Εθνικό Μετσόβιο Πολυτεχνείο, ΣΗΜΜΥ
Ψηφιακά Συστήματα VLSI

3η Εργαστηριακή Άσκηση:
Σχεδίαση Μονάδων Υλικού με την Τεχνική Pipelining

Ημ/νια : 20.04.2024
Ομάδα 19
Θεοδώρα Εξάρχου, ΑΜ : 03120865
Παναγιώτης Μπέλσης, ΑΜ: 03120874

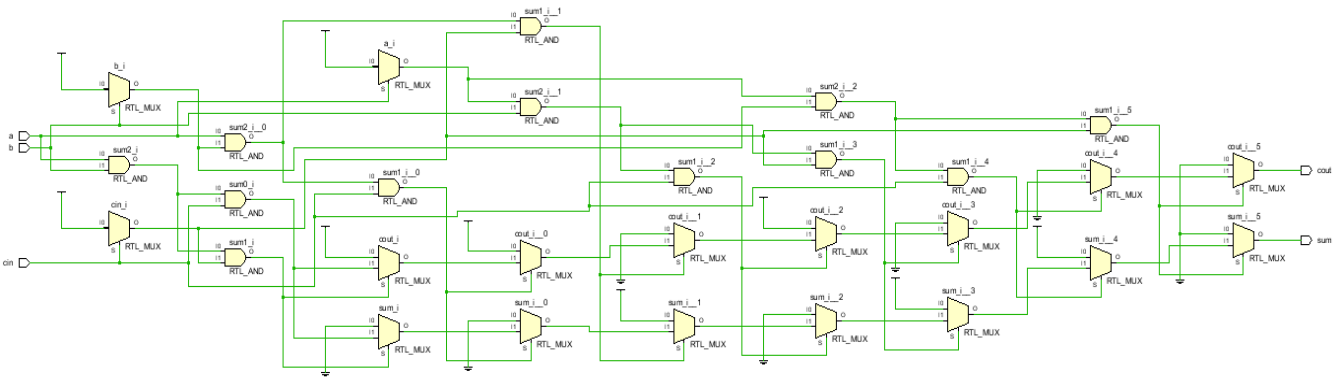
1) Υλοποιήστε έναν σύγχρονο Πλήρη Αθροιστή (Full Adder - FA) με περιγραφή συμπεριφοράς (Behavioral).

κώδικας FullAdder σε VHDL:

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

entity Pipelined_Full_Adder is
Port ( cin : in STD_LOGIC;
sum : out STD_LOGIC;
a : in STD_LOGIC;
b : in STD_LOGIC;
cout : out STD_LOGIC);
end Pipelined_Full_Adder;

architecture Behavioral of Pipelined_Full_Adder is
begin

process(a,b,cin)
begin

if(a='0' and b='0' and cin='0')then
sum<='0';
cout<='0';
elsif( a='0' and b='0' and cin='1')then
sum<='1';
cout<='0';
elsif( a='0' and b='1' and cin='0')then
sum<='1';
```

```
cout<='0';
elsif( a='0' and b='1' and cin='1')then
sum<='0';
cout<='1';
elsif( a='1' and b='0' and cin='0')then
sum<='1';
cout<='0';
elsif( a='1' and b='0' and cin='1')then
sum<='0';
cout<='1';
elsif( a='1' and b='1' and cin='0')then
sum<='0';
cout<='1';
elsif ( a='1' and b='1' and cin='1')then
sum<='1';
cout<='1';
else
sum<='U';
cout<='U';
end if;

end process;
end Behavioral;
```

Παρουσίαση δομικού διαγράμματος (RTL schematic) του Πλήρη Αθροιστή.



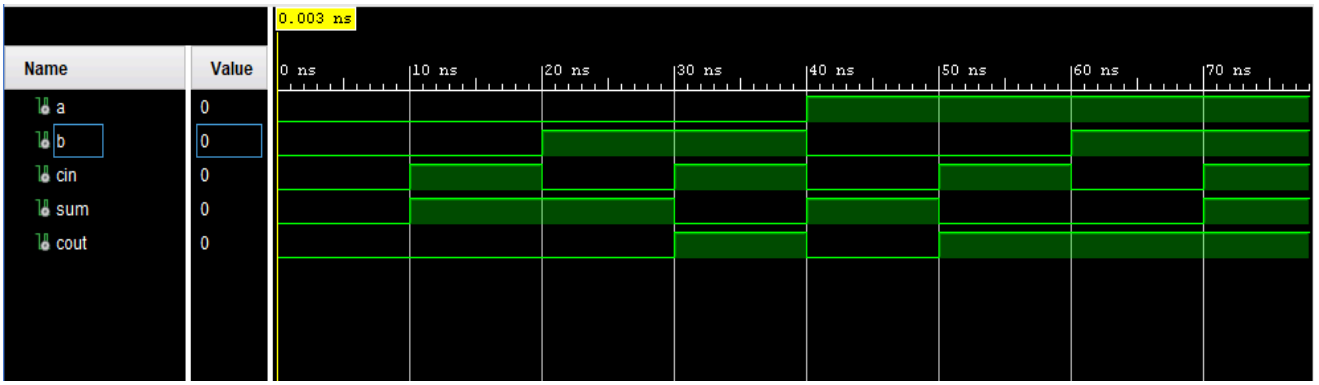
Δημιουργία testbench, με το οποίο γίνεται ο έλεγχος ορθής λειτουργίας του κυκλώματος.

κώδικας testbench αρχείου:

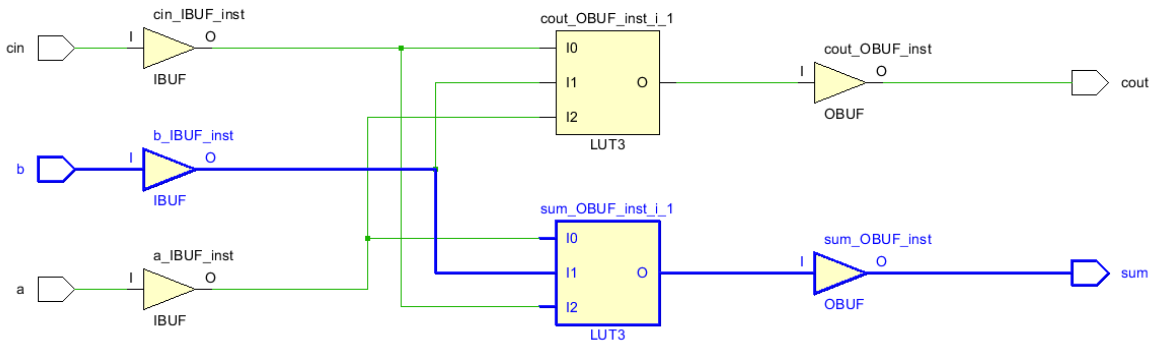
```
--------------------------------------------------------------------------------
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;

entity Pipelined_Full_Adder_tb is
end;

architecture bench of Pipelined_Full_Adder_tb is

  component Pipelined_Full_Adder
  Port ( cin : in STD_LOGIC;
  sum : out STD_LOGIC;
  a : in STD_LOGIC;
  b : in STD_LOGIC;
  cout : out STD_LOGIC);
  end component;

  signal cin: STD_LOGIC;
  signal sum: STD_LOGIC;
  signal a: STD_LOGIC;
  signal b: STD_LOGIC;
  signal cout: STD_LOGIC;

begin

  uut: Pipelined_Full_Adder port map ( cin  => cin,
                       sum  => sum,
                       a    => a,
                       b    => b,
                       cout => cout );

  stimulus: process
  begin

   A <= '0';
  B <= '0';
  Cin <= '0';
  wait for 10 ns;

  A <= '0';
```

```
  B <= '0';
  Cin <= '1';
  wait for 10 ns;

  A <= '0';
  B <= '1';
  Cin <= '0';
  wait for 10 ns;

  A <= '0';
  B <= '1';
  Cin <= '1';
  wait for 10 ns;

  A <= '1';
  B <= '0';
  Cin <= '0';
  wait for 10 ns;

  A <= '1';
  B <= '0';
  Cin <= '1';
  wait for 10 ns;

  A <= '1';
  B <= '1';
  Cin <= '0';
  wait for 10 ns;

  A <= '1';
  B <= '1';
  Cin <= '1';
  wait for 10 ns;

    wait;
  end process;


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

Από τον παραπάνω κώδικα προκύπτει το ακόλουθο simulation το οποίο συμφωνεί με τον πίνακα αλήθειας ενός FullAdder:



Εύρεση του κρίσιμου μονοπατιού (critical path), καθώς και της χρονικής του καθυστέρησης.



