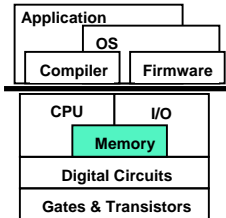
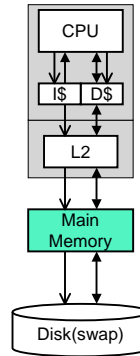


## This Unit: Main Memory



- Memory hierarchy review
- DRAM technology
  - A few more transistors
  - Organization: two level addressing
- Building a memory system
  - Bandwidth matching
  - Error correction
- Organizing a memory system
- Virtual memory
  - Address translation and page tables
  - A virtual memory hierarchy

## Building a Memory System

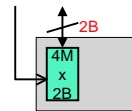


- How do we build an efficient main memory out of standard DRAM chips?
  - How many DRAM chips?
  - What width/speed (data) bus to use?
    - Assume separate address bus

## An Example Memory System

- Parameters
  - 32-bit machine
  - L2 with 32B blocks (must pull 32B out of memory at a time)
  - 4Mx16b DRAMs, 20ns access time, 40ns cycle time
    - Each chip is 4Mx2B = 8 MB
  - 100MHz (10ns period) data bus
  - 100MHz, 32-bit address bus
- How many DRAM chips?
- How wide to make the data bus?

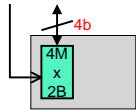
## First Memory System Design



- 1 DRAM + 16b (=2B) bus
  - Access time: 630ns
    - Not including address
  - Cycle time: 640ns
    - DRAM ready to handle another miss
  - Observation: data bus idle 75% of time!
    - We have over-designed bus
    - Can we use a cheaper bus?

T (ns)	DRAM	Data Bus
10	[31:30]	
20	[31:30]	
30	refresh	[31:30]
40	refresh	
50	[29:28]	
60	[29:28]	
70	refresh	[29:28]
80	refresh	
...	...	...
600	refresh	
610	[1:0]	
620	[1:0]	
630	refresh	[1:0]
640	refresh	

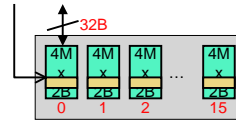
## Second Memory System Design



- 1 DRAM + **4b** bus
  - One DRAM chip, don't need 16b bus
  - DRAM: 2B / 40ns → 4b / 10ns
  - Balanced system → match bandwidths
- Access time: 660ns (30ns longer = +4%)
- Cycle time: 640ns (same as before)
- + Much cheaper!

T (ns)	DRAM	Bus
10	[31:30]	
20	[31:30]	
30	refresh	[31H]
40	refresh	[31L]
50	[29:28]	[30H]
60	[29:28]	[30L]
70	refresh	[29H]
80	refresh	[29L]
...	...	...
600	[1:0]	[2H]
610	[1:0]	[2L]
620	refresh	[1H]
640	refresh	[1L]
650		[0H]
660		[0L]

## Third Memory System Design



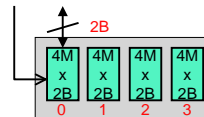
- How fast can we go?
- 16 DRAM chips + 32B bus
  - **Stripe data across chips**
  - Byte M in chip  $(M/2) \% 16$  (e.g., byte 38 is in chip 3)
  - Access time: 30ns
  - Cycle time: 40ns
  - 32B bus is very expensive

T (ns)	DRAM0	DRAM1	DRAM15	Bus
10	[31:30]	[29:28]	[1:0]	
20	[31:30]	[29:28]	[1:0]	
30	refresh	refresh	refresh	[31:0]
40	refresh	refresh	refresh	

## Latency and Bandwidth

- In general, given bus parameters...
  - Find smallest number of chips that minimizes cycle time
  - Approach: match bandwidths between DRAMs and data bus
    - If they don't match, you're paying too much for the one with more bandwidth

## Fourth Memory System Design



- 2B bus
  - Bus b/w: 2B/10ns
  - DRAM b/w: 2B/40ns
  - 4 DRAM chips
  - Access time: 180ns
  - Cycle time: 160ns

T (ns)	DRAM0	DRAM1	DRAM2	DRAM3	Bus
10	[31:30]	[29:28]	[27:26]	[25:24]	
20	[31:30]	[29:28]	[27:26]	[25:24]	
30	refresh	refresh	refresh	refresh	[31:30]
40	refresh	refresh	refresh	refresh	[29:28]
50	[23:22]	[21:20]	[19:18]	[17:16]	[27:26]
60	[23:22]	[21:20]	[19:18]	[17:16]	[25:24]
...	...	...	...	...	...
110	refresh	refresh	refresh	refresh	[15:14]
120	refresh	refresh	refresh	refresh	[13:12]
130	[7:6]	[5:4]	[3:2]	[1:0]	[11:10]
140	[7:6]	[5:4]	[3:2]	[1:0]	[9:8]
150	refresh	refresh	refresh	refresh	[7:6]
160	refresh	refresh	refresh	refresh	[5:4]
170					[3:2]
180					[1:0]

