Practice homework, computer engineering part, Engineering 100, section 250

Questions with a "$" after them are intended to be challenging. Those questions with "$$" after them are probably unreasonable. Later I will post answers to *selected* questions.

(updated 11/29@9:20am)

Number representation

1. Provide the 8-bit two-complement representation for the following values. If the value can't be represented write "no such representation" instead.
    a. 1
    b. 0
    c. -1
    d. 14
    e. -14
    f. 70
    g. -70
    h. 99
    i. 127
    j. -127
    k. 128
    l. -128
    m. 255

2. What is the range of representation of the following representation schemes? All answers should be in decimal (e.g. "4 to -4")
    a. 12-bit two's complement number
    b. 8-bit unsigned number
    c. 3-bit two's complement number
    d. 32-bit two's complement number

3. Shifting
    a. If the unsigned number 12 is shifted to the left by 3, what is the value of the number after the shift?
    b. In general shifting a number to the left by X is the same as multiplying a number by what?
    c. In general shifting a number to the right by X is the same as _______ a number by _____?

4. If you had the value 2 in a memory location in the e100 and then you drove it out to HEX3-HEX0, what would be displayed on the HEX displays?
    a. What if the value were 17?
    b. -2?
    c. -15?
    d. 1024?

Digital Logic

In this section "+" means OR, "*" means AND, "!" means "NOT", and "⊕" is XOR

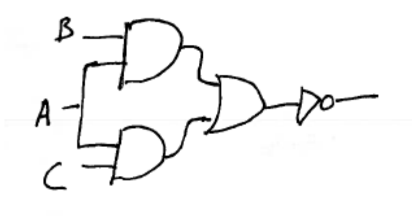
1. Write the truth table for
    a. (A+B)*C
    b. (!A*C)+!B
    c. A⊕B

2. Write the logical statement which corresponds to each of the following truth tables (a-g)

X	Y	Z	a	b	c	d	e	f	g
0	0	0	0	0	0	1	1	0	1
0	0	1	0	1	0	1	0	1	1
0	1	0	0	1	1	1	0	1	1
0	1	1	0	1	0	1	0	1	0
1	0	0	0	1	0	0	0	1	0
1	0	1	0	1	0	1	0	1	0
1	1	0	0	1	0	1	1	0	1
1	1	1	1	1	0	1	0	1	0

(For example, the answer to "a" is X*Y*Z)
e, f and g are all pretty tricky ($)

3. Draw logic gates which correspond to the following logic expressions
   a. (A+B)*C   b. (!A*C)+!B   c. $A \oplus B$
4. Write a truth table that corresponds to the following circuit.



<u>E100 programming</u>

1. Write an e100 assembly *<u>function</u>* named SI which takes two arguments, SI_a and SI_b and returns a single value SI_rv. The function is to computer SI_a times SI_b ***<u>by using repeated additions</u>*** rather than using the multiply instruction. The caller uses SI_ra to store the return address. Your function should include the declaration of all memory locations used (including those mentioned above).

2. Write an e100 assembly *<u>program</u>* which ***continuously*** reads a 16-bit number from the dip switches, multiplies the value by 4 and then displays the results on the red LEDs.
   a. Of LED_RED[15:0], what LED(s) is/are never lit? Why? (HINT: This is more of a number representation problem.)

3. Write an e100 assembly *<u>program</u>* which initially displays "0000" on the HEX digits HEX3-HEX0 and then increments that value once per second.

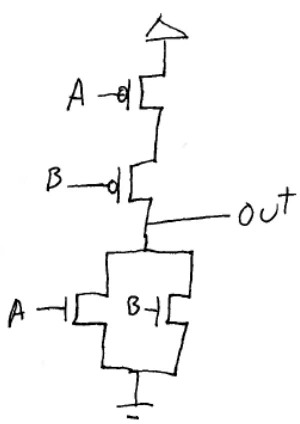
4. Say we have a new device added to the e100. It uses our standard protocol and has the following ports:

Port number	Port type	Definition	Use
150	in	bit 0: Bob_valid	
151	out	bit 0: Bob_ack	Bobbity Bob Bob
152	out	bits 15-0: Bob_data	

