מעבדה ב-VHDL: תרגיל 1 – Decoder

חלק 1: הדלקת LED באמצעות SW.

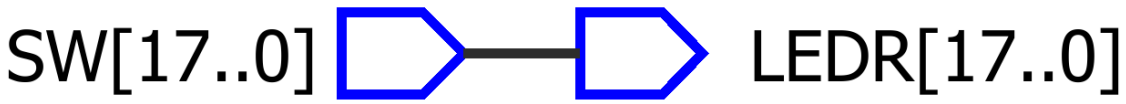
בתרגיל זה קיבלנו את הקוד הבא:

```
LIBRARY ieee;
USE ieee.std logic 1164.all;
- - Simple module that connects the SW switches to the LEDR lights
ENTITY part1 IS
PORT ( SW : IN STD LOGIC VECTOR(17 DOWNTO 0);
LEDR : OUT STD LOGIC VECTOR(17 DOWNTO 0)); - - red LEDs
END part1;
ARCHITECTURE Behavior OF part1 IS
BEGIN
LEDR <= SW;
END Behavior
```

התוכנית מכילה כמה שגיאות SYNTAX כך שהיינו צריכים לתקן אותה ולהריץ על הערכה DE2-115.
לאחר תיקון השגיאות התקבלה התוכנית הבאה:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-- Simple module that connects the SW switches to the LEDR lights
ENTITY part1 IS
PORT ( SW : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
LEDR : OUT STD_LOGIC_VECTOR(17 DOWNTO 0)); -- red LEDs
END part1;
ARCHITECTURE Behavior OF part1 IS
BEGIN
LEDR <= SW;
END Behavior;
```

ניתן לראות שהתוכנית פשוט מבצעת חומרה אשר מעבירה את מצב המפסקים לנורות LED.
ולכן נקבל ב-RTL פשוט חוט קצר בין הכניסה שנקראת SW למוצא LEDR.



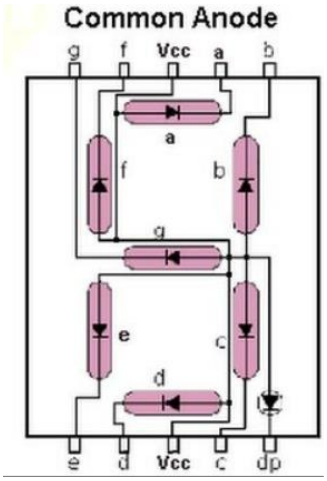
לאחר כתיבת התוכנית העלנו את הקובץ Excel להגדרת ההדקים על הערכה ל-Quartus ולאחר הצריבה התקבלה מערכת שמתפעלת את הנורות LED לפי מצב הלחצנים (מה שציפינו לקבל).

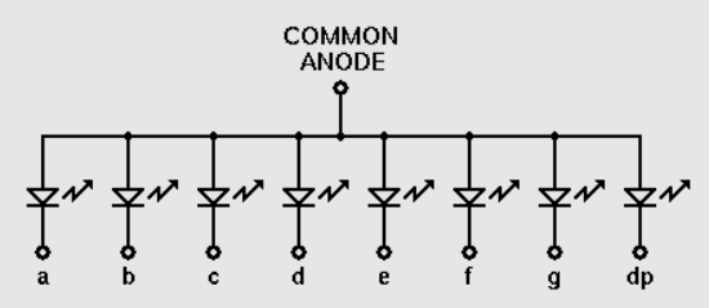
<u>חלק 2:</u> כתיבת מפענח (Decoder) לתצוגת 7Segments.

