Dynamic RAM

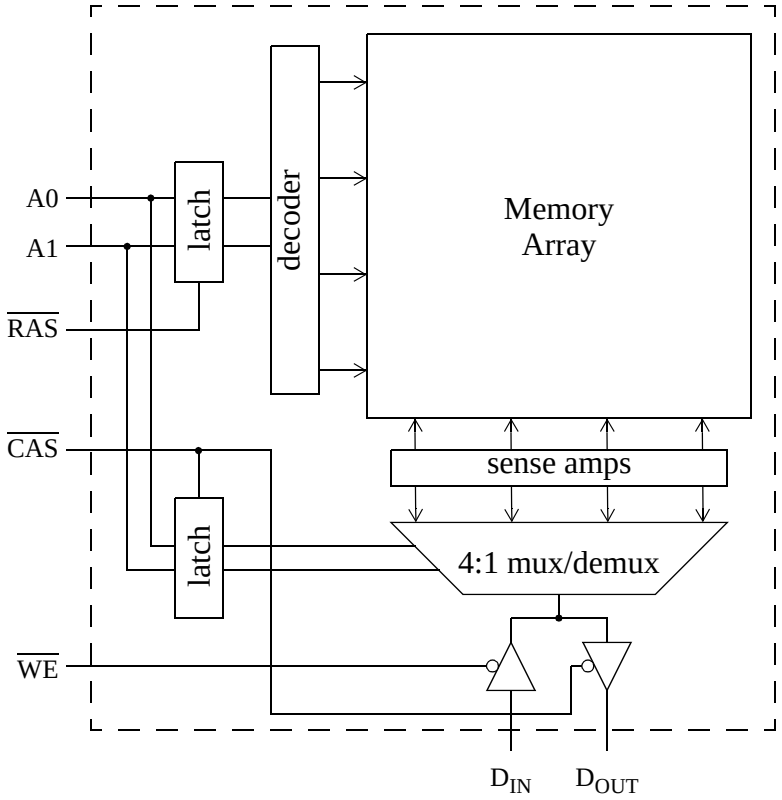
Dynamic RAM (DRAM) is the highest density, lowest cost memory currently available. For these reasons it is univerally used in any microprocessor-based system that requires more than a small amount of non-volatile writable storage.

- One transistor per cell (drain acts as capacitor)
- Very small charges involved
  - bit lines must be *precharged* to detect bit values
  - voltage swing on bit lines is small; *sense amp* required to convert to logic levels
  - reads are *destructive*; DRAM devices internally write data back on read
  - leakage current can flip 1s to 0s: values must be *refreshed* (rewritten) every few milliseconds or data will be lost
- To reduce package cost and size, DRAM devices minimize pin count by:
  - using narrow logical configurations (x1, x4)
  - multiplexing the internal row and column addresses on the same pins

DRAM Interfacing

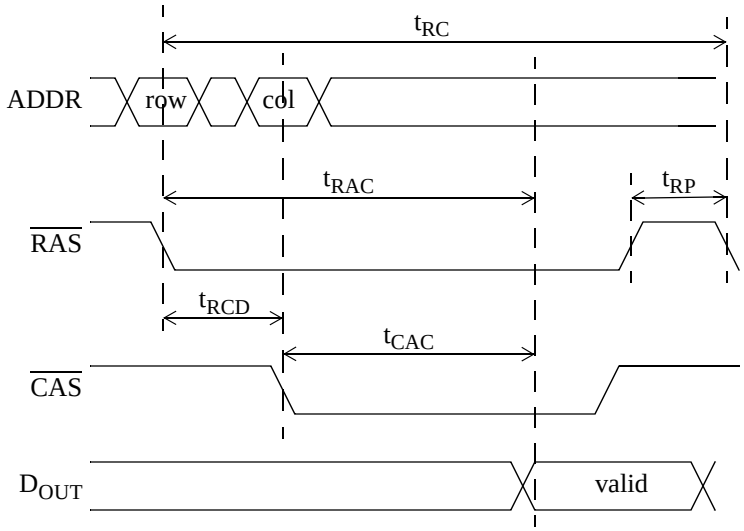
Internally, DRAMs are much like other memories, except:

- $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ signals strobe latches row and column halves of multiplexed address
- $\overline{\text{CAS}}$ may also serve as output enable
- Most x1 devices have separate input and output data pins



