Computer Architecture

Babak Kia
Adjunct Professor
Boston University
College of Engineering
Email: bkia -at- bu.edu

Computer Architecture

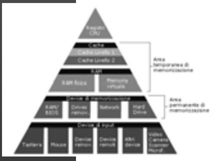
- Computer Architecture is the theory behind the operational design of a computer system
- This is a term which is applied to a vast array of computer disciplines ranging from low level instruction set and logic design, to higher level aspects of a computer's design such as the memory subsystem and bus structure
- In this lecture we will focus on the latter definition of the term

Topics Discussed

- **Memory Hierarchy**
- **Memory Performance**
  - Amdahl's Law and the Locality of Reference
- **Cache Organization and Operation**
- **Random Access Memory (DRAM, SRAM)**
- **Non Volatile Memory (Flash)**
- **Bus Interfaces**
  - How to connect the processor to memory & I/O

Memory Hierarchy

- A simple axiom of hardware design:
  smaller is faster



Memory Hierarchy	Access Times
Register	1 cy.
Cache	1-2 cy.
Main Memory	10-15 cy.
Disk	1000+ cy.

Memory Performance

- Performance is measured either in terms of throughput or response time
- The goal of memory design is to increase memory bandwidth and decrease access time latencies
- We take advantage of three principles of computing in order to achieve this goal:
  - Make the common case faster
  - Principle of Locality
  - Smaller is Faster

Make the Common Case Faster

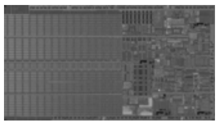
- Always want to improve the frequent event as opposed to the infrequent event
- Amdahl's Law quantifies this process:
  - *The performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used*
- Amdahl's Law essentially guides you to spend your resources on an area where the time is most spent

The Principle of Locality

- Locality of Reference is one of the most important properties of a program:
  - It is a widely held rule of thumb that 90% of execution time is spent in only 10% of the code
- Temporal Locality
  - If a location is referenced, there is a high likelihood that it will be referenced again in the near future (time)
- Spatial Locality
  - If you reference instruction or data at a certain location, there is a high likelihood that nearby addresses will also be referenced

Cache

- Cache is a small, fast memory which holds copies of recently accessed instruction and data
- Key to the performance of modern Microprocessors
- There could be up to two levels of cache, the picture to the right is Intel Pentium M, the large block to the left showing 2 MB of L2 cache



Cache

- Taking advantage of the Principle of Locality, cache loads itself with contents of data which was recently accessed
- But this only addresses temporal locality!
- Therefore to take advantage of spatial locality as well, cache subsystems are designed to read anywhere between 16-128 bytes of data at a time.

Cache Definitions

- Cache is simply defined as a temporary storage area for data which is frequently accessed
- A *cache hit* occurs when the CPU is looking for data which is already contained within the cache
- A *cache miss* is the alternate scenario, when a *cache line* needs to be read from main memory
- The percentages of accesses which result in cache hits are known as the cache *hit rate* or *hit ratio* of the cache
- If the cache is full, it needs to *evict* a cache line before it can bring in a new cache line. This is done through a heuristic process and is known as the cache *replacement policy*

Cache Organization

- Cache is organized not in bytes, but as blocks of cache lines, with each line containing some number of bytes (16-64)
- Unlike normal memory, cache lines do not have fixed addresses, which enables the cache system to populate each cache line with a unique (non-contiguous) address
- There are three methods for filling a cache line
  - Fully Associative – The most flexible
  - Direct Mapped – The most basic
  - Set Associative – A combination of the two

Fully Associative

- In a fully associative cache subsystem, the cache controller can place a block of bytes in any of the available cache lines
- Though this makes the system greatly flexible, the added circuitry to perform this function increases the cost, and worst, decreases the performance of the cache!
- Most of today's cache systems are not fully associative for this reason

Direct Mapped

- In contrast to the fully associative cache is the direct mapped cache system, also called the *one-way set associative*
- In this system, a block of main memory is always loaded into the *same* cache line, evicting the previous cache entry
- This is not an ideal solution either because in spite of its simplicity, it doesn't make an efficient use of the cache
- For this reason, not many systems are built as direct mapped caches either

Set Associative

- Set associative cache is a compromise between fully associative and direct mapped caching techniques
- The idea is to break apart the cache into *n*-sets of cache lines. This way the cache subsystem uses a direct mapped scheme to select a set, but then uses a fully associative scheme to places the line entry in any of the *n* cache lines within the set
- For n = 2, the cache subsystem is called a *two-way set associative cache*

