Solutions - Midterm Exam

(February 19th @ 5:30 pm)

Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (15 PTS)

a) Complete the following table. The decimal numbers are unsigned: (6 pts.)

Decimal	BCD	Binary	Reflective Gray Code
89	10001001	1011001	1110101
119	000100011001	1110111	1001100
76	01110110	1001100	1101010
135	000100110101	10000111	11000100

b) Complete the following table. Use the fewest number of bits in each case: (7 pts.)

	REPRESENTATION		
Decimal	**Sign-and-magnitude**	**1's complement**	**2's complement**
-89	11011001	10100110	10100111
52	0110100	0110100	0110100
-51	1110011	1001100	1001101
-20	110100	101011	101100
-64	11000000	10111111	1000000

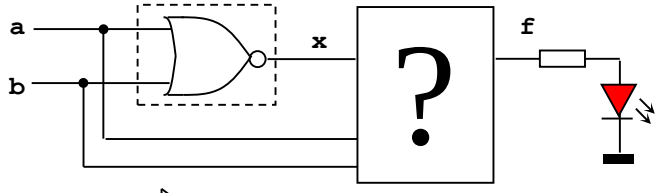
c) Convert the following decimal numbers to their 2's complement representations. (2 pts)

✓ -16.1875
+16.1875 = 010000.0011
⇒ 101111.1101

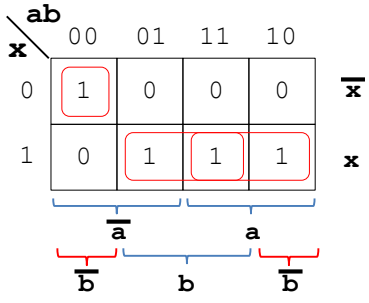
✓ 37.3125
+37.3125 = 0100101.0101

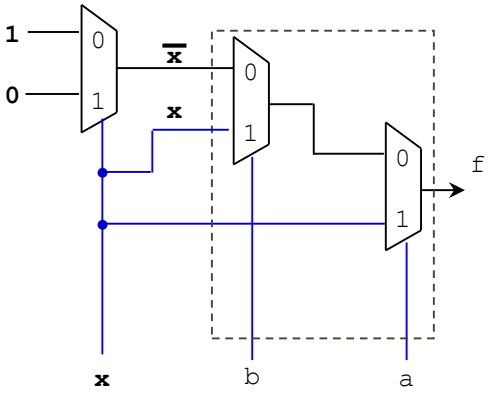
PROBLEM 2 (20 PTS)

a) Using only 2-to-1 MUXs, design a circuit that verifies the logical operation of a NOR gate. f = '1' (LED ON) if the NOR gate does NOT work properly.
Assumption: when the NOR gate is not working, it generates 1's instead of 0's and vice versa.
Tip: Minimize your function first. (15 pts)



a	b	x	f
0	0	0	1
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



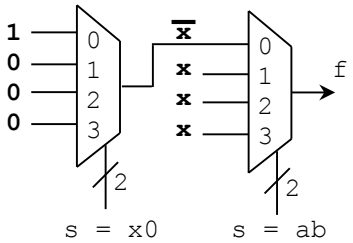


$f = \bar{a}\bar{b}\bar{x} + bx + ax$

$f(a,b,x) = \bar{a}f(0,b,x) + af(1,b,x) = \bar{a}(\bar{b}\bar{x} + bx) + a(bx + x) = \bar{a}(\bar{b}\bar{x} + bx) + a(x) = \bar{a}g(b,x) + ax$

$g(b,x) = \bar{b}g(0,x) + bg(1,x) = \bar{b}(\bar{x}) + b(x)$ Also: $\bar{x} = \bar{x}(1) + x(0)$

b) Implement the previous function f using only 4-to-1 MUXs. You might need to implement a NOT gate using a 4-to-1 MUX.



PROBLEM 3 (10 PTS)

- The figure below depicts the entire memory space of a microprocessor. Each memory address occupies one byte. 1KB = 2^{10} bytes, 1MB = 2^{20} bytes, 1GB = 2^{30} bytes
  - ✓ What is the size of the memory space? What is the address bus size of the microprocessor?

