EECS 151/251A Homework 7

Instructor: Prof. John Wawrzynek, TAs: Christopher Yarp, Arya Reais-Parsi

Due Monday, Apr 8th, 2019

Problem 1: Fast Unsigned Integer Comparison [15 pts]

Consider the design of a combinational logic circuit for comparing unsigned integers. The circuit accepts two n-bit unsigned integers, A and B, and generates an output, eq, that is equal to 1 iff A and B are equal, and another output, lt, that is equal to 1 iff A is less than B. We would like the circuit to have relatively small delay for large n, so you need take an approach that will lead to delay scaling with log(n). One solution would be to use a fast subtractor circuit, but that is not the correct answer here. *Hint: Divide the input words in half and think about defining the function recursively.*

Sketch out enough of your circuit so that we understand your approach and detailed circuits.

Problem 2: Extending RISC-V [8 pts]

Based on the RISC-V datapath presented in lecture that included up to the `lw` instruction, draw a modified single cycle datapath that could execute a new add instruction named `addm` which takes one of its operands from memory and has the following behavior:

```
Reg[rd] <- Reg[rs2] + DMEM[ Reg[rs1] + offset ]
```

You don't need to describe how the instruction would be encoded in the instruction word.

Problem 3: 3 Stage RISC-V Pipeline Branching [8 pts]

Based on the RISC-V single cycle datapath in the lecture slides, draw a simplified version of the datapath that includes just those components necessary for the branch instructions.

Now, draw another version of the datapath with modifications necessary to permit the 3-stage pipelining scheme with the "predict 'not taken' " strategy described in lecture. You don't need to show the details of how to kill an instruction. However, remember that in this approach only one instruction is killed when a branch is taken. For this part assume that the instruction memory has *asynchronous read.*

Problem 4: 3 Stage RISC-V Pipeline [8 pts]

Using the RISC-V datapath presented in class that includes the `lw` instruction, draw a new version of the datapath that is modified to permit the 3-stage pipelining scheme described in lecture.

Remember that the load delay should be only 1 cycle. You do not need to show the details of instruction stall in the case of an instruction dependent on the load. Assume that both the IMEM and DMEM have asynchronous read.

Problem 5: 3 Stage RISC-V Pipeline with Synchronous Memory [6 pts]

Now consider how your answers to the previous two problems would change if the memories we used for IMEM and DMEM had *synchronous read* instead of *asynchronous read.*

Synchronous read is defined as follows. *On the rising edge of the clock, the read address is registered on the read port and the stored data is then sent to the output.* We assume the read address hold time is short (similar to registers), and the access time is long compared to the clock period. (That is the reason we dedicate an entire pipeline stage to instruction fetch and to data memory access.)

Problem 6: 3 Stage RISC-V Pipeline Bypassing [5 pts]

We have a 3-stage pipelined RISC-V datapath that passes the 32-bit instruction through the pipeline. The instruction register separating the I and X stages is $IReg_{IX}$ and between the X and M stages is $IReg_{XM}$. As shown in page 63 of the lecture notes, the ALU has a bypass mux which allows ALU data hazards to execute without any delay. Describe the logic needed in the controller for correct operation in the case of back to back r-type instruction execution. You do not need to show detailed circuit diagrams, but describe precisely what operation(s) need to be done.

Problem 7: Line Drawing Accelerator [20 pts]

Draw a block diagram for the design of a datapath and controller for an implementation of the line-drawing accelerator presented in class. For this problem, you may assume that $x_0 < x_1, y_0 < y_1$, and slope is $\leq 45°$.

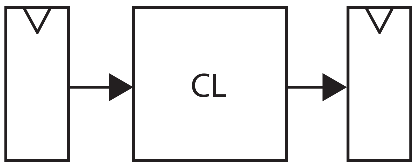
Try to minimize the number of cycles per output result.

You don't need to show your design to the gate level and may use components at the level of arithmetic blocks, comparator blocks, muxes, registers, etc. Make sure you show the details of how the looping is controlled.

Assume that the x_0, x_1, y_0 and y_1 values are available in registers. The outputs from the accelerator should be a signal, wr, that indicates when it is time to write a pixel, and x, y which define the pixel location to write.

Problem 8: Energy Efficiency Improvements [10 pts]

You are given the circuit shown below with $\tau_{CL} = 16ns$, and $\tau_{setup} = \tau_{clk-Q} = 1ns$.



On average, at some V_{dd} the energy for one data item passed through the combinational logic block is 1 Joule. The registers each consume 0.1 Joule on average for each new data word stored.

Your application for this circuit requires results be computed at a rate of 50MHz (one result per cycle). Also, for this application, the latency from input to output is not important.

Assume the combinational logic block can be split evenly (in terms of both delay and energy) into multiple blocks.

Devise a scheme that would improve the switching energy efficiency while meeting the application requirements. Compare the switching energy per result of the original circuit and your new one.

Assume that a $1/n$ reduction in clock frequency can accommodate a $1/n$ reduction in V_{dd}.

Problem 9: Race to Halt [4 pts]

An effective scheme for improving energy efficiency when static power consumption is a significant component of total power consumption is a technique call "race to halt". The basic idea is to run the hardware at maximum speed to quickly compute the necessary set of computations, then turn off the power, thus preventing leakage.

Suppose you have a CPU that runs your application with an average power consumption of 8 Watts, where 50% of the power is dynamic and 50% is static. Your application requires an average of 100 Million operations per second but the CPU is capable of 400M ops/sec. Assume that no other program also running on the CPU.

You would like to determine the most effective way to run your application to preserve the battery life. You have the ability to control the supply voltage (V_{dd}), the clock frequency (f), and if needed can put the CPU into a sleep mode where static power is essentially zero. You consider two schemes.

1. Reduce supply voltage and clock frequency.
2. Keep V_{dd} and f unchanged, but use the sleep mode.

Which approach will be better at conserving your battery charge? Show your work and justify your answer.

Problem 10: Memory [9 pts]

(a) Suppose you want to design a 1-Byte wide memory block with a capacity of 2K Bytes of storage (remember 1K = 1024). We would like to have the core of the block square (equal number of rows and columns). How many total address bits are needed for this memory? How many address bits are used by the row-decoder? How many address bits are used by the column-decoder.

(b) Now you want to design the row decoder using the predecoder technique presented in lecture. Try two difference approaches. The first approach can use only gates with no more than 2-inputs. The second scheme can use some gates with 4-inputs.

Map out each scheme and describe the design of each of these decoders.

Problem 11: SRAM Waveform [5 pts]

Imagine you are designing the controller for an SRAM block. Draw the waveforms indicating the sequence of operations for a *READ*. The sequence begins when the address is registered on the positive-edge of the clock. Draw the approximate waveforms for all the signals.

1. CLK
2. ADDR (address)
3. WL (word line)
4. PC (precharge)
5. SE (sense amp enable)
6. D (data out)

Problem 12: SRAM vs. DRAM [4 pts]

For each of the following attributes, compare SRAM to DRAM and explain your reasoning.

(a) density (bits stored per unit area)

(b) access time

(c) noise resilience

(d) cost per bit

Problem 13: DRAM [4 pts]

1-transistor DRAM designs usually include a "row buffer"—a register on the periphery that is used to register an entire row. Explain how this register could be used and why it's a good idea.