הטכניון - מכון טכנולוגי לישראל
הפקולטה להנדסת חשמל

מעבדה 1

ניסוי VHDL1
דוח הכנה למעבדה

גרסה 1.71

קיץ 2018

עורכים: דודי בר-און, אברהם קפלן , ליאת שורץ
על פי חוברות של עמוס זסלבסקי

תאריך הגשת דו"ח ההכנה	
שם המדריך	

סטודנט	שם פרטי	שם משפחה
1	ברק	זן
2	בועז	טייטלר

תוכן עניינים

הערות:

1. בכתיבת הקוד חובה להשתמש בשמות ה- entities, הכניסות והיציאות המופיעים בהגדרת התרגילים.
2. יש לתת שמות קבצים ותיקיות באנגלית בלבד וללא רווחים
3. שם הקובץ צריך להיות כשם ה-Entity
4. בכל פעם יש להגדיר את הקובץ כ- TOP
5. **יש לקמפל את הקוד** בקווארטוס, לראות שאין שגיאות סינטקס
6. בסוף התהליך יש להעתיק את הקוד בצורה **קריאה** מה NOTEPAD++ לקובץ התשובות

1 תרגיל תכנון MUX בשיטות שונות

1.1 מימוש Selected Assignment_MUX

כתוב קוד בשם mux_SA. המערכת הינה Multiplexer בעל ממדים 3=>1. כניסות המידע ind, כניסות בחירה - sel ויציאת המידע – outd.

כתוב את הקוד שמתאר את הרכיב באמצעות **התניית** Selected Assignment.

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_signed.all ;

--mux_SA 3->1, in: ind, sel. out: outd
entity mux_SA is
 port (ind : in bit_vector(2 downto 0 ) ;
       sel : in bit_vector(1 downto 0) ;
       outd : out bit ) ;
 end mux_SA ;

architecture arc_mux_SA of mux_SA is
   begin
      with sel select
      outd <= ind(0) when "00"    ,
              ind(1) when "01"    ,
              ind(2) when others ;
   end arc_mux_SA ;
```

1.2 מימוש **Conditional Assignment MUX**

כתוב קוד בשם mux_CA. המערכת הינה Multiplexer בעל ממדים 3=>1. כניסות המידע ind, כניסות בחירה - sel ויציאת המידע – outd.

כתוב את הקוד שמתאר את הרכיב באמצעות **התניית** Conditional Assignment.

```
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_signed.all ;

--mux_CA 3->1, in: ind, sel. out: outd
entity mux_CA is
 port (ind : in bit_vector(2 downto 0 ) ;
       sel : in bit_vector(1 downto 0) ;
       outd : out bit ) ;
 end mux_CA ;

architecture arc_mux_CA of mux_CA is
   begin
      outd <= ind(0) when (sel="00") else ind(1) when (sel="01") else ind(2) ;
   end arc_mux_CA ;
```

2 תרגיל תכנון MUX הירארכי

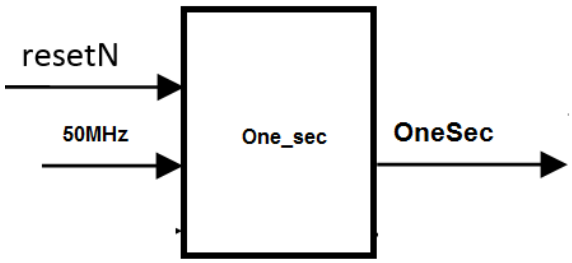
כתוב קוד בשם : mux9to1. שיממש Multiplexer בעל 9 כניסות מידע ind, 4 כניסות בחירה sel ויציאה אחת outd. המערכת מורכבת מ 4 רכיבי Multiplexer מהתרגיל הקודם בעלי ממדים 3>1, אותות הכניסות הן מסוג std_logic_vector

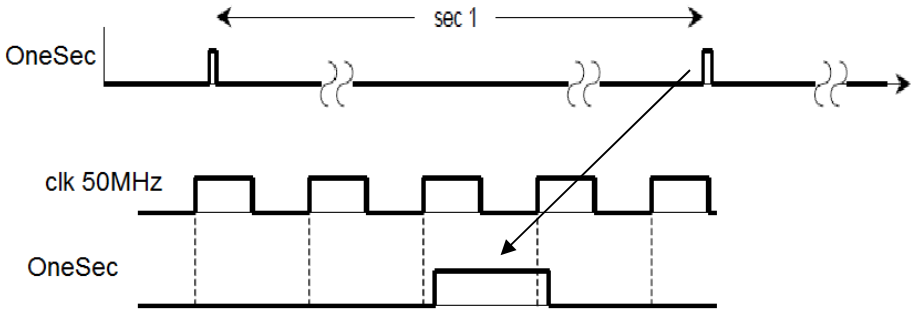
ממש תכן הירארכי בVHDL השתמש בקוד מהתרגיל הקודם

```
entity mux9to1 is
 port (ind : in std_logic_vector(8 downto 0) ;
       sel0 : in std_logic_vector(1 downto 0) ;
       sel1 : in std_logic_vector(1 downto 0) ;
       sel2 : in std_logic_vector(1 downto 0) ;
       selM : in std_logic_vector(1 downto 0) ;
       outd : out std_logic ) ;
  end mux9to1 ;

architecture arc_mux9to1 of mux9to1 is
       component mux_CA is
          port(ind : in std_logic_vector(2 downto 0 ) ;
               sel : in std_logic_vector(1 downto 0) ;
              outd : out std_logic ) ;
       end component ;

 signal d0, d1, d2 : std_logic ;

  begin
       u0: mux_CA port map (ind=>ind(8 downto 6),sel=>sel0,outd=>d0) ;
       u1: mux_CA port map (ind=>ind(5 downto 3),sel=>sel0,outd=>d1) ;
       u2: mux_CA port map (ind=>ind(2 downto 0),sel=>sel0,outd=>d2) ;
       u3: mux_CA port map (ind=>(d0, d1, d2) ,sel=>selM,outd=>outd) ;
  end arc_mux9to1 ;
```

3 מחלק תדר

כתוב מחלק תדר המקבל שעון של 50MHz ומוציא פולס שעון צר בתדר של 1Hz בשיטה סינכרונית. רוחב הפולס הצר יהיה כזמן מחזור יחיד של השעון המהיר .
כמתואר באיור הבא :





העזר בקוד הבא :
יש להקפיד לא להשתמש ב MOD וחילוק שהם יקרים למימוש

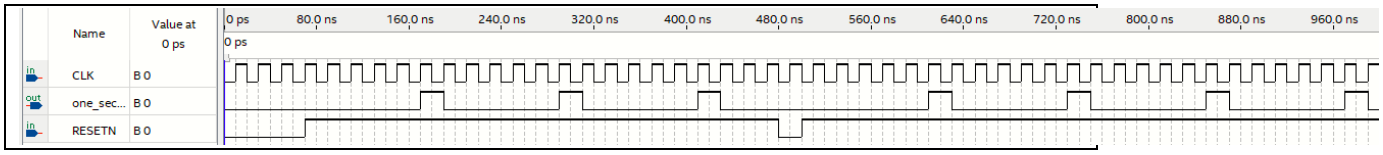
```
signal one_sec_flag : std_logic ;

process(CLK,RESETN)
    variable one_sec: integer ;
--  constant sec: integer := 50000000 ; -- for Real operation
    constant sec: integer := 5 ; -- for simulation
begin
    if RESETN = '0' then
        one_sec := 0 ;
        one_sec_flag <= '0' ;
    elsif rising_edge(CLK) then
        one_sec := one_sec + 1 ;
        if (one_sec  **fill) then
            one_sec_flag **fill ;
            one_sec **fill;
        else
            one_sec_flag **fill;
        end if;
    end if;
end process;

end Counter_arch;
```

```
-- Frequency divider
library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_signed.all ;

