A black background with a black square

Description automatically generated with medium confidence **Εθνικό Μετσόβιο Πολυτεχνείο**

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εικόνα που περιέχει ηλεκτρονικές συσκευές, ηλεκτρονικό εξάρτημα, στοιχείο κυκλώματος, παθητικό στοιχείο κυκλώματος

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

**Μάθημα: Ψηφιακά Συστήματα VLSI**

**Θέμα: 2η Εργαστηριακή Άσκηση – Σχεδιασμός Αριθμητικών Μονάδων με Ιεραρχική Σχεδίαση**

Εξάμηνο: 8ο

Ομάδα: 6

Συνεργάτες:

* Ακύλας Αντώνιος 03121152
* Κουμπιάς Ιωάννης-Χρυσοβαλάντης 03121053

**Άσκηση 1**

Στο πρώτο κομμάτι του εργαστηρίου σχεδιάσαμε έναν Ημιαθροιστή (Half Adder - HA) σε περιγραφή ροής δεδομένων (Dataflow). Παρουσιάζουμε τον κώδικα VHDL καθώς και το RTL Schematic του κυκλώματος που προκύπτει:

library ieee;

use ieee.std\_logic\_1164.all;

entity half\_adder is

    port(

        a: in std\_logic;

        b: in std\_logic;

        s: out std\_logic;

        c: out std\_logic

    );

end half\_adder;

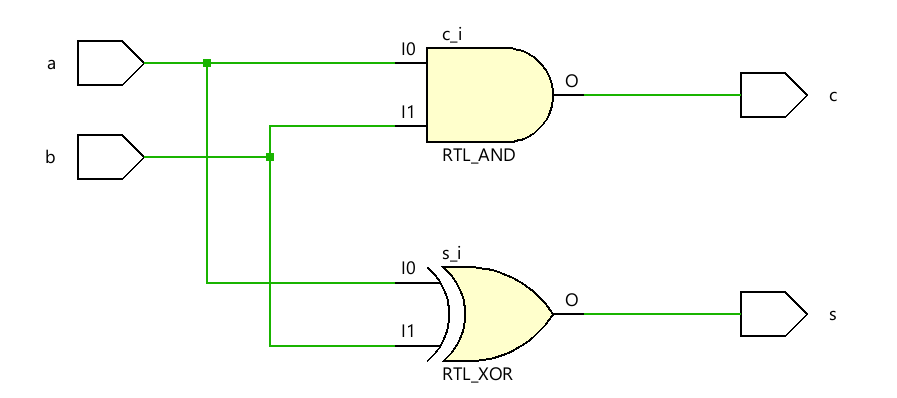
architecture dataflow of half\_adder is

begin

    s <= a xor b;

    c <= a and b;

end dataflow;



Στην συνέχεια δημιουργήσαμε κατάλληλο testbench που παράγει όλα τα διαφορετικά inputs για τον έλεγχο της ορθής λειτουργίας του κυκλώματος. Ακολουθούν ο κώδικας VHDL του testbench καθώς και τα αποτελέσματα του simulation:

library ieee;

use ieee.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

entity ha\_tb is

end ha\_tb;

architecture ha\_tb\_arch of ha\_tb is

    component half\_adder

        port(

            a: in std\_logic;

            b: in std\_logic;

            s: out std\_logic;

            c: out std\_logic

        );

    end component;

    signal in\_s: std\_logic\_vector(1 downto 0);

    signal s\_s,c\_s: std\_logic;

begin

    UUT: half\_adder port map (in\_s(0),in\_s(1),s\_s,c\_s);

    testSequence: process

        begin

            in\_s <= "00";

            for i in 0 to 3 loop

                wait for 10 ns;

                in\_s <= in\_s + 1;

            end loop;

    end process;

end ha\_tb\_arch;

A screenshot of a video game

AI-generated content may be incorrect.

