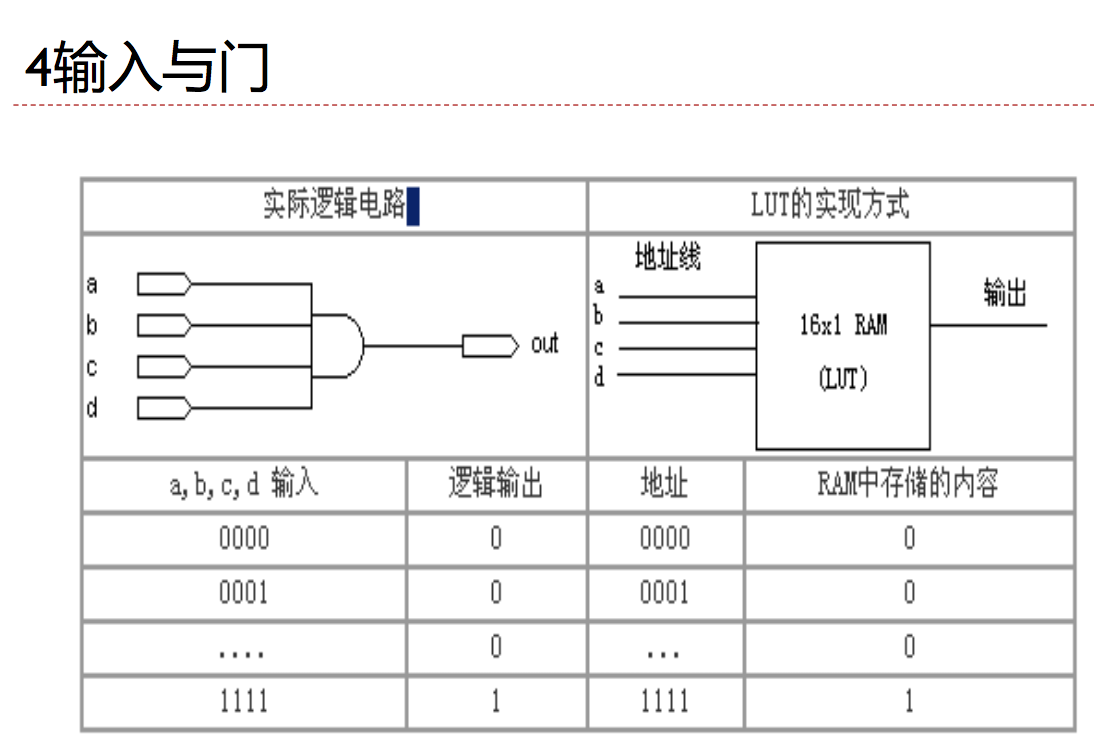
PPT3

Intro to FPGAs

* FPGA (Field Programmable Gate Array) is based on the SRAM architecture and has logic elements such as lookup tables (查找表), flip-flops (触发器) as basic units. It also has memory, DSP, and IO standards
  + 查找表
    - Pretty much a RAM
    - How it works:
      * A software will solve a logical equation and store the answer in a RAM



* Xilinx FPGA main components
  + 可编程输入输出单元
  + 可编程逻辑快
  + 始终管理模块
  + 片内RAM(BRAM)
  + 布线资料
  + 内嵌功能单元
  + 内嵌功能硬核
* FPGA 的片内资源
  + LUT结构
  + FF结构
  + 逻辑片 SLICE
  + 嵌入存储器
  + 时钟管理单元
  + 数字信号处理单元
  + IO单元

Hardware Programming Theory

* Methods and Principles of Hardware Design
  + 面积和速度的平衡与互换(Area and Speed)
    - Area is the number of system resources that are occupied after an FPGA design is synthesized. (typically measured by the number of occupied logical units and the # of IO interfaces)
    - Speed pretty much means the frequency at which the system works. (higher frequency = higher speed)
    - Speed and Area typically have a indirect correlation: you might have to sacrifice one for the other
      * Ex: make multiplexed modules run in parallel to optimize speed (area increases)
  + 功耗考虑(Power Consumption Consideration)
    - Dynamic Power Consumption (动态功耗) is proportional to the frequency of logic flip changes
    - When designing, try to avoid simultaneous flips as much as possible
  + 硬件原理(Hardware Principles)
    - When designing, think first of the hardware, not software implementation
  + 系统原理(System Principles)
    - Designs should involve the big picture
      * Ex: module reuse is much more effective with large modules rather than small modules
    - Know the hardware resources of the FPGA before designing (mentioned above)
  + 同步设计原理(Synchronous design principles)
    - 同步 is less resourceful and is less costly
    - Avoids 毛刺 or glitches