entity frequency_divider is
    port ( CLK,RESETN : in std_logic ;
           one_sec_flag : out std_logic ) ;
end frequency_divider ;


architecture arc_frequency_divider of frequency_divider is
begin
 process(CLK,RESETN)
    variable one_sec: integer ;
--  constant sec: integer := 50000000 ; -- for Real operation
    constant sec: integer := 5 ; -- for simulation
 begin
    if RESETN = '0' then
        one_sec := 0 ;
        one_sec_flag <= '0' ;
    elsif rising_edge(CLK) then
        one_sec := one_sec + 1 ;
        if (one_sec > sec) then
            one_sec_flag <= '1' ;
            one_sec := 0;
        else
            one_sec_flag <= '0';
        end if;
    end if;
 end process;

end arc_frequency_divider;
```

תכנן ופרט להלן מה תרצה לבדוק בסימולציה, אילו אותות ומצבים (מומלץ לבדוק מצב עבודה רגיל ומקרי קצה, כמו מצב RESET, סיום ספירה והתחלתה, וכו').

נרצה לבדוק שהמחלק תדר סופר רק כאשר RESETN ב 1, בדיקה של כמה מחזורים ברצף ובדיקה שהספירה מתאפסת כאשר RESETN עולה ומתחילה מההתחלה כשהוא יורד

צור קובץ WAVEFORM והרץ סימולציה של המעגל (שנה את הקבוע SEC לזמן סימולציה קצר סביר)

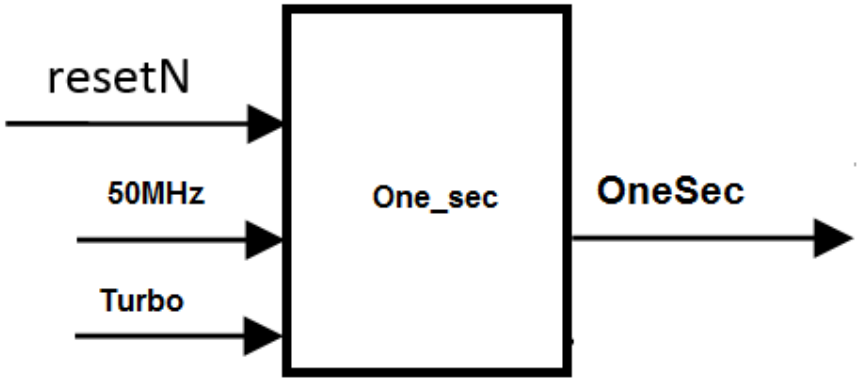


דד

4 מחלק תדר- כניסת TURBO

הדגמת רכיב על הכרטיס עם שעון שמתחלף פעם בשניה היא איטית ומייגעת ולכן מומלץ להוסיף כניסת טורבו שמאיצה את תהליך ההדגמה פי -10.

לשם כך הוסף למחלק התדר כניסת טורבו : כשהיא ב-1 תדר הפולס הופך ל- 10 Hz וב-0 ללא שינוי. הקפד על קוד יעיל וסינכרוני.

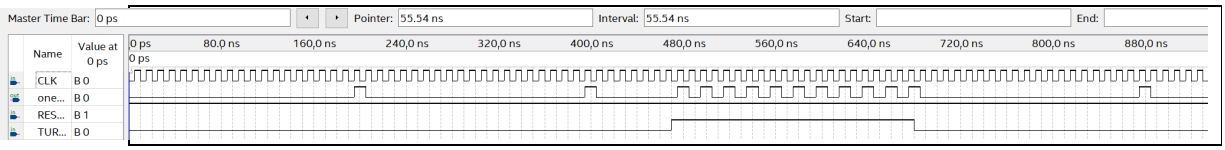



```
entity turbo_freq_divider is
    port ( CLK,RESETN : in std_logic ;
           TURBO : in std_logic;
        one_sec_flag : out std_logic ) ;
end turbo_freq_divider  ;

architecture arc_turbo_freq_divider of turbo_freq_divider is
begin
 process(CLK,RESETN)
    variable one_sec: integer ;
--  constant sec: integer := 50000000 ; -- for Real operation
    constant sec: integer := 20 ; -- for simulation
    variable max : integer;
 begin
    if RESETN = '0' then
        one_sec := 0 ;
        one_sec_flag <= '0' ;
    elsif rising_edge(CLK) then
        one_sec := one_sec + 1 ;

        if (TURBO = '1') then
            max := sec / 10;
        else
            max := sec;
        end if;

        if (one_sec >= max) then
            one_sec_flag <= '1' ;
            one_sec := 0;
        else
            one_sec_flag <= '0';
        end if;
    end if;
 end process;

end arc_turbo_freq_divider ;
```

הגדר מה תרצה לראות בסימולציה

נצפה לראות כי כאשר כניסת Turbo דולקת, התדר מהיר פי 10 ביחס למתי שכניסת ה-Turbo כבוייה.

הרץ סימולציה של המעגל בקווארטוס במצב Functional העזר ב COOK-BOOK
(שנה את הקבוע SEC לזמן סימולציה קצר סביר)
שנה את כניסת טורבו **<u>במהלך הריצה</u>** לשני הכיוונים וראה שהמונה אינו מתבדר
יש לבדוק גם את המצב שבו TURBO משתנה כשערך המונה גדול מה SEC החדש



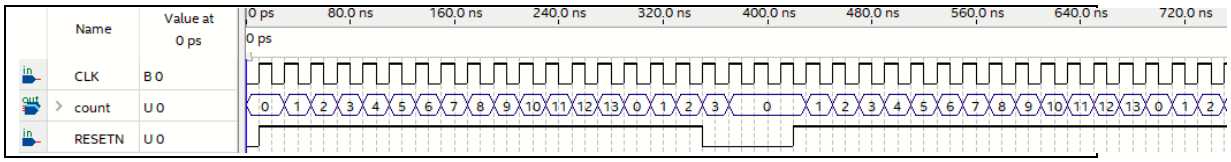
5 מונה עולה פשוט

כתוב יישות (entity) של מונה בינארי סינכרוני עולה ,שהוא בעל מחזור של 14. זאת אומרת המונה מתחיל לספור מ- 0, סופר עד 13, חוזר ל- 0 ושוב סופר עד 13, וכן הלאה בצורה מחזורית.
למונה כניסות : שעון ו- resetN שמאפס את יציאת המונה
ויציאות : count – וקטור של 4 ביט המראה את מצב הספירה בכל פולס שעון.

העתק את הקוד שכתבת כאן.

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

entity counter is
    port ( CLK,RESETN : in std_logic ;
           count : out std_logic_vector(3 downto 0) ) ;
end counter ;


architecture arc_counter of counter is

 signal tick_count : std_logic_vector(3 downto 0) ;

begin
   process(CLK,RESETN)
      begin
      if rising_edge(CLK) then
         if (RESETN = '0' or unsigned(tick_count) = 13) then
            tick_count <= (others => '0');
         else
            tick_count <= std_logic_vector(unsigned(tick_count) + 1);
         end if;
      end if;
   end process;

   count <= tick_count ;

end arc_counter;
```

צור קובץ WAVEFORM והרץ סימולציה של המעגל והוסף את תוצאות הסימולציה לדו״ח.



6 תצוגת 7Segment עם הדלקה מלאה וכיבוי

באחד הניסויים הקודמים ראינו שקיימים ב - Quartus רכיבים מוכנים שממירים מקוד BCD לתצוגות Seven Segment (רכיבים תואמי TTL שנקראו 7447 ו 7448). רכיבים אלו היו מאוד מוגבלים, היות ולא יכולנו להמיר באמצעותם צירופי כניסה שהם גדולים מ:
$(1001)_2 = (9)_{10}$

בשאלת תכן זו יהיה עליך לתכנן ממיר צירופי פשוט מקוד בינארי ברוחב ארבע סיביות + 2 סיביות נוספות, לתצוגת Seven Segment עבור כל 16 הצירופים האפשריים של 4 סיביות. תפקיד הסיביות הנוספות יוסבר בהמשך.

