1. Suppose we use the asynchronous bus of Fig 7.6, and the CPU attempts to read an address which does not correspond to any valid memory or I/O device. What might happen, and can we do anything about it?
   1. What might happen is that the lines will take the time for propagation/skew and then just turn off the signal since an address does not exists. There also might be some sort of error. The best way to fix it would be to check during the master decode to see if the device exists.
2. An arbiter controls access to a common resource. It uses a rotating-priority scheme and responds to requests on lines R1 through R4. Initially, R1 has the highest priority and R4 the lowest priority. After a request on one of the lines receives service, that line drops to the lowest priority, and the next line in sequence becomes the highest-priority line. For example, after R2 has been serviced, the priority order, starting with the highest, becomes R3, R4, R1, R2. What will be the sequence of grants for the following sequence of requests: R3, R4, R4, R2? assume that the last three requests arrive while the first one is being serviced.
   1. R1, R3, R4, R2
3. The bus protocol of Figure 7.4 specifies that the slave device should send its data only in the second phase of the clock.
   1. It is possible that some device may recognize its address and is ready to send data sooner. Why is it not allowed to do so? Would the processor receive wrong data?
      1. The device will be able to recognize its address and be able to send data sooner than it is told to do so by the clock, but it is not allowed to because of the control signals, delay, and the data could be incomplete. The first portion/ first cycle helps the process to know how many cycles to dedicate to this task, since different devices can require different cycle amounts; so getting and sending data before the appointed time by the clock could result in the wrong data.
   2. Would any other problem arise?
      1. Some other “problems” are that the device could be ready to get/send the correct data, but the clock limits it. The book mentions how some cycles are customized so not everything takes the same amount of cycles and can be more efficient.
4. PCI Express uses the 8b/10b encoding (https://en.wikipedia.org/wiki/8b/10b\_encoding) to send data. What would be the encoding for the raw data 00101110 11101100?
   1. Answers are reversed since the wiki article says it reads from
   2. 00101110 ->
      1. 00101110 -> 01110[E-A] 100[H-F]
      2. 01110 -> 011100 [A-E, I]
      3. 100 -> 1101 [F-J]
      4. 011100 1101
   3. 11101100->
      1. 11101100-> 00110[E-A] 111[H-F]
      2. 00110 -> 011001 [A-E, I]
      3. 111 -> 0111 [F-J]
      4. 011001 0111
   4. Full Answer
      1. 011100 1101 011001 0111