

2η Εργαστηριακή Άσκηση

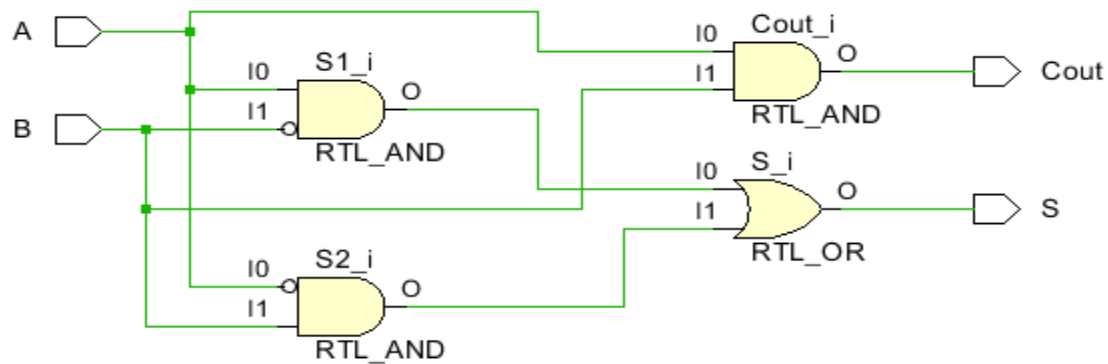
VLSI

Παντελαΐος Δημήτριος Α.Μ.: 03118049

Μοίρας Αλέξανδρος Α.Μ.: 03118081

Ζήτημα 1:

Το RTL schematic του ημιαθροιστή που υλοποιήσαμε:



Πρόκειται για ένα απλό συνδυαστικό κύκλωμα με μία πύλη AND μεταξύ των δύο bit εισόδου ώστε να παράγεται κρατούμενο αν και τα δύο bits εισόδου είναι 1 και το λογικό άθροισμα των γινομένων $A'B$ και AB' με δύο πύλες AND και μία OR καθώς άθροισμα 1 προκύπτει αν ένα από τα δύο bits της εισόδου είναι 1. Το κύκλωμα δημιουργήθηκε από το Vivado βασισμένο στη dataflow περιγραφή μας.

Ο VHDL κώδικας για την dataflow περιγραφή του half adder:

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

entity half_adder_dataflow is
    Port ( A : in STD_LOGIC;
          B : in STD_LOGIC;
          S : out STD_LOGIC;
          Cout : out STD_LOGIC);
end half_adder_dataflow;

architecture Dataflow of half_adder_dataflow is

    SIGNAL S1: STD_LOGIC;
    SIGNAL S2: STD_LOGIC;

begin

    S1 <= A AND (NOT B);
    S2 <= (NOT A) AND B;
    S <= S1 OR S2;

    Cout <= A AND B;

end Dataflow;
```

Ο VHDL κώδικας για το testbench της dataflow περιγραφής του half adder:

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

entity half_adder_dataflow_testbench is
end half_adder_dataflow_testbench;

architecture Behavioral of half_adder_dataflow_testbench is

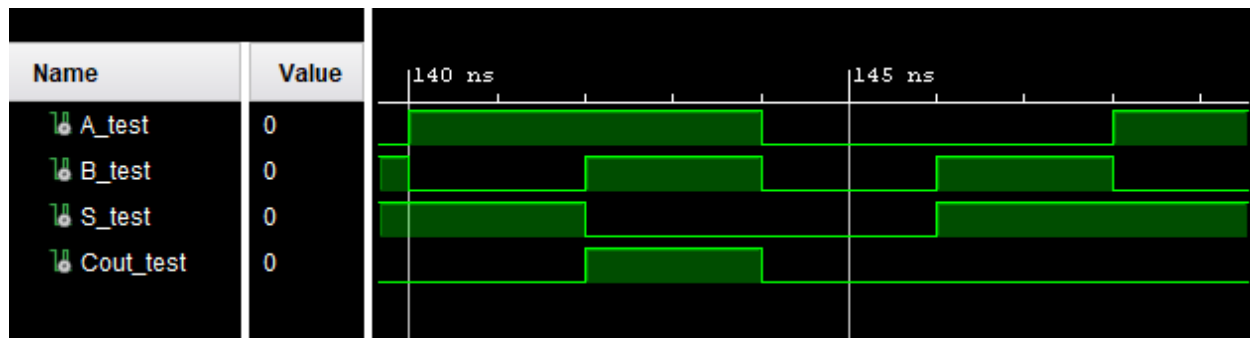
    signal A_test, B_test :std_logic;
    signal S_test, Cout_test: std_logic;

    component half_adder_dataflow is
        port(
            A, B: in std_logic;
            S, Cout: out std_logic
        );
    end component;

begin
    uut: half_adder_dataflow
        port map(
            A=>A_test,
            B=>B_test,
            S=>S_test,
            Cout=>Cout_test
        );

    stimulus: process begin
        A_test<='0';
        B_test<='0';
        wait for 2ns;
        A_test<='0';
        B_test<='1';
        wait for 2ns;
        A_test<='1';
        B_test<='0';
        wait for 2ns;
        A_test<='1';
        B_test<='1';
        wait for 2ns;
    end process;
end architecture;
```

Η προσομοίωση που επιβεβαιώνει την ορθή λειτουργία του κυκλώματός μας όπως περιεγράφη και παραπάνω:

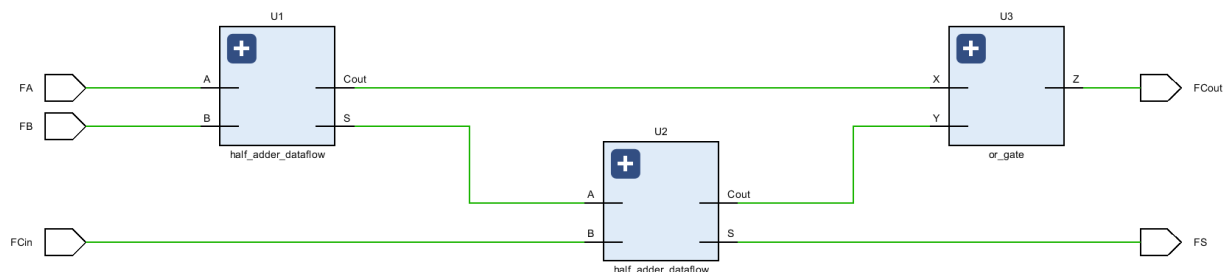


Το κρίσιμο μονοπάτι του κυκλώματος είναι αυτό από το B στο S που είναι λογικό καθώς για να παραχθεί το άθροισμα S από τις εισόδους μεσολαβούν δύο επίπεδα πυλών, ενώ στο κρατούμενο 1. Η χρονική του καθυστέρηση είναι 5.377ns.

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	3	4	2	B	S	5.377	3.778	1.599
Path 2	∞	3	4	2	B	Cout	5.351	3.752	1.599

Ζήτημα 2:

Τώρα εκτός από τα FA, FB bits εισόδου έχουμε και κρατούμενο εισόδου. Το RTL Schematic του Full Adder που υλοποιήσαμε:



Εδώ εμείς ορίζουμε τη δομή του κυκλώματος χρησιμοποιώντας structural περιγραφή. Χρησιμοποιούμε δύο ημιαθροιστές όπου στον πρώτο αθροίζουμε τα δύο bits εισόδου A και B, ενώ στον δεύτερο αθροίζουμε το αποτέλεσμα της πρώτης άθροισης με το κρατούμενο εισόδου, ώστε να προκύψει το τελικό άθροισμα εξόδου, ενώ κρατούμενο στην έξοδο έχουμε αν οποιοσδήποτε από τους 2 ημιαθροιστές δώσει κρατούμενο (μόνο ένας από τους δύο μπορεί να δίνει) για αυτό και διασυνδέουμε τις εξόδους κρατουμένου τους με μια πύλη OR (την οποία υλοποιήσαμε με dataflow περιγραφή ώστε να τη χρησιμοποιήσουμε στη structural) η έξοδος της οποίας είναι το κρατούμενο του Full Adder.

Ο VHDL κώδικας για την structural περιγραφή του full adder:

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

entity full_adder_structural is
    Port ( FA, FB, FCin : in STD_LOGIC;
          FS, FCout : out STD_LOGIC
          );
end full_adder_structural;

architecture Structural of full_adder_structural is
    component half_adder_dataflow is
        Port ( A, B : in STD_LOGIC;
              S, Cout : out STD_LOGIC
              );
    end component;

    component or_gate is
        Port ( X, Y : in STD_LOGIC;
              Z : out STD_LOGIC
              );
    end component;

    signal S0, C0, C1 : STD_LOGIC;
begin

    U1: half_adder_dataflow PORT MAP (A=>FA, B=>FB, S=>S0, Cout=>C0);
    U2: half_adder_dataflow PORT MAP (A=>S0, B=>FCin, S=>FS, Cout=>C1);
    U3: or_gate PORT MAP (X=>C0, Y=>C1, Z=>FCout);

end Structural;
```

Ο VHDL κώδικας για την dataflow περιγραφή της πύλης OR:

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

entity or_gate is
    Port ( X : in STD_LOGIC;
          Y : in STD_LOGIC;
          Z : out STD_LOGIC);
end or_gate;

architecture Dataflow of or_gate is

begin
    Z <= X OR Y;

end Dataflow;
```

Ο VHDL κώδικας για το testbench της structural περιγραφής του full adder:

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

entity full_adder_structural_testbench is
end full_adder_structural_testbench;

architecture test_full_adder of full_adder_structural_testbench is

    signal FA_test, FB_test, FCin_test :std_logic;
    signal FS_test, Fcout_test: std_logic;

    component full_adder_structural is
        port(
            FA, FB, FCin: in std_logic;
            FS, Fcout: out std_logic
        );
    end component;

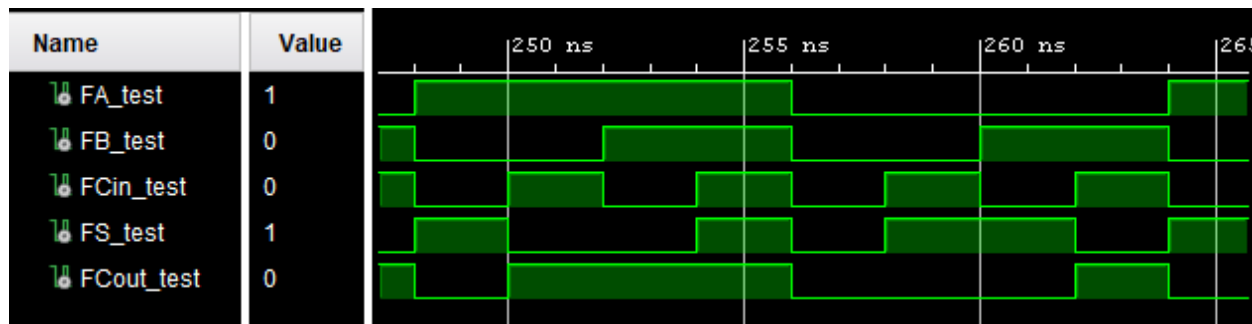
begin
    uut: full_adder_structural
        port map(
            FA=>FA_test,
            FB=>FB_test,
            FCin => FCin_test,
            FS=>FS_test,
            Fcout=>Fcout_test
        );

