<u>שפות חומרה (VHDL).</u>

המרצה: פרופסור אלי פלקסר

<u>הוראות לנבחן</u>

<u>א. משך הבחינה : 2.5 שעות.</u>

<u>ב. מבנה השאלון:</u>

שאלה מס' 1 - 50%

שאלה מס' 2 - 50%

<u>ג. חומר עזר מותר בשימוש: תקציר מצורף.</u>

<u>ד. הערות מיוחדות:</u>

- **כתוב בכתב ברור ונקי.**
- הסבר בפירוט את תכניותיך ושיקולך.
- תשובות לא מנומקות ציונן יופחת בצורה משמעותית.
- כל שאלה תתחיל בדף חדש בצד שמאל.

<u>בהצלחה !</u>

שאלה 1 (50%):

תכנן, בשפת VHDL, מכונת מצבים אלגוריתמית ASM בעלת process יחיד, המזהה בכניסה טורית (בעלת ביט אחד) את הרצף "AFEKA", בקוד ASCII באורך 40 ביט (כל תו 8 ביטים X 5 תווים = 40). לאחר הזיהוי, המכונה מוציאה בצורה סידרתית את הקוד כרצף תווי ASCII ביציאה מקבילית בעלת 8 ביטים. בסיום השיגור המכונה חוזרת למצבה הראשוני. כמו כן, המכונה מוציאה '1' לוגי ביציאה SEND רק במהלך השיגור. בזמן השידור אין קליטת תווים חדשים.

שם לב:

הרכיב העומד לרשותך הינו גדול מאד, בעל יותר מ 1000 אלמנטים לוגיים - MACROCELL או LE. אתה רשאי לבזבז חומרה ככול שתרצה.

ניתן להשתמש בתכונה (ATTRIBUTE) מסוג POS בכדי לדעת מהו הערך המיספרי של תו, לדוגמה.

N := CHARACTER'***POS*** ('A') -- The ASCII value of A.

או להשתמש לחליפין בערכים המצורפים להלן:

'A' = 16#41#; 'F' = 16#46#; 'E' = 16#45#; 'K' = 16#4B#;

א. שרטט דיאגראמת מצבים של המכונה. (10)

ב. הסבר מה קורה בכל מצב, את תנאי המעבר ואת החומרה המבוקרת. (5)

ג. כתוב את הקוד המתאים לסינטיזה Entity ו Architecture. (25)

ד. האם ניתן להמשיך לקרוא את הכניסה גם במהלך השיגור, הסבר. (10)

שאלה 2 (50%):

תכנן, בשפת VHDL, מונה עולה/יורד ברוחב 12 ביטים בעל טעינה מקבילית וכניסת ENABLE, הסופר בקפיצות של K צעדים קדימה/אחורה, בכל מחזור שעון, אם ה-ENABLE מאפשר זאת. הערך K יהיה בתחום 20-1 ויטען לאוגר המוגדר בתוך הרכיב (REGK) ע"י בקר טעינה LK, בעוד הערך למונה נטען ע"י בקר LD. ה BUS לטעינת המספר למונה ולטעינת המספר K הוא אותו BUS והעדיפות תינתן לטעינת המונה (LD).

א. כתוב entity ו architecture המתאימים לסינטיזה. השתמש בספרייה תקנית מתאימה. (35)

ב. האם ניתן לוותר על כניסת הכיוון – שקובעת האם המונה עולה או יורד, כיצד. (10)

ג. מה תהיה המחזוריות המקסימלית של מונה זה ובאיזה תנאי זה יקרה. (5)

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity XControl is port (
    clk         :   in  std_logic;
    reset       :   in  std_logic;
    SerIn       :   in  std_logic;
    StrOut      :   out unsigned(7 downto 0);
    Z           :   out std_logic);
end XControl;

architecture Flaxer of XControl is
TYPE   StatusType   IS (S0, S1, S2, S3, S4, S5);
-----------------------------------------------------------------
constant AFEKA          :string(4 downto 0):= "AFEKA";
--constant MyString :unsigned(39 downto 0) := X"41"&X"46"&X"45"&X"4B"&X"41";
signal MyString         :unsigned(39 downto 0);
signal XState           :StatusType;
signal SampReg          :unsigned(39 downto 0);
begin
-----------------------------------------------------------------
initProc:process(MyString)
begin
    for k in 4 downto 0 loop
        MyString(k*8+7 downto k*8) <= To_Unsigned(character'POS(AFEKA(k)), 8);
    end loop;
end process initProc;
-----------------------------------------------------------------
Status: process(reset, clk)
    begin
      IF reset = '1' THEN
            XState <= S0;
            SampReg <= (others => '0');
            StrOut  <= (others => '0');
            Z <= '0';
      ELSIF (clk'event and clk = '1') THEN
            CASE XState IS
                WHEN S0 =>
                    IF SampReg = MyString THEN
                        XState <= S1;
                    ELSE
                        XState <= S0;
                        SampReg <= SampReg(38 downto 0) & SerIn;
                    END IF;
                    Z <= '0';
                WHEN S1 =>
                    XState <= S2;
                    StrOut <= SampReg(39 downto 32);
                    Z <= '1';
                WHEN S2 =>
                    XState <= S3;
                    StrOut <= SampReg(31 downto 24);
                    Z <= '1';
                WHEN S3 =>
                    XState <= S4;
                    StrOut <= SampReg(23 downto 16);
                    Z <= '1';
                WHEN S4 =>
                    XState <= S5;
                    StrOut <= SampReg(15 downto 8);
                    Z <= '1';
                WHEN S5 =>
                    XState <= S0;
                    StrOut <= SampReg(7 downto 0);
                    Z <= '1';
                    SampReg <= (others => '0');
            END CASE;
      END IF;
    end process;
-----------------------------------------------------------------
end Flaxer;

```

```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3 use ieee.numeric_std.all;
 4
 5 entity XControl is port (
 6     clk         :   in  std_logic;
 7     reset       :   in  std_logic;
 8     SerIn       :   in  std_logic;
 9     StrOut      :   out unsigned(7 downto 0);
10     Z           :   out std_logic);
11 end XControl;
12
13 architecture Flaxer of XControl is
14 TYPE   StatusType   IS (S0, S1);
15 ---------------------------------------------------------------------
16 constant AFEKA          :string(4 downto 0):= "AFEKA";
17 --constant MyString :unsigned(39 downto 0) := X"41"&X"46"&X"45"&X"4B"&X"41";
18 signal MyString         :unsigned(39 downto 0);
19 signal XState           :StatusType;
20 signal SampReg          :unsigned(39 downto 0);
21 begin
22 ---------------------------------------------------------------------
23 initProc:process(MyString)
24 begin
25     for k in 4 downto 0 loop
26         MyString(k*8+7 downto k*8) <= To_Unsigned(character'POS(AFEKA(k)), 8);
27     end loop;
28 end process initProc;
29 ---------------------------------------------------------------------
30 Status: process(reset, clk)
31     begin
32       IF reset = '1' THEN
33             XState <= S0;
34             SampReg <= (others => '0');
35             StrOut  <= (others => '0');
36             Z <= '0';
37       ELSIF (clk'event and clk = '1') THEN
38             CASE XState IS
39
40                 WHEN S0 =>
41                     IF SampReg = MyString THEN
42                         XState <= S1;
43                         Z <= '1';
44                     ELSE
45                         XState <= S0;
46                         SampReg <= SampReg(38 downto 0) & SerIn;
47                         Z <= '0';
48                     END IF;
49                 WHEN S1 =>
50                     IF SampReg = X"00000" THEN
51                         XState <= S0;
52                         Z <= '0';
53                     ELSE
54                         XState <= S1;
55                         StrOut <= SampReg(39 downto 32);
56                         SampReg <= SampReg(31 downto 0) & X"00";
57                         Z <= '1';
58                     END IF;
59             END CASE;
60       END IF;
61     end process;
62 -----------------------------------------------------------------------
63 end Flaxer;
64
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity ldcnt is port (
    clk             :   in std_logic;
    ena             :   in std_logic;
    ld, la          :   in std_logic;
    up_dw           :   in std_logic;
    count_io        :   inout std_logic_vector(7 downto 0));
end ldcnt;

architecture archldcnt of ldcnt is

SUBTYPE CunterType IS std_logic_vector(7 downto 0);
SUBTYPE DivType    IS std_logic_vector(3 downto 0);
-----------------------------------------------------------------
signal counter  :CunterType;
signal divider  :DivType;

begin
    process (clk)
    begin
        IF (clk'event and clk='1') THEN
            IF ld = '1' THEN
                counter <= count_io;
            ELSIF la = '1' THEN
                divider <= count_io(DivType'RANGE);
            ELSIF ena = '1' THEN
                IF up_dw = '1' THEN
                    counter <= counter + divider;
                ELSE
                    counter <= counter - divider;
                END IF;
            END IF;
        END IF;
    END process;

    count_io <= counter WHEN (ld = '0' and la = '0') ELSE (OTHERS => 'Z') ;
end archldcnt;


```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity ldcnt is port (
    clk             :   in std_logic;
    ena             :   in std_logic;
    ld, la          :   in std_logic;
    up_dw           :   in std_logic;
    count_in        :   in std_logic_vector(7 downto 0);
    counter         :   buffer std_logic_vector(7 downto 0));
end ldcnt;

architecture archldcnt of ldcnt is

------------------------------------------------------------------
signal divider  :std_logic_vector(3 downto 0);

begin
    process (clk)
    begin
        IF (clk'event and clk='1') THEN
            IF ld = '1' THEN
                counter <= count_in;
            ELSIF la = '1' THEN
                divider <= count_in(3 downto 0);
            ELSIF ena = '1' THEN
                IF up_dw = '1' THEN
                    counter <= counter + divider;
                ELSE
                    counter <= counter - divider;
                END IF;
            END IF;
        END IF;
    END process;
end archldcnt;


```