1. (10 points) List the minterms and maxterms for the expression \( A'B + (A + B')C + (A+C)'B \)?

<table>
<thead>
<tr>
<th>ABC</th>
<th>F</th>
<th>minterms</th>
<th>maxterms</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td></td>
<td>( A+B+C )</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>( A'B'C )</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>( A'BC' )</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>( A'BC )</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>( A'+B +C )</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>( AB'C )</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>( A'+B'+C )</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>( ABC )</td>
<td></td>
</tr>
</tbody>
</table>

Express the complement of \( A'B + (A + B')C + (A+C)'B \) in the simplest form you can using Boolean algebra. Show your work.

\[
F = A'B + (A + B')C + (A+C)'B = A'B + AC + B'C + A'BC' = A'B + AC + B'C
\]

\[
F' = (A+B')(A'+C')(B+C') = (A+B')(A'B+C') = AC'+B'C' = (A+B')C'
\]
2. (10 points) For each VHDL code fragment shown below, write an equivalent set of simple signal assignments (each involving just a single signal, not a vector of signals).

(a)  
\[ a \leq '1'; b \leq x \text{xor} z; \]
\[ \text{case } c \text{ is} \]
\[ \text{when } "00" \Rightarrow a \leq x \text{ or } y; \]
\[ \text{when } "01" \Rightarrow b \leq y; \]
\[ \text{when others } \Rightarrow a \leq c(1); \]
\[ \text{end case;} \]
\[ a \leq (\text{not } c(0) \text{ and not } c(1) \text{ and } (x \text{ or } y)) \text{ or } c(0) \text{ or } c(1); \]
\[ b \leq ((c(1) \text{ or not } c(0)) \text{ and } (x \text{ xor } z)) \text{ or } (\text{not } c(1) \text{ and } c(0) \text{ and } y); \]

(b)  
\[ x \leq c; \]
\[ \text{for i in 0 to 2 loop} \]
\[ \hspace{1cm} z(i) \leq a(i) \text{ xor } x; \]
\[ \hspace{1cm} x \leq (\text{not } a(i)) \text{ or } x; \]
\[ \text{end loop;} \]
\[ x \leq (\text{not } a(2)) \text{ or } x; \]
\[ z(0) \leq a(0) \text{ xor } x; \]
\[ z(1) \leq a(1) \text{ xor } x; \]
\[ z(2) \leq a(2) \text{ xor } x; \]
3. (10 points) The simulation output shows selected signals from the processor introduced in section 1 of the course notes. The portions of the output corresponding to four different instructions are outlined. Note that some parts of the simulation output have been blanked out. For the labeled blanks, fill in the missing information below.

A. __________ 0001 __________  
B. __________ 1FFF __________  
C. __________ A008 __________  
D. __________ 000B __________  
E. __________ 000D __________  

0000  halt – halt execution  
0001  negate – ACC := –ACC  
1xxx  immediate load – if sign bit of xxx is 0 then ACC := 0xxx else ACC := fxxx  
2xxx  direct load – ACC := M[0xxx]  
3xxx  indirect load – ACC := M[M[0xxx]]  
4xxx  direct store – M[0xxx] := ACC  
5xxx  indirect store – M[M[0xxx]] := ACC  
6xxx  branch – PC := 0xxx  
7xxx  branch if zero – if ACC = 0 then PC := 0xxx  
8xxx  branch if positive – if ACC > 0 then PC := 0xxx  
9xxx  branch if negative – if ACC < 0 then PC := 0xxx  
axxx  add – ACC := ACC + M[0xxx]  
dxxx  and – ACC := ACC and M[0xxx]
4. (12 points) Consider the circuit shown below. Suppose this circuit is implemented directly, using CMOS, with each AND gate implemented using a NAND and an inverter (similarly for OR gates). What is the worst-case delay for the circuit, assuming that NAND and NOR gates have a delay of 2 ns each, and inverters have a delay of 1 ns? Highlight the path through the circuit that accounts for the worst-case delay.

![Circuit Diagram]

The worst-case delay is 10 ns for the highlighted path.

Show an alternate implementation, using only NANDs, NORs and inverters that reduces the delay by at least 2 ns.

![Alternate Circuit Diagram]

This circuit has a worst-case delay of 7 ns.

Compare the number of transistors required to implement each of these circuits.

The original uses 42 transistors. The second version uses 28.
5. (12 points) Use a Karnaugh map to find a simplest *product-of-sums* expression for \( F(X,Y,Z) = \Sigma m(0,1,4), \, d(X,Y,Z) = \Sigma m(2,6). \)

\[
F' = YXZ \quad F = Y'X'Z'
\]

Use a Karnaugh map to find a simplest *sum-of-products* expression for \( F(A,B,C,D) = \Sigma m(1,3,4,5,9,10,12), \, d(A,B,C,D) = \Sigma m(8,11,14,15). \)

\[
F = AD' + B'D + A'BC'
\]

How many gates of each type are needed to implement this expression directly?

*4 and gates, 2 or gates, 4 inverters*
6. (12 points) The diagram at right shows a three digit, excess-3 adder. Complete the VHDL module shown below, so that it implements this circuit. Note that the digit type has been declared to be five bits, instead of four to make it easier for you to write the required VHDL. However, you will need to take this into account when you specify the logic.

```vhdl
package commonDefs is
    subtype digit is std_logic_vector(4 downto 0);
    type word is array(2 downto 0) of digit;
end package commonDefs;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use work.commonDefs.all;

entity xs3adder is port(
    A, B: in word;
    S: out word;
    carryOut: out std_logic);
end xs3adder;

architecture xs3arch of xs3adder is
    signal carry: std_logic_vector(3 downto 0);
    signal X: word;
begin
    process (A,B,carry,X) begin
        carry(0) <= '0';
        for i in 0 to 2 loop
            X(i) <= A(i) + B(i) + (“0000” & carry(i));
            carry(i+1) <= X(i)(4);
            if carry(i+1) = '1' then
                S(i) <= X(i) - “01101”;
            else
                S(i) <= X(i) - “00011”;
            end if;
        end loop;
    end process;
    carryOut <= carry(3);
end xs3arch;
```
7. (10 points) Draw a schematic for a four bit version of the count-ones circuit from design problem 2, using four input LUTs. Each output of your circuit should have three bits. Use as few LUTs as you can and make the worst-case delay as small as possible.
8. (6 points) Define each of the following terms, with respect to a positive edge-triggered $D$ flip flop.

(a) propagation delay

the propagation delay of a flip flop is the time from when the clock goes high until the time that the output changes

(b) setup time

the setup time of a flip flop is the period of time before the clock goes high during which the $D$ input is required to be stable (not changing)

(c) hold time

the hold time of a flip flop is the period of time after the clock goes high during which the $D$ input is required to be stable (not changing)