1. (12 points) The expression \((A + D)(B' + C)(C' + D')\) has the minterm \(AB'CD'\) and the maxterm \((A+B'+C+D')\). List three more of its minterms and three more of its maxterms.

\[
\begin{array}{c|c}
ABCD & F \\
0000 & 0 \\
0001 & 1 \\
0010 & 0 \\
0011 & 0 \\
0100 & 0 \\
0101 & 0 \\
0110 & 0 \\
0111 & 0 \\
1000 & 1 \\
1001 & 1 \\
1010 & 1 \\
1011 & 0 \\
1100 & 0 \\
1101 & 0 \\
1110 & 1 \\
1111 & 0 \\
\end{array}
\]

Three more minterms are \(A'B'C'D, AB'C'D', A'B'C'D\).

Three more maxterms are \((A+B+C+D)(A+B+C'+D), (A+B+C'+D')\).
2. (12 points) Use Karnaugh maps to derive a minimal sum of products expression and a minimal product of sums expression for the function. \( F(A,B,C,D) = \Sigma m(4,5,7,12,15) \), \( d(A,B,C,D) = \Sigma m(0,3,11,13,14) \). Make full use of the don’t care conditions.

The minimal sum of products expression is \( BC' + BD \).

The minimal product of sums expression is \( (B' + CD')' = B'(C + D) \).

How many gates are needed to implement the simpler of the two expressions you derived?

One AND gate, one OR gate and one inverter.

If CMOS gates are used, how many transistors do they contain?

14
3. (15 points) Draw a logic diagram that directly corresponds to the VHDL module shown below. Every signal that appears in the VHDL should be labeled in your logic diagram. Your logic diagram may include gates, flip flops and multiplexors.

```vhdl
entity foo is port (
    A, B: in std_logic;
    X: out std_logic);
end foo;
architecture a1 of foo is
begin
    process(clk) begin
        if clk'event and clk = '1' then
            if reset = '1' then
                s0 <= '0'; s1 <= '0';
            elsif s0 = '1' and A > B then
                s1 <= not s1;
            elsif s0 /= s1 then
                s0 <= B;
            end if;
        end if;
        X <= s1 and A;
    end process;
end a1;
```

![Logic Diagram](image-url)
4. (10 points) The output of the combinational circuit shown below is 0 up until the first 01 transition in the input, then it becomes 1. So for example, if the input is 10101, then the output is 00111.

What's the worst-case propagation delay for a 32 bit version of this circuit (assume each gate has a delay of 1 ns)?

*The worst-case propagation delay for the 32 bit version is 33 ns.*

Draw a lookahead version of this circuit (the 5 bit version) using simple gates only.

What is the worst-case propagation delay of a 32 bit version of this design?

*7 ns*
5. (10 points) Is the sequential circuit shown below, subject to internal hold time violations? If so, show how you would correct them (be specific). Assume that the flip flop setup time is 1.5 ns, the hold time is .4 ns and the flip flop propagation delay ranges from .6 to 2.5 ns. Also, assume that the gate delay ranges from .3 to 1.2 ns and that the maximum clock skew is 1 ns.

The circuit is subject to internal hold time violations on the path from the output of the top flip flop to the input of the bottom one. It can be corrected by inserting a pair of inverters in this path, just before the OR gate.

What is smallest clock period for which this circuit has no internal setup time violations?

$2.5 + 3(1.2) + 1.5 + 1 = 8.6$ ns
6. (10 points). Is the state table shown below for a Mealy model sequential circuit, or a Moore model circuit? Construct a state transition diagram corresponding to the state table.

Mealy.

<table>
<thead>
<tr>
<th>$S_1S_0 AB$</th>
<th>$XY$</th>
<th>$D_1D_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 00</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>00 01</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>00 10</td>
<td>11</td>
<td>01</td>
</tr>
<tr>
<td>00 11</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>01 00</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>01 01</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>01 10</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>01 11</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>10 00</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>10 01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10 10</td>
<td>11</td>
<td>00</td>
</tr>
<tr>
<td>10 11</td>
<td>01</td>
<td>10</td>
</tr>
</tbody>
</table>

![State Transition Diagram](image)
7. (15 points) The state diagram shown below is for a circuit containing a four bit counter called count, two four bit inputs A and B and a four bit output X, which is equal to A+B when the value of count is even, and A−B when the value of count is odd.

The circuit also has a reset signal which has been omitted from the diagram. When reset is high, the circuit should go to the single state and count should be set to 4. Complete the VHDL module outlined on the next page so that it implements the sequential circuit specified by the state diagram. Include code for the reset.