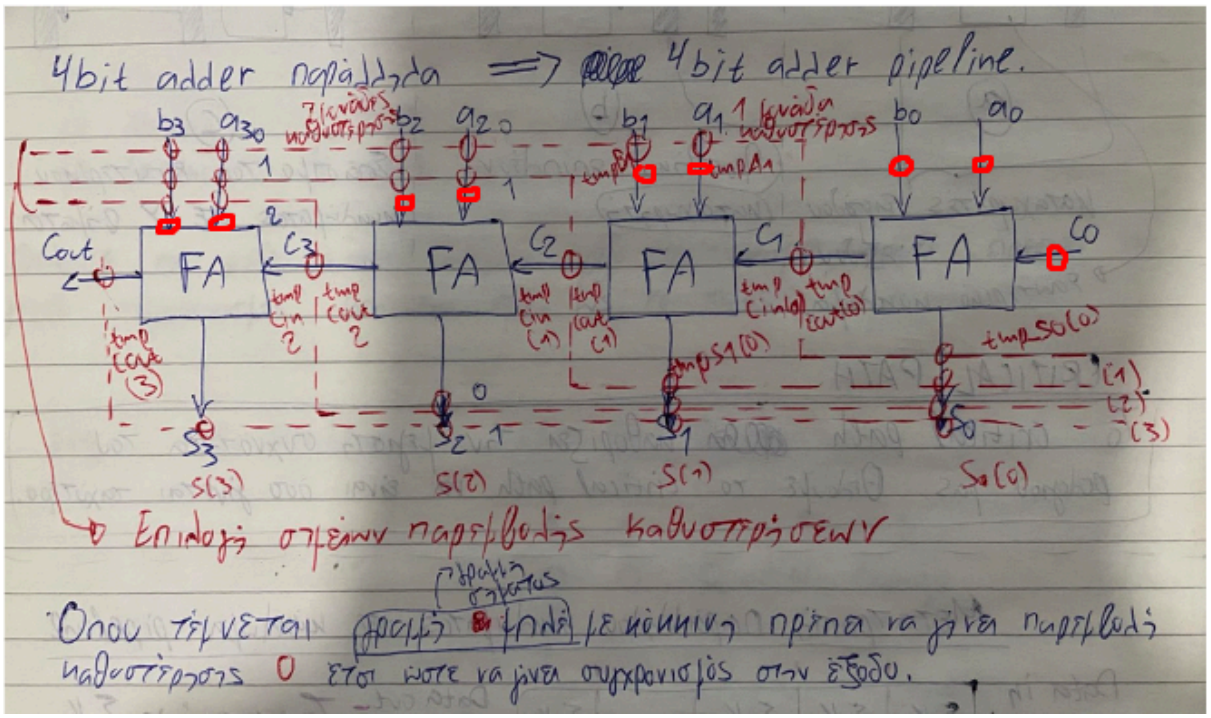
Unconstrained Paths - NONE - NONE - Setup

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

Το κρίσιμο μονοπάτι είναι η διαδρομή b→sum με καθυστέρηση 5.377.

2) Υλοποιήστε έναν σύγχρονο Αθροιστή διάδοσης κρατουμένου των 4 bits με χρήση της τεχνικής Pipeline.

Από την εκφώνηση ζητείται σε κάθε κύκλο ρολογιού να τροφοδοτείται και ένα διαφορετικό ζεύγος εισόδων και να δίνει ορθό αποτέλεσμα σε κάθε κύκλο ρολογιού έπειτα από κάποια αρχική καθυστέρηση Tlatency.
Συνεπώς επιλέγουμε να κάνουμε 4stage pipeline όπου τοποθετούμε καταχωρητές για να σπάσει το κύκλωμα σε παραπάνω στάδια. Χρησιμοποιούμε την μέθοδο CUT-SET.



κώδικας VHDL για την υλοποίηση του 4bit-adder με pipeline

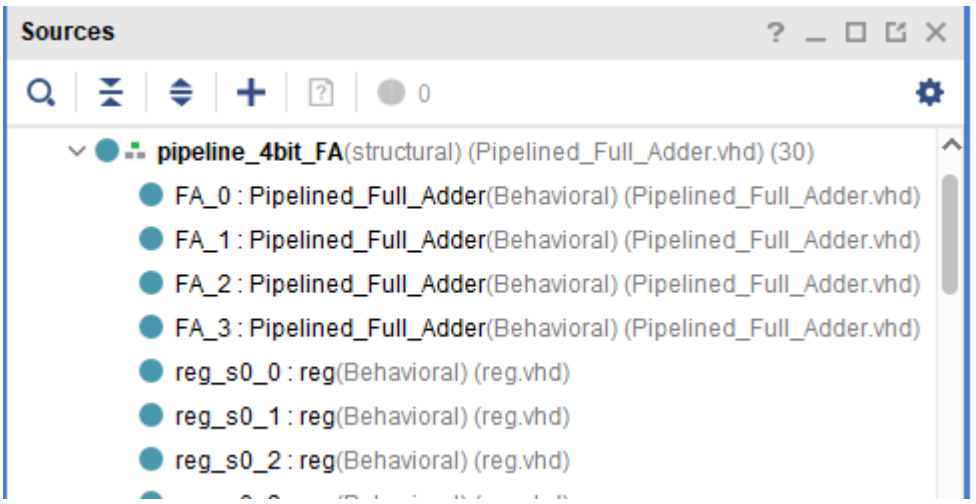
```
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity pipeline_4bit_FA is
    port(
        a, b : in std_logic_vector(3 downto 0);
        cin, clk, rst : in std_logic;
        sum : out std_logic_vector(3 downto 0);
        cout : out std_logic
    );
end pipeline_4bit_FA;
```

```
architecture Structural of pipeline_4bit_FA is
    component reg
        port (
            D : in STD_LOGIC;
            CLK : in STD_LOGIC;
            RSTn : in STD_LOGIC;
            Q : out STD_LOGIC
        );
    end component;

    component Pipelined_Full_Adder
        port (
            c : in STD_LOGIC;
            a : in STD_LOGIC;
            b : in STD_LOGIC;
            sum : out STD_LOGIC;
            carry_out : out STD_LOGIC
        );
    end component;

    signal tmp_a0, tmp_b0, tmp_s3 : std_logic;
    signal tmp_a1, tmp_b1, tmp_s2 : std_logic_vector(1 downto 0);
    signal tmp_a2, tmp_b2, tmp_s1 : std_logic_vector(2 downto 0);
    signal tmp_a3, tmp_b3, tmp_s0 : std_logic_vector(3 downto 0);
    signal tmp_cout : std_logic_vector(3 downto 0);
    signal tmp_cin : std_logic_vector(3 downto 0);

begin

    FA_0 : Pipelined_Full_Adder port map(a => tmp_a0, b => tmp_b0, c =>
tmp_cin(0), sum => tmp_s0(0), carry_out => tmp_cout(0));
    FA_1 : Pipelined_Full_Adder port map(a => tmp_a1(1), b => tmp_b1(1),
c => tmp_cin(1), sum => tmp_s1(0), carry_out => tmp_cout(1));
    FA_2 : Pipelined_Full_Adder port map(a => tmp_a2(2), b => tmp_b2(2),
c => tmp_cin(2), sum => tmp_s2(0), carry_out => tmp_cout(2));
    FA_3 : Pipelined_Full_Adder port map(a => tmp_a3(3), b => tmp_b3(3),
c => tmp_cin(3), sum => tmp_s3, carry_out => tmp_cout(3));

    -- 1 delay tmp_a0
    reg_a0 : reg port map(D => a(0), CLK => clk, Q => tmp_a0, RSTn =>
rst);
    -- 2 delay tmp_a1
    reg_a1_0 : reg port map(D => a(1), CLK => clk, Q => tmp_a1(0), RSTn
=> rst);
    reg_a1_1 : reg port map(D => tmp_a1(0), CLK => clk, Q => tmp_a1(1),
RSTn => rst);
```

```
    -- 3 delay tmp_a2
    reg_a2_0 : reg port map(D => a(2), CLK => clk, Q => tmp_a2(0), RSTn
=> rst);
    reg_a2_1 : reg port map(D => tmp_a2(0), CLK => clk, Q => tmp_a2(1),
RSTn => rst);
    reg_a2_2 : reg port map(D => tmp_a2(1), CLK => clk, Q => tmp_a2(2),
RSTn => rst);
    -- 4 delay tmp_a3
    reg_a3_0 : reg port map(D => a(3), CLK => clk, Q => tmp_a3(0), RSTn
=> rst);
    reg_a3_1 : reg port map(D => tmp_a3(0), CLK => clk, Q => tmp_a3(1),
RSTn => rst);
    reg_a3_2 : reg port map(D => tmp_a3(1), CLK => clk, Q => tmp_a3(2),
RSTn => rst);
    reg_a3_3 : reg port map(D => tmp_a3(2), CLK => clk, Q => tmp_a3(3),
RSTn => rst);

    -- 1 delay tmp_b0
    reg_b0 : reg port map(D => b(0), CLK => clk, Q => tmp_b0, RSTn =>
rst);
    -- 2 delay tmp_b1
    reg_b1_0 : reg port map(D => b(1), CLK => clk, Q => tmp_b1(0), RSTn
=> rst);
    reg_b1_1 : reg port map(D => tmp_b1(0), CLK => clk, Q => tmp_b1(1),
RSTn => rst);
    -- 3 delay tmp_b2
    reg_b2_0 : reg port map(D => b(2), CLK => clk, Q => tmp_b2(0), RSTn
=> rst);
    reg_b2_1 : reg port map(D => tmp_b2(0), CLK => clk, Q => tmp_b2(1),
RSTn => rst);
    reg_b2_2 : reg port map(D => tmp_b2(1), CLK => clk, Q => tmp_b2(2),
RSTn => rst);
    -- 4 delay tmp_b3
    reg_b3_0 : reg port map(D => b(3), CLK => clk, Q => tmp_b3(0), RSTn
=> rst);
    reg_b3_1 : reg port map(D => tmp_b3(0), CLK => clk, Q => tmp_b3(1),
RSTn => rst);
    reg_b3_2 : reg port map(D => tmp_b3(1), CLK => clk, Q => tmp_b3(2),
RSTn => rst);
    reg_b3_3 : reg port map(D => tmp_b3(2), CLK => clk, Q => tmp_b3(3),
RSTn => rst);

    --1 delay for tmp_s3
    reg_s3 : reg port map(D => tmp_s3, CLK => clk, Q => sum(3), RSTn =>
rst);
    -- 2 delays for tmp_s2
```

```
    reg_s2_0 : reg port map(D => tmp_s2(0), CLK => clk, Q => tmp_s2(1),
RSTn => rst);
    reg_s2_1 : reg port map(D => tmp_s2(1), CLK => clk, Q => sum(2),
RSTn => rst);
    -- 3 delays for tmp_s1
    reg_s1_0 : reg port map(D => tmp_s1(0), CLK => clk, Q => tmp_s1(1),
RSTn => rst);
    reg_s1_1 : reg port map(D => tmp_s1(1), CLK => clk, Q => tmp_s1(2),
RSTn => rst);
    reg_s1_2 : reg port map(D => tmp_s1(2), CLK => clk, Q => sum(1),
RSTn => rst);
    -- 4 delays for tmp_s0
    reg_s0_0 : reg port map(D => tmp_s0(0), CLK => clk, Q => tmp_s0(1),
RSTn => rst);
    reg_s0_1 : reg port map(D => tmp_s0(1), CLK => clk, Q => tmp_s0(2),
RSTn => rst);
    reg_s0_2 : reg port map(D => tmp_s0(2), CLK => clk, Q => tmp_s0(3),
RSTn => rst);
    reg_s0_3 : reg port map(D => tmp_s0(3), CLK => clk, Q => sum(0),
RSTn => rst);

    -- 1 delay for cin
    reg_c0 : reg port map(D => cin, CLK => clk, Q => tmp_cin(0), RSTn =>
rst);
    -- 1 delay for cout0
    reg_c1 : reg port map(D => tmp_cout(0), CLK => clk, Q => tmp_cin(1),
RSTn => rst);
    -- 1 delay for cout1
    reg_c2 : reg port map(D => tmp_cout(1), CLK => clk, Q => tmp_cin(2),
RSTn => rst);
    -- 1 delay for cout2
    reg_c3 : reg port map(D => tmp_cout(2), CLK => clk, Q => tmp_cin(3),
RSTn => rst);
    -- 1 delay for cout3
    reg_c4 : reg port map(D => tmp_cout(3), CLK => clk, Q => cout, RSTn
=> rst);


end Structural;
```

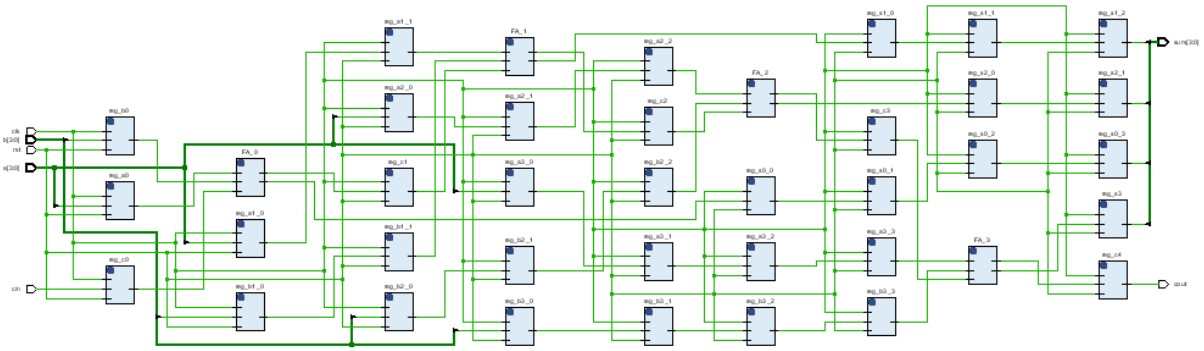
Για να βρεί ο κώδικας τα component `Pipelined_Full_Adder,reg` πρέπει να κάνουμε αρχικά include το αρχείο κώδικα της 1ης άσκησης σαν source code στο τωρινό μας project pipeline_4bit_FA.vhd. Επιπλέον πρέπει να δημιουργήσουμε άλλο ένα source code με όνομα reg.vhd για να δώσουμε τον κώδικα για τη δημιουργία ενός καταχωρητή.



κώδικας για τον register :

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

entity reg is
    Port ( D : in STD_LOGIC;
           CLK : in STD_LOGIC;
           RSTn : in STD_LOGIC;
           Q : out STD_LOGIC);
end reg;

architecture Behavioral of reg is
begin
process(CLK, RSTn)
begin

if RSTn = '1' then Q <= '0';
 elsif rising_edge(CLK)then Q <= D;
 end if;
 end process;

end Behavioral;
----------------------------------------------------------------------------
```

Παρουσίαση δομικού διαγράμματος (RTL schematic) του Πλήρη Αθροιστή.



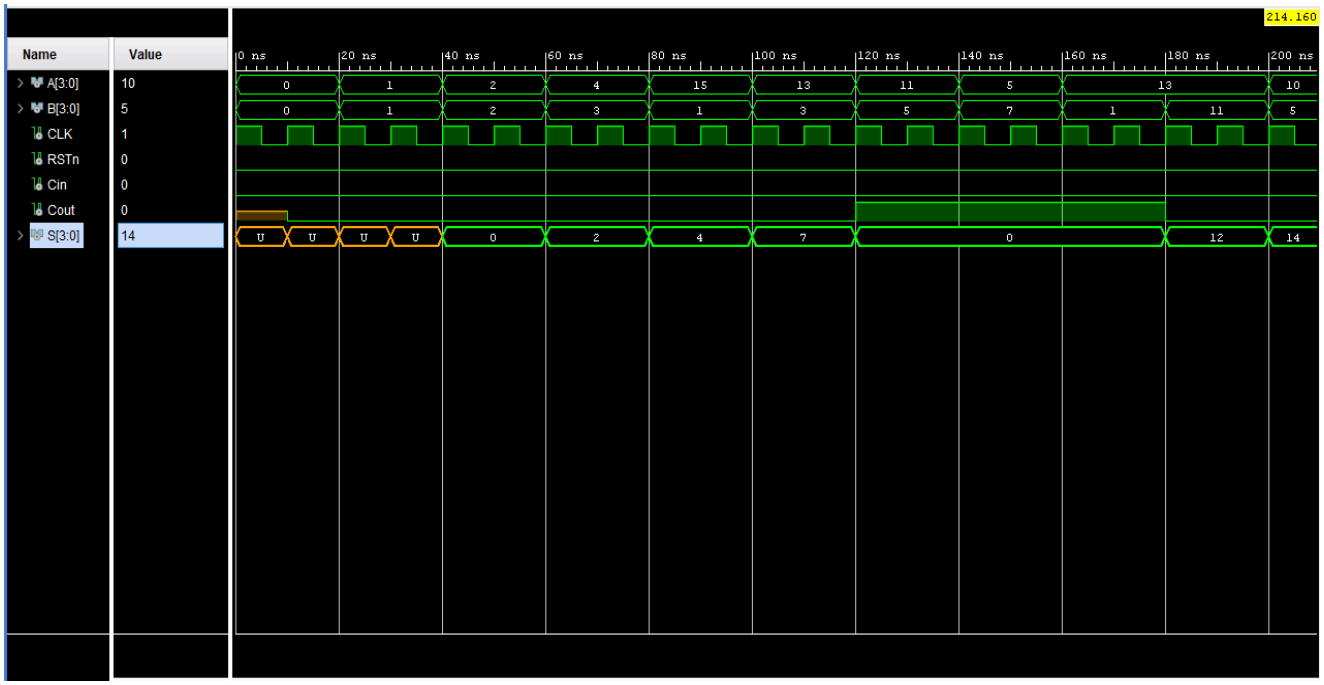
Δημιουργία testbench, με το οποίο γίνεται ο έλεγχος ορθής λειτουργίας του κυκλώματος.

κώδικας testbench αρχείου:

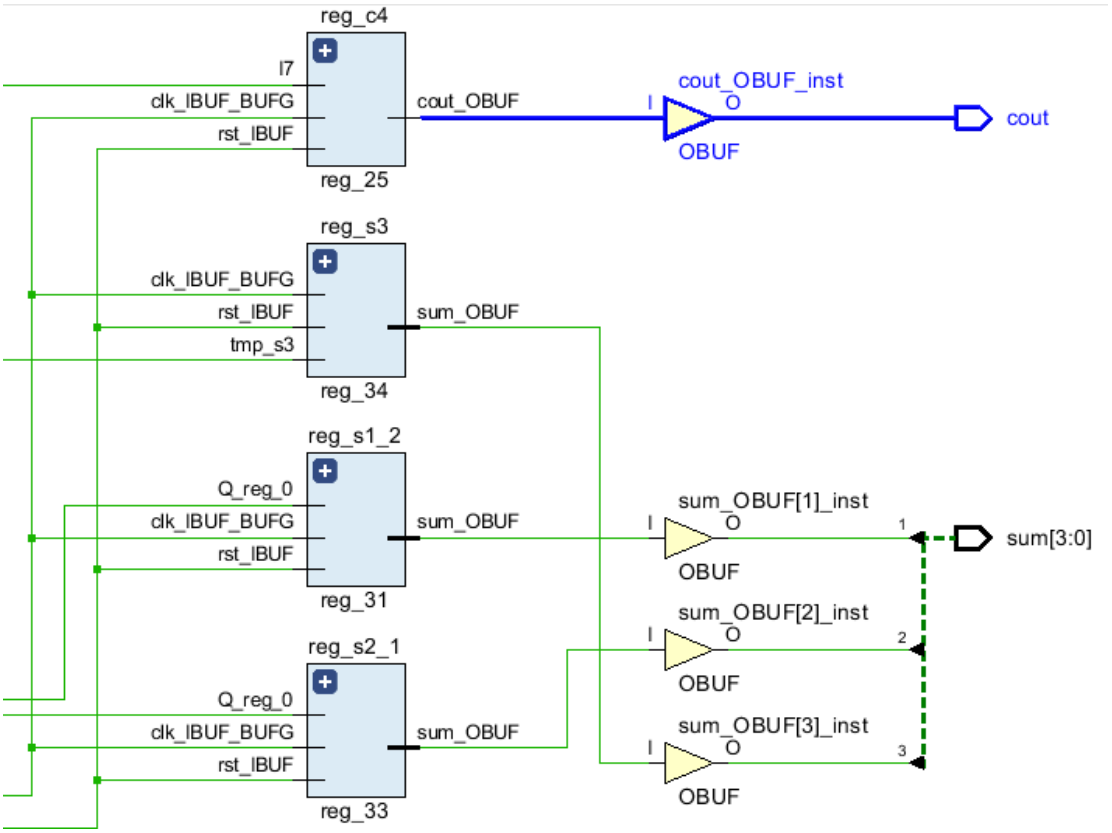
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity pipeline_4bit_FA_tb is
end pipeline_4bit_FA_tb;

architecture testbench of pipeline_4bit_FA_tb is

    -- Component declaration
    COMPONENT pipeline_4bit_FA is
        port(
a, b : in std_logic_vector(3 downto 0);
cin, clk, rst : in std_logic;
sum : out std_logic_vector(3 downto 0);
cout : out std_logic
);
    end COMPONENT;

    -- Signals for testbench
    signal A : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');
    signal B : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');
    signal CLK : STD_LOGIC := '0';
    signal RSTn : STD_LOGIC := '1';  -- Active low reset
    signal Cin : STD_LOGIC := '0';
    signal Cout : STD_LOGIC := '0';  -- Initialize Cout
    signal S : STD_LOGIC_VECTOR (3 downto 0) := (others => '0');  --
```

```
begin
    UUT : pipeline_4bit_FA
    port map (
        a => A,
        b => B,
        clk => CLK,
        rst => RSTn,
        cin => Cin,
        cout => Cout,
        sum => S
    );

    clk_process: process
    begin
        while now < 1000 ns loop
            CLK <= not CLK; -- Toggle clock
            wait for 5 ns; -- Clock period
        end loop;
        wait;
    end process;

    stimulus: process
    begin

        RSTn <= '0';

        A <= "0000"; B <= "0000"; Cin <= '0'; wait for 20 ns;
        A <= "0001"; B <= "0001"; Cin <= '0'; wait for 20 ns;
        A <= "0010"; B <= "0010"; Cin <= '0'; wait for 20 ns;
        A <= "0100"; B <= "0011"; Cin <= '0'; wait for 20 ns;
        A <= "1111"; B <= "0001"; Cin <= '0'; wait for 20 ns;
        A <= "1101"; B <= "0011"; Cin <= '0'; wait for 20 ns;
        A <= "1011"; B <= "0101"; Cin <= '0'; wait for 20 ns;
        A <= "0101"; B <= "0111"; Cin <= '0'; wait for 20 ns;
        A <= "1101"; B <= "0001"; Cin <= '0'; wait for 20 ns;
        A <= "1101"; B <= "1011"; Cin <= '0'; wait for 20 ns;
        A <= "1010"; B <= "0101"; Cin <= '0'; wait for 20 ns;
        A <= "0100"; B <= "1111"; Cin <= '0'; wait for 20 ns;

        wait;
    end process;

end testbench;
```

Από τον παραπάνω κώδικα προκύπτει το ακόλουθο simulation το οποίο συμφωνεί με τον πίνακα αλήθειας ενός 4bit-fulladder:



Εύρεση του κρίσιμου μονοπατιού (critical path), καθώς και της χρονικής του καθυστέρησης.



