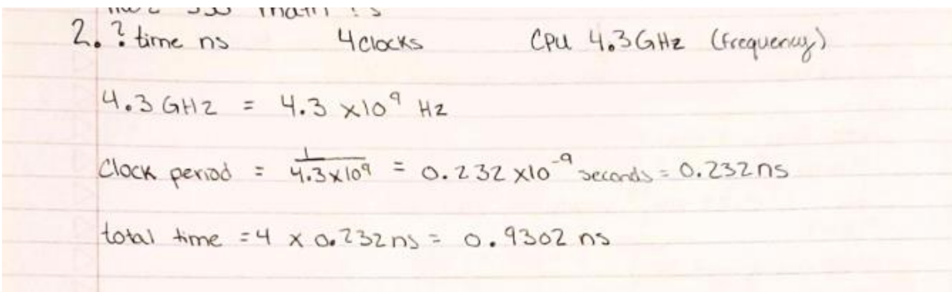
1) The principle of equivalence states that any task that can be done with software can be done with hardware. This goes in the opposite direction, and it can be said that any task that can be achieved with hardware can also be achieved with software. In general, hardware is faster than software but is more expensive to implement. Software on the other hand is slower but cheaper. Operations that we do a lot are often put in the hardware to take advantage of these tradeoffs. This principle is used to get the most out of your design. For example, if you have the task of converting analog signal to a digital one and you perform this task often it is more beneficial to put this in the hardware because it will run fast when it is needed. The cost will be offset by the task running quickly often. This conversion can happen in software, or we can build hardware that does this conversion for us.

2) The picture below shows my work for this problem. The unknown variable we are solving for is time. We have four clocks at 4.3 GHz. The first thing I did was find the period of the clock which was 0.232 ns. This number was then multiplied by 4 to get the total time because we have four clocks. The result was 0.9302 ns.
Answer: 0.9302ns



2. ? time ns   4clocks   CPU 4.3GHz (frequency)

4.3 GHz = 4.3×10^9 Hz

Clock period = $\frac{1}{4.3 \times 10^9} = 0.232 \times 10^{-9}$ seconds = 0.232ns

total time = 4 × 0.232ns = 0.9302 ns

3) The picture below is my work for problem 3. I started by counting the number of CPU clocks and system clocks. There are 3 CPU clocks and 4 system clocks. Next, I worked on finding the period for a CPU clock. This was 0.33ns which we multiplied by three to get the total time for CPU clocks. This total time came to 1 ns. Then, I found the period for the system clock which was 0.384ns. I multiplied this time by 4 for the four system clocks and got a total time of 1.538ns. Finally, to get the total time of the operation I added the total CPU time and the system clock time together (1ns + 1.538ns) and got 2.538ns.
Answer: 2.538ns

3. CPU → 3 clocks $\quad$ 3 GHz → 3×10^9 Hz
System clocks → 4 clocks $\quad$ 2600 MHz → 2600×10^6 Hz

Clock period for CPU:

$$\frac{1}{3 \times 10^9} = 0.3\bar{3} \times 10^{-9} = 0.33 \text{ ns}$$

we have 3 CPU clocks → $0.33\text{ns} \times 3 = 1\text{ns}$

Clock period for system clocks:

$2600 \times 10^6 = 2.6 \times 10^9$ Hz

$$\frac{1}{2.6 \times 10^9} = 0.3846 \times 10^{-9} = 0.384 \text{ ns}$$

we have 4 system clocks

$4 \times 0.384\text{ns} = 1.538\text{ns}$

total $= 1\text{ns} + 1.538\text{ ns} = 2.538\text{ ns}$

4) The picture below shows my work for problem four. The first thing I did was find the time it took for one clock. The result of this was 0.25ns. The next thing I had to figure out was the number of transfers needed. For this problem we needed 0.25Y transfers (where Y is Yottabyte). This was found by dividing the 2Y by 8 bytes (the size of the bus in Bytes). Finally, I multiplied the number of transfers by the time we got in step one and got 0.0625Ps (where Ps is Petta seconds).
Answer: 0.0625 Ps

4. 2 yottabyte $\quad$ 64 bit bus $\quad$ transfer clock 4GHz
1 clock per transfer → every clock transfers 64 bits

How long is 1 clock?

4GHz → 4×10^9 Hz

$$\frac{1}{4 \times 10^9} = 0.25 \times 10^{-9} \text{ seconds} \Rightarrow 0.25 \text{ ns}$$

How many transfers?

8 bits = 1 byte

64 bits = 8 bytes $\quad$ 8 bytes per transfer

$\frac{2}{8} = 0.25$ Y transfers

$$(0.25 \times 10^{24}) \times (0.25 \times 10^{-9}) = 0.0625 \times 10^{15} \text{ seconds}$$

0.0625 Ps

Ps = petta seconds

5) The picture below shows my work for problem 5. In this problem I started by finding the total length. This was found by multiplying the average length by the number of gates. This gave me 6400nm. Next, I took the distance and divided it by the speed of light to get time. This gave me 21.3fs (Femto seconds). Then, I took this time and turned it into the frequency. This resulted in 0.0469PHz (Peta hertz).

Answer: 0.0469 PHz

5. Speed of light 3×10^8 m/sec   Avg length 50nm   gates 128

total length = 50nm × 128 = 6400nm ⇒ 6400×10^{-9} meters

6.400×10^{-6} meters

Time $\dfrac{6.4 \times 10^{-6}\text{ meters}}{} \bigg| \dfrac{\text{second}}{3 \times 10^8 \text{ m}} \bigg| = 2.1\overline{3} \times 10^{-14}$ seconds

21.3×10^{-15} seconds

frequency $\dfrac{1}{21.3 \times 10^{-15}} = 0.0469 \times 10^{15}$ seconds or 0.0469 PHz

$0.0469 \times 10^{15} = 46.9 \times 10^{12} \sim> 46.9$ THz

6) Abstraction is a critical concept in computer architecture because it allows us to make a change in one layer of the computer's architecture and not necessarily have to worry about messing up the other layers in the abstraction. It also allows us to understand the minimum levels we need to do our job. If you work primarily in one level of abstraction you do not need to understand all the intricacies of the other layers. Without abstraction, any change we made to the computer could result in a redesign of a lot of the software.

Ex1) Memory

Memory is one example of abstraction. Memory is broken down into registers, L1 cache, L2 cache, main, and storage types. The cost and speed decreases down this list. The abstraction in this case is that when a program accesses memory it does not care where it is from in this stack.

Ex 2) Internet protocols (cs 235 networks)

In the case of internet protocol abstraction, the computer knows the only piece the computer sending the information cares about is the destinations address. How it reaches that address is up to other pieces of the internet protocols. The internet protocols are abstracted into the categories of application protocols, transport layer protocols, internet layer protocols, and network interface hardware. Each piece supports a different part of the process of sending information via internet protocols.

Ex 3) Modern computing systems

Modern computing systems also show abstraction. In this case the levels go from 0 to 6. Level 0 is digital logic (circuits, gates), level 1 is control (microcode or hardwired), level 2 is machine (instruction set architecture), level 3 is system software (operating system and library code), level 4 is assembly language, level 5 is high-level languages (c++/Java), and level 6 is the user (executable programs). The user at level 6 is often unaware of the levels beneath them and only delves into those layers when they have a job to do within a level lower in the abstraction.