Date: 2012-05-29 (5 hours)
Allowed aids: none
Responsible Teacher: Mikael Bylund, tel: 073-7272926, e-mail: mikael.bylund@miun.se

Number of tasks: 9

Number of pages: 3

Maximum points: 100 (50 points required to pass)

Instructions for submitted solutions:

- Rationale and justifications may not be so scarce that they become difficult to follow.
- The reasoning behind used equations should be explained.
- The calculations shall be sufficiently complete to show how the final result was obtained.
- Each task must be concluded with a clearly written answer.

1. **Convert the hexadecimal number DB3 into binary. Then convert the obtained binary number into octal. (8p)**

2. **Convert the decimal number -35 into an 8-bit binary number in two's complement form. (8p)**

3. **Minimize the following Boolean expression:** $(x+y+z)(x+y+\bar{z})$ **(10p)**

4. **Determine if the following equality is true**: $\overline{a+\overline{b\cdot c}}\cdot\overline{\bar{a}\cdot(b+c)}=0$ **(10p)**

5. **Minimize the function** $f(x_3,x_2,x_1,x_0)=\sum(1,6,7,13)+d(4,5,9,10,15)$ **by using a Karnaugh map. Write the function as a minimal sum-of-product (SP) expression. (10p)**

6. **Design a function that counts how many of the bits in the input-vector** $X=(x_2,x_1,x_0)$ **that are 0. The result should be given as a binary number on the output** $Z=(z_1,z_0)$**.**
   Give your answer as a Boolean expression. (12p)

7. **Write a function in VHDL that takes a 4-bit vector as input, determines the parity bit required for odd parity, and gives a 5-bit output that contains the parity bit and the original 4-bit vector. (12p)**

8. **Draw the state transition graph for the VHDL code given in Appendix 1. (14p)**

9. **Develop the circuit diagram for the state transition graph given below. Use D flip-flops and binary coding for the state memory. (16p)**

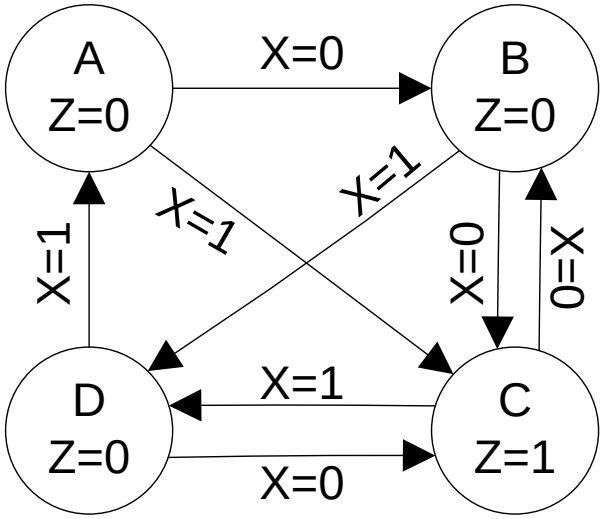

Figure 1

APPENDIX 1.

```
entity seq_design is
port(      a, clock, reset: in std_logic;
           x: out std_logic
);
end seq_design

architecture FSM of seq_design is
    type state_type is (S0, S1, S2, S3, S4);
    signal next_state, current_state: state_type;

begin
    process(clock, reset)
    begin
        if (reset='1') then
            current_state <= S0;
        elsif (clock'event and clock='1') then
            current_state <= next_state;
        end if;
    end process;

    process(current_state, a)
    begin
        case current_state is

            when S0 => x <= '0';
                       if a='0' then
                           next_state <= S4;
                       elsif a ='1' then
                           next_state <= S1;
                       end if;
            when S1 => x <= '0';
                       if a='0' then
                           next_state <= S4;
                       elsif a='1' then
                           next_state <= S2;
                       end if;
            when S2 => x <= '0';
                       if a='0' then
                           next_state <= S4;
                       elsif a='1' then
                           next_state <= S3;
                       end if;
            when S3 => x <= '1';
                       if a='0' then
                           next_state <= S0;
                       elsif a='1' then
                           next_state <= S4;
                       end if;
            when S4 => x <= '1';
                       next_state <= S0;

    end case;
  end process;
end FSM;
```