

1.串行总线是一种将数据位逐个传输的总线，数据以连续的序列传送。它的优点包括：高速传输，线路简单，长距离传输。

串行总线缺点：传输延迟，数据对齐：在接收端，需要将逐位传输的数据重新组合成完整的数据包，这可能会引入一些复杂的对齐和解析问题。

并行总线是一种同时传输多个数据位的总线，各位数据同时传送。它的优点包括：高带宽，实时性。

并行总线也有一些缺点：线路复杂，成本增加，信号干扰。

造成接口速率不同的原因主要有以下几点：传输方式，信道带宽，传输距离，信号干扰。

2.

1) $960 \times 10 = 9600$

2)7680bps

3.

1) I2C 数据包由起始条件、地址字节、数据字节和停止条件组成。

2) 数据的传输只能在一个方向上进行。在传输过程中，主设备负责发送起始条件、地址字节和数据字节，而从设备负责发送应答信号和接收数据。因此，I2C 总线上的通信是通过交替的方式实现的，主设备发送数据后需要等待从设备的应答才能进行下一步操作。

3) 起始条件和停止条件是用于标识一次完整的通信过程的开始和结束的信号。起始条件是由主设备发送的低电平信号，停止条件是由主设备发送的高电平信号。通过在数据传输前发送起始条件，从设备可以准确识别出通信的开始，并准备好接收数据。而在通信结束时，主设备发送停止条件，使从设备知道本次通信已经结束，可以释放总线。

4.1) MTTF\_RAID0 =  $N/4$  小时

2)2 个磁盘 raid0 排列，分成两组，再以 raid 1 排成阵列。

Raid0 的 mtff 为  $N/2$  小时，故障概率为  $2/N$ ，由于 raid1 的正常概率为  $1-p^2$ ，所以总的的正常概率为  $1-4/N^2$ ，mttf 为  $1/(1-4/N^2)$

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的时间。旋转时间：磁头移动到目标磁道后，目标扇区随盘片转动经过磁头所需的时间。数据传输时间：磁头完成读出或写入所需用时间

6.

1) $6 \times 240 \times 12 = 17280$ KB

2) $5400/60 \times 12 = 10800$ KBps

3)1.6s

7. 磁盘调度的目标是优化磁盘访问的顺序，以减少磁头移动和寻道时间，从而提高数据访问速度和效率。以下是几种常见的磁盘调度算法：

先来先服务 (FCFS)：按照磁盘请求的到达顺序执行。这是最简单的调度算法，但可能会导致磁头在不同磁道之间频繁移动，造成较长的平均寻道时间。

最短寻道时间优先 (SSTF)：选择离当前磁头位置最近的请求进行执行。这个算法能够减少寻道时间，但可能导致某些请求长时间等待。

扫描算法 (SCAN)：磁头按照一个方向移动，依次执行请求，直到达到最边缘，然后改变方向继续执行。这个算法可以保证较低的平均等待时间，但可能导致一些请求长时间等待。

循环扫描算法 (C-SCAN)：类似于扫描算法，但在到达最边缘后，磁头直接返回到最内侧，形成一个循环。这个算法可以减少请求等待时间的变化，但可能会导致一些请求长时间等待。

最短时间优先 (STF)：根据每个请求的执行时间进行排序，优先执行执行时间最短的请求。这个算法可以最大程度地减少请求的等待时间。

不同的磁盘调度算法适用于不同的工作负载和使用场景。在实际应用中，系统会根据具体情况选择合适的调度算法，以平衡磁盘性能和响应时间。此外，现代磁盘驱动器通常会有自己的内部调度算法，它们尽可能地隐藏了物理磁道和寻道的细节，提供更高效的访问性能。

