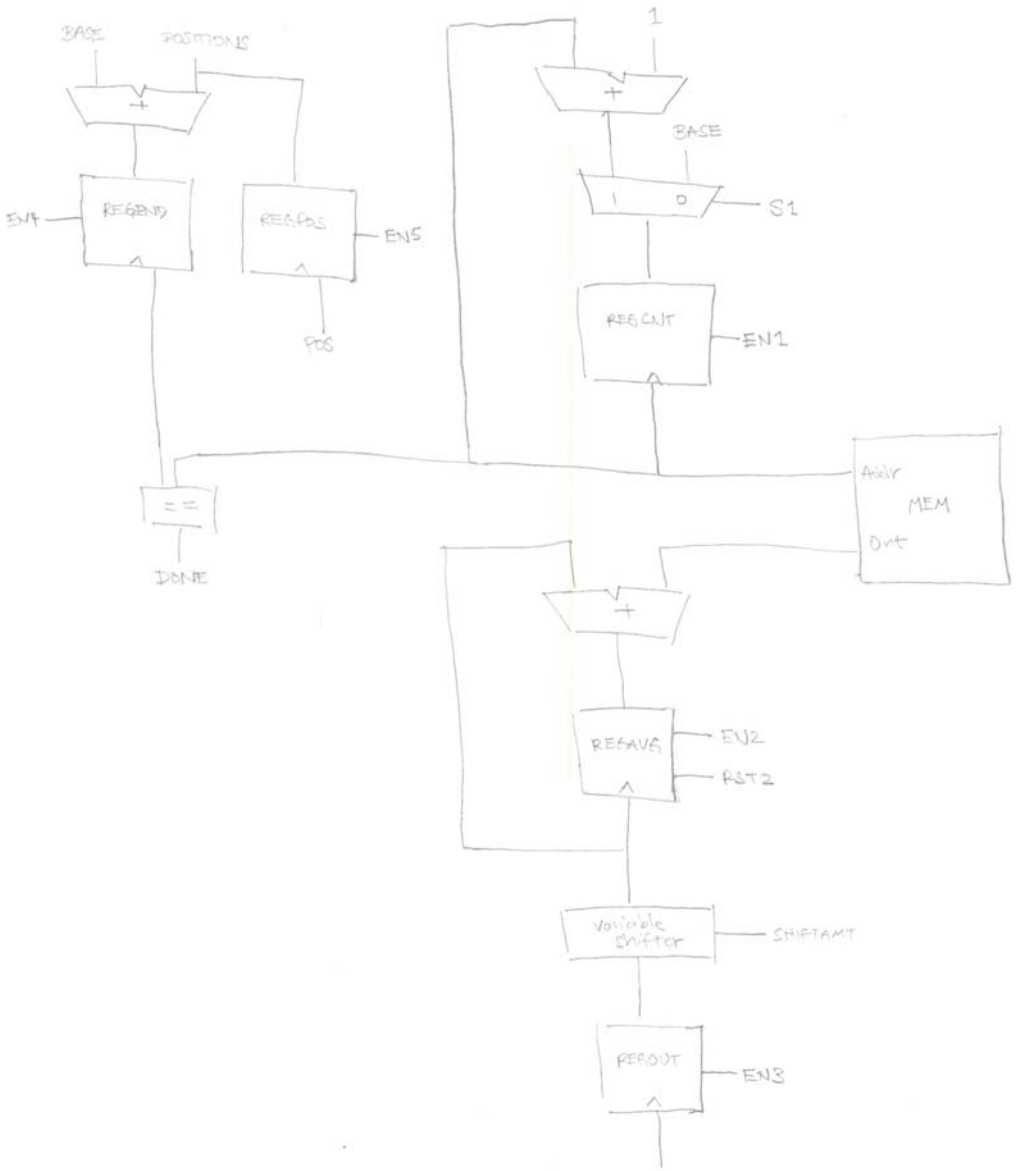
UNIVERSITY OF CALIFORNIA AT BERKELEY
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MIDTERM 2 REVIEW EXAMPLE

Given the following datapath and control signals, design a system that performs averaging of values read from the memory module. When the En input is pulsed, start from memory address BASE and average a total of POSITIONS consecutive memory locations. For example, if BASE is 3 and POSITIONS is 4, then the output will be (MEM[3] + MEM[4] + MEM[5] + MEM[6])/4. BASE and POSITIONS are only valid when En is high. Assume that POSITIONS is always a power of 2 and that data appears at the memory output one cycle after the address changes. Do not update the output register until the final average is available.



