VHDL1 לבוחן

סוגי הצבות

הגדרת IF
לשמוש רק בתהליך.
	`if a = b then`
	`  equal <= '1' ;`
	`else`
	`  equal <= '0' ;`
	`end if ;`

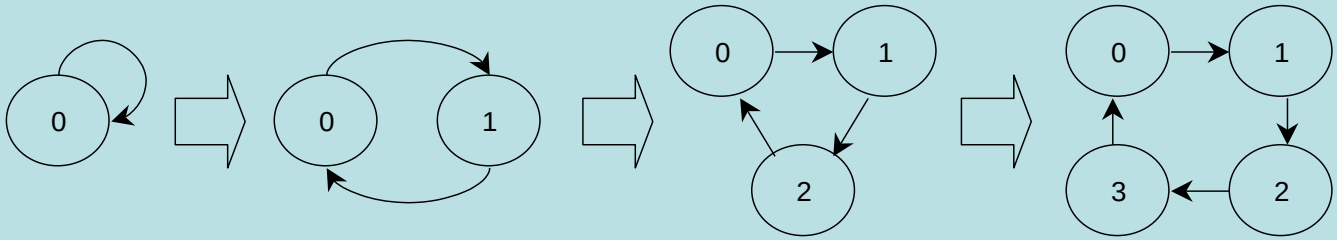
הגדרת CASE
לשמוש רק בתהליך.
	`case din is`
	`  when "11"  => y <= '1' ;`
	`  when others => y <= '0' ;`
	`end case ;`

הגדרת Conditional Assign לשמוש רק בארכיטקטורה.	`  equal <= '1' when (a = b) else '0' ;`

הגדרת Selected Assign
לשמוש רק בארכיטקטורה.
	`with sel select`
	`  dout <= ('0', '0', din) when "00" ,`
	`  ('0', din, '0') when "01" ,`
	`  (din, '0', '0') when "10" ,`
	`  (din, '0', '0') when others ;`

מעבדות בהנדסת חשמל 1,1ח'
044160 - 044151

ניסוי VHDL1




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

VHDL1 - תוכן המעבדה

1. הגדרות וכללים ב- VHDL
2. הוראות להעתקת קוד VHDL ל- WORD בצורה קריאה ב- Notepad ++
   I. סינתזה ובדיקת התכן בחמרה באמצעות מימוש בורר 1<=2
3. מונה עולה פשוט
4. שעון איטי
5. מונה מתנפח
   I. שמוש נכון בשעונים בעלי קצבים שונים
   II. הוספת שעון איטי למונה המתנפח
6. תכן היראכי
   I. הוספת קבצים לפרויקט
   II. יצירת סימבולים גרפים
   III. חיווט היראכיה עליונה
   IV. בדיקת המערכת השלמה

VHDL1 - הגדרות

סוגי אותות והגדרות של ספריות

1. נשתמש רק באותות מסוג (std_logic(_vector ולא בסוג (bit(_vector
לשם כך יש צורך להצהיר על הספריה IEEE.std_logic_1164.all.

2. לביצוע פעולות חשבוניות על וקטורים נצהיר על IEEE.std_logic_unsigned.all

3. לשימוש בפונקציות המרה $(\text{std\_logic} \rightleftarrows \text{integer})$ נצהיר על: IEEE.std_logic_arith.all

(המוטיבציה: in, out מוצהרים כ std_logic, חישובים נוח לבצע בייצוג integer)

```
1   library ieee ;
2   use IEEE.std_logic_1164.all;
3   use IEEE.std_logic_unsigned.all;
4   use IEEE.std_logic_arith.all;
```

```
6   entity bin2bcd is
7   port (     input     : in  std_logic_vector(5 downto 0);
8              dig1      : out std_logic_vector(3 downto 0);
9              dig2     : out std_logic_vector(3 downto 0);
10             dig3      : out std_logic_vector(3 downto 0)) ;
11  end bin2bcd;
```

VHDL1 - פונקציות להמרת משתנים

דוגמא לשמוש בפונקציות להמרת ייצוג של מספרים:

```
13  architecture arc_bin2bcd of bin2bcd is
14  begin
15      process (input)
16      variable tmp : integer := 0;
17      variable tmp1 : integer := 0;
18      variable units1 : integer := 0;
19      variable tens :integer :=0;
20      variable hundreds: integer :=0;
21          begin
22          tmp := conv_integer(input);
23
24
25
26
27
28
29          dig1 <= conv_std_logic_vector(units1,4);
30          dig2 <= conv_std_logic_vector(tens,4);
31          dig3 <= conv_std_logic_vector(hundreds, 4);
32      end process;
33  end arc_bin2bcd;
```

