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 00000001
   b. 0 00000000
   c. -1 11111111
   d. 14 00001110
   e. -14 11110010
   f. 70 01000110
   g. -70 1011 1010
   h. 99 01100011
   i. 127 01111111
   j. -127 10000001
   k. 128 No such rep.
   l. -128 10000000
   m. 255 no such rep.

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 2047 to -2048
   b. 8-bit unsigned number 0 to 255
   c. 3-bit two's complement number -4 to 3
   d. 32-bit two's complement number -4294967296 to 4294967295

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? 96
   b. In general shifting a number to the left by X is the same as multiplying a number by what? 2^X
   c. In general shifting a number to the right by X is the same as dividing a number by 2^X?

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? 0002
   a. What if the value were 17? 0011
   b. -2? FFFE
   c. -15? FFF0
   d. 1024? 0400

Digital Logic

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

1. Write the truth table for
   a. (A+B)*C

A	B	C	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

   b. (!A*C)+!B

A	C	B	
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

   c. A⊕B

A	B	
0	0	0
0	1	1
1	0	1
1	1	0

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 ($)

a) $X \cdot Y \cdot Z$

c) $X \cdot \overline{Y} \cdot Z$

d) $\overline{X} + Y + Z$

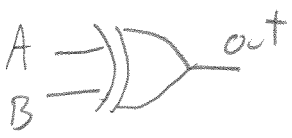
f) $(X + Y + Z) \cdot (\overline{X} + \overline{Y} + Z)$

e) $(\overline{X} \cdot \overline{Y} \cdot \overline{Z}) + (X \cdot Y \cdot \overline{Z})$

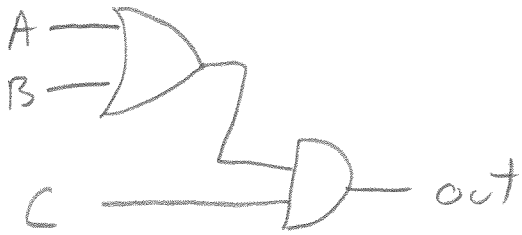
g) $(\overline{X} \cdot \overline{Y} \cdot \overline{Z}) + (\overline{X} \cdot \overline{Y} \cdot Z) + (\overline{X} Y Z) + (X \cdot Y \cdot \overline{Z})$

3. Draw logic gates which correspond to the following logic expressions
   a. (A+B)*C
   b. (!A*C)+!B
   c. A⊕B

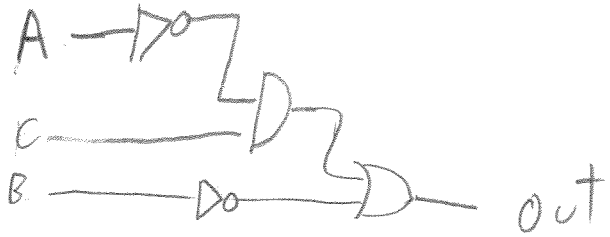
c)



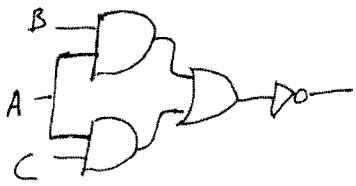
a)



b)



4. Write a truth table that corresponds to the following circuit.



$(A \cdot B + A \cdot C)$

A	B	C	A·B	A·C	$\overline{(A \cdot B)} + \overline{(A \cdot C)}$
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

E100 programming

1. Write an e100 assembly *function* 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 ***by using repeated additions*** 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).

```
SI      add     SI_total    SI_a        SI_a
        sub     SI_b        SI_b        SI_one
        bne     SI          SI_b        SI_zero
        ret     SI_ra
```

```
SI_total    .data   0
SI_a        .data   0
SI_b        .data   0
SI_one      .data   1
SI_zero     .data   0
SI_ra       .data   0
```

2. Write an e100 assembly *program* 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.)

```
start   in      0           switch
        mult    switch      switch      four
        out     1           switch
        be      start       one         one

one     .data   1
four    .data   4
switch  .data   0
```

LEDs 1, 0 never light up, you always multiply an input by 4, so you will never get 1, 2, or 3 as a result.

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

See next sheet

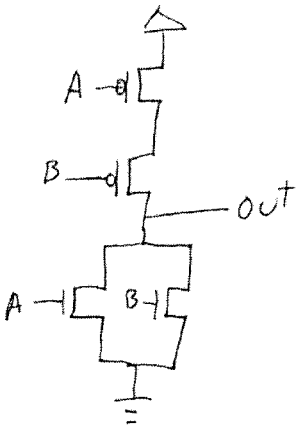
4. Say we have a new device added to the e100. It uses our standard protocol and has the following ports:

Port number	Port type	Definition	Use
150	in	bit 0: Bob_valid	Bobbity Bob Bob
151	out	bit 0: Bob_ack	
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.

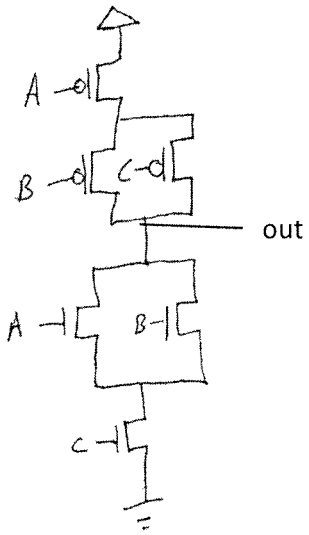
Transistors

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



a.

A	B	out
0	0	1
0	1	0
1	0	0
1	1	0



b.

A	B	C	out
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	Hi-Z
1	0	1	0
1	1	0	Hi-Z
1	1	1	0

3)

```
start   out   3    count
        in    5    start
wait    in    5    end
        sub   time end  start
        blt   wait time c28
        add   count count c1
        be    start c1   c1

c1      .data 1
c28     .data 28
time    .data 0
start   .data 0
end     .data 0
count   .data 0
```

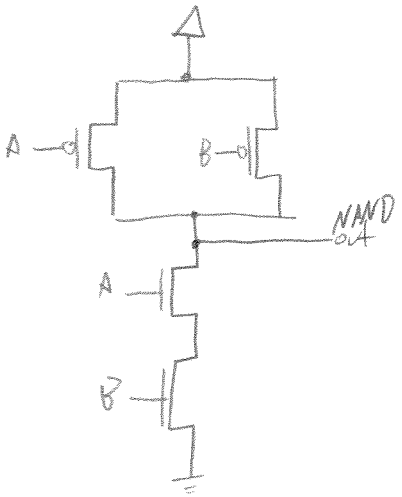
See next sheet

2. Draw a "NAND" gate using transistors.

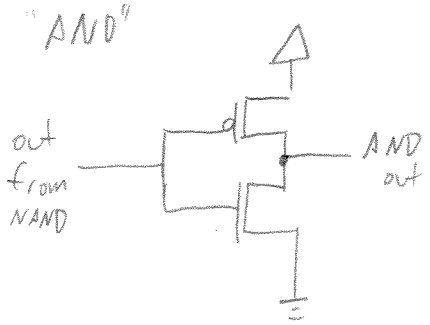
3. Draw an "AND" gate using transistors.
4. Draw a 3-input "OR" gate using transistors.
    a. Do it again and use 8 or fewer transistors if you didn't the first time.
5. Draw an XOR gate using transistors. ($)

6. Draw a tri-state device using transistors. ($, we did this in class but try to figure it out without looking).

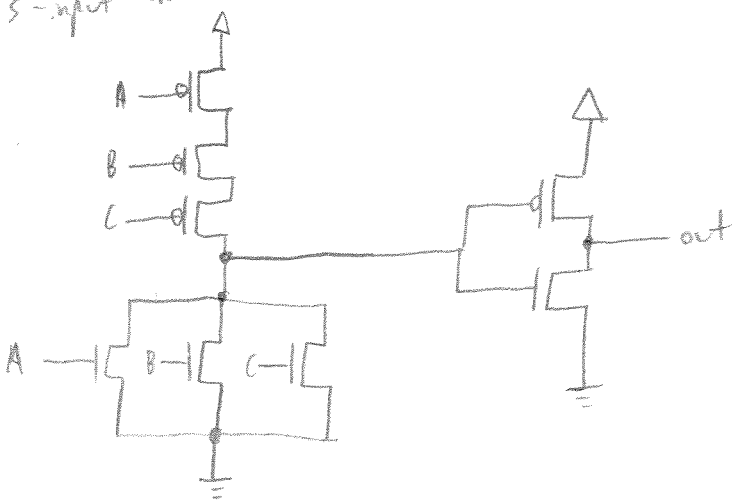
2) "NAND"



