ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ (MICROLAB)

2η Εργαστηριακή Αναφορά στο μάθημα "**ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ VLSI "** του 8ου Εξαμήνου

των φοιτητών της **ομάδας 17**,

Εμμανουήλ Αναστάσιου Σερλή, Α.Μ. 03118125
Κωνσταντίνου Ιωάννου, ΑΜ: 03119840

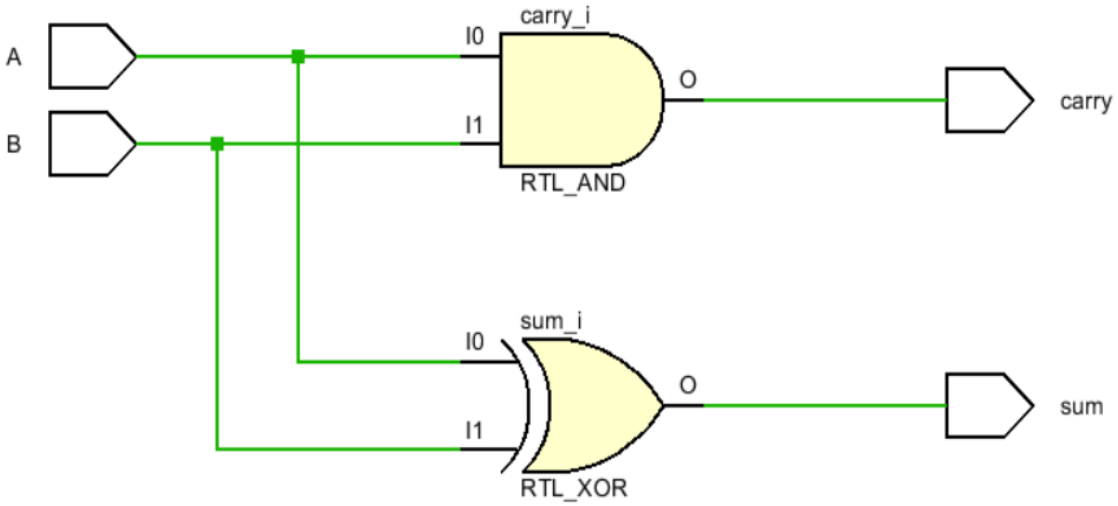
Ερώτημα 1:

Σκοπός του ερωτήματος είναι να υλοποιήσουμε έναν Ημιαθροιστή (Half Adder - HA) σε περιγραφή ροής δεδομένων (Dataflow), ενώ στα επόμενα ερωτήματα με βάση αυτόν τον αθροιστή θα δημιουργήσουμε πιο περίπλοκα λογικά κυκλώματα με χρήση δομικής περιγραφής.

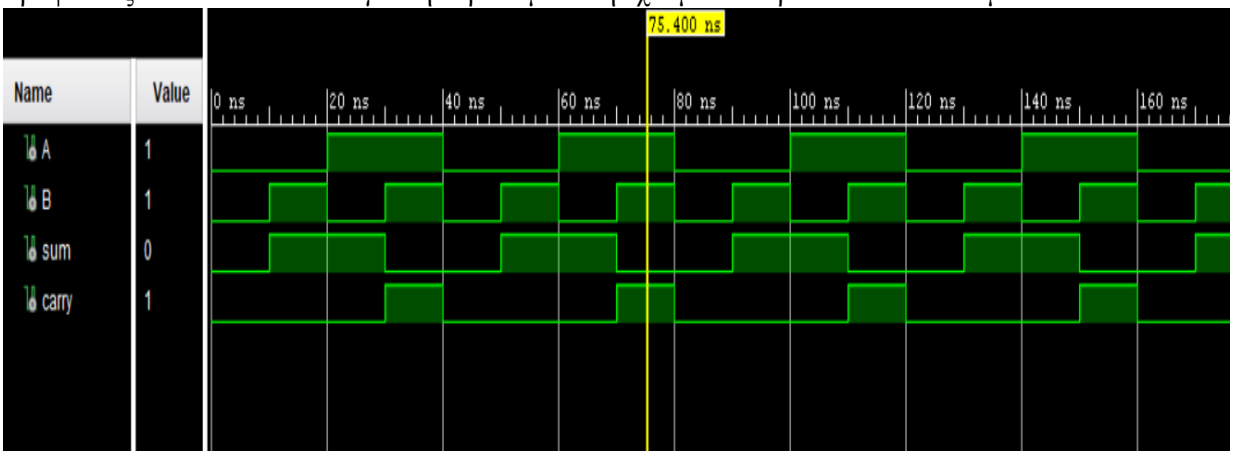
- Κώδικας για την αρχιτεκτονική:

```
7
8  entity haldAdder is
9    Port (
10   A: in std_logic ;
11   B: in std_logic ;
12   sum: out std_logic ;
13   carry: out std_logic
14    );
15 end haldAdder;
16
17 architecture dataflow of haldAdder is
18 begin
19
20 carry <= A AND B ;
21 sum <= A XOR B;
22
23 end dataflow;
```

- RTL Schematic:



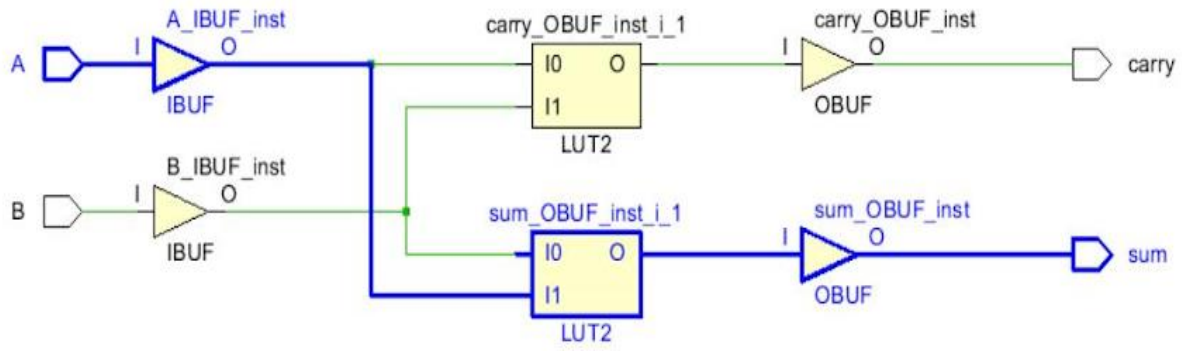
- Αποτελέσματα Testbench:
  Γράφοντας έναν απλό κώδικα για την προσομοίωση έχουμε το παρακάτω αποτέλεσμα:



Από την παραπάνω προσομοίωση επιβεβαιώνεται η αντιστοίχιση του πίνακα αλήθειας με τα αποτελέσματα του half-adder :

Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Critical Path A→Sum (delay = 6.9ns)



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	D
Path 1	∞	3	2	2	A	sum	6.900	3.937	2.963	∞	input port clock	
Path 2	∞	3	2	2	A	carry	6.491	3.672	2.819	∞	input port clock	

Το κρίσιμο μονοπάτι είναι το μονοπάτι με την μεγαλύτερη καθυστέρηση. Στη συγκεκριμένη περίπτωση είναι το Path1, δηλαδή η διαδρομή από το σήμα εισόδου A ως την έξοδο sum.

