DRAM Technology

- Data stored as a charge in a capacitor
  - Single transistor used to access the charge
  - Must periodically be refreshed
    - Read contents and write back
    - Performed on a DRAM “row”
Advanced DRAM Organization

- Bits in a DRAM are organized as a rectangular array
  - DRAM accesses an entire row
  - Burst mode: supply successive words from a row with reduced latency
- Double data rate (DDR) DRAM
  - Transfer on rising and falling clock edges
- Quad data rate (QDR) DRAM
  - Separate DDR inputs and outputs
# DRAM Generations

<table>
<thead>
<tr>
<th>Year</th>
<th>Capacity</th>
<th>$/GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980</td>
<td>64Kbit</td>
<td>$1500000</td>
</tr>
<tr>
<td>1983</td>
<td>256Kbit</td>
<td>$500000</td>
</tr>
<tr>
<td>1985</td>
<td>1Mbit</td>
<td>$200000</td>
</tr>
<tr>
<td>1989</td>
<td>4Mbit</td>
<td>$50000</td>
</tr>
<tr>
<td>1992</td>
<td>16Mbit</td>
<td>$15000</td>
</tr>
<tr>
<td>1996</td>
<td>64Mbit</td>
<td>$10000</td>
</tr>
<tr>
<td>1998</td>
<td>128Mbit</td>
<td>$4000</td>
</tr>
<tr>
<td>2000</td>
<td>256Mbit</td>
<td>$1000</td>
</tr>
<tr>
<td>2004</td>
<td>512Mbit</td>
<td>$250</td>
</tr>
<tr>
<td>2007</td>
<td>1Gbit</td>
<td>$50</td>
</tr>
</tbody>
</table>

![Graph showing the cost per GB for DRAM generations from 1980 to 2007](image)
DRAM Performance Factors

- **Row buffer**
  - Allows several words to be read and refreshed in parallel

- **Synchronous DRAM** (SDRAM, Operates in Synchrony with external clock)
  - Allows for consecutive accesses in bursts without needing to send each address
  - Improves bandwidth

- **DRAM banking** (Interleaves Memory Banks)
  - Allows simultaneous access to multiple DRAMs
  - Improves bandwidth
Increasing Memory Bandwidth

- **4-word wide memory**
  - Miss penalty = $1 + 15 + 1 = 17$ bus cycles
  - Bandwidth = $16$ bytes / $17$ cycles = 0.94 B/cycle

- **4-bank interleaved memory**
  - Miss penalty = $1 + 15 + 4 \times 1 = 20$ bus cycles
  - Bandwidth = $16$ bytes / $20$ cycles = 0.8 B/cycle
Interleaved Memory Banks (Διαφύλλωση Μνήμης)

Requests (Addresses)

(Optional Request Queues)

In case of Bank Conflicts

Data back

Addrs:
00
10
20
E0
FO

Addrs:
04
14
24
E4
F4

Addrs:
08
18
28
E8
F8

Addrs:
0C
1C
2C
EC
FC

E.g.: 100ns per Bank Access, one new Request every 25ns

"Split Transactions" on request/reply "bus"... pipelining

...multiple transactions interleaved in time
"Split Transactions":
- do NOT "hold" the "bus" exclusively between request and reply;
  release it for other transactions.
- need Transaction ID with every request and its corresponding reply,
  especially in case of replied out-of-order.
Disk Storage

- Nonvolatile, rotating magnetic storage
Disk Sectors and Access

- Each sector records
  - Sector ID
  - Data (512 bytes, 4096 bytes proposed)
  - Error correcting code (ECC)
    - Used to hide defects and recording errors
  - Synchronization fields and gaps
- Access to a sector involves
  - Queuing delay if other accesses are pending
  - Seek: move the heads
  - Rotational latency
  - Data transfer
  - Controller overhead
Disk Access Example

- Given
  - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk

- Average read time
  - 4ms seek time
    + $\frac{1}{2} / (15,000/60) = 2\text{ms}$ rotational latency
    + $\frac{512}{100\text{MB/s}} = 0.005\text{ms}$ transfer time
    + 0.2ms controller delay
    = $6.2\text{ms}$

- If actual average seek time is 1ms
  - Average read time = $3.2\text{ms}$
Disk Performance Issues

- Manufacturers quote average seek time
  - Based on all possible seeks
  - Locality and OS scheduling lead to smaller actual average seek times

- Smart disk controller allocate physical sectors on disk
  - Present logical sector interface to host
  - SCSI, ATA, SATA

- Disk drives include caches
  - Prefetch sectors in anticipation of access
  - Avoid seek and rotational delay
Amortize cost over Large data blocks

- Pipeline
  - Total cost = 5 kWatt
  - Power = 1400 Watt/Kilo

- DRAM
  - Access time = (raw) access time
  - Access rate = n x 500MHz clock
  - DDR timing:
    \[ \frac{T_{ck}}{2} = \frac{2\text{ns}}{2} = 1\text{us} \]

- Distance
  - Cost: \( c_{cost} \approx 10\text{ms} \)
  - Speed: \( c_{speed} \approx 100 \text{MB/s} \)
    \[ 1\text{Byte/sec} = 1\text{ns} \]

- Distance
  - N x 20 km, 10 Gb/s
    \[ \text{new 20 bit:} \frac{20}{200} = 0.1\text{ms} \]
    \[ \text{bit time:} \frac{100}{5} = 20\text{ns} \]
Instruction Set Architecture for I/O

° Some machines have special input and output instructions

° Alternative model (used by MIPS):
  • Input: ~ reads a sequence of bytes
  • Output: ~ writes a sequence of bytes

° Memory also a sequence of bytes, so use loads for input, stores for output
  • Called “Memory Mapped Input/Output”
  • A portion of the address space dedicated to communication paths to Input or Output devices (no memory there)
Memory Mapped I/O

- Certain addresses are not regular memory

- Instead, they correspond to registers in I/O devices
Example: keyboard... if only a Data Register:

What did the user type?

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Data Reg. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>'t'</td>
</tr>
<tr>
<td>0.5</td>
<td>'t'</td>
</tr>
<tr>
<td>1.0</td>
<td>'o'</td>
</tr>
<tr>
<td>1.5</td>
<td>'o'</td>
</tr>
<tr>
<td>2.0</td>
<td>'o'</td>
</tr>
</tbody>
</table>
Example: keyboard... if only a Data Register:

What did the user type?

0.0
read Data Reg. = 't'

0.5
read Data Reg. = 't'

1.0
read Data Reg. = 'o'

1.5
read Data Reg. = 'o'

2.0
read Data Reg. = 'o'

"to"
"too"
"two"
"ttoo"
Processor Checks Status before Acting

° Path to device generally has 2 registers:
  • 1 register says it’s OK to read/write (I/O ready), often called **Control Register**
  • 1 register that contains data, often called **Data Register**

° Processor reads from Control Register in loop, waiting for device to set Ready bit in Control reg to say its OK (0 ⇔ 1)

° Processor then loads from (input) or writes to (output) data register
  • Load from device/Store into Data Register resets Ready bit (1 ⇔ 0) of Control Register
I/O Address Pages must be non-cacheable.

if they were allowed to be cached...

- transitional ("non-coherent") caching does NOT work when other devices (I/O, other proc. cores) access memory independently

- note: write-through is a "half-solution": works for output, but not for input...
I/O/Communication Registers ≠ Normal Memory Semantics (non-shared)

Normal Memory:

- Read always yields the last written value.
- Writes only affect the word being written.

<table>
<thead>
<tr>
<th>Some word:</th>
<th>write x</th>
<th>read x</th>
<th>read x</th>
<th>read x</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I/O/Communication Registers:

- Successive reads from a same location (without any interfering writes from processor) may yield different values.

