# Homework 1

(Due date: January $21^{st}$ @ 5:30 pm)
Presentation and clarity are very important!

## PROBLEM 1 (25 PTS)

a) Simplify the following functions using ONLY Boolean Algebra Theorems. For each resulting simplified function, sketch the logic circuit using AND, OR, XOR, and NOT gates. (12 pts)

- ✓ $F = \overline{A(\bar{B}\oplus\bar{C}) + \bar{B}}$
- ✓ $F = (\bar{C} + \bar{B})(C + A)(\bar{B} + A) + CA$
- ✓ $F(X,Y,Z) = \prod(M_1, M_3, M_6, M_7)$
- ✓ $F = \overline{(\overline{X + Z})Y + X\bar{Y}Z}$

b) Based on the formula $x\oplus y = x\bar{y} + \bar{x}y$, demonstrate that $(a\oplus b)\oplus c = a\oplus(b\oplus c) = b\oplus(a\oplus c)$. You can express each function using the canonical sum of products, or complete the truth table for each function. (5 pts)

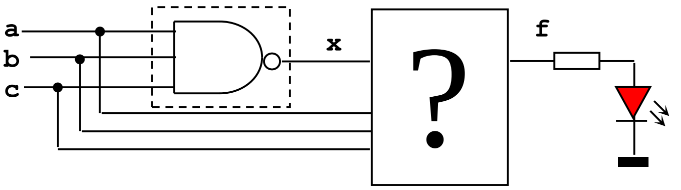
c) For the following Truth table with two outputs: (8 pts)

- Provide the Boolean functions using the Canonical Sum of Products (SOP), and Product of Sums (POS).
- Express the Boolean functions using the minterms and maxterms representations.
- Sketch the logic circuits as Canonical Sum of Products and Product of Sums.

| x | y | z | $f_1$ | $f_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

## PROBLEM 2 (10 PTS)

- Design a circuit (<u>simplify your circuit</u>) 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.



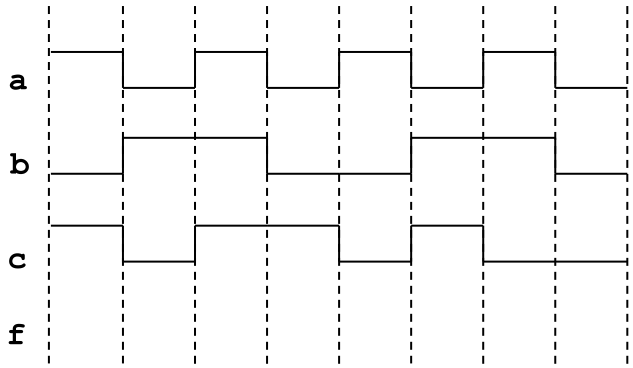
## PROBLEM 3 (15 PTS)

- Complete the truth table for a circuit with 4 inputs $x, y, z, w$ that activates an output ($f = 1$) when the number of 1's in the inputs is odd. For example: If $xyzw = 1100 \rightarrow f = 0$. If $xyzw = 1011 \rightarrow f = 1$.
- Provide the Boolean function using the minterm representation.
- Sketch the logic circuit using <u>ONLY</u> 2-input NAND gates. Tip: try to simplify the function using XOR gates.

## PROBLEM 4 (20 PTS)

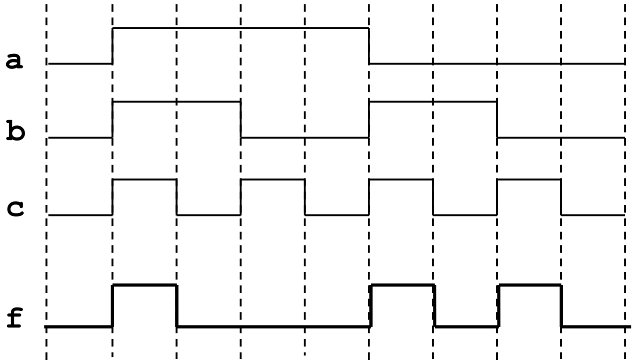
a) Complete the timing diagram of the logic circuit whose VHDL description is shown below: (5 pts)

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

entity circ is
  port ( a, b, c: in std_logic;
         f: out std_logic);
end circ;

architecture st of circ is
  signal x, y: std_logic;

begin
  x <= not(a xor b);
  y <= x nand c;
  f <= y xor (not a);
end st;
```

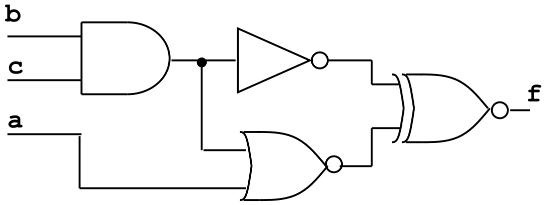


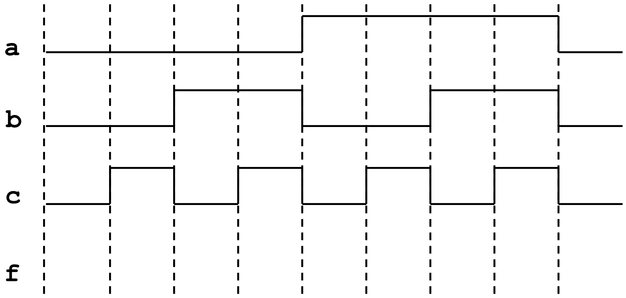
b) The following is the timing diagram of a logic circuit with 3 inputs. Sketch the logic circuit that generates this waveform. Then, complete the VHDL code. (10 pts)

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

entity wav is
  port ( a, b, c: in std_logic;
             f: out std_logic);
end wav;

architecture st of wav is
-- ???
begin
   -- ???
end st;
```



c) Complete the timing diagram of the following circuit: (5 pts)



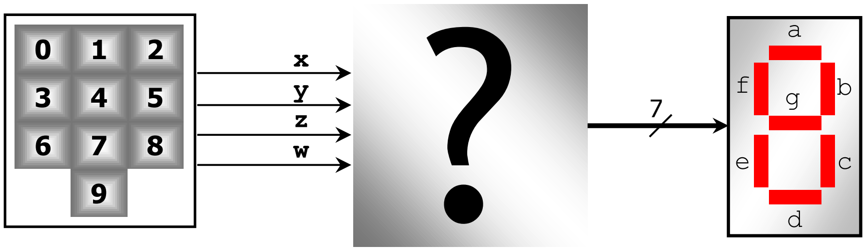


## PROBLEM 5 (30 PTS)

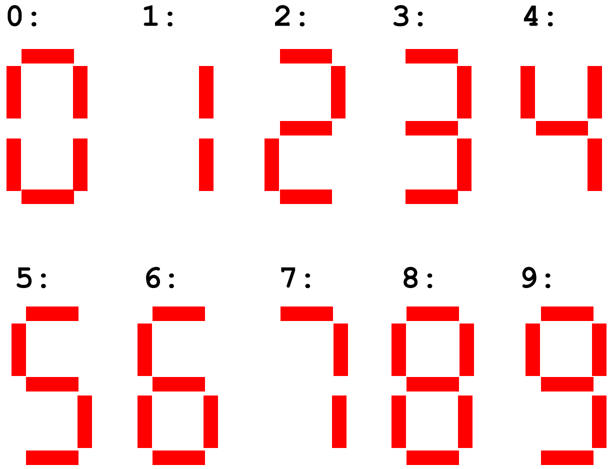
- A numeric keypad produces a 4-bit code as shown below. We want to design a logic circuit that converts each 4-bit code to a 7-segment code, where each segment is an LED: A LED is ON if it is given a logic '1'. A LED is OFF if it is given a logic '0'.

✓ Complete the truth table for each output $(a,b,c,d,e,f,g)$.

✓ Provide the simplified expression for each output $(a,b,c,d,e,f,g)$. Use Karnaugh maps for $a,b,c,d,e$ and the Quine-McCluskey algorithm for $f,g$. Note it is safe to assume that the codes `1010` to `1111` will not be produced by the keypad.



| Value | X | Y | Z | W | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | | | | | | |
| 1 | 0 | 0 | 0 | 1 | | | | | | | |
| 2 | 0 | 0 | 1 | 0 | | | | | | | |
| 3 | 0 | 0 | 1 | 1 | | | | | | | |
| 4 | 0 | 1 | 0 | 0 | | | | | | | |
| 5 | 0 | 1 | 0 | 1 | | | | | | | |
| 6 | 0 | 1 | 1 | 0 | | | | | | | |
| 7 | 0 | 1 | 1 | 1 | | | | | | | |
| 8 | 1 | 0 | 0 | 0 | | | | | | | |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 0 | | | | | | | |
| | 1 | 0 | 1 | 1 | | | | | | | |
| | 1 | 1 | 0 | 0 | | | | | | | |
| | 1 | 1 | 0 | 1 | | | | | | | |
| | 1 | 1 | 1 | 0 | | | | | | | |
| | 1 | 1 | 1 | 1 | | | | | | | |


0:
1:
2:
3:
4:
5:
6:
7:
8:
9: