שפת תאור חומרה VHDL

תוכן עניינים

מבנה בסיסי של תוכנית 2
דוגמא לשער AND..... 2
אופרטורים לוגים נוספים..... 3
דוגמא- רכיב 1 → MUX 2 4
תאור התנהגותי של מערכת..... 5
רכיב 1 → MUX 2 בעזרת פקודת when …. else..... 5
רכיב 1 → MUX 2 בעזרת פקודת with …. select 6
משתנה מסוג BIT_VECTOR 7
משתנה מסוג INTEGER..... 7
סוגי כניסות ויציאות 8
משתנה מרובת ערכים - STD_LOGIC 9
דוגמאות שונות 10
צורה 1 – שימוש בשרשור &..... 10
צורה 2 – שימוש במשתנה s כ- integer..... 10
מפענח מ-BCD ל- 7seg בעזרת פקודת select….with 11
מפענח מבינארי 5 ביטים ל-BCD..... 12
התהליך - PROCESS..... 13
תחביר פקודת if else 13
תחביר פקודת case 14
רשימת רגישויות 14
דוגמאות:..... 15
DFF הפועל בעליית שעון 15
DFF הפועל בעליית שעון עם reset אסינכרוני..... 15
DFF הפועל בעליית שעון עם reset ו- preset אסינכרוני..... 16
DFF הפועל בעליית שעון עם reset ו- preset סינכרוני..... 16
דוגמאות לתכנון מונים 17
מונה מעלה מ- 0 עד 255 עם איפוס סינכרוני..... 17
מונה מעלה/מטה מ- 0 עד 255 עם איפוס, טעינה ואפשור סינכרוני..... 18
מונה מעלה מודולו 200..... 19
מונה מעלה BCD עם reset אסינכרוני ואפשור סינכרוני 20
מכונת מצבים..... 21
דוגמא למכונת Moore 21
דוגמא למכונת Mealy..... 23
Mealy עם מוצא סינכרוני 25
תכנון היררכי..... 27
דוגמא – תכנון מסכם מלא – FA..... 28

מבנה בסיסי של תוכנית

מעגל דיגיטלי מתואר כקופסא עם כניסות ויציאות.

קופסא – מתוארת בשפה כישות entity

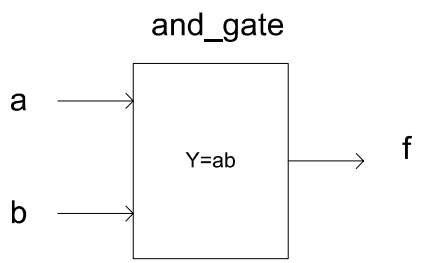
כניסה הפשוטה ביותר in מסוג **BIT** (0 או 1 לוגי)

יציאה הפשוטה ביותר out מסוג **BIT** (0 או 1 לוגי)

תפקוד המתאר את הקשר בין הכניסות והיציאות מוגדר ע"י המלה ARCHITECTURE

דוגמא לשער AND

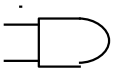
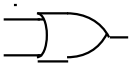
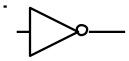
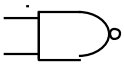
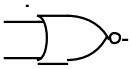
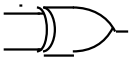
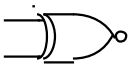
```
entity AND_GATE  is
        port(A,B : in BIT;
                f: out BIT);
end AND_GATE;

architecture TWO_INPUTS of AND_GATE  is
begin

        f<=A and B;

end TWO_INPUTS;
```



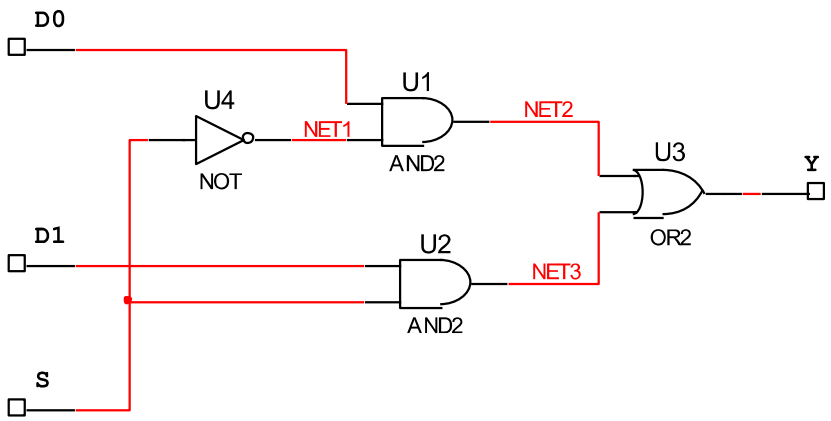
הסבר

- שם הקופסא - entity
- מבואות ומוצאים בתוך הסוגריים של port, אחרי סוגריים – נקודה פסיק.
- לחלק התפקודי שם , בדוגמא נבחר TWO_INPUTS
- לאחר begin הפעולה עם השמה =>
- סיום עם המילה end ושם TWO_INPUTS וסיום עם נקודה פסיק.