Hardware Programming Process

* 设计，输入和综合
  + 原理图
  + 硬件描述·语言
* 设计实现
  + 生成比特流文件(generate bitstream file)
* 实际验证
  + 门级仿真
  + 下载

COME BACK IF THERE’S TIME

Summary:

This PPT explained the resources that an FPGA used as well as the design principles and workflow of designing and creating your own FPGA

PPT4

Computer是什么

* A computer has the following characteristics:
  + High speed digital device
  + Used to calculate data
  + Can accept input signals
  + Can process information given the user’s requests
  + Can output results
* 计算机程序
  + Language that computer and coders communicate
  + Divided into three categories
    - High level language (高级语言)
    - Assembly Language (汇编语言)
    - Machine Language (机器语言)
* Mechanics of computer operation
  + Meh I get it

数据表示的需求

* 数据的编码与表示(Data encoding and representation)
  + Things that need to be represented in the computer:
    - Code, integers, floating points, character strings (and characters), logical values
    - Represented by encoding
  + 表示方式
    - ??? ppt 4 pg 9
  + 编码原则
    - Should have small amount of characters
    - Unified rules
    - Can represent complex information
    - Computing performance/storage space
* 编码表示
  + Basic element:
    - 0, 1
  + Letters:
    - 26 letters -> 5 bits
    - Capital/lowercase + other symbols -> 7 bits
    - All other languages in the world -> 16 bits (unicode)
  + Unsigned integers (0, 1, … , 2^n-1)
  + Signed integers
  + Floating points
  + Logical Values
    - 0 is false, 1 is true
  + Color (RGB)
  + 位置／地址／指令

逻辑型数据表示

* Logical data
  + True/False
* Data representation
  + 1/0
* Data operations
  + And, or, not, xor

字符的表示

* Character encoding standareds
  + ASCII(American Standard Code for Information Interchange)
    - Uses a 7 bit binary encoding, uses one word
    - Has 128 western characters
  + UNICODE
    - 16 bits per word, so it can represent 65536 words
    - saves 6400 points for localization
    - still can’t cover all characters
  + UTF-8
    - Character length is variable
    - First bit represents the length of the character
    - Other than the first byte, they all start with “10”
    - Dominant encoding on the internet
* Other encoding
  + 点阵字体 (Dot matrix font)
    - Meh
  + 矢量字体(True Type)
    - Meh

整数与浮 (fú)点数

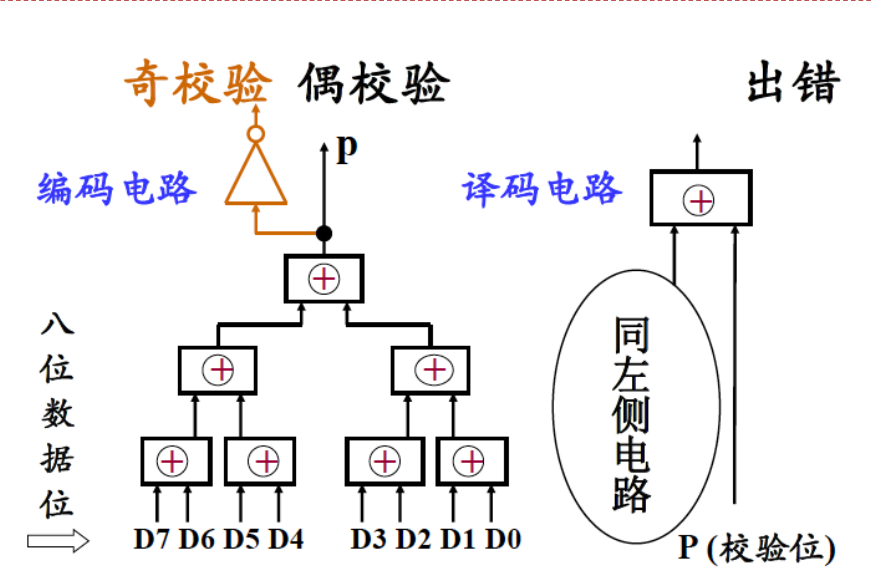
* Numerical data representation
  + 定点数
    - 小数点位置固定
    - 整数
    - 定点小数
  + 浮点数
    - 小数点位置浮动
* Integers
  + 原码，反码，补码(see <https://www.youtube.com/watch?v=UJa8mTw0R3c>)

^(or: <https://www.youtube.com/watch?v=4qH4unVtJkE>)

* + 符号扩展
  + 大断，小断
  + 加法，减法，乘法，除法
* 浮点数
  + STUDY FLOATING OPERATIONS

检错纠错码

* How to detect errors
  + Have the encoding have certain characteristics. Check that the characteristics exist, if so encoding is correct.
  + In the event of an error, you can detect the location of the error and correct it
    - 编码
    - 检查
    - 出错后纠正
* Commonly used error detection encodings
  + 奇偶校验码(parity)：并行数据传输
    - Last bit is the “check bit”. It’s value (0/1) is determined by which parity it is: even/odd
    - In even parity, there will be an even number of 1’s
    - In odd parity, there will be an odd number of 1’s



* + 海明校验码(hamming)：并行数据传输
  + 循环冗余校验码(cyclic redundancy)：串行数据传输 (see <https://www.youtube.com/watch?v=6gbkoFciryA>)
    - Basically, you have a “checksum” like string attached to the end of the message. Both computers agree upon a patter P that is used to verify the message.
    - The Message + n bits (n = “checksum” length) of 0’s are divided by the message. Then, the remainder of this string divded by P is appended to the message and sent
    - To verify, Divide the received packet by P and the remainder should be 0
* FINISH LATER IF TIME ALLOWS

Summary:

This chapter basically talked about the structure of data (in terms of logic, integers, floating point numbers, and cpu’s understanding of said data types). It also explained how verification is done for messages.

重点:

* Character Encoding:
  + ASCII
  + Unicode
  + UTF
* Data Types
  + Floating Point Numbers (浮点数)
  + Integers
    - 原码，反码，补码
* 检错纠错码
  + Concepts
    - 码距
      * K+1位码 only has 2k states, code distance = 2.
    - 编码方案
  + 奇偶校验码 (码距 = 2)
  + 海明码 (码距 = 4)

PPT4

运算器功能

* Takes place in the datapath portion of a CPU
* Executes arithmetic and logic calculations
  + + - \* / ∧∨¬
* States of the resulting calculations
  + C Z V S
* Obtain results
  + 寄存器组，数据总线
* Output, store results
  + 寄存器组(registers), 数据总线 (data bus)
* Save middle values of an operation
  + Q寄存器 (q register)，移位寄存器 (shift register)

用于实现运算功能的基础逻辑电路

* 逻辑门电路
* 加法器
* 触发器
* 多路选择器，移位器

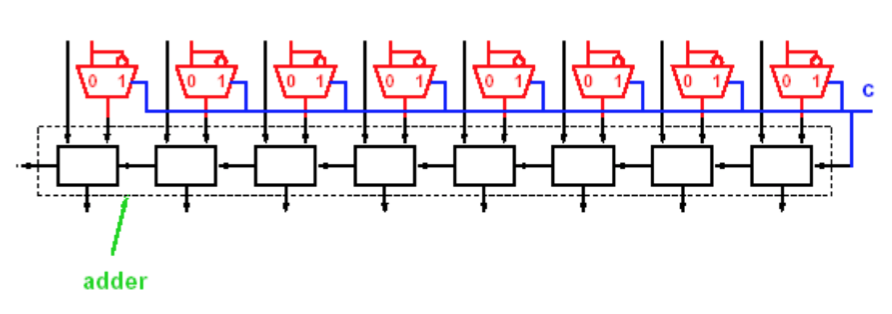
ALU设计

* 一位ALU design process
  + Confirm ALU features
  + Confirm ALU input parameters
  + Get the truth table according to the 功能要求, get the 逻辑表达式
* 4位ALU实现方法
  + 思路1: repeat the process above for four digits instead or
  + 思路2: serialize 1位ALU
  + 超前进位生成
  + 其它的结果标志
    - Z = (F1 = 0) \* (F2 = 0) \* (F3 = 0) \* (F4 = 0)
    - S = 最高位
    - OV = ¬F1\*¬F2\*S+F1\*F2\*¬S

