Columbia University
Department of Electrical Engineering
Midterm Quiz
EE 4340. Spring, 1999
March 11, 1999
Time: 75 minutes
Total: 100 points

NAME ______________________________

Read carefully before you proceed:

1. This exam is *open book, open notes.*
2. Put all answers in the blue books. Please turn in your exam sheets with your blue book as well.
3. Partial credit will be given. If you don't have time to work a problem completely, explain how you would solve it and try to set up the problem. No credit will be given for answers without explanation.

1. (50 points) In this problem, you are to write a program in PDP/8 Assembly Language (PAL) to multiply two six-bit unsigned binary numbers, returning a 12-bit product. *You may use only the instructions supported by the PDP/8I that we have discussed in class.* A shift-and-add multiply algorithm you should implement to accomplish this is shown by example below for the multiplication of "101011" by "011101" (another example is shown on page 234 of Hayes):

```
      101011   multiplicand
      011101   multiplier
     -------
      000000   partial product
      101011
     -------
      101011   partial product
     000000
     -------
     0101011   partial product
    101011
    --------
    11010111   partial product
   101011
  ----------
  1000101111   partial product
  101011
 -----------
 10011011111   partial product
 000000
------------
010011011111   product
```

You may assume that the multiplicand is stored in memory location 4000, the multiplier is stored in memory location 4001, and the product is stored in memory location 4002. When stored in memory, you may assume that the multiplier and multiplicand are padded on the left with zeros.

2. (50 points) In this problem, you design the controls for a simple first-in, first-out (FIFO) buffer. FIFOs are commonly used in communications systems to buffer data to handle cases in which the transmitter sends burst of data faster than the receiver can handle them. Our FIFO is built from a register file with four 8-bit words. Two pointers, `rdptr` and `wrptr`, are used to address the location in the register file that could be read or written in the current cycle. In addition, there are two control indications, `empty` and `full`. `empty` indicates that all the the bytes of the FIFO that have been written has also been read out. `full` indicates that all four bytes of the FIFO have been written but that none of these bytes have been written out. When the FIFO is full, no further write requests are honored.

The operation of the FIFO is illustrated in Figure 1.

1. The FIFO has no data in it. Both `wrptr` and `rdptr` are set to the same address and `empty` is 1.

2. A write operation occurred. The value `10` was written into the register file and `wrptr` was incremented. The `empty` flag is now zero since the FIFO contains data.

3. Another write operation has occurred, storing `15` into the FIFO.

4. A read operation occurred. The value `10` is read out of the register file and `rdptr` is incremented. The value stored in the FIFO is not removed after the read operation.

5. The value of `20` is written into the FIFO.

6. The value of `25` is written into the FIFO. `wrptr` is rolled over so that it now point to the first register file address again.

7. The value of `30` overwrites the value of `10`. This makes the `rdptr` and `wrptr` pointers equal. All four locations of the register file are loaded with data that has not yet been read out. The `full` flag is set to 1.

A datapath implementation for the FIFO is in Figure 2. The control signals `rdptr_inc` and `wrptr_inc` determine whether the wrap-around counters driving `wrptr` and `rdptr` increment or not. In addition, we compare the values of `wrptr` and `rdptr` and send this as the signal `compare` to the controls.

`read` and `write` are synchronous signals which are '1' on cycles in which a read or a write of the FIFO, respectively, should be performed. `read` and `write` cannot both be '1' in the same cycle. When both `read` and `write` are '0', there is no change in the state of the register file, `wrptr`, or `rdptr`.

Draw an ASM chart for the controls for this FIFO datapath. The trickiest part of the controls is recognizing the condition that determines the outputs `empty` and `full`. In particular, note that if the read and write pointers are equal and the last operation was a write, then the FIFO is full. If the read and write pointers are equal and the last operation was a read, then the FIFO is empty.

Write a complete VHDL description of your FIFO, both controls and datapath. It is probably easiest to write this as a single entity and architecture for the entire FIFO.