## More Bandwidth From One DRAM

- **EDO**: extended data out
  - Multiple row buffer reads/writes
    - Send only column addresses
- **SDRAM**: synchronous DRAM
  - Read/write row buffer chunks on clock edge
    - No need to send column addresses at all
  - **DDR SDRAM**: double-data rate SDRAM
    - Read/write on both clock edges
  - Popular these days
- **RDRAM**: aka RAMBUS
  - Multiple row buffers, "split" transactions, other complex behaviors
  - Very expensive, high end systems only

This is not information that I expect you to memorize

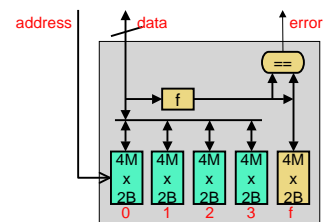
## Memory Access and Clock Frequency

- Nominal **clock frequency** applies to CPU and caches
  - Memory bus has its own clock, typically much slower
  - DRAM has no clock (SDRAM operates on bus clock)
- Another reason why processor clock frequency isn't perfect performance metric
  - Clock frequency increases don't reduce memory or bus latency
  - May make misses come out faster
    - At some point memory bandwidth may become a **bottleneck**
    - Further increases in (core) clock speed won't help at all

## Error Detection and Correction

- One last thing about DRAM technology: **errors**
  - DRAM fails at a higher rate than SRAM or CPU logic
    - Capacitor wear
    - Bit flips from energetic  $\alpha$ -particle strikes
    - Many more bits
  - Modern DRAM systems: built-in error detection/correction
- **Key idea: checksum-style redundancy**
  - Main DRAM chips store data, additional chips store  $f(\text{data})$ 
    - $|f(\text{data})| < |\text{data}|$
  - On read: re-compute  $f(\text{data})$ , compare with stored  $f(\text{data})$ 
    - Different? Error...
  - Option I (**detect**): kill program
  - Option II (**correct**): enough information to fix error? fix and go on

## Error Detection and Correction



- Error detection/correction schemes distinguished by...
  - How many (simultaneous) errors they can detect
  - How many (simultaneous) errors they can correct

## Error Detection Example: Parity

- **Parity**: simplest scheme
  - $f(\text{data}_{N-1,0}) = \text{XOR}(\text{data}_{N-1}, \dots, \text{data}_1, \text{data}_0)$
- + Single-error detect: detects a single bit flip (common case)
  - Will miss two simultaneous bit flips...
  - But what are the odds of that happening?
- Zero-error correct: no way to tell which bit flipped
- Many other schemes exist for detecting/correcting errors
  - Take ECE 254 (Fault Tolerant Computing) for more info

## Memory Organization

- So data is striped across DRAM chips
- But how is it organized?
  - Block size?
  - Associativity?
  - Replacement policy?
  - Write-back vs. write-thru?
  - Write-allocate vs. write-non-allocate?
  - Write buffer?
  - Optimizations: victim buffer, prefetching, anything else?

## Low %<sub>miss</sub> At All Costs

- For a memory component:  $t_{\text{hit}}$  vs. %<sub>miss</sub> tradeoff
- Upper components (I\$, D\$) emphasize low  $t_{\text{hit}}$ 
  - Frequent access → minimal  $t_{\text{hit}}$  important
  - $t_{\text{miss}}$  is not bad → minimal %<sub>miss</sub> less important
  - Low capacity/associativity/block-size, write-back or write-through
- Moving down (L2) emphasis turns to %<sub>miss</sub>
  - Infrequent access → minimal  $t_{\text{hit}}$  less important
  - $t_{\text{miss}}$  is bad → minimal %<sub>miss</sub> important
  - High capacity/associativity/block size, write-back
- For memory, emphasis entirely on %<sub>miss</sub>
  - $t_{\text{miss}}$  is disk access time (measured in ms, not ns)

## Memory Organization Parameters

Parameter	I\$/D\$	L2	Main Memory
$t_{\text{hit}}$	1-2ns	10ns	30ns
$t_{\text{miss}}$	10ns	30ns	10ms (10M ns)
Capacity	8-64KB	128KB-2MB	512MB-4GB
Block size	16-32B	32-256B	8-64KB pages
Associativity	1-4	4-16	Full
Replacement Policy	NMRU	NMRU	working set
Write-through?	Either	No	No
Write-allocate?	Either	Yes	Yes
Write buffer?	Yes	Yes	No
Victim buffer?	Yes	No	No
Prefetching?	Either	Yes	Either

## One Last Gotcha

---

- On a 32-bit architecture, there are  $2^{32}$  byte addresses
  - Requires 4 GB of memory
  - But not everyone buys machines with 4 GB of memory
  - And what about 64-bit architectures?
- Let's take a step back...