    stimulus: process begin
        FA_test<='0';
        FB_test<='0';
        FCin_test<='0';
        wait for 2ns;
        FA_test<='0';
        FB_test<='0';
        FCin_test<='1';
        wait for 2ns;
        FA_test<='0';
        FB_test<='1';
        FCin_test<='0';
        wait for 2ns;
        FA_test<='0';
        FB_test<='1';
        FCin_test<='1';
        wait for 2ns;
        FA_test<='1';
        FB_test<='0';
        FCin_test<='0';
        wait for 2ns;
        FA_test<='1';
        FB_test<='0';
        FCin_test<='1';
        wait for 2ns;
        FA_test<='1';
        FB_test<='1';
        FCin_test<='0';
        wait for 2ns;
        FA_test<='1';
        FB_test<='1';
        FCin_test<='1';
        wait for 2ns;

    end process;

end architecture;
```

Η προσομοίωση που επιβεβαιώνει την ορθή λειτουργία του κυκλώματός μας για τις 8 δυνατές διαφορετικές εισόδους:

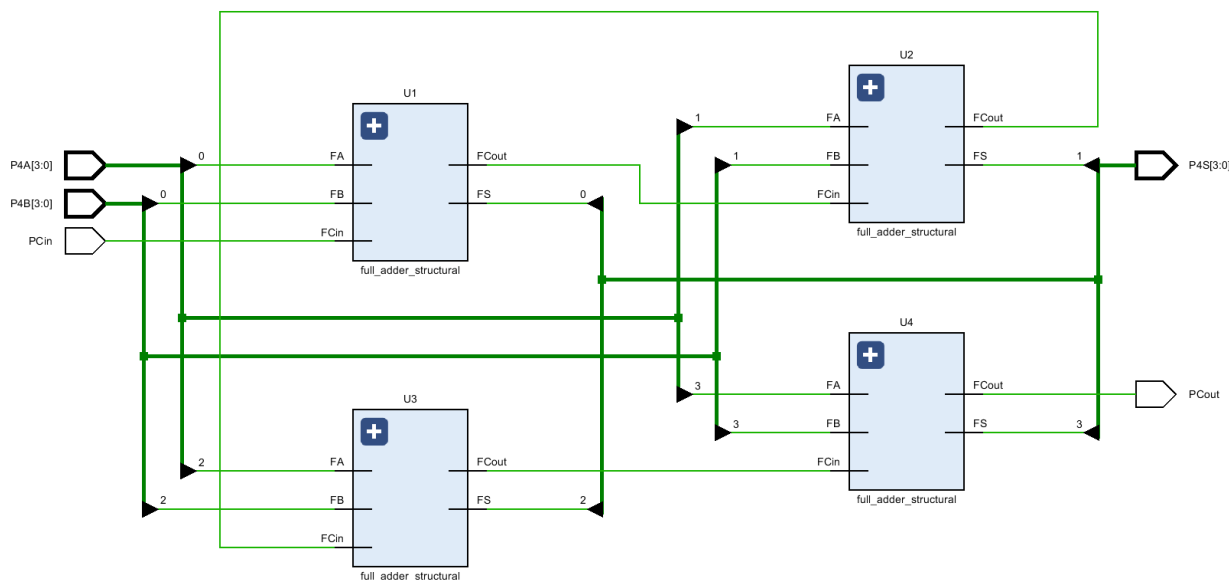


Το κρίσιμο μονοπάτι του κυκλώματος είναι από το FA στο FS καθώς χρειάζεται να υπολογιστούν δύο αθροίσματα από τους ημιαθροιστές που όπως εξηγήθηκε παραπάνω είναι χρονοβόρες διαδικασίες ενώ η πύλη OR δε φαίνεται να αυξάνει τόσο την καθυστέρηση του μονοπατιού FA-FCout. Η συνολική καθυστέρηση του πιο κρίσιμου μονοπατιού είναι 5.377ns.

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	3	4	2	FA	FS	5.377	3.778	1.599
Path 2	∞	3	4	2	FA	FCout	5.351	3.752	1.599

Ζήτημα 3:

Για τον παράλληλο αθροιστή των 4 bits θα διασυνδέσουμε 4 πλήρεις αθροιστές έναν για κάθε bit, με τον πρώτο (LSB) να δέχεται από την είσοδο το κρατούμενο εισόδου του, τον 2ο από να το δέχεται από το κρατούμενο εξόδου της 1ης βαθμίδας κλπ. Το RTL Schematic στο οποίο φαίνεται η παραπάνω ιδέα είναι το εξής:



Ο VHDL κώδικας για την structural περιγραφή του 4 bit parallel adder:

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

entity four_bit_parallel_adder_structural is
    Port ( P4A : in STD_LOGIC_VECTOR (3 downto 0);
          P4B : in STD_LOGIC_VECTOR (3 downto 0);
          PCin : in STD_LOGIC;
          P4S : out STD_LOGIC_VECTOR (3 downto 0);
          PCout : out STD_LOGIC);
end four_bit_parallel_adder_structural;

architecture Structural of four_bit_parallel_adder_structural is

    component full_adder_structural is
        Port ( FA, FB, FCin : in STD_LOGIC;
              FS, FCout : out STD_LOGIC
              );
    end component;

    signal C1, C2, C3 : STD_LOGIC;

begin

    U1: full_adder_structural PORT MAP (FA=>P4A(0), FB=>P4B(0), FCin=>PCin, FS=>P4S(0), FCout=>C1);
    U2: full_adder_structural PORT MAP (FA=>P4A(1), FB=>P4B(1), FCin=>C1, FS=>P4S(1), FCout=>C2);
    U3: full_adder_structural PORT MAP (FA=>P4A(2), FB=>P4B(2), FCin=>C2, FS=>P4S(2), FCout=>C3);
    U4: full_adder_structural PORT MAP (FA=>P4A(3), FB=>P4B(3), FCin=>C3, FS=>P4S(3), FCout=>PCout);

end Structural;
```


