Wawrzynek & Weaver
Sp 2018

EECS 151/251A
Digital Design

MT 2

Print your name: ______________________________, ______________________________
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct will be reported to the Center for Student Conduct, and may result in partial or complete loss of credit.

Sign your name: ____

You may consult two sheets of paper (double-sided) of notes. You may not consult other notes, textbooks, etc. Calculators, computers, and other electronic devices are not permitted.

Please write your answers in the spaces provided in the test. We will not grade anything on the back of an exam page.

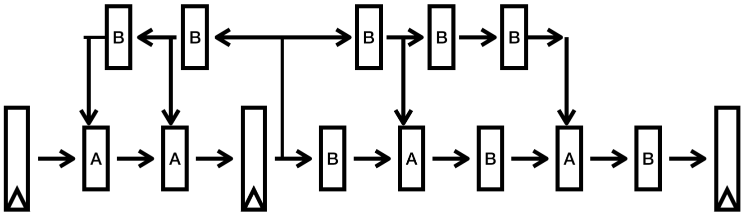
You have 3 hours (minus 10 minutes) although we don't expect you to take more than half that time. There are 6 questions, of varying credit (95 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

Do not turn this page until your instructor tells you to do so.

Question:	1	2	3	4	5	6	Total
Points:	23	18	20	8	8	18	95
Score:							

Problem 1 *Pipelining and C-slowing* (23 points)

Consider the following simple circuit. Elements of type A have a 10ns delay, elements of type B have a 5ns delay, and all registers have a 1ns setup time and 1ns CLK to Q time. The registers on the input and output are mandatory and can't be moved:



(a) (3 points) What is the minimum clock period for this circuit?

(b) (8 points) You want to maximize the throughput of this circuit through adding two internal pipeline stages but without introducing any other control logic. Draw the resulting design:

(c) (2 points) What is the minimum clock period for your new design?

(d) (8 points) You now want to pipeline it in a C-slow manner where C=2, again without adding any other control logic. You can also add as many other pipeline stages as you desire to the output path. Draw the resulting design.

(e) (2 points) What is the minimum clock period for your new design?

Problem 2 *Ben Bitdiddle Builds a FIFO* (18 points)

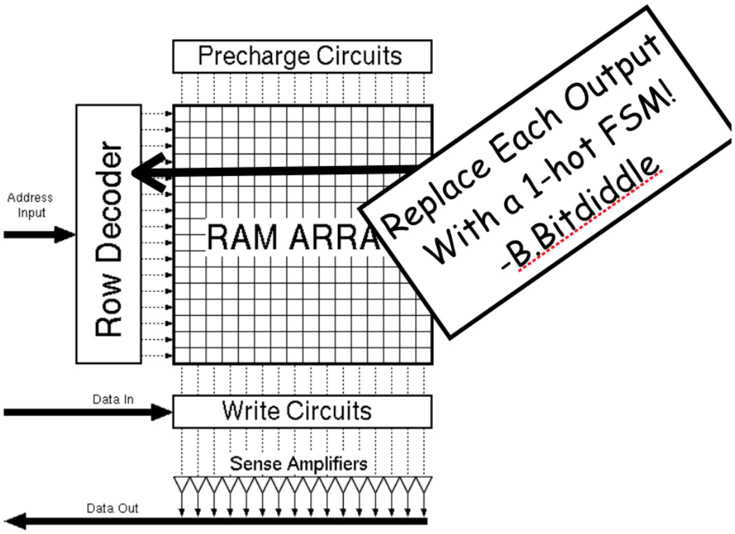
Normally when we build a FIFO buffer we have two registers which index into a dual-ported SRAM with one address for reading and one for writing. To write we write into the current address and increment the current write address by 1, and do the same for the read address. The FIFO is considered full if the write address +1 equals the read address, and empty of the write and read addresses are equal.

Thus we can read as long as the current write address != the current read address, and we can write as long as the next address after the current write address is not the current read address. The normal way to build this is to have a counter drive the address input to the decoder for both the reading and the writing address.