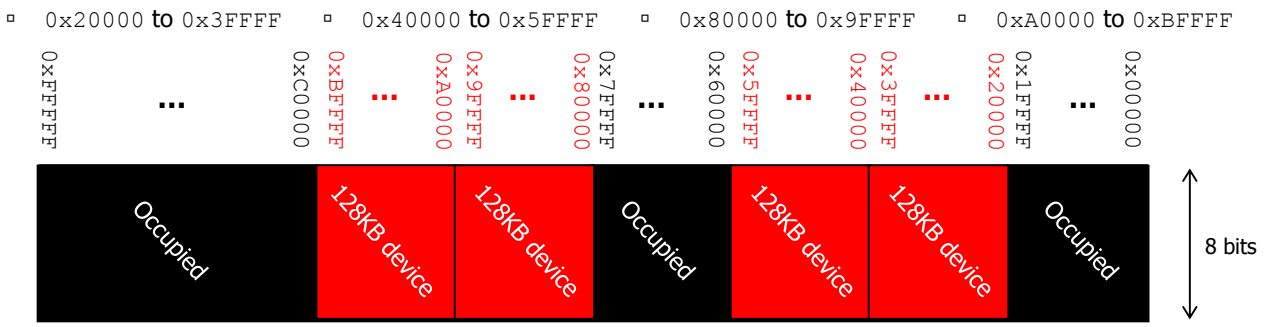
    Address space: `0x00000` to `0xFFFFF`. To represent all these addresses, we require 20 bits. So, the address bus size of the microprocessor is 20 bits. The size of the memory space is then 2^{20} = 1 MB.

  - ✓ If we have a memory chip of 128KB, how many bits do we require to address those 128KB of memory?

    128 KB memory device: 128KB = 2^{17} bytes. Thus, we require 17 bits to address the memory device.

  - ✓ We want to connect the 128KB memory chip to the microprocessor. The figure shows all the occupied portions of the memory space. Provide a list of all the possible ranges that the 128 KB memory chip can occupy.

    128KB of memory require 17 bits. The 17-bit address range would be from `0x00000` to `0x1FFFF`. Within the entire 20-bit memory space, there are four options to place those 128KB in the figure:



PROBLEM 4 (20 PTS)

a) Perform the following additions and subtractions of the following unsigned integers. Use the fewest number of bits n to represent both operators. Indicate every carry (or borrow) from c_0 to c_n (or b_0 to b_n). For the addition, determine whether there is an overflow. For the subtraction, determine whether we need to keep borrowing from a higher byte. (8 pts)

✓ 51 + 15

```
                 c6=1 c5=1 c4=1 c3=1 c2=1 c1=1 c0=0
51 = 0x33 =        1    1    0    0    1    1   +
15 = 0x0F =        0    0    1    1    1    1
Overflow! →   1    0    0    0    0    1    0
```

✓ 25 - 35

```
Borrow out! → b6=1 b5=0 b4=0 b3=1 b2=1 b1=0 b0=0
25 = 0x19 =        0    1    1    0    0    1   -
35 = 0x23 =        1    0    0    0    1    1
0x36 =             1    1    0    1    1    0
```

b) Perform the following operations, where numbers are represented in 2's complement. For each case, use the fewest number of bits to represent the summands and the result so that overflow is avoided. (8 pts)

✓ -89 + 128

n = 9 bits

```
                 c9=1 c8=1 c7=0 c6=0 c5=0 c4=0 c3=0 c2=0 c1=0 c0=0
c9⊕c8=0
No Overflow
-89 =              1    1    0    1    0    0    1    1    1   +
128 =              0    1    0    0    0    0    0    0    0
39 =               0    0    0    1    0    0    1    1    1
```

-89 + 128 = 39 ∈ $[-2^8, 2^8-1]$ → no overflow

✓ -61 -13

n = 7 bits

```
                 c7=1 c6=0 c5=0 c4=0 c3=0 c2=1 c1=1 c0=0
c7⊕c6=1
Overflow!
-61 =              1    0    0    0    0    1    1   +
-13 =              1    1    1    0    0    1    1
                   0    1    1    0    1    1    0
```

-61 -13 = -74 ∉ $[-2^6, 2^6-1]$ → overflow!

To avoid overflow: n = 8 bits (sign-extension)

```
                 c8=1 c7=1 c6=0 c5=0 c4=0 c3=0 c2=1 c1=1 c0=0
c9⊕c8=0
No Overflow
-61 =              1    1    0    0    0    0    1    1   +
-13 =              1    1    1    1    0    0    1    1
-74 =              1    0    1    1    0    1    1    0
```

-61 -13 = -74 ∈ $[-2^7, 2^7-1]$ → no overflow

c) Get the multiplication result of the following numbers that are represented in 2's complement arithmetic with 4 bits. (4 pts)
✓ 7 x -6.

