

1. 解: 串行传输: 优点: 使用数据线少, 在远距离通信中成本较低, 可以用很高时钟频率

缺点: 传输速度比较低

并行传输: 优点: 传输速度较快, 适合并发操作 ^{近距离传输}

缺点: 内存有多少位, 就要用多少数据线, 成本很高

串行接口的频率可以比并行接口高很多, 因为随着时钟频率越来越高,

并行导线间的干扰越严重, 串行总线由于只需一条传输线路所以可以

采用高频传输

2. 解:

(1) 波特率为 $960 \times 10 = 9600$

(2) 有效数据传输速率 $9600 \times \frac{1}{10} = 960 \text{ bits/s}$

3. 解:

(1) I2C 的数据包包括起始和终止条件, 传输起始的 7-10 位的地址帧, 1 位的读/写位以及各字节的数据帧和 1 位的 ACK/NACK 位

(2) 因为 I2C 仅使用一条数据线 SDA, 所以其传输是半双工的

(3) SCL 为高电平时, SDA 由高向低跳变

4. 解:

(1) RAID0 时以磁盘

$$MTTF = \frac{MTTF}{4} = \frac{14}{4}$$

(2) 采用 RAID-6, 因为如 4 个磁盘需要两个校验位, 两个磁盘进行冗余, 剩余 2 个磁盘 50% 的空间可以用来存两个校验位, 这样可以提高 MTTF

Ch5

5. 解: 寻道时间: 磁头臂移动到正确位置并消除抖动所需要的时间; 与磁头臂需要移动的距离和移动速度有关

旋转时间: 磁头移动到目标磁道后, 目标扇区随着盘片转动而经过磁头下(上)方所需的时间。与盘片旋转多少有关, 最坏情况要旋转一周才能定位

数据传输时间: 磁头完成读出或写入所需时间; 和盘片旋转速度有关

6. 解 (1) $6 \times 240 \times 12 \text{ KB} = 17280 \text{ KB} = 16.875 \text{ MB}$

(2) $12 \text{ KB} / (5400 \text{ r/min} \div 60) = 960 \text{ KB/s}$

(3) $60 \div 5400 \div 2 = 0.00625 \text{ s}$

7.解:控制电路可以先缓存一段时间内的磁盘 I/O 请求然后通过一些算法决定最佳执行次序。↵

最短寻道时间优先算法 (SSTF): 总是从等待访问者中挑选寻找时间最短的那个请求先执行的, 而不管访问者到来的先后次序。与先来先服务、算法比较, 大幅度地减少了寻找时间, 因而缩短了为各访问者请求服务的平均时间, 也就提高了系统效率。但 SSTF 查找模式有高度局部化的倾向, 会推迟一些请求的服务, 甚至引起无限拖延 (又称饥饿)。↵

扫描算法 (SCAN): 首先自里向外访问, 下一个对象是其欲访问的磁道既在当前磁道之外, 又是距离最近的; 直至无更外的磁道需要访问时, 才将磁臂换向为自外向里移动; 下一个访问的磁道在当前位置内为距离最近者; 直至再无更里面的磁道要访问。SCAN 算法不仅考虑到欲访问的磁道与当前磁道间的距离, 更优先考虑了磁头当前的移动方向。避免了出现“饥饿”。缺点是当磁道刚从里向外移动而越过了某一磁道时, 刚好一进程请求访问此磁道, 这时此进程会等待, 待磁头继续从里向外, 然后从外向里扫描完处于外面的所有要访问的磁道后, 才处理此进程, 致使该进程的请求被大大推迟。↵

循环扫描算法 (CSCAN): 首先自里向外访问, 当磁头移到最外的磁道并访问后, 磁头返回到最里的欲访问磁道, 即将最小磁道号紧接着最大磁道号构成循环, 继续循环扫描直至无更外的磁道需要访问时, 才将磁臂换向为自外向里移动; 下一个访问的磁道在当前位置内为距离最近者; 直至再无更里面的磁道要访问。↵

↵

8.解: RAID-4 的写入过程可以优化, 因为对于写入某物理磁盘上的一个数据块, 其他物理磁盘将产生读取任务, 以计算出新的奇偶校验数据, 并将校验位写入奇偶校验磁盘。RAID-4 引入了一种简单的优化方法来避免这一点: 将数据块写入某物理磁盘前, 首先出该位置原先的数据块, 对比将要写入的新数据块, 计算出发生翻转的位, 并由此计算出奇偶校验磁盘中的该位置数据块对应位是否需要翻转。优化后, 该写入方式只牵涉到两个物理磁盘: 待写入数据的物理磁盘和奇偶校验磁盘。因此该方法在物理磁盘数量很多时能够显著提升性能。↵

↵

↵

9. 解 I/O 请求减少, 平均到达率 λ 减少, $W = \frac{1}{\mu - \lambda}$, $\frac{dW}{d\lambda} = \frac{1}{(\mu - \lambda)^2}$

随着 λ 减小, $\frac{dW}{d\lambda}$ 减小
即性能提升幅度减小了

10. 解:

当 CPU 和 DMA 都需要进行访存时, 如果 DMA 传输采用突发模式 (burst mode) 和周期窃取模式 (cycle-stealing mode) 时会争抢 ^对 处理器的内存带宽资源, 但若采用透明模式 (Transparent Mode) 就不会 ^对 处理器有影响。

如果存储器层次设计的好, CPU 可直接从缓存中读取数据, 就不用占用 ^{内存} 总线带宽资源。

1. 解:

(1) 集中仲裁: 将^{所有}总线请求集中起来, 利用一个特定裁决算法进行裁决, 可以分为三种:

a. 链式查询: 所有部件共有一根总线请求线, 当有部件请求使用总线时, 经此线发总线请求信号到总线控制器查询总线是否忙碌, 不忙碌便发总线响应信号通过BG线串行地一个一个查询哪个部件有总线请求, 优点是优先级固定, 控制线较少, 结构简单, 缺点是对硬件电路故障敏感, 且优先级不能改变

b. 计数器定时查询: 总线控制器收到总线请求信号并判断空闲时, 计数器开始计数, 计数值通过地址线^{设备}与设备匹配, 一致时, 该设备获总线控制权, 同时计数停止, 优点是计数初值可^点是“0”, 可^次由程序赋值优先级可调, 缺点: 控制线增加, 若设备 n 个, 则大致要 $(\log_2 n)$ 根线, 且控制较复杂

c. 独立请求: 每个部件都有一对总线请求(BR)和总线允许线(BG), 部件通过各自BR线发送请求信号排队, 并按一定优先次序进行仲裁, 通过对应BG线将允许信号给设备, 优点是响应快, 对优先次序控制灵活, ^{缺点:}控制线数量多, 若设备 n 个, 需要 $2n+1$ 条控制线, 1是反馈线

(2) 分布式仲裁: 不需要中央仲裁器, 每个设备有自己的仲裁器与仲裁号, 当设备有请求时, 仲裁器将仲裁号共享到仲裁总线, 并将总线上的仲裁号与自身仲裁号比较优先级, 若自己低, 则撤销自己仲裁号, 最终留在仲裁总线上的仲裁号的需求获回应, 优点, 所有设备都有平等机会访问总线, 优先级定义灵活, 缺点: 竞争消耗资源, 逻辑控制复杂

2. 解: APB: 低功耗, 低带宽, 用于连接低速外设, 如UART接口, 时钟模块, 无仲裁机制

AHB: 高性能, 高带宽, 用于连接处理器, DMA控制器等高性能模块, 有仲裁机制, 链式

AXI: 高性能高带宽, 可流水化设计, 应用于多处理系统, 支持乱序传输, Outstanding传输: 即对操作

ACE: 基于AXI扩展, 用于提供高级特性, 如缓存一致性和事务的原子性

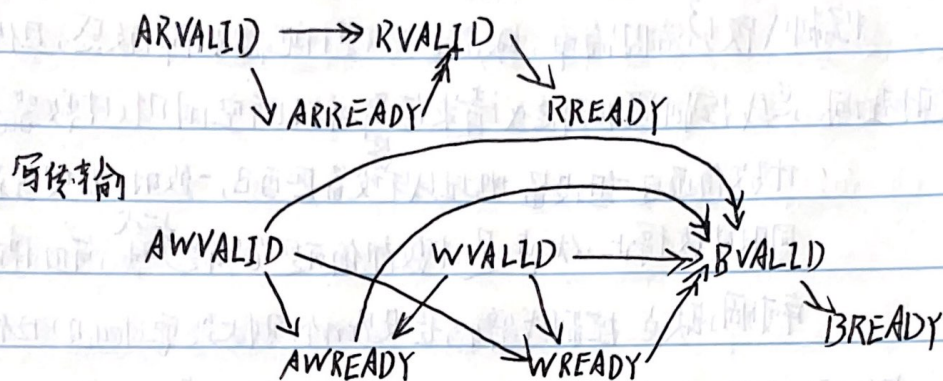
CHI: 用于^{连接}高级一致性协议, 接多个处理器和缓存于系统, 适用于服务器, 高性能计算

3. 解: (1) 共 5 个通道, 读地址 AR, 读数据 R, 写地址 AW, 写数据 W, 写响应 B 通道

因为读响应或这个操作是 (AR) 读地址当中完成的

(2) 读 VALD 信号发送后置位, READY 信号接收后置位, VALD 在完成传输前不能拉低
不能将 READY 信号置高作为置高 VALD 条件

读传输: (单箭头指向信号能在起点信号之前或之后断信; 双箭头指向必须在此之后断信)



保证事物正确性, 如写响应必须在写地址和写数据完成后才能表明, 同时防止冲突

B) 突发传输 (Burst transfer) 是指, 在地址线上进行依次地址传输后,

连续地传输多个地址相邻的数据。第一次传输的地址作为起始

地址, 根据突发传输类型不同:

后续数据存储地址在起始地址上递增 (INCR)

首先递增, 到达上限地址后回到起始地址, 继续递增 (WRAP)

后续数据都写入不断起始地址, 刷新起始地址上的数据 (FIXED)