Αναλυτικότερα τρέχοντας την εντολή “report_timing_summary -report_unconstrained” στο terminal, λαμβάνουμε:

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack:                    inf
  Source:                 A
                            (input port)
  Destination:            sum
                            (output port)
  Path Group:             (none)
  Path Type:              Max at Slow Process Corner
  Data Path Delay:        6.900ns  (logic 3.937ns (57.056%)  route 2.963ns (42.944%))
  Logic Levels:           3  (IBUF=1 LUT2=1 OBUF=1)

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
    T19                                               0.000     0.000 r  A (IN)
                         net (fo=0)                   0.000     0.000    A
    T19                  IBUF (Prop_ibuf_I_O)         0.950     0.950 r  A_IBUF_inst/O
                         net (fo=2, routed)           1.147     2.097    A_IBUF
    SLICE_X43Y1          LUT2 (Prop_lut2_I1_O)        0.150     2.247 r  sum_OBUF_inst_i_1/O
                         net (fo=1, routed)           1.816     4.063    sum_OBUF
    P18                  OBUF (Prop_obuf_I_O)         2.837     6.900 r  sum_OBUF_inst/O
                         net (fo=0)                   0.000     6.900    sum
    P18                                                          r  sum (OUT)
  -------------------------------------------------------------------    -------------------
```

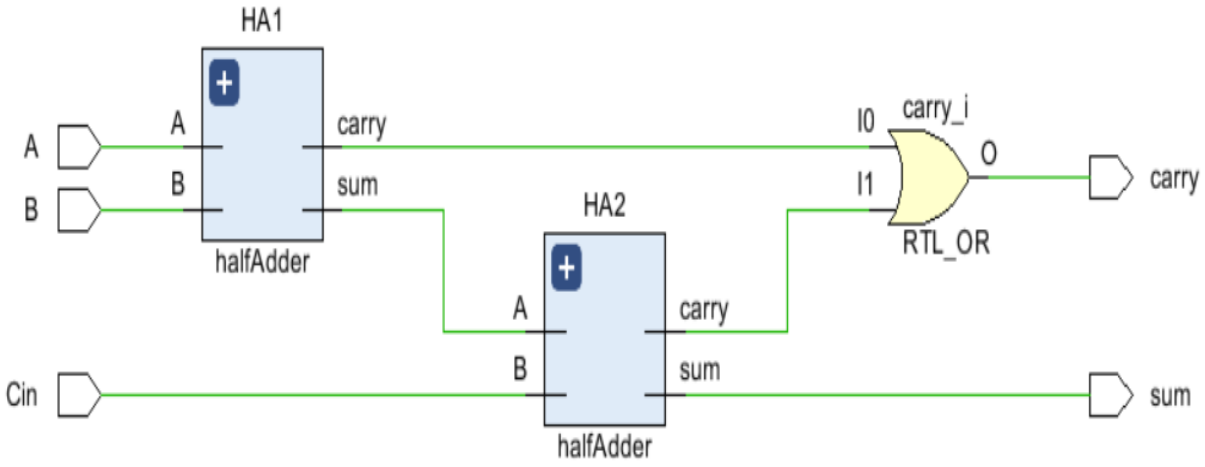
Ερώτημα 2

Τώρα βασιζόμενοι στον HA του πρώτου ερωτήματος θα δημιουργήσουμε έναν πλήρη αθροιστή FA.

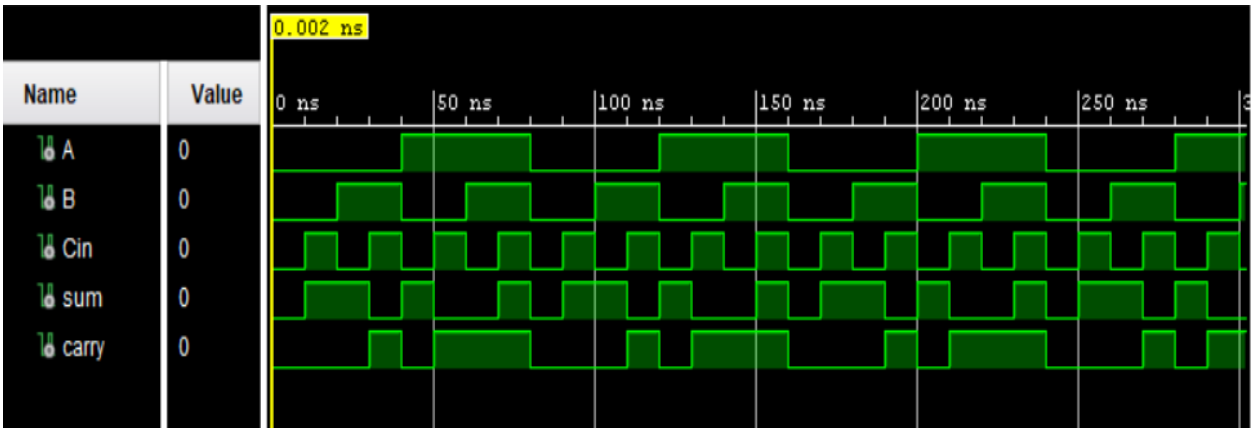
- Κώδικας για την αρχιτεκτονική
  Προσθέτουμε τον HA ως component, και χρησιμοποιώντας 2 HAs και λογικές πύλες έχουμε το ζητούμενο αποτέλεσμα.

```
28
29  entity FullAdder is
30    Port (
31    A,B,Cin: in std_logic ;
32    sum,carry: out std_logic
33     );
34  end  FullAdder;
35
36  architecture structural of FullAdder is
37
38  component haldAdder ...
46
47  signal s1,c1,c2 : std_logic := '0';
48
49  begin
50  HA1 : haldAdder port map (A,B,s1,c1);
51  HA2 : haldAdder port map (s1,Cin,sum,c2);
52  carry <= c1 OR c2;
53  end architecture;
54
```

- RTL Schematic:



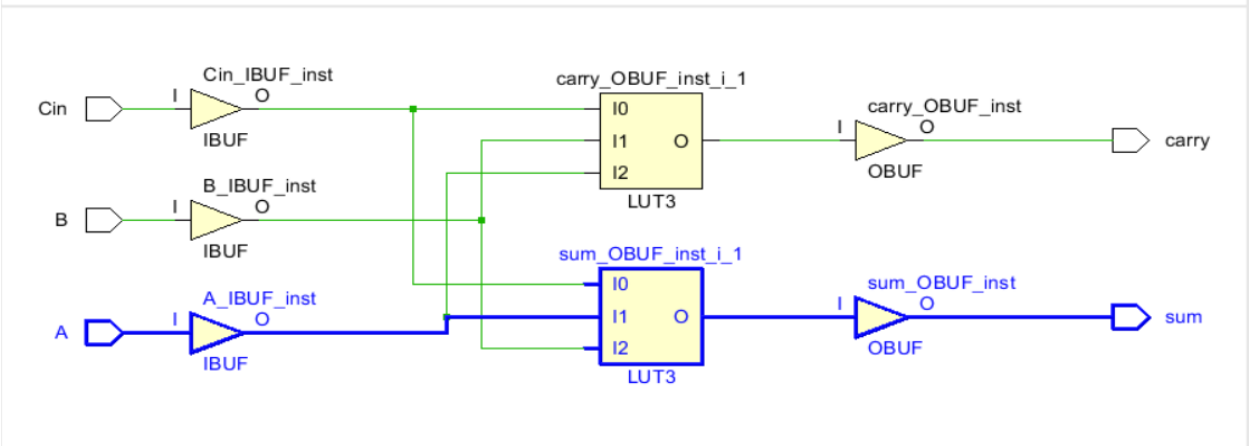
- Αποτελέσματα Testbench:



Τα αποτελέσματα του testbench συμφωνούν με τον πίνακα αλήθειας του FA

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- Critical Path: A → Sum (7.125ns)



Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	3	2	2	A	sum	7.125	3.947	3.177
Path 2	∞	3	2	2	A	carry	6.893	3.709	3.184

Το μονοπάτι με την μεγαλύτερη καθυστέρηση είναι το Path-1, δηλαδή το μονοπάτι από την είσοδο A μέχρι την έξοδο sum.

Πιο αναλυτικά έχουμε:

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack:                  inf
  Source:               A
                          (input port)
  Destination:          sum
                          (output port)
  Path Group:           (none)
  Path Type:            Max at Slow Process Corner
  Data Path Delay:      7.125ns  (logic 3.947ns (55.403%)  route 3.177ns (44.597%))
  Logic Levels:         3  (IBUF=1 LUT3=1 OBUF=1)

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
    T19                                               0.000     0.000 r  A (IN)
                         net (fo=0)                   0.000     0.000    A
    T19                  IBUF (Prop_ibuf_I_O)         0.950     0.950 r  A_IBUF_inst/O
                         net (fo=2, routed)           1.370     2.320    A_IBUF
    SLICE_X43Y1          LUT3 (Prop_lut3_I1_O)        0.152     2.472 r  sum_OBUF_inst_i_1/O
                         net (fo=1, routed)           1.808     4.279    sum_OBUF
    N17                  OBUF (Prop_obuf_I_O)         2.845     7.125 r  sum_OBUF_inst/O
                         net (fo=0)                   0.000     7.125    sum
    N17                                                             r  sum (OUT)
  -------------------------------------------------------------------    -------------------
```