PPT 4 pg 18 I THINK THIS WILL BE ON THE TEST

算术运算的实现

* 补码的减法
  + According to operation rules
  + [a - b]补 = [a]补 + [-b]补
  + [-b]补 should be [b]补’s one’s complement + 1
  + Subtraction is realized by adding negative numbers
* Combining addition and subtraction



* + We can add a MUX where when C = 0 we add and when C = 1 we subtract
* 原码 multiplication (CHECK IF THERE’S TIME) (ppt4 24-28)
* 补码 multiplication
  + Method 1:
    - I’m guessing the PPT is saying whatever we did above with 原码
  + Method 1:
    - 布斯算法 (see <https://www.youtube.com/watch?v=1ubyXuXxIWU>)
* 原码 division
  + See
* 补码division
  + Difference between 原码 and 补码 division
* FINISH LATER

Summary:

This ppt explained the digital circuits involved in ALU design as well as the calculations each type of operation (logical, add, subtract, multiply, and divide) is computed with 原码 and 补码

PPT 5 (6.1)

运算器功能与概述

定点运算器实例AM2901

用Verilog语言描述AM2901

PPT6 控制器概述 指令和指令系统

复杂指令集(CISC)/精简指令集 (RISC)

* RISC is more performance oriented: instructions are very simple but you might need a lot of instructions to complete one task
* One instruction is completed in a single cycle
* CISC has more complex instructions: one instruction can complete a more complex task
* CISC instructions might take multiple cycles

指令与指令系统

* 指令与指令系统的概念
  + Instructions must:
    - Be a string of multiple bits
    - 用于设计程序???
    - Can be directly executed by CPU hardware
  + Five components of a computer system:
    - 运算器部件
    - 控制器部件
    - 存储器部件
    - 输入设备
    - 输出设备
* 指令系统设计要求
  + 完备性
    - Instruction set is complete, code is easily implemented
  + 规整性 （Regularity）
    - Instruction format is simple and uniform
  + 高效性
    - Takes little memory, high performance
  + 兼容性
    - Compatible with software
* 指令功能和分类
  + 算术逻辑运算指令
    - Add, sub, multiply, divide, 变符号 arithmetic operations
    - And, or, not, xor logical operations
  + 移位操作指令
    - 算术移位(usually just right shift)，逻辑移位，循环移位
  + 数据传送指令
    - Transmit data from registers to themselves
    - Transmit data from registers to main memory
    - Transfer between different storage units in main memory
  + 输入输出指令
    - Transfer between registers to input/output devices
  + 转移指令 (Branch instructions)
    - Instructions that change the order in which instructions are executed, divided into unconditional and conditional branch instructions
  + 子程序调用与返回指令
    - Subroutine call and returns must work together. After the subroutine is called, the breakpoint in the main code is returned
  + 堆栈操作指令
    - Stack operations such as push or pop
  + Others
    - Set condition code instruction, open interrupt instruction, close interrupt instruction
    - Stop instruction, no-operation instruction, privileged instruction
* 指令表示
  + Instructions have two parts: an opcode and operand/address
    - Arithmetic logic instructions, data source or result output
    - Data transit instructions, data source location and new location
    - IO instructions, which device and where the data should come from/go to
    - Branch instructions, branch conditional and branch location
* 指令格式
  + Instruction format
    - 指令字(instruction word): complete instruction in binary representation
    - 指令长(instruction length): the number of bits in an instruction
      * 机器字长 (machine word length): the # of bits of binary the computer can directly process
      * 指令字长(字节倍数 aka byte multiple) = 0.5, 1, 2, … 个机器字长
      * 变长指令字structure／定长指令字structure
  + 变长指令字／定长指令字
    - 操作码opcode
      * Fixed length opcodes
        + Having a fixed length for the opcode simplifies the design of the CPU and improves the instruction recognition speed
        + Ex: IBM360, THINPAD
      * Variable length opcodes
        + A fix length is used for the significant part of the instruction, but the remaining areas in the instruction can be varying length
        + This makes instruction decoding more complicated in the hardware design, but allows for more instructions without changing the instruction word length
        + Ex: PDP-11计算机、TEC-2000的8位机
      * NOVA (DJS-130) has a scenario where the opcode and the operand address might overlap
  + 操作码扩展
* 寻址方式

