第五章习题

1.

串行总线的优点在于它所占空间小，也不会发生串扰之类的并行总线的问题，但是控制电路或者协议的设计难度比较大，而且线内频率较高，带宽较小等等。

并行总线的优点在于它带宽大，一次可传输的数据较多，但是有着串扰等问题限制最高频率使得传输速度难以提升。

2.

1) 960*10/1=9600(baud)
2) 960*7/1=6720(baud)

3.

1) I2C 的数据包是由起始条件、地址帧、读写位、ACK/NACK 位、数据帧 1、ACK/NACK 位、数据帧 2、ACK/NACK 位、终止条件构成的。
2) 因为 I2C 协议仅使用一条数据线 SDA。
3) 起始条件：在 SCL 为高电平时，SDA 由高向低跳变；
   终止条件：在 SCL 为高电平时，SDA 由低向高跳变。

4.

1) N/4 小时。
2) 每块磁盘分出 30G 的奇偶校验码，20G 的数据块位，可以达到 3N/4

5.

寻道时间：磁头比移动到正确位置并消除抖动所需要的时间。与磁头臂所需要移动的距离和移动速度有关，而且不是简单的比例关系。

旋转时间：盘片需要通过旋转来使得正确的扇区被旋转到磁头的正下方所需要的延时。与转速和扇区与当前位置的远近相关。

数据传输时间：传送一个扇区的数据所需要的时间。与盘片转速有关。

6.

1) 6*240*12=17280KB=16.875MB
2) 5400*72/60=6480KB/s=6.328125MB/s
3) (1/5400/2)min/r=(60*1000/5400/2)ms/r，平均旋转时间≈5.56ms

7.

磁盘控制电路通过决定请求的最优执行次序来减少磁盘访问用时的一种常见技术是磁盘调度算法。磁盘调度算法根据磁盘访问请求的特性和磁盘的物理结构，决定请求的执行顺序，以优化磁盘的访问效率。

以下是几种常见的磁盘调度算法：

1. 先来先服务（FIFO）：按照请求的到达顺序进行执行。这种算法简单直观，但可能导致请求按照随机顺序散布在磁盘上，造成较长的寻道时间。

2. 最短寻道时间优先（SSTF）：选择与当前磁头位置最接近的请求进行执行。这种算法能够减少寻道时间，但可能导致某些请求长时间等待。

3. 扫描算法（SCAN）：磁头按照一个方向移动，执行该方向上的请求，直到到达磁盘的

边界，然后改变方向继续执行。这种算法能够平衡请求的等待时间，但可能导致某些请求长时间等待或延迟较大。

4. 循环扫描算法 (C-SCAN)：类似于 SCAN 算法，但当磁头到达磁盘的边界时，直接返回到另一侧的起始位置，而不是改变方向。这种算法可以减少延迟，并提供较好的性能。

5. 最短访问时间优先 (SATF)：综合考虑寻道时间和旋转延迟，选择最短访问时间的请求进行执行。这种算法需要对磁盘的性能特性有较好的了解，但可以提供较高的访问效率。

通过选择合适的磁盘调度算法，磁盘控制电路可以根据不同的情况优化请求的执行次序，减少磁盘的寻道时间和旋转延迟，从而提高磁盘的访问效率和响应速度。然而，磁盘调度算法的实现也会带来一定的计算和判断开销，因此需要在提高访问效率和减少延迟之间进行权衡。

8.

在 RAID4 中，写入优化对于读取速度的影响是有限的。RAID4 是一种条带化存储方案，其中数据被分割成块并分布在多个磁盘驱动器上。每个数据块的校验信息被存储在一个独立的校验盘上。

RAID4 的写入操作包括以下步骤：

1. 读取原始数据块和校验信息块。

2. 修改数据块。

3. 计算新的校验信息并更新校验信息块。

4. 将修改后的数据块和校验信息块写回到磁盘驱动器。

5. 写入优化是通过将写入操作缓冲到磁盘控制器的缓存中，然后批量写入磁盘驱动器，以提高写入性能。这种优化在减少磁盘访问次数和最小化磁盘驱动器的旋转延迟方面非常有效。

然而，对于读取操作，写入优化对速度的影响是有限的。因为在 RAID4 中，读取操作仍然需要访问多个磁盘驱动器来获取所需的数据块。无论是否进行写入优化，读取操作都需要在条带中进行寻道和旋转延迟，并且读取速度主要取决于磁盘驱动器的性能和访问模式。

需要注意的是，RAID4 中的读取性能可以通过其他技术来进行优化，例如使用读取缓存、读取预取和磁盘调度算法等。这些技术可以提高读取操作的效率，减少寻道时间和旋转延迟，并进一步提高读取速度。但是，与写入优化相比，它们是独立的优化方法，对于写入优化本身并没有直接影响。

9.