אופרטורים לוגים נוספים

- and 
- or 
- Not 
- Nand 
- nor 
- xor 
- xnor 

דוגמא- רכיב MUX 2 → 1




```
entity MUX_2_TO_1 is
        port(   D0,D1 : in BIT;
                S : in BIT;
                Y : out BIT );
end MUX_2_TO_1;


architecture  arc_Mux  of MUX_2_TO_1  is
begin

        Y <= ((not S) and  D0) or (S  and  D1);

end  arc_Mux  ;
```

<u>ניתן לתאר את אותה המערכת עם שימוש בסיגנלים פנימיים</u>

```
entity MUX_2_TO_1 is
        port(   D0,D1 : in BIT;
                S : in BIT;
                Y : out BIT );
end MUX_2_TO_1;


architecture  arc_Mux   of MUX_2_TO_1  is
        signal  NET1, NET2, NET3 :BIT;
begin
        NET1 <= not S;
        NET2 <= NET1 and D0;
        NET3 <= S and D1;
        Y <= NET2 or NET3;

end  arc_Mux  ;
```

תאור התנהגותי של מערכת

ניתן לתאר את המערכת הלוגית כמערכת התנהגותית בעזרת 2 פקודות:

- פקודת when .... else Conditional Assignment

מבנה הפקודה:

```
var    <=        value_1  when condition_1 else
                 value_2  when condition_2 else
                 value_3  when condition_3 else
                 .
                 .
                 value_n;
```

- פקודת with .... select Selected Assignment

מבנה הפקודה

```
with expression select
var    <=        value_1  when choice_1,
                 value_2  when choice_2,
                 value_3  when choice_3,
                 .
                 .
                 value_n  when others;
```

נתאר את רכיב 1 ← MUX 2 בעזרת שתי הפקודות:

רכיב 1 ← MUX 2 בעזרת פקודת when .... else

```
entity  MUX_2_TO_1  is
        port(    D0,D1 : in BIT;
                 S : in BIT;
                 Y : out BIT );
end  MUX_2_TO_1;


architecture  arc_Mux  of  MUX_2_TO_1  is
begin

        Y<= D0 when S='0' else
             D1;

end   arc_Mux  ;
```

רכיב 1 ← MUX 2 בעזרת פקודת with .... select

```
entity MUX_2_TO_1 is
        port(
                D0,D1 : in BIT;
                S : in BIT;
                Y : out BIT
                );
end MUX_2_TO_1;


architecture arc_Mux  of MUX_2_TO_1 is
begin
        with  S select
        Y<= D0 when '0',
        D1 when others;

End  arc_Mux  ;
```

<u>הערה:</u> משתנה מסוג **BIT** מכיל שני ערכים והיא נרשמת בין גרש ימין לגרש שמאל : '0' או '1'

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:

```
Data<="11000011";          -- השמה בבינארי
Data<=x"C3";               -- השמה בהקס דצימלי
Data<=(others =>'0');      -- הצבה 0 לכל הסיביות
Data<=('1','1',OTHERS=>'0');  -- הצבה 11 לסיביות 7,6 וכל השאר 0 לוגי
Data(5 downto 2)<="1011";  -- הצבת הערך רק לסיביות 5 עד 2
Data(2)<='1';              -- הצבה 1 לסיבית 2 בלבד
```

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

סוגי כניסות ויציאות

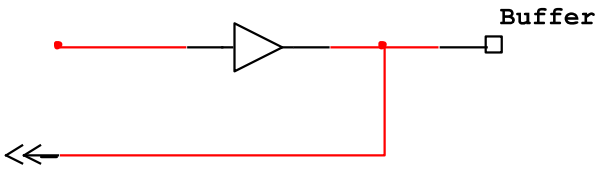
עד כה ראינו כניסה – IN הניתן לקריאה בלבד ויציאה – OUT הניתן לכתיבה בלבד

קיימים עוד שני סוגי -PORT

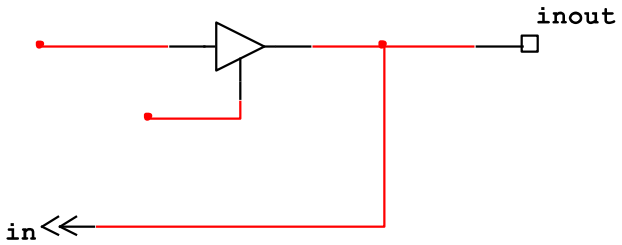
- Buffer - הניתן לכתיבה ולקריאה
- Inout – קו נתונים דו כווני הניתן לכתיבה וקריאה

מה ההבדל בין סוגי ה- PORT

Buffer - זהו PORT יציאה בלבד ולא ניתן לחבר אליו רכיב קלט, הקריאה מהיציאה היא לצורך פנימי בלבד לשם התניות או חיבור לכניסה פנימית .



Inout – זהו PORT יציאה וכניסה וניתן לחבר אליו רכיב קלט או פלט, כאשר הוא במצב של קלט, צריך לדאוג לעכבה גבוהה של הפלט הפנימי (אם משתמשים בעכבה גבוהה צריך להשתמש במשתנה מסוג **STD_LOGIC**).



בכל זאת ניתן להגדיר PORT מסוג INOUT כמשתנה **BIT** אבל לא ניתן לקבל עכבה גבוהה.

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);
```