מעבדה בVHDL – Adder

חלק 1 - כתיבת מקודד לתצוגת שבעת המקטעים

כדי לכתוב את התוכנית למקודד לתצוגה עלינו היה לתכנן תחילה את הרכיב מתוך הטבלת אמת:

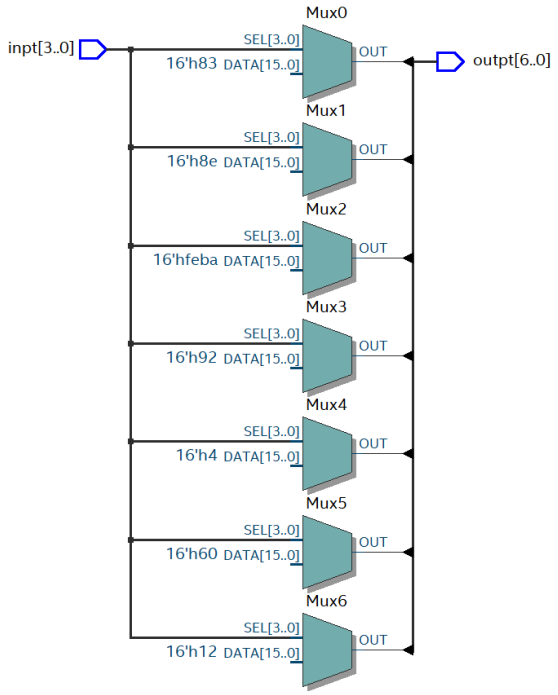
סכלה	INPUT	g-MSB HEX[6]	f HEX[5]	e HEX[4]	d HEX[3]	c HEX[2]	b HEX[1]	a-LSB HEX[0]	output
	"0000" 0	1	0	0	0	0	0	0	"1000000"
	"0001" 1	1	1	1	1	0	0	1	"1111001"
	"0010" 2	0	1	0	0	1	0	0	"0100100"
	"0011" 3	0	1	1	0	0	0	0	"0110000"
	"0100" 4	0	0	1	1	0	0	1	"0011001"
	"0101" 5	0	0	1	0	0	1	0	"0010010"
	"0110" 6	0	0	0	0	0	1	0	"0000010"
	"0111" 7	1	1	1	1	0	0	0	"1111000"
	"1000" 8	0	0	0	0	0	0	0	"0000000" $0x0 = (0)_{10}$
	"1001" 9	0	0	1	0	0	0	0	"0010000"

לאחר שקיבלנו טבלת אמת הכנסנו את הנתונים לפי LUT לתוכנית.

התוכנית

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

entity Decoder7Segment is
port(
inpt:  in  std_logic_vector(3 downto 0);
outpt: out std_logic_vector(6 downto 0));
end;

architecture one of Decoder7Segment is
begin

with inpt select
outpt <= "1000000" when "0000", -- '0'
         "1111001" when "0001", -- '1'
         "0100100" when "0010", -- '2'
         "0110000" when "0011", -- '3'
         "0011001" when "0100", -- '4'
         "0010010" when "0101", -- '5'
         "0000010" when "0110", -- '6'
         "1111000" when "0111", -- '7'
         "0000000" when "1000", -- '8'
         "0010000" when OTHERS; -- '9'
--       "" when "1010", -- 'A'
--       "" when "1011", -- 'B'
--       "" when "1100", -- 'C'
--       "" when "1101", -- 'D'
--       "" when "1110", -- 'E'
--       "" when "1111", -- 'F' (OTHERS)