בתרגיל זה היה עלינו לכתוב רכיב בעל 3 כניסות (כניסות הבוחרות את התו שרוצים להציג על התצוגה) ו-7 יציאות כאשר כל יציאה מתחברת ל-LED אחר בתצוגה ומתפעלת אותו בהתאם לכניסה.

התווים שהיה עלינו להציג על התצוגה הם H,E,L,O,BLANK.

התצוגת 7Segment שקיימת בערכה במעבדה בנויה לפי שיטת CA (Common Anode).
מבנה התצוגה באופן סכמתי:





ניתן לראות את השרטוט המלא של חיבור התצוגה לרכיב FPGA בקובץ של השרטוט בעמוד 8.

בשיטת CA מספקים לנקודה VCC מתח גבוהה של '1' לוגי כך שההדק Anode בכל הנורות LED מקבל מתח בפוטנציאל גבוהה וכאשר נספק '0' לוגי בהדק a נדליק את הנורת LED ששייכת למקטע העליון בתצוגה.
ואם נספק '1' לוגי להדק a נכבה את הנורת LED ששייכת למקטע זה.

ככה יהיה עלינו לתאם בין המתחים שאנו מספקים למקטעים לתו שאנו רוצים להציג בתצוגה (זה יהיה בעצם הרכבה של נורות LED אשר חלקם כבויים וחלקם דלוקים).

תחילה נרשום טבלת אמת המתארת את אופן פעולת המפענח:

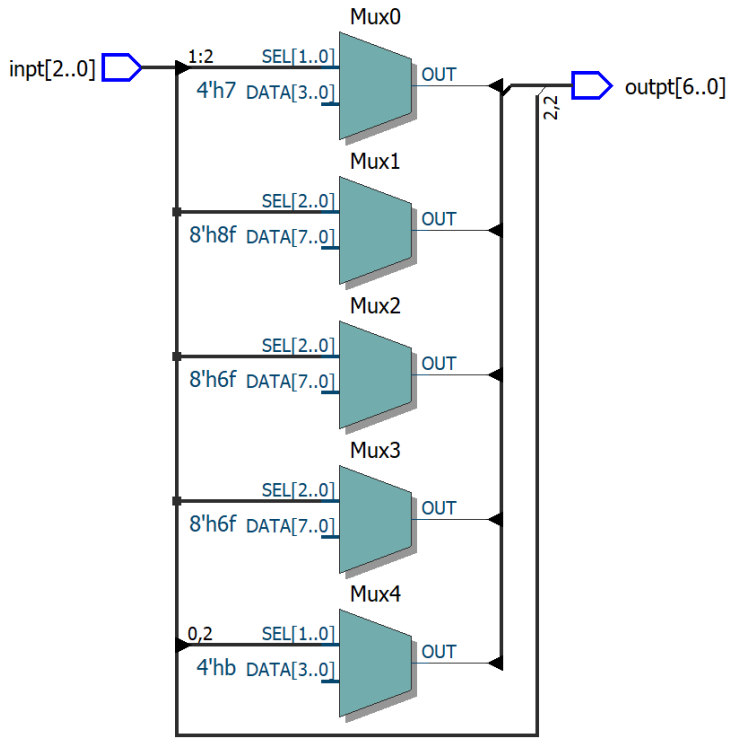
input	תו להצגה	g	f	e	d	c	b	a	Output
000	H	0	0	0	1	0	0	1	$(0001001)_2=(9)_{16}$
001	E	0	0	0	0	1	1	0	$(0000110)_2=(6)_{16}$
010	L	1	0	0	0	1	1	1	$(1000111)_2=(47)_{16}$
011	O	1	0	0	0	0	0	0	$(1000000)_2=()_{16}$
1--	BLANK	1	1	1	1	1	1	1	$(1111111)_2=()_{16}$

מכיוון שפיתוח אלגוריתם מתמטי (באמצעות נוסחאות מתמטיות) או אלגוריתם בוליאני (המבוסס על שערים לוגים) עלול להיות מסובך מדי או בלתי אפשרי נרשום את הארכיטקטורה באמצעות case אשר בודק מה יש בכניסה ומוציא את המידע בהתאם.

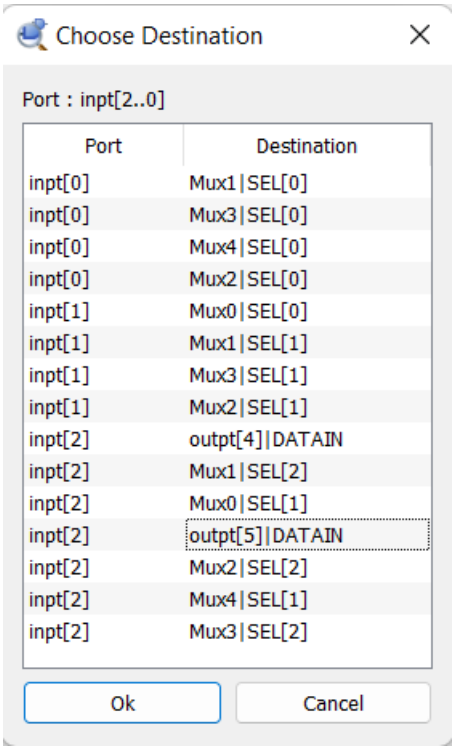
התוכנית:

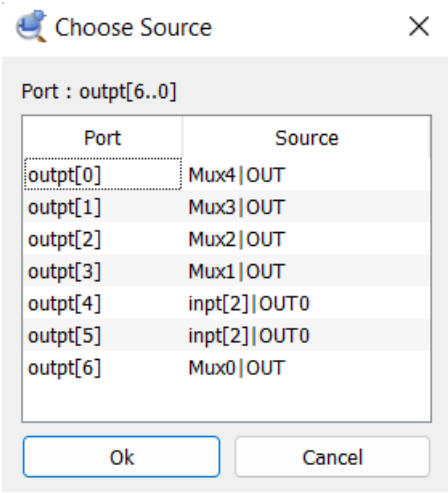
```
library ieee;
use ieee.std_logic_1164.all;

