CprE 381 Homework 6

[Note: This homework's purpose is to increase your comfort at calculating and analyzing the performance of processors. By the end, you should be able to accurately estimate how a change to software or hardware will likely impact the overall execution time of an application.]

1. Amdahl's Law

Your company builds hardware for doing machine learning inference for image classification. 'Inference' is predicting the class of an image based on the model you learned using sophisticated machine learning algorithms. While everyone is busy trying to develop the next Neuromorphic or Quantum computing chip, your supervisor assigns you the tasks of optimizing the execution of the time of the only application that actually generates the $$$ for your company (i.e., image classification of cats). For image inference: **Convolution** is the most fundamental operations which consists of convolving or "sliding" a filter (sometimes called the learned kernel 2D array) of size k*k across the input image of size n*n which generates an output of size (n-k+1)*(n-k+1). If you want more information about this sort of convolution, you can view the following video: https://www.youtube.com/watch?v=XuD4C8vJzEQ&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF&index=2.

The attached C and MIPS code implement a vertical edge detector that is an important portion of the cat image classification algorithm. As you can see, if there is a hardware floating point multiplier in the system the program will use * that will compile to mul.s MIPS instructions, otherwise it will use library call that performs software floating point multiplication. Your specific task is to determine whether or not you should add a hardware multiplier to your MIPS processor.

a. Based on your understanding of the MIPS and the C code *estimate* the total number of clocks cycles the application will require to execute on your single-cycle

processor. Also estimate the fraction of these cycles that the software multiplication function requires. *[Note: this is a somewhat open-ended problem and you can make any reasonable assumption. Please state all the calculations and assumptions you make. Note that you can actually run the MIPS code on MARS with and without the hardware multiplier to get instruction counts.]*

*Loops: 4 * 4 * 3 * 3 = 144 iterations for the for loops*
Assuming that each for loop has 3 instructions: slt, beq, increment
*144 * 3 = 432*
*432 * multiply instructions*
Compiling program you get 14 instructions for the multiply process, feels like it could be more because of the data3.raw.sign= data1.raw.sign^data2.raw.sign; operation but making assumption it is 14

*432 * 14 = 6048*
Actual 19644

14/432

b. Based on the above question identify which part of the application will benefit from the hardware multiplier and why? What is maximum speed up possible? *[Again, you will have to make assumptions regarding the multiplier unit.]*

The multiply function instructions will be reduced. The for loops number of instructions should stay the same

If the floating point multiplication was implemented in hardware I assume that it would be a single cycle process. Not depending on the time this instruction takes to execute it could increase our critical path.

This means that you would do two loads and then a signal instruction for the multiplication of the floating point numbers.

432 * 3 = 1296

Assuming same frequency: Et old / Et new = 6048/freq / 1296/freq = 6048/1296 = 460% speed up

c. What would the maximum speedup be if your hardware multiplier slowed the clock frequency by 25%? *[You can still solve this by using Amdahl's law—you just have to appropriately adjust the equation from b.]*

6048/1 / 1296/.75 = 350% Speed up

This is not consistent with the MIPS tests I'm sure it comes from my assumptions between the software and hardware implementation of floating point multiplication

2. Pipelining Cycle Time
   a. Assuming the following worst-case latencies for components, what is the cycle time for the pipelined processor in Figure 4.51 (COD) from your textbook? You must quantitatively justify your answer (e.g., specify what set's the cycle time and why it set's the cycle time).

I-Mem	Adder	MUX	ALU	Reg Read	D-Mem	Sign Extend	Shift Left-2	Control	ALU Control	AND gate
220ps	70ps	30ps	90ps	100ps	250ps	15ps	5ps	40ps	20ps	10ps

The instruction that takes the longest sets the cycle time. If you set it to anything lower that instruction could still be executing by the time the next positive clock edge creating a race condition.

In this case it is the D-Mem which is 250ps.

b. Compared to a single-cycle processor (Figure 4.17 COD) with the above component latencies, what would the pipeline design's CPI need to be in order to benefit Performance?

ET(mc) = IC * CPI * cycle period
ET(sc) = IC * CPI * cycle period

IC cancel out the same for both

MC has a cycle period of 250 ps
SC has a cycle period of critical path = 220(IMEM) + 90(ALU) + 30(MUX) + 100(RegRead) + 30(MUX) + 250(Dmem) + 30(Mux) = 750 ps

Performance = 1/ET

(1/ET(MC)) > (1/ET(SC)) = (1/(250*CPI(MC))) > (1/(750*CPI(SC))) = 1/250*CPI(MC) > 1/750

= 750/250 * CPI(MC) > 1 = 1/CPI(MC) > ⅓ = CPI(MC) < 3