8. 在传统的 RAID 4 配置中，每次写入操作都需要计算并更新校验盘上的校验信息。这导致写入速度相对较慢，因为写入操作需要读取和写入多个磁盘上的数据。

为了提高写入性能，可以使用写入缓存 (Write Cache) 和写回 (Write-back) 策略。写入缓存允许将写入数据暂时存储在缓存中，以便更快地完成写入操作。写回策略则延迟了校验盘的更新操作，将校验数据暂时保存在缓存中，并在后续合适的时间点一次性地更新校验盘。这样可以减少校验盘的频繁读写，提高写入性能和读取速度。

9. 随着磁盘的 io 请求减少，生长率数值下降，更换磁盘前后提高的是消亡率。

根据公式  $W=1/(u-\lambda)$ ，生长率越小，W 的数值绝对值越大，更换前后的变化量越小，所以提升就越不明显。

10. DMA 设备 (Direct Memory Access，直接内存访问) 和处理器是共享系统内存资源的两个主要组件。它们在访问内存时可能会存在资源争用的情况。

存储器层次设计的优劣对 DMA 设备和处理器争抢内存带宽资源的影响很大。存储器层次结构的设计旨在通过缓存来减少对内存的访问次数，提高访问速度。

如果系统中存在高效的缓存层次结构，包括多级缓存和高速缓存 (如 CPU 缓存)，那么 DMA 设备和处理器很可能能够从缓存中获取所需的数据，而不必频繁地访问主存。这样可以减少 DMA 设备和处理器之间的资源争用，并提高整体系统性能。

另一方面，如果系统中的缓存层次设计较差或者缓存容量有限，那么 DMA 设备和处理器可

能更频繁地从主存中读取数据，导致资源争用更加明显。这可能会降低系统的效率，并导致性能瓶颈。

因此，一个优秀的存储器层次结构设计应该考虑到 DMA 设备和处理器之间的资源争用，并尽量减少它们之间的竞争。这可以通过增加缓存容量、改善缓存一致性协议、使用更高速的缓存和总线等方式来实现。

总结起来，良好的存储器层次设计可以减少 DMA 设备和处理器之间的资源争用，提高系统性能。

### 1. 集中式仲裁 (Centralized Arbitration) :

在集中式仲裁机制中，存在一个中央仲裁器 (Arbiter)，它负责协调总线上的各个设备对总线的访问请求。常见的集中式仲裁机制包括固定优先级仲裁和旋转优先级仲裁。

优点：简单实现，适用于设备数量较少的系统，易于调度和管理访问请求。

缺点：存在单点故障，当中央仲裁器发生故障时会导致整个总线系统不可用。随着设备数量的增加，仲裁延迟和性能瓶颈可能会增加。

### 分布式仲裁 (Distributed Arbitration) :

在分布式仲裁机制中，每个设备根据一定的算法或协议，通过总线上的仲裁信号进行仲裁。常见的分布式仲裁机制包括二进制计数器、旋转仲裁和随机仲裁。

优点：无需中央仲裁器，不存在单点故障。适用于设备数量较多的系统，能够提供更好的并发性和可扩展性。

缺点：实现较为复杂，需要设备之间进行仲裁协议的协调。可能存在冲突和死锁等问题。

### 基于优先级的仲裁 (Priority-Based Arbitration) :

基于优先级的仲裁机制中，每个设备被赋予不同的优先级，高优先级设备优先获得总线访问权限。

优点：能够根据设备的优先级灵活地分配总线带宽，适用于具有不同访问需求的设备。可以提供优先级控制和服务质量保证。

缺点：可能导致低优先级设备的长时间等待和资源饥饿，对于某些场景可能不公平。

不同的仲裁机制适用于不同的场景。集中式仲裁适用于设备数量较少且对实时性要求不高的系统。分布式仲裁适用于设备数量较多、对并发性和可扩展性要求较高的系统。基于优先级的仲裁适用于具有不同访问需求和优先级要求的系统，如多媒体应用、实时控制等。