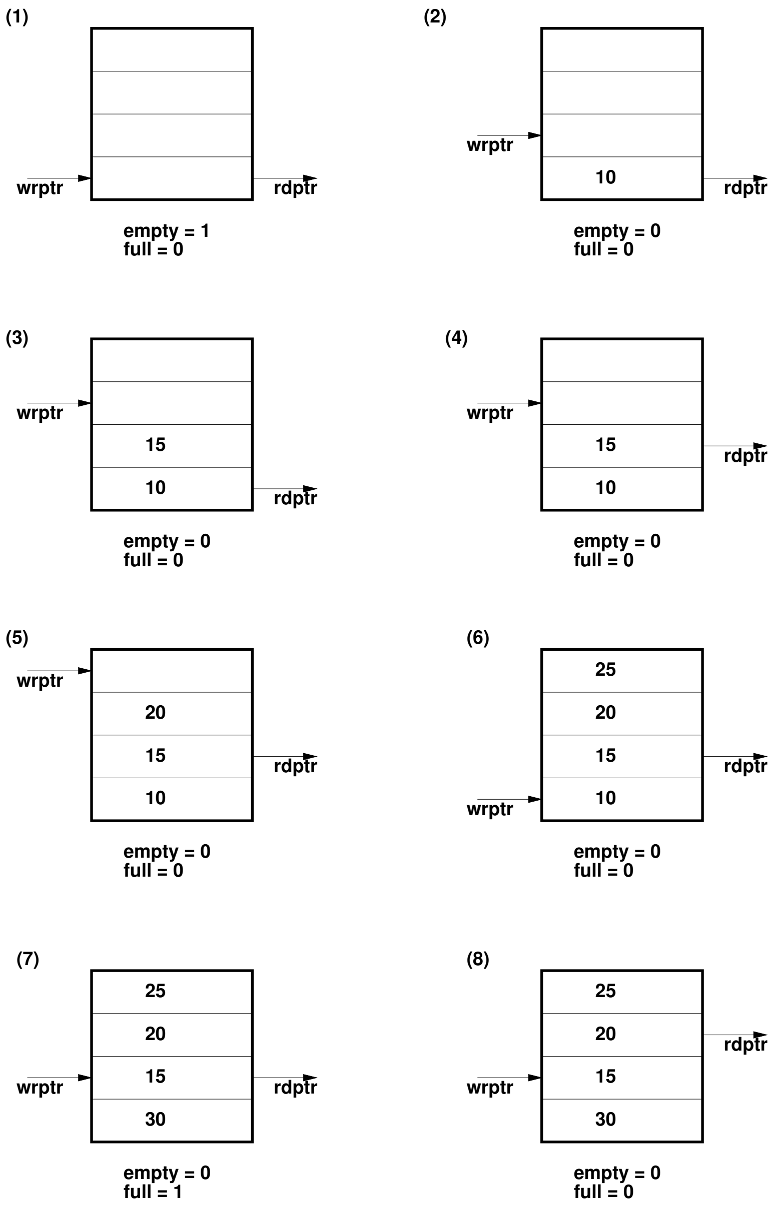

Figure 1: FIFO operation

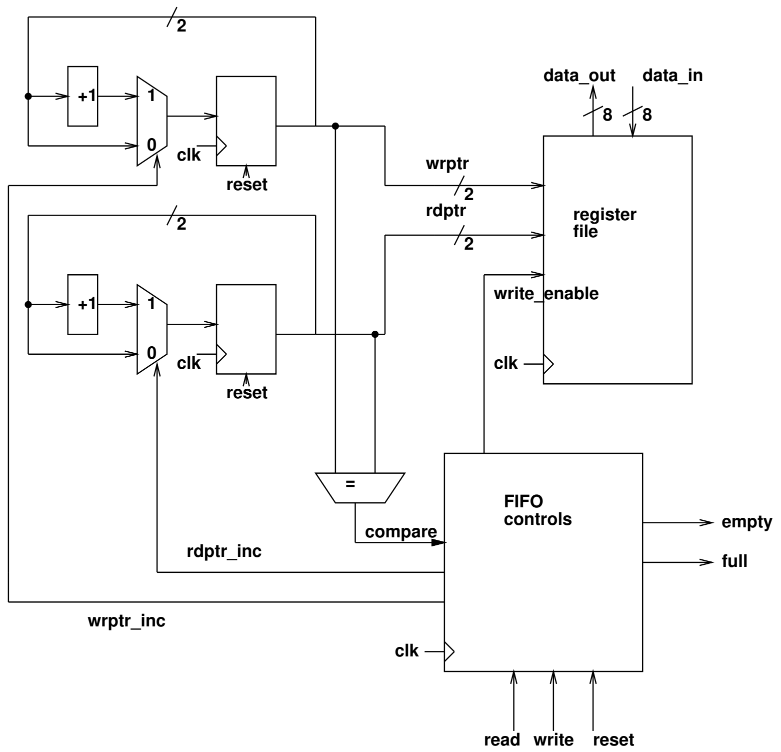

Figure 2: FIFO datapath.