如 $\lambda_1 < \lambda_2$，$W=1/(\mu-\lambda_1) < W=1/(\mu-\lambda_2)$，相同的服务能力性能提升的范围减小了。

10.

不同模式情况不同，突发模式会，周期窃取模式和透明模式不会。

存储器层次设计的优劣会对这个问题产生影响。存储器层次结构包括多级缓存和主存，其设计目的是提供更快的数据访问速度和更大的容量。存储器层次设计的优势在于通过缓存层次的引入，可以减少对主存的频繁访问，从而减轻了处理器和 DMA 设备之间对内存带宽的争抢。

具体来说，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如下：

1. 更大的缓存容量：如果系统的缓存容量足够大，可以容纳更多的数据和指令。这样，处理器和 DMA 设备可以更频繁地从缓存中获取数据，而不必每次都访问主存，减少了它们之间对内存带宽的竞争。

2. 更高的缓存命中率：当处理器或 DMA 设备需要访问的数据在缓存中已经存在时，发生了缓存命中。较高的缓存命中率意味着更少的访问主存的需求，减少了处理器和 DMA 设备之间的内存带宽争用。
3. 更快的缓存访问速度：较快的缓存访问速度意味着处理器和 DMA 设备可以更快地获取所需的数据。这样可以减少它们之间的等待时间，降低了对内存带宽的争抢。

总的来说，优化的存储器层次设计可以提供更好的数据访问性能和更好的内存带宽利用效率，从而减轻 DMA 设备和处理器之间的内存带宽争抢问题。较大的缓存容量、较高的缓存命中率和更快的缓存访问速度都有助于减少 DMA 设备和处理器之间的竞争，提高系统整体性能。

第六章习题

1.

常见的总线仲裁机制主要包括轮询机制和优先级仲裁机制。

1. 轮询机制：
   - 工作原理：在轮询机制中，各个设备按照固定的顺序依次请求总线的使用权。总线控制器依次查询每个设备是否需要访问总线，并根据设备的请求顺序分配总线的使用权。
   - 优点：
     - 简单易实现，硬件成本低。
     - 公平性较好，每个设备都有机会获取总线的使用权。
   - 缺点：
     - 效率较低，当总线上连接的设备数量较多时，轮询会导致较长的等待时间，降低总线的利用率。
     - 不适用于实时性要求较高的场景，无法满足对响应时间的严格要求。
2. 优先级仲裁机制：
   - 工作原理：在优先级仲裁机制中，每个设备都被赋予一个优先级，并根据优先级来确定设备访问总线的优先顺序。具有高优先级的设备会优先获取总线的使用权。
   - 优点：
     - 可以根据系统需求灵活设置设备的优先级，满足对不同设备的访问优先级要求。
     - 适用于实时性要求较高的场景，可以根据实时性需求给予重要设备更高的优先级。
   - 缺点：
     - 复杂度较高，需要额外的硬件支持和调度逻辑。
     - 可能会导致低优先级设备长时间等待，造成资源浪费。
     - 如果优先级设置不合理，可能会导致低优先级设备无法获得访问总线的机会，造成饥饿现象。

轮询机制适用于设备数量较少、对实时性要求不高的场景，具有简单、公平的特点，但效率相对较低。优先级仲裁机制适用于对实时性要求较高、需根据不同设备的优先级灵活调度的场景，能够满足对响应时间的严格要求，但复杂度较高。在具体应用中，需要根据系统需求和设备特点选择适合的仲裁机制来平衡公平性、实时性和效率。

2.