Ο VHDL κώδικας για το testbench της structural περιγραφής του 4 bit parallel adder:

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

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

architecture test_four_bit_parallel_adder_structural_testbench of four_bit_parallel_adder_structural_testbench is

    signal PCin_test :std_logic := '0';
    signal P4A_test, P4B_test : std_logic_vector(3 downto 0) := (others => '1');
    signal PCout_test: std_logic;
    signal P4S_test : std_logic_vector(3 downto 0);

    component four_bit_parallel_adder_structural is
        Port ( P4A : in STD_LOGIC_VECTOR (3 downto 0);
              P4B : in STD_LOGIC_VECTOR (3 downto 0);
              PCin : in STD_LOGIC;
              P4S : out STD_LOGIC_VECTOR (3 downto 0);
              PCout : out STD_LOGIC);
    end component;
begin
    uut: four_bit_parallel_adder_structural
        port map(
            P4A=>P4A_test,
            P4B=>P4B_test,
            PCin=>PCin_test,
            P4S=>P4S_test,
            PCout=>PCout_test
        );
    stimulus: process begin
        for i in 0 to 1 loop
            PCin_test <= not PCin_test;
            for j in 0 to 15 loop
                P4A_test <= P4A_test + 1;
                for k in 0 to 15 loop
                    P4B_test <= P4B_test + 1;
                    wait for 2ns;
                end loop;
            end loop;
        end loop;
    end process;
end architecture;
```

Ένα τμήμα της προσομοίωσης (σημαντικό καθώς εξετάζει και περιπτώσεις με κρατούμενο εισόδου και χωρίς αλλά και περιπτώσεις όπου προκύπτει κρατούμενο εξόδου) η οποία εκτελείται εξαντλητικά για όλες τις περιπτώσεις (ωστόσο δε χωράει σε ένα screenshot για να συμπεριληφθεί εξ ολοκλήρου στην αναφορά):

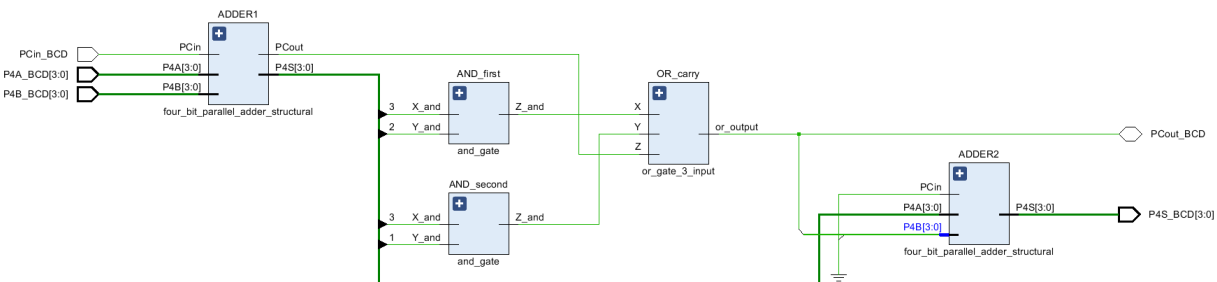


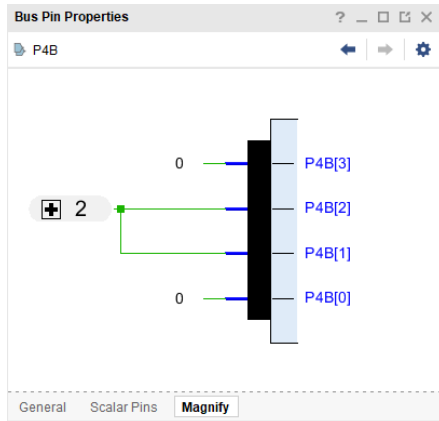
Το critical path του κυκλώματος είναι από το P4B[0] στα P4S[2], P4S[3] δηλαδή από το LSB του δεύτερου προσθετέου στα MSB του τελικού αθροίσματος που είναι λογικό γιατί για να υπολογιστεί το MSB αυτού του αθροίσματος πρέπει να υπολογιστούν ένα ένα και τα 4 αθροίσματα του 1 bit ξεκινώντας από το LSB, με κάθε ένα από αυτά να περιμένει έως ότου το προηγούμενο παράξει το κρατούμενο εξόδου του. Οπότε το MSB εξαρτάται από το κρατούμενο που θα παράξει η πρώτη βαθμίδα και σιγά σιγά θα γίνει propagate ως αυτό. Η συνολική του καθυστέρηση είναι 5.970ns.

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	P4B[0]	P4S[2]	5.970	3.904	2.066	∞	input port clock			0.000
Path 2	∞	4	5	3	P4B[0]	P4S[3]	5.970	3.904	2.066	∞	input port clock			0.000
Path 3	∞	4	5	3	P4B[0]	PCout	5.964	3.898	2.066	∞	input port clock			0.000
Path 4	∞	3	4	3	P4B[0]	P4S[0]	5.351	3.752	1.599	∞	input port clock			0.000
Path 5	∞	3	4	2	P4B[1]	P4S[1]	5.351	3.752	1.599	∞	input port clock			0.000

Ζήτημα 4:

Για να κατασκευάσουμε BCD πλήρη αθροιστή θα χρησιμοποιήσουμε παράλληλους αθροιστές των τεσσάρων bit σε συνδυασμό με λογικές πύλες AND και μια OR 3 εισόδων τις οποίες κατασκευάζουμε με dataflow περιγραφή. Συγκεκριμένα θεωρούμε ότι οι δύο προσθετέοι είναι σε BCD μορφή και τους δίνουμε ως είσοδο στον πρώτο παράλληλο αθροιστή των 4 bit μαζί με το κρατούμενο εισόδου. Αν προκύψει κρατούμενο εξόδου ή είναι 1 τα bit 3 και 2 ή 3 και 1 του αθροίσματος εξόδου (εδώ χρησιμοποιούνται οι 2 AND και η OR 3 εισόδων) τότε το αποτέλεσμα της άθροισης είναι μεγαλύτερο ή ίσο του 10 άρα το κρατούμενο εξόδου τίθεται σε 1. Αν το άθροισμα είναι μεγαλύτερο ή ίσο του 10 σε αυτό προστίθεται το 6 (μέσω ενός δεύτερου παράλληλου αθροιστή 4 bit ο οποίος προσθέτει στο άθροισμα του πρώτου αθροιστή το 6 αν Cin=1 αλλιώς το 0) εκτελώντας με αυτόν τον τρόπο δεκαδική διόρθωση. Το κρατούμενο εξόδου αυτού του αθροιστή δεν έχει κάποια σημασία οπότε αγνοείται. Το RTL Schematic του κυκλώματος που περιεγράφη παραπάνω είναι το εξής:





Ο VHDL κώδικας για την structural περιγραφή του BCD adder:

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

entity BCD_full_adder_structural is
    Port ( P4A_BCD : in STD_LOGIC_VECTOR (3 downto 0);
          P4B_BCD : in STD_LOGIC_VECTOR (3 downto 0);
          PCin_BCD : in STD_LOGIC;
          P4S_BCD : out STD_LOGIC_VECTOR (3 downto 0);
          PCout_BCD : inout STD_LOGIC);
end BCD_full_adder_structural;

architecture Structural_BCD of BCD_full_adder_structural is

    component four_bit_parallel_adder_structural is
        Port ( P4A : in STD_LOGIC_VECTOR (3 downto 0);
              P4B : in STD_LOGIC_VECTOR (3 downto 0);
              PCin : in STD_LOGIC;
              P4S : out STD_LOGIC_VECTOR (3 downto 0);
              PCout : out STD_LOGIC );
    end component;

    component and_gate is
        Port ( X_and : in STD_LOGIC;
              Y_and : in STD_LOGIC;
              Z_and : out STD_LOGIC);
    end component;

    component or_gate_3_input is
        Port ( X : in STD_LOGIC;
              Y : in STD_LOGIC;
              Z : in STD_LOGIC;
              or_output : out STD_LOGIC);
    end component;

    signal and1, and2, carry_of_first_adder, dontcare : STD_LOGIC;
    signal result_of_first_adder, input_of_second_adder : STD_LOGIC_VECTOR(3 downto 0);

begin
    ADDER1: four_bit_parallel_adder_structural PORT MAP (P4A=>P4A_BCD, P4B=>P4B_BCD, PCin=>PCin_BCD, P4S=>result_of_first_adder, PCout=>carry_of_first_adder);
    AND_first: and_gate PORT MAP (X_and =>result_of_first_adder(3) , Y_and=>result_of_first_adder(2), Z_and=>and1);
    AND_second: and_gate PORT MAP (X_and =>result_of_first_adder(3) , Y_and=>result_of_first_adder(1), Z_and=>and2);
    OR_carry: or_gate_3_input PORT MAP (X =>and1, Y=>and2, Z=>carry_of_first_adder, or_output=>PCout_BCD);
    ADDER2: four_bit_parallel_adder_structural PORT MAP (P4A=>result_of_first_adder, P4B(3)=>'0', P4B(2)=>PCout_BCD, P4B(1)=>PCout_BCD, P4B(0)=>'0', PCin=>'0', P4S=>P4S_BCD, PCout=>dontcare);
end Structural_BCD;
```

Ο VHDL κώδικας για την dataflow περιγραφή της πύλης AND:

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

entity and_gate is
    Port ( X_and : in STD_LOGIC;
           Y_and : in STD_LOGIC;
           Z_and : out STD_LOGIC);
end and_gate;

architecture Dataflow_and of and_gate is

begin
    Z_and <= X_and AND Y_and;

end Dataflow_and;
```

Ο VHDL κώδικας για την dataflow περιγραφή της πύλης OR 3 εισόδων:

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

entity or_gate_3_input is
    Port ( X : in STD_LOGIC;
           Y : in STD_LOGIC;
           Z : in STD_LOGIC;
           or_output : out STD_LOGIC);
end or_gate_3_input;

architecture Dataflow_3 of or_gate_3_input is

begin
    or_output <= X OR Y OR Z;

end Dataflow_3;
```

Ο VHDL κώδικας για το testbench της structural περιγραφής του BCD adder:

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

entity BCD_full_adder_structural_testbench is
end BCD_full_adder_structural_testbench;
architecture Behavioral of BCD_full_adder_structural_testbench is

    signal PCin_BCD_test :std_logic := '0';
    signal P4A_BCD_test, P4B_BCD_test : std_logic_vector(3 downto 0) := (others => '1');
    signal PCout_BCD_test: std_logic;
    signal P4S_BCD_test : std_logic_vector(3 downto 0);

    component BCD_full_adder_structural is
        Port ( P4A_BCD : in STD_LOGIC_VECTOR (3 downto 0);
              P4B_BCD : in STD_LOGIC_VECTOR (3 downto 0);
              PCin_BCD : in STD_LOGIC;
              P4S_BCD : out STD_LOGIC_VECTOR (3 downto 0);
              PCout_BCD : inout STD_LOGIC);
    end component;

begin

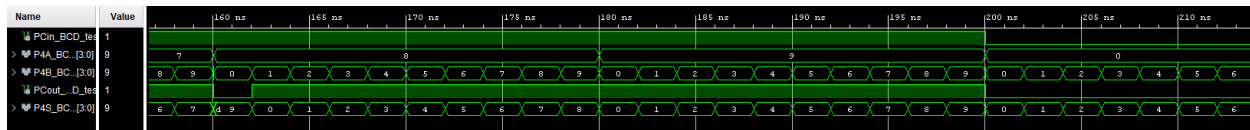
    uut: BCD_full_adder_structural
        port map(
            P4A_BCD=>P4A_BCD_test,
            P4B_BCD=>P4B_BCD_test,
            PCin_BCD=>PCin_BCD_test,
            P4S_BCD=>P4S_BCD_test,
            PCout_BCD=>PCout_BCD_test
        );

    stimulus: process begin
        for i in 0 to 1 loop
            PCin_BCD_test <= not PCin_BCD_test;
            for j in 0 to 9 loop
                P4A_BCD_test <= P4A_BCD_test + 1;
                for k in 0 to 9 loop
                    P4B_BCD_test <= P4B_BCD_test + 1;
                    wait for 2ns;
                end loop;
                P4B_BCD_test <= "1111";
                wait for 2ps;
            end loop;
            P4A_BCD_test <= "1111";
            wait for 2ps;
        end loop;

    end process;