Cache Line Addressing

- But how are cache lines addressed?
- Caches include an address tag on each line which gives it the frame address
  - First, the tag of every cache line is checked *in parallel* to see if it matches the address provided by the CPU
  - Then there must be a way to identify which cache line is invalid, which is done through adding a *valid bit* to the tag line
  - Finally, a random or a Least Recently Used (LRU) algorithm can be used to evict an invalid cache line

Cache example

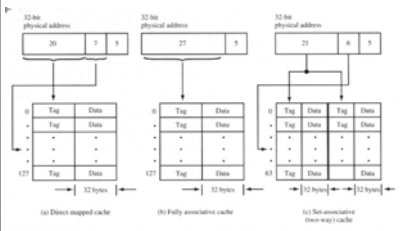


FIGURE 10.12 Cache organization: (a) direct-mapped, (b) fully associative, and (c) set-associative (two-way)

Tag	Index	Offset

Cache Operation

- Most of the time the cache is busy filling cache lines (reading from memory)
- But the processor doesn't write a cache line which can be up to 128 bytes - it only writes between 1 and 8 bytes
- Therefore it must perform a read-modify-write sequence on the cache line
- Also, the cache uses one of two write operations:
  - Write-through, where data is updated both on the cache and in the main memory
  - Write-back, where data is written to the cache, and updated in the main memory only when the cache line is replaced
- Cache coherency – a very important subject! TBD in a separate lecture…

Memory Technologies

- There are basically two different memory technologies used to store data in RAM: Static RAM and Dynamic RAM

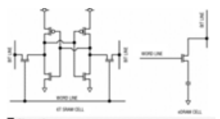


Figure 1

An SRAM cell requires six times more transistors than a DRAM cell. The result: much higher power consumption. A 16 Mbit SRAM consumes 1 A while a 16 Mbit DRAM uses only 10 microA of power.

Static RAM

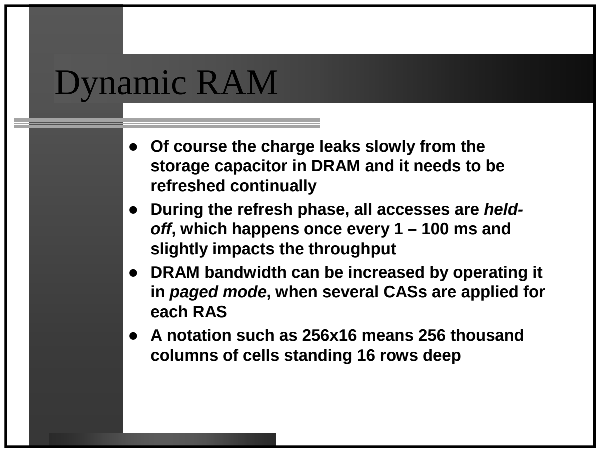
- Uses 4-6 transistors to store a single bit of data
- Provides a fast access time at the expense of lower bit densities
- For this reason registers and cache subsystems are fabricated using SRAM technology
- Static RAM is considerably more expensive than Dynamic RAM
- However, since it doesn't need to be refreshed, its power consumption is much lower than DRAM
- Also, the absence of the refresh circuitry makes it easier to interface to
- The simplicity of the memory circuitry compensates for the more costly technology

Dynamic RAM

- The bulk of a processor's main memory is comprised of dynamic RAM
- Manufacturers have focused on memory sizes rather than speed
- In contrast to SRAM, DRAM uses a single transistor and capacitor to store a bit
- DRAM requires that the address applied to the device be asserted in a row address (RAS) and a column address (CAS)
- The requirement of RAS and CAS of course kills the access time, but since it reduces package pinout, it allows for higher memory densities

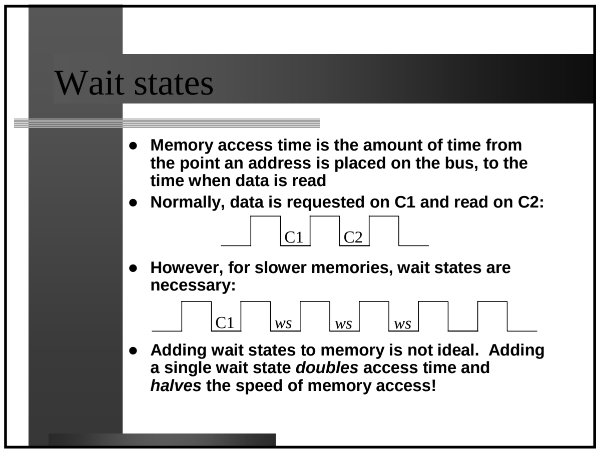
Dynamic RAM

- RAS and CAS use the same pins, with each being asserted during either the RAS or the CAS phase of the address
- There are two metrics used to describe DRAM's performance:
  - *Access time* is defined as the time between assertion of RAS to the availability of data
  - Cycle time is defined as the minimum time before a next access can be granted
- Manufacturers like to quote access times, but cycle times are more relevant because they establish throughput of the system