Έχοντας βεβαιωθεί πως το κύκλωμα λειτουργεί όπως είναι αναμενόμενο, προχωράμε στη διαδικασία σύνθεσης με στόχο τον υπολογισμό του critical path και κατ’ επέκταση της μέγιστης χρονικής καθυστέρησης με την βοήθεια του Vivado. Παραθέτουμε το schematic της σύνθεσης αξιοποιώντας πλέον πόρους του FPGA και τις μετρήσεις των χρονικών καθυστερήσεων ανά μονοπάτι:

A diagram of a computer circuit

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Το critical path της σύνθεσης φάινεται πρώτο στα Unconstrained Paths του timing report. Στις μετρήσεις συνυπολογίζονται τόσο η καθυστέρηση λόγω λογικής (Logic Delay) όσο και η καθυστέρηση λόγω του routing των συνδέσεων στο FPGA (Net Delay). Παρατηρούμε πως το κρίσιμο μονοπάτι είναι από το input b προς το output s με μικρή διαφορά από το b – c (που οφείλεται στη λογική), γεγονός που αιτιολογείται από την ομοιότητα των δύο μονοπατιών.

**Άσκηση 2**

Στο δεύτερο κομμάτι του εργαστηρίου σχεδιάσαμε έναν Πλήρη Αθροιστή (Full Adder - FA) σε περιγραφή δομής (Structural) βασιζόμενοι στην δομική μονάδα της προηγούμενης άσκησης. Παρουσιάζουμε τον κώδικα VHDL καθώς και το RTL Schematic του κυκλώματος που προκύπτει:

library ieee;

use ieee.std\_logic\_1164.all;

entity full\_adder is

    port(

        a: in std\_logic;

        b: in std\_logic;

        cin: in std\_logic;

        s: out std\_logic;

        cout: out std\_logic

    );

end full\_adder;

architecture structural of full\_adder is

    component half\_adder

        port(

            a: in std\_logic;

            b: in std\_logic;

            s: out std\_logic;

            c: out std\_logic

        );

    end component;

    signal sum,c1,c2: std\_logic;

begin

    s1: half\_adder port map(

        a => a,

        b => b,

        s => sum,

        c => c1

    );

    s2: half\_adder port map(

        a => sum,

        b => cin,

        s => s,

        c => c2

    );

    cout <= c1 or c2;

end structural;

A diagram of a diagram

AI-generated content may be incorrect.

Στην συνέχεια δημιουργήσαμε κατάλληλο testbench που παράγει όλα τα διαφορετικά inputs για τον έλεγχο της ορθής λειτουργίας του κυκλώματος. Ακολουθούν ο κώδικας VHDL του testbench καθώς και τα αποτελέσματα του simulation:

library ieee;

use ieee.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

entity fa\_tb is

end fa\_tb;

architecture fa\_tb\_arch of fa\_tb is

    component full\_adder

        port(

            a: in std\_logic;

            b: in std\_logic;

            cin: in std\_logic;

            s: out std\_logic;

            cout: out std\_logic

        );

    end component;

    signal in\_s: std\_logic\_vector(2 downto 0);

    signal s\_s,c\_s: std\_logic;

begin

    UUT: full\_adder port map (in\_s(0),in\_s(1),in\_s(2),s\_s,c\_s);

    testSequence: process

        begin

            in\_s <= "000";

            for i in 0 to 7 loop

                wait for 10 ns;

                in\_s <= in\_s + 1;

            end loop;

    end process;

end fa\_tb\_arch;

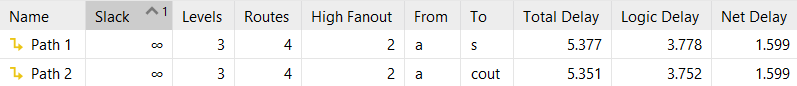
A screenshot of a video game

AI-generated content may be incorrect.