<table>
<thead>
<tr>
<th>Some word:</th>
<th>read x_1</th>
<th>potentially ≠ read x_2</th>
<th>potentially ≠ read x_3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"Side-Effects"
Memory Consistency

1. Out of order delivery?
2. What if these reside on different banks in an interleaved memory?

- Flag 2: Wait to see if Flag 1 becomes True
- Flag 0: Read data 2
- Flag 1: Read data 3
- Flag 2: Read data 2
- Flag 3: Read data 3

Flag Order Constraint

Time A

Interleaved network

From input, device, or communication processor(s)

Write to slow dram to slow dram

Then read dram to read dram

3.2 Gbps

Memory Banks

Swenelx Muths
What is the alternative to polling?

° Wasteful to have processor spend most of its time “spin–waiting” for I/O to be ready

° Wish we could have an unplanned procedure call that would be invoked only when I/O device is ready

° Solution: use exception mechanism to help I/O. Interrupt program when I/O ready, return when done with data transfer
I/O Interrupt

- An I/O interrupt is like an overflow exceptions except:
  - An I/O interrupt is “asynchronous”
  - More information needs to be conveyed

- An I/O interrupt is asynchronous with respect to instruction execution:
  - I/O interrupt is not associated with any instruction, but it can happen in the middle of any given instruction
  - I/O interrupt does not prevent any instruction from completion
Definitions for Clarification

° Exception: signal marking that something “out of the ordinary” has happened and needs to be handled

° Interrupt: asynchronous exception

° Trap: synchronous exception

° Note: These are different from the book’s definitions.
Interrupt Driven Data Transfer

1. I/O interrupt
2. Save PC
3. Interrupt service addr
4. Read, store, etc.
5. JR

Memory

User program
Interrupt service routine
Questions Raised about Interrupts

° Which I/O device caused exception?
  • Needs to convey the identity of the device generating the interrupt
    Cause register, or Vectored Interrupts

° Can avoid interrupts during the interrupt routine?
  • What if more important interrupt occurs while servicing this interrupt?
  • Allow interrupt routine to be entered again?

° Who keeps track of status of all the devices, handle errors, know where to put/supply the I/O data?
Χρονικό κόστος Διακοπών και Δειγματοληψίας

– Παρ’ ότι μιά εξαίρεση/διακοπή μοιάζει με άλμα, εντούτοις...

• Σχεδόν πάντα αυτά κοστίζουν ~1000 κύκλους ρολογιού (!)
  – ένα μικρό μυστήριο το γιατί ακόμα ισχύει αυτό στη μεγάλη
    πλειοψηφία, παρ’ όλο που πολλοί προσπάθησαν να το μειώσουν
  – μάλλον το άθροισμα κάμποσων παραγόντων, κυρίως:
    – σώσιμο/επαναφορά καταχωρητών διακοπείσας διεργασίας
    – αναζήτηση αιτίας / “housekeeping”
    – αστοχίες κρυφών μνημών και TLB

• Δειγματοληψίες επίσης κοστίζουν: non-cacheable I/O reg.
  – μία τεχνική: όταν διακοπή από real-time clock (~50-120 Hz), τότε
    το Λειτουργικό δειγματοληπτεί πολλές περιφερειακές μαζί
Fast Devices need I/O Buffer — not just a Register

... Amortize the cost of interrupt over many data

example:
(just) 1 Gbit/s
↓
1 bit every 1 ns
↓
32 bits every 32 ns

one "Register"

- Cost to poll status register
  (non-cacheable, off-chip)
  usually ≈ DRAM access
  usually ∼ 100 ns (or more)
- Then read the data register
  similarly ∼ 100 ns
- Cost of interrupt + kernel interrupt handler usually ∼ 1 μs (1000 ns)
Direct Memory Access (DMA)

Alternatives for cacheability:

- DMA onto non-cacheable memory pages ... too slow when processor processes the I/O data
- Flush the cache before/after I/O DMA ... quite expensive operation < total flush?
- Cache-Coherent DMA ← good! → next chapter...

Write-through only solves half the problem