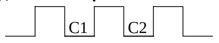
Dynamic RAM

- Of course the charge leaks slowly from the storage capacitor in DRAM and it needs to be refreshed continually
- During the refresh phase, all accesses are *held-off*, which happens once every 1 – 100 ms and slightly impacts the throughput
- DRAM bandwidth can be increased by operating it in *paged mode*, when several CASs are applied for each RAS
- A notation such as 256x16 means 256 thousand columns of cells standing 16 rows deep

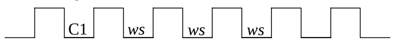


Wait states

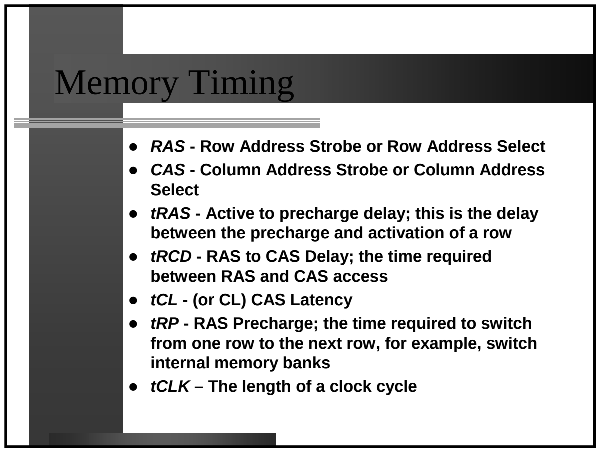
- Memory access time is the amount of time from the point an address is placed on the bus, to the time when data is read
- Normally, data is requested on C1 and read on C2:



- However, for slower memories, wait states are necessary:



- Adding wait states to memory is not ideal. Adding a single wait state *doubles* access time and *halves* the speed of memory access!



Memory Timing

- *RAS* - Row Address Strobe or Row Address Select
- *CAS* - Column Address Strobe or Column Address Select
- *tRAS* - Active to precharge delay; this is the delay between the precharge and activation of a row
- *tRCD* - RAS to CAS Delay; the time required between RAS and CAS access
- *tCL* - (or CL) CAS Latency
- *tRP* - RAS Precharge; the time required to switch from one row to the next row, for example, switch internal memory banks
- *tCLK* – The length of a clock cycle

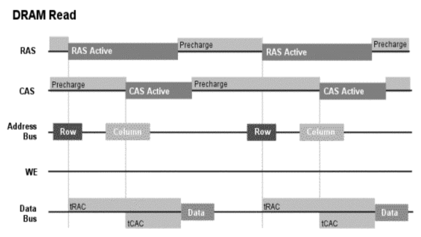
Memory Timing

- *Command Rate* - the delay between Chip Select (CS), or when an IC is selected and the time commands can be issued to the IC
- *Latency* - The time from when a request is made to when it is answered; the total time required before data can be written to or read from the memory.
- Memory timing can be displayed as a sequence of numbers such as 2-3-2-6-T1
  - While all parameters are important to system reliability, some parameters are more significant:
  - This refers to the *CL-tRCD-tRP-tRAS-Command Rate sequence* and is measured in clock cycles

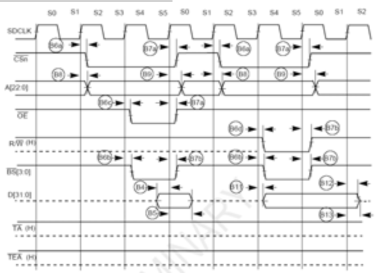
Memory Timing

- CAS Latency
  - One of the most important parameters
  - Delay between the CAS signal and the availability of the data on the DQ pin
  - Specially important since data is often accessed sequentially (within the same row), so CAS timing plays a key role in the performance of a system
- tRCD
  - The delay between the time a row is activated, to when a column is activated
  - For most situations (sequential), it is not a problem, however becomes an issue for non-sequential addressing

