Solutions - Midterm Exam

(October 18th @ 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: (5 pts.)

Decimal	BCD	Binary	Reflective Gray Code
37	00110111	100101	110111
162	000101100010	10100010	11110011
69	01101001	1000101	1100111

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
-16	110000	101111	10000
-11	11011	10100	10101
29	011101	011101	011101
-32	1100000	1011111	100000
0	00	11111	0

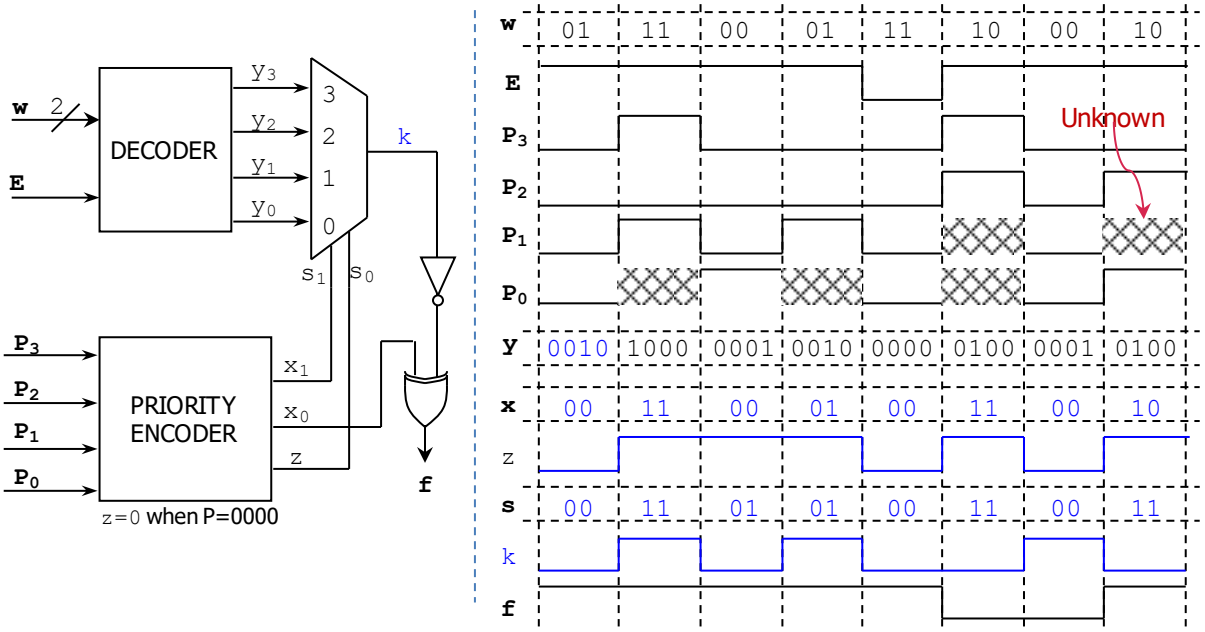
c) Convert the following decimal numbers to their 2's complement representations. (3 pts)

✓ 17.875
+17.875 = 010001.111

✓ -16.3125
+16.3125 = 010000.0101
⇒ -16.3125 = 101111.1011

Problem 2 (15 pts)

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

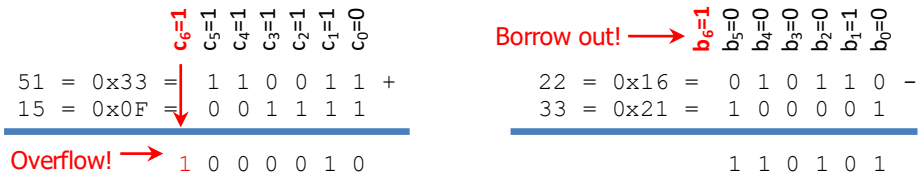


Problem 3 (17 pts)

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)

✓ 51 + 15

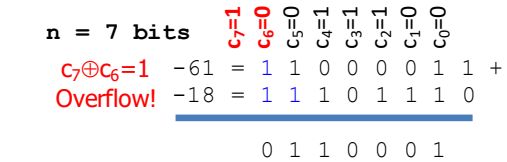
✓ 22 - 33



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)

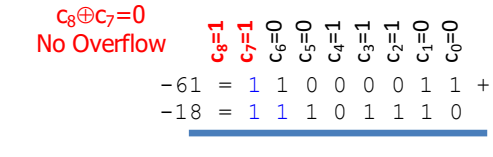
✓ -61 – 18

n = 7 bits



-61 -18 = -79 ∉ [-2^6, 2^6-1] → overflow!

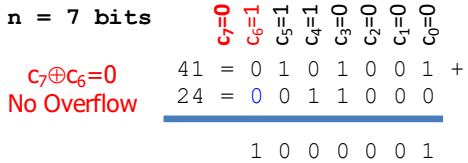
To avoid overflow: n = 8 bits (sign-extension)



-61 -18 = -79 ∈ [-2^7, 2^7-1] → no overflow

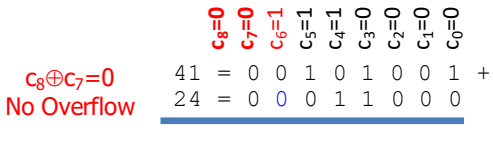
✓ 41 + 24

n = 7 bits



41 + 24 = 65 ∉ [-2^6, 2^6-1] → overflow!