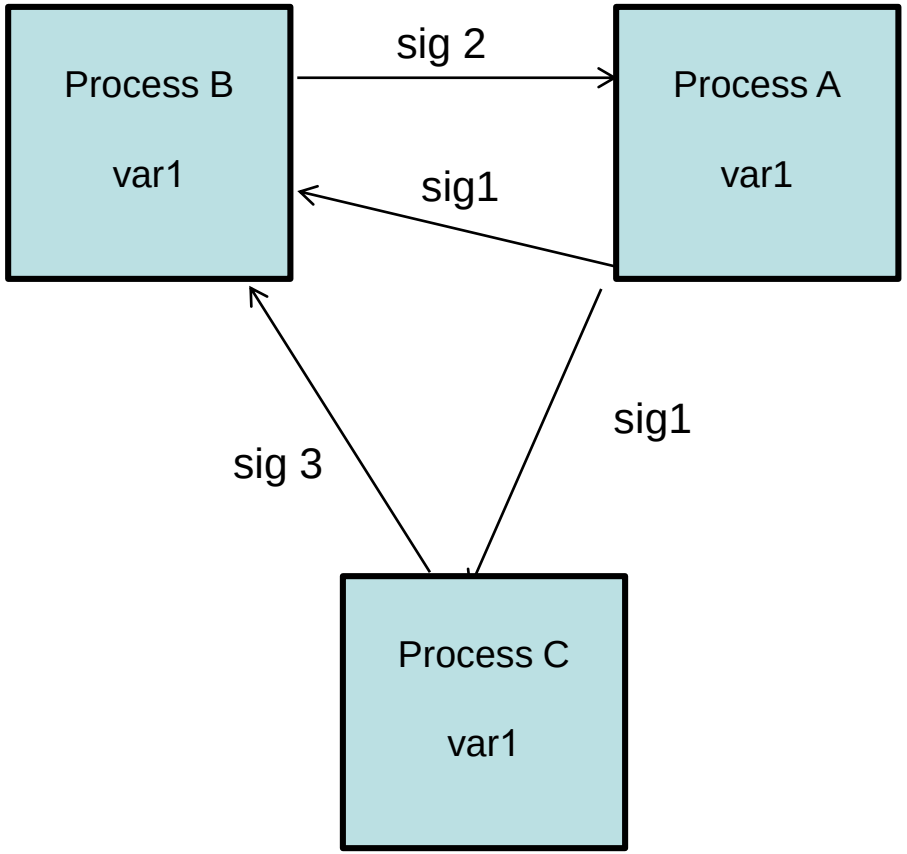
המרה מ- std_logic ל- integer

מקום לחישובים שלכם

המרה מ- integer ל-std_logic

VHDL1 - לוקליות של משתנים

```
1   library ieee ;
2   use ................
3
4   entity sfg is
5       port (..........
6               .........);
7   end sfg;
8
9   architecture arc_sfg of sfg is
10      signal  sig1, sig2, sig3:   std_logic;
11  begin
12
13      process (clk, ) --process A
14          variable  var1: integer;
15      begin
16      end process;
17
18      process (clk, ) --process B
19         variable  var1: integer;
20      begin
21      end process;
22
23      process (clk, resetN)--process C
24          variable  var1: integer;
25      begin
26      end process;
27
28  end arc_sfg;
29
```



משתנה var1 הוא לוקלי – שונה בכל Process

VHDL1 - מותר ואסור בהשמות VHDL

1. תאור

מספר השמות לאותו סיגנל בתוך תהליך

```
architecture arc_proj of proj is
    signal  a:      std_logic;
begin

    process (clk,resetN)
    begin
        .
        a <= 5;
        .
        a <= 6;
        .
    end process;

end arc_proj;
```

מותר

השמות לסיגנל מתבצעות בסוף התהליך. הסיגנל יקבל את הערך של ההשמה האחרונה.

2. תאור

מספר השמות לאותו סיגנל בתוך הארכיטקטורה

```
architecture arc_proj of proj is
    signal  a:      std_logic;
begin
        .
        a <= 5;
        .
        .
        a <= 6;
        .

end arc_proj;
```

שים לב! תהליך ללא רשימת רגישויות מתבצע כל הזמן כמו גם הוראה בודדת בארכיטקטורה - ראה לעיל

אסור

כל ההשמות בארכיטקטורה מתבצעות בו זמנית, אי אפשר לעשות בו זמנית השמה משני מקומות שונים.

3. תאור

השמה בשני תהליכים לאותו סיגנל

```
architecture arc_proj of proj is
    signal  a:      std_logic;
begin

    process (clk,resetN)
    begin
        .
        a <= 5;
        .
    end process;

    process (clk,resetN)
    begin
        .
        a <= 6;
        .
    end process;

end arc_proj;
```

אסור

השמה לסיגנל בתהליך מתבצעת בסופו. בכל התהליכים ההשמות בסוף מתבצעות בו זמנית. אי אפשר לעשות בו זמנית השמה משני תהליכים שונים לאותו סיגנל.

VHDL1 - תהליך סינכרוני ואסינכרוני

מבנה של תהליך סינכרוני הכולל אותות אסינכרוניים

1. ברמה העליונה של התהליך יש **רק** 2 חלקים:
   א. התניות של האותות האסינכרוניים וההשמות המתאימות שלהן.
   ב. התניה של השעון , clk, וההשמות המתבצעות כתוצאה ממנו.

2. כל ההתניות הנוספות הבאות בעקבות השעון, הן ברמות פנימיות ביחס להתניה של אות השעון.

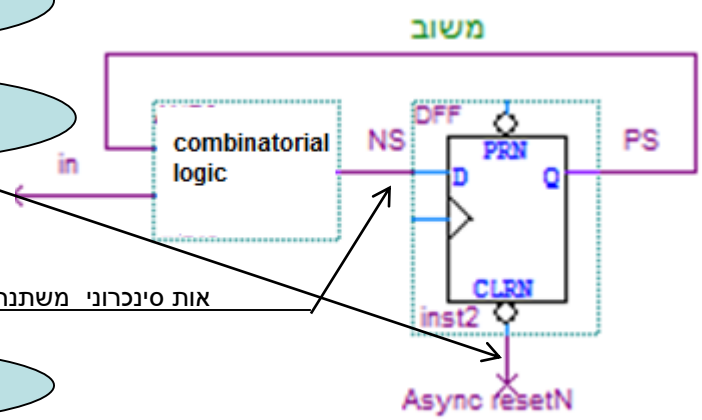
```
library ieee ;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity counter is
    port (  clk:     in  std_logic;
            resetN:  in  std_logic;
            cnt_ena: in  std_logic;
            loadN:   in  std_logic;
            din:     in  std_logic_vector(3 downto 0);
            dout:    out std_logic_vector(3 downto 0)
        );
end counter;

architecture arc_counter of counter is
begin
    process (clk, resetN)
        variable count: std_logic_vector(3 downto 0);
    begin
        if resetN = '0' then
            count := "0000";
        elsif rising_edge(clk) then
            if loadN = '0'then
                count := din;
            elsif cnt_ena = '1' then
                count := count + "0001";
            end if;
        end if;
        dout <= count;
    end process;
end arc_counter;
```