end;
```

אנו מצפים לקבל בRTL מימוש באמצעות מרבבים

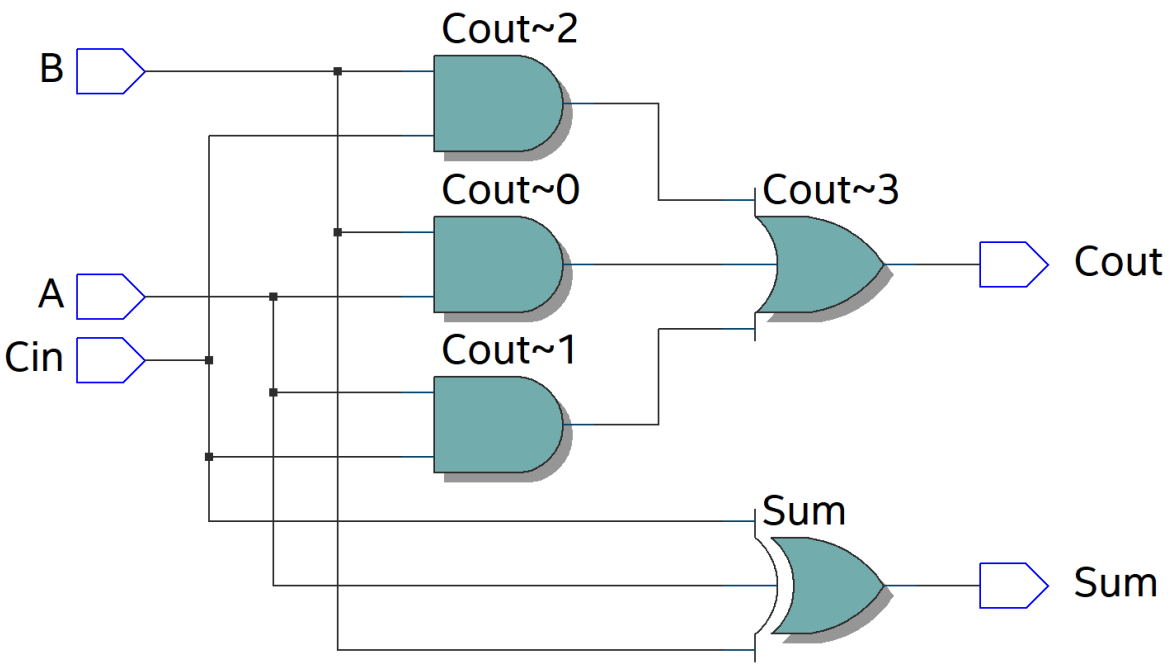


חלק 2 – רכיב FA

ידוע לנו כבר כיצד מורכב FA מקורסים קודמים לכן המימוש שלו התבסס על ידע קודם שלנו.

```
-- / FA - Component Architecture \ --

library ieee;
use ieee.std_logic_1164.all;

entity FA is
port(
A,B,Cin: in std_logic;
Sum,Cout: out std_logic);
end;