entity Decoder_7Segment is
port(
inpt: in std_logic_vector(2 downto 0);
outpt: out std_logic_vector(6 downto 0));
end;

architecture one of Decoder_7Segment is
begin
process(inpt)
begin
case inpt is
when "000" => outpt <= "0001001"; -- H "0001001"
when "001" => outpt <= "0000110"; -- E "0000110"
when "010" => outpt <= "1000111"; -- L "1000111"
when "011" => outpt <= "1000000"; -- O "1000000"
when OTHERS => outpt <= "1111111"; -- Blank
end case;
end process;
end;
```

בRTL נקבל מרבבים שבודקים את הכניסה ולפי זה מוציאים מספרים קבועים למוצא לפי הכניסה.
סכמת RTL:

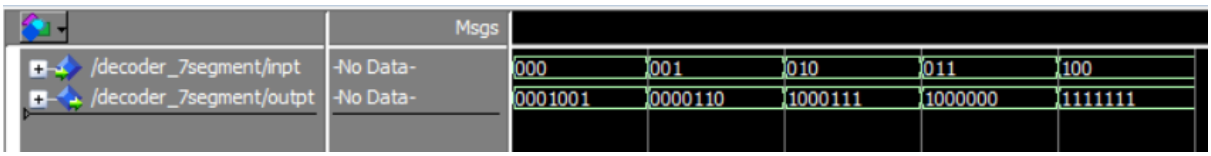


על מנת לראות בבירור מה מהכניסות מחוברות לאן נפתח את ה- Destination ששייך ל-net של הכניסה והמוצא





כדי לעשות את הניתוח פשוט יותר אראה דיאגרמת זמנים של המערכת.



לאחר שבדקנו וראינו שהמפענח עובד עברנו לכתיבה של הצגת רצף על התצוגה:
היה עלינו לעבוד כעת לפי הטבלת אמת הבאה:

SW_{17} SW_{16} SW_{15}	Character pattern				
000	H	E	L	L	O
001	E	L	L	O	H
010	L	L	O	H	E
011	L	O	H	E	L
100	O	H	E	L	L

כדי שזה יעבוד תכננו את המערכת באמצעות מרבבים שיקלטו את הכניסות ובהתאם לכניסות יעבירו את הקידוד המתאים לתצוגה.