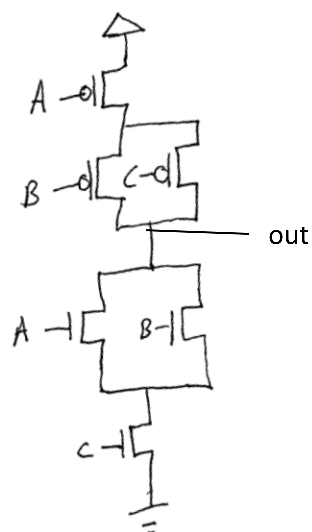
   a. Is this an input or output device? How can you tell?
   b. Write an e100 assembly *function* called BobD which takes a single input, BobD_value, and properly passes that data off to the Bob device.

Transistors

1. Write the truth table for the following devices. Each entry of the truth table should be either "0", "1", "HiZ" or "Smoke".



a.

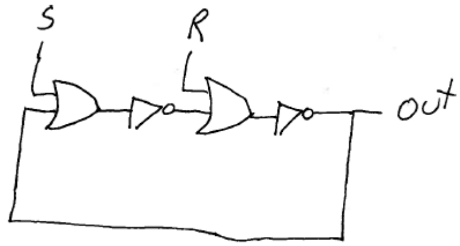


b.

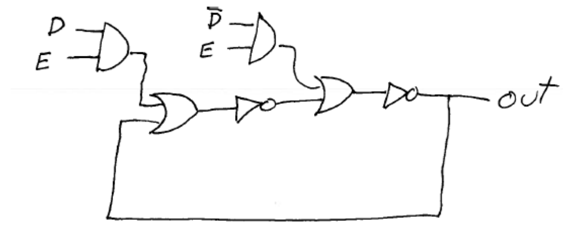
2. Draw a "NAND" gate using transistors.
3. Draw an "AND" gate using transistors.
4. Draw a 3-input "OR" gate using transistors.
   a. Do it again and use 8 or fewer transistors if you didn't the first time.
5. Draw an XOR gate using transistors. ($)
6. Draw a tri-state device using transistors. ($, we did this in class but try to figure it out without looking).

Latches and Flip-flops

1. Write a truth table for the following circuit. Each entry should be either “0”, “1”,”holds value”.

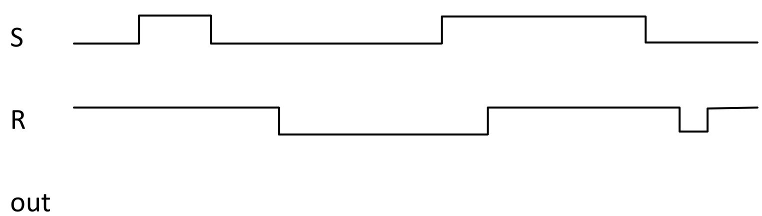


a.

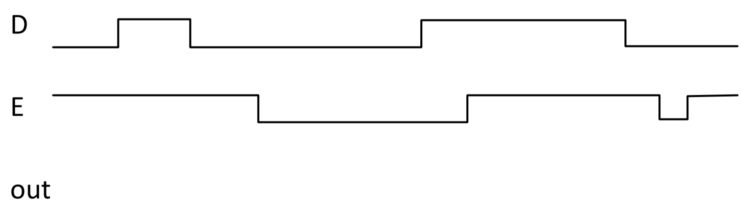


b.

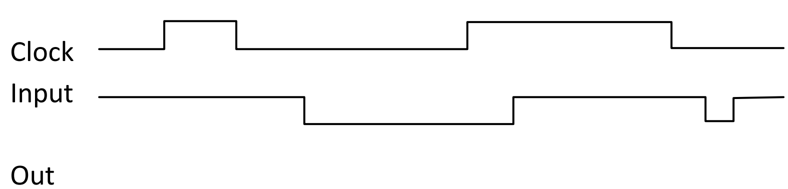
2. Consider a SR-latch (the thing in problem 1a). Finish the following timing diagram.



3. Now do the same for the D-latch (the thing in problem 1b).



4. Now do the same thing for a 1-bit register with no write enable.



5. Design a one-bit register with a write enable using standard gates (AND, OR, NOT). $$