איפוס בהתחלה

קידום כל שעון

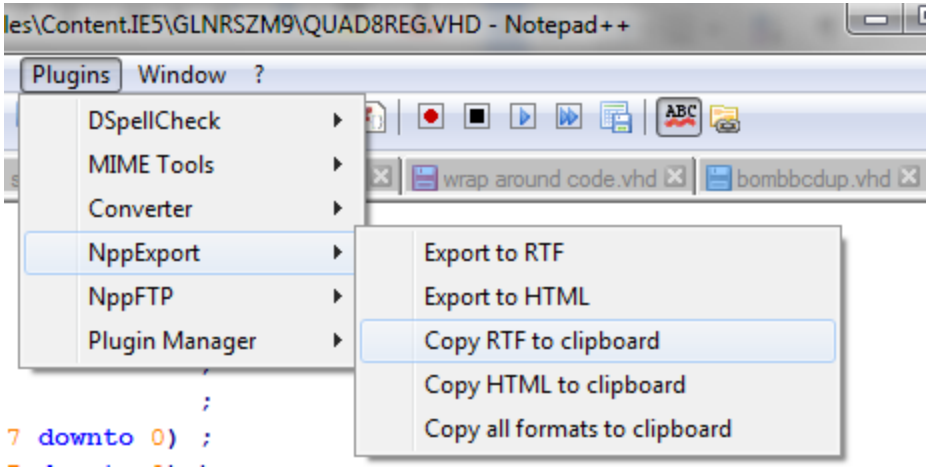
הצבה אסינכרונית בסוף

מכונת מצבים



VHDL1 - הוראות להעתקת קוד VHDL

ל- WORD בצורה קריאה באמצעות Notepad++



את NppExport ניתן להוריד מ- http://sourceforge.net/projects/npp-plugins/files/NppExport/

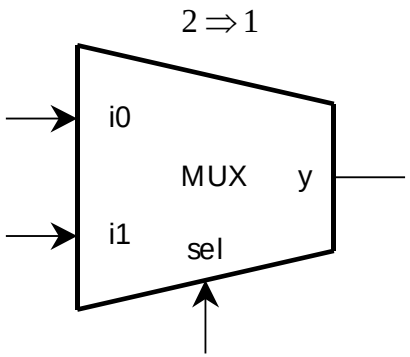
ל64 BIT קחו DLL מהמודל

VHDL1 - כתיבת VHDL

כתיבת VHDL – לימוד באמצעות מימוש בתוכנה של הבורר 2=>1

sel	y
0	i0
1	i1

$$y = \overline{sel} \bullet i_0 + sel \bullet i_1$$




```
-- a very simple example
entity mux2 is
   port( i1 , i0 , sel : in  bit;
         y      : out bit ) ;
end mux2 ; architecture arc_mux2 of mux2 is
begin
  y <= ( not sel and i0 ) or ( sel and i1 ) ;
end arc_mux2 ;
```

VHDL1 - כתיבת VHDL (המשך)

<u>כתיבת VHDL</u> – שלבי העבודה – בדומה לתכן סכמתי:

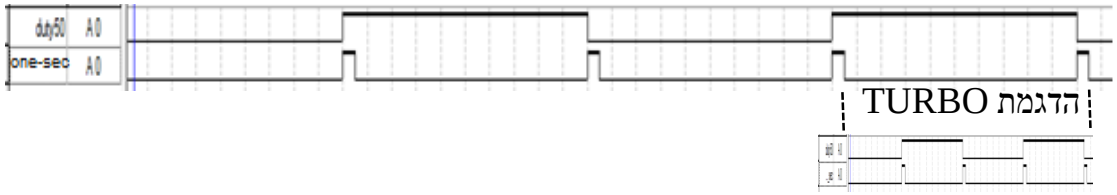
1. פתיחת פרויקט
   I. הגדרת תיקיה, שם לפרויקט והיראכיה עליונה Top-level
   II. הגדרת הכרטיס
2. כתיבה ב- VHDL
3. קומפילציה של הפרויקט
4. סימולציה פנימית בסביבת Quartus (ב modelsim)
5. הקצאת הדקים
6. תכנות/צריבה של המעגל על הכרטיס
7. בדיקת המעגל על הכרטיס

<u>תירגול</u> – בדיקת המונה הפשוט מעבודת ההכנה על הכרטיס

VHDL1 - שעון איטי

מבוסס על **רכיב השעון האיטי** מעבודת ההכנה

- פתח פרויקט חדש בשם INFLATING COUNTER
- הוסף לו את קובץ השעון האיטי
- הוסף לו יציאה בשם duty50, שתהבהב בתדר 1/2 Hz, ותחובר ל- LED.
  אות זה יהיה 50% מהזמן (שניה אחת) בגבוה ו- 50% מהזמן בנמוך
- חבר את כניסת הטורבו
  למפסק (לא ללחצן)



זכור להחזיר את משתנה

```
constant sec: integer := 50000000 ;
```

לערך הנכון אחרי הסימולציה

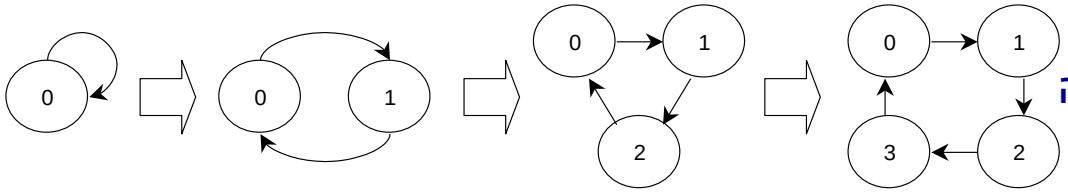
- עבוד לפי שלבי הפיתוח
- הורד לכרטיס ובדוק את נכונות התכן

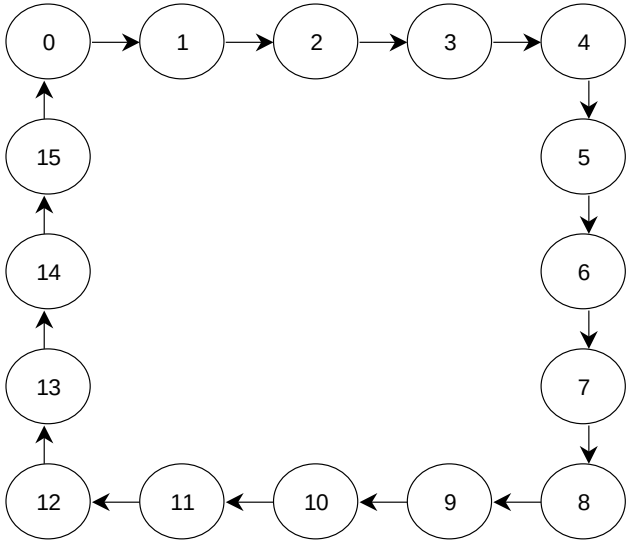
הפסקה בעבודה – מימוש מונה רגיל

- הסטודנטים ימשו את המונה הרגיל

VHDL1 - מונה מתנפח

יש לממש ב- VHDL את המונה המתנפח שבנית בתכן סכמתי ע"י כתיבה ב- VHDL של חלקים חסרים בקוד נתון.

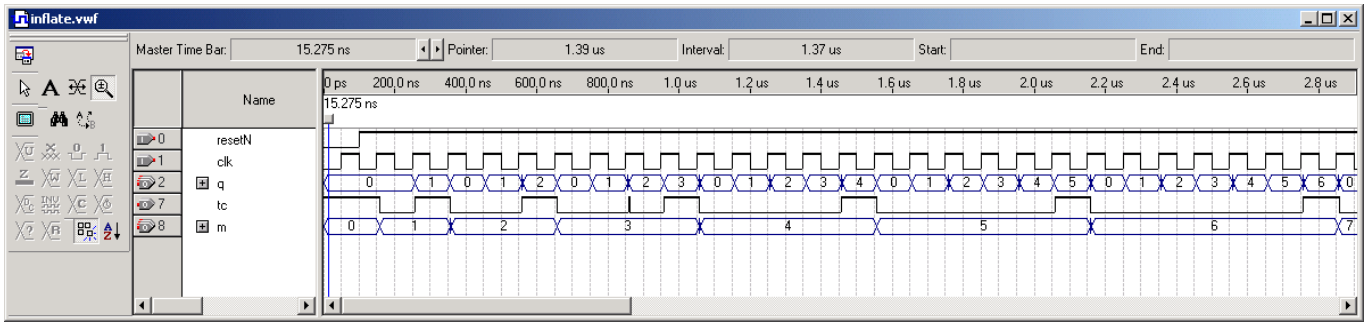




<u>הגדרת המונה המתנפח</u>

עליך לממש מונה שיחס החלוקה שלו הולך וגדל.
בתחילת הספירה (מיד לאחר האיפוס ה-א-סינכרוני של המונה), הספירה המכסימלית של המונה מגיעה ל – 0.
במחזור הספירה הבא, הספירה המכסימלית מגיעה ל - 1.
במחזור הספירה הבא הספירה המכסימלית מגיעה ל - 2.
במחזור הספירה הבא הספירה המכסימלית מגיעה ל - 3.
בסופו של דבר מחזור הספירה עולה ומגיע לספירה מכסימלית של 15:
לאחר מכן, הספירות המכסימליות של המונה הן שוב פעם 0, 1, 2, 3 וכו'.

