Homework 1

(Due date: January 17th @ 5:30 pm)

Presentation and clarity are very important!

PROBLEM 1 (27 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. (14 pts)

- ✓ $F = \overline{\bar{A}(B+\bar{C})+A}$
- ✓ $F(X,Y,Z) = \prod(M_2, M_4, M_6, M_7)$
- ✓ $F = (Z+X)(\bar{Z}+\bar{Y})(\bar{Y}+X)$
- ✓ $F = \overline{(X+\bar{Y})Z+\bar{X}\bar{Y}\bar{Z}}$

b) Using ONLY Boolean Algebra Theorems, demonstrate that the XOR operation is associative: (5 pts)

$$(a \oplus b) \oplus c = a \oplus (b \oplus c) = b \oplus (a \oplus c)$$

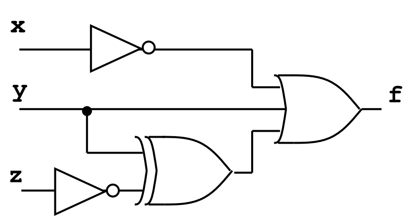
c) For the following Truth table with two outputs: (8 pts)

- 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	0
0	0	1	1	0
0	1	0	1	1
0	1	1	1	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	1
1	1	1	0	1

PROBLEM 2 (25 PTS)

a) Construct the truth table describing the output of the following circuit and write the simplified Boolean equation (6 pts).

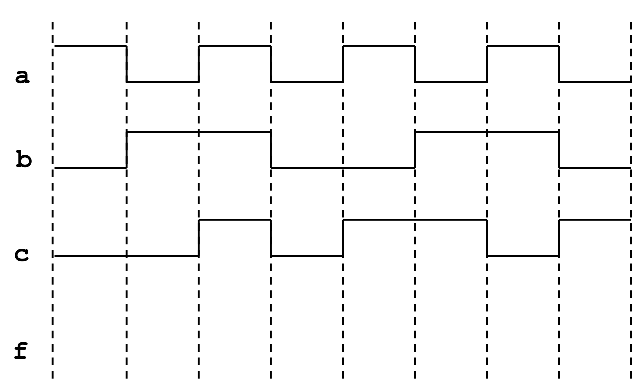


x	y	z	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

$f =$

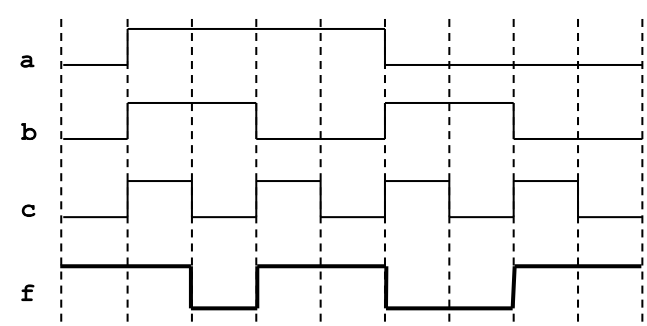
b) Complete the timing diagram of the logic circuit whose VHDL description is shown below: (6 pts)

```
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 struct of circ is
  signal x, y: std_logic;

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

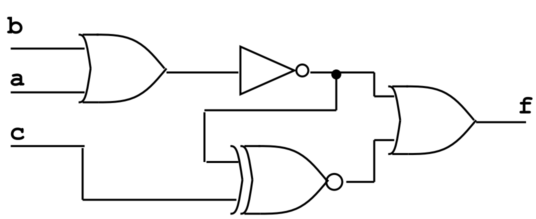


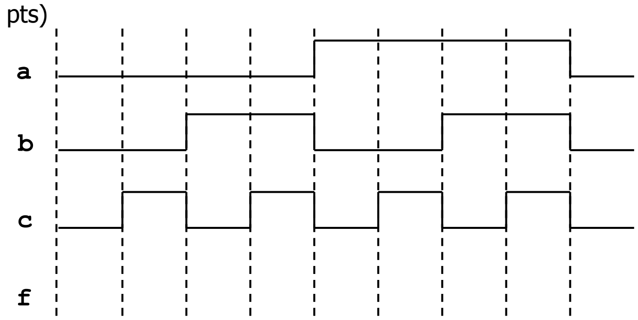
c) The following is the timing diagram of a logic circuit with 3 inputs. Sketch the logic circuit that generates this waveform. Then, complete the VHDL code. (8 pts)

```
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 struct of wav is
-- ???
begin
   -- ???
end struct;
```



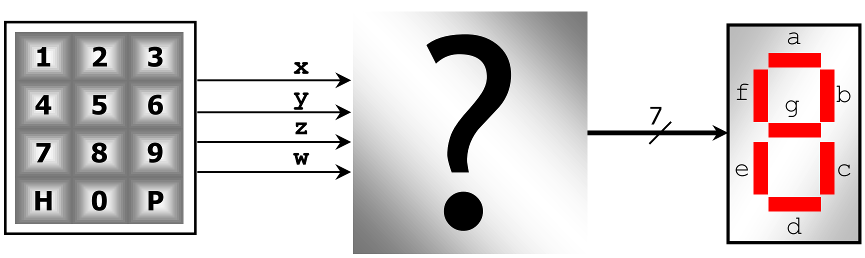
d) Complete the timing diagram of the following circuit: (5 pts)



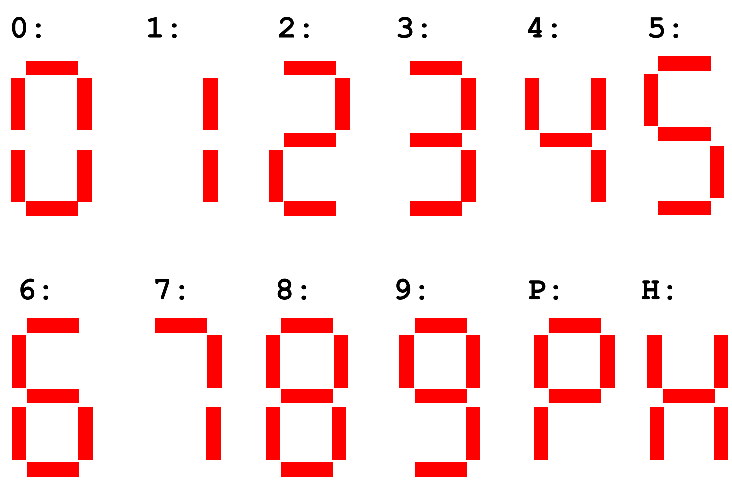


PROBLEM 3 (25 PTS)

- 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 c, d, e, f, g and the Quine-McCluskey algorithm for a, b. Note: It is safe to assume that the codes `1100` to `1111` will not be produced by the keypad.



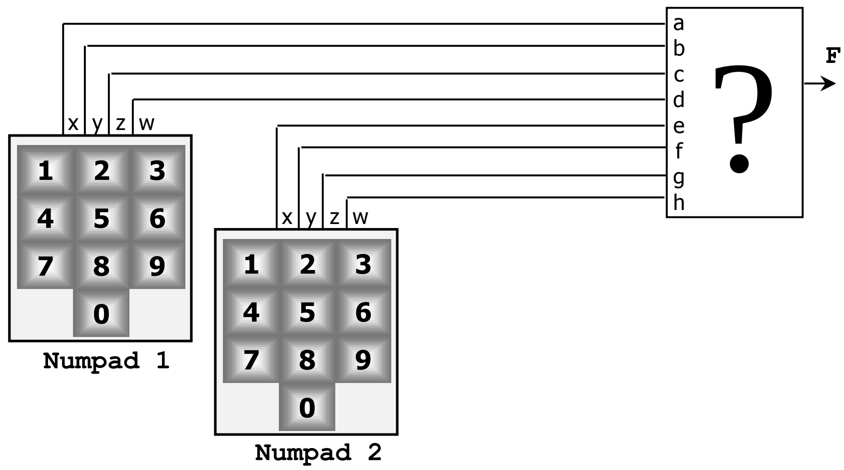
Value	X Y Z W	a b c d e f g
0	0 0 0 0	
1	0 0 0 1	
2	0 0 1 0	
3	0 0 1 1	
4	0 1 0 0	
5	0 1 0 1	
6	0 1 1 0	
7	0 1 1 1	
8	1 0 0 0	
9	1 0 0 1	1 1 1 1 0 1 1
P	1 0 1 0	
H	1 0 1 1	
	1 1 0 0	
	1 1 0 1	
	1 1 1 0	
	1 1 1 1	



PROBLEM 4 (12 PTS)

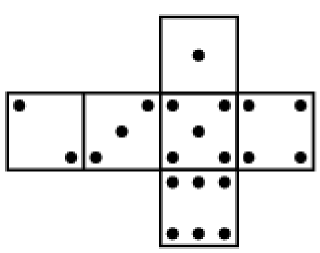
- Design a logic circuit (<u>simplify your circuit</u>) that opens a lock ($f = 1$) whenever the user presses the correct number on each numpad (numpad 1: **7**, numpad2: **2**). The numpad encodes each decimal number using BCD encoding (see figure). We expect that the 4-bit groups generated by each numpad be in the range from `0000` to `1001`. Note that the values from `1010` to `1111` are assumed not to occur.
  <u>Suggestion</u>: Create two circuits: one that verifies the first number (**7**), and another that verifies the second number (**2**). Then perform the AND operation on the two outputs. This avoids creating a truth table with 8 inputs.

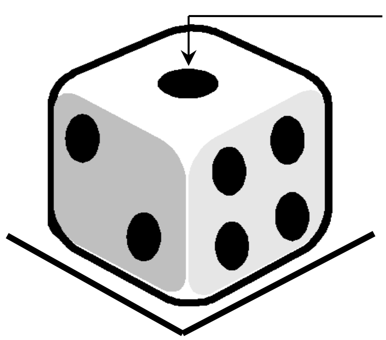
BCD code x y z w	Number pressed
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9