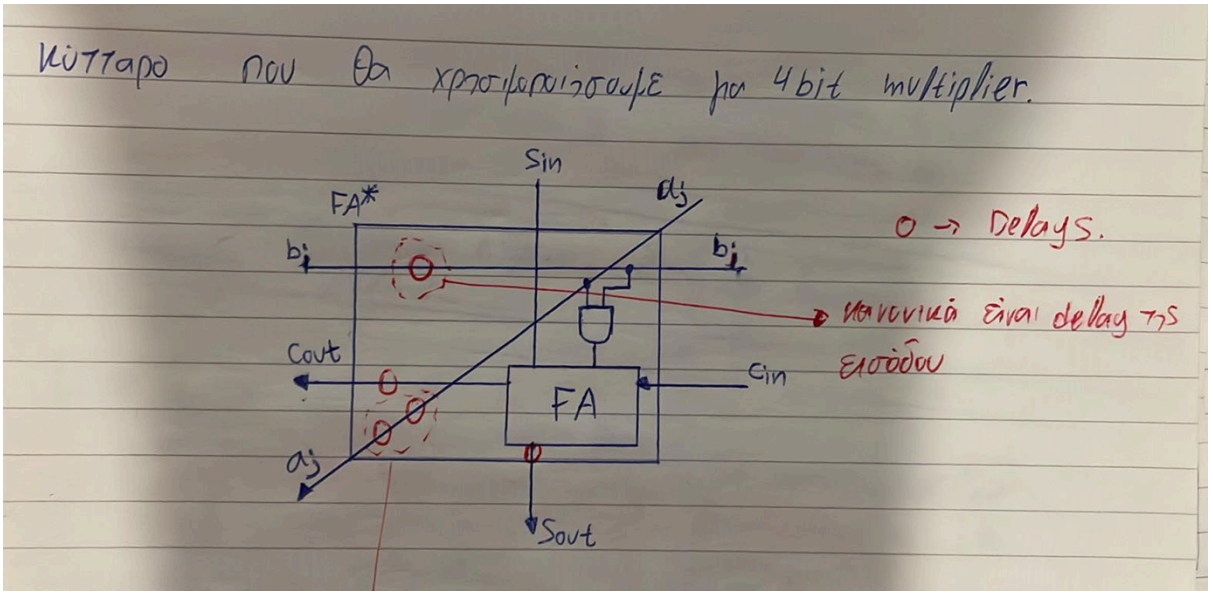
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	2	2	1	reg_c4/Q_reg/C	cout	4.076	3.276	0.800	∞	
Path 2	∞	2	2	1	reg_s0_3/Q_reg/C	sum[0]	4.076	3.276	0.800	∞	
Path 3	∞	2	2	1	reg_s1_2/Q_reg/C	sum[1]	4.076	3.276	0.800	∞	
Path 4	∞	2	2	1	reg_s2_1/Q_reg/C	sum[2]	4.076	3.276	0.800	∞	
Path 5	∞	2	2	1	reg_s3/Q_reg/C	sum[3]	4.076	3.276	0.800	∞	
Path 6	∞	2	2	2	reg_c0/Q_reg/C	reg_s0_1/Q_re...3_1_Q_reg_c/D	1.836	0.751	1.085	∞	

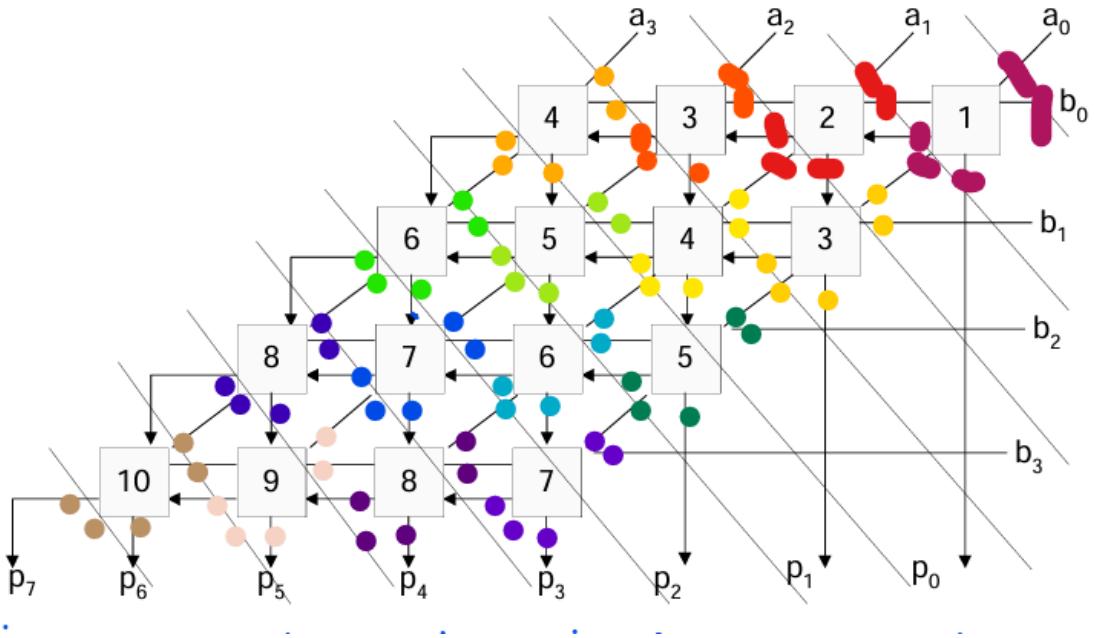
Το κρίσιμο μονοπάτι είναι οι διαδρομές Path1-Path5 όπως φαίνεται στο παραπάνω screenshot με total delay 4.076.

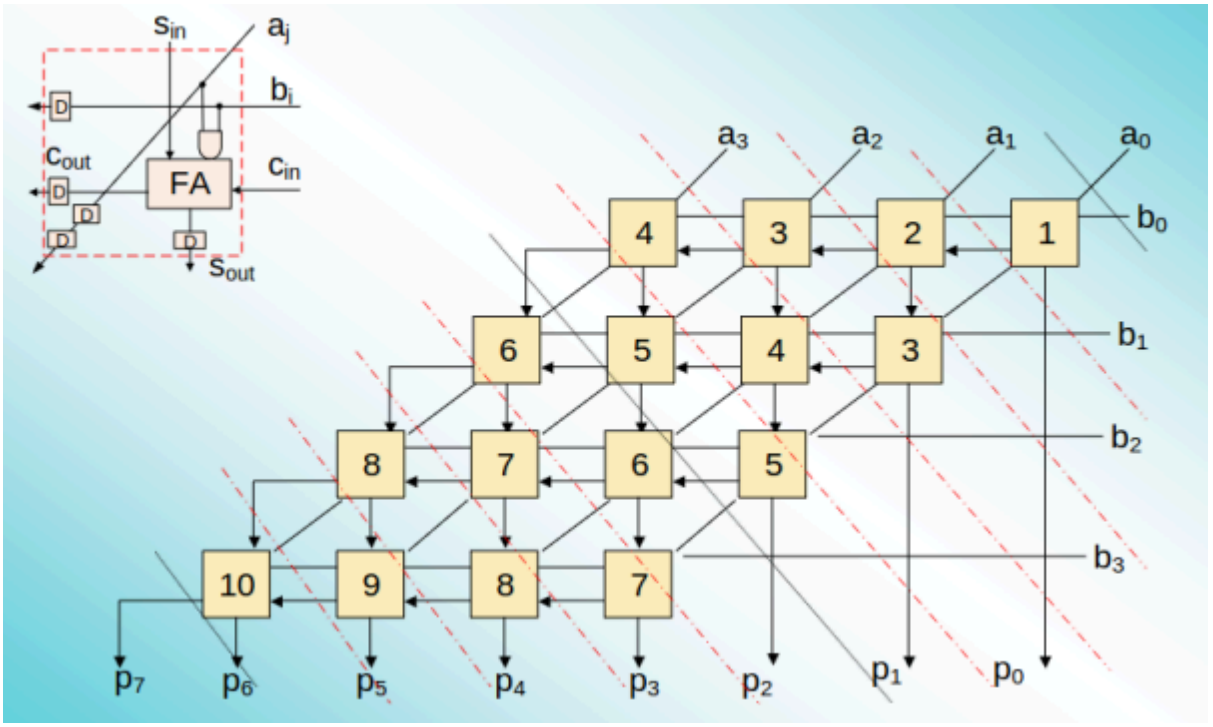
3) Υλοποιήστε ένα συστολικό (είδος pipeline) Πολλαπλασιαστή διάδοσης κρατουμένων των 4 bits κάνοντας χρήση σύγχρονων Πλήρων Αθροιστών (Full Adders).