architecture one of FA is
begin
Sum <= A xor B xor Cin;
Cout <= (A and B) or (A and Cin) or (B and Cin);
end;
```

:RTL



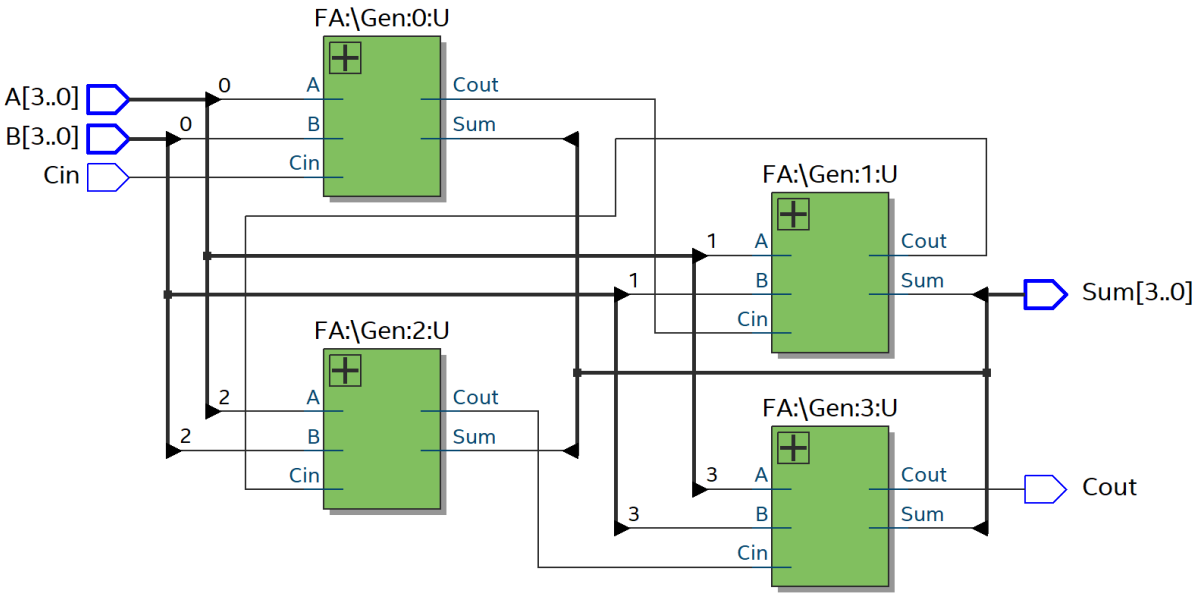
חלק 3 – כתיבת FA ל4 סיביות

ברכיב זה צריך לשרשר את סיביות הנשא (Carry).

התוכנית:

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

entity FA_4Bit is
generic (N: integer:=4);
port(
A,B:  in  std_logic_vector(N-1 downto 0);
Cin:  in  std_logic;
Sum:  out std_logic_vector(N-1 downto 0);
Cout: out std_logic);
end;

architecture one of FA_4Bit is

component FA
port(A,B,Cin:  in  std_logic;
     Sum,Cout: out std_logic);
end component;

signal Carry: std_logic_vector(N downto 0);

begin
Gen: for i in 0 to N-1 Generate
U: FA port map (A(i),B(i),Carry(i),Sum(i),Carry(i+1));
end Generate;
Carry(0) <= Cin;
Cout <= Carry(N);
--Carry(N) <= Cout;
end;
```

:RTL



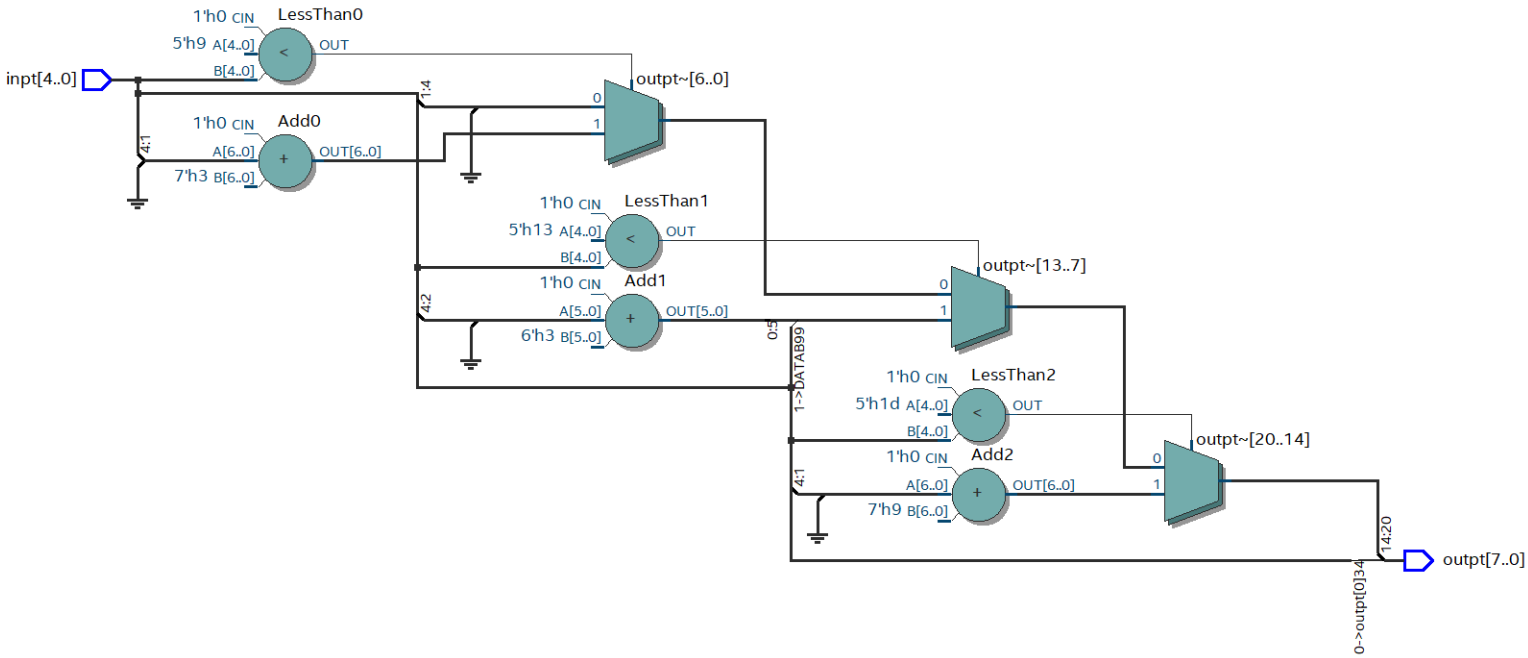
חלק 4 – מפענח BCD לתיאום בין המחבר לתצוגה

עבדנו לפי האלגוריתם שהיה נתון בעבודה.

התוכנית:

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

entity BCD_Decoder is
port(inpt:  in  std_logic_vector(4 downto 0);
     outpt: out std_logic_vector(7 downto 0));
end;

architecture one of BCD_Decoder is
signal s: std_logic_vector(7 downto 0);
begin

s <= ("000")&inpt;

outpt <= s + 18 when inpt > 29 else
         s + 12 when inpt > 19 else
         s + 6  when inpt > 9 else
         s;

end;
```

:RTL



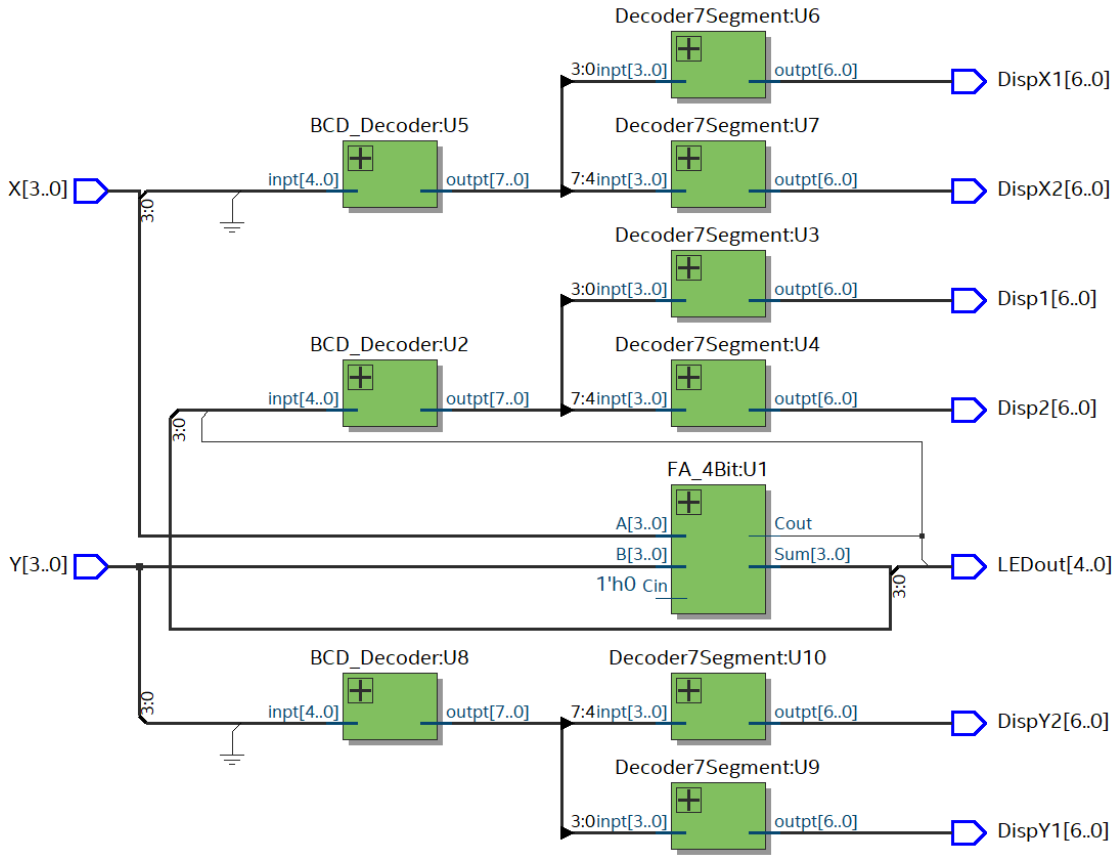
חלק 5 – שילוב כל המערכות לתפעול המסכם על התצוגה

להלן התוכנית שמקשרת בין כל הרכיבים:

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

entity Adder_7Segment is
port(
X,Y: in std_logic_vector(3 downto 0);
LEDout: out std_logic_vector(4 downto 0);
DispX1: out std_logic_vector(6 downto 0);
DispX2: out std_logic_vector(6 downto 0);
DispY1: out std_logic_vector(6 downto 0);
DispY2: out std_logic_vector(6 downto 0);
Disp1: out std_logic_vector(6 downto 0);
Disp2: out std_logic_vector(6 downto 0));
end;

architecture one of Adder_7Segment is

-- Components --
component FA_4Bit
generic(N: integer:=4);
port(A,B:  in  std_logic_vector(N-1 downto 0);
     Cin:  in  std_logic;
     Sum:  out std_logic_vector(N-1 downto 0);
     Cout: out std_logic);
end component;

component BCD_Decoder
port(inpt:  in  std_logic_vector(4 downto 0);
     outpt: out std_logic_vector(7 downto 0));
end component;
component Decoder7Segment
port(inpt:  in  std_logic_vector(3 downto 0);
     outpt: out std_logic_vector(6 downto 0));
end component;
-- Components --

-- Signals --
signal s_FA_BCD: std_logic_vector(4 downto 0);
signal s_D1:     std_logic_vector(3 downto 0);
signal s_D2:     std_logic_vector(3 downto 0);

signal s_D31:    std_logic_vector(3 downto 0);
signal s_D32:    std_logic_vector(3 downto 0);
signal s_D41:    std_logic_vector(3 downto 0);
signal s_D42:    std_logic_vector(3 downto 0);
-- Signals --

begin

-- / Architecture for the output (main function HERE) \ --
U1: FA_4Bit         port map (X, Y, '0', s_FA_BCD(3 downto 0), s_FA_BCD(4));
U2: BCD_Decoder     port map (inpt => s_FA_BCD, outpt(3 downto 0) => s_D1, outpt(7 downto 4) => s_D2);
U3: Decoder7Segment port map (s_D1, Disp1);
U4: Decoder7Segment port map (s_D2, Disp2);
-- / Architecture for the output (main function HERE) \ --

LEDout <= s_FA_BCD;

-- / This architecture for diplay the input on the 7Segment Display \ --
U5: BCD_Decoder     port map (inpt => '0' & X, outpt(3 downto 0) => s_D31, outpt(7 downto 4) => s_D32);
U6: Decoder7Segment port map (s_D31, DispX1);
U7: Decoder7Segment port map (s_D32, DispX2);
```

```
U8:  BCD_Decoder     port map (inpt => '0' & Y, outpt(3 downto 0) => s_D41, outpt(7 downto 4) => s_D42);
U9:  Decoder7Segment port map (s_D41, DispY1);
U10: Decoder7Segment port map (s_D42, DispY2);
-- / This architecture for diplay the input on the 7Segment Display \ --

end;
```

:RTL



בצריבה הגדרנו את ההדקים דרך החלון Pin Planner (לא העלנו קובץ Excel לQuartus).