PROBLEM 5 (11 PTS)

- The following die has a sensor on each side. Whenever a side rests on a surface, the sensor on that side generates a logic '1' (transmitted wirelessly to a controller); otherwise, it generates a '0'. The sensors outputs are named S1, S2, S3, S4, S5, S6.
- We want to design a circuit that reads the state of the 6 sensors and outputs a 3-bit value L representing the decimal value (unsigned integer) of the opposite side (upper surface). The output L is connected to 3 LEDs: A LED ON is represented by a logic '1', while the LED OFF is represented by '0'. For example, in the figure below:
  - ✓ The resting side has six dots. This means that the state of the sensors is S6=1, S5=0, S4=0, S3=0, S2=0, S1=0.
  - ✓ The opposite side (upper surface) has one dot representing the decimal number '**1**'. Thus, the output L must be `001`.
- Under normal operation, we expect only one sensor activated at a time. However, due to a variety of problems, we might have the following cases:
  - ✓ Two or more sensors produce a '1' at the same time: Here, the state of the LEDs must be `000`.
  - ✓ No sensor produces a '1': In this case, the state of the LEDs must be `000`.
- Using the state of the sensors as inputs, provide the Boolean expression for each LED: L_2, L_1, L_0. First, build the truth table where the inputs are S6-S1 and the outputs are L_2-L_0.

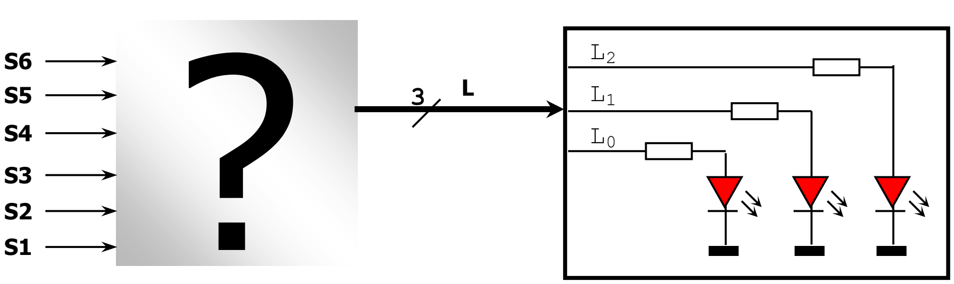




Resulting number: 1

State of Sensors

S6	S5	S4	S3	S2	S1
1	0	0	0	0	0



L_2	L_1	L_0	Number
0	0	0	-
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	-