לרכיב זה קרא בשם HEXSS והוא יהיה שימושי עבורנו בניסויים שבהמשך.
לתצוגות Seven Segment יש את המבנה המרחבי הבא:

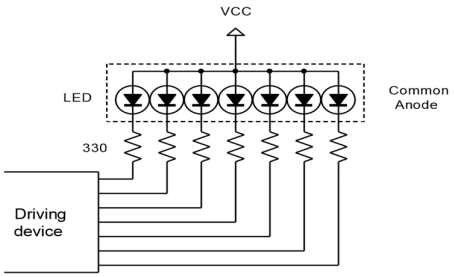
שבעת המקטעים נקראים בדרך כלל בשמות של הספרות 0 – 6 (או האותיות a עד g). לעתים קיימת בצד שמאל או בצד ימין של המבנה הנ״ל גם תצוגה קטנה נוספת שנקראת dp ושמשמשת כנקודה עשרונית (Decimal Point). נקודות dp קיימת בתצוגות שבלוח DE10 אך היא לא מחוברת לרכיב Cyclone.

במערכת שתתכנן השתמש בווקטור באורך 7 עבור היציאות ל 7Segment. קטע הקוד הבא מתאר חלק מהדקי הרכיב שעליך לתכנן עבור תצוגה של ספרה אחת:

```
-- HEX to Seven-Segment (active low) - vector out
library ieee ;
use ieee.std_logic_1164.all ;

entity hexss is
port ( din : in  std_logic_vector(3 downto 0) ;
       ss  : out std_logic_vector(6 downto 0) );
  --    ADD additional signals
```

```
end hexss ;
```

כזכור לך מהניסוי הראשון, מבחינה חשמלית קיימים שני סוגים של תצוגות. סוג אחד נקרא תצוגת מסוג Common Anode (CA) והיא מחוברת באופן הבא:



לצורת פעולה כזו קוראים לעתים בשם Active Low. התצוגות בלוח התרגול DE10 הן מסוג CA והן כולן פועלות בנמוך (Active Low).

טבלת האמת הבאה מתארת את הלוגיקה הרצויה של רכיב ההמרה מספרה הקסדצימלית (ארבע סיביות) לספרת Seven-Segment שפועלת ב– Active Low:

Din(HEX)	Din(bin)	6	5	4	3	2	1	0	pattern
0	0000	1	0	0	0	0	0	0	0
1	0001	1	1	1	1	0	0	1	1
2	0010	0	1	0	0	1	0	0	2
3	0011	0	1	1	0	0	0	0	3
4	0100	0	0	1	1	0	0	1	4
5	0101	0	0	1	0	0	1	0	5
6	0110	0	0	0	0	0	1	0	6
7	0111	1	1	1	1	0	0	0	7
8	1000	0	0	0	0	0	0	0	8
9	1001	0	0	1	0	0	0	0	9
a	1010	0	0	0	1	0	0	0	A
b	1011	0	0	0	0	0	1	1	b
c	1100	1	0	0	0	1	1	0	C
d	1101	0	1	0	0	0	0	1	d
e	1110	0	0	0	0	1	1	0	E
f	1111	0	0	0	1	1	1	0	F

כתוב קובץ VHDL בשם HEXSS.VHD שמקבל 4 + 2 סיביות ומציג ספרה הקסדצימלית אחת כמתואר לעיל.

