No. Date

5.1 串行：优点：需要的物理连线数目少，硬件资源消耗少、功耗更低，稳定有明显的步负干优势

缺点：单次传输数据少，传输速率慢

并行：优点：可同时传输数据量更大，同步负下传输速率更快

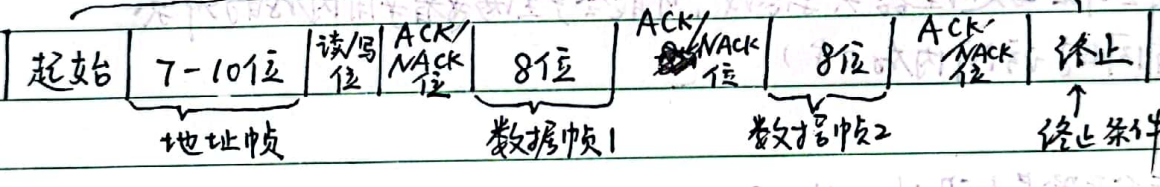
缺点：需要较多的物理连线及相应的硬件资源，功耗高，并行数据线间易产生干扰，不利于长距离传输。

速率不同的原因：单次传输数据量不同，步负率不同。

2. (1) $960\times(1+7+1+1)=9600$

(2) $960\times7=6720$

3. (1)



(2) I2C仅使用2根信号线，其一为时钟线，另一为数据线SDA，属于同步通信，输入输出共用一根线，不可同时进行，只能单向通信，所以为半双工。

(3) 起始：SCL为高电平 SDA由高向低跳变

终止：SCL为高电平 SDA由低向高跳变。

4. (1). RAID0 $MTTF=\frac{N}{4}$ 小时，（一块坏）

(2). 使用行-对角线双重奇偶校验的RAID-6形式。

5. 寻道时间：磁头臂控制磁头移动到所需寻找的磁道的正确位置并消除抖动所需要的时间

影响因素：磁头移动速度，移动距离，寻道策略

旋转时间：盘片旋转到使正确扇区处于磁头正下方消耗的延时

影响因素：盘片转速，转动圈数。

数据传输时间：传送一个扇区的数据所需时间。

影响因素：盘片转速，数据量大小。

NO. Date

6. (1) 12KB × 240 × 6 = 17280 kB.

(2) 5400 r/min × 12kB = 64800 kB/min

(3) 平均下来，相当于转半圈的时间 $\frac{60s}{5400 \times 2} = \frac{1}{180} s$.

（数据随机出现在一圈的

~~7-通~~ 7.8 题见附页电脑版.

9. $W = \frac{1}{\mu - \lambda} = W(\lambda)$ $W'(\lambda) = +\frac{1}{(\mu-\lambda)^2}$

~~不~~ I/O 请求减少 $\lambda \downarrow$ $(\mu-\lambda)^2 \uparrow$ $W(\lambda) \downarrow$

性能提升幅度（$W'(\lambda)$）下降.

10. DMA 控制器与处理器共享总线，可能会产生两者争用内存的冲突.
（当二者同时访问内存时）

通过存储器层次设计，例如 CaChe，可以让处理器更多地访问较低级别的存储结构，DMA 访问更高级别及主存，以避免争抢.
但若层次设计不合理，如缓存一致性不好，可能导致处理器频繁地从内存中重新加载数据，反而增加与 DMA 设备间的竞争.

第 5 章第 7 题：

通过使用一种称为"磁盘调度算法"的技术实现。磁盘调度算法旨在优化磁盘上的数据访问顺序，以减少寻道时间和旋转延迟，从而提高磁盘的读写效率。

以下是两种常见的磁盘调度算法：

先来先服务（First-Come, First-Served，FCFS）：这是最简单的调度算法，按照请求的到达顺序进行服务。然而，FCFS 算法没有考虑磁盘磁头的位置和移动，可能导致磁头在磁盘上无效地移动，从而增加了访问时间。

最短寻道时间优先（Shortest Seek Time First，SSTF）：SSTF 算法选择与当前磁头位置最接近的下一个请求进行服务。通过减少寻道距离，SSTF 算法可以显著减少磁盘访问时间。然而，SSTF 算法可能会导致一些请求长时间等待，即产生"饥饿"问题。

