Solutions - Midterm Exam

(October 13th @ 5:30 pm)

Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (20 PTS)

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

Decimal	BCD	Binary	Reflective Gray Code
52	01010010	110100	101110
34	00110100	100010	110011
85	10000101	1010101	1111111
122	000100100010	1111010	1000111

b) Complete the following table. Use the fewest number of bits in each case: (12 pts.)

	REPRESENTATION		
Decimal	**Sign-and-magnitude**	**1's complement**	**2's complement**
-63	1111111	1000000	1000001
-26	111010	100101	100110
0	00	11111	0
110	01101110	01101110	01101110
-32	1100000	1011111	100000
-16	110000	101111	10000

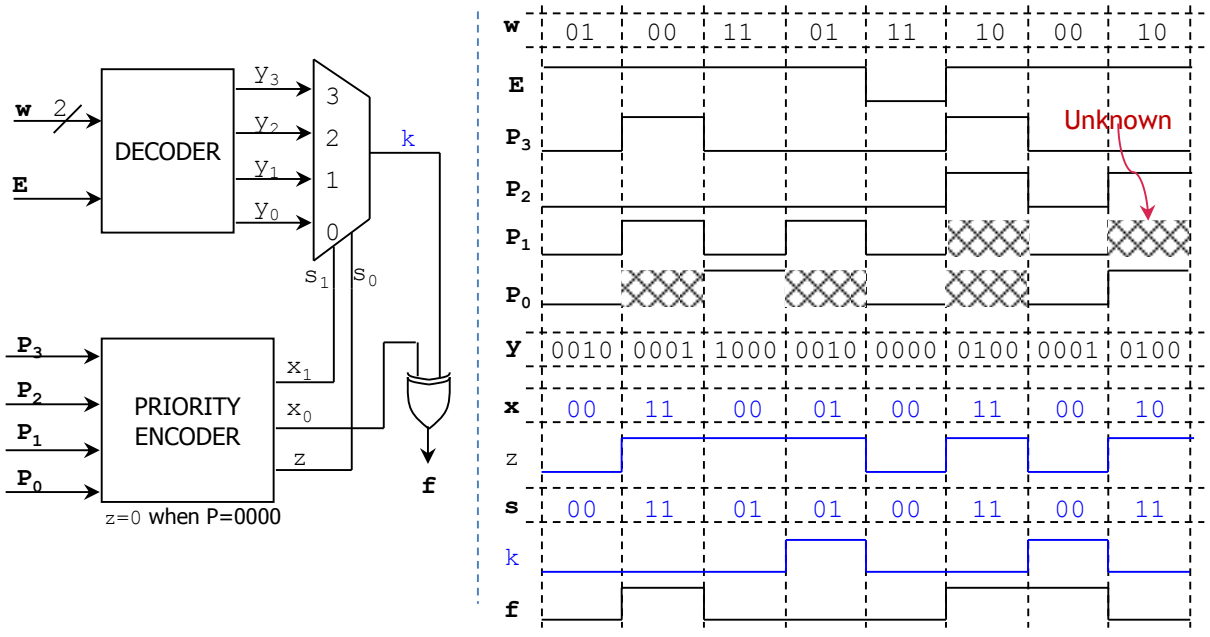
c) Convert the following decimal numbers to their 2's complement representations. (2 pts)

✓ -19.6875
+19.6875 = 010011.1011
⇒ -19.6875 = 101100.0101

✓ 16.3125
+16.3125 = 010000.0101

PROBLEM 2 (15 PTS)

- Complete the timing diagram of the circuit shown below. $y = y_3y_2y_1y_0$, $x = x_1x_0$, $s = s_1s_0$



PROBLEM 3 (18 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. (6 pts)

✓ 52 + 17

	c_6=1	c_5=1	c_4=0	c_3=0	c_2=0	c_1=0	c_0=0	
52 = 0x34 =	1	1	0	1	0	0		+
17 = 0x11 =	0	1	0	0	0	1		
Overflow! →	1	0	0	0	1	0	1	

✓ 24 - 34

Borrow out! →	b_6=1	b_5=0	b_4=0	b_3=1	b_2=1	b_1=0	b_0=0	
24 = 0x18 =		0	1	1	0	0	0	-
34 = 0x22 =		1	0	0	0	1	0	
		1	1	0	1	1	0	

b) Perform the following operations, where numbers are represented in 2's complement. Indicate every carry from c_0 to c_n. For each case, use the fewest number of bits to represent the summands and the result so that overflow is avoided. (8 pts)

✓ -26 + 45

n = 7 bits

$c_7 \oplus c_6 = 0$ No Overflow

	c_7=1	c_6=1	c_5=0	c_4=1	c_3=1	c_2=0	c_1=0	c_0=0	
-26 =	1	1	0	0	1	1	0		+
45 =	0	1	0	1	1	0	1		
19 =	0	0	1	0	0	1	1		

-26 + 45 = 19 ∈ $[-2^6, 2^6-1]$ → no overflow

✓ -63 - 15

n = 7 bits

$c_7 \oplus c_6 = 1$ Overflow!

	c_7=1	c_6=0	c_5=0	c_4=0	c_3=0	c_2=0	c_1=1	c_0=0	
-63 =	1	0	0	0	0	0	1		+
-15 =	1	1	1	0	0	0	1		
	0	1	1	0	0	1	0		

-63 -15 = -78 ∉ $[-2^6, 2^6-1]$ → overflow!