![State Diagram](image-url)
entity ballgame is Port ( 
    clk, reset: in std_logic;
    A, B : in std_logic_vector(3 downto 0);
    X : out std_logic_vector(3 downto 0));
end ballgame;

architecture a1 of ballgame is

    type state_type is (single, double, homer);
    signal state: state_type;
    signal count: std_logic_vector(3 downto 0);

begin

    process (clk) begin
        if clk'event and clk = '1' then
            if reset = '1' then
                state <= single; count <= x"4";
            else
                if state = single and B < count then
                    state <= double; count <= A;
                elsif state = single and B = count then
                    state <= homer; count <= x"0";
                elsif state = homer and B > A then
                    state <= double; count <= count + 1;
                elsif state = double and A /= B then
                    state <= single;
                elsif state = double then
                    count <= count - '1';
                end if;
            end if;
        end if;
    end process;

    X <= A+B when count(0) = '0' else A-B;
end a1;
8. (15 points) The VHDL module defined below implements a circuit that counts the number of *stable periods* on the data input *Din* where a stable period is a period of 4 or more consecutive clock ticks during which the value of *Din* does not change. The output *nStable* is equal to the number of stable periods observed. So for example, with an input of 001011110010000011011100 the final value of *nStable* should be 2. Complete the block diagram at the bottom of the page so it implements the same circuit.

```vhdl
entity filter is
    Port(
        clk, reset: in std_logic;
        Din : in std_logic;
        nStable : out std_logic_vector(15 downto 0);
    );
end filter;
architecture a1 of filter is
    signal count: std_logic_vector(2 downto 0);
    signal stableCount: std_logic_vector(15 downto 0);
    signal prevDin: std_logic;
    begin
        process (clk) begin
            if clk'event and clk = '1' then
                prevDin <= Din;
                if reset = '1' then
                    count <= (count'range =>'0');
                    stableCount <= (stableCount'range => '0');
                else
                    if count = 3 then
                        stableCount <= stableCount + 1;
                    end if;
                    if Din /= prevDin then
                        count <= (count'range =>'0');
                    elsif count <= 3 then
                        count <= count + 1;
                    end if;
                end if;
            end if;
        end process;
        nStable <= stableCount;
    end a1;
```

---

**Block Diagram:**

- **Inputs:**
  - *clk*
  - *reset*
  - *Din*

- **Outputs:**
  - *nStable*

- **Internal Signals:**
  - *count*: 3-bit register
  - *stableCount*: 16-bit register
  - *prevDin*

- **Logic Operations:**
  - Comparisons
  - Incrementation
  - Logic Gates

---

**Notes:**

- The block diagram should reflect the functionality described in the VHDL code.
- The diagram should show how the signals interact to count stable periods.
- The diagram should include all necessary logic gates and registers to implement the circuit.
- Ensure all inputs and outputs are correctly connected.

9. (12 points) The simulation output shows a program executing on the simple processor. Fill in the blanks below with the information that belongs in each of the labeled blank areas in the simulation output. Note that 32_{10} = 20_{16}.

A. 3024  B. 0008  C. a022
D. 000c  E. 0022  F. 0009

0000 halt execution
0001 \text{ACC} := \text{--ACC} \hspace{1cm} 5xxx \quad M[M[0xxx]] := \text{ACC}
1xxx \text{if sign bit of xxx is 0 then} \hspace{1cm} 6xxx \quad \text{PC} := 0xxx
\quad \text{ACC} := 0xxx \text{ else } \text{ACC} := fxxx \hspace{1cm} 7xxx \quad \text{if } \text{ACC} = 0 \text{ then } \text{PC} := 0xxx
2xxx \text{ACC} := M[0xxx] \hspace{1cm} 8xxx \quad \text{if } \text{ACC} > 0 \text{ then } \text{PC} := 0xxx
3xxx \text{ACC} := M[M[0xxx]] \hspace{1cm} 9xxx \quad \text{if } \text{ACC} < 0 \text{ then } \text{PC} := 0xxx
4xxx M[0xxx] := \text{ACC} \hspace{1cm} axxx \quad \text{ACC} := \text{ACC} + M[0xxx]
10. (12 points) The table shown below represents a 2-way set associative cache (combined instruction and data cache). The ‘<’ and ‘>’ symbols in the center point toward the least recently used of the two entries in a given row. So for example, the right entry in row 4 has been used more recently than the left entry.

<table>
<thead>
<tr>
<th>tag</th>
<th>data</th>
<th>tag</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0a4</td>
<td>&lt;</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>b23</td>
<td>&lt;</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0a4</td>
<td>&gt;</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>20c</td>
<td>&lt;</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>b4c</td>
<td>&gt;</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>302 b4a</td>
<td>&gt; &lt;</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>a01d</td>
<td>&lt; &gt;</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>b4a</td>
<td>&lt;</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>b23</td>
<td>&gt;</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>acb</td>
<td>&lt;</td>
<td>9</td>
</tr>
<tr>
<td>a</td>
<td>90c</td>
<td>&gt;</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>03f</td>
<td>&lt;</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>0a3</td>
<td>&gt;</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>0c2</td>
<td>&lt;</td>
<td>d</td>
</tr>
<tr>
<td>e</td>
<td>20c</td>
<td>&lt;</td>
<td>e</td>
</tr>
<tr>
<td>f</td>
<td>0a4</td>
<td>&gt;</td>
<td>f</td>
</tr>
</tbody>
</table>

Show how the state of the cache changes, when the processor fetches and executes the two instructions shown below, assuming that a new value replaces the least recently used entry. Assume that cache entries are selected using the low order 4 bits of the memory address and that the tag consists of the high order 12 bits. Be sure to update cache contents, include the ‘<’ and ‘>’ symbols where appropriate.

<table>
<thead>
<tr>
<th>address</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>b4a5</td>
<td>2432</td>
</tr>
<tr>
<td>b4a6</td>
<td>95cd</td>
</tr>
</tbody>
</table>

What is the value in the program counter, after the first two instructions are executed?

05cd

What is the value of the instruction register after the next instruction is fetched?

401f