PPT 15 动态存储器DRAM

存储器系统功能

* Requirements for memory
  + Can use two stable states to represent binary “0” and “1”
  + Easy to identify
  + The two states can easily be switched
  + Some common storage methods:
    - 磁颗粒 (magnetic particles)，半导体 (semiconductors), 光
* Early storage
  + 水银延迟线存储器
  + EDSAC, 1949
  + Maurice Wilkes
  + 1967 Turing
  + 存储原理
    - 水波
* 磁芯存储器 (Magnetic core storage)
  + Meh
* 半导体存储器
  + ROM, RAM
  + SRAM, DRAN
* Categorized by access method
  + 随机访问存储器 (RAM)–Random
    - Access time is independent of storage
    - Semiconductor storage
  + 顺序访问存储器 (SAM)–Sequential
    - Sequential access by storage location
    - 磁带 (tape) storage
  + 直接访问存储器 (DAM)–Direct
    - Random and sequenced
    - 磁盘 (disc) storage
  + 关联访问存储器 (CAM)–Content
    - Access by content
    - Cache and TLB
* Impact memory has on computer performance:
  + Suppose a computer process is working on:
    - Main frequency = 1 GHz (machine period is 1 ns)
    - CPI = 1.1
    - 50% logic/arithmetic instructions, 30% storage instructions, 20% branch instructions
  + Then assume 10% of the 存取instructions will have missing data, and requires 50 period delays
    - CPI = ideal CPI + every instruction’s average delay = 1.1 + (0.3 \* 0.1 \* 50) = 1.1 + 1.5 cycles = 2.6 CPI
    - In other words, 58% of the time will be spent on waiting for the storage unit to get data

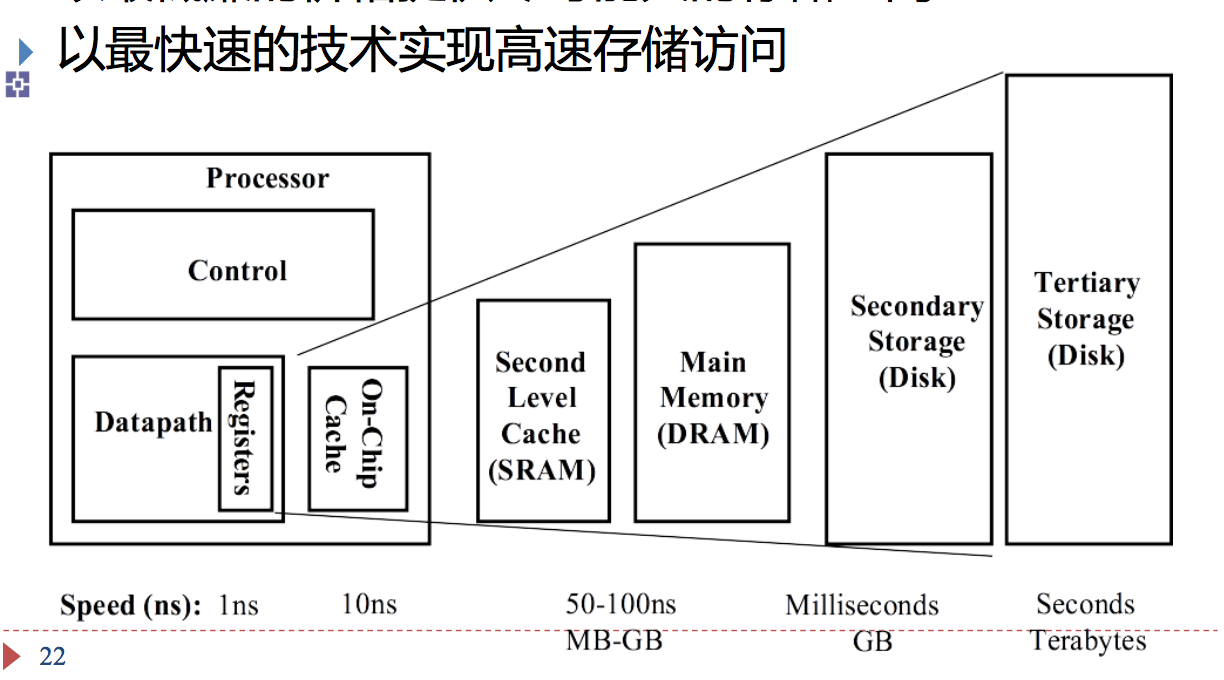
Design principles of存储器系统

* Goals
  + Store and retrieve as fast as possible (高速度)
    - Should be able to satisfy the CPU’s requirements for data access
  + Storage should be as large as possible (大容量)
    - Satisfy the storage requirements of a program
  + Should have a low unit of cost (price/unit)(低成本)
    - Within a user’s budget
  + High reliability(高可靠性)
* Current situation
  + Mass storage is slow
  + Fast storage is small
* Goal Realization
  + Have a hierarchical memory system 层次存储器系统

