1. (8 points) Draw a logic diagram that directly implements the expression

\[ A(B' + C'D') + (C + (A'+B')D) \]

(do not simplify the expression first).

How many transistors are needed to implement this expression in CMOS technology?

The circuit has 3 AND and 4 OR gates, plus 4 inverters, so that's \(7 \times 6 + 4 \times 2 = 50\) transistors.

2. (6 points) The expression \(A'B + (B' + A)C\) has the minterm \(A'BC\) and the maxterm \((A' + B + C)\). List all its other minterms and maxterms.

<table>
<thead>
<tr>
<th>ABC</th>
<th>X</th>
<th>minterms: (A'B'C, A'BC', AB'C, ABC)</th>
<th>maxterms: ((A + B + C), (A' + B' + C))</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. (12 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. Identify the instructions that are being executed. Give the name of the instruction (e.g. direct load, branch-on-positive) and its complete numeric representation. Note that some parts of the simulation output have been blanked out.

The first instruction changes the ACC but does not access memory, so it must be either an immediate load or a negate instruction. It cannot be an immediate load, since the high order hex digit of the ACC becomes 5 and an immediate load can only make the high order hex digit 0 or F. So, it must be a negate instruction, 0001. This conclusion is confirmed by the fact that the new value of the ACC is the 2s complement of the previous value.

The second instruction also changes the ACC, but does not access memory. Since the new ACC value is not the 2s complement of the previous value, this must be an immediate load 1FFF.

The third instruction does read from memory, so it must be a direct load or an add instruction. Since the value returned from memory (which appears on the data bus) is 0001, and the new ACC value is 0000, it must be an add instruction, and since the value on the address bus is 0008, it must be, A008. This is confirmed by the fact that the new ACC value is 1 more than the previous value.

The last instruction changes the PC, so it must be a branch instruction. In fact, we can see the instruction returned on the data bus in this case is 700d, which is the branch-on-zero instruction.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>halt execution</td>
</tr>
<tr>
<td>0001</td>
<td>negate the value in the ACC</td>
</tr>
<tr>
<td>1xxx</td>
<td>change the value of the ACC to xxx</td>
</tr>
<tr>
<td>2xxx</td>
<td>load the contents of memory location xxx into the ACC</td>
</tr>
<tr>
<td>3xxx</td>
<td>load the ACC from the memory location whose address is stored in memory location xxx</td>
</tr>
<tr>
<td>4xxx</td>
<td>store the value in the ACC in memory location xxx</td>
</tr>
<tr>
<td>5xxx</td>
<td>store the value in the ACC in the memory location whose address is stored in location xxx</td>
</tr>
<tr>
<td>6xxx</td>
<td>change the value of the PC to xxx</td>
</tr>
<tr>
<td>7xxx</td>
<td>change the value of the PC to xxx if ACC = 0</td>
</tr>
<tr>
<td>8xxx</td>
<td>change the value of the PC to xxx if ACC &gt; 0</td>
</tr>
<tr>
<td>9xxx</td>
<td>change the value of the PC to xxx if ACC &lt; 0</td>
</tr>
<tr>
<td>Axxx</td>
<td>add the value in memory location xxx to the ACC</td>
</tr>
</tbody>
</table>
4. (6 points) Find the simplest equivalent expression for \((A' + BD' + AD)(B + C')(A + B'C)\) using Boolean algebra alone.

\[
(A' + BD' + AD)(B + C')(A + B'C) = (A' + BD' + D)(AB + AC') \\
= (A' + B + D)(AB + AC') \\
= (AB + ABC' + ABD + AC'D) \\
= A(B + BC' + BD + C'D) \\
= A(B + C'D)
\]

5. (6 points) Simplify the function \(F(A,B,C,D) = \Sigma m(0,2,5,7,8,9,12,13,15)\) using a 4 variable Karnaugh map.

\[
\begin{array}{cccc}
00 & 01 & 11 & 10 \\
00 & 1 & 0 & 0 & 1 \\
01 & 0 & 1 & 1 & 0 \\
11 & 1 & 1 & 1 & 0 \\
10 & 1 & 1 & 0 & 0 \\
\end{array}
\]

\[
AC' + BD + A'B'D'
\]
6. (10 points) Draw a logic diagram that directly corresponds to the VHDL module shown below. Use simple gates only.

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity foo is port (
    A: in std_logic;
    B: in std_logic_vector(3 downto 0);
    X: out std_logic_vector(2 downto 0));
end foo;

architecture a1 of foo is
    signal Z: std_logic_vector(2 downto 0);
begin
    process(A, B, Z) begin
        Z(0) <= A xor B(0);
        for i in 1 to 2 loop
            Z(i) <= B(i) and Z(i-1);
        end loop;
    end process;
    X(2) <= Z(0);
    X(1 downto 0) <= Z(2 downto 1);
end a1;
```

```
A
B(0)   X(2)
      /
    /
B(1) --- X(0)
      \
    /
B(2) --- X(1)
```
7. (15 points) Give an expression for output $S_i$ of an $n$ input increment circuit. Express $S_i$ as a function of the carry input $C_{\text{in}}$ and data inputs $A_0, A_1, \ldots, A_i$.

$$S_i = A_i \oplus (C_{\text{in}}A_0A_1 \ldots A_{i-1})$$

Show that the circuit below implements a 4 bit increment function.

![Logic Diagram]

*First, note that the NOR gates can be rewritten as AND gates with inverted inputs. With this understanding, we can derive the following equations by inspection of the logic diagram.*

$$S_0 = A_0 \oplus C_{\text{in}}$$
$$S_1 = A'_1 \oplus (C_{\text{in}}A_0)' = A_0 \oplus (C_{\text{in}}A_0)$$
$$S_2 = A_2 \oplus (C_{\text{in}}A_0A_1)$$
$$S_3 = A'_3 \oplus (C_{\text{in}}A_0A_1A_2)' = A_3 \oplus (C_{\text{in}}A_0A_1A_2)$$

If a CMOS NAND or NOR gate has a delay of 2 ns and an inverter has a delay of 1 ns, how much faster is a 64 bit version of this circuit than the ripple-carry circuit discussed in class? Explain.

*The carry chain substitutes a NAND or NOR in each stage for an AND (which requires a NAND and an inverter), so the circuit saves an inverter delay in each stage, so the delay of the carry chain is about 64 ns faster than the original circuit.*
8. (8 points) For the sequential circuit shown below, complete the timing diagram to show when each of the two inputs must be stable. Label the start and end of each stable period relative to the rising clock edge (so write −3 to indicate 3 ns before the clock edge, for example). Assume that the flip flop setup time is 1.5 ns, the hold time is 1 ns and that all gates have delays that vary between .3 ns and 1.2 ns.

9. (10 points) A state table for a sequential circuit is shown below. Is this a Mealy model circuit, or a Moore model circuit? Draw a state diagram that corresponds to the state table.

This is a Moore model circuit. The state diagram is shown at right.