Έχοντας βεβαιωθεί πως το κύκλωμα λειτουργεί όπως είναι αναμενόμενο, προχωράμε στη διαδικασία σύνθεσης με στόχο τον υπολογισμό του critical path και κατ’ επέκταση της μέγιστης χρονικής καθυστέρησης με την βοήθεια του Vivado. Παραθέτουμε το schematic της σύνθεσης αξιοποιώντας πλέον πόρους του FPGA και τις μετρήσεις των χρονικών καθυστερήσεων ανά μονοπάτι:

A diagram of a computer

AI-generated content may be incorrect.



Παρατηρούμε πως το κρίσιμο μονοπάτι είναι από το input a προς το output s με μικρή διαφορά από το a – cout (που οφείλεται στη λογική), γεγονός που αιτιολογείται από την ομοιότητα των δύο μονοπατιών.

**Άσκηση 3**

Στο τρίτο κομμάτι του εργαστηρίου σχεδιάσαμε έναν Παράλληλο Αθροιστή των 4 bits (4-bit Parallel Adder – 4-bit PA) σε περιγραφή δομής (Structural) βασιζόμενοι στην δομική μονάδα της προηγούμενης άσκησης. Παρουσιάζουμε τον κώδικα VHDL καθώς και το RTL Schematic του κυκλώματος που προκύπτει:

library ieee;

use ieee.std\_logic\_1164.all;

entity par\_adder4 is

    port(

        a: in std\_logic\_vector(3 downto 0);

        b: in std\_logic\_vector(3 downto 0);

        cin: in std\_logic;

        s: out std\_logic\_vector(3 downto 0);

        cout: out std\_logic

    );

end par\_adder4;

architecture structural of par\_adder4 is

    component full\_adder

        port(

            a: in std\_logic;

            b: in std\_logic;

            cin: in std\_logic;

            s: out std\_logic;

            cout: out std\_logic

        );

    end component;

    signal c: std\_logic\_vector(3 downto 1);

begin

    p0: full\_adder port map(a(0),b(0),cin,s(0),c(1));

    p1: full\_adder port map(a(1),b(1),c(1),s(1),c(2));

    p2: full\_adder port map(a(2),b(2),c(2),s(2),c(3));

    p3: full\_adder port map(a(3),b(3),c(3),s(3),cout);

end structural;

A diagram of a computer program

AI-generated content may be incorrect.

Στην συνέχεια δημιουργήσαμε κατάλληλο testbench που παράγει όλα τα διαφορετικά inputs για τον έλεγχο της ορθής λειτουργίας του κυκλώματος. Ακολουθούν ο κώδικας VHDL του testbench καθώς και κάποια αντιπροσωπευτικά αποτελέσματα του simulation:

library ieee;

use ieee.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

entity pa4\_tb is

end pa4\_tb;

architecture pa4\_tb\_arch of pa4\_tb is

    component par\_adder4

        port(

            a: in std\_logic\_vector(3 downto 0);

            b: 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 a\_s: std\_logic\_vector(3 downto 0);

    signal b\_s: std\_logic\_vector(3 downto 0);

    signal cin\_s: std\_logic;

    signal s\_s: std\_logic\_vector(3 downto 0);

    signal cout\_s: std\_logic;

begin

    UUT: par\_adder4 port map (a\_s,b\_s,cin\_s,s\_s,cout\_s);

    testSequence: process

        begin

            a\_s <= "0000";

            b\_s <= "0000";

            cin\_s <= '0';

            for i in 0 to 15 loop

                for j in 0 to 15 loop

                    wait for 1 ns;

                    a\_s <= a\_s + 1;

                end loop;

                b\_s <= b\_s + 1;

            end loop;

            cin\_s <= '1';

            for i in 0 to 15 loop

                for j in 0 to 15 loop

                    wait for 1 ns;

                    a\_s <= a\_s + 1;

                end loop;

                b\_s <= b\_s + 1;

            end loop;