תפקידן של 2 סיביות הכניסה הנוספות הוא:

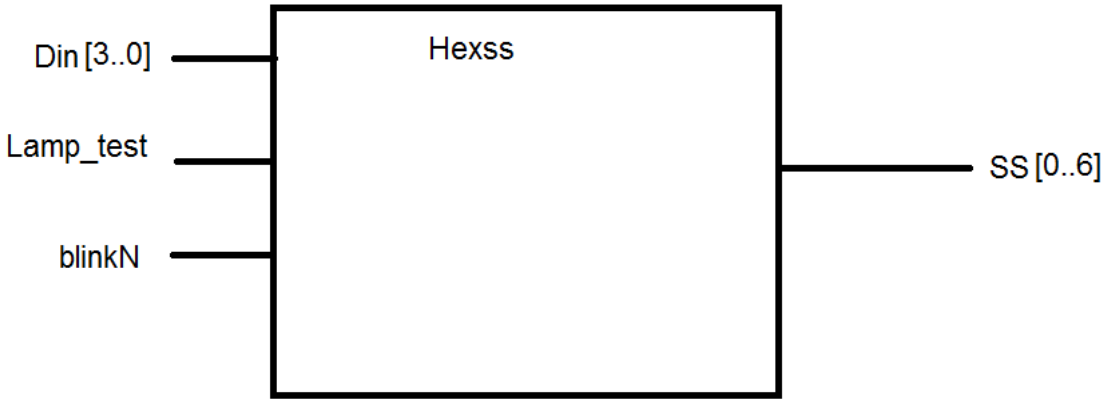
- במקרה של blinkN=0- יש להחשיך את הספרה " " -ללא תלות בכניסה
- במקרה של LAMP_TEST=1 יש להדליק את כל הסגמנטים –"8" ללא תלות בכניסה

- אם שניהם מאופשרים יש להדליק את התצוגה

הקפד על כתיבת קוד צירופי חוקי שיעבור קומפילציה.

הערה: מותר להיעזר בקוד HEXSS שקיבלת במעבדה סכמתי1 /ספרתי 1 בתור דוגמה

סרטט **בעפרון** סמל גרפי של הרכיב (כניסות ויציאות)



קוד VHDL בעמוד הבא

```vhdl
-- HEX to Seven-Segment (active low) - vector out
library ieee ;
use ieee.std_logic_1164.all ;

entity hexss is
port ( din : in  std_logic_vector(3 downto 0) ;
       blinkN : in bit ;
       lamp_test : in bit;
       ss  : out std_logic_vector(6 downto 0) );
  --    ADD additional signals
end hexss ;

architecture arc_hexss of hexss is
begin
      process(blinkN, lamp_test)
      begin
         if lamp_test = '1' then
            ss <= (others => '0');
         elsif blinkN = '0' then
            ss <= (others => '1');
         else
            case din is
               when "0000" => ss <= "1000000"; -- 0
               when "0001" => ss <= "1111001"; -- 1
               when "0010" => ss <= "0100100"; -- 2
               when "0011" => ss <= "0110000"; -- 3
               when "0100" => ss <= "0011001"; -- 4
               when "0101" => ss <= "0010010"; -- 5
               when "0110" => ss <= "0000010"; -- 6
               when "0111" => ss <= "1111000"; -- 7
               when "1000" => ss <= "0000000"; -- 8
               when "1001" => ss <= "0010000"; -- 9
               when "1010" => ss <= "0001000"; -- A
               when "1011" => ss <= "0000011"; -- B
               when "1100" => ss <= "1000110"; -- C
               when "1101" => ss <= "0100001"; -- D
               when "1110" => ss <= "0000110"; -- E
               when "1111" => ss <= "0001110"; -- F
               when others => ss <= (others => '1');
            end case;
         end if;
      end process;
end arc_hexss;
```

העתק דוח קומפילציה

Flow Status	Successful - Tue Aug 07 14:37:22 2018
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	lab_vhdl1
Top-level Entity Name	hexss
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	8 / 41,910 (< 1 %)
Total registers	0
Total pins	13 / 499 (3 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

היות ומדובר במערכת פשוטה שאותה נבדוק מיד בתחילת הניסוי באופן ישיר בחמרה, אין צורך לבצע סימולציה כל שהיא.

להזכירכם יש להביא למעבדה את כל קבצי הקוד והפרוייקטים שכתבתם, כי תשתמשו בהם

לאחר שסיימת - לחץ על ה LINK ומלא בבקשה את השאלון המצורף

מלא את הטופס

שמור כ PDF והעלה למודל