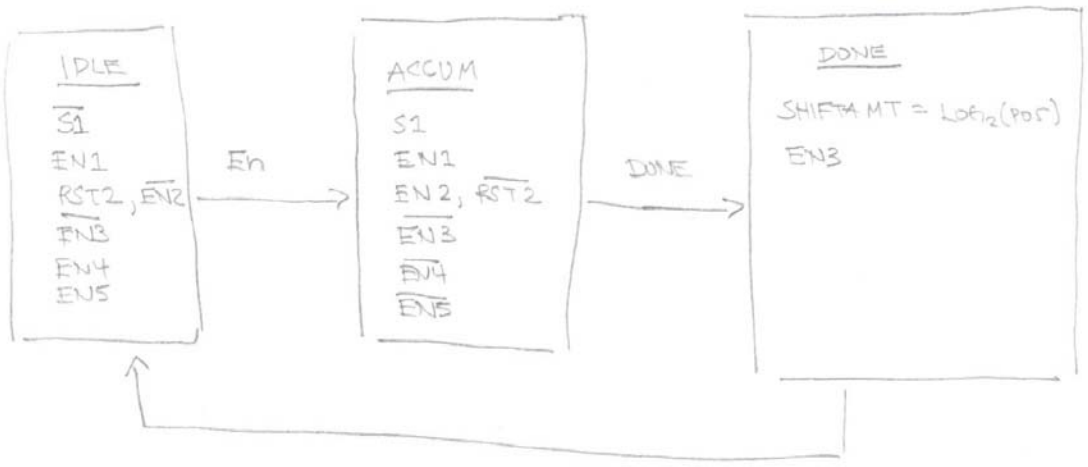
a) Write RTL that describes the system.

RTL is pseudo-code that describes the movement of data from register to register. Notice that control signals are not part of the RTL below. Each line executes in one clock cycle. Given that it's pseudo-code, don't get too caught up on how loops are written or why there's a "log" in the code. After the first line, I stopped writing uninteresting cases where registers retain their old value.

1. REGCNT ← BASE, REGAVG ← 0, REGOUT ← REGOUT, REGEND ← BASE + POSITIONS, REGPOS ← POSITIONS
2. WHILE (~DONE)
   REGAVG ← REGAVG + MEM[REGCNT], REGCNT ← REGCNT + 1
3. REGOUT ← REGAVG >> $\log_2$(POS)

b) Design a state machine that implements the system.

In *idle*, the system is continually initialized (line 1 of RTL above) to keep the controller simple.



c) Where is the bug in the timing of this state machine?

The DONE signal is asserted one cycle too late. Look at the case where BASE = 0 and POSITIONS = 2. On enable, the state machine transitions to *accum*. After 2 cycles there, REGAVG contains MEM[0] + MEM[1] and DONE is asserted since REGCNT is 2. After the transition to *done*, REGAVG contains MEM[0] + MEM[1] + MEM[2]; this is incorrect. Correct this error by decrementing the value saved into REGEND by 1, but be careful not to decrement the value saved into REGPOS, since it is used to compute shifting.

d) If you are only allowed to use 2 adders in the datapath, how do you implement the system.

Notice that in any given cycle, a maximum of 2 additions are made. This means that with proper muxing, the adder that computes either REGCNT or REGAVG can be used to compute the value stored into REGEND. This makes the datapath more complex, but eliminates the need for 3 adders. It is often necessary to make such a tradeoff between controller/datapath complexity and hardware resource use. Fortunately for you, you don't have to worry about this since you have an FPGA!