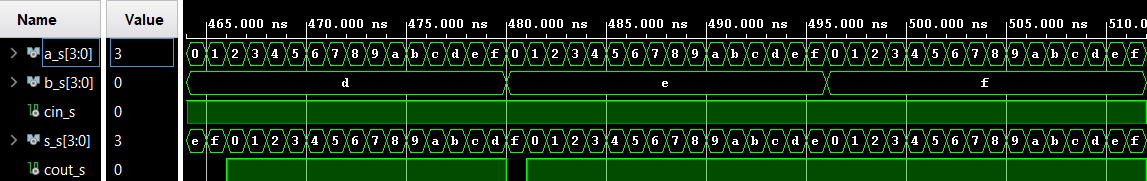
    end process;

end pa4\_tb\_arch;

A screen shot of a computer

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.



Έχοντας βεβαιωθεί πως το κύκλωμα λειτουργεί όπως είναι αναμενόμενο, προχωράμε στη διαδικασία σύνθεσης με στόχο τον υπολογισμό του critical path και κατ’ επέκταση της μέγιστης χρονικής καθυστέρησης με την βοήθεια του Vivado. Παραθέτουμε το schematic της σύνθεσης αξιοποιώντας πλέον πόρους του FPGA και τις μετρήσεις των χρονικών καθυστερήσεων ανά μονοπάτι:

A diagram of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Παρατηρούμε πως το κρίσιμο μονοπάτι είναι από το input b[0] προς το output cout με μικρή διαφορά από το b[0] – s[2] και b[0] – s[3] (που οφείλεται στη λογική). Από την άλλη τα μονοπάτια b[0] – s[0] και b[1] – s[1] φαίνεται να έχουν μικρότερες καθυστερήσεις τόσο στη λογική όσο και στη δικτύωσή τους.

**Άσκηση 4**

Στο τέταρτο κομμάτι του εργαστηρίου σχεδιάσαμε έναν BCD Πλήρη Αθροιστή (BCD Full Adder – BCD FA) σε περιγραφή δομής (Structural). Η υλοποίηση βασίζεται τόσο στην δομική μονάδα του 4-bit Parallel Adder της προηγούμενης άσκησης όσο και σε επιπλέον λογική για την μετατροπή των αποτελεσμάτων της άθροισης σε BCD. Συγκεκριμένα εκτελούμε κανονικά την δυαδική άθροιση και στην συνέχεια αν το αποτέλεσμα είναι μεγαλύτερο του 10 τότε προσθέτουμε το 0110 (δυαδική αναπαράσταση του συμπληρώματος ως προς 2 του 10 – αφαίρεση) αλλιώς το 0000 (δηλαδή αφήνουμε το αποτέλεσμα όπως έχει). Τέλος έχουμε κρατούμενο εάν προκύψει κρατούμενο από κάποια εκ των προσθέσεων. Ιδιαίτερο ενδιαφέρον παρουσιάζει ο τρόπος ελέγχου ο οποίος βασίζεται στην παρατήρηση πως το αποτέλεσμα της πρώτης πρόσθεσης είναι μεγαλύτερο ή ίσο του 10 ανν προκύπτει κρατούμενο ή το 3ο bit είναι 1 και τουλάχιστον ένα από τα bit 2 ή 1 είναι 1. Παρουσιάζουμε τον κώδικα VHDL καθώς και το RTL Schematic του κυκλώματος που προκύπτει:

library ieee;

use ieee.std\_logic\_1164.all;

entity bcd\_fa is

    port(

        a: in std\_logic\_vector(3 downto 0);

        b: in std\_logic\_vector(3 downto 0);

        cin: in std\_logic;

        s: out std\_logic\_vector(3 downto 0);

        cout: out std\_logic

    );

end bcd\_fa;

architecture structural of bcd\_fa is

    component par\_adder4

        port(

            a: in std\_logic\_vector(3 downto 0);

            b: 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 s\_inter: std\_logic\_vector(3 downto 0);

    signal corr: std\_logic\_vector(3 downto 0);

    signal c\_inter,c\_bcd,comp: std\_logic;

begin

    add: par\_adder4 port map(a,b,cin,s\_inter,c\_inter);

    comp <= c\_inter or (s\_inter(3) and (s\_inter(2) or s\_inter(1)));

    with comp select corr <=

        "0000" when '0',

        "0110" when others;

    final: par\_adder4 port map(s\_inter,corr,'0',s,c\_bcd);

    cout <= c\_inter or c\_bcd;

end structural;

A diagram of a circuit

AI-generated content may be incorrect.