Αρχικά κάναμε include στο project τα source code για το register (που χρησιμοποιήθηκε στη 2η άσκηση) και το fulladder (που χρησιμοποιήθηκε στη 1η άσκηση).

Υλοποιήσαμε το cell που φαίνεται στην παρακάτω φωτογραφία. Ο τελικός 4bit-multiplier αποτελείται απο 16 cell.







Πέρα από τα delays που χρησιμοποιήσαμε κατά την κατασκευή του cell, χρειάζονται επιπλέον :

Στις εισόδους :

α1 → 1 delay b1 → 2 delays , για να συγχρονιστεί με τον 3 κυκλο
α2 → 2 delays b2 → 4 delays , για να συγχρονιστεί με τον 5 κυκλο
α3 → 3 delays b3 → 6 delays , για να συγχρονιστεί με τον 7 κυκλο

Στις εξόδους :

P0 → 9 delays P3 → 3 delays
P1 → 7 delays P4 → 2 delays
P2 → 5 delays P5 → 1 delay

έτσι ώστε το αποτέλεσμα να εμφανίζεται συγχρονισμένα στην έξοδο.

Κώδικας VHDL για την υλοποίηση του `4bit_mult` με pipeline

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


entity pipeline_4bit_mult is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           P : out STD_LOGIC_VECTOR (7 downto 0);
          -- Cinall : in STD_LOGIC_VECTOR (3 downto 0);
           CLK : in STD_LOGIC;  -- Added CLK signal to port list
         RSTn : in STD_LOGIC  -- Added RSTn signal to port list
          --  Sinall : in STD_LOGIC_VECTOR (3 downto 0)
         );


end pipeline_4bit_mult;

 architecture structural of pipeline_4bit_mult is

signal tmp_B1 : std_logic_vector(1 downto 0);
signal tmp_B2 : std_logic_vector(3 downto 0);
signal tmp_B3 : std_logic_vector(5 downto 0);

signal tmp_P4 : STD_LOGIC;
signal tmp_P3 : std_logic_vector(1 downto 0);
signal tmp_P2 : std_logic_vector(3 downto 0);
signal tmp_P1 : std_logic_vector(5 downto 0);
signal tmp_P0 : std_logic_vector(7 downto 0);
signal cout_cell4 : std_logic_vector(3 downto 0);
signal cout_cell4_delay : std_logic_vector(3 downto 0);

signal A0_first_delay :STD_LOGIC;
signal B0_first_delay :STD_LOGIC;
```

```
signal A1_first_delay :STD_LOGIC;
signal A2_first_delay : std_logic_vector(1 downto 0);
signal A3_first_delay : std_logic_vector(2 downto 0);
signal cout_first_4_parallel : std_logic_vector(3 downto 0);  ---cout
ths ka8e tetradas gia thn prwth
signal cout_second_4_parallel : std_logic_vector(3 downto 0);   ---cout
ths ka8e tetradas gia thn deyterh
signal cout_third_4_parallel : std_logic_vector(3 downto 0);    ---cout
ths ka8e tetradas gia thn trith
signal cout_fourth_4_parallel : std_logic_vector(3 downto 0);   ---cout
ths ka8e tetradas gia thn tetarti

signal b0_output : std_logic_vector(3 downto 0);
signal b1_output : std_logic_vector(3 downto 0);
signal b2_output : std_logic_vector(3 downto 0);
signal b3_output : std_logic_vector(3 downto 0);

signal a0_output : std_logic_vector(3 downto 0);
signal a1_output : std_logic_vector(3 downto 0);
signal a2_output : std_logic_vector(3 downto 0);
signal a3_output : std_logic_vector(3 downto 0);

signal Sout_cell : std_logic_vector(15 downto 0);



component cell is
   Port ( Aj : in STD_LOGIC;
          Bi : in STD_LOGIC;
          Cin : in STD_LOGIC;
          Sin : in STD_LOGIC;
          Cout : out STD_LOGIC;
          Sout : out STD_LOGIC;
          CLK : in STD_LOGIC;
          Aj_out:out STD_LOGIC;
          Bi_out:out STD_LOGIC;
          RSTn : in STD_LOGIC);
end component;

  component reg is
    Port ( D : in STD_LOGIC;
           CLK : in STD_LOGIC;
           RSTn : in STD_LOGIC;
           Q : out STD_LOGIC);
  end component;
```

```
begin


  ----------------------Delay for B2
reg_B2_1_delay_shift: reg port
map(D=>B(2),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B2(0));
reg_B2_2_Delay_shift: reg port
map(D=>tmp_B2(0),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B2(1));
reg_B2_3_Delay_shift: reg port
map(D=>tmp_B2(1),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B2(2));
reg_B2_4_Delay_shift: reg port
map(D=>tmp_B2(2),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B2(3));
 ----------------------------DElay for B3
reg_B3_1_delay_shift: reg port
map(D=>B(3),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B3(0));
reg_B3_2_delay_shift: reg port
map(D=>tmp_B3(0),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B3(1));
reg_B3_3_delay_shift: reg port
map(D=>tmp_B3(1),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B3(2));
reg_B3_4_delay_shift: reg port
map(D=>tmp_B3(2),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B3(3));
reg_B3_5_delay_shift: reg port
map(D=>tmp_B3(3),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B3(4));
reg_B3_6_delay_shift: reg port
map(D=>tmp_B3(4),CLK=>CLK,RSTn=>RSTn,Q=>tmp_B3(5));


------------------------CELLS-----------------------------------------

  --------------------FIRST 4PARALLEL



 cell_1_first_4_parallel: cell port map(
           Aj=>A(0),
           Bi=>B(0),
           Cin=>'0',
           Sin=>'0',
           Cout=>cout_first_4_parallel(0),
           Sout=>Sout_cell(0),
           CLK=>CLK,
           Aj_out=>a0_output(0),
           Bi_out=>b0_output(0),
           RSTn=>RSTn);
```

```
 ---------------------------delay for A1
 reg_A1_1_first_delay: reg port map(D=>A(1)
,CLK=>CLK,RSTn=>RSTn,Q=>A1_first_delay);

  cell_2_first_4_parallel: cell port map(
           Aj=>A1_first_delay,
           Bi=>b0_output(0),
           Cin=>cout_first_4_parallel(0),
           Sin=>'0',
           Cout=>cout_first_4_parallel(1),
           Sout=>Sout_cell(1),
           CLK=>CLK,
           Aj_out=>a1_output(0),
           Bi_out=>b0_output(1),
           RSTn=>RSTn);