end architecture;
```

Πάλι παρουσιάζουμε ένα κρίσιμο τμήμα της προσομοίωσης καθώς η εξαντλητική είναι πολύ μεγάλη:

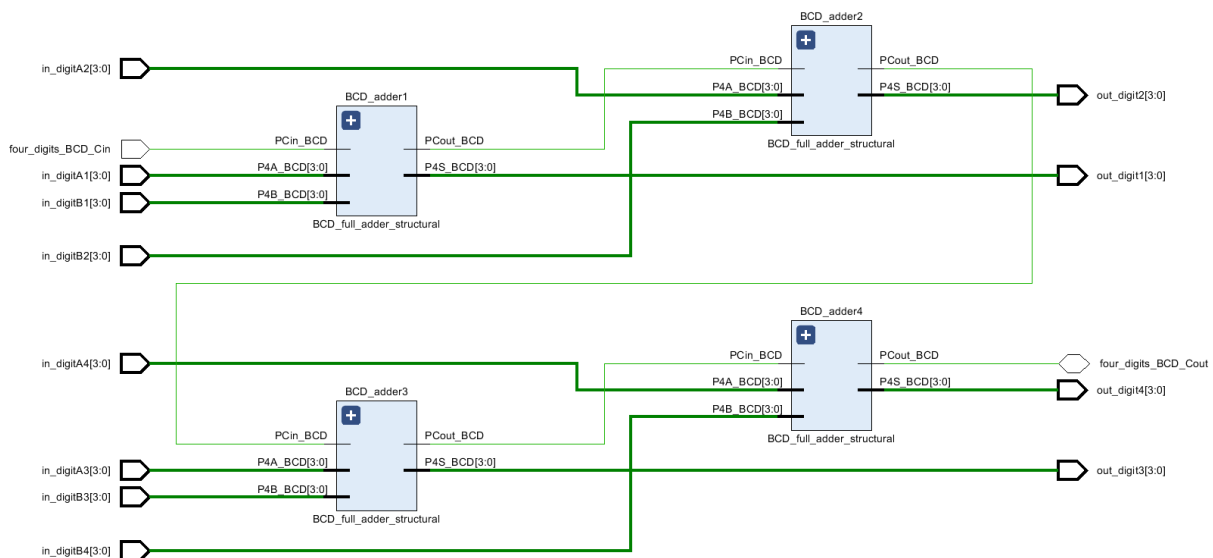


Το critical path του κυκλώματος είναι το μονοπάτι από το P4B_BCD[0] (LSB δεύτερου προσθετέου) στο P4S_BCD[3] (MSB αθροίσματος εξόδου) που είναι λογικό καθώς για τον υπολογισμό του αθροίσματος εξόδου μεσολαβούν δύο παράλληλοι αθροιστές των 4 bit που για κάθε έναν το πιο χρονοβόρο μονοπάτι είναι ο υπολογισμός του τελευταίου bit του αθροίσματος όπως εξηγήθηκε παραπάνω ενώ πχ το κρατούμενο εξόδου εδώ υπολογίζεται διατρέχοντας μόνο έναν παράλληλο αθροιστή. Η συνολική του καθυστέρηση είναι 7.14ns.

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	∞	6	7	4	P4B_BCD[0]	P4S_BCD[3]	7.140	4.146	2.994	∞	input port clock			0.000
Path 2	∞	5	6	4	P4B_BCD[0]	P4S_BCD[1]	6.573	4.028	2.545	∞	input port clock			0.000
Path 3	∞	5	6	4	P4B_BCD[0]	P4S_BCD[2]	6.573	4.028	2.545	∞	input port clock			0.000
Path 4	∞	4	5	4	P4B_BCD[0]	PCout_BCD	5.976	3.904	2.072	∞	input port clock			0.000
Path 5	∞	3	4	3	P4B_BCD[0]	P4S_BCD[0]	5.351	3.752	1.599	∞	input port clock			0.000

Ζήτημα 5:

Για να υλοποιήσουμε τον παράλληλο BCD αθροιστή των 4 ψηφίων απλώς διασυνδέουμε 4 BCD Full Adders του προηγούμενου ερωτήματος με το κρατούμενο εξόδου του LSB να είναι κρατούμενο εισόδου του 2ου LSB κ.ο.κ. εντελώς αντίστοιχα με τη λογική του ζητήματος 3. Το RTL schematic αυτής της δομής:



Ο VHDL κώδικας για την structural περιγραφή του four digits BCD adder:

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

entity four_bit_parallel_BCD_adder_structural is
    Port ( in_digitA1 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitA2 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitA3 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitA4 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitB1 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitB2 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitB3 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitB4 : in STD_LOGIC_VECTOR (3 downto 0);
          out_digit1 : out STD_LOGIC_VECTOR (3 downto 0);
          out_digit2 : out STD_LOGIC_VECTOR (3 downto 0);
          out_digit3 : out STD_LOGIC_VECTOR (3 downto 0);
          out_digit4 : out STD_LOGIC_VECTOR (3 downto 0);
          four_digits_BCD_Cin : in STD_LOGIC;
          four_digits_BCD_Cout : inout STD_LOGIC);
end four_bit_parallel_BCD_adder_structural;

architecture Structural of four_bit_parallel_BCD_adder_structural is

    component BCD_full_adder_structural is
        Port ( P4A_BCD : in STD_LOGIC_VECTOR (3 downto 0);
              P4B_BCD : in STD_LOGIC_VECTOR (3 downto 0);
              PCin_BCD : in STD_LOGIC;
              P4S_BCD : out STD_LOGIC_VECTOR (3 downto 0);
              PCout_BCD : inout STD_LOGIC);
    end component;

    signal C1, C2, C3 : STD_LOGIC;

begin

    BCD_adder1: BCD_full_adder_structural PORT MAP (P4A_BCD=>in_digitA1, P4B_BCD=>in_digitB1, PCin_BCD=>four_digits_BCD_Cin, P4S_BCD=>out_digit1, PCout_BCD=>C1);
    BCD_adder2: BCD_full_adder_structural PORT MAP (P4A_BCD=>in_digitA2, P4B_BCD=>in_digitB2, PCin_BCD=>C1, P4S_BCD=>out_digit2, PCout_BCD=>C2);
    BCD_adder3: BCD_full_adder_structural PORT MAP (P4A_BCD=>in_digitA3, P4B_BCD=>in_digitB3, PCin_BCD=>C2, P4S_BCD=>out_digit3, PCout_BCD=>C3);
    BCD_adder4: BCD_full_adder_structural PORT MAP (P4A_BCD=>in_digitA4, P4B_BCD=>in_digitB4, PCin_BCD=>C3, P4S_BCD=>out_digit4, PCout_BCD=>four_digits_BCD_Cout);

end Structural;
```

Ο VHDL κώδικας για το testbench της structural περιγραφής του four digits BCD adder:

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

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

architecture Behavioral of four_bit_parallel_BCD_adder_structural_testbench is

signal four_digits_BCD_Cin_test : STD_LOGIC := '0';

signal in_digitA4_test, in_digitA3_test, in_digitA2_test, in_digitA1_test,
       in_digitB4_test, in_digitB3_test, in_digitB2_test, in_digitB1_test,
       out_digit4_test, out_digit3_test, out_digit2_test, out_digit1_test : STD_LOGIC_VECTOR(3 downto 0);

signal four_digits_BCD_Cout_test : STD_LOGIC;

component four_bit_parallel_BCD_adder_structural is
    Port ( in_digitA1 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitA2 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitA3 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitA4 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitB1 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitB2 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitB3 : in STD_LOGIC_VECTOR (3 downto 0);
          in_digitB4 : in STD_LOGIC_VECTOR (3 downto 0);
          out_digit1 : out STD_LOGIC_VECTOR (3 downto 0);
          out_digit2 : out STD_LOGIC_VECTOR (3 downto 0);
          out_digit3 : out STD_LOGIC_VECTOR (3 downto 0);
          out_digit4 : out STD_LOGIC_VECTOR (3 downto 0);
          four_digits_BCD_Cin : in STD_LOGIC;
          four_digits_BCD_Cout : inout STD_LOGIC);
end component;
```