Στην συνέχεια δημιουργήσαμε κατάλληλο testbench που παράγει όλα τα διαφορετικά inputs για τον έλεγχο της ορθής λειτουργίας του κυκλώματος. Ακολουθούν ο κώδικας VHDL του testbench καθώς και αντιπροσωπευτικά αποτελέσματα του simulation:

library ieee;

use ieee.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

entity bcd\_fa\_tb is

end bcd\_fa\_tb;

architecture bcd\_fa\_tb\_arch of bcd\_fa\_tb is

    component bcd\_fa

        port(

            a: in std\_logic\_vector(3 downto 0);

            b: 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 a\_s: std\_logic\_vector(3 downto 0);

    signal b\_s: std\_logic\_vector(3 downto 0);

    signal cin\_s: std\_logic;

    signal s\_s: std\_logic\_vector(3 downto 0);

    signal cout\_s: std\_logic;

begin

    UUT: bcd\_fa port map (a\_s,b\_s,cin\_s,s\_s,cout\_s);

    testSequence: process

        begin

            b\_s <= "0000";

            cin\_s <= '0';

            for i in 0 to 9 loop

                a\_s <= "0000";

                for j in 0 to 9 loop

                    wait for 1 ns;

                    a\_s <= a\_s + 1;

                end loop;

                b\_s <= b\_s + 1;

            end loop;

            b\_s <= "0000";

            cin\_s <= '1';

            for i in 0 to 9 loop

                a\_s <= "0000";

                for j in 0 to 9 loop

                    wait for 1 ns;

                    a\_s <= a\_s + 1;

                end loop;

                b\_s <= b\_s + 1;

            end loop;

    end process;

end bcd\_fa\_tb\_arch;

A screenshot of a video game

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a video game

AI-generated content may be incorrect.

Έχοντας βεβαιωθεί πως το κύκλωμα λειτουργεί όπως είναι αναμενόμενο, προχωράμε στη διαδικασία σύνθεσης με στόχο τον υπολογισμό του critical path και κατ’ επέκταση της μέγιστης χρονικής καθυστέρησης με την βοήθεια του Vivado. Παραθέτουμε το schematic της σύνθεσης αξιοποιώντας πλέον πόρους του FPGA και τις μετρήσεις των χρονικών καθυστερήσεων ανά μονοπάτι:

A diagram of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Παρατηρούμε πως τα κρίσιμα μονοπάτια είναι τα b[0] – cout και b[0] – s[3] με μικρή διαφορά από τα b[1] – s[1] και b[1] – s[2] (οφείλεται στη λογική). Από την άλλη το μονοπάτι b[0] – s[0] φαίνεται να έχει μικρότερες καθυστερήσεις τόσο στη λογική όσο και στη δικτύωσή του.

**Άσκηση 5**

Στο πέμπτο κομμάτι του εργαστηρίου σχεδιάσαμε έναν Παράλληλο BCD Αθροιστή των 4 ψηφίων (4-BCD Parallel Adder – 4-BCD PA) σε περιγραφή δομής (Structural) βασιζόμενοι στην δομική μονάδα της προηγούμενης άσκησης. Παρουσιάζουμε τον κώδικα VHDL καθώς και το RTL Schematic του κυκλώματος που προκύπτει:

library ieee;

use ieee.std\_logic\_1164.all;