DRAM Timing: Reads

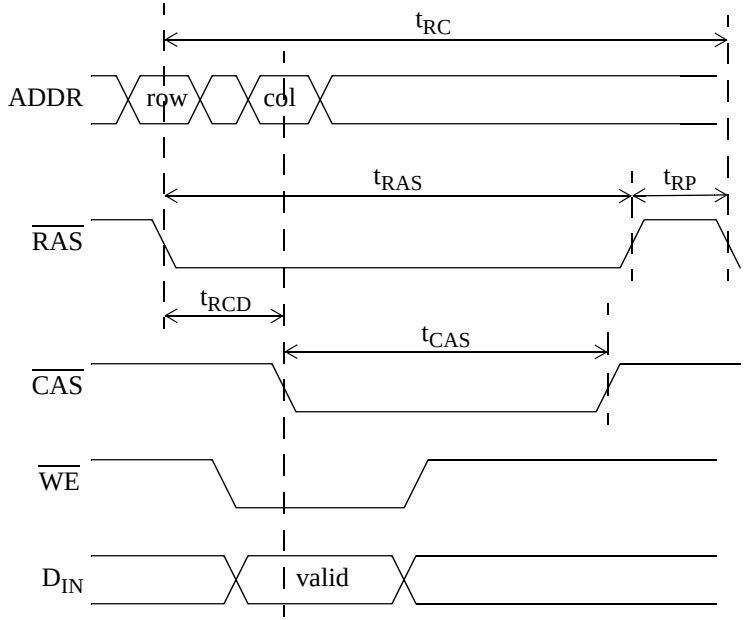
- The falling edges of RAS and CAS strobe the address bits into the row and column latches, respectively. These must be separated by at least t_{RCD}.
- As with other memories, multiple access times are specified, and the time to valid data out will depend on which is the critical path. For DRAMs, there are two access times, t_{RAC} and t_{CAC}, for access time from valid row address and valid column address, respectively.



- Setup and hold times for both row and column addresses exist but are not shown.
- The read cycle time (t_{RC}) is typically much larger than the access time due to the required precharge time t_{RP} (*not* drawn to scale).

DRAM Timing: Writes

Write timing is similar. If $\overline{WE}$ is asserted on the falling edge of $\overline{CAS}$, data is written from D_{IN} instead of being read to D_{OUT}. Note that this is different from SRAMs, which perform writes at the end of the cycle (rising edge of $\overline{WE}$).



- Most timing parameters are identical to the read cycle. t_{RAS} and t_{CAS} are minimum pulse widths that also apply to the read cycle but were left out of that diagram for clarity.
- Required setup and hold times on D_{IN} and $\overline{WE}$ not shown.

Refresh

Each cell must be refreshed every few milliseconds to avoid losing data. Whenever a row is read, the sense amps automatically write back entire row, so we only need to access every row once during the refresh interval.

- Do one row at a time (not in big burst) to avoid tying up DRAM for long period
- Typically done in hardware using counter (to track next row index) and timer (to initiate one refresh)
- Example: Hitachi 64Mbit DRAM requires 8192 refreshes every 64 ms. Access one row every (64 ms/8192) = 7.8 μs.
- Need not provide column address: "RAS-only refresh"
- Can also insert refresh cycle at end of unrelated read access; if CAS is not deasserted, read data remains valid: "hidden refresh"
- Some DRAMs have internal counter; system needs only to indicate when to refresh "next" row by asserting CAS *then* RAS (opposite of regular access): "CAS-before-RAS (CBR) refresh"

Other solutions?

Interface Optimizations

Problem: DRAM bandwidth (bits/second) has not kept up with CPU speed increases over the years.

Simple observation: reading out entire row and throwing out all but one bit is inefficient. Wider chips (x4, x8, x16) help some, but only as a stopgap. A number of DRAM enhancements let systems read out many bits from each row:

nibble mode: pusling CAS 4 times w/o deasserting RAS gives 4 adjacent bits (only first column address used)

fast page mode (FPM): pulsing CAS w/o deasserting RAS takes new column address (do as often as desired)

static column: like FPM, but no need to deassert CAS (latch is transparent); just change address and it flow through

extended data out (EDO): like FPM, but data stays valid after deasserting CAS (limited pipelining, lets you pump in column addresses faster)

Synchronous DRAM (SDRAM)

As the name implies, a synchronous DRAM has a clock input, and all signals are defined with respect to clock edges.

- After providing row and column addresses, a programmable number of bits can be provided one per clock without any other control inputs
- The next pair of row and column addresses can be provided while the previous access is still completing
- SDRAMs have multiple (e.g., 4) memory arrays (banks) so that independent accesses can be even further overlapped:
  - provide row address for bank 1
  - provide row address for bank 2
  - provide column address for bank 1
  - provide column address for bank 2
  - get data from bank 1
  - get data from bank 2
- Clock speeds of 66, 100, 125 MHz
- Sega Saturn was one of the earliest widespread users
- Becoming common in PCs now

Rambus

Rambus is a revolutionary new DRAM interface that combines multiple banks per chip with a high-speed bus interface. Two earlier generations ("Rambus" and "Concurrent Rambus") have been superseded by the latest "Direct Rambus" protocol. Intel is pushing this to be the high-end PC memory technology by 2000.

- 400 Mhz clock, data transferred on both edges
- 16-bit bus
- result: 1.6 Gbytes per second on one "channel"
- high-end servers will have multiple channels
- 16 banks per chip: if enough accesses go to different banks, a single chip can provide the full 1.6 GB/s
- nice solution to *granularity* problem: what is minimum memory system size/increment given very dense chips?
- SDRAM vendors using two-edge trick in new "dual data rate" (DDR) SDRAMs
- Sync-Link DRAM (SLDRAM) uses similar ideas as Rambus (with a few key changes, mostly at the electrical level), but is standards-based instead of proprietary