### 2. APB (Advanced Peripheral Bus) :

特点：APB 是 AMBA 中最简单的总线协议，具有低功耗和低复杂度的特点。它适用于连接低带宽、低速外设，如配置寄存器、控制器等。

使用场景：适用于对性能要求不高的低速外设，例如串口控制器、定时器、GPIO 等。

### AHB (Advanced High-performance Bus) :

特点：AHB 是 AMBA 中性能较高的总线协议，具有高带宽和低延迟的特点。它支持多主设

备和多从设备，具有分片传输和优先级控制功能。

使用场景：适用于连接具有较高性能要求的内存、外设和处理器等模块，如存储器控制器、DMA 控制器、高性能处理器等。

AXI (Advanced eXtensible Interface)：

特点：AXI 是 AMBA 中最复杂和功能最强大的总线协议。它提供高性能、高带宽和低延迟的数据传输，并支持多通道和乱序传输。AXI 还包括 AXI4、AXI4-Lite 和 AXI4-Stream 等不同变种。

使用场景：适用于连接高性能处理器、外设和存储器等复杂模块，如图形处理器、高速存储器控制器、网络接口控制器等。

ACE (AXI Coherency Extensions)：

特点：ACE 是在 AXI 基础上添加了一致性扩展的协议。它提供了高性能的缓存一致性和事务协议，支持多核处理器和多级缓存体系结构。

使用场景：适用于具有多个处理器核心和共享缓存的系统，例如多核处理器、高性能计算系统等。

CHI (Coherent Hub Interface)：

特点：CHI 是 AMBA 的最新一代总线协议，专为高性能、多核心和多级缓存系统设计。它提供高度可扩展的缓存一致性、事务协议和低功耗特性。

使用场景：适用于具有高度并行处理、大规模多核处理器和复杂缓存层次结构的系统，如服务器、大型网络设备等。

3.

1) 读通道，写通道，写地址通道，读数据通道。AXI 协议没有设置独立的读响应通道，而是将读取响应通过读数据通道返回，是为了简化设计，数据对齐，以及带宽利用。

2)

在读写传输事务中，通道的握手信号时序需要满足以下依赖关系：

写事务的依赖关系：

写地址通道 (Write Address Channel) 在发送地址和控制信息之后，等待接收到有效的写响应 (Write Response) 信号之前，不能发送下一个写事务请求。

写数据通道 (Write Data Channel) 在发送数据之后，等待接收到有效的写响应信号之前，不能发送下一个写事务请求。

读事务的依赖关系：

读地址通道 (Read Address Channel) 在发送地址和控制信息之后，等待接收到有效的读响应 (Read Response) 信号之前，不能发送下一个读事务请求。

读数据通道 (Read Data Channel) 在发送读取的数据之后，等待接收到有效的读响应信号之前，不能发送下一个读事务请求。

这些依赖关系的约束是为了确保通信的正确性和一致性，以防止数据冲突和乱序操作。

3) 突发传输 (burst transfer) 是一种高效的数据传输方式，允许在单个事务中传输多个连续的数据或地址。

突发传输可以大大提高数据传输的效率，减少了传输的开销和延迟。它通过在单个事务中传输多个数据或地址，减少了事务的数量和相关的控制信号，从而提高了总线的带宽利用率。

AXI 总线协议中支持以下几种突发传输类型：

固定突发 (Fixed Burst)：在固定突发传输中，主设备发送一次事务请求，并在连续的地址范围内传输固定数量的数据。

递增突发 (Incrementing Burst)：在递增突发传输中，主设备发送一次事务请求，并在连续的地址范围内传输递增的地址和数据。每个地址对应一个数据。

递减突发 (Decrementing Burst)：在递减突发传输中，主设备发送一次事务请求，并在连续的地址范围内传输递减的地址和数据。每个地址对应一个数据。

持续突发 (Wrap Burst)：在持续突发传输中，主设备发送一次事务请求，并在连续的地址范围内循环传输一组数据。当传输到最后一个地址后，下一个地址将会是起始地址，形成循环。