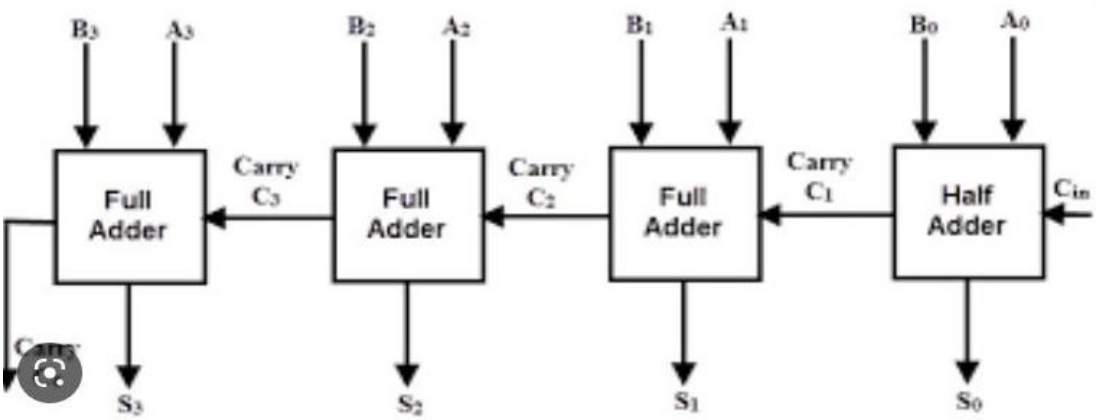
Ερώτημα 3:

Τώρα βασιζόμενοι στους FA που φτιάξαμε και χρησιμοποιώντας τους ως components θα φτιάξουμε ένα πλήρη αθροιστή 4 bits.

- Κώδικας για την αρχιτεκτονική:

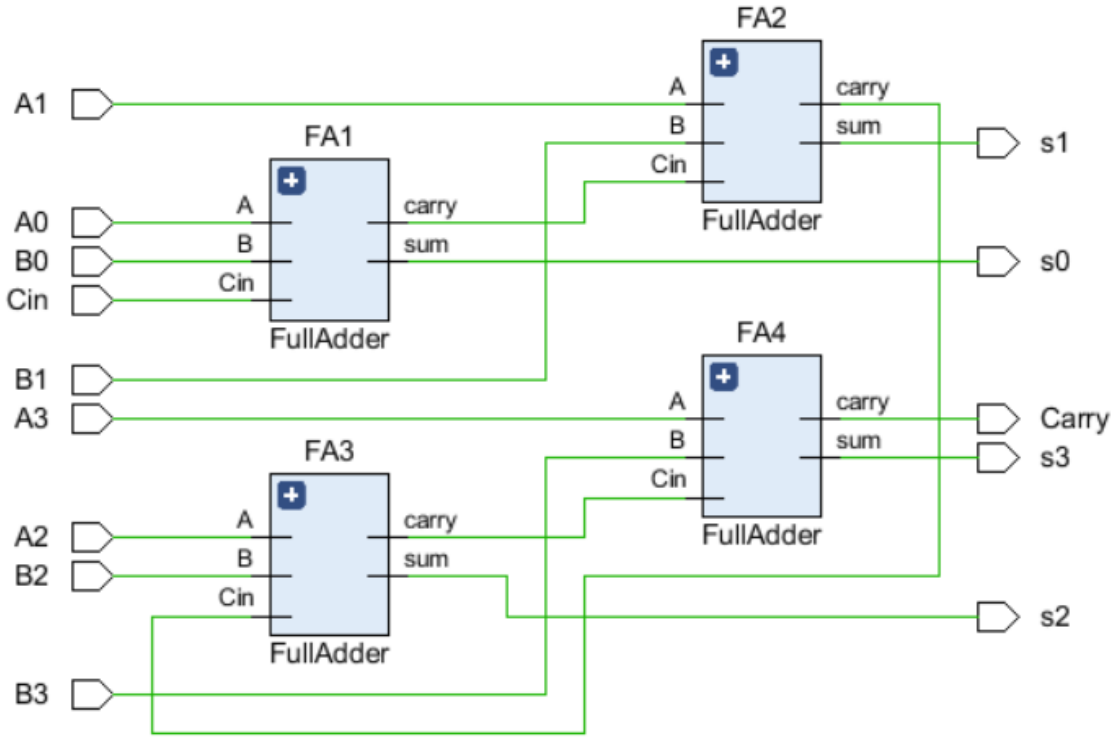
```
55  entity bit4_Adder is
56    Port (
57    As: in std_logic_vector(3 downto 0);
58    Bs: in std_logic_vector(3 downto 0);
59    Cin: in std_logic;
60    s: out std_logic_vector(3 downto 0);
61    cout: out std_logic
62     );
63  end  bit4_Adder;
64
65  architecture structural of bit4_Adder is
66
67  component FullAdder ...
76
77  signal c0: std_logic;
78  signal c1: std_logic;
79  signal c2: std_logic;
80
81  begin
82  FA1 : FullAdder  port map (A=>As(0),B=>Bs(0),Cin=>Cin,sum=>s(0),carry=>c0);
83  FA2 : FullAdder  port map (A=>As(1),B=>Bs(1),Cin=>c0,sum=>s(1),carry=>c1);
84  FA3 : FullAdder  port map (A=>As(2),B=>Bs(2),Cin=>c1,sum=>s(2),carry=>c2);
85  FA4 : FullAdder  port map (A=>As(3),B=>Bs(3),Cin=>c2,sum=>s(3),carry=>cout);
86
87  end architecture;
```

Ουσιαστικά συνδέουμε 4 FAs σε σειρά και δίνουμε στον καθένα ως κρατούμενο εισόδου την έξοδο του ακριβώς προηγούμενου πλήρη αθροιστή. Περιμένουμε λοιπόν να δούμε ένα λογικό κύκλωμα όπως αυτό που φαίνεται παρακάτω.



- RTL Schematic:

Το RTL schematic λοιπόν συμφωνεί με την υλοποίηση που αναμέναμε. Απομένει λοιπόν να επιβεβαιώσουμε ότι έχουμε σωστά αποτελέσματα μέσω προσομοίωσης. Αξίζει να σημειώσουμε ότι στο schematic φαίνεται ξεκάθαρα η structural λογική που επιλέξαμε καθώς χρησιμοποιούμε τους FA ως black boxes.

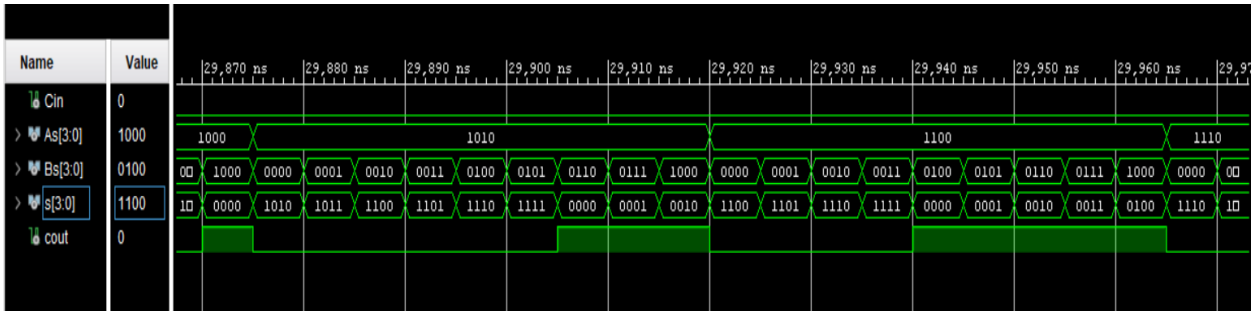


Στην συνέχεια γράψαμε κώδικα για την προσομοίωση ο οποίος με την χρήση εμφολευμένων for loops δοκιμάζει διάφορες τιμές για τις εισόδους. Ένα κομμάτι της λογικής του κώδικα φαίνεται στην συνέχεια.

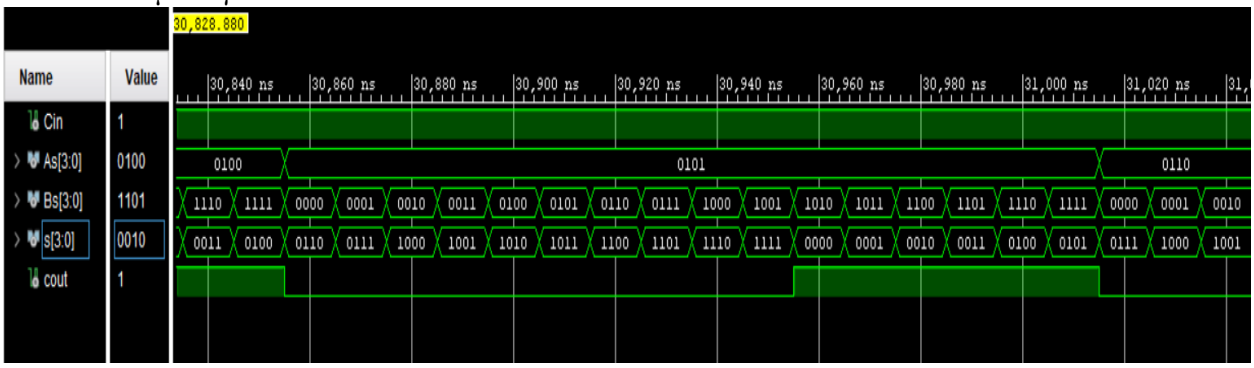
```
Cin<= '0';
     for i in 0 to 15 loop
         As <= std_logic_vector(to_unsigned(i, 4));
         for j in 0 to 15 loop
             Bs <= std_logic_vector(to_unsigned(j, 4));
             wait for 5ns;
          end loop;
    end loop;
```

- Αποτελέσματα Testbench:

  Τα αποτελέσματα για Cin =0:

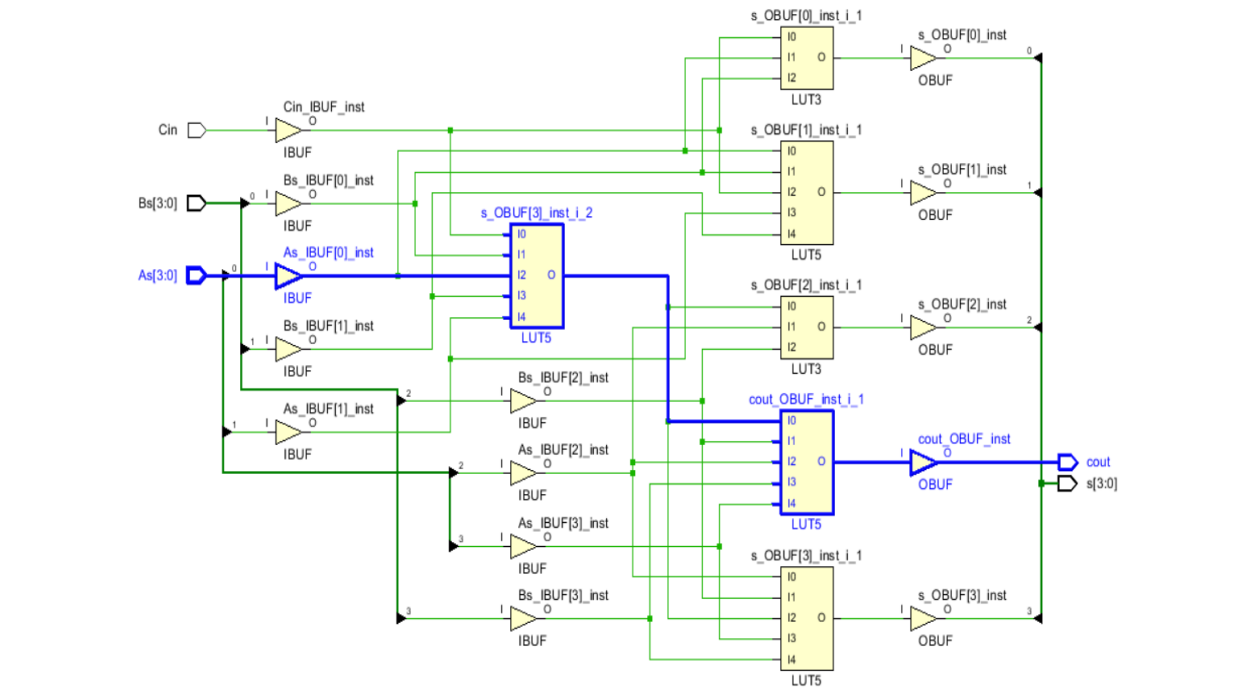


Τα αποτελέσματα για Cin =1:



*Σημειώνουμε ότι όταν cout =1 έχουμε ουσιαστικά μια επιπλέον δεκάδα.

- Critical Path: A[3:0] → Cout (delay = 7.96ns)



Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	4	3	3	As[0]	cout	7.960	4.050	3.911
Path 2	∞	4	3	3	As[0]	s[3]	7.677	3.797	3.880
Path 3	∞	3	2	3	As[0]	s[1]	7.378	3.915	3.462
Path 4	∞	4	3	3	As[0]	s[2]	7.372	3.792	3.579
Path 5	∞	3	2	3	As[0]	s[0]	7.092	3.681	3.411

Ερώτημα 4:

Χρησιμοποιώντας λοιπόν τον 4bit FA του προηγούμενου ερωτήματος θα φτιάξουμε έναν BCD FA.

Υπενθυμίζουμε ότι :

Ένας BCD full adder είναι ένα κύκλωμα που χρησιμοποιείται για να προσθέσει δύο BCD (Binary-Coded Decimal) αριθμούς, δηλαδή αριθμούς που έχουν κωδικοποιηθεί σε δυαδική μορφή, ώστε να αντιπροσωπεύσουν δεκαδικούς αριθμούς.

Σε έναν BCD full adder, κάθε αριθμός BCD διαιρείται σε τέσσερα δυαδικά ψηφία, τα οποία προστίθενται μεταξύ τους σύμφωνα με τους κανόνες της πρόσθεσης δυαδικών αριθμών. Το κυκλώματος προσθέτει επίσης τυχόν υπόλοιπα (carry) από τις προηγούμενες προσθέσεις και παράγει το άθροισμα BCD των δύο αριθμών εισόδου.

Η έξοδος του κυκλώματος είναι ένας BCD αριθμός, δηλαδή μια σειρά τεσσάρων δυαδικών ψηφίων που αντιπροσωπεύουν ένα δεκαδικό ψηφίο. Στον κάτωθι πίνακα, αναδεικνύεται η διαφορά στην αναπαράσταση ενός δεκαδικού αριθμού με binary και με BCD μορφή.