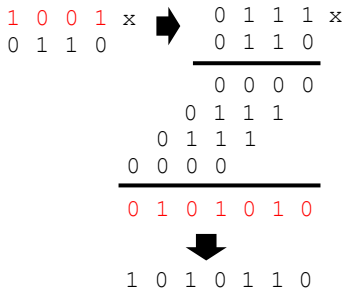
To avoid overflow: n = 8 bits (sign-extension)



41 + 24 = -65 ∈ [-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. (3 pts)

✓ -7 x 6.



PROBLEM 4 (10 PTS)

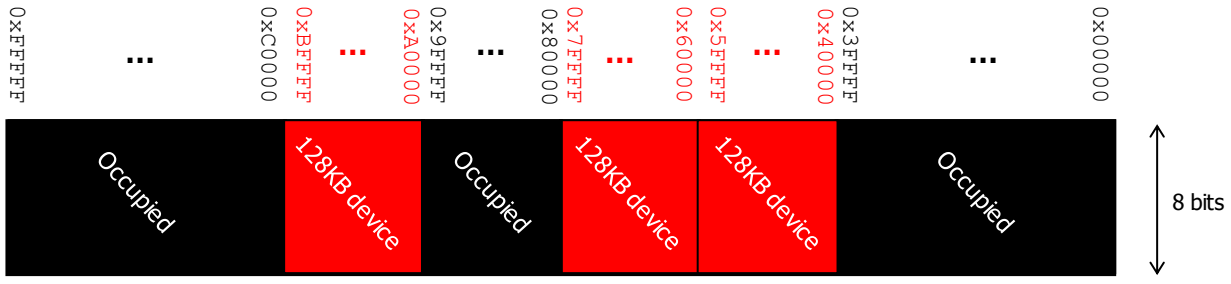
- The figure below depicts the entire memory space of a microprocessor. Each memory address occupies one byte. 1KB = 2^{10} bytes, 1MB = 2^{20} bytes, 1GB = 2^{30} bytes
  - ✓ What is the size (in bytes, KB, or MB) of the memory space? What is the address bus size of the microprocessor? (2 pts.)

    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? (1 pt.)

    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 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 the non-occupied portions of the memory space as shown below.
    - 0x40000 to 0x5FFFF
    - 0x60000 to 0x7FFFF
    - 0xA0000 to 0xBFFFF

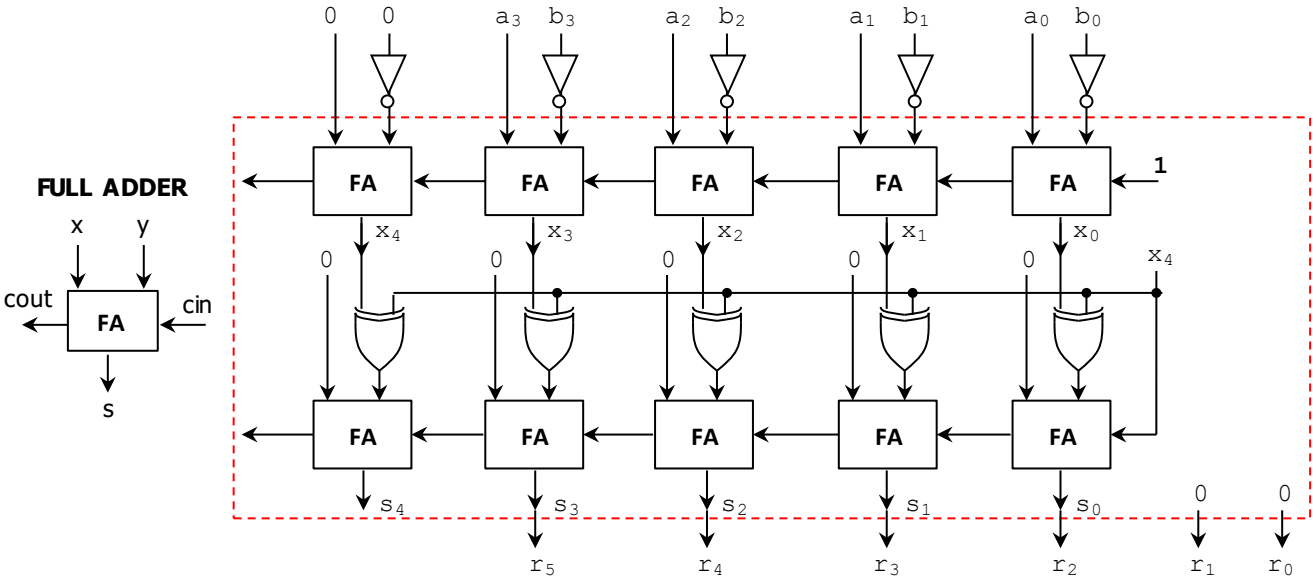


PROBLEM 5 (10 PTS)

- Sketch the circuit that computes $|A-B|\times 4$, where A, B are 4-bit unsigned numbers. For example: $A=0101, B=1101 \rightarrow |A-B|\times 4 = 8\times 4 = 32$. You can only use full adders and logic gates. Your circuit must avoid overflow.

