1 Memory

- Cycle Time - Minimum time between unrelated accesses to memory
- Access Time - Time between a read is requested and when a word arrives.

1.1 SRAM

Static Random Access Memory. Typically used in caches, but not main memory. No refresh needed due to static nature. Access time is very close to cycle time. Typically uses 6 transistors/bit to avoid data disruption. Low power consumption at idle.

1.2 DRAM

Dynamic Random Access Memory.

The dynamic nature of circuits in DRAM requires data to be written back after being read. This is what causes disparity between access time and cycle time. Single transistor used per bit.

Due to electronic properties, reads from a DRAM row destroy information. This means that it must be written back from the ??row buffer??. Bits must also be refreshed to prevent leakage (this usually happens in ms)

SDRAM

There was an asynchronous interface for doing multiple column accesses on a single row that required an async interface. SDRAM adds a clock signal so repeated transfers would not bear overhead. Multiple banks added (with overlapping) allowing multiple row buffers on one chip. Requests being sent to a new bank are subject to additional delays due to the need to 'open' the bank.

DDR DRAM

Double data rate. Can transfer data on rising and falling edges of clock.

1.2.1 GDDR DRAM

Graphics DDR DRAM. Sacrafices latency/speed for increased bandwidth. Good for graphics-intensive tasks. GDDR5 was based on DDR3 specifications.

1.3 Flash Memory

A type of EEPROM (Electronically erasable programmable read only memory). Read only but can be erased. This is the technology used in most SSDs. Reads are sequential and read an entire 512 to 4096 byte page. Data must be erased before being written over. Much slower than DRAM. Keeps data even when power is lost.

1.4 Phase-Change Memory

Research toy. Uses heating element to change conductivity element of substrate.

1.5 Enhancing Dependability in memory systems

Memory systems have 2 kinds of faults: Hard Errors (occur during fabrication or circuit cahnge during operation such as failure of a cell, mitigated by adding spare rows to tolerate failure) and Soft Errors/Transient Faults/Dynamic Errors (occur during normal operation whenever changes happen in a cell's contents i.e. bitflip). Errors can be detected with parity bits and fixed by error correcting codes.

2 Cache Improvement Strategies

- Smaller and less-associative high level caches
- Add banks to cache so that access does not activate whole cache
- Keep extra bits in cache to predict the next block access in cache (similar to branch prediction)
- Pipeline access and multibanked caches can increase bandwidth
- Non blocking caches allow for out-of-order execution computers to not stall waiting for cache
- On a miss, request the missed word first so processor isn't waiting as long (Critical word first) OR On a miss, fetch in normal order and when missed word is rec'd send to processor (Early restart)
- Compiler optimizations can fix things to have better spatial/temporal locality
- Prefetching: predict next accesses and fetch them into cache before requested
- Compiler Controlled prefetching: compiler inserts prefetch instructions
- Have an L4 cache with high bandwidth memory technology

3 Virtual Memory

Basically what was learned in OS.

4 Cache Coherency Protocols

Used to synchronize the data across caches.

4.1 MSI

Three states: Invalid, Modified, and Shared

- Invalid - $v = 0, d = 0$ treated as not in cache
- Modified - $v = 1, d = 1$ we 'own' this data and can make writes
- Shared - $v = 1, d = 0$ this data is up to date but we can't write to it as others have it

[shape=circle,draw=black] (Invalid) at (0,3) Invalid; [shape=circle,draw=black] (Modified) at (0,0) Modified; [shape=circle,draw=black] (Shared) at (3,3) Shared;

[-¿] (Invalid) edge node[left] Local Write (Bus) (Modified); [-¿] (Invalid) edge node[left] Local Read (Bus) (Shared); [-¿] (Invalid) edge node[left] Snoop Read (Invalid); [-¿] (Invalid) edge node[left] Snoop Write (Invalid);

[-¿] (Modified) edge node[left] Local Write (Modified); [-¿] (Modified) edge node[left] Local Read (Modified); [-¿] (Modified) edge node[left] Snoop Read (WB/InvalRQ) (Shared); [-¿] (Modified) edge node[left] Snoop Write (WB/InvalRQ) (Invalid);

[-¿] (Shared) edge node[left] Local Write (BusInval) (Modified); [-¿] (Shared) edge node[left] Local Read (Shared); [-¿] (Shared) edge node[left] Snoop Read (Shared); [-¿] (Shared) edge node[left] Snoop Write (Invalid);

4.2 MESI

Four states: Modified, Exclusive, Shared, and Invalid. This extension reduces amount of communication on bus.

- Exclusive - Bock is present in 1 class and is clean. Can write without invalidating