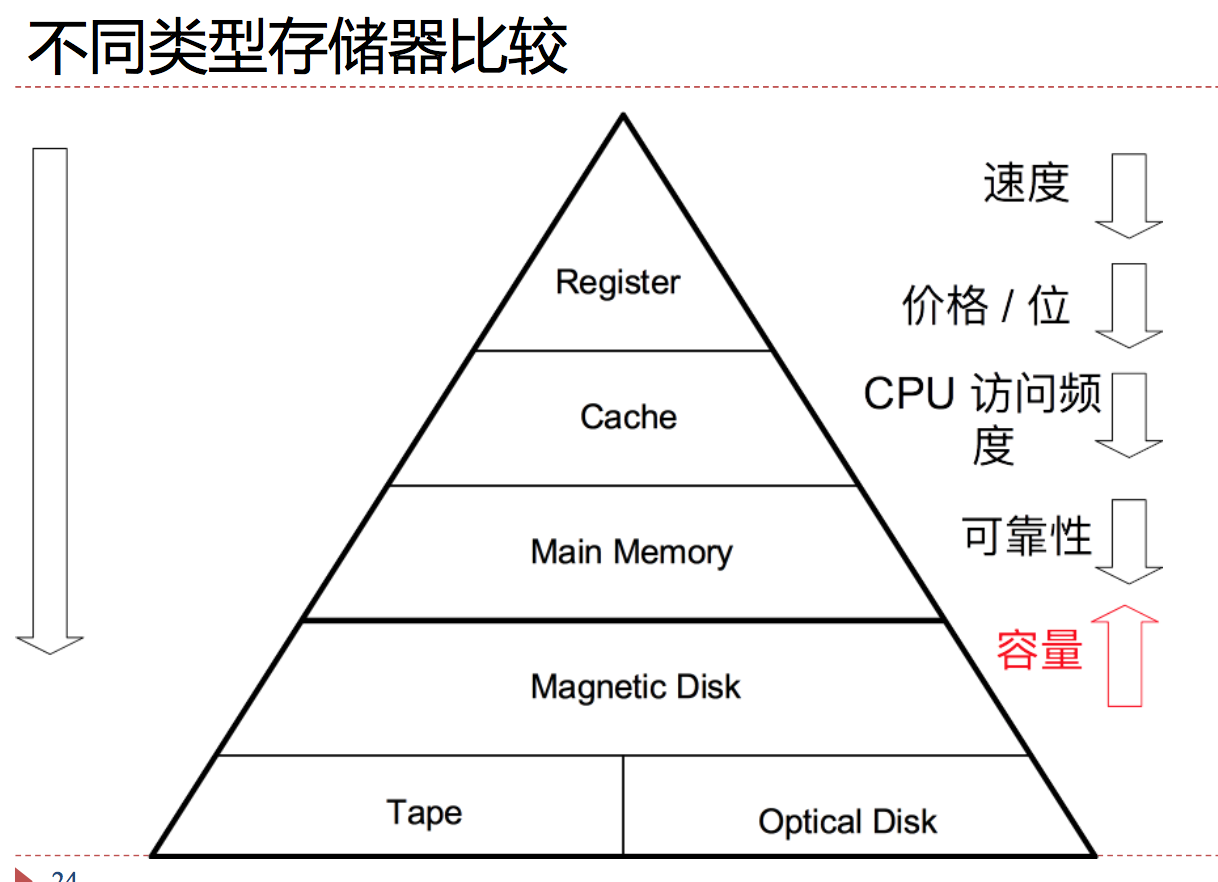
Some Issues

层次寄存器系统 hierarchical memory system

* Characteristics
  + High Speed
    - High static memory speed
    - Smaller cache size
  + Large Storage Capacity
    - Dynamic memory is moderately fast and moderately fast
    - Can be used as main memory
  + Low Cost
    - Disk storage is cheap
    - As auxiliary storage, it can store stuff that isn’t frequently used by the CPU
    - Can load virtual memory
* Principle of Locality 程序运行的局部性原理
  + In a short period of time, a recently used memory is likely to be used again
  + In storage space, the program may refer to data near the address that it’s currently looking into
  + In terms of sequence, instructions are more likely to execute the next-in-line instruction instead of branching (about 5x more likely)
  + Therefore, we should allocate data in appropriate storage media
* Hierarchical Memory Principles
  + 1, 一致性Consistency: the same information in different hierarchies should have the same value
  + 2, 包含性Inclusion: Information at the top hierarchies must be included in the lower hierarchies



* Modern Storage System
  + 主存储器Main Memory
    - 寄存器Register
    - 高速缓存Cache
    - 主寄存器 Main Memory
  + 辅助存储器
    - 磁盘 Disk
    - 磁带 Tape
    - 光盘 Compact Disc



* Parallel Technology 并行技术 page 25
* BUSes
  + Address Bus 地址总线
    - Used to select a memory unit (word/byte) in main memory
    - # of bits determines the # of memory units that can be obtained (最大可寻址空间)
    - Ex: when addressing by bytes, a 20-bit address can access 1MB of storage, while a 32 bit address can access 4GB of storage
  + Data Bus 数据总线
    - The data bus is used to transfer data between the functional components of the computer. The product of the number of bits (the width of the bus) and the clock frequency of the bus is directly proportional to the highest data throughput (input / output) capability supported by the bus.
  + Control Bus 控制总线
    - Meh
* Main Memory Read/Write Process
  + Read:
    - Send address
    - Give chip enable and read request
    - Save read content
  + Write:
    - Give Address
    - Give chip enable and data
    - Give Write request

DRAM architecture and principles

* Working Principle
  + [电子学 so NO ]
  + See (<https://www.youtube.com/watch?v=Lozf9sceW_o>)
* Characteristics of DRAM
  + Type of 半导体 memory
  + 破坏性读出 (Destructive Readout?)
    - Forced to 清零 when reading, read out value is rewritten immediately
    - ^phenomenon called预充电延迟 (recharge delay). Next value cannot read until recharge is complete
  + Needs to be refreshed regularly 定期刷新
    - The capacitors will lose their charge as time passes, so the DRAM needs to be refreshed regularly
    - Two types of refresh methods:
      * 集中刷新 (Centralized refresh)
        + Stop all memory read and write functions, then refresh all rows one by one
      * 分散刷新 (Scatter refresh)
        + Refresh a row when a read/write function is performed, or refresh all rows after a period (i.e. 2 ms)
  + 快速分页组织 Quick Paging organization
    - Idk
* Process
  + [FINISH LATER]

PPT 16 Cache

DRAM 原理

* See previous ppt

静态存储器SRAM原理

* Read/Write
  + Write
    - Set the Bit line (bit = 1, ~bit = 0)
    - Select word line
  + Read
    - Charge bit and ~bit with 高电平Vdd
    - Select the word line
    - Depending on the state of the 触发器 one of the bit lines will be low
    - 放大器will detect bit and ~bit’s changes, read stored value
* Characteristics
  + High speed
  + Storage density is low, 单位面积存储容量小 (small storage capacity per unit area)
  + 数据入／出公用管脚 data entering/exiting uses same pins
  + 能耗高 high energy
  + 成本高 high cost



高速缓冲存储(Cache)概念

* Because of the principle of Locality, Cache should
  + Load the most recently used data into the cache
  + Load the information near the recently used data into the cache
* Definition
  + The memory between the CPU and main memory that stores the most frequently accessed information by the CPU
* Features:
  + High speed: matches the operating speed of CPU
  + 透明 transparent: managed completely by hardware; programmers do not have to interact with it.
* Issues to be solved:
  + Mapping relationship between address and cache line地址和Cache行之间的一致性:
    - How do we get the data in the cache according to the main memory address?
  + Data consistency:
    - Does the content in the cache match the content in the main memory?
  + Granularity of data exchange数据交换的粒度:
    - Meh
  + Cache 内容装入和替换策略 Content loading and replacement strategy
    - How can we improve the cache hit rate?
* Cache Parameters
  + 块(line/block): the smallest unit of data transferred to the cache
  + 命中(hit): found the thing you’re looking for
    - 命中率(Hit Rate): 命中次数／访问次数
    - 命中时间: the time it takes to send the data from the higher tier of memory to the CPU
  + 缺失(miss): blocks that you have to look for in lower tiers of the hierarchy
    - 缺失率 (miss penalty): 1 – 命中率
    - 缺失损失 (miss penalty): the time it takes for send data back to CPU
  + 命中时间<缺失损失
  + Average 访问时间= HR \* 命中时间 + (1 - HR) \* 缺失损失
* Typical values for parameters
  + 块大小:4~128 Bytes
  + 命中时间:1~4周期
  + 失效损失:
    - 访问时间:6~10个周期
    - 传输时间:2~22个周期
  + 命中率:80%~99%
  + Cache容量:1KB~256KB

Cache Mapping地址映射

* 直接映射 Direct Mapping
  + (see: <https://www.youtube.com/watch?v=pSarQQTJbDA>)
  + Process:
  + Characteristics:
    - Most simple
    - The tag is short, so the comparative circuit would be simple,
    - If the main memory has 2m blocks, cache has 2c blocks, then the tag only needs m – c bits.
    - 如果主存空间有2m块，Cache中字块有2c块，则标志位只要有m-c位。
    - Comparison only happens once
    - **利用率低，命中率低** Most amount of misses**，效率较低** Slowest
* 全相联
  + Process:
  + Characteristics:
    - The main memory can correspond to any block in the cache, making Full associative cache mapping very flexible.
    - The tag (标志位) is long, so it takes a lot of resources to compare. If the main memory has 2m blocks, then the tag must have m bits
    - If the Cache has n blocks, then there needs to be n comparative circuits (比较电路)
    - **使用成本太高**
* 多路组相连
  + Characteristics:
    - Pretty much a compromise between the above two methods. Blocks are fully connected, and the groups are directly mapped
    - 成本不太高
    - **常用方式**
  + FILL IN LATER

PPT 18 Cache

Overview

* How do we use map main memory to cache?
  + Address mapping methods (see below/above)
* How do we 保证一致性? (the data in main memory and the data in the cache are the same)
  + 有效位, 写策略
* Cache Parameter’s effect on performance
  + Cache 块大小
  + 替换策略 replacement strategy
  + 接入方式 access method

Cache address mapping

* 全相联映射
* 直接映射
* 多路组相联

Comparison of three mapping methods

} Direct mapping

} A block in main memory can only be mapped to the only position in the cache} When positioning, no judgment is needed, just replace

} Fully connected mapping

} A block in main memory can be mapped to any position in the cache

} N-way group connected map

} A block in main memory can be mapped to N locations in the cache

} Failure handling of fully connected maps and N-way group connected maps

} Remove new block from main memory

} In order to free up cache space, a cache block needs to be replaced.} Not unique, you need to determine which block should be replaced.

一致性保证

* 写直达(Write through)
* 拖后写 (Write back)

Cache 写策略

Ways to improve Cache性能

* 组织结构
* Cache参数（大小、块大小、替换策略）

PPT 23 IO

输入输出系统的作用，功能，及与其他系统的关系

输入／输出系统组成

Issues to be solved

* 控制方式
  + How does the CPU control 输入／输出?
* Transmission
  + Transmission channel, mode, speed, etc. (bus, interface)
* 数据识别和转换
  + Digital-to-analog conversion, speech recognition, etc., converted to characters, data and other computer-readable formats (devices)

Input/Output methods

* 程序设计直接控制
  + Computer directly uses input/output instructions to control external devices
  + Characteristics:
    - Low cost
    - Low efficiency
    - Seriously depletes CPU resources
    - Not always applicable
* 程序中断
  + External devices requests, CPU 响应 response, CPU works in parallel with external device
  + Increases CPU efficiency
  + Can simultaneously manage multiple devices
  + Some concepts:
    - 中断源
    - 中断优先级
    - 禁止中断与中断屏蔽 (Interrupt disable and Interrupt Masks)
* 直击存储访问DMA （Direct Memory Access）
  + Specialized input/output devices
* 通道
* 外围处理机