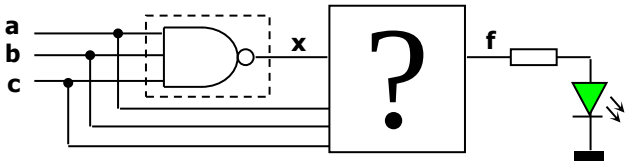
$A = a_3a_2a_1a_0$, $B = b_3b_2b_1b_0$
$A, B \in [0,15] \rightarrow A, B$ require 4 bits in unsigned representation. However, to get the proper result of $A-B$, we need to use the 2C representation, where A, B require 5 bits in 2C.

- ✓ $X = A - B \in [-15,15]$ requires 5 bits in 2C. Thus, we need to zero-extend A and B to convert them to 2C representation.
- ✓ $|X| = |A-B| \in [0,15]$ 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. The unsigned result only requires 6 bits.

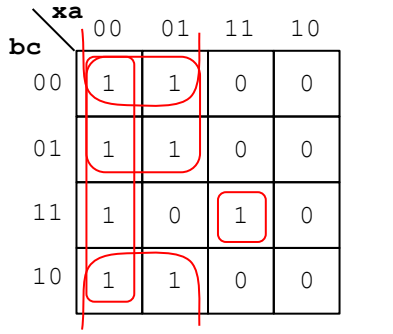


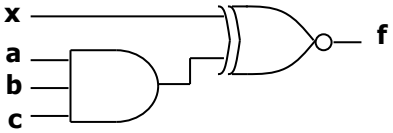
PROBLEM 6 (16 PTS)

- We want to design a circuit that verifies the logical operation of a 3-input 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).
  - ✓ Provide the Boolean equation for f and sketch the circuit using logic gates. (4 pts.)
  - ✓ Implement the circuit using ONLY 2-to-1 MUXs (AND, OR, NOT, XOR gates are not allowed). (12 pts)



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





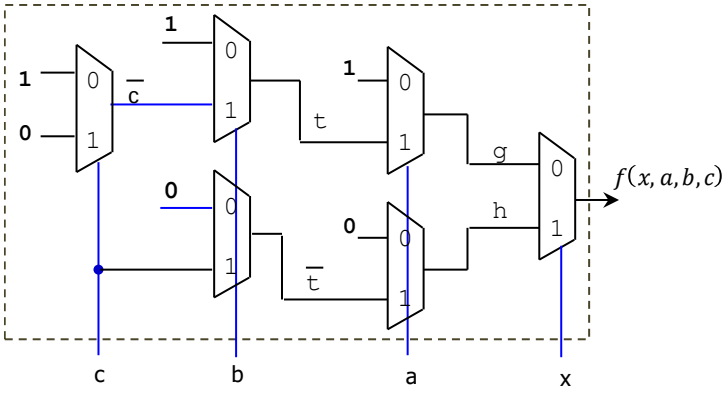
$$f = \bar{x}\bar{b} + \bar{x}\bar{c} + \bar{x}\bar{a} + xabc = \bar{x}\overline{(abc)} + x(abc)$$

$$f = \overline{x \oplus (abc)}$$

$f(x,a,b,c) = \bar{x}f(0,a,b,c) + xf(1,a,b,c) = \bar{x}(\overline{abc}) + x(abc) = \bar{x}g(a,b,c) + xg(a,b,c)$
$g(a,b,c) = \bar{a}g(0,b,c) + ag(1,b,c) = \bar{a}(1) + a(\overline{bc})$
$h(a,b,c) = \bar{a}h(0,b,c) + ah(1,b,c) = \bar{a}(0) + a(bc)$

$t(b,c) = \overline{bc} = \bar{b}(1) + b(\bar{c})$
$\bar{t}(b,c) = bc = \bar{b}(0) + b(c)$

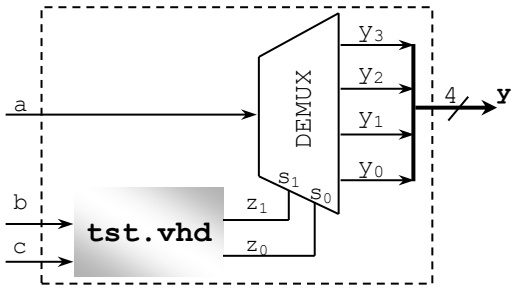
Also: $\bar{c} = \bar{c}(1) + c(0)$



PROBLEM 7 (12 PTS)

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

$z = z_1z_0$, $y = y_3y_2y_1y_0$




```
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 <= b & c;
     if c = '1' then
        z <= b&'1';
     end if;
  end process;

end bhv;
```