----------------------Delay for A2
reg_A2_1_first_delay: reg port map(D=>A(2)
,CLK=>CLK,RSTn=>RSTn,Q=>A2_first_delay(0));
reg_A2_2_first_delay: reg port map(D=>A2_first_delay(0)
,CLK=>CLK,RSTn=>RSTn,Q=>A2_first_delay(1));

  cell_3_first_4_parallel: cell port map(
           Aj=>A2_first_delay(1),
           Bi=>b0_output(1),
           Cin=>cout_first_4_parallel(1),
           Sin=>'0',
           Cout=>cout_first_4_parallel(2),
           Sout=>Sout_cell(2),
           CLK=>CLK,
           Aj_out=>a2_output(0),
           Bi_out=>b0_output(2),
           RSTn=>RSTn);


 ----------------------Delay for A3
reg_A3_1_first_delay: reg port map(D=>A(3)
,CLK=>CLK,RSTn=>RSTn,Q=>A3_first_delay(0));
reg_A3_2_first_delay: reg port map(D=>A3_first_delay(0)
,CLK=>CLK,RSTn=>RSTn,Q=>A3_first_delay(1));
reg_A3_3_first_delay: reg port map(D=>A3_first_delay(1)
,CLK=>CLK,RSTn=>RSTn,Q=>A3_first_delay(2));

  cell_4_first_4_parallel: cell port map(
```

```
            Aj=>A3_first_delay(2),
            Bi=>b0_output(2),
            Cin=>cout_first_4_parallel(2),
            Sin=>'0',
            Cout=>cout_first_4_parallel(3),
            Sout=>Sout_cell(3),
            CLK=>CLK,
            Aj_out=>a3_output(0),
            Bi_out=>b0_output(3),
            RSTn=>RSTn);


  --------------------SECOND 4PARALLEL

------------DElay for cout_cell4_delay_first
reg_cout_cell4_delay_first: reg port map(D=>cout_first_4_parallel(3)
,CLK=>CLK,RSTn=>RSTn,Q=>cout_cell4_delay(0));

   -----------------Delay for B1
reg_B1_1delay_shift: reg port map(D=>B(1)
,CLK=>CLK,RSTn=>RSTn,Q=>tmp_B1(0));
reg_B1_2_delay_shift: reg port map(D=>tmp_B1(0)
,CLK=>CLK,RSTn=>RSTn,Q=>tmp_B1(1));

   cell_1_second_4_parallel: cell port map(
            Aj=>a0_output(0),
            Bi=>tmp_B1(1) ,
            Cin=>'0',
            Sin=>Sout_cell(1),
            Cout=>cout_second_4_parallel(0),
            Sout=>Sout_cell(4),
            CLK=>CLK,
            Aj_out=>a0_output(1),
            Bi_out=>b1_output(0),
            RSTn=>RSTn);



     cell_2_second_4_parallel: cell port map(
            Aj=>a1_output(0),
            Bi=>b1_output(0) ,
            Cin=>cout_second_4_parallel(0),
            Sin=>Sout_cell(2),
            Cout=>cout_second_4_parallel(1),
            Sout=>Sout_cell(5),
            CLK=>CLK,
```

```
            Aj_out=>a1_output(1),
            Bi_out=>b1_output(1),
            RSTn=>RSTn);

      cell_3_second_4_parallel: cell port map(
            Aj=>a2_output(0),
            Bi=>b1_output(1),
            Cin=>cout_second_4_parallel(1),
            Sin=>Sout_cell(3),
            Cout=>cout_second_4_parallel(2),
            Sout=>Sout_cell(6),
            CLK=>CLK,
            Aj_out=>a2_output(1),
            Bi_out=>b1_output(2),
            RSTn=>RSTn);


        cell_4_second_4_parallel: cell port map(
            Aj=>a3_output(0),
            Bi=>b1_output(2),
            Cin=>cout_second_4_parallel(2),
            Sin=>cout_cell4_delay(0),
            Cout=>cout_second_4_parallel(3),
            Sout=>Sout_cell(7),
            CLK=>CLK,
            Aj_out=>a3_output(1),
            Bi_out=>b1_output(3),
            RSTn=>RSTn);


  --------------------THIRD 4PARALLEL

-----------------------DElay for cout_cell4_delay_second
reg_cout_cell4_delay_second: reg port map(D=>cout_second_4_parallel(3)
,CLK=>CLK,RSTn=>RSTn,Q=>cout_cell4_delay(1));


    cell_1_third_4_parallel: cell port map(
            Aj=>a0_output(1),
            Bi=>tmp_B2(3) ,
            Cin=>'0',
            Sin=>Sout_cell(5),
            Cout=>cout_third_4_parallel(0),
            Sout=>Sout_cell(8),
            CLK=>CLK,
            Aj_out=>a0_output(2),
```

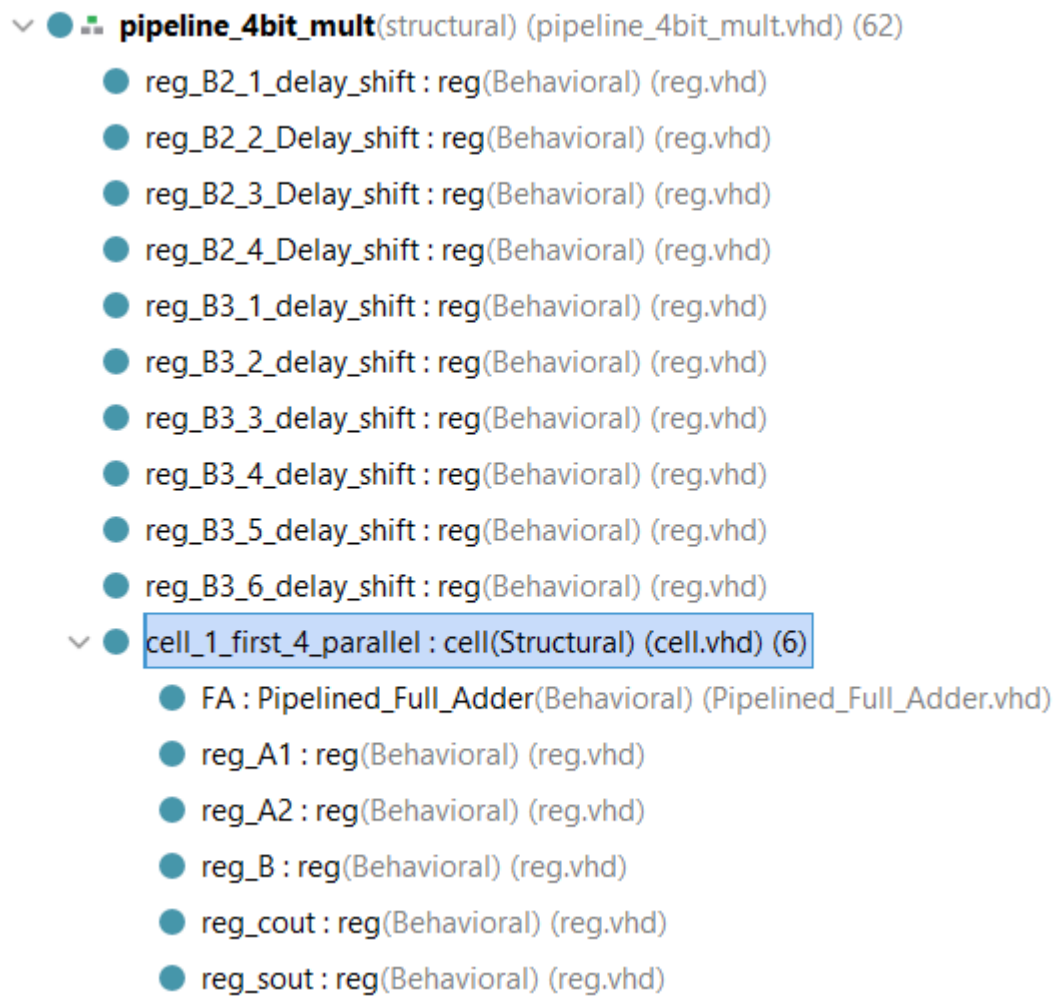
```
            Bi_out=>b2_output(0),
            RSTn=>RSTn);