除了这两种算法，还有其他的磁盘调度算法，如电梯算法（Elevator Algorithm，也称为扫描算法或者扫描磁头算法）和最短旋转时间优先（Shortest Rotational Latency First，SRLF）等。这些算法基于不同的策略，考虑磁头的位置和移动，以最小化寻道时间和旋转延迟。

通过选择合适的磁盘调度算法，磁盘控制电路可以优化磁盘上的数据访问顺序，从而减少磁头移动和等待时间，提高磁盘的访问效率。这种最优的请求执行次序可以最大程度地减少磁盘访问时间，提高系统的整体性能。

第 5 章第 8 题：

在 RAID4 中，写入优化对于读取速度可以有正面的影响。

在 RAID4 中，写入操作需要更新数据块以及对应的奇偶校验块。为了提高写入效率，RAID4 采用了写入缓冲区（Write Buffer）或写入缓存（Write Cache）的技术。写入缓冲区允许将写入的数据暂时缓存在高速缓存中，然后以批量的方式写入磁盘。

由于写入缓冲区的引入，写入操作不需要立即写入磁盘，而是可以先暂存起来，从而减少了对磁盘的实际访问次数。这样可以提高写入操作的效率和吞吐量。当写入操作较为频繁时，写入优化可以更好地利用磁盘带宽，减少磁盘访问的开销。

此外，由于 RAID4 的奇偶校验块存储在单独的盘上，写入优化可以通过在写入操作期间只更新相关的数据块，而不需要更新整个条带（stripe）来提高写入速度。这样可以减少写入操作对整个条带的影响，从而更快地完成写入操作。

写入优化对读取速度的影响在 RAID4 中主要体现在两个方面。首先，通过减少写入操作对磁盘的实际访问次数，可以释放更多的磁盘带宽用于读取操作，提高读取速度。其次，由于写入优化只更新相关的数据块，不会涉及整个条带的更新，读取操作可以更快地定位到所需的数据块，进一步提高读取速度。

然而，需要注意的是，如果写入操作过于频繁或写入缓冲区过小，可能会导致缓冲区溢出或写入操作的延迟增加，从而对读取速度产生负面影响。因此，在 RAID4 中，适当配置和管理写入缓冲区的大小以及写入操作的优先级是重要的，以充分利用写入优化的好处并确保读取性能的稳定性。

第 6 章第 1 题：

常见的总线仲裁机制包括集中式仲裁、分布式仲裁和冲突检测仲裁。它们各有不同的优缺点和适用场景。

- 集中式仲裁：
- 优点：集中式仲裁简单且易于实现，只需要一个仲裁器来控制总线访问权。它可以提供公平性，确保每个设备按照一定的顺序获得总线访问权限。
- 缺点：集中式仲裁可能导致单点故障，如果仲裁器出现故障，整个总线系统将无法工作。此外，由于所有设备都需要通过仲裁器进行请求和响应，总线访问的延迟可能增加。

- 分布式仲裁：

•优点：分布式仲裁避免了集中式仲裁的单点故障问题。每个设备都负责自己的仲裁，并根据一定的算法来决定是否获得总线访问权限。
•缺点：分布式仲裁可能引起冲突和死锁问题。当多个设备同时请求总线访问权时，可能会出现冲突，需要额外的算法来解决。此外，设备之间的同步和协调也会增加设计和实现的复杂性。

•冲突检测仲裁：
•优点：冲突检测仲裁适用于总线上设备数量较少的场景。设备可以在发送数据之前先进行冲突检测，如果检测到总线正在被占用，则等待直到总线空闲。
•缺点：冲突检测仲裁可能导致访问总线的延迟增加，特别是当设备数量增多时，竞争情况会增加，可能导致更多的等待时间。此外，冲突检测仲裁需要额外的硬件逻辑来进行冲突检测，增加了总线系统的成本。

选择合适的总线仲裁机制取决于系统的需求和约束。一般而言，集中式仲裁适用于设备数量较少、对实时性要求不高的简单系统；分布式仲裁适用于设备数量较多、需要较高并发性和可扩展性的复杂系统；冲突检测仲裁适用于设备数量较少、对延迟要求较高的系统。需要根据具体的应用场景和系统设计要求综合考虑仲裁机制的优缺点，并根据性能、可靠性和成本等方面的权衡来选择适合的总线仲裁方案。

第 6 章第 2 题：

•APB（Advanced Peripheral Bus）：
特点：APB 是一种低功耗、低带宽的总线协议，适用于连接较简单的外设设备，如控制器、寄存器等。它具有简单的时序和灵活的连接性。
使用场景：APB 常用于连接较慢的外设，如 GPIO 控制器、定时器等，对带宽要求不高的场景。
•AHB（Advanced High-performance Bus）：
特点：AHB 是一种高性能的总线协议，具有高带宽和低延迟的特点。它支持多主机和多从机的配置，提供了分段访问、突发传输和流水线处理等功能。
使用场景：AHB 广泛应用于多个从设备的高性能数据传输场景，例如存储控制器、DMA 控制器等。
•AXI（Advanced eXtensible Interface）：
特点：AXI 是一种高性能、高带宽的总线协议，具有流水线传输和乱序访问的能力。它支持多主机和多从机的配置，并提供了高级的质量服务（QoS）机制，用于优化系统的性能和响应时间。
使用场景：AXI 广泛用于高性能处理器和外设之间的数据传输，适用于需要高带宽、低延迟和高度可靠性的系统。
•ACE（AXI Coherency Extensions）：
特点：ACE 是在 AXI 协议基础上添加了一致性扩展的总线协议。它提供了高级的缓存一致性和事务一致性支持，用于处理多核处理器之间的数据一致性问题。
使用场景：ACE 适用于多核处理器系统中，用于保持多个处理器的缓存一致性，并提供高性能的共享内存访问。
•CHI（Coherent Hub Interface）：
特点：CHI 是 AMBA 协议的最新版本，具有高度的可扩展性和一致性支持。它支持多核处理器系统中的高性能缓存一致性，并提供灵活的片上互连和内存控制器的接口。
使用场景：CHI 适用于高性能、复杂的 SoC 设计，尤其是需要支持多核处理器、高带宽内存和高度可扩展性的场景。

第 6 章第 3 题：

（1）AXI 总线包含的独立事务通道：AXI 总线包含以下五个独立的通道：
Read Address Channel：用于发送读地址和相关的控制信号。

Write Address Channel：用于发送写地址和相关的控制信号。
Write Data Channel：用于发送写入的数据和相关的控制信号。
Read Data Channel：用于返回读取的数据和相关的控制信号。
Write Response Channel：用于返回写操作的状态。

在 AXI 协议中，并没有设置独立的读响应通道。原因在于，读取操作的响应信息可以通过 Read Data Channel 一同返回，这些信息包括操作是否成功，是否有错误等。这样可以在不额外增加通道的情况下，实现读取操作的状态反馈。

（2）在读/写传输事务中，通道的握手信号时序需要满足的依赖关系：
AXI 协议采用两线握手信号机制，对于每个通道，都有两个握手信号：VALID 和 READY。发送方通过 VALID 信号表明它已经在对应的通道上放置了有效的信息，接收方则通过 READY 信号来表明它已经准备好接收信息。
数据或地址只有在 VALID 和 READY 信号都为高时才能被接收方接收。并且，接收方只有在 READY 信号为高时，发送方才能更新其 VALID 信号及相应的数据或地址。
设置这样的约束主要是为了实现流控制，以确保接收方在准备好接收数据时才会接收到数据，防止因接收方处理速度跟不上发送方而导致的数据丢失。

（3）什么是 AXI 的突发传输？有哪些突发传输类型？：
在 AXI 协议中，突发传输是指一次事务中连续传输多个数据项的操作。这种方式可以减少事务开销，提高数据传输效率。

AXI 协议中定义了以下三种突发传输类型：
FIXED Burst：在突发期间，地址保持不变。这种类型通常用于多次访问相同地址的情况，如访问 FIFO 缓存。
INCREMENT Burst：在突发期间，每传输一个数据项，地址就增加一次。这种类型通常用于顺序读写内存的情况。
WRAP Burst：在突发期间，地址以特定的边界进行循环递增。这种类型通常用于环形缓冲区的情况。