<u>תכנון המרבב</u>

לצורך תפעול המערכת תכננו מרבב בעל 5 כניסות מידע כאשר כל כניסה היא ווקטור בגודל 4[Bit]
הselector צריך לבחור אחד מארבעת הכניסות לכן הוא יהיה בגודל 3[Bit]
ולמערכת קיימת רק יציאה אחת שצריכה להיות באותו גודל כמו הכניסה (ווקטור 4[Bit]).

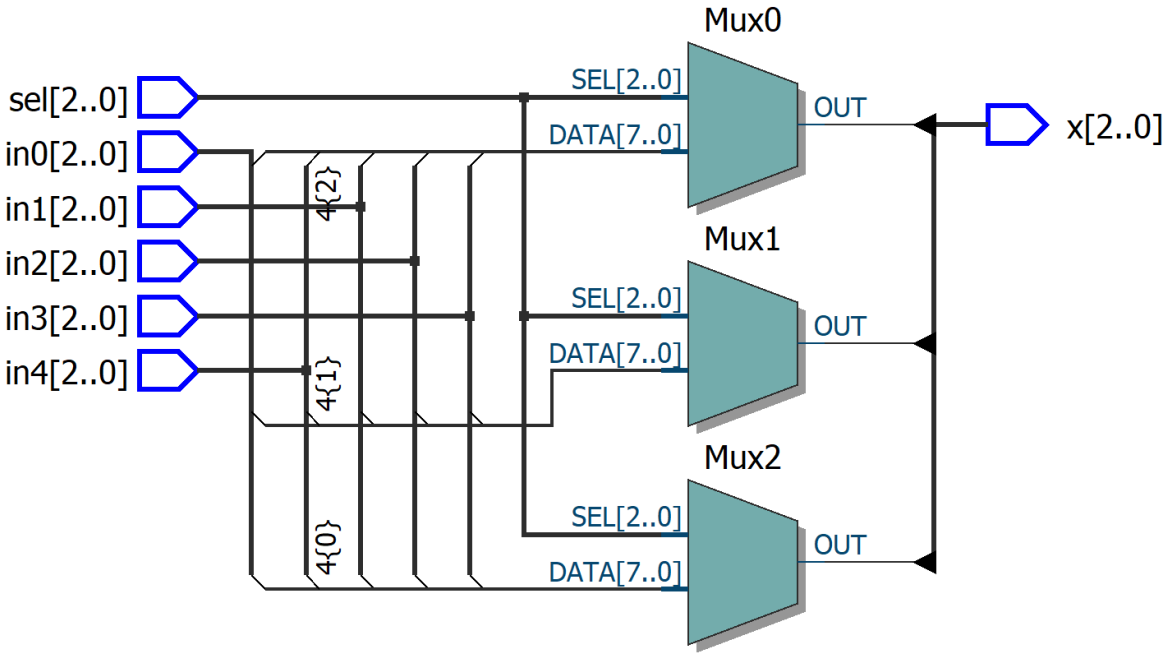
טבלת אמת לתיאור המערכת:

Selector	Output
000	Input0
001	Input1
010	Input2
011	Input3
100	Input4

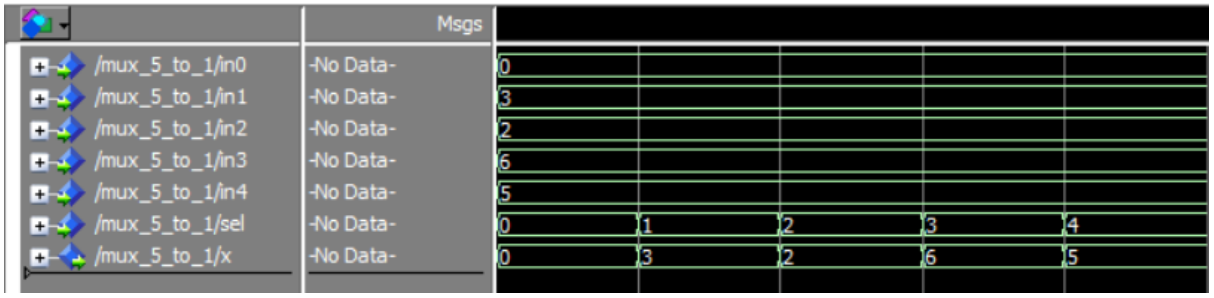
תוכנית לתיאור המערכת:

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

entity Mux_5_To_1 is
generic (M: integer := 3);
port(
in0: in  std_logic_vector(M-1 downto 0);
in1: in  std_logic_vector(M-1 downto 0);
in2: in  std_logic_vector(M-1 downto 0);
in3: in  std_logic_vector(M-1 downto 0);
in4: in  std_logic_vector(M-1 downto 0);
sel: in  std_logic_vector(2 downto 0);
x:   out std_logic_vector(M-1 downto 0));
end;

architecture one of Mux_5_To_1 is
begin
with sel select
x <= in0 when "000",
     in1 when "001",
     in2 when "010",
     in3 when "011",
     in4 when others;
end;
```