להלן דוגמה לתוצאות סימולציה:

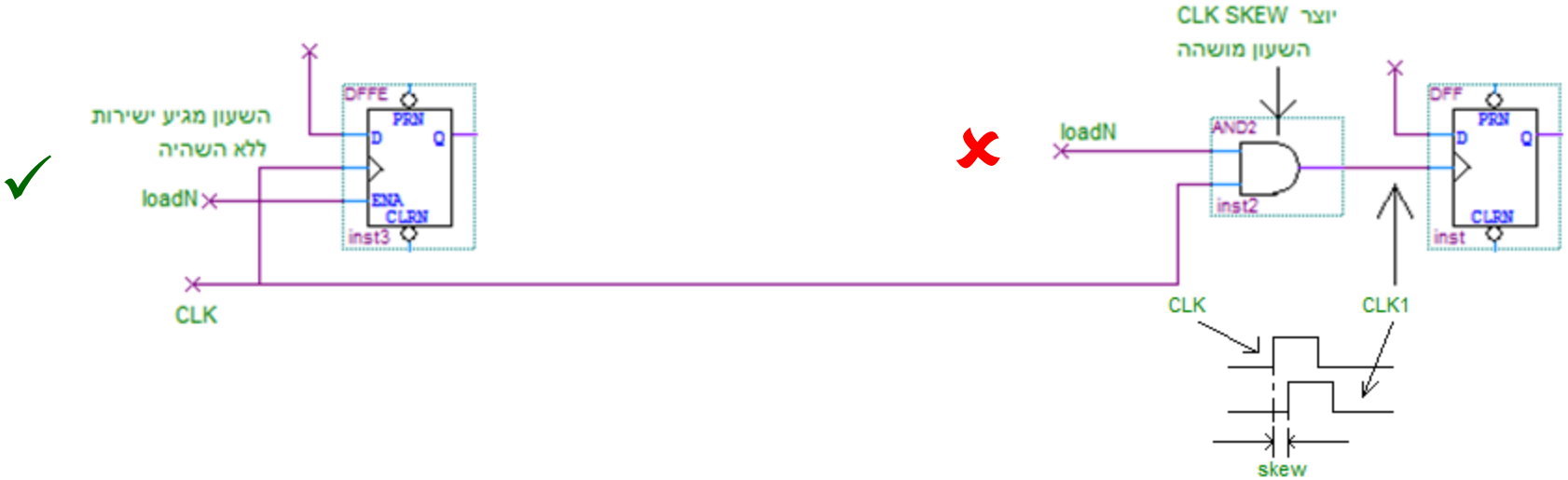


VHDL1 - שמוש נכון בשעונים בעלי קצבים שונים

דוגמא לכתיבה נכונה וכתיבה לא נכונה הגורמת ליצירת gated clock

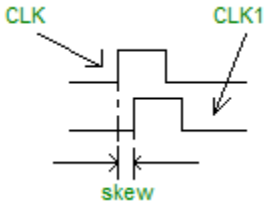
```
elsif rising_edge(clk) then
    if loadN = '0' then
```

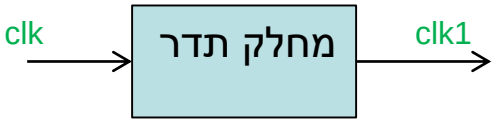
```
elsif rising_edge(clk) and loadN = '0' then
```



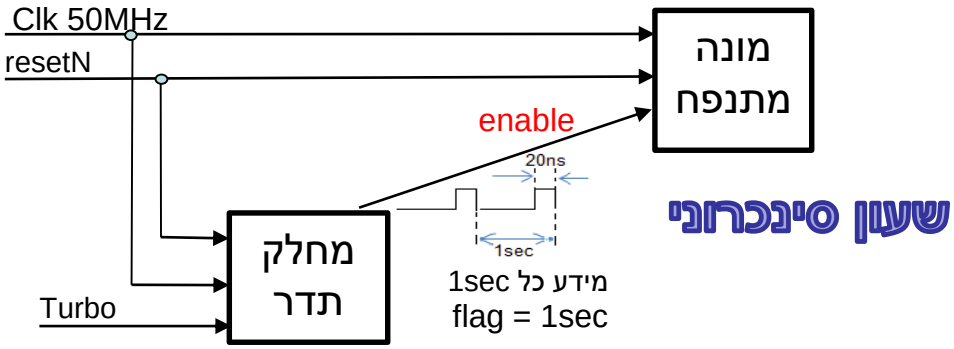
בעיה של **clock skew** עלולה לקרות גם כאשר נשתמש באותה מערכת בשני שעונים בתדרים שונים, כאשר השעון האיטי הוא נגזרת של השעון המהיר, מפני שקיימת השהיה בין שני השעונים.

