

5.1 并行总线: 优点: 传输速度快, 并发性高

缺点: 并行总线对信号的同步传输, 接收时序要求高, 且传输距离受限

串行总线: 优点: 传输线路少, 硬件控制方面逻辑简单, 可长距离传输

缺点: 逐位传输速率较低, 对数据包也需要额外的解析

接口速率不同主要是由传输线路的数量决定的, 而串行总线的上限还受时钟频率的限制

从而, 并发性是决定速率的主要因素

$$5.2 \ 1) \text{波特率} = (1+7+1+1) \times 960 = 9600$$

$$2) \text{有效数据传输速率} = 960 \times 7 \text{ bit/s} = 840 \text{ byte/s}$$

5.3 起始信号 + 地址帧 + 读/写位 + ACK/NACK 位 + (数据帧 + ACK/NACK 位) * n +

中止信号

(每个字节的传输都伴随着应答信号的发送和接收)

- 2) I2C 仅由两条线路, 即 SDA 和 SCL 组成, SCL 为时钟, 所以仅有一条数据/地址通路, 双方可进行通信, 但不能同时双向通信, 为半双工
- 3) 起始条件: 主设备发出, 从高电平转为低电平, 同时时钟线保持高电平
- 停止条件: 主设备发出, 从低电平转为高电平, 同时时钟线保持高电平

5.4 1) 4 块磁盘的 RAID 0: $MTTF = \frac{N}{4}$

- 2) 设置 8 GB 的数据存储与 120 GB 的校验位存储, 此时 4 块磁盘中可允许 2 块磁盘同时出错而可被重建, $MTTF = \frac{3N}{4}$

5.5 寻道时间: 磁头从当前位置移动到目标磁道并消除抖动所需要的时间

(影响因素: 历史磁盘请求, 磁头的移动速度, 定位精度等)

旋转时间: 磁头移到目标磁道后, 目标扇区随盘片转动而经过磁头所需的时间

(影响因素: 磁盘转速, 扇区大小等)

数据传输时间: 磁头完成读出或写入所用的时间

(影响因素: 数据块的大小, 磁盘的传输速率等)

5.6 1) $12 \text{ KB} \times 240 \times 6 = 17280 \text{ KB} = 16.875 \text{ MB}$

2) $12 \text{ KB} \times 5400 \text{ r/min} = 1080 \text{ KB/s}$

3) $0.5 \text{ r} \times \frac{1}{5400 \text{ r/min}} = 5.56 \text{ ms}$

5.7 决定请求的最优执行顺序即 决定某个磁道上某个扇区访问的先后顺序, 根据不同的磁盘调度算法, 决定磁头依次访问过哪些扇区, 哪些磁道, 从而对寻道时间和旋转时间的总和产生影响

5.8 RAID 3 以字节为单位进行存储, 每次写入均需改变所有磁盘的数据以改变数据块, 同时需要改变奇偶校验位, RAID 4 以若干数据块为一个单位, 使得小数据写入时只需单独改变某个磁盘内的数据块, 和奇偶校验位, 同时相比发生翻转的位以计算奇偶校验位, 避免写入时读取所有磁盘以计算奇偶校验位, 从而实现了写入优化

5.9 I/O 请求减少即 λ 下降, $W = \frac{1}{\mu - \lambda}$ 下降, 单个磁盘处理 I/O 请求的响应时间下降
I/O 请求未减少时提升幅度为 $\frac{W'}{W} = \frac{\mu - \lambda}{\mu - \lambda'} = 1 + \frac{\lambda' - \lambda}{\mu - \lambda}$, λ 下降时提升幅度下降

5.10 会, 当 DMA 采用突发模式、交错模式时, 有可能会与 CPU 访存请求冲突,

通过对缓存等多级存储器的利用, 既可以开发出更高带宽的主存, 以实现 DMA 和 CPU 的同时访问, 也可以支持 DMA 与 CPU 各自的缓存, 减少访问同一存储单元的冲突

6.1 集中式仲裁方式: 将所有的总线请求集中起来, 利用一个特定的裁决算法进行裁决, 主要由中央仲裁器进行解决

分布式仲裁方式: 每个潜在的主模块都有自己的仲裁器, 当它们有总线请求时, 会将自己的仲裁号发送至仲裁主线, 由仲裁主线决定保留哪个仲裁号

优缺点: 集中式仲裁使用单一模块管理总线请求, 实现简单, 可避免竞争冲突等, 但仲裁器的工作状态会较大制约系统的工作性能

分布式仲裁: 允许系统并行并逐个响应, 在大规模系统中允许每个设备独立地进行仲裁决策, 具有可扩展性, 但设计上较为复杂, 需要考虑同步性问题与冲突等应用场景。集中式仲裁适用于小型化设备, 要求实时性与确定性的系统

分布式仲裁适用于大型、高并发性的系统, 要求可靠性和较好扩展性的场合

6.2 APB: 简单低功耗的总线协议, 连接低速低功率外围设备, 如 UART, 键盘等

AHB: 高性能总线协议, 支持高带宽、低延迟数据传输, 连接处理器, DMA, 存储器等

AXI: 高性能、更好的扩展性, 支持 outstanding 传输, 乱序传输, 窄带非对齐传输等复杂交互模式, 适用于高性能 SoC, 以及 AHB 所有的应用场景

ACE: 在 AXI 基础上增加缓存一致性扩展, 提供了缓存行操作和共享状态信息, 主要用于多核处理器系统

CHI: 专为处理器之间的高速通信设计, 支持复杂的缓存一致性和高带宽传输

6.3 (1) 读地址, 读数据, 写地址, 写数据, 写响应; 由于读操作是从从设备到主设备的, 主设备能成功接收到数据即可说明读响应成功, 因而无需设置读响应

(2) 发送信息的 AXI 端口的 VALID 信号不能依赖于接收信息的 AXI 接口的 READY 信号;

接收信号的 AXI 端口在其自身的 READY 信号有效前可以等待对方的 VALID 信号, 也可以直接发出

这些约束可以保证读写传输的正确性与一致性, 避免数据错乱

(3) 突发传输可以通过单次读写传输多个数据项, 通过首地址从设备可以计算出未来多次读或写的地址

突发类型: FIXED: 突发事务中每次访问相同的地址, 如读取或清空 FIFO 时

INCR: 在一次突发事务中, 每次传输时地址会递增, 递增量为传输大小

WRAP: 与递增型类似, 但地址到达上界后会从下界重新递增, 如 cache