDR5X-9600.
- LPCAMM2 module is cheaper than four sticks of ddr5. Faster too LPDDR usually has to be soldered to motherboard (Low voltage, short traces) LPCAMM2 compression attached (screwed down) for

better connection but allows expansion in laptops

Refresh (more on this later)

- Need to read out each row, then write it back. every 32 to 64ms
- Old days; the CPU had to do this. Slow
  Digression: what the Apple II does
- Newer chips have “auto refresh”

Low-Level Memory Bus

- JEDEC-style. address/command bus, data bus, chip select
- Row address sends to decoder, activates transistor
- Transistor turns on and is discharged down the column rows to the sense amplifier which amplifies
- The sense amplifier is first "pre-charged" to a value halfway between 0 and 1. When the transistors are enabled the very small voltage swing is amplified.
- This goes to column mux, where only the bits we care

about are taken through

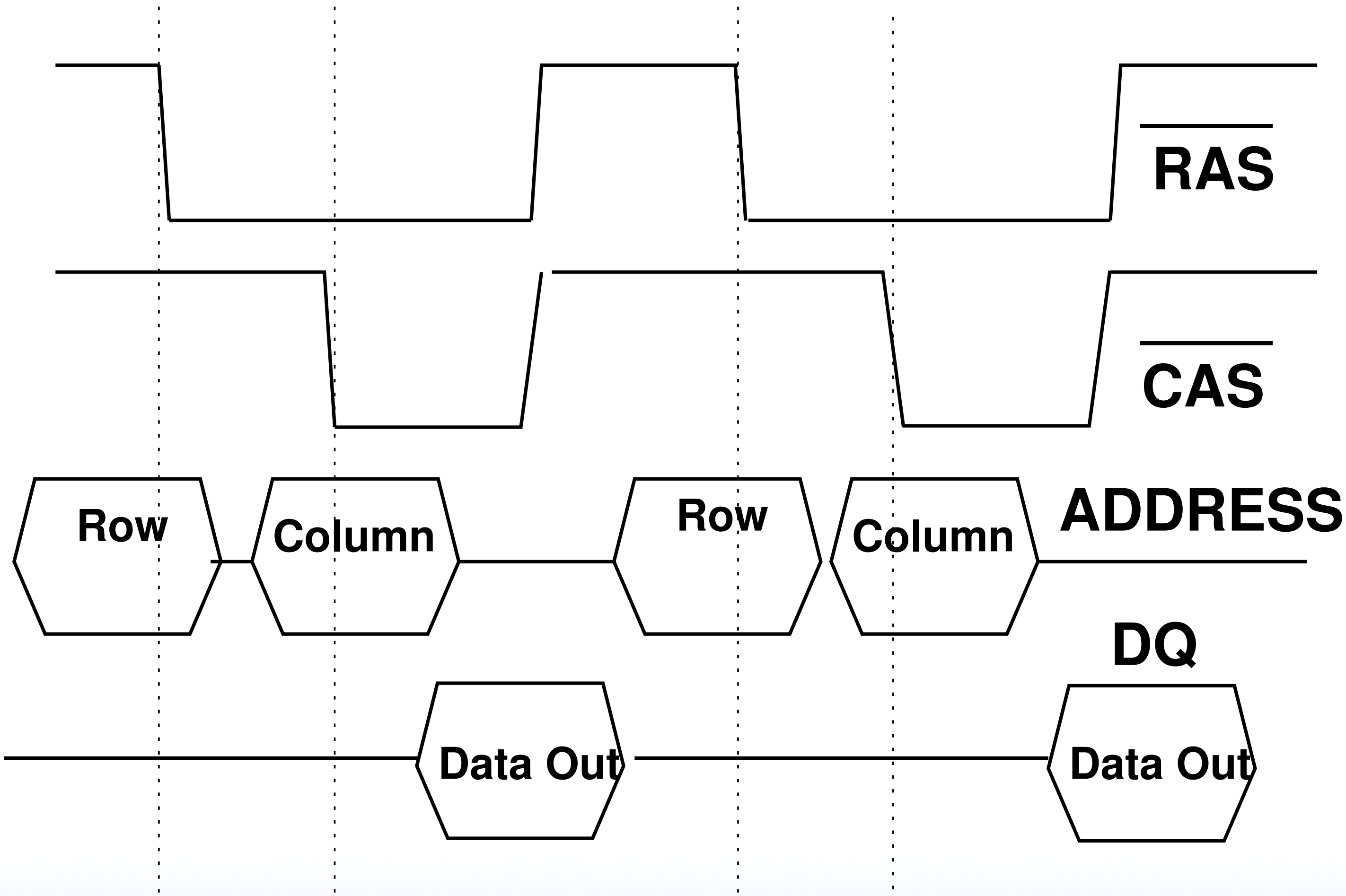
Memory Access

- CPU wants value at address (cache miss?)
- Passed to memory controller
- Memory controller breaks into rank, bank, and row/column
- Proper bitlines are pre-charged
- Row is activated, then $\overline{RAS}$, row address strobe, is signaled, which sends all the bits in a row to the sense

amp. can take tens of ns.

- Then the desired column bits are read. The $\overline{CAS}$ column address strobe sent.
- Again takes tens of ns, then passes back to memory controller.
- Unlike SRAM, have separate CAS and RAS? Why? Original DRAM had low pincount.
- Also a clock signal goes along. If it drives the device it's synchronous (SDRAM) otherwise asynchronous

Async DRAM Timing Diagram



Memory Controller

- Formerly on the northbridge
- Now usually on same die as CPU