Ben Bitdiddle observes that since the output of the address registers are pretty much only used to fed into row decoder logic for the array. So he thinks: "Why should I do that? Why not just build a 1-hot state machine to run the wordlines directly?".

So he does. His SRAM array's column width is the same as the output width, making the problem somewhat simpler (so address selection only occurs with the decoder, not with an additional mux on the output).

So for each wordline (one per row for reading and one per row for writing) he has a small state machine consisting of a flip flop and some associated logic. The wordline itself is directly driven by a flip-flop.



(a) (3 points) For an N-entry FIFO, the cost of Ben Bittdiddle's state machine is (in big-O notation):

(b) (3 points) For a classical N-entry FIFO, the cost of a column decoder is (in big-O notation):

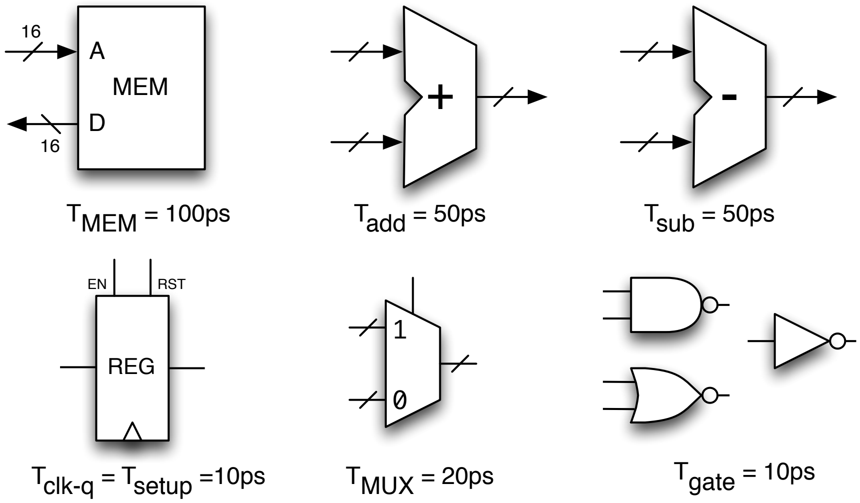
(c) (6 points) In a conventional approach a wordline is only asserted when there is an explicit request to read or write. Is Ben Bitdiddle's approach superior in terms of power consumption? Why or why not?

(d) (6 points) Do you think Ben Bitdiddle's approach is a good idea over the conventional approach in terms of area? Why or why not?

Problem 3 ***Accelerator Design*** **(20 points)**

Your job is to design an accelerator for the following function. Assume that we have a 16-bit wide memory block that stores an array of 2's complement integers. The address of the first element in the array is 0, the second element is at address 1, etc. The function of the accelerator is to find the maximum value in the array and to output that value on OUT[15:0]. The accelerator inputs (besides the clock) are RST, used to reset the accelerator, and an input TOP[15:0] which is used to input the address of the last valid element of the array.

Use instances of the following components. For simplicity use the larger blocks instead of logic gates when possible.



The memory block has asynchronous read. The register block includes a clock enable input (EN) and a reset input (RST).

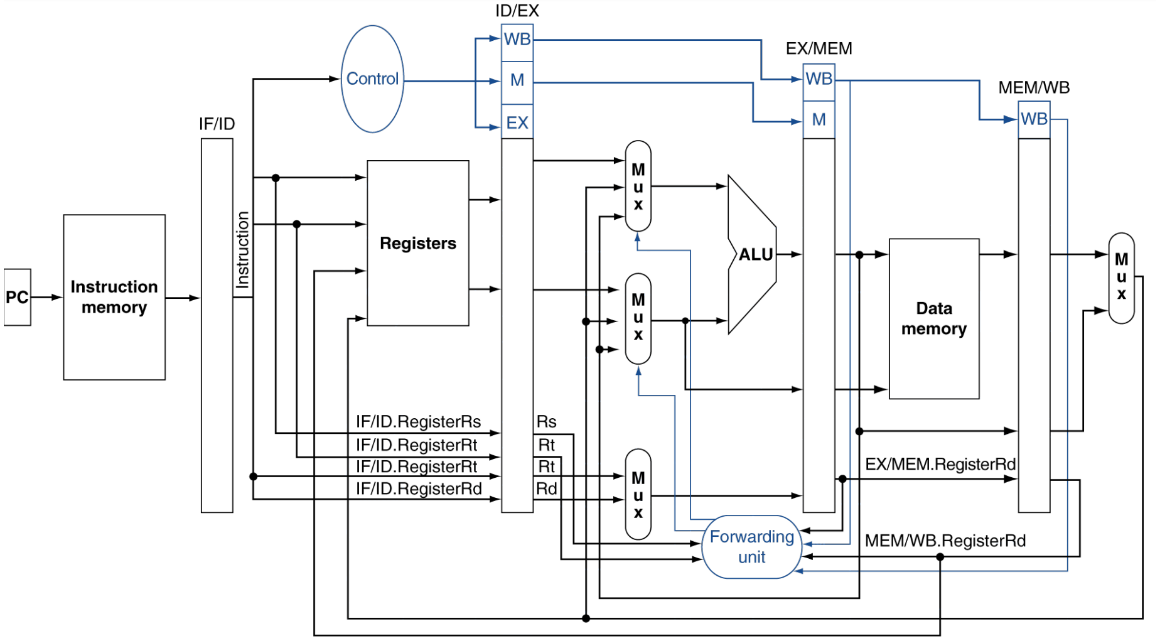
(a) (15 points) Schetch the circuit design of an accelerator. Don't worry too much about exact cost, but try to minimize the total number of components. (Work out your design on scratch paper first, then neatly copy it onto the next page.)

How long does it take your accelerator to process an array with N elements?

(b) (5 points) Make a simple modification to your design to maximize its performance. Redraw only those parts of the design that are modified. With this modified design how long does it take your accelerator to process an array with N elements?

Problem 4 *Ben Bitdiddle Forwards to Nowhere* (8 points)

Ben Bitdiddle is in charge of modifying a typical RISC-V implemented as a 5 stage pipeline with forwarding. He has this, umm, genius idea. He observes that a common motif in programming is simply copying lots of memory, often in a loop.

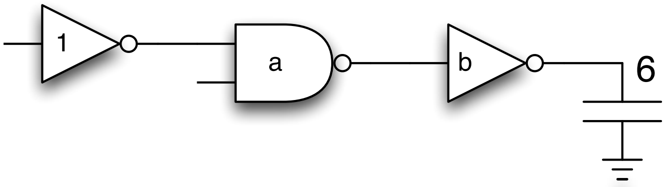


(a) (4 points) What stage in the above pipeline will Ben have to modify to add a forwarding path that will eliminate the stall that occurs when memory is immediately read and then written into another location?

(b) (4 points) Explain to Ben why, using simple words and one to two sentences, in practice, his new forwarding path is effectively useless on real programs:

Problem 5 ***Logical Effort*** **(8 points)**

In the path shown by the circuit below, a and b represent the input capacitances normalized to the unit inverter. Using the *method of logic effort*, find the values for a and b that minimize the path delay. Show your work.



Problem 6 *Energy Efficiency* (18 points)

Assume we have a circuit that is used to compute the following operation on streaming input:

$$y_i = ax_i + bx_{i-1} + cx_{i-2}$$

The circuit operates at 2 volts and has a throughput of 1 Gsamples/sec while consuming 10mWatts. We observe that 25% of the power consumption is due to leakage (static) and the remaining 75% is in switching (dynamic).

In our implementation technology, $\tau_{multiply} = 2\tau_{add}$. *For simplicity, we will ignore the delays and power associated with registers and clocking.* Also, assume that there is a linear relationship between supply voltage and circuit delay.

(a) (2 points) What is the energy efficiency in joules/sample of this circuit?

Describe two techniques that we could use to improve the energy efficiency of the circuit. For each:

- Show how to modify the circuit to achieve the optimization,
- Derive the approximate improvement in energy efficiency,
- Describe the approximate change in circuit cost.

(b) (8 points) Improvement #1:

(c) (8 points) Improvement #2: