Homework 1

(Due date: January 23rd @ 5:30 pm)

Presentation and clarity are very important! Show your procedure!

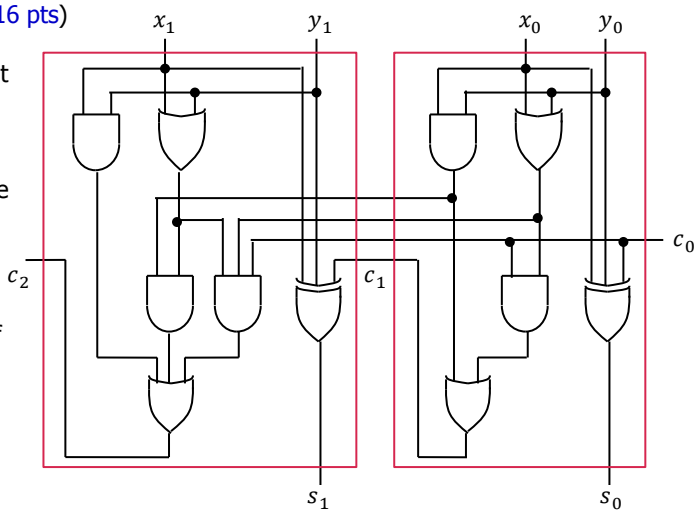
Problem 1 (30 pts)

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)

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

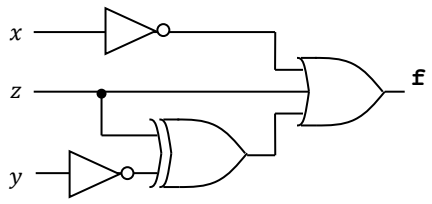
b) Given the following circuit with inputs x_0, x_1, y_0, y_1, c_0: (16 pts)

- Provide the Boolean expression (based on the circuit inputs) for s_0, s_1, c_1, c_2. (7 pts)
- For s_0 and c_1: (9 pts)
  - ✓ Express the Boolean functions using both the minterms and maxterms representations. (4 pts)
  - ✓ Provide the Boolean functions using the Canonical Sum of Products (SOP), and Product of Sums (POS).
  - ✓ Sketch the logic circuits as Canonical Sum of Products and Product of Sums. (3 pts)



Problem 2 (24 pts)

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

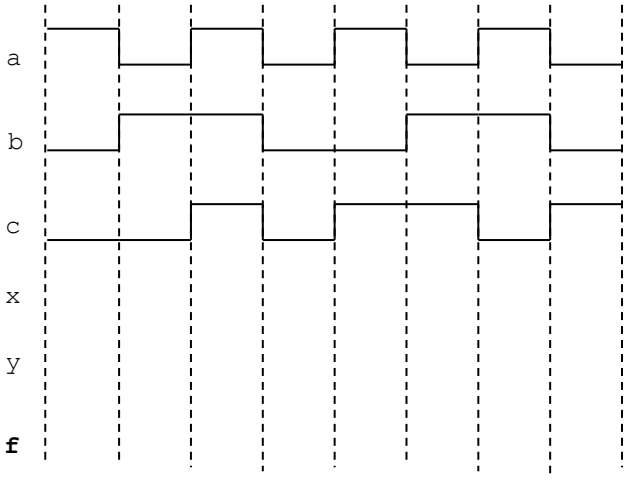


x	y	z	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

$f =$

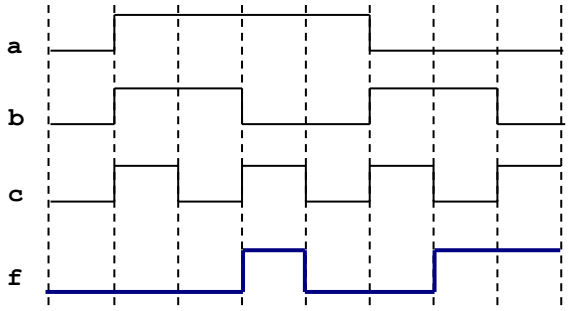
b) Complete the timing diagram of the logic circuit whose VHDL description is shown below: (5 pts)

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

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

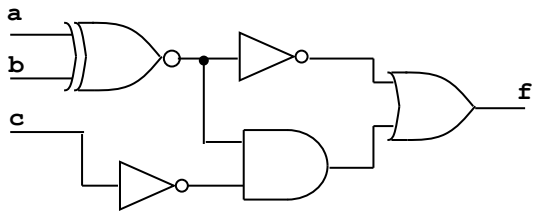


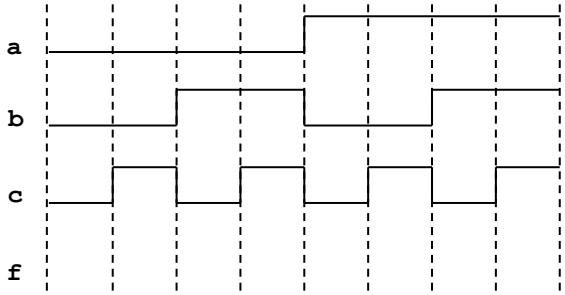
c) The following is the timing diagram of a logic circuit with 3 inputs. Sketch the logic circuit that generates this waveform. Then, complete the VHDL code. (8 pts)

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

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

architecture struct of wave is
-- ???
begin

   -- ???

end struct;
```



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





PROBLEM 3 (10 PTS)

- A majority gate has an output value of 1 if there are more 1's than 0's on its inputs. The output is 0 otherwise.
- Design (provide the simplified Boolean equation and sketch the logic circuit) a 4-input majority gate with inputs a, b, c, d, and output f.

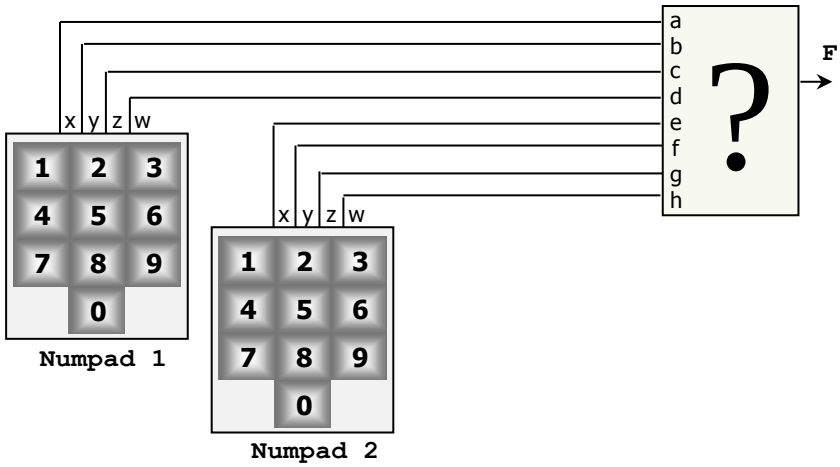
  The function break ties in favor of zeros when the number of inputs is even. Example: abcd=1100 → f = 0.

PROBLEM 4 (11 PTS)

- Design a logic circuit (simplify your circuit) that opens a lock ($f = 1$) whenever the user presses the correct number on each numpad (numpad 1: **9**, numpad 2: **4**). 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.

  Suggestion: Create two circuits: one that verifies the first number (**9**), and another that verifies the second number (**4**). Then perform the AND operation on the two outputs. This avoids creating a truth table with 8 inputs.

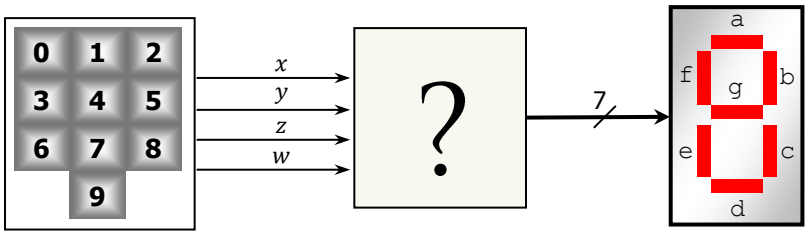
BCD code x y z w	Number pressed
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9



PROBLEM 5 (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 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