1. APB（Advanced Peripheral Bus）：
   - 特点：
     - 简单、低功耗的总线协议。
     - 采用点对点连接方式，支持多主设备和多从设备。
     - 针对低带宽外设设计，传输速率较低。
   - 使用场景：
     - 适用于较简单的外设，如 UART、GPIO 等。
     - 对带宽和实时性要求不高的应用场景。
2. AHB（Advanced High-performance Bus）：
   - 特点：
     - 性能较高的总线协议，支持高带宽和实时性要求。
     - 采用多主设备和多从设备的分布式总线架构。
     - 支持 burst 传输和 split 传输方式，提高数据传输效率。
   - 使用场景：

- 适用于中等复杂度的外设和存储器控制器，如 DMA 控制器、SRAM 控制器等。
- 对带宽和实时性要求较高的应用场景。

3. AXI（Advanced eXtensible Interface）：
- 特点：
  - 高性能、高带宽的总线协议。
  - 采用点对点连接方式，支持多主设备和多从设备。
  - 支持 out-of-order 传输、乱序响应和阻塞机制。
- 使用场景：
  - 适用于复杂的外设和存储器控制器，如图形处理器、DDR 控制器等。
  - 对带宽和实时性要求非常高的应用场景。

4. ACE（AXI Coherency Extensions）：
- 特点：
  - 在 AXI 基础上增加了一致性扩展，用于处理多处理器系统中的缓存一致性问题。
  - 提供了一致性协议，确保多个处理器的缓存内容保持一致。
- 使用场景：
  - 适用于多处理器系统中，需要处理缓存一致性的应用场景。

5. CHI（Cache Coherent Interconnect）：
- 特点：
  - 高性能、高带宽的缓存一致性互联协议。
  - 提供了更高级别的一致性支持和扩展性。
  - 支持多级互连网络拓扑结构。
- 使用场景：
  - 适用于大规模多处理器系统，对缓存一致性和性能要求非常高的应用场景。

AMBA 总线协议提供了多个层次的总线接口，适用于不同级别和复杂度的系统设计。APB 适用于简单外设，AHB 适用于中等复杂度的外设和存储器控制器，AXI 适用于复杂外设和存储器控制器，ACE 用于处理缓存一致性问题，而 CHI 则是更高级别的缓存一致性互联协议，适用于大规模多处理器系统。选择合适的总线协议取决于系统的性能要求、复杂度以及缓存一致性需求。

3.

1) AXI 总线包含以下独立的事务通道：
- 读数据通道（Read Data Channel）：用于从从设备读取数据的传输。
- 写数据通道（Write Data Channel）：用于向从设备写入数据的传输。
- 写地址通道（Write Address Channel）：用于指定写入操作的地址和其他控制信息。
- 写响应通道（Write Response Channel）：用于返回写操作的响应状态。

协议没有设置独立的读响应通道是因为在 AXI 总线中，读响应的传输与读数据的传输是通过同一个通道进行的。读数据通道中的数据和读响应的状态信息被打包在一起传输，这样可以减少总线上的信号线数量和复杂度。

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：
- 读传输：读地址通道和读数据通道的握手是基于读地址通道的读请求发起方和读数据通道的数据提供方之间的协商。读地址通道的请求必须先到达，并且被接收后，才能进行数据传输。
- 写传输：写地址通道和写数据通道的握手是基于写地址通道的写请求发起方和写数据通道的数据接收方之间的协商。写地址通道的请求必须先到达，并且被接收后，才能进行数

据传输。

这样的约束是为了确保数据传输的正确性和一致性。通过依赖关系的设置，保证了在读/写事务中的通道握手信号按照正确的顺序进行，避免数据的错误读取或写入。

3) AXI 的突发传输（Burst Transfer）是一种高效的数据传输机制，可以在一次地址传输中连续传输多个数据。突发传输可以减少总线开销和传输延迟。

AXI 总线支持以下突发传输类型：

- 固定突发传输（Fixed Burst Transfer）：在连续地址范围内进行连续传输，传输的数据长度固定。

- 递增突发传输（Incrementing Burst Transfer）：在连续地址范围内进行连续传输，每个地址按照递增规则进行。

- 未对齐突发传输（Unaligned Burst Transfer）：传输的起始地址和数据长度不是对齐的，需要进行额外的处理。

突发传输可以提高数据传输的效率，减少总线占用时间，特别适用于需要连续传输大量数据的场景，如图像处理、存储器访问等。