משתנים

משתנה מסוג BIT

משתנה המכיל שני '0' או '1' ערכים והיא נרשמת בין גרש ימין לגרש שמאל .

משתנה מסוג BIT_VECTOR

מערך חד ממדי של משתנים מסוג **BIT** הניתן לכתיבה וקריאה, גם במיעון סיבית.

דוגמא להגדרה: Data :in **bit_vector** (7 downto 0)

בדוגמא מוגדר וקטור בגודל 8 סיביות

Data(7)	Data(6)	Data(5)	Data(4)	Data(3)	Data(2)	Data(1)	Data(0)
							LSB

דוגמאות להשמה:

```
Y<="11000011";              -- השמה בבינארי
Y<=x"C3";                   -- השמה בהקסה דצימלי
Y<=(others =>'0');          -- הצבה 0 לכל הסיביות
Y<=('1','1',OTHERS=>'0');   -- הצבה 11 לסיביות 7,6 וכל השאר 0 לוגי
Y<=(A, B ,OTHERS=>'0');     -- הצבה ערך של A לסיבית 7,ערך של B לסיבית 6 וכל השאר 0
Y(5 downto 2)<="1011";      -- הצבת הערך רק לסיביות 5 עד 2
Y(2)<='1';                  -- הצבה 1 לסיבית 2 בלבד
Y<=A xor B;                 -- פעולה לוגית , A,B הם מאותו סוג וגודל כמו Y
```

משתנה מסוג INTEGER

בגודל עד 32 סיביות ומסוגל לקבל ערך בטווח: 2147483648- עד 2147483647

דוגמאות להגדרת המשתנה:

```
Data_1 : in integer range  12 downto 0;      -- 4bit
Data_2 :  in integer range 0 to 255;         -- 8bit
```

כללים לגבי המשתנה

- נכתב ללא גרש או גרשיים
- שימושי לפעולות אריתמטיות ולא לוגיות
- התוכנה מקצה מספר סיביות לפי התחום

<u>השמה למשתנה מסוג INTEGER</u>

```
Y<=25;       -- השמת ערך עשרוני
Y<=A + B;    -- פעולה אריתמטית , A,B משתנים מאותו סוג כמו Y
Y<=Y+1;      -- קידום ב-1
```

משתנה מרובת ערכים - STD_LOGIC

משתנה מוגדר תחת הספרייה – STD_LOGIC_1164

ההצהרה בתחילת התוכנית:

```
library IEEE;

use IEEE.STD_LOGIC_1164.all;
```

למשתנה מסוג זה יש 9 מצבים לוגים:

1. '1' לוגי
2. '0' לוגי
3. 'Z' – עכבה גבוהה (Z נרשם באות גדולה)
4. 'X' – לא ידוע , יכול להתקבל כתוצאה מהתנגשות במידע (X נרשם באות גדולה)
5. '-' – DON'T CARE אפשר לתת ערך זה למשתנה
6. 'U' – ערך לא מאותחל, ערך של משתנה לפני שבוצעה השמה (U נרשם באות גדולה)
7. 'L' – '0' לוגי חלש , ערך הקרוב לסף המוגדר כ- 0 לוגי
8. 'H' – '1' לוגי חלש , ערך הקרוב לסף המוגדר כ- 1 לוגי
9. 'W' – לא ידוע חלש ('X' חלש)

אופן הרישום

- הגדרה של **BIT** - במקום **BIT** רושמים **STD_LOGIC**

דוגמא:

```
port( D0,D1 : in  STD_LOGIC;
      S : in  STD_LOGIC;
      Y : out  STD_LOGIC);
```

- וקטור מורכב ממערך של ביטים מסוג **STD_LOGIC** ונרשם בדומה **BIT_VECTOR** רגיל .

דוגמא:

```
a : in  STD_LOGIC_VECTOR (7 DOWNTO 0);

b : in  STD_LOGIC_VECTOR (0 to 7);
```

משתנה עזר פנימיים

Signal – נרשם בין ARCHITECTURE לבין begin.

משתנה מקבילי , במידה ומבוצעות עליו מספר פעולות , הפעולה האחרונה בלבד מתבצעת.

לדוגמא: שתי הפקודות הבאות יקדמו את cnt ב-3 בלבד(פקודה אחרונה)

Cnt<=cnt+1;

Cnt<=cnt+3;

Variable – נרשם בין process לבין begin ומוכר אך ורק על ידי אותו process.

משתנה טורי, במידה ומבוצעות עליו מספר פעולות, כל הפעולות מתבצעות.

לדוגמא: שתי הפקודות הבאות יקדמו את cnt ב-4 .

cnt:=cnt+1;

cnt:=cnt+3;

משתנה כללי – generic ומשתנה קבוע constant

Generic - נרשם בתוך entity , שימושי להגדרת גודל או תכונה משתנה של ה- entity שניתן להגדירה מבחוץ.

Constant – הגדרת ערך קבוע ומוכר רק על יד ה- entity

דוגמא:

ללא generic

```
entity register8 is
        port ( clk, rst, en: in std_logic;
                data: in std_logic_vector(7 downto 0);
                q: out std_logic_vector(7 downto 0));
end register8;
```

עם generic

```
entity register8 is
  generic ( width: integer := 8 );
        port ( clk, rst, en: in std_logic;
                data: in std_logic_vector(width-1  downto  0);
                q: out std_logic_vector(width-1   downto  0));
end register8;
```

עם constant

```
entity register8 is
  constant  width:  integer := 8 ;
          port ( clk, rst, en: in std_logic;
                  data: in std_logic_vector(width-1  downto  0);
                  q: out std_logic_vector(width-1   downto  0));
end register8;
```