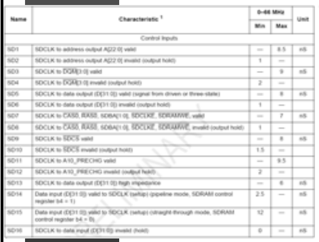
Basic DRAM Timing



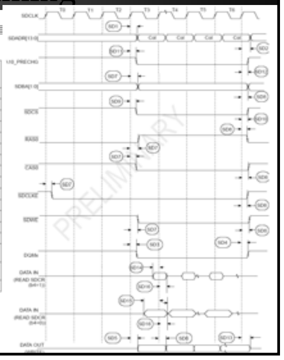
ColdFire SRAM Timing



ColdFire SDRAM Timing

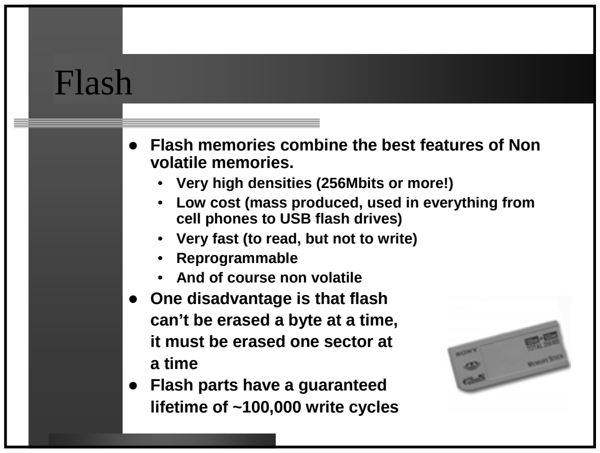


Name	Characteristic [1]	0–66 MHz		Unit
		Min	Max	
Control Inputs				
SD1	SDCLK to address output A[22:0] valid	—	8.5	nS
SD2	SDCLK to address output A[22:0] invalid (output hold)	1	—	
SD3	SDCLK to DQM[3:0] valid	—	9	nS
SD4	SDCLK to DQM[3:0] invalid (output hold)	2	—	
SD5	SDCLK to data output (D[31:0]) valid (signal from driven or three-state)	—	8	nS
SD6	SDCLK to data output (D[31:0]) invalid (output hold)	1	—	
SD7	SDCLK to CAS0, RAS0, SDBA[1:0], SDCLKE, SDRAMWE, valid	—	7	nS
SD8	SDCLK to CAS0, RAS0, SDBA[1:0], SDCLKE, SDRAMWE, invalid (output hold)	1	—	
SD9	SDCLK to SDCS valid	—	8	nS
SD10	SDCLK to SDCS invalid (output hold)	1.5	—	
SD11	SDCLK to A10_PRECHG valid	—	9.5	
SD12	SDCLK to A10_PRECHG invalid (output hold)	2	—	
SD13	SDCLK to data output (D[31:0]) high impedance	—	6	nS
SD14	Data input (D[31:0]) valid to SDCLK (setup) (pipeline mode, SDRAM control register b4 = 1)	2.5	—	nS
SD15	Data input (D[31:0]) valid to SDCLK (setup) (straight-through mode, SDRAM control register b4 = 0)	12	—	nS
SD16	SDCLK to data input (D[31:0]) invalid (hold)	0	—	nS



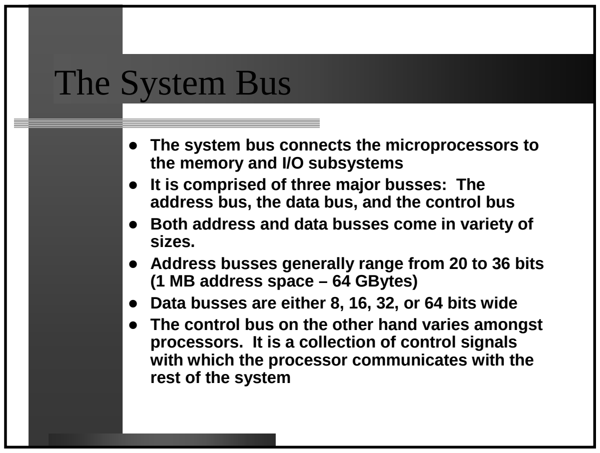
NvRAM

- Both SRAM and DRAM are known as volatile memories – turn the power off and they lose their contents
- Computers also rely on a wide array of Non-volatile Random Access Memories or NvRAMs, which retain their data even if the power is turned off. They come in many forms:
  - ROM - Read Only Memory
  - EPROM - Erasable Programmable Read Only Memory
  - EEPROM - Electrically Erasable Programmable Read Only Memory
  - Flash – A form of EEPROM which allows multiple locations to be erased or written in one programming operation. Normal EEPROM only allows for one location to be erased or programmed