סכמת RTL של המערכת:



דיאגרמת זמנים לניתוח ובדיקת המערכת:



כעת ניתן לתכנן את המערכת הנדרשת לשאלה.
במערכת צריך לתפעל חמישה תצוגות ולכן נצטרך לעשות שימוש בחמישה מפענחים.
לכניסה של כל מפענח נחבר מרבב שישלוט על מה שנרצה להציג בתצוגה ולבורר של המרבב נחבר את המפסקים.

תוכנית:

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

entity Letter_Sequence is
port(
Sw: in std_logic_vector(2 downto 0);
disp1: out std_logic_vector(6 downto 0);
disp2: out std_logic_vector(6 downto 0);
disp3: out std_logic_vector(6 downto 0);
disp4: out std_logic_vector(6 downto 0);
disp5: out std_logic_vector(6 downto 0));
end;

architecture one of Letter_Sequence is

signal s1: std_logic_vector(2 downto 0);
signal s2: std_logic_vector(2 downto 0);
signal s3: std_logic_vector(2 downto 0);
signal s4: std_logic_vector(2 downto 0);
signal s5: std_logic_vector(2 downto 0);

component Mux_5_To_1
generic (M: integer := 3);
port(in0: in  std_logic_vector(M-1 downto 0);
in1: in  std_logic_vector(M-1 downto 0);
in2: in  std_logic_vector(M-1 downto 0);
in3: in  std_logic_vector(M-1 downto 0);
in4: in  std_logic_vector(M-1 downto 0);
sel: in  std_logic_vector(2 downto 0);
x:   out std_logic_vector(M-1 downto 0));
end component;
```

```
component Decoder_7Segment
port(inpt: in std_logic_vector(2 downto 0);
outpt: out std_logic_vector(6 downto 0));
end component;
--H - "000"
--E - "001"
--L - "010"
--O - "011"
begin
U1: Mux_5_To_1 port map ("000","001","010","010","011",Sw,s1);
U2: Mux_5_To_1 port map ("001","010","010","011","000",Sw,s2);
U3: Mux_5_To_1 port map ("010","010","011","000","001",Sw,s3);
U4: Mux_5_To_1 port map ("010","011","000","001","010",Sw,s4);
U5: Mux_5_To_1 port map ("011","000","001","010","010",Sw,s5);

M1: Decoder_7Segment port map (s5,disp1);
M2: Decoder_7Segment port map (s4,disp2);
M3: Decoder_7Segment port map (s3,disp3);
M4: Decoder_7Segment port map (s2,disp4);
M5: Decoder_7Segment port map (s1,disp5);
end;
```

סכמת RTL: