CS 250 Spring 2017 Homework 11

Due 11:58pm Thursday, April 20, 2017

Submit your typewritten file in PDF format to Blackboard.

1. A serial interface operates at a throughput of 10 million bits per second and requires 100 microseconds to configure (prepare) a packet of bits to be transmitted regardless of the size of the packet.
   a. What is the effective throughput, expressed first in packets/second and then in bits/second, for this interface for an infinite series of identically-sized packets for sizes 1 bit, 100 bits, 10^4 bits, and 10^6 bits? Express answers rounded to 6 significant digits.

   $$total\ time\ per\ packet = configuration\ time + \left(\frac{packet\ size\ in\ bits}{interface\ throughput}\right)$$

   1 bit total time = 1.001*10^-4 sec, 9990.01 packets/sec = **9990.01 bits/sec**
   100 bits total time = 1.1*10^-4 sec, 9090.91 packets/sec = **909091 bits/sec**
   10^4 bits total time = 0.0011 sec, 909,091 packets/sec = **9.09091*10^-6 bits/sec**
   10^6 bits total time = 0.1001 sec, 9.9901 packets/sec = **9.99001*10^-6 bits/sec**

   b. What is being amortized?
   The throughput is what is being amortized. This is because as the packets grow in size (therefore carrying more bits), less packets are needed to be sent. This means that more data can be sent in fewer but larger packets instead of using smaller packets at a more rapid rate.

2. How many simultaneous transfers can occur over a crossbar witching fabric of N inputs and M outputs?
   The amount of simultaneous transfer that can occur would be the smaller value of either M or N. Since a single input is linked to a single output, the limiting factor would be the smaller number of the two as it would not have a corresponding input/output.

3. Assume that a RISC processor takes two microseconds to execute each instruction and an I/O device can wait at most 1 millisecond before its interrupt is serviced. What is the maximum number of instructions that can be executed with interrupts disabled?
   Given the above information, our RISC processor would execute a maximum number of 500 instructions before the interrupt is serviced.
   1 millisecond = 1,000 microseconds 1,000 / 2 = 500 instructions.

4. Suppose a user installs ten devices that all perform DMA into a single computer and attempts to operate the devices simultaneously. What components of the computer might become a bottleneck?
   The bottleneck would be the central bus. Considering how DMA (Direct Memory Access) works, each device contains its own process to handle the fetch and store operations, meaning there is already a direct path between the devices and memory by using the central bus. This in turn means that the central bus could be a bottleneck if it is not able to handle the simultaneous operations of these ten devices.

5. A user invokes an app that writes a file. The app displays a progress bar that shows how much of the file has been written. Just as the progress bar reaches 50%, the power fails and the computer crashes. When the power is restored and the computer rebooted, the user discovers that less than 20% of the file was actually written. Why did the app report 50%?
   The app reported 50% because that percent was how much was in the buffer, not necessarily what had been written to the disk yet. As the buffer is filled it flushes and writes the filled buffer to the disk, however the power reboot caused it to stop whilst it was filling the buffer which forced the buffer to clear and never write that full 50%, thus showing that only less than 20% had been truly written (the rest was lost in the buffer on reboot).

6. A user invokes an app that writes a file. The app displays a progress bar that shows how much of the file has been written. The progress bar has been advancing quickly, but just as the progress bar reaches 99%, the progress bar seems to freeze, then after a delay the bar show 100%, then disappears, and the app GUI moves on to another phase of activity. What might explain why the progress bar seemed to freeze at 99% for a time before reporting 100%?
   The progress bar seemed to 'freeze' at 99% because at that point all the data had been loaded into the buffer and it was just a matter of writing everything in the buffer to reach that final 100%, and thus complete the process and move on.

7. Discuss the advantages of multiplexing with respect to buses. What are the two major disadvantages? Which of these two disadvantages is minimized, even rendered moot, by Moore's Law?
   The advantages of multiplexing with respect to buses is that we can improve performance whilst also reducing the number of data lines. One major disadvantage is that multiplexing requires a sophisticated bus protocol which means in turn means more complex bus interface hardware. This of course is minimized (or even rendered moot) by Moore's Law since we can always add more transistors to the hardware if needed. Another major disadvantage is that due to the fewer data lines the multiplexing process takes more time. This is because the store operation now requires two bus cycles, one to transfer the address and one to transfer the data items.

8. What are the two types of bus error?
   The two types of buss error are unsigned address error and address conflict error. An unsigned address error occurs when a processor attempts to access an address that is not assigned to any interface. An address conflict error is when two or more interfaces correspond to a single address.

9. Run a system info command on your computer, and use its output to find at least three different buses that your computer contains. For each bus, (1) describe in specific detail the physical hardware from which it would be constructed, (2) name the units within your computer that are connected to the bus (if any) and the external units (if any) typically connected to the bus, (3) classify the bus as serial or parallel, (4) state if the bus is proprietary or if is it standard and when was its standard specification released and what entity released the spec, and (5) state how the bus addresses are configured (manually per I/O device, by hardwiring on the bus, or automatically), and (6) state for automatically configured buses, state whether the bus is hot-pluggable or not. The web, especially Wikipedia, can be a helpful resource.
    1.1. Universal Serial Bus (USB). Specifically USB Root Hub. This is utilizes a set of logical pipes to transfer information between the host and the controller. The USB port on any computer is where this would be located as the hardware makes the connection there through grounded wiring and transfers data from an external device to the internal components of the PC. This is a serial bus and is also proprietary as it comes with all standard PCs and is a standard in general. Each device plugged in is assigned a 7-bit address automatically and USBs are in fact hot pluggable.
    1.2. Corsair Bus. This is a bus I found within my computer that must be operated within the CPU (possibly even GPU, or on-board the keyboards own processor!) but it was not clear for me to find out. This bus allows me to communicate via software on my computer as input to change RGB color values on my keyboard, mouse, and interior case lightings. The components involved would of course be the processor and all the Corsair components (keyboard, mouse, and internals). This is not hot pluggable as that would likely clear the saved settings/data for the settings. I believe this is a serial bus, not a parallel bus. This is a proprietary bus that Corsair releases with their products as a drive to communicate specifically with their hardware and products.
    1.3. Composite Bus Enumerator. This is a Microsoft Windows driver that allows for communication between hardware and connected device. This can tie in with all the pre-installed components to allow the CPU, RAM, GPU, etc. to ‘talk’ to one another. I believe this is a parallel bus. This is a standard bus that all of the latest Windows should have. This is not a hot-pluggable bus. I believe this is launched and monitoring the internal computer components during regular computer usage.