entity bcd4 is

    port(

        a: in std\_logic\_vector(15 downto 0);

        b: in std\_logic\_vector(15 downto 0);

        cin: in std\_logic;

        s: out std\_logic\_vector(15 downto 0);

        cout: out std\_logic

    );

end bcd4;

architecture structural of bcd4 is

    component bcd\_fa

        port(

            a: in std\_logic\_vector(3 downto 0);

            b: 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 c: std\_logic\_vector(3 downto 1);

begin

    b0: bcd\_fa port map(a(3 downto 0),b(3 downto 0),cin,s(3 downto 0),c(1));

    b1: bcd\_fa port map(a(7 downto 4),b(7 downto 4),c(1),s(7 downto 4),c(2));

    b2: bcd\_fa port map(a(11 downto 8),b(11 downto 8),c(2),s(11 downto 8),c(3));

    b3: bcd\_fa port map(a(15 downto 12),b(15 downto 12),c(3),s(15 downto 12),cout);

end structural;

A diagram of a computer program

AI-generated content may be incorrect.

Στην συνέχεια δημιουργήσαμε κατάλληλο αντιπροσωπευτικό testbench που παράγει 6 τυχαία inputs για τον έλεγχο της ορθής λειτουργίας του κυκλώματος. Ακολουθούν ο κώδικας VHDL του testbench καθώς και τα αποτελέσματα του simulation:

library ieee;

use ieee.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity bcd4\_tb is

end bcd4\_tb;

architecture bcd4\_tb\_arch of bcd4\_tb is

    component bcd4

        port(

            a: in std\_logic\_vector(15 downto 0);

            b: in std\_logic\_vector(15 downto 0);

            cin: in std\_logic;

            s: out std\_logic\_vector(15 downto 0);

            cout: out std\_logic

        );

    end component;

    signal a\_s: std\_logic\_vector(15 downto 0);

    signal b\_s: std\_logic\_vector(15 downto 0);

    signal cin\_s: std\_logic;

    signal s\_s: std\_logic\_vector(15 downto 0);

    signal cout\_s: std\_logic;

begin

    UUT: bcd4 port map (a\_s,b\_s,cin\_s,s\_s,cout\_s);

    testSequence: process

        begin

            cin\_s <= '0';

            a\_s <= X"0123";

            b\_s <= X"6789";

            wait for 10 ns;

            cin\_s <= '1';

            a\_s <= X"4567";

            b\_s <= X"2345";

            wait for 10 ns;

            cin\_s <= '0';

            a\_s <= X"0345";

            b\_s <= X"6789";

            wait for 10 ns;

            cin\_s <= '1';

            a\_s <= X"7890";

            b\_s <= X"9876";

            wait for 10 ns;

            cin\_s <= '0';

            a\_s <= X"0987";

            b\_s <= X"6543";

            wait for 10 ns;

            cin\_s <= '1';

            a\_s <= X"1357";

            b\_s <= X"2468";

            wait for 10 ns;

    end process;

end bcd4\_tb\_arch;

A screen shot of a video player

AI-generated content may be incorrect.

Έχοντας βεβαιωθεί πως το κύκλωμα λειτουργεί όπως είναι αναμενόμενο, προχωράμε στη διαδικασία σύνθεσης με στόχο τον υπολογισμό του critical path και κατ’ επέκταση της μέγιστης χρονικής καθυστέρησης με την βοήθεια του Vivado. Παραθέτουμε το schematic της σύνθεσης αξιοποιώντας πλέον πόρους του FPGA και τις μετρήσεις των χρονικών καθυστερήσεων ανά μονοπάτι:

A diagram of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Παρατηρούμε πως το κρίσιμο μονοπάτι είναι από το cin προς το s[14] με αρκετή καθυστέρηση τόσο εξαιτίας της λογικής όσο και του τρόπου δικτύωσης, σε σχέση με τα υπόλοιπα μονοπάτια του πίνακα μετρήσεων.