```

begin

uut: four_bit_parallel_BCD_adder_structural
    port map(
        in_digitA1=>in_digitA1_test,
        in_digitA2=>in_digitA2_test,
        in_digitA3=>in_digitA3_test,
        in_digitA4=>in_digitA4_test,
        in_digitB1=>in_digitB1_test,
        in_digitB2=>in_digitB2_test,
        in_digitB3=>in_digitB3_test,
        in_digitB4=>in_digitB4_test,
        out_digit1=>out_digit1_test,
        out_digit2=>out_digit2_test,
        out_digit3=>out_digit3_test,
        out_digit4=>out_digit4_test,
        four_digits_BCD_Cin=>four_digits_BCD_Cin_test,
        four_digits_BCD_Cout=>four_digits_BCD_Cout_test
    );

stimulus: process begin
    for i in 0 to 1 loop
        four_digits_BCD_Cin_test <= not four_digits_BCD_Cin_test;
        in_digitA1_test<="1001";
        in_digitA2_test<="1001";
        in_digitA3_test<="1001";
        in_digitA4_test<="1001";
        in_digitB1_test<="1001";
        in_digitB2_test<="1001";
        in_digitB3_test<="1001";
        in_digitB4_test<="1001";
        wait for 2ns;
    end loop;
end process;

```

```

        in_digitA1_test<="1001";
        in_digitA2_test<="1001";
        in_digitA3_test<="1001";
        in_digitA4_test<="1001";
        in_digitB1_test<="0001";
        in_digitB2_test<="0000";
        in_digitB3_test<="0000";
        in_digitB4_test<="0000";
        wait for 2ns;

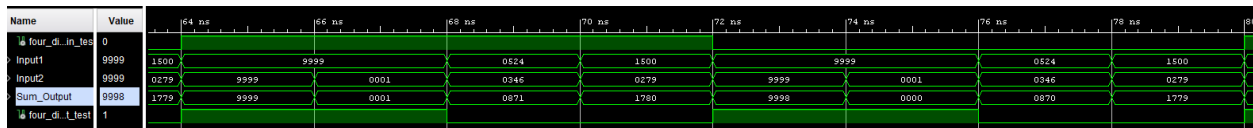
        in_digitA1_test<="0100";
        in_digitA2_test<="0010";
        in_digitA3_test<="0101";
        in_digitA4_test<="0000";
        in_digitB1_test<="0110";
        in_digitB2_test<="0100";
        in_digitB3_test<="0011";
        in_digitB4_test<="0000";
        wait for 2ns;

        in_digitA1_test<="0000";
        in_digitA2_test<="0000";
        in_digitA3_test<="0101";
        in_digitA4_test<="0001";
        in_digitB1_test<="1001";
        in_digitB2_test<="0111";
        in_digitB3_test<="0010";
        in_digitB4_test<="0000";
        wait for 2ns;
    }
    end loop;
}
end process;







} end Behavioral;

```

Τώρα για να ελέγξουμε το κύκλωμα, επειδή εξαντλητικά οι περιπτώσεις θα ήταν υπερβολικά πολλές ($10000 \cdot 10000 \cdot 2$), προσομοιώσαμε μόνο ορισμένες που θεωρήθηκαν κρίσιμες, όπως την άθροιση αριθμών που το αποτέλεσμα δε χωρά σε 4 δεκαδικά ψηφία και κάποιες περιπτώσεις αναμενόμενης λειτουργίας, με και χωρίς κρατούμενο εισόδου.



Το critical path του κυκλώματος είναι από το in_digitB1[1] (2ο LSB του πρώτου ψηφίου του δεύτερου προσθετέου) ως τα out_digit4[2], out_digit4[3] (MSB του τελευταίου ψηφίου του αθροίσματος εξόδου) καθώς τόσο εντός των BCD Full Adders όσο και μεταξύ τους πρέπει το κρατούμενο να διαδίδεται από τα λιγότερο σημαντικά bits ή ψηφία στα περισσότερα. Η χρονική του καθυστέρηση είναι 11.262ns.

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	∞	13	14	7	in_digitB1[1]	out_digit4[2]	11.262	4.992	6.270	∞	input port clock			0.000
 Path 2	∞	13	14	7	in_digitB1[1]	out_digit4[3]	11.262	4.992	6.270	∞	input port clock			0.000
 Path 3	∞	12	13	7	in_digitB1[1]	out_digit4[1]	10.689	4.868	5.821	∞	input port clock			0.000
 Path 4	∞	11	12	7	in_digitB1[1]	four_digits_BCD_Cout	10.088	4.744	5.344	∞	input port clock			0.000
 Path 5	∞	9	10	7	in_digitB1[1]	out_digit3[1]	8.931	4.496	4.435	∞	input port clock			0.000
 Path 6	∞	9	10	7	in_digitB1[1]	out_digit3[2]	8.931	4.496	4.435	∞	input port clock			0.000