במקרה כזה נעדיף שהשעון המהיר יהיה השעון היחיד במערכת.





VHDL1 - שמוש נכון בשעונים בעלי קצבים שונים



ביצוע:

- להוסיף כניסת enable למונה המתנפח
- להתנות את הפעולות הסינכרוניות של המונה המתנפח ב- enable
- לייצר סימבולים למונה המתנפח והשעון כולל מתג ה- Turbo
- להשתמש בשעון המהיר (50MHz) לכל היחידות

```
entity smart_inflate is
        enable : in std_logic ;
…
architecture smart_inflate_arch of smart_inflate is
   process (RESETN, CLK)  - - the fast clock for both
processes
…
  elsif (rising_edge(CLK)) then
   if ( enable = '1' ) then    - -the inflating
counter will work in 1Hz
…
  end process;
end smart_inflate_arch;
architecture slow_clk_arch of slow_clk is
  process(RESETN, CLK)   - - 50MHz is the clock of
slow clk process
  …
  enable = …                    - - 1Hz slow clk is
generated in this process
```

- **תוצאה**: לוודא שבקומפילציה אין הודעת אזהרה/שגיאה!

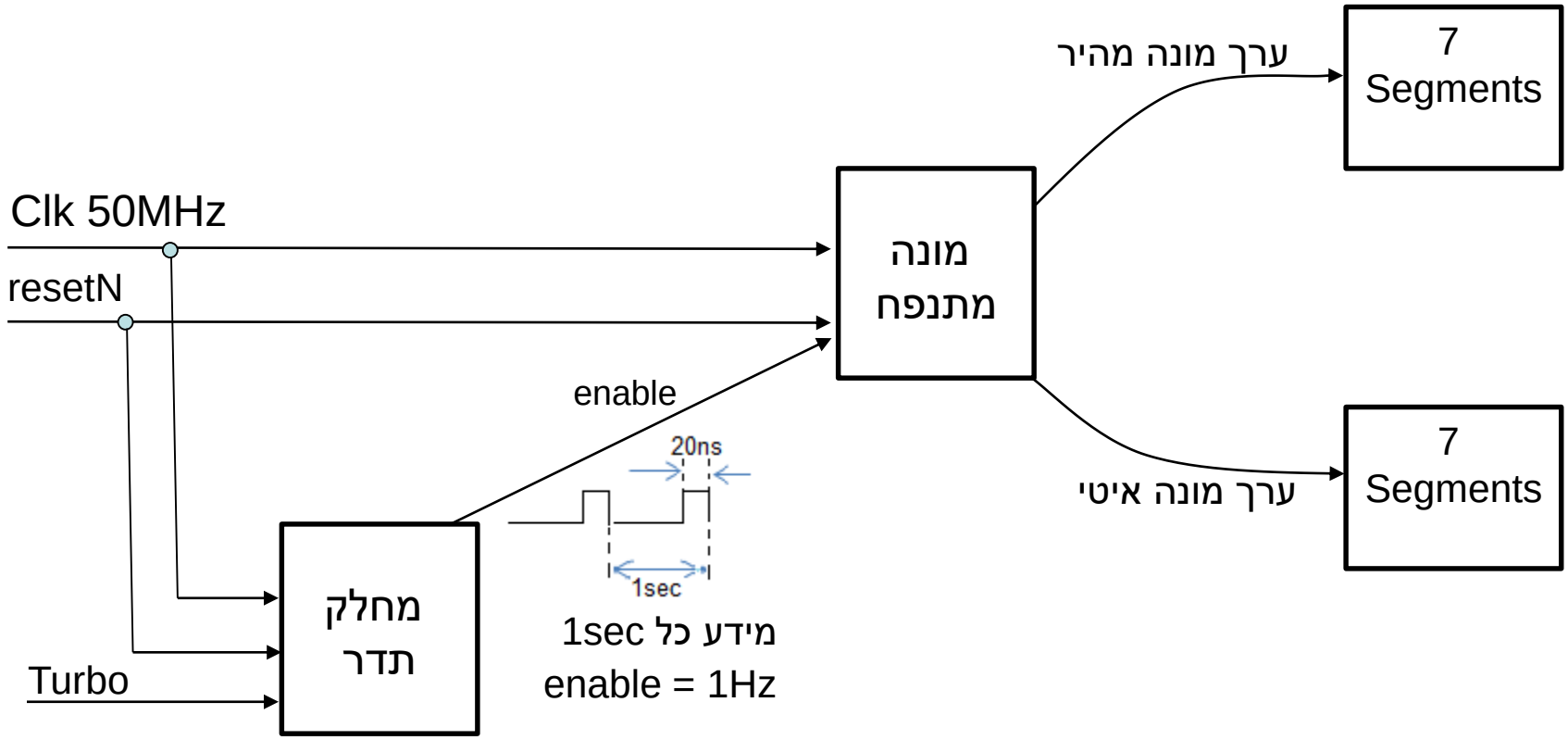
VHDL1 - מונה מתנפח - תכן הירארכי

מצב 3 – מונה מתנפח – כתכן סכימתי הירארכי – עם שעון איטי דרך enable

דיאגרמת בלוקים – שני התהליכים מופרדים ל- 2 רכיבים (2 ישויות)

– השעון האיטי הוא מודול חיצוני שמספק אפשור למונה המתנפח

– מוסיפים תצוגות 7 Segments – אחת לכל מונה

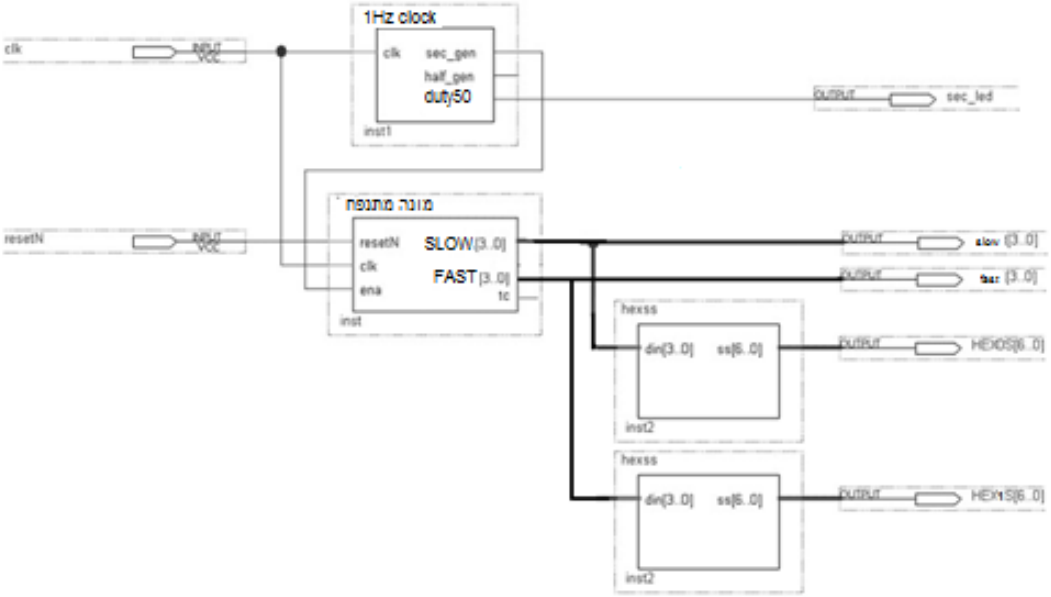


VHDL1 - מונה מתנפח - תכן הירארכי (המשך)

מונה מתנפח – כתכן סכימתי הירארכי

בפרויקט המונה המתנפח עבוד לפי השלבים הבאים:

- השתמש בשני קבצי VHDL נפרדים של השעון האיטי והמונה המתנפח
- התאם אותם לפי הצורך (הוספת כניסות, יציאות, מתג TURBO, אות ENABLE)
- הוסף לתיקיה את הקובץ HEXSS.VHD מדו"ח ההכנה
- בצע קומפילציה בסיסית – **אנליזה** לכל אחד מהקבצים
- צור או וודא שיש **Symbol גרפי** לכל אחד מהקבצים הנ"ל (כדי שאפשר יהיה להשתמש בהם בהיראכיה גבוהה יותר)
- חווט את ההיראכיה העליונה בקובץ גרפי חדש
- עדכן קובץ הקצאת הדקים
- קמפל
- צרוב לכרטיס
- בדוק את המערכת הסופית



סיום תכן היראַרכי והגשת דו״ח

לשמור את הקובץ ב- PDF ולהגיש במודל