Date: 2012-11-02 (5 hours)
Allowed aids: none
Responsible Teacher: Mikael Bylund, tel: 060-148748, 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 octal number 5263 into binary. Then convert the obtained binary number into hexadecimal. (8p)**

2. **Convert the decimal number -37 into an 8-bit binary number in two's complement form. (8p)**

3. **Minimize the following Boolean expression:** $\overline{a+b}+a\cdot b+\overline{a\cdot\bar{b}+\bar{a}\cdot b}$ **(10p)**

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

5. **Minimize the function** $f(x_3,x_2,x_1,x_0)=\sum(1,2,3,7,8,12)+d(0,5,10,15)$ **by using a Karnaugh map. Write the function as a minimal sum-of-product (SP) expression. (10p)**

6. **Design a function that takes a 4-bit binary number** $X=X(x_3,x_2,x_1,x_0)$ **as input, calculates** $|X-7|$**, and gives the result as a 4-bit binary number on the output (**$|\ |$ **indicates absolute value).**
   Give your answer as Boolean expressions for the output bits. (12p)

7. **Write a function in VHDL that takes a 5-bit input vector (4 bit data + 1 bit for odd parity), and then verifies if the parity bit matches the data or not (i.e. checks if the data has been correctly transmitted). (12p)**

8. **Draw the state transition graph for the VHDL code given in Appendix 1. Is this a Moore or a Mealy machine? Explain your reasoning. (14p)**

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

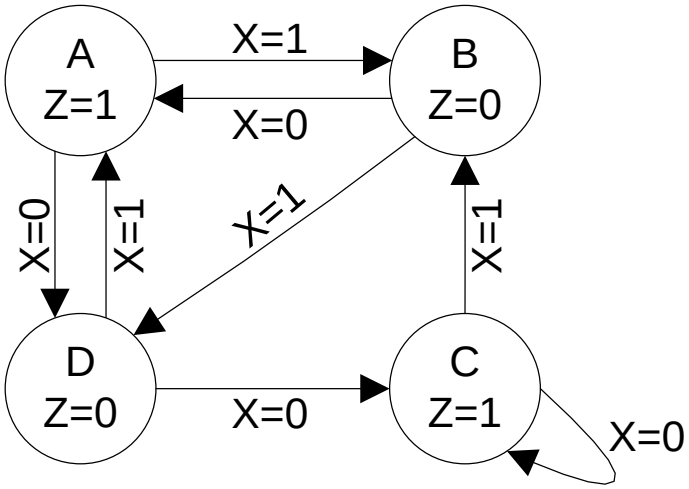

Figure 1

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);
    signal next_state, current_state: state_type;



    process(clock, reset)
    begin
      if (reset='1') then
           current_state <= S0;

      else
        case current_state is

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

      current_state <= next_state;

    end process;
end FSM;
```