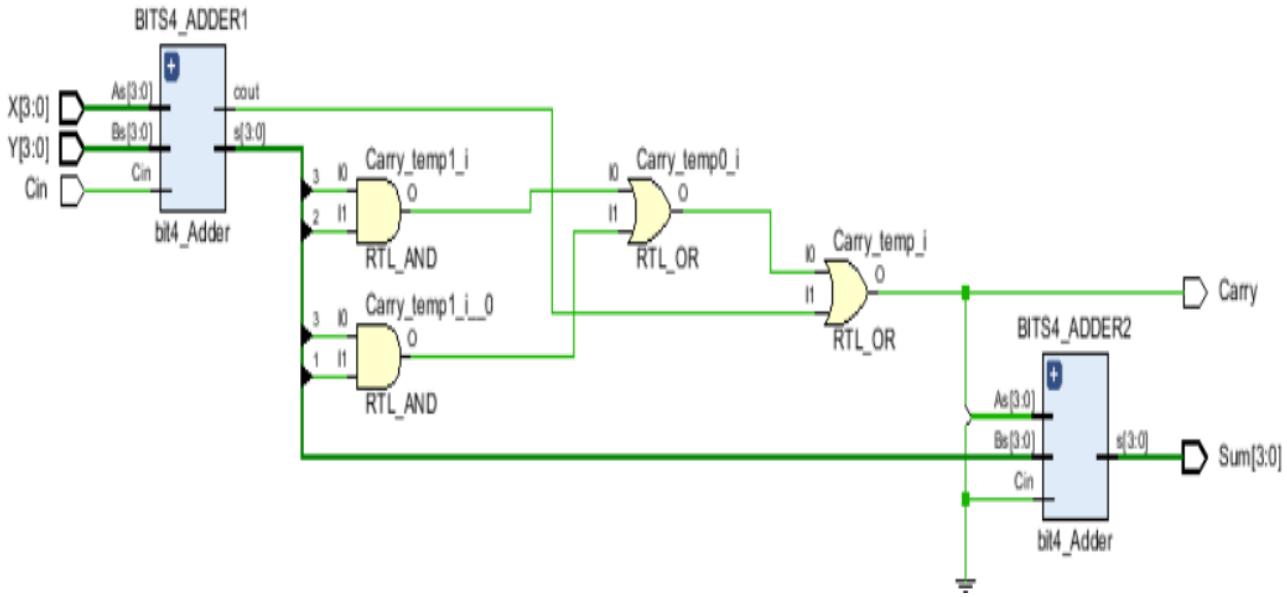
Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

Κώδικας για την αρχιτεκτονική:

Χρησιμοποιώντας ως component τον 4bit-FA έχουμε

```
93  entity BCD_Adder is
94    Port (
95    X : in std_logic_vector ( 3 downto 0) ;
96    Y : in std_logic_vector ( 3 downto 0) ;
97    Cin : in std_logic;
98    Sum  :  out std_logic_vector ( 3 downto 0) ;
99    Carry : out std_logic
100    );
101 end BCD_Adder;
102
103 architecture structural of BCD_Adder is
104
105 component  bit4_Adder...
114
115 signal sum_up, X_down,Y_down: std_logic_vector(3 downto 0) ;
116 signal carry_up,Carry_temp, carry_down: std_logic;
117 begin
118
119  BITS4_ADDER1 :  bit4_Adder  port map (As => X, Bs => Y, Cin => Cin, s => sum_up, cout => carry_up);
120  Carry_temp <= (sum_up(3) and sum_up(2))  or (sum_up(3) and sum_up(1)) or carry_up ;
121  Carry <= Carry_temp;
122  X_down <= '0' & Carry_temp & Carry_temp & '0';
123  BITS4_ADDER2 :  bit4_Adder  port map (As => X_down, Bs => sum_up, Cin => '0', s => Sum, cout => carry_down);
124
125 end architecture;
126
```

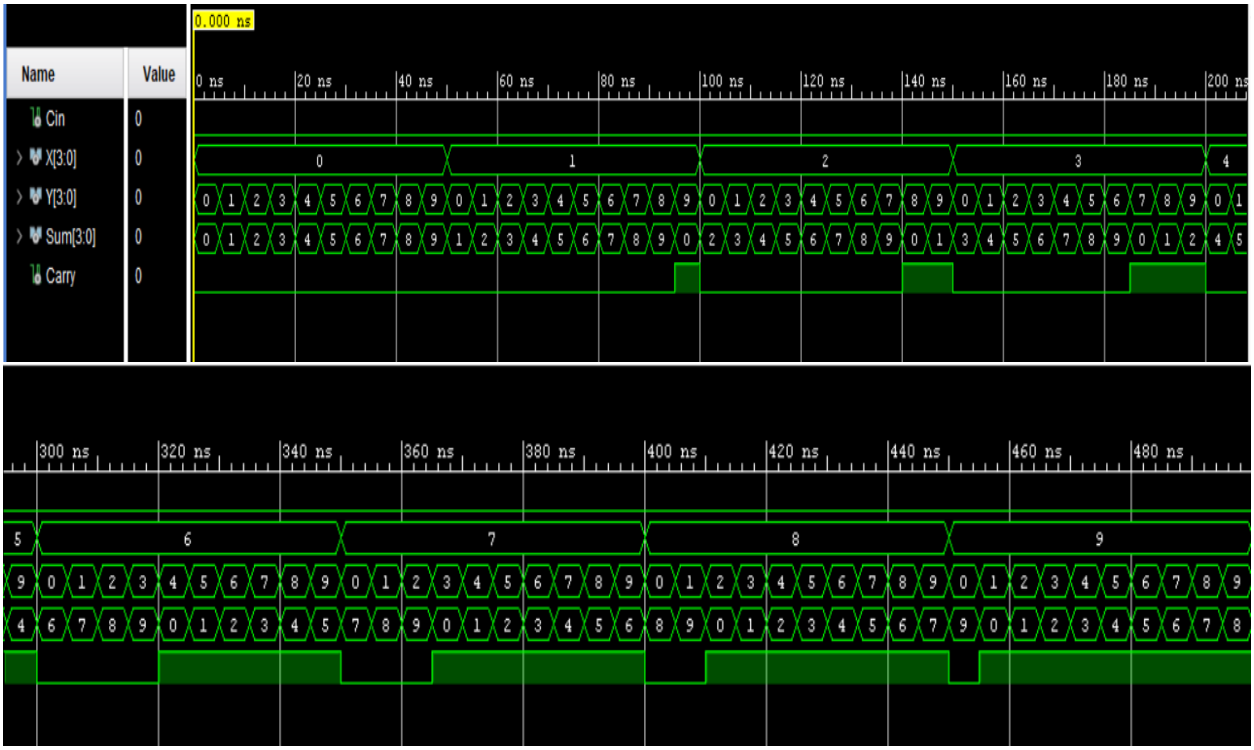
- RTL Schematic:



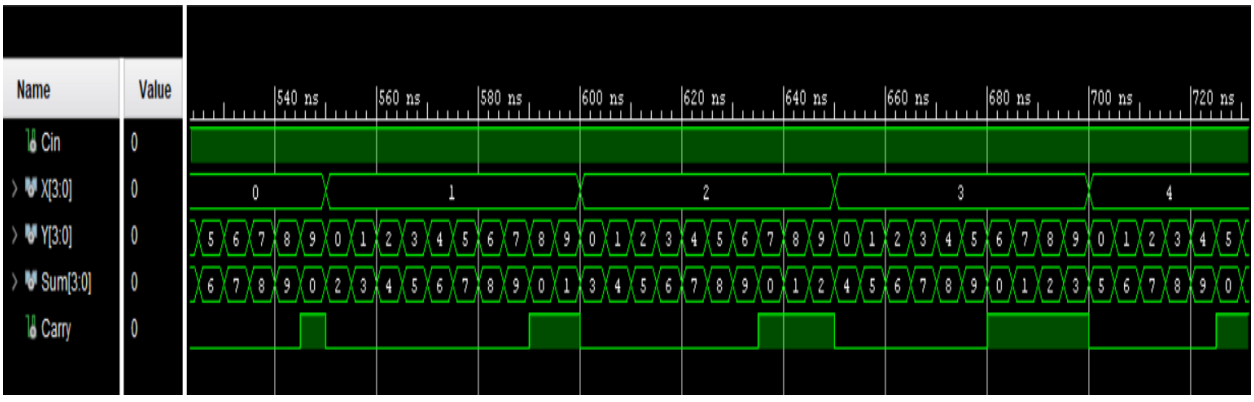
Από το schematic γίνεται φανερό ότι όταν το άθροισμα των δύο BCD 4bit αριθμών είναι μικρότερο ίσο του εννέα τότε αυτό περνά ως έξοδος του BCD, διαφορετικά μέσω του output carry γίνεται μια “κανονικοποίηση” για αναπαράσταση του αριθμού στο δεκαδικό σύστημα.