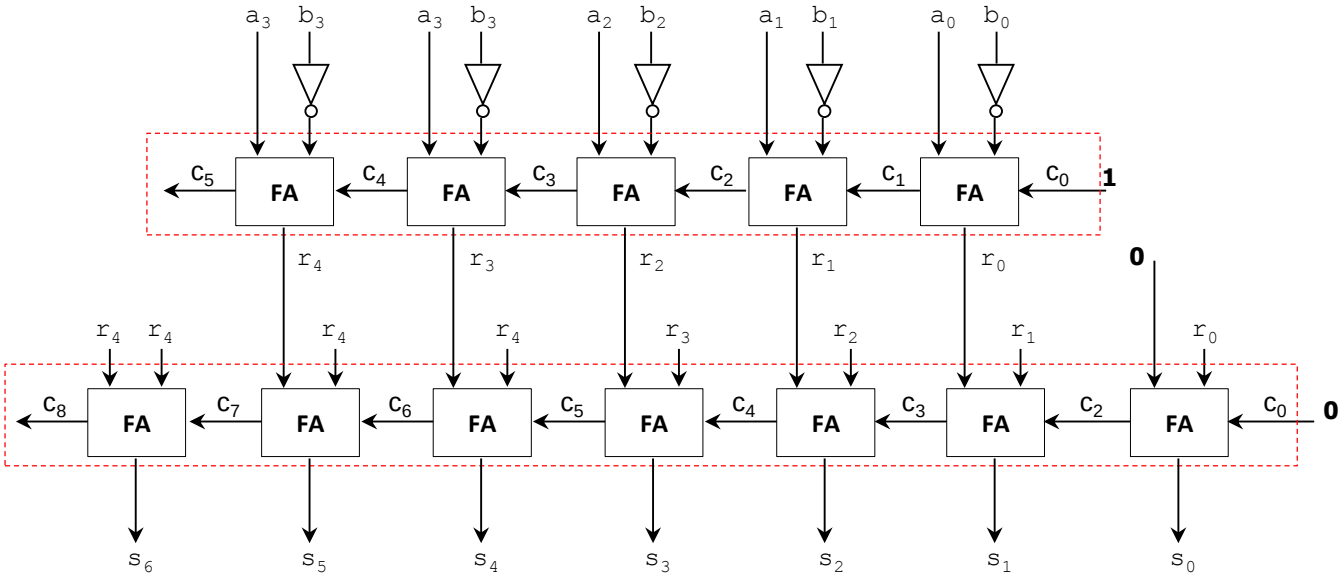
```
1 0 1 0 x        0 1 1 0 x
0 1 1 1          0 1 1 1
                 -------
                 0 1 1 0
               0 1 1 0
             0 1 1 0
           0 0 0 0
         -----------
         0 0 1 0 1 0 1 0
                 ↓
         1 1 0 1 0 1 1 0
```

PROBLEM 5 (10 PTS)

- Given two 4-bit signed (2's complement) numbers A, B, sketch the circuit that computes $(A - B) \times 3$. You can use full adders and logic gates. Make sure your circuit avoids overflow.
- BONUS POINTS (+2 PTS): Use the fewest amount of hardware resources.

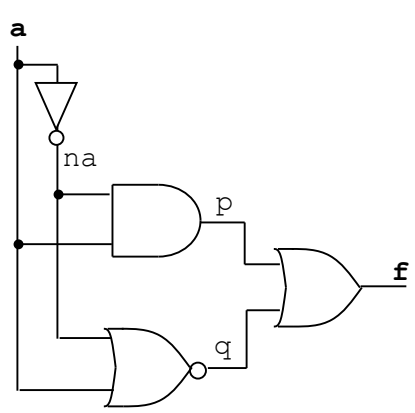
$$(A - B) \times 3 = (A - B) \times 2 + (A - B)$$

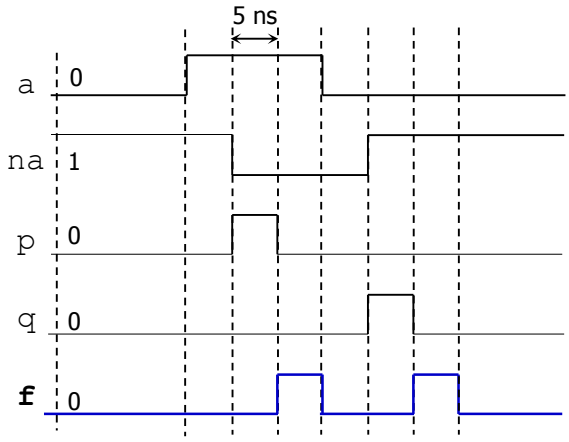
Worst case: $15 \times 3 = 45$. 45 requires 7 bits, therefore, we need to sign extend $(A - B) \times 2$ on the last addition.



PROBLEM 6 (10 PTS)

- Complete the timing diagram of the digital circuit shown below. You must consider the propagation delays. Assume the propagation delay of every gate is 5 ns. The initial values of all signals are plotted in the figure.





PROBLEM 7 (15 PTS)

- Complete the timing diagram of the circuit shown below: