Εθνικό Μετσόβιο Πολυτεχνείο, ΣΗΜΜΥ
Ψηφιακά Συστήματα VLSI

2η Εργαστηριακή Άσκηση:
Σχεδιασμός Αριθμητικών Μονάδων με Ιεραρχική Σχεδίαση

Ημ/νια : 13.04.2024
Ομάδα 19
Θεοδώρα Εξάρχου, ΑΜ : 03120865
Παναγιώτης Μπέλσης, ΑΜ: 03120874

1) Υλοποιήστε έναν Ημιαθροιστή (Half Adder - HA) σε περιγραφή ροής δεδομένων (Dataflow).

```
----------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity halfadder is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Sum : out STD_LOGIC;
           Carry : out STD_LOGIC);
end halfadder;

architecture Dataflow of halfadder is
begin

Sum <= A xor B;
Carry <= A and B;

end Dataflow;
----------------------------------------------------------------------------------------------------------------------
```

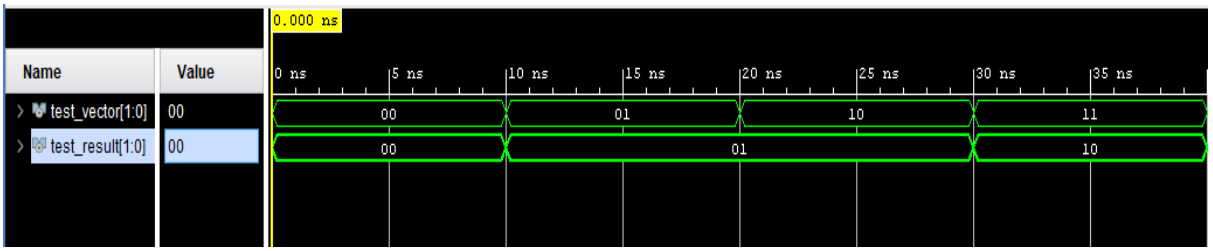
Το testbench αρχείο που χρησιμοποιήσαμε είναι το εξής:

—----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity testbench is
--  Port ( );
end testbench;

architecture arch_testbench of testbench is
    component halfadder
    port(
    A: in std_logic;
    B: in std_logic;
    Sum: out std_logic;
    Carry: out std_logic);
    end component;

    signal test_vector: std_logic_vector(1 downto 0);
    signal test_result : std_logic_vector(1 downto 0);
begin
uut: halfadder
port map(
    A=>test_vector(0),
    B=>test_vector(1),
    Sum=>test_result(0),
    Carry=>test_result(1)
);
testing: process
begin
test_vector<="00";
wait for 10ns;
test_vector<="01";
wait for 10ns;
test_vector<="10";
wait for 10ns;
test_vector<="11";
wait for 10ns;
```

```
end process;

end arch_testbench;
```

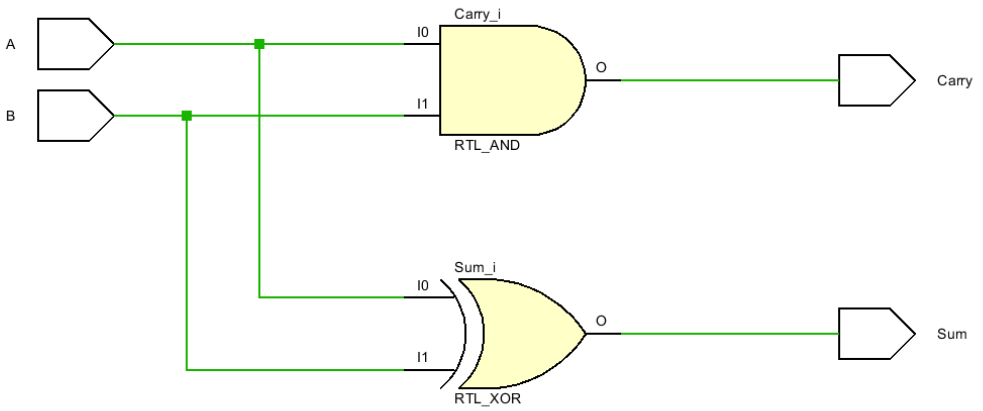
Το simulation που προκύπτει από το testbench επιβεβαιώνει την ορθή λειτουργία του halfadder.



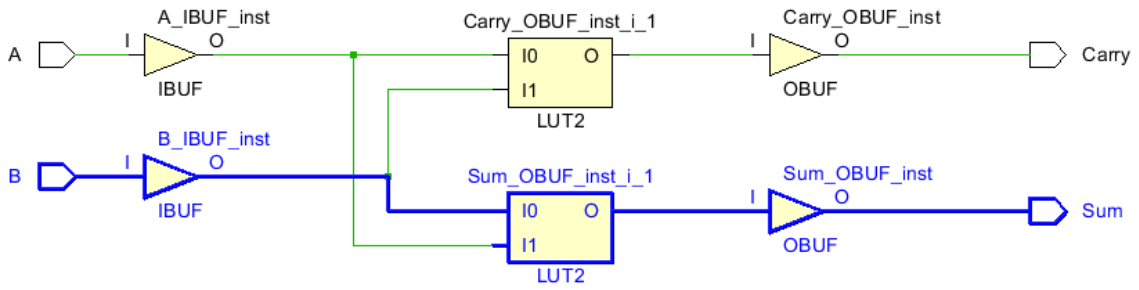
HalfAdder truth table:

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

Παρουσίαση δομικού διαγράμματος (RTL schematic) της αρχιτεκτονικής του κυκλώματος:



Εύρεση κρίσιμου μονοπατιού (Critical Path) του κυκλώματος, καθώς και της χρονικής του καθυστέρησης:



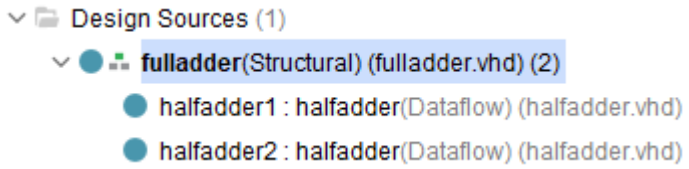
Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	∞	3	4	2	B	Sum	5.377	3.778	1.599	∞	input port clock			0.000
Path 2	∞	3	4	2	B	Carry	5.351	3.752	1.599	∞	input port clock			0.000

Παρατηρούμε ότι το μονοπάτι με την μεγαλύτερη καθυστέρηση είναι το μονοπάτι του Sum, δηλαδή η XOR πύλη, με συνολική καθυστέρηση **5.377 ns**.

2) Υλοποιήστε έναν Πλήρη Αθροιστή (Full Adder - FA) με περιγραφή δομής (Structural), βασιζόμενοι στην δομική μονάδα του Ερωτήματος 1.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fulladder is
    Port ( Af : in STD_LOGIC;
           Bf : in STD_LOGIC;
           Cf : in STD_LOGIC;
           Sumf : out STD_LOGIC;
           Carryf : out STD_LOGIC);
end fulladder;

architecture Structural of fulladder is

signal Sum1:std_logic;
signal Carry1:std_logic;
signal Carry2:std_logic;

component halfadder is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Sum : out STD_LOGIC;
           Carry : out STD_LOGIC);
end component;

begin
halfadder1 : halfadder port map(A=>Af,B=>Bf,Sum=>Sum1,Carry=>Carry1);
halfadder2 : halfadder port map(A=>Sum1,B=>Cf,Sum=>Sumf,Carry=>Carry2);
Carryf<=Carry1 or Carry2;

end Structural;
```

Για να βρεί ο κώδικας το component halfadder πρέπει να κάνουμε include το αρχείο κώδικα της 1ης άσκησης σαν source code στο τωρινό μας project fulladder.

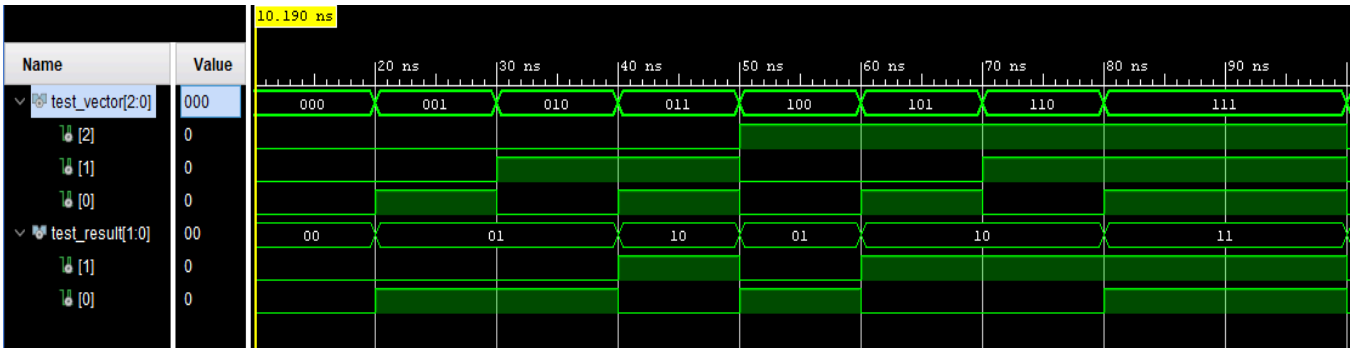


Το testbench αρχείο που χρησιμοποιήσαμε είναι το εξής:

```
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity testbench is

end testbench;

architecture arch_testbench of testbench is
    component fulladder
    port ( Af : in STD_LOGIC;
           Bf : in STD_LOGIC;
           Cf : in STD_LOGIC;
           Sumf : out STD_LOGIC;
           Carryf : out STD_LOGIC);
    end component;

    signal test_vector: std_logic_vector(2 downto 0);
    signal test_result : std_logic_vector(1 downto 0);

begin
uut: fulladder
port map(
    Af=>test_vector(0),
    Bf=>test_vector(1),
    Cf=>test_vector(2),
    Sumf=>test_result(0),
    Carryf=>test_result(1)
);
testing: process
begin
wait for 10ns;
test_vector<="000";
wait for 10ns;
test_vector<="001";
wait for 10ns;
test_vector<="010";
```

```
wait for 10ns;
test_vector<="011";
wait for 10ns;
test_vector<="100";
wait for 10ns;
test_vector<="101";
wait for 10ns;
test_vector<="110";
wait for 10ns;
test_vector<="111";
wait for 10ns;
end process;

end arch_testbench;
```

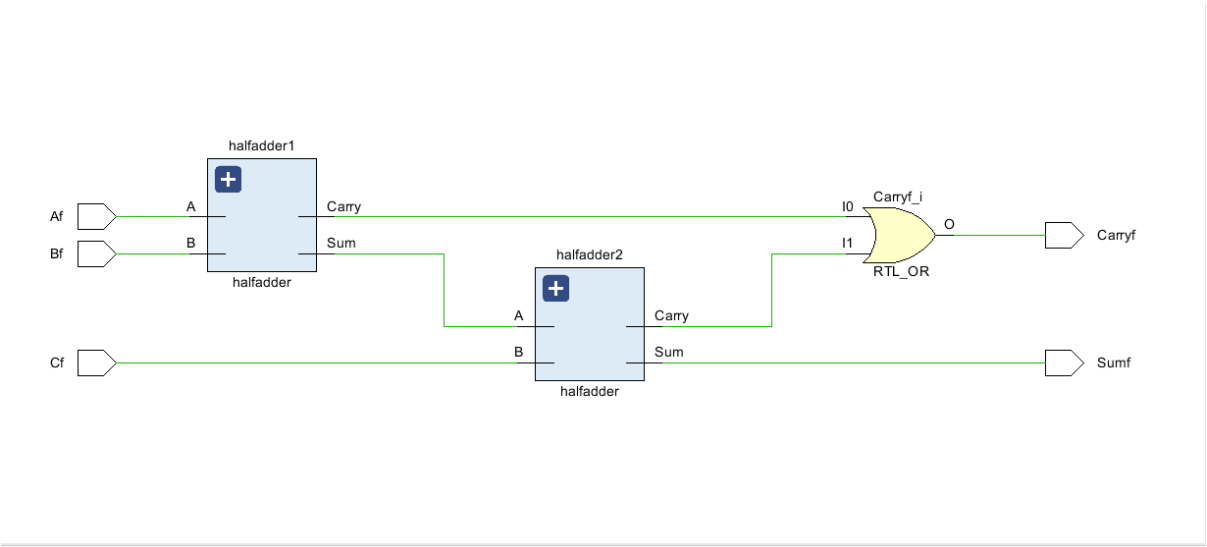
Το simulation που προκύπτει από το testbench επιβεβαιώνει την ορθή λειτουργία του fulladder.



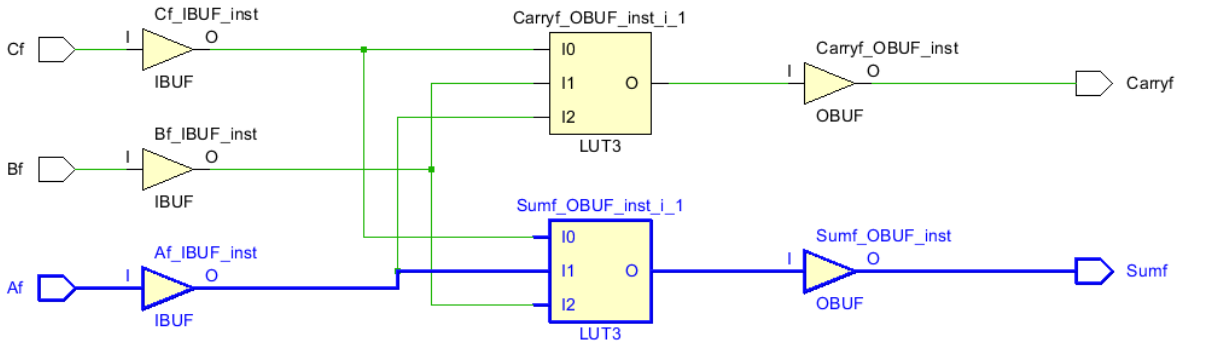
Truth table fulladder:

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

Παρουσίαση δομικού διαγράμματος (RTL schematic) της αρχιτεκτονικής του κυκλώματος:



Εύρεση κρίσιμου μονοπατιού (Critical Path) του κυκλώματος, καθώς και της χρονικής του καθυστέρησης:



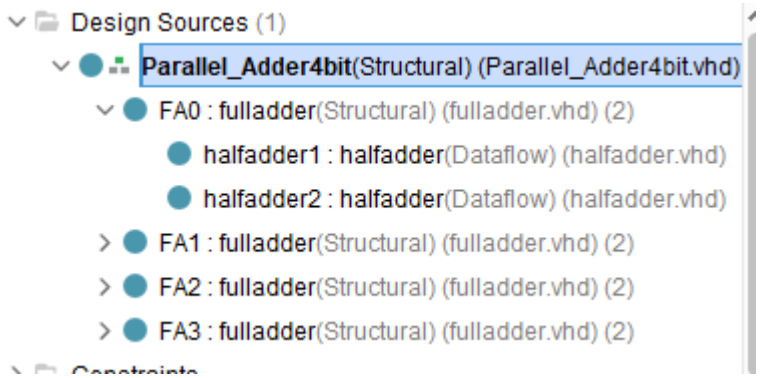
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	∞	3	4	2	Af	Sumf	5.377	3.778	1.599	∞	input port clock			0.000
Path 2	∞	3	4	2	Af	Carryf	5.351	3.752	1.599	∞	input port clock			0.000

Παρατηρούμε ότι το μονοπάτι με την μεγαλύτερη καθυστέρηση είναι το μονοπάτι του Af→Sumf, με συνολική καθυστέρηση **5.377 ns**.

3) Υλοποιήστε έναν Παράλληλο Αθροιστή των 4 bits (4-bit Parallel Adder - 4-bit PA) με περιγραφή δομής (Structural), βασιζόμενοι στην δομική μονάδα του Ερωτήματος 2.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Parallel_Adder4bit is
 Port ( A : in STD_LOGIC_VECTOR(3 downto 0);
        B : in STD_LOGIC_VECTOR(3 downto 0);
        Cin : in STD_LOGIC;
        Sum : out STD_LOGIC_VECTOR(3 downto 0);
        Cout : out STD_LOGIC);
end Parallel_Adder4bit;

architecture Structural of Parallel_Adder4bit is
component fulladder is
    Port (
     Af : in STD_LOGIC;
     Bf : in STD_LOGIC;
     Cf : in STD_LOGIC;
     Sumf : out STD_LOGIC;
     Carryf : out STD_LOGIC
   );
end component;

signal Carry : std_logic_vector(3 downto 0);

begin
  FA0 : fulladder port map(Af=>A(0), Bf=>B(0), Cf=>Cin, Sumf=>Sum(0), Carryf=>Carry(0));
  FA1 : fulladder port map(Af=>A(1), Bf=>B(1), Cf=>Carry(0), Sumf=>Sum(1),
Carryf=>Carry(1));
  FA2 : fulladder port map(Af=>A(2), Bf=>B(2), Cf=>Carry(1), Sumf=>Sum(2),
Carryf=>Carry(2));
  FA3 : fulladder port map(Af=>A(3), Bf=>B(3), Cf=>Carry(2), Sumf=>Sum(3), Carryf=>Cout);
end Structural;
```

Για να βρεί ο κώδικας τα component halfadder, fulladder πρέπει να κάνουμε include τα αρχεία κώδικα της 1ης, 2ης άσκησης σαν source code στο τωρινό μας project fulladder.

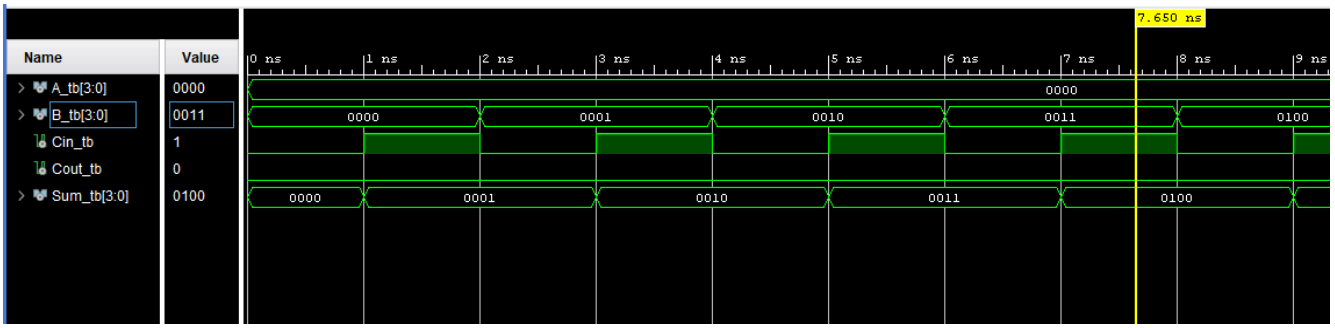


Το testbench αρχείο που χρησιμοποιήσαμε είναι το εξής:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;


entity Parallel_Adder4bit_tb is
--  Port ( );
end Parallel_Adder4bit_tb;

architecture testbench of Parallel_Adder4bit_tb is

    -- Component declaration for the DUT (Device Under Test)
    component Parallel_Adder4bit
        Port (
            A : in STD_LOGIC_VECTOR(3 downto 0);
            B : in STD_LOGIC_VECTOR(3 downto 0);
            Cin : in STD_LOGIC;
            Sum : out STD_LOGIC_VECTOR(3 downto 0);
            Cout : out STD_LOGIC
        );
    end component;

    -- Signals for stimulus generation and result capturing
    signal A_tb, B_tb: STD_LOGIC_VECTOR(3 downto 0);
    signal Cin_tb, Cout_tb: STD_LOGIC;
    signal Sum_tb: STD_LOGIC_VECTOR(3 downto 0);

begin
    -- Instantiate the DUT (Device Under Test)
    uut: Parallel_Adder4bit port map (
```

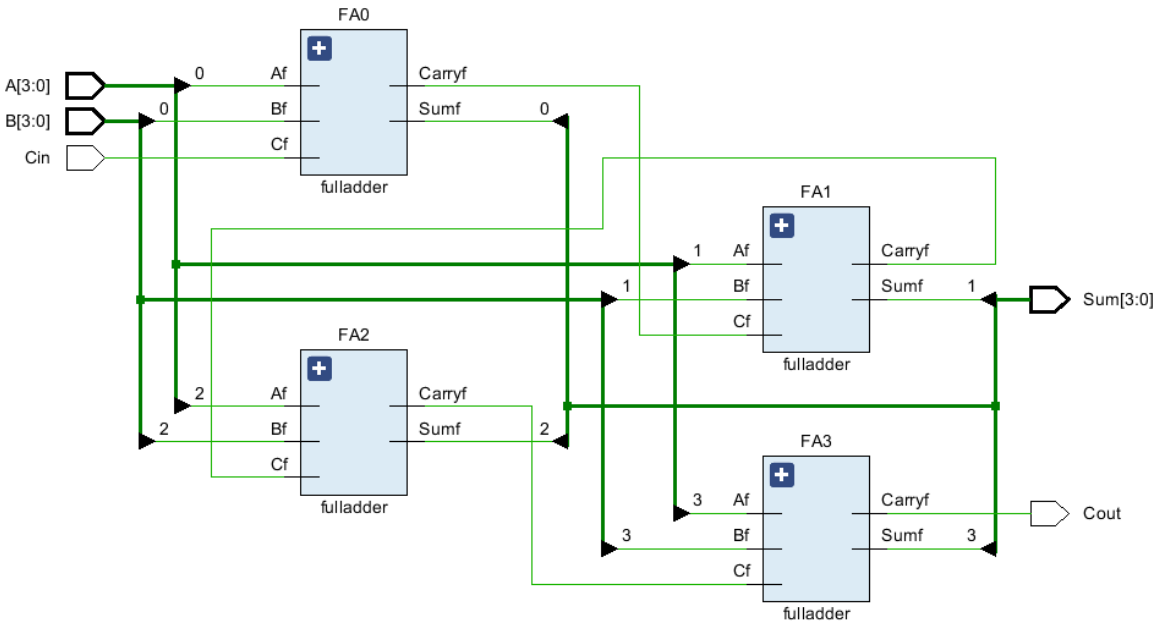
```
        A => A_tb,
        B => B_tb,
        Cin => Cin_tb,
        Sum => Sum_tb,
        Cout => Cout_tb
    );

   -- Stimulus process
    stimulus: process
   begin
        for i in 0 to 15 loop
            for j in 0 to 15 loop

                        A_tb <= std_logic_vector(to_unsigned(i, 4));
                        B_tb <= std_logic_vector(to_unsigned(j, 4));
                        Cin_tb <= '0';
                        wait for 1  ns;
                        Cin_tb <= '1';
                        wait for 1  ns;

            end loop;
        end loop;
        -- End of test
        wait;
end process stimulus;
end testbench;
```

Το simulation που προκύπτει από το testbench επιβεβαιώνει την ορθή λειτουργία του fulladder.

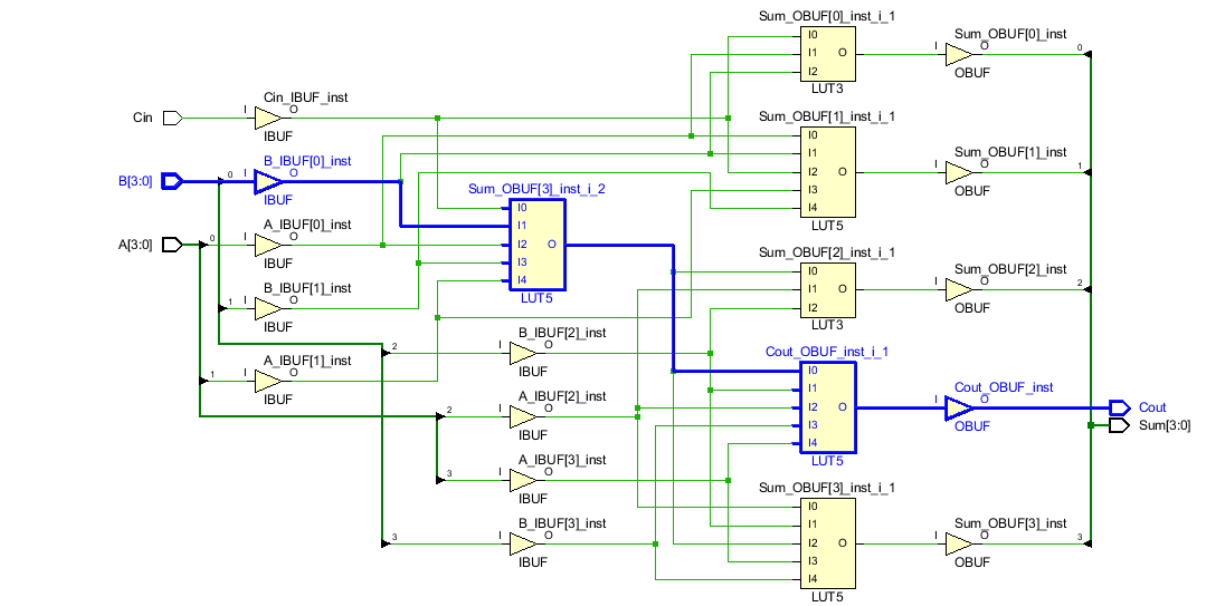


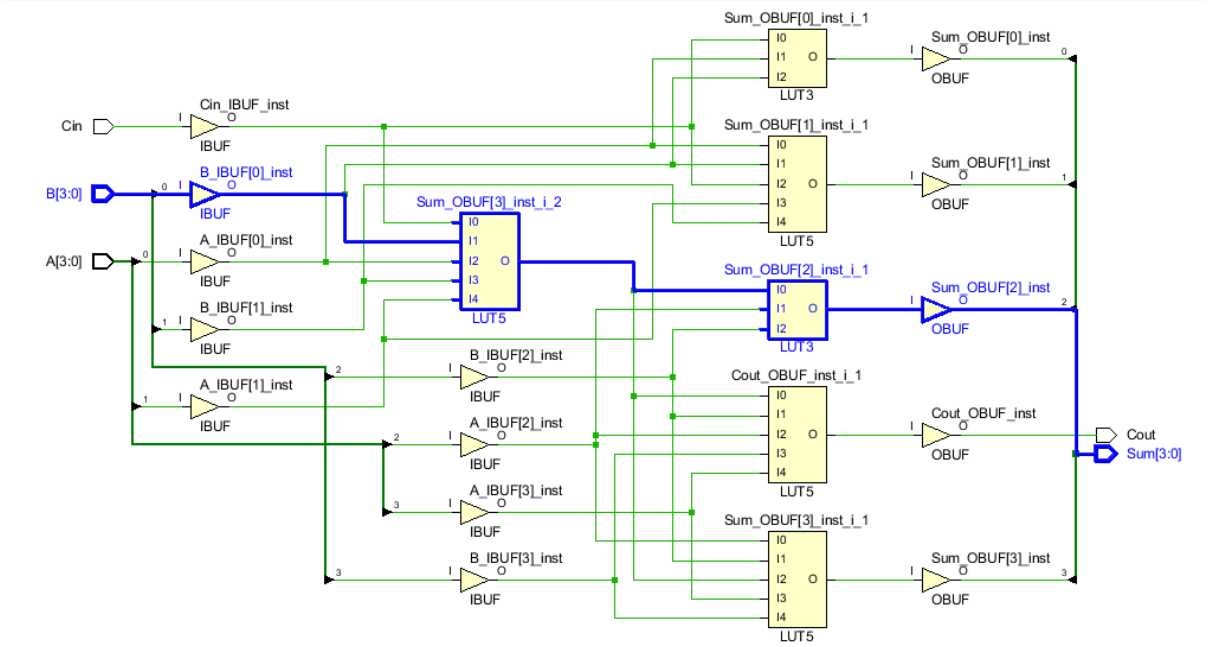
πχ
Για το συγκεκριμένο στιγμιότυπο έχουμε
A→ 0000
B→ 0011
Cin→1
Cout→0
Sum→0100

Παρουσίαση δομικού διαγράμματος (RTL schematic) της αρχιτεκτονικής του κυκλώματος:



Εύρεση κρίσιμου μονοπατιού (Critical Path) του κυκλώματος, καθώς και της χρονικής του καθυστέρησης:





Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	∞	4	5	3	B[0]	Cout	5.970	3.904	2.066	∞	input port clock			0.000
Path 2	∞	4	5	3	B[0]	Sum[2]	5.970	3.904	2.066	∞	input port clock			0.000
Path 3	∞	4	5	3	B[0]	Sum[3]	5.964	3.898	2.066	∞	input port clock			0.000
Path 4	∞	3	4	3	B[0]	Sum[0]	5.351	3.752	1.599	∞	input port clock			0.000
Path 5	∞	3	4	2	B[1]	Sum[1]	5.351	3.752	1.599	∞	input port clock			0.000

Παρατηρούμε ότι τα μονοπάτια με την μεγαλύτερη καθυστέρηση είναι τα μονοπάτια του B0→Cout και B0→Sum2 με συνολική καθυστέρηση **5.970 ns**.

4) Υλοποιήστε έναν BCD Πλήρη Αθροιστή (BCD Full Adder - BCD FA) με περιγραφή δομής (Structural). Να χρησιμοποιήσετε τη δομική μονάδα που υλοποιήθηκε στο Ερώτημα 3, οποιαδήποτε δομική μονάδα από τα προηγούμενα ερωτήματα, καθώς και επιπλέον λογική που θεωρείτε απαραίτητη.

Το Binary Coded Decimal (BCD) αναπαριστά μια μέθοδο αριθμητικής αναπαράστασης, όπου κάθε δεκαδικό ψηφίο αντιστοιχεί σε έναν συνδυασμό δυαδικών ψηφίων. Αυτή η προσέγγιση, με την οποία κάθε δεκαδικό ψηφίο αντιστοιχεί σε μια σταθερή αριθμητική τιμή, επιτρέπει την ακριβή αναπαράσταση των αριθμών χωρίς καμία απώλεια πληροφορίας.
Η συγκεκριμένη μέθοδος χρησιμοποιείται ευρέως σε εφαρμογές όπου η ακρίβεια και η ακρίβεια είναι ζωτικής σημασίας, όπως σε ψηφιακά ρολόγια, χρηματικές μονάδες και συστήματα ελέγχου. Η εξαιρετική ακρίβεια της αναπαράστασης BCD την καθιστά ιδανική για τέτοιες εφαρμογές, καθώς διατηρεί τη σημαντική ακρίβεια κατά τη μετατροπή μεταξύ δεκαδικής και δυαδικής αναπαράστασης.

Οι έγκυρες τιμές στο BCD αντιστοιχούν στις δεκαδικές αριθμητικές τιμές από 0 έως 9, και αναπαρίστανται με τους ακόλουθους συνδυασμούς δυαδικών ψηφίων:

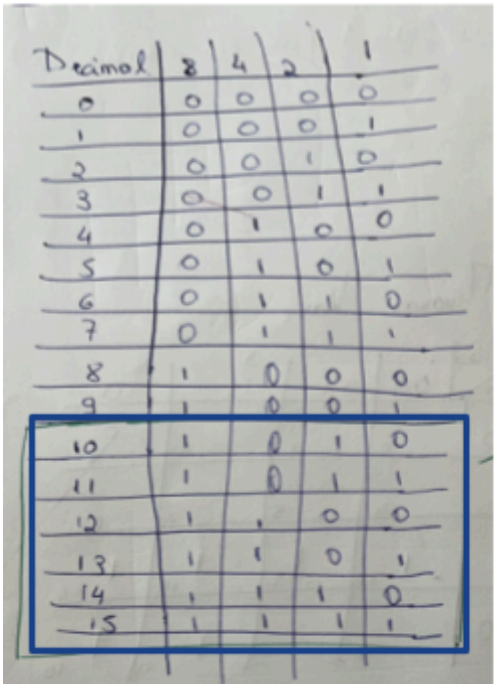
- 0: 0000
- 1: 0001
- 2: 0010
- 3: 0011
- 4: 0100
- 5: 0101
- 6: 0110
- 7: 0111
- 8: 1000
- 9: 1001

Από την άλλη πλευρά, οι μη έγκυρες τιμές στο BCD είναι αυτές που δεν αντιστοιχούν σε δεκαδικές αριθμητικές τιμές, όπως για παράδειγμα:

- 1010 (ή μεγαλύτερο από 9): Δεν έχει νόημα σε δεκαδικό BCD.
- Οποιαδήποτε συνδυασμός με περισσότερα από 4 ψηφία: Δεν μπορεί να αναπαρασταθεί από μια μονάδα BCD.

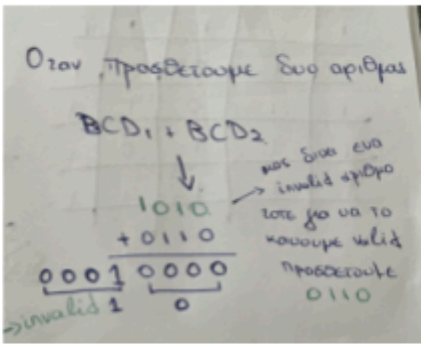
Αυτή η σταθερή σχέση μεταξύ δεκαδικών αριθμών και των αντίστοιχων δυαδικών συνδυασμών είναι ουσιώδης για τη σωστή λειτουργία του συστήματος BCD.

Για να υλοποιήσουμε το **BCD Πλήρη Αθροιστή (BCD Full Adder - BCD FA)** ακολουθήσαμε τα εξης βηματα **:**

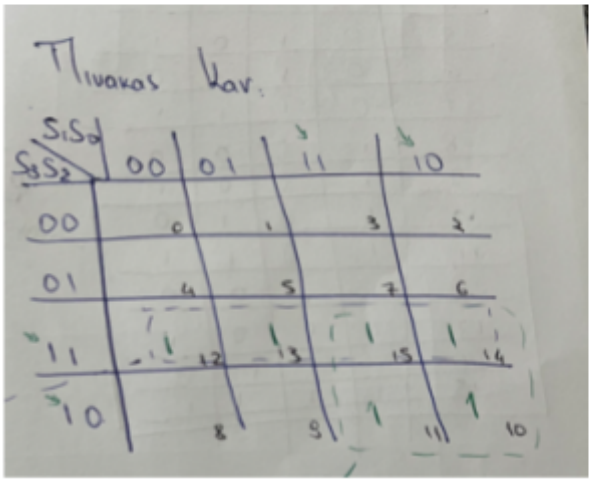


Invalid values

Για να κανουμε μια invalid τιμη να ειναι valid πρεπει να κανουμε την εξης προσθεση

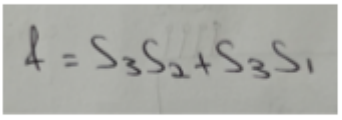


Δημιουργήσαμε μια συναρτηση με την βοηθεια του πινακα Karnaugh ετσι ωστε να γινεται ο ελεγχος valid ,invalid value



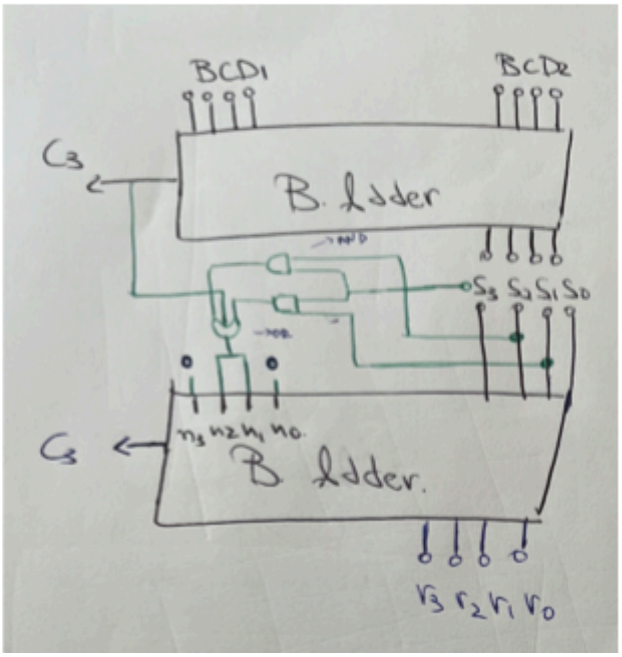
Στις Invalid στο πινακα βαλαμε 1

Συναρτηση :



Στη τελικη συναρτηση προσθετουμε και το C3(κρατουμενο απο το πρωτο αθροιστη)

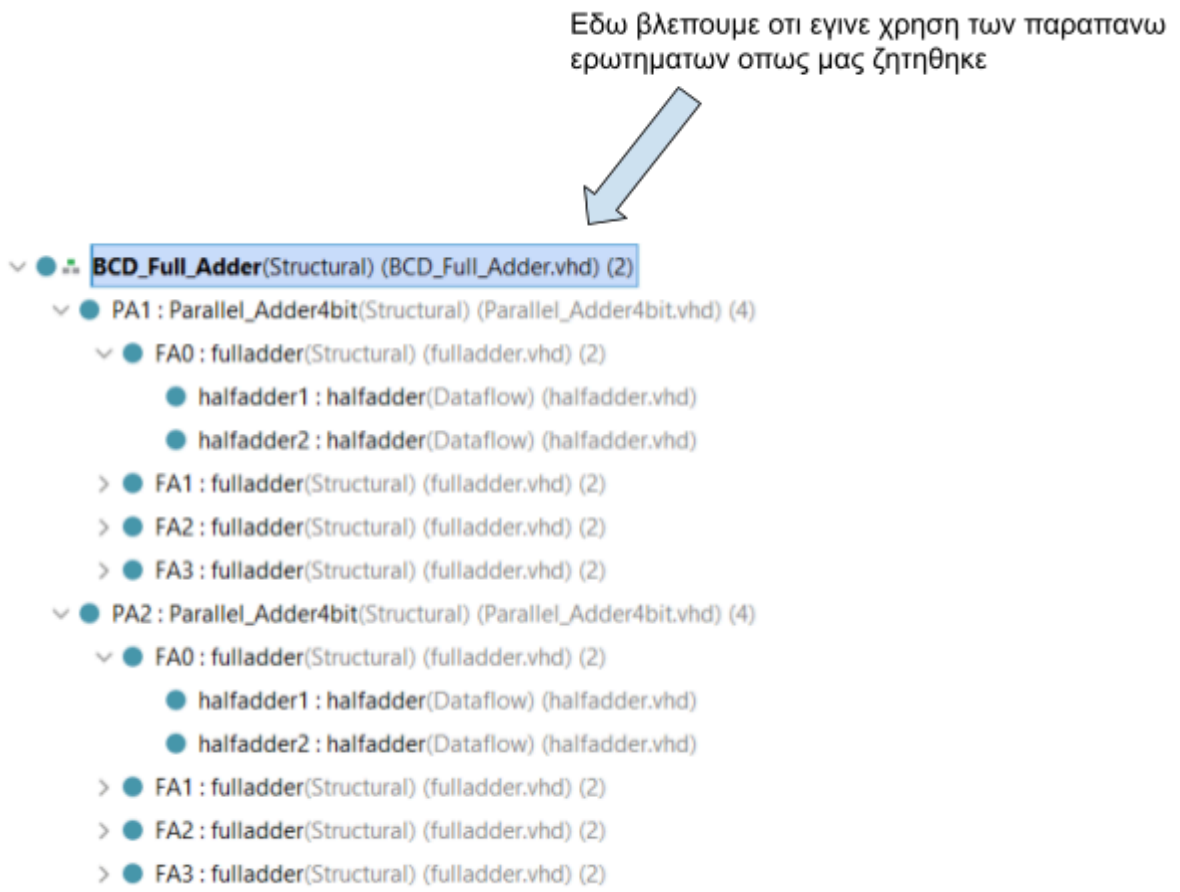
Για την υλοποιηση του χρησιμοποιησαμε δύο (4-bit Parallel Adder - 4-bit PA) από το ερωτημα 3 και ενδιαμεσα υλοποιήσαμε την συνάρτηση μας ,οπως φαινεται και στο παρακατω σχημα



Κωδικας

```
----------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity BCD_Full_Adder is
    Port (
        bcd1 : in STD_LOGIC_VECTOR(3 downto 0);
        bcd2 : in STD_LOGIC_VECTOR(3 downto 0);
        Cinbcd : in STD_LOGIC;

        Sumbcd  : out STD_LOGIC_VECTOR(3 downto 0);
        Coutbcd  : out STD_LOGIC
    );
end BCD_Full_Adder;

architecture Structural of BCD_Full_Adder is

    -- Component declaration for the Parallel_Adder4bit
    component Parallel_Adder4bit is
        Port (
            A : in STD_LOGIC_VECTOR(3 downto 0);
            B : in STD_LOGIC_VECTOR(3 downto 0);
```

```
            Cin : in STD_LOGIC;
            Sum: out STD_LOGIC_VECTOR(3 downto 0);
            Cout : out STD_LOGIC
        );
    end component;

    -- Signals for the first Parallel_Adder4bit
    signal sumFPD: std_logic_vector(3 downto 0); --Sum of first
    signal CoutFPD : std_logic; --carry of first

    -- Signals for the second Parallel_Adder4bit
    signal F : std_logic;
    signal second_value : std_logic_vector(3 downto 0);
    signal Cout_second : std_logic;

begin

    -- Instantiate the first Parallel_Adder4bit
    PA1 : Parallel_Adder4bit port map (
        A => bcd1,
        B => bcd2,
        Cin => Cinbcd,
        Sum => sumFPD,
        Cout => CoutFPD
    );


    -- Calculate F based on your requirement (F = S3*S2 + S3*S1 + C3)
    F <= (sumFPD(3) and sumFPD(2)) or (sumFPD(3) and sumFPD(1)) or CoutFPD;

    -- Connect S3S2S1S0 to the output S
    second_value(0) <= '0';
    second_value(1) <= F;
    second_value(2) <= F;
    second_value(3) <= '0';


    -- Instantiate the second Parallel_Adder4bit
    PA2 : Parallel_Adder4bit port map (
        A => sumFPD,
        B => second_value,
        Cin => '0',
        Sum => Sumbcd,
        Cout => Cout_second
    );
 Coutbcd <= F;
end Structural;
-------------------------------------------------------------------------------------------------------------------
```

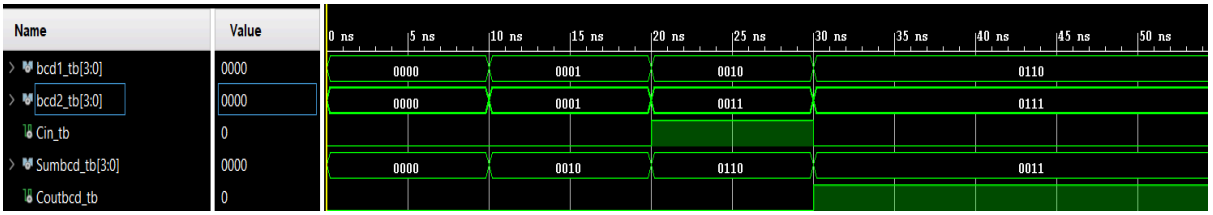


Το testbench αρχείο που χρησιμοποιήσαμε είναι το εξής:

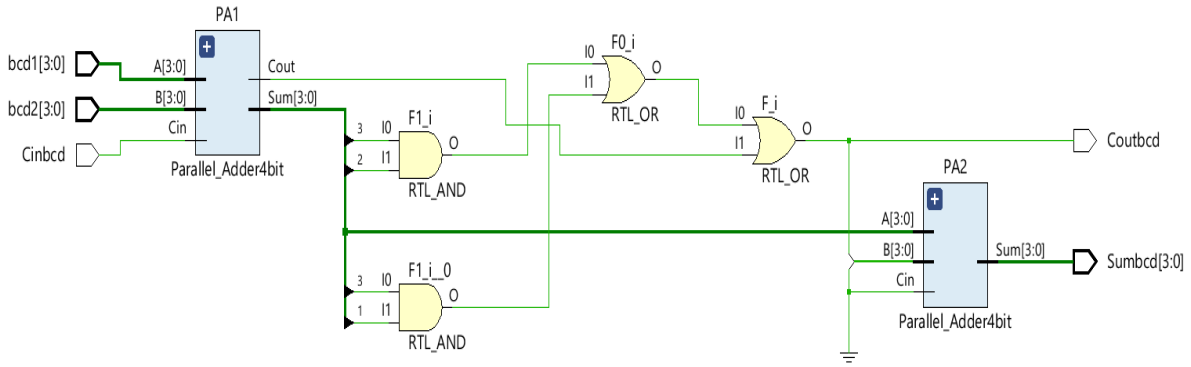
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity BCD_Full_Adder_tb is
--  Port ( );
end BCD_Full_Adder_tb;

architecture testbench of BCD_Full_Adder_tb is

    -- Signals for inputs
    signal bcd1_tb : STD_LOGIC_VECTOR(3 downto 0) := (others => '0');
    signal bcd2_tb : STD_LOGIC_VECTOR(3 downto 0) := (others => '0');
    signal Cin_tb  : STD_LOGIC := '0';
    -- Signals for outputs
    signal Sumbcd_tb  : STD_LOGIC_VECTOR(3 downto 0);
```

```
    signal Coutbcd_tb : STD_LOGIC;
begin


    -- Instantiate the unit under test (UUT)
    UUT: entity work.BCD_Full_Adder
        port map (
            bcd1 => bcd1_tb,
            bcd2 => bcd2_tb,
            Cinbcd => Cin_tb,
            Sumbcd => Sumbcd_tb,
            Coutbcd => Coutbcd_tb
        );

    -- Stimulus process
    stim_proc: process
    begin
        -- Apply inputs and observe outputs
        bcd1_tb <= "0000";  -- Input 0
        bcd2_tb <= "0000";  -- Input 0
        Cin_tb <= '0';      -- Carry-in 0
        wait for 10 ns;

        bcd1_tb <= "0001";  -- Input 1
        bcd2_tb <= "0001";  -- Input 1
        Cin_tb <= '0';      -- Carry-in 0
        wait for 10 ns;

        bcd1_tb <= "0010";  -- Input 2
        bcd2_tb <= "0011";  -- Input 3
        Cin_tb <= '1';      -- Carry-in 1
        wait for 10 ns;


        bcd1_tb <= "0110";
        bcd2_tb <= "0111";
        Cin_tb <= '0';
        wait for 10 ns;



        wait;
    end process stim_proc;

end testbench;
```

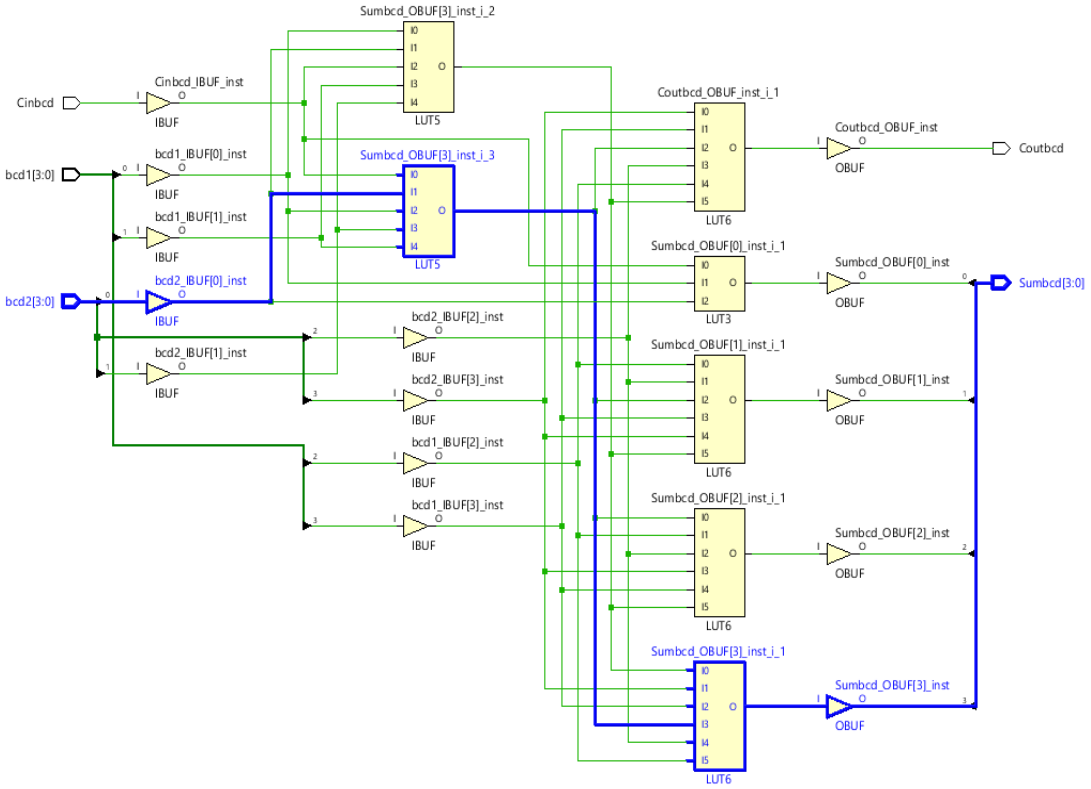
Το simulation που προκύπτει από το testbench επιβεβαιώνει την ορθή λειτουργία του **BCD Full Adder - BCD FA**.

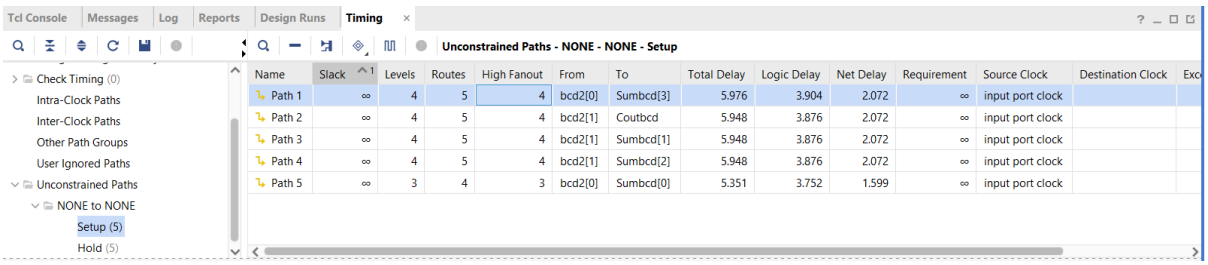


Παρουσίαση δομικού διαγράμματος (RTL schematic) της αρχιτεκτονικής του κυκλώματος:



Εύρεση κρίσιμου μονοπατιού (Critical Path) του κυκλώματος, καθώς και της χρονικής του καθυστέρησης:





Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 1	∞	4	5	4	bcd2[0]	Sumbcd[3]	5.976	3.904	2.072	∞	input port clock	
Path 2	∞	4	5	4	bcd2[1]	Coutbcd	5.948	3.876	2.072	∞	input port clock	
Path 3	∞	4	5	4	bcd2[1]	Sumbcd[1]	5.948	3.876	2.072	∞	input port clock	
Path 4	∞	4	5	4	bcd2[1]	Sumbcd[2]	5.948	3.876	2.072	∞	input port clock	
Path 5	∞	3	4	3	bcd2[0]	Sumbcd[0]	5.351	3.752	1.599	∞	input port clock	

Παρατηρούμε ότι το μονοπάτι με την μεγαλύτερη καθυστέρηση είναι το μονοπάτι , με συνολική καθυστέρηση **5.976 ns**.

5) Υλοποιήστε έναν Παράλληλο BCD Αθροιστή των 4 ψηφίων (4-BCD Parallel Adder - 4-BCD PA) με περιγραφή δομής (Structural), βασιζόμενοι στην δομική μονάδα του Ερωτήματος 4.

Ένας παράλληλος BCD αθροιστής των 4 ψηφίων είναι ένας ψηφιακός κυκλώνας που πραγματοποιεί την πρόσθεση δύο δεκαδικών αριθμών σε δυαδική κωδικοποίηση BCD. Οι BCD αθροιστές είναι βασικά σχήματα στην ψηφιακή λογική και χρησιμοποιούνται ευρέως σε εφαρμογές που απαιτούν ακρίβεια και αξιοπιστία στις αριθμητικές πράξεις.
Για να υλοποιήσουμε έναν BCD αθροιστή των 4 ψηφίων, κάθε ψηφίο BCD πρέπει να προστεθεί με το αντίστοιχο ψηφίο BCD του δεύτερου αριθμού, λαμβάνοντας υπόψη τυχόν προσπελάσεις (carry) από τα προηγούμενα ψηφία. Η δομή του BCD αθροιστή θα περιλαμβάνει συνδυαστική λογική για την πρόσθεση κάθε ψηφίου, καθώς και λογική για τον έλεγχο των προσπελάσεων.Συνοπτικά, η δομή του BCD αθροιστή θα αποτελείται από διάφορες υπομονάδες, κάθε μια αναλαμβάνει μια συγκεκριμένη λειτουργία, όπως πρόσθεση ψηφίων και ελέγχου προσπελάσεων. Η συνδυαστική λογική συνδέει αυτές τις υπομονάδες μεταξύ τους, δημιουργώντας έναν πλήρη BCD αθροιστή.

Κωδικας

```
----------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity BCD_4digit_FA is
    Port ( A : in STD_LOGIC_VECTOR (15 downto 0);
           B : in STD_LOGIC_VECTOR (15 downto 0);
           FinalCin : in STD_LOGIC;
           FinalSum : out STD_LOGIC_VECTOR (15 downto 0);
           FinalCout : out STD_LOGIC);
end BCD_4digit_FA;

architecture Structural of BCD_4digit_FA is
```

```
component BCD_Full_adder is
Port (
        bcd1 : in STD_LOGIC_VECTOR(3 downto 0);
        bcd2 : in STD_LOGIC_VECTOR(3 downto 0);
        Cin : in STD_LOGIC;

        Sum  : out STD_LOGIC_VECTOR(3 downto 0);
        Cout  : out STD_LOGIC
    );
end component;
signal Carry : std_logic_vector(2 downto 0);

begin
  BCD0 :  BCD_Full_adder port map(bcd1=>A(3 downto 0), bcd2=>B(3 downto 0),
Cin=>FinalCin, Sum=>FinalSum(3 downto 0), Cout=>Carry(0));
  BCD1 :  BCD_Full_adder port map(bcd1=>A(7 downto 4), bcd2=>B(7 downto 4),
Cin=>Carry(0), Sum=>FinalSum(7 downto 4), Cout=>Carry(1));
  BCD2 :  BCD_Full_adder port map(bcd1=>A(11 downto 8), bcd2=>B(11 downto 8),
Cin=>Carry(1), Sum=>FinalSum(11 downto 8), Cout=>Carry(2));
  BCD3 :  BCD_Full_adder port map(bcd1=>A(15 downto 12), bcd2=>B(15 downto 12),
Cin=>Carry(2), Sum=>FinalSum(15 downto 12), Cout=>FinalCout);

end Structural;
```

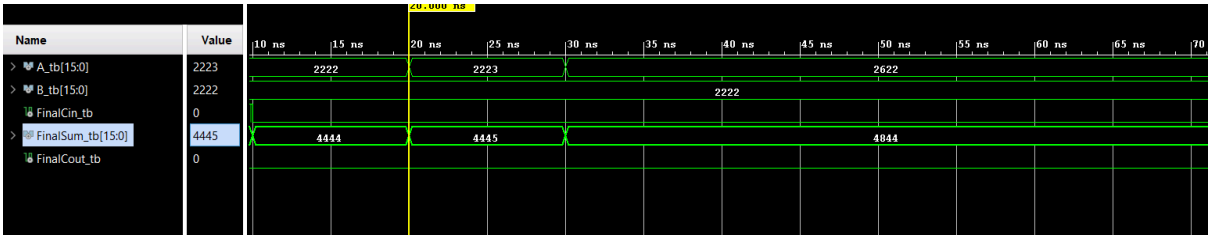
Εδω βλεπουμε οτι εγινε χρηση των παραπανω ερωτηματων οπως μας ζητηθηκε

Το testbench αρχείο που χρησιμοποιήσαμε είναι το εξής:

```
 library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity BCD_4digit_FA_tb is
end BCD_4digit_FA_tb;

architecture testbench of BCD_4digit_FA_tb is

    -- Component declaration for the BCD_4digit_FA module
    component BCD_4digit_FA is
        Port (
            A : in STD_LOGIC_VECTOR (15 downto 0);
            B : in STD_LOGIC_VECTOR (15 downto 0);
            FinalCin : in STD_LOGIC;
            FinalSum : out STD_LOGIC_VECTOR (15 downto 0);
            FinalCout : out STD_LOGIC
        );
    end component;

    -- Inputs
    signal A_tb, B_tb : STD_LOGIC_VECTOR(15 downto 0);
    signal FinalCin_tb : STD_LOGIC;

    -- Outputs
    signal FinalSum_tb : STD_LOGIC_VECTOR(15 downto 0);
    signal FinalCout_tb : STD_LOGIC;

begin

    -- Instantiate the BCD_4digit_FA module
    UUT: BCD_4digit_FA port map (
        A => A_tb,
        B => B_tb,
        FinalCin => FinalCin_tb,
        FinalSum => FinalSum_tb,
        FinalCout => FinalCout_tb
    );