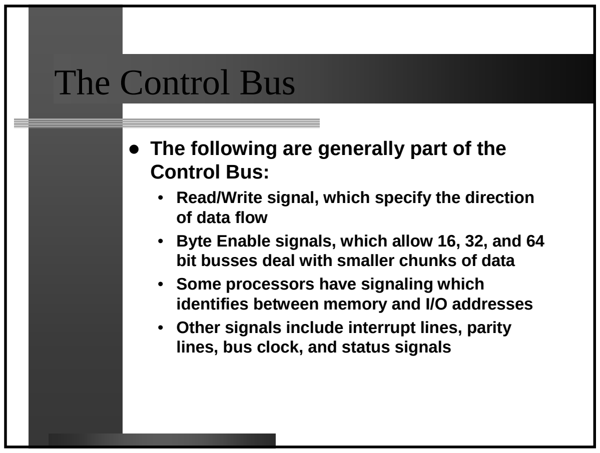
Flash

- Flash memories combine the best features of Non volatile memories.
  - Very high densities (256Mbits or more!)
  - Low cost (mass produced, used in everything from cell phones to USB flash drives)
  - Very fast (to read, but not to write)
  - Reprogrammable
  - And of course non volatile
- One disadvantage is that flash can't be erased a byte at a time, it must be erased one sector at a time
- Flash parts have a guaranteed lifetime of ~100,000 write cycles



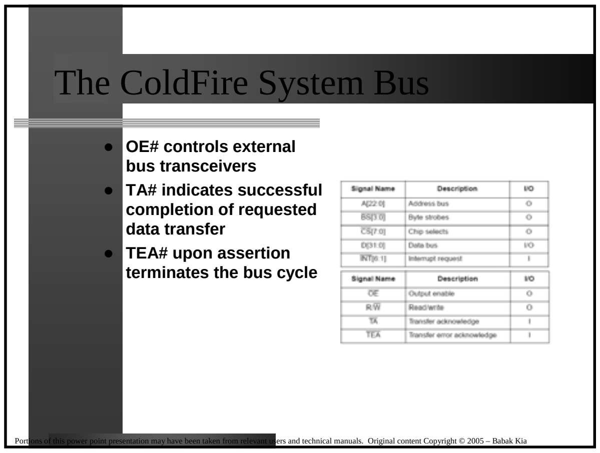
The System Bus

- The system bus connects the microprocessors to the memory and I/O subsystems
- It is comprised of three major busses: The address bus, the data bus, and the control bus
- Both address and data busses come in variety of sizes.
- Address busses generally range from 20 to 36 bits (1 MB address space – 64 GBytes)
- Data busses are either 8, 16, 32, or 64 bits wide
- The control bus on the other hand varies amongst processors. It is a collection of control signals with which the processor communicates with the rest of the system



The Control Bus

- The following are generally part of the Control Bus:
  - Read/Write signal, which specify the direction of data flow
  - Byte Enable signals, which allow 16, 32, and 64 bit busses deal with smaller chunks of data
  - Some processors have signaling which identifies between memory and I/O addresses
  - Other signals include interrupt lines, parity lines, bus clock, and status signals



The ColdFire System Bus

- OE# controls external bus transceivers
- TA# indicates successful completion of requested data transfer
- TEA# upon assertion terminates the bus cycle

Signal Name	Description	I/O
A[22:0]	Address bus	O
$\overline{\text{BS}}$[3:0]	Byte strobes	O
$\overline{\text{CS}}$[7:0]	Chip selects	O
D[31:0]	Data bus	I/O
$\overline{\text{INT}}$[6:1]	Interrupt request	I

Signal Name	Description	I/O
$\overline{\text{OE}}$	Output enable	O
R/$\overline{\text{W}}$	Read/write	O
$\overline{\text{TA}}$	Transfer acknowledge	I
$\overline{\text{TEA}}$	Transfer error acknowledge	I

Portions of this power point presentation may have been taken from relevant users and technical manuals. Original content Copyright © 2005 – Babak Kia