     cell_2_third_4_parallel: cell port map(
            Aj=>a1_output(1),
            Bi=>b2_output(0) ,
            Cin=>cout_third_4_parallel(0),
            Sin=>Sout_cell(6),
            Cout=>cout_third_4_parallel(1),
            Sout=>Sout_cell(9),
            CLK=>CLK,
            Aj_out=>a1_output(2),
            Bi_out=>b2_output(1),
            RSTn=>RSTn);

      cell_3_third_4_parallel: cell port map(
            Aj=>a2_output(1),
            Bi=>b2_output(1),
            Cin=>cout_third_4_parallel(1),
            Sin=>Sout_cell(7),
            Cout=>cout_third_4_parallel(2),
            Sout=>Sout_cell(10),
            CLK=>CLK,
            Aj_out=>a2_output(2),
            Bi_out=>b2_output(2),
            RSTn=>RSTn);

        cell_4_third_4_parallel: cell port map(
            Aj=>a3_output(1),
            Bi=>b2_output(2),
            Cin=>cout_third_4_parallel(2),
            Sin=>cout_cell4_delay(1),
            Cout=>cout_third_4_parallel(3),
            Sout=>Sout_cell(11),
            CLK=>CLK,
            Aj_out=>a3_output(2),
            Bi_out=>b2_output(3),
            RSTn=>RSTn);


  --------------------FOURTH 4PARALLEL

----------------------Delay for cout_cell4_delay_third
 reg_cout_cell4_delay_third: reg port map(D=>cout_third_4_parallel(3)
,CLK=>CLK,RSTn=>RSTn,Q=>cout_cell4_delay(2));
```

```
    cell_1_fourth_4_parallel: cell port map(
           Aj=>a0_output(2),
           Bi=>tmp_B3(5) ,
           Cin=>'0',
           Sin=>Sout_cell(9),
           Cout=>cout_fourth_4_parallel(0),
           Sout=>Sout_cell(12),
           CLK=>CLK,
           Aj_out=>a0_output(3),
           Bi_out=>b3_output(0),
           RSTn=>RSTn);

     cell_2_fourth_4_parallel: cell port map(
           Aj=>a1_output(2),
           Bi=>b3_output(0),
           Cin=>cout_fourth_4_parallel(0),
           Sin=>Sout_cell(10),
           Cout=>cout_fourth_4_parallel(1),
           Sout=>Sout_cell(13),
           CLK=>CLK,
           Aj_out=>a1_output(3),
           Bi_out=>b3_output(1),
           RSTn=>RSTn);

      cell_3_fourth_4_parallel: cell port map(
           Aj=>a2_output(2),
           Bi=>b3_output(1),
           Cin=>cout_fourth_4_parallel(1),
           Sin=>Sout_cell(11),
           Cout=>cout_fourth_4_parallel(2),
           Sout=>Sout_cell(14),
           CLK=>CLK,
           Aj_out=>a2_output(3),
           Bi_out=>b3_output(2),
           RSTn=>RSTn);

        cell_4_fourth_4_parallel: cell port map(
           Aj=>a3_output(2),
           Bi=>b3_output(2),
           Cin=>cout_fourth_4_parallel(2),
           Sin=>cout_cell4_delay(2),
           Cout=>cout_fourth_4_parallel(3),
           Sout=>Sout_cell(15),
           CLK=>CLK,
           Aj_out=>a3_output(3),
           Bi_out=>b3_output(3),
```

```
            RSTn=>RSTn);



----9 DELAY FOR Sout from first cell (P(0))

  reg_P01_Delay: reg port
map(D=>Sout_cell(0),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P0(0));
  reg_P02_Delay: reg port
map(D=>tmp_P0(0),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P0(1));
  reg_P03_Delay: reg port
map(D=>tmp_P0(1),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P0(2));
  reg_P04_Delay: reg port
map(D=>tmp_P0(2),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P0(3));
  reg_P05_Delay: reg port
map(D=>tmp_P0(3),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P0(4));
  reg_P06_Delay: reg port
map(D=>tmp_P0(4),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P0(5));
  reg_P07_Delay: reg port
map(D=>tmp_P0(5),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P0(6));
  reg_P08_Delay: reg port map(D=>tmp_P0(6),CLK=>CLK,RSTn=>RSTn,Q=>P(0));
 -- reg_P09_Delay: reg port
map(D=>tmp_P0(7),CLK=>CLK,RSTn=>RSTn,Q=>P(0));


----7 DELAY FOR Sout from first cell FROM second cell4 parallel (P(1))

reg_P11_Delay: reg port
map(D=>Sout_cell(4),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P1(0));
reg_P12_Delay: reg port
map(D=>tmp_P1(0),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P1(1));
reg_P13_Delay: reg port
map(D=>tmp_P1(1),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P1(2));
reg_P14_Delay: reg port
map(D=>tmp_P1(2),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P1(3));
reg_P15_Delay: reg port
map(D=>tmp_P1(3),CLK=>CLK,RSTn=>RSTn,Q=>tmp_P1(4));
reg_P16_Delay: reg port map(D=>tmp_P1(4),CLK=>CLK,RSTn=>RSTn,Q=>P(1));
--reg_P17_Delay: reg port map(D=>tmp_P1(5),CLK=>CLK,RSTn=>RSTn,Q=>P(1));


  ----5 DELAY FOR Sout from first cell FROM third cell4 parallel (P(2))

reg_P21_Delay: reg port map(D=>Sout_cell(8)
,CLK=>CLK,RSTn=>RSTn,Q=>tmp_P2(0));
reg_P22_Delay: reg port map(D=>tmp_P2(0)
```

```
,CLK=>CLK,RSTn=>RSTn,Q=>tmp_P2(1));
reg_P23_Delay: reg port map(D=>tmp_P2(1)
,CLK=>CLK,RSTn=>RSTn,Q=>tmp_P2(2));
reg_P24_Delay: reg port map(D=>tmp_P2(2)
,CLK=>CLK,RSTn=>RSTn,Q=>tmp_P2(3));
reg_P25_Delay: reg port map(D=>tmp_P2(3) ,CLK=>CLK,RSTn=>RSTn,Q=>P(2));


----3 DELAY FOR Sout from first cell FROM -four cell4 parallel (P(3))

reg_P31_Delay: reg port map(D=>Sout_cell(12)
,CLK=>CLK,RSTn=>RSTn,Q=>tmp_P3(0));
reg_P32_Delay: reg port map(D=>tmp_P3(0)
,CLK=>CLK,RSTn=>RSTn,Q=>tmp_P3(1));
reg_P33_Delay: reg port map(D=>tmp_P3(1) ,CLK=>CLK,RSTn=>RSTn,Q=>P(3));


 ----2 DELAY FOR Sout from second cell FROM -four cell4 parallel (P(4))

reg_P41_Delay: reg port map(D=>Sout_cell(13)
,CLK=>CLK,RSTn=>RSTn,Q=>tmp_P4);
reg_P42_Delay: reg port map(D=>tmp_P4 ,CLK=>CLK,RSTn=>RSTn,Q=>P(4));


----1 DELAY FOR Sout from third cell FROM -four cell4 parallel (P(5))

reg_P5_Delay: reg port map(D=>Sout_cell(14)
,CLK=>CLK,RSTn=>RSTn,Q=>P(5));


-- 0 delay for P6, P7
P(6)<=Sout_cell(15)  ;
P(7)<= cout_fourth_4_parallel(3) ;



end structural;
```

Για να βρεί ο κώδικας τα component `cell,reg` πρέπει να κάνουμε αρχικά include το αρχείο κώδικα της 2ης άσκησης σαν source code οπου δημιουργήσαμε το reg.vhd στο τωρινό μας project pipeline_4bit_mult.vhd. Επιπλέον πρέπει να δημιουργήσουμε άλλο ένα source code με όνομα cell.vhd (τον οποίιο παραθέτουμε παρακάτω)