- Αποτελέσματα Testbench:
  Στην συνέχεια μέσω ενός tenchbench code με for loops, τρέχουμε προσομοίωση για εισαγωγή bcd αριθμών μέχρι το εννέα.(έτσι ζητήθηκε)

  Για Cin = 0:

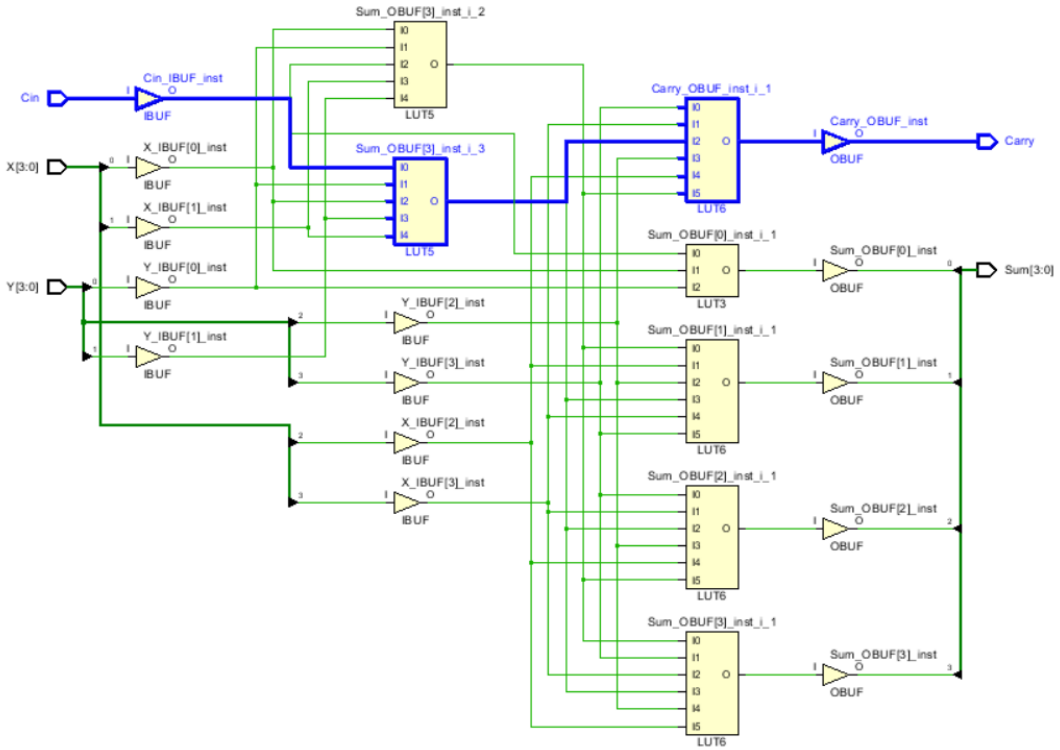


  Για Cin=1:



  Σημειώνεται ότι το Carry = 1 αναπαριστά μια δεκάδα.

- Critical Path: Cin -> Carry (delay = 8.160ns)



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	4	3	4	Cin	Carry	8.160	4.083	4.078
Path 2	∞	4	3	4	Cin	Sum[3]	8.053	4.074	3.979
Path 3	∞	4	3	4	Cin	Sum[2]	7.801	3.808	3.994
Path 4	∞	4	3	4	Cin	Sum[1]	7.678	4.040	3.638
Path 5	∞	3	2	3	Cin	Sum[0]	7.131	3.698	3.434

Το κρίσιμο μονοπάτι είναι η διαδρομή από την είσοδο Cin μέχρι να παραχθεί το κρατούμενο Carry.

Πιο αναλυτικά:

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack:                  inf
  Source:               Cin
                          (input port)
  Destination:          Carry
                          (output port)
  Path Group:           (none)
  Path Type:            Max at Slow Process Corner
  Data Path Delay:      8.160ns  (logic 4.083ns (50.029%)  route 4.078ns (49.971%))
  Logic Levels:         4  (IBUF=1 LUT5=1 LUT6=1 OBUF=1)

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
    W16                                               0.000     0.000 r  Cin (IN)
                         net (fo=0)                   0.000     0.000    Cin
    W16                  IBUF (Prop_ibuf_I_O)         0.961     0.961 r  Cin_IBUF_inst/O
                         net (fo=3, routed)           1.583     2.544    Cin_IBUF
    SLICE_X43Y6          LUT5 (Prop_lut5_I0_O)        0.152     2.696 r  Sum_OBUF[3]_inst_i_3/O
                         net (fo=4, routed)           0.685     3.381    BITS4_ADDER1/c1
    SLICE_X42Y6          LUT6 (Prop_lut6_I2_O)        0.326     3.707 r  Carry_OBUF_inst_i_1/O
                         net (fo=1, routed)           1.810     5.517    Carry_OBUF
    N17                  OBUF (Prop_obuf_I_O)         2.643     8.160 r  Carry_OBUF_inst/O
                         net (fo=0)                   0.000     8.160    Carry
    N17                                                               r  Carry (OUT)
  -------------------------------------------------------------------    -------------------
```

Ερώτημα 5:

Ο BCD παράλληλος αθροιστής του ερωτήματος 5 αποτελείται από 4 4-bit BCD Full Adders σειριακά συνδεδεμένους μεταξύ τους. Έτσι, οι είσοδοι A και B του παράλληλου αθροιστή είναι των 16 bits, με 4 bits να αναλογούν σε κάθε επιμέρους 4-bit BCD Full Adder.

Στην συνέχεια παραδίδουμε τον κώδικα όλης της εργασίας συνοπτικά μαζεμένο.

Κώδικας για την αρχιτεκτονική:

```
-- half Adder (1)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity haldAdder is
  Port (
  A: in std_logic ;
  B: in std_logic ;
  sum: out std_logic ;
  carry: out std_logic
   );
end haldAdder;

architecture dataflow of haldAdder is
begin
carry <= A AND B ;
sum <= A XOR B;
end architecture;
```

```
-- Full Adder (2)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FullAdder is
  Port (
  A,B,Cin: in std_logic ;
  sum,carry: out std_logic
   );
end  FullAdder;

architecture structural of FullAdder is

component haldAdder
 port (
  A: in std_logic;
```

```
  B: in std_logic;
  sum: out std_logic;
  carry:out std_logic
  );
end component;

signal s1,c1,c2 : std_logic := '0';

begin
HA1 : haldAdder port map (A,B,s1,c1);
HA2 : haldAdder port map (s1,Cin,sum,c2);
carry <= c1 OR c2;
end architecture;
```

```
-- 4-bit Parallel Adder (3)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity bit4_Adder is
  Port (
  As: in std_logic_vector(3 downto 0);
  Bs: in std_logic_vector(3 downto 0);
  Cin: in std_logic;
  s: out std_logic_vector(3 downto 0);
  cout: out std_logic
   );
end  bit4_Adder;

architecture structural of bit4_Adder is

component FullAdder
  port (
  A: in std_logic;
  B: in std_logic;
  Cin: in std_logic;
  sum: out std_logic;
  carry:out std_logic
  );
```

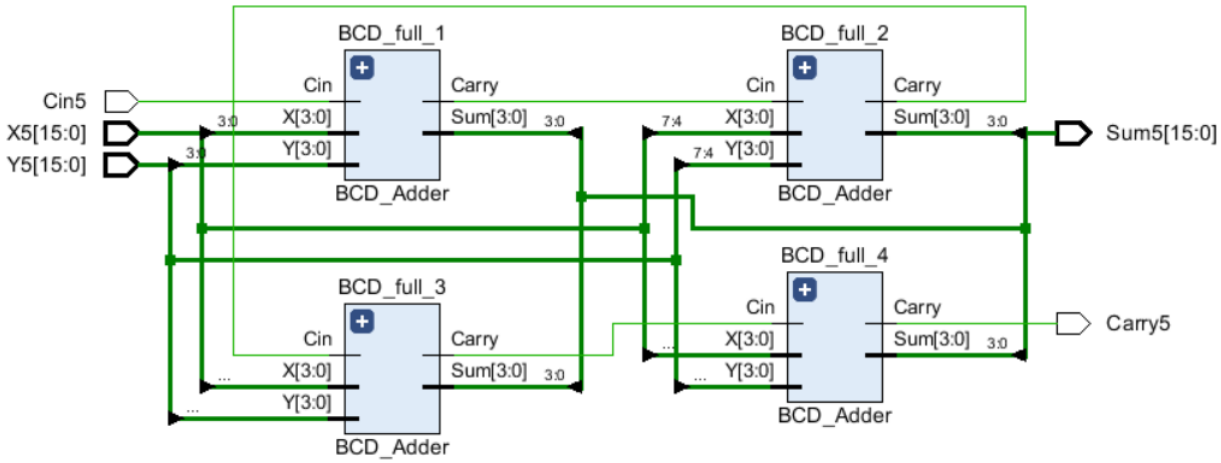
```
end component;