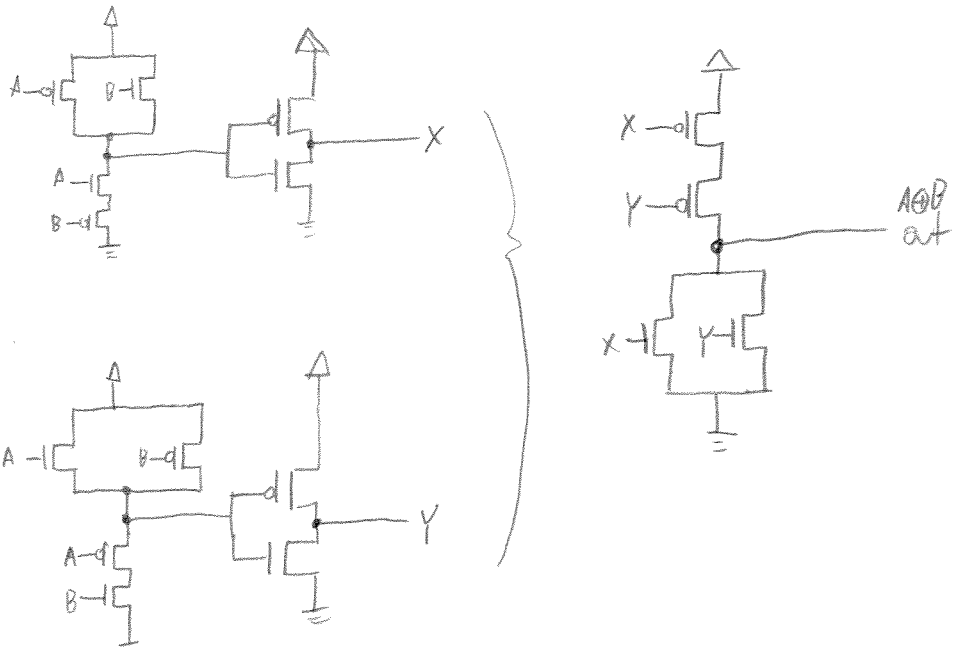
3) "AND"



4) 3-input "OR"

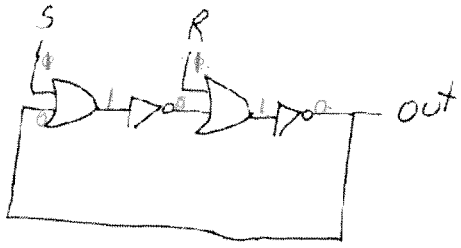


5) XOR ($A \cdot \overline{B} + \overline{A} \cdot B$)

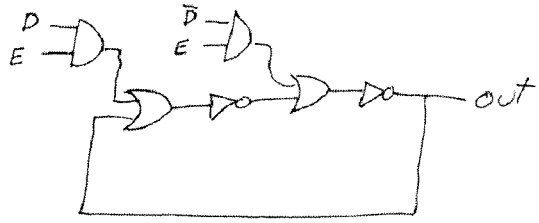


Latches and Flip-flops

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



a.

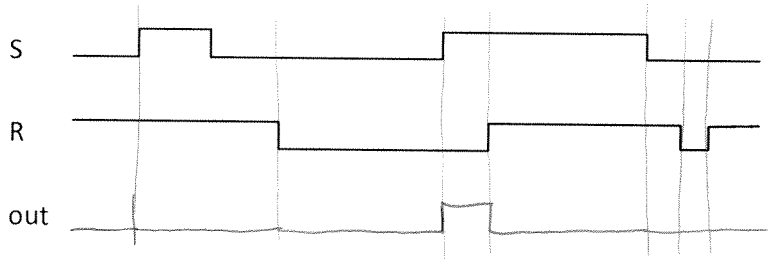


b.

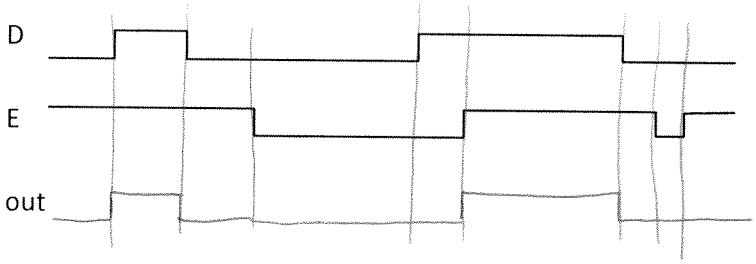
S	R	out
0	0	holds
0	1	0
1	0	1
1	1	0

D	E	out
0	0	holds
0	1	0
1	0	holds
1	1	1

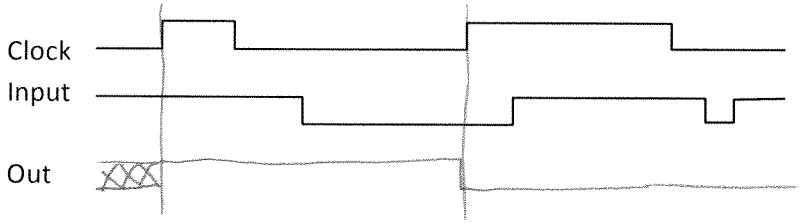
2. Consider a SR-latch (the thing in problem 1a). Finish the following timing diagram.



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). $$