    -- Stimulus process
    stimulus: process
    begin



       A_tb <= "0010001000100010";
```

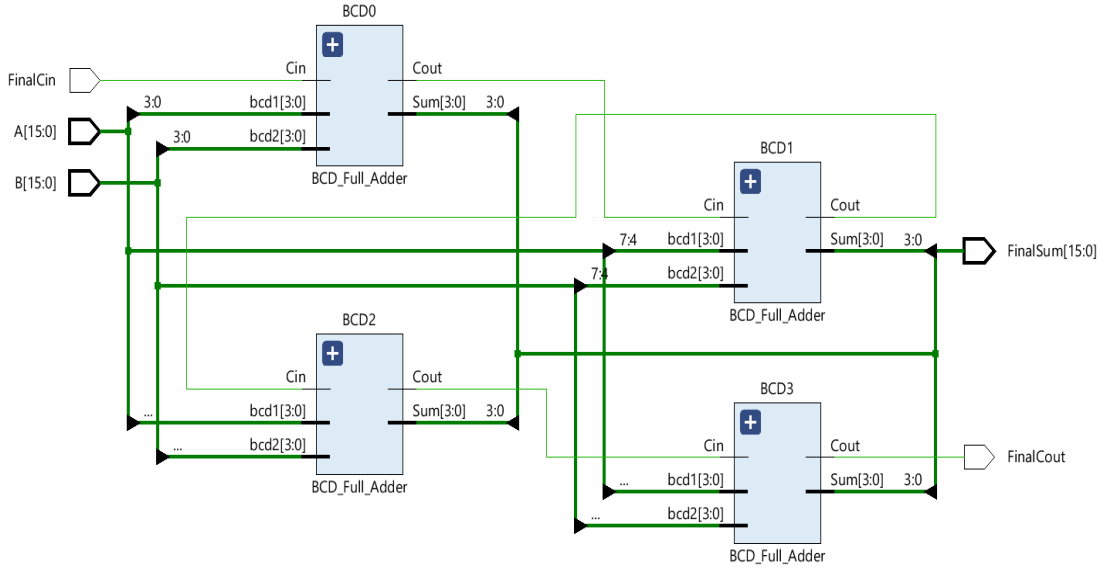
```
        B_tb <= "0010001000100010";
        FinalCin_tb <= '1';
         -- Wait for some time to observe outputs
         wait for 10 ns;


         A_tb <= "0010001000100010";
        B_tb <= "0010001000100010";
        FinalCin_tb <= '0';


         -- Wait for some time to observe outputs
         wait for 10 ns;



         A_tb <= "0010001000100011";
        B_tb <= "0010001000100010";
        FinalCin_tb <= '0';


         -- Wait for some time to observe outputs
         wait for 10 ns;

         A_tb <= "0010011000100010";
        B_tb <= "0010001000100010";
        FinalCin_tb <= '0';


         -- Wait for some time to observe outputs
         wait for 10 ns;



         -- End the simulation
         wait;
    end process stimulus;

end testbench;
--------------------------------------------------------------------------------------------------------------
```

Το simulation που προκύπτει από το testbench επιβεβαιώνει την ορθή λειτουργία του **4-BCD Parallel Adder - 4-BCD PA.**

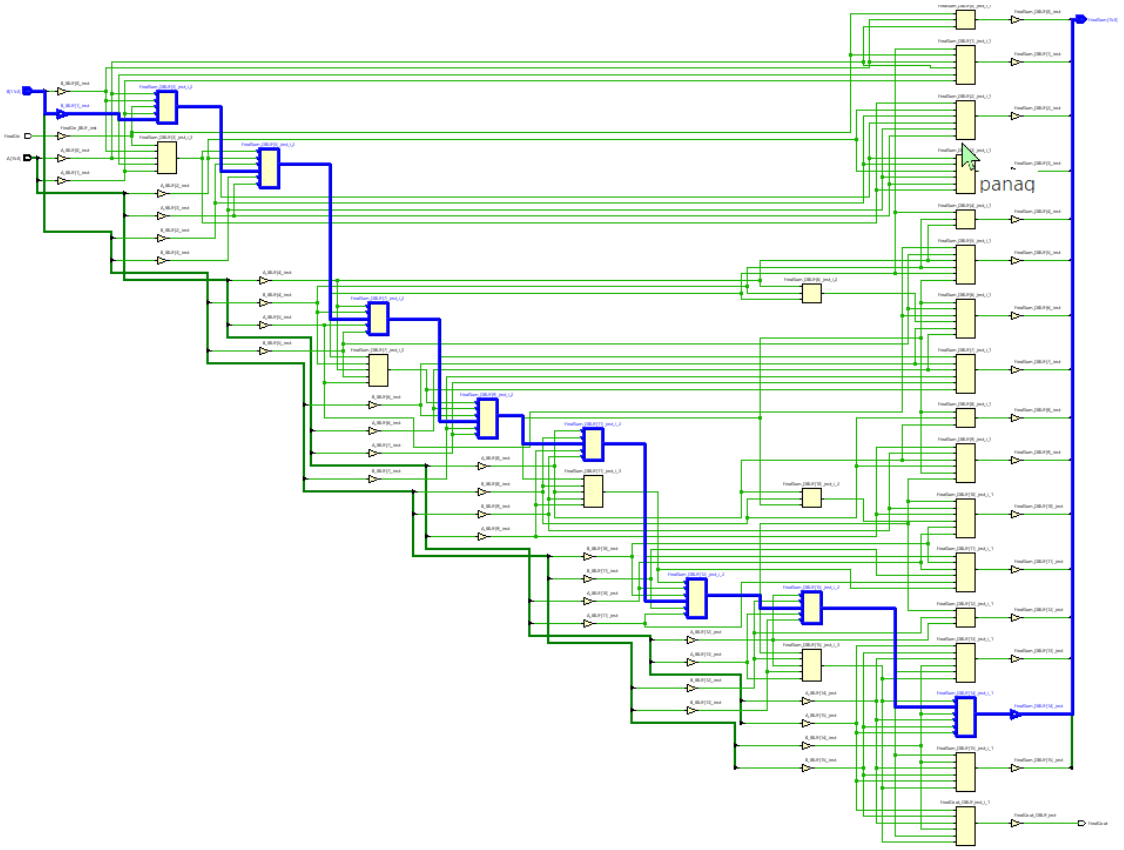


Παρατηρουμε οτι εχουμε το αθροισμα των BCD αριθμων με valid τιμη

Παρουσίαση δομικού διαγράμματος (RTL schematic) της αρχιτεκτονικής του κυκλώματος:



Εύρεση κρίσιμου μονοπατιού (Critical Path) του κυκλώματος, καθώς και της χρονικής του καθυστέρησης:



General Information
Timer Settings
Design Timing Summary
Check Timing (0)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths
NONE to NONE
Setup (10)
Hold (10)

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exc
Path 1	∞	10	11	7	B[1]	FinalSum[14]	9.521	4.620	4.901	∞	input port clock		
Path 2	∞	10	11	7	B[1]	FinalCout	9.515	4.614	4.901	∞	input port clock		
Path 3	∞	10	11	7	B[1]	FinalSum[13]	9.515	4.614	4.901	∞	input port clock		
Path 4	∞	10	11	7	B[1]	FinalSum[15]	9.515	4.614	4.901	∞	input port clock		
Path 5	∞	9	10	7	B[1]	FinalSum[10]	8.924	4.496	4.428	∞	input port clock		
Path 6	∞	9	10	7	B[1]	FinalSum[12]	8.924	4.496	4.428	∞	input port clock		
Path 7	∞	9	10	7	B[1]	FinalSum[9]	8.924	4.496	4.428	∞	input port clock		
Path 8	∞	8	9	7	B[1]	FinalSum[11]	8.317	4.366	3.951	∞	input port clock		
Path 9	∞	7	8	7	B[1]	FinalSum[5]	7.739	4.248	3.491	∞	input port clock		
Path 10	∞	7	8	7	B[1]	FinalSum[6]	7.739	4.248	3.491	∞	input port clock		

Timing Summary - timing_1

Παρατηρούμε ότι το μονοπάτι με την μεγαλύτερη καθυστέρηση είναι το μονοπάτι , με συνολική καθυστέρηση **9.521 ns**.