signal c0: std_logic;
signal c1: std_logic;
signal c2: std_logic;


begin
```

```
FA1 : FullAdder  port map (A=>As(0),B=>Bs(0),Cin=>Cin,sum=>s(0),carry=>c0);
FA2 : FullAdder  port map (A=>As(1),B=>Bs(1),Cin=>c0,sum=>s(1),carry=>c1);
FA3 : FullAdder  port map (A=>As(2),B=>Bs(2),Cin=>c1,sum=>s(2),carry=>c2);
FA4 : FullAdder  port map (A=>As(3),B=>Bs(3),Cin=>c2,sum=>s(3),carry=>cout);

end architecture;
```

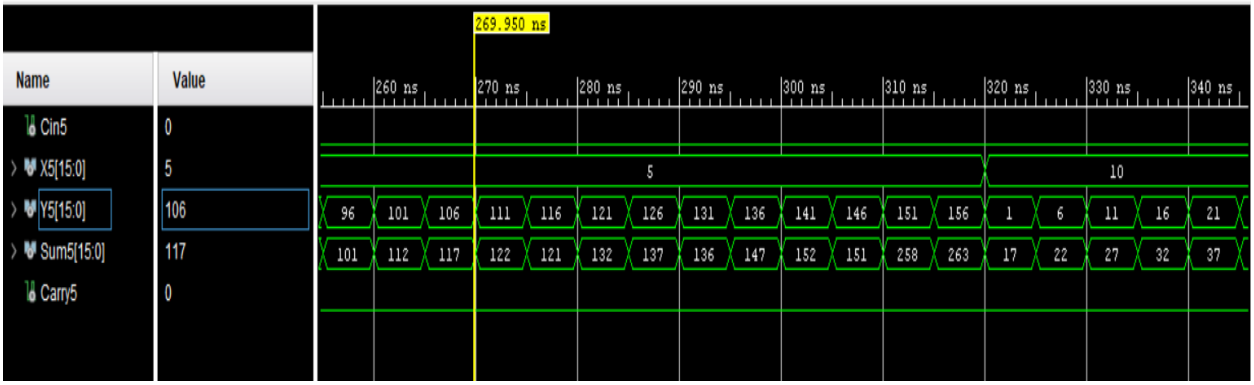
```
-- BCD Adder (4)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity BCD_Adder is

  Port (
  X : in std_logic_vector ( 3 downto 0) ;
  Y : in std_logic_vector ( 3 downto 0) ;
  Cin : in std_logic;
  Sum  :  out std_logic_vector ( 3 downto 0) ;
  Carry : out std_logic
   );
end BCD_Adder;
architecture structural of BCD_Adder is
component  bit4_Adder
  port (
    As: in std_logic_vector(3 downto 0);
    Bs: in std_logic_vector(3 downto 0);
    Cin: in std_logic;
    s: out std_logic_vector(3 downto 0);
    cout: out std_logic
  );
end component;

signal sum_up, X_down,Y_down: std_logic_vector(3 downto 0) ;
signal carry_up,Carry_temp, carry_down: std_logic;
begin
 BITS4_ADDER1 :  bit4_Adder  port map (As => X, Bs => Y, Cin => Cin, s => sum_up, cout
=> carry_up);
 Carry_temp <= (sum_up(3) and sum_up(2))  or (sum_up(3) and sum_up(1)) or carry_up ;
 Carry <= Carry_temp;
 X_down <= '0' & Carry_temp & Carry_temp & '0';

 BITS4_ADDER2 :  bit4_Adder  port map (As => X_down, Bs => sum_up, Cin => '0', s => Sum,
cout => carry_down);

end architecture;
```

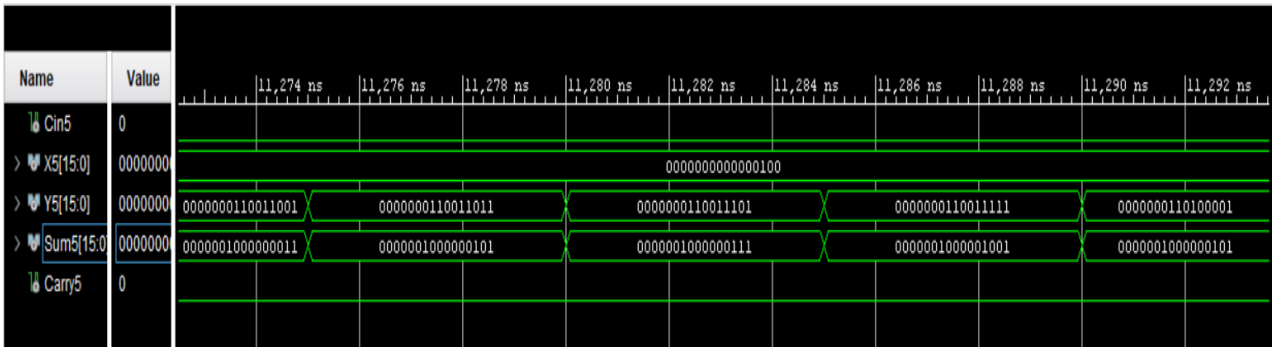
```
--BCD parallel adder(5)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lab02_ex05 is
Port (
  X5 : in std_logic_vector ( 15 downto 0) ;
  Y5 : in std_logic_vector ( 15 downto 0) ;
  Cin5 : in std_logic;
  Sum5  :  out std_logic_vector ( 15 downto 0) ;
  Carry5 : out std_logic
   );
end lab02_ex05;

architecture Structural of lab02_ex05 is

component BCD_Adder
  Port (
  X : in std_logic_vector ( 3 downto 0) ;
  Y : in std_logic_vector ( 3 downto 0) ;
  Cin : in std_logic;
  Sum  :  out std_logic_vector ( 3 downto 0) ;
  Carry : out std_logic
   );
end component;

signal s1,s2,s3,s4,x_temp1,y_temp1,x_temp2,y_temp2,x_temp3,y_temp3,x_temp4,y_temp4:
std_logic_vector(3 downto 0);
signal c51,c52,c53: std_logic;

begin

x_temp1 <= (X5(3)&X5(2)&X5(1)&X5(0));
y_temp1 <= (Y5(3)&Y5(2)&Y5(1)&Y5(0));
BCD_full_1: BCD_Adder port map(X => x_temp1, Y => y_temp1, Cin=>Cin5, Sum=>s1, Carry=>c51);
x_temp2 <= (X5(7)&X5(6)&X5(5)&X5(4));
y_temp2 <= (Y5(7)&Y5(6)&Y5(5)&Y5(4));
BCD_full_2: BCD_Adder port map(X => x_temp2, Y => y_temp2, Cin=>c51, Sum=>s2, Carry=>c52);
x_temp3 <= (X5(11)&X5(10)&X5(9)&X5(8));
y_temp3 <= (Y5(11)&Y5(10)&Y5(9)&Y5(8));
BCD_full_3: BCD_Adder port map(X => x_temp3, Y => y_temp3, Cin=>c52, Sum=>s3, Carry=>c53);
x_temp4 <= (X5(15)&X5(14)&X5(13)&X5(12));
y_temp4 <= (Y5(15)&Y5(14)&Y5(13)&Y5(12));
BCD_full_4: BCD_Adder port map(X => x_temp4, Y => y_temp4, Cin=>c53, Sum=>s4, Carry=>Carry5);
Sum5 <= s4&s3&s2&s1;

end architecture;
```

- RTL Schematic:

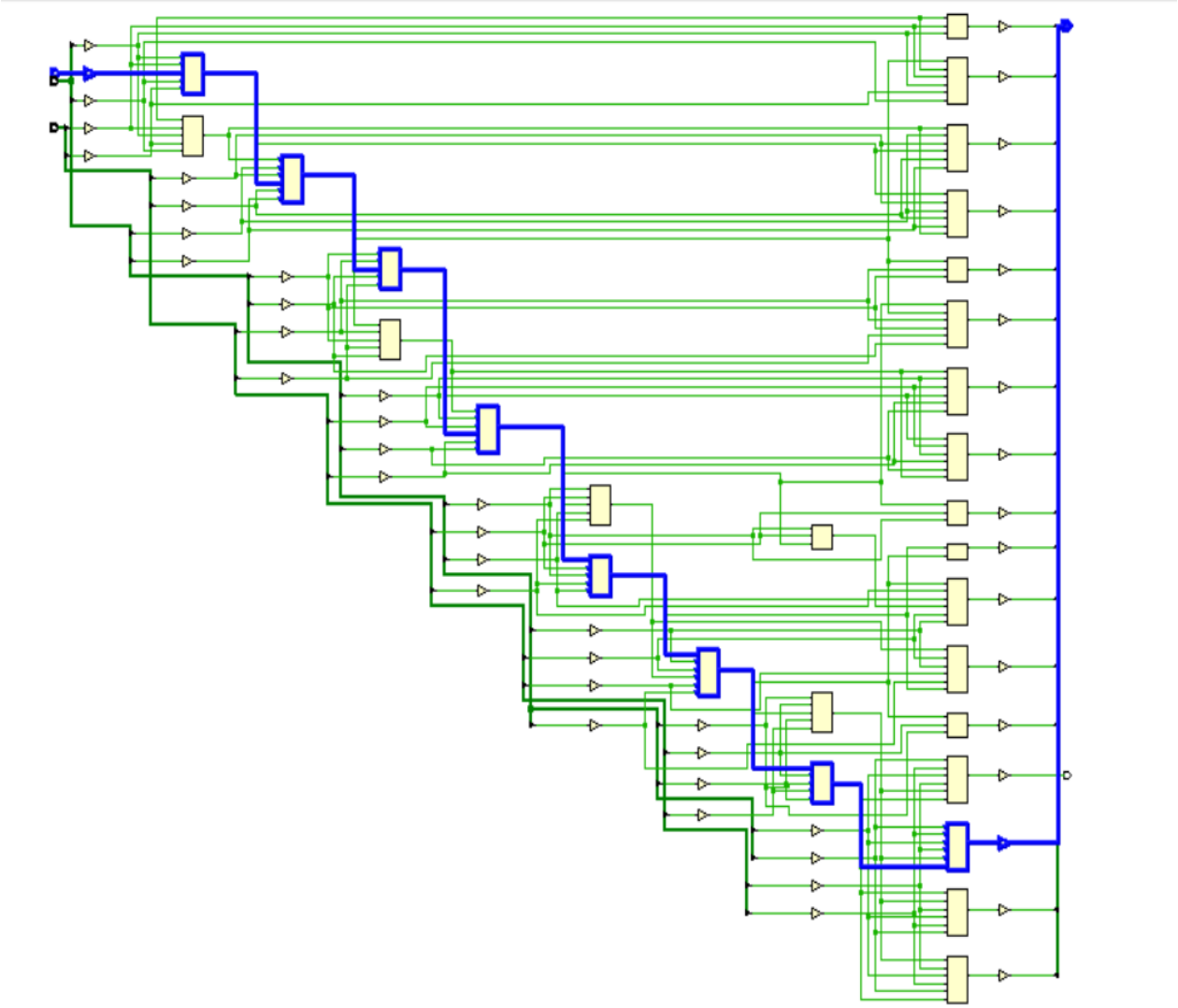


- Αποτελέσματα testbench:
  Γράφοντας κώδικα με for loops για να έχουμε όλες τις περιπτώσεις εισόδων έχουμε τα παρακάτω αποτελέσματα στη προσομοίωση.





- Critical Path: Cin5 -> Sum5[13] (delay = 15.265ns)



Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	10	9	5	Cin5	Sum5[13]	15.265	5.284	9.981
Path 2	∞	10	9	5	Cin5	Sum5[15]	15.210	5.070	10.140
Path 3	∞	10	9	5	Cin5	Carry5	15.105	5.307	9.798
Path 4	∞	10	9	5	Cin5	Sum5[14]	15.001	5.049	9.951
Path 5	∞	9	8	5	Cin5	Sum5[12]	14.385	5.135	9.250
Path 6	∞	9	8	5	Cin5	Sum5[10]	14.170	4.912	9.258
Path 7	∞	9	8	5	Cin5	Sum5[9]	14.147	4.913	9.234
Path 8	∞	8	7	5	Cin5	Sum5[11]	13.320	4.777	8.543
Path 9	∞	7	6	5	Cin5	Sum5[8]	12.570	4.908	7.662
Path 10	∞	7	6	5	Cin5	Sum5[5]	11.348	4.678	6.671

Το κρίσιμο μονοπάτι είναι το μονοπάτι από την είσοδο cin5 ως την έξοδο Sum5[13].

Πιο αναλυτικά :

```
Destination:            Sum5[13]
                          (output port)
Path Group:             (none)
Path Type:              Max at Slow Process Corner
Data Path Delay:        15.265ns  (logic 5.284ns (34.617%)  route 9.981ns (65.383%))
Logic Levels:           10  (IBUF=1 LUT5=4 LUT6=4 OBUF=1)

  Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------------------------------    -------------------
  R19                                               0.000     0.000 r  Cin5 (IN)
                       net (fo=0)                   0.000     0.000    Cin5
  R19                  IBUF (Prop_ibuf_I_O)         0.982     0.982 r  Cin5_IBUF_inst/O
                       net (fo=4, routed)           2.056     3.038    Cin5_IBUF
  SLICE_X43Y19         LUT5 (Prop_lut5_I2_O)        0.150     3.188 r  Sum5_OBUF[3]_inst_i_2/O
                       net (fo=3, routed)           0.668     3.856    Sum5_OBUF[3]_inst_i_2_n_0
  SLICE_X42Y19         LUT6 (Prop_lut6_I3_O)        0.326     4.182 r  Sum5_OBUF[5]_inst_i_2/O
                       net (fo=5, routed)           0.804     4.985    Sum5_OBUF[5]_inst_i_2_n_0
  SLICE_X43Y22         LUT5 (Prop_lut5_I2_O)        0.150     5.135 r  Sum5_OBUF[7]_inst_i_2/O
                       net (fo=3, routed)           0.668     5.803    Sum5_OBUF[7]_inst_i_2_n_0
  SLICE_X42Y22         LUT6 (Prop_lut6_I3_O)        0.326     6.129 r  Sum5_OBUF[8]_inst_i_2/O
                       net (fo=5, routed)           0.988     7.117    Sum5_OBUF[8]_inst_i_2_n_0
  SLICE_X42Y26         LUT5 (Prop_lut5_I0_O)        0.124     7.241 f  Sum5_OBUF[11]_inst_i_3/O
                       net (fo=3, routed)           1.072     8.313    Sum5_OBUF[11]_inst_i_3_n_0
  SLICE_X43Y26         LUT6 (Prop_lut6_I0_O)        0.124     8.437 r  Sum5_OBUF[12]_inst_i_2/O
                       net (fo=5, routed)           0.971     9.408    Sum5_OBUF[12]_inst_i_2_n_0
  SLICE_X43Y29         LUT5 (Prop_lut5_I0_O)        0.152     9.560 r  Sum5_OBUF[15]_inst_i_3/O
                       net (fo=4, routed)           0.602    10.162    Sum5_OBUF[15]_inst_i_3_n_0
  SLICE_X43Y30         LUT6 (Prop_lut6_I5_O)        0.326    10.488 r  Sum5_OBUF[13]_inst_i_1/O
                       net (fo=1, routed)           2.153    12.641    Sum5_OBUF[13]
  W16                  OBUF (Prop_obuf_I_O)         2.624    15.265 r  Sum5_OBUF[13]_inst/O
                       net (fo=0)                   0.000    15.265    Sum5[13]
  W16                                                             r  Sum5[13] (OUT)
-------------------------------------------------------------------    -------------------
```