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

$c_8 \oplus c_7 = 0$ No Overflow

	c_8=1	c_7=1	c_6=0	c_5=0	c_4=0	c_3=0	c_2=0	c_1=1	c_0=0	
-63 =	1	1	0	0	0	0	0	1		+
-15 =	1	1	1	1	0	0	0	1		
	1	0	1	1	0	0	1	0		

-63 -15 = -78 ∈ $[-2^7, 2^7-1]$ → no overflow

c) Perform binary multiplication of the following numbers that are represented in 2's complement arithmetic with 4 bits. (4 pts)

✓ -7 x 6.

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

PROBLEM 4 (10 PTS)

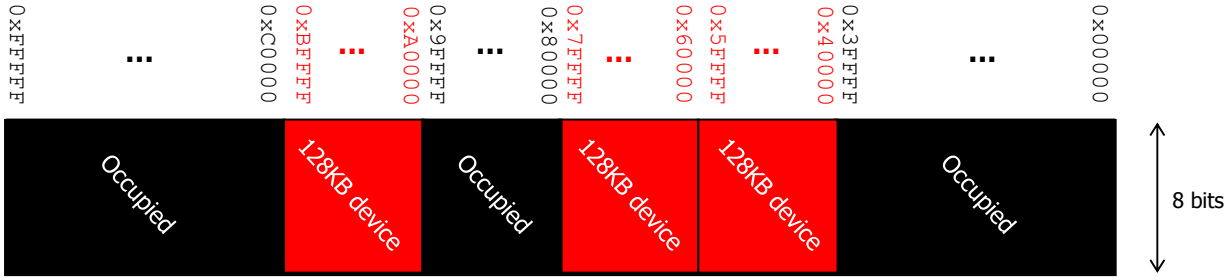
- The figure below depicts the entire memory space of a microprocessor. Each memory address occupies one byte. 1KB = 2^{10} bytes, 1MB = 2^{20} bytes, 1GB = 2^{30} bytes
  - ✓ What is the size (in bytes, KB, or MB) of the memory space? What is the address bus size of the microprocessor?

    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 128 KB, how many bits do we require to address those 128 KB of memory?

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

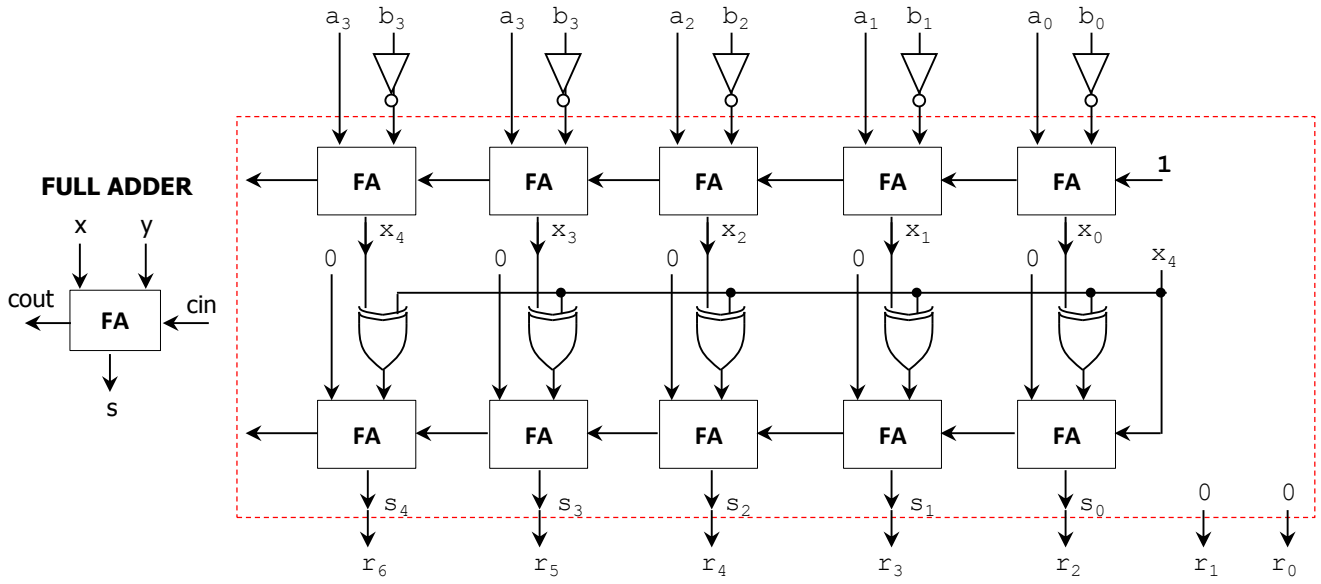
  - ✓ We want to connect the 128 KB memory chip to the microprocessor. For optimal implementation, we must place those 128 KB in an address range where every single address shares some MSBs. Provide a list of all the possible address ranges that the 128 KB memory chip can occupy. You can only use any of the non-occupied portions of the memory space as shown below.
    - `0x40000` to `0x5FFFF`
    - `0x60000` to `0x7FFFF`
    - `0xA0000` to `0xBFFFF`



PROBLEM 5 (10 PTS)

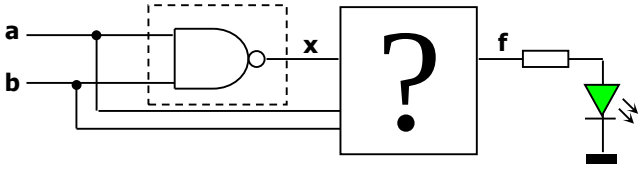
- Sketch the circuit that computes $|A-B|\times 4$, where A, B are 4-bit <u>signed</u> (2's complement) numbers. For example: $A = 1010, B = 0111 \rightarrow |A-B|\times 4 = 13\times 4 = 52$. You can only use full adders and logic gates. Your circuit must avoid overflow.

  $A = a_3a_2a_1a_0$, $B = b_3b_2b_1b_0$
  $A, B \in [-8,7] \rightarrow A, B$ require 4 bits in 2C representation.
  - ✓ $X = A - B \in [-15,15]$ requires 5 bits in 2C.
  - ✓ $|X| = |A-B| \in [0,15]$ requires 5 bits in 2C. Thus, the second operation $0 \pm X$ only requires 5 bits.
    - If $x_4 = 1 \rightarrow X < 0 \rightarrow$ we do $0 - X$.
    - If $x_4 = 0 \rightarrow X \geq 0 \rightarrow$ we do $0 + X$.
  - ✓ $R = |A-B|\times 4 \in [0,60]$ requires 7 bits in 2C. Note that the MSB is always 0.

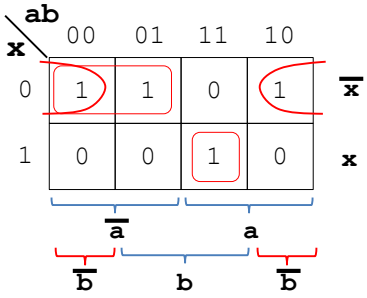


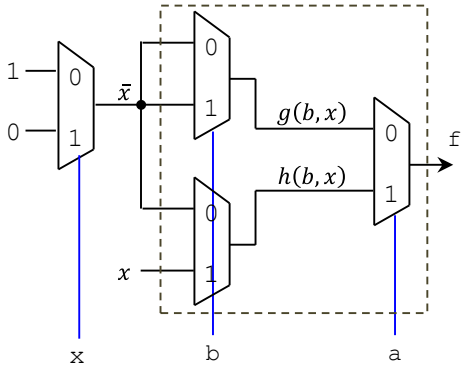
PROBLEM 6 (15 PTS)

- Using only 2-to-1 MUXs, design a circuit that verifies the logical operation of an NAND gate. f='1' (LED ON) if the NAND gate does NOT work properly.
  Assumption: when the NAND gate is not working, it generates 1's instead of 0's and vice versa.



a	b	x	f	x_{good}
0	0	0	1	1
0	0	1	0	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0





$f = xab + \bar{x}\bar{b} + \bar{x}\bar{a}$

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

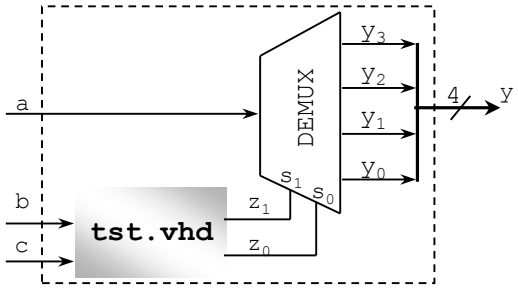
$g(b,x) = \bar{b}g(0,x) + bg(1,x) = \bar{b}(\bar{x}) + b(\bar{x})$

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

PROBLEM 7 (12 PTS)

- Complete the timing diagram of the following circuit. The VHDL code (`tst.vhd`) corresponds to the shaded circuit.

$z = z_1 z_0$, $y = y_3 y_2 y_1 y_0$




```
library ieee;
use ieee.std_logic_1164.all;

entity tst is
  port (b,c : in std_logic;
        z: out std_logic_vector(1 downto 0));
end tst;
```

```
architecture bhv of tst is

begin

  process (b, c)
  begin
     z <= c & b;
     if c = '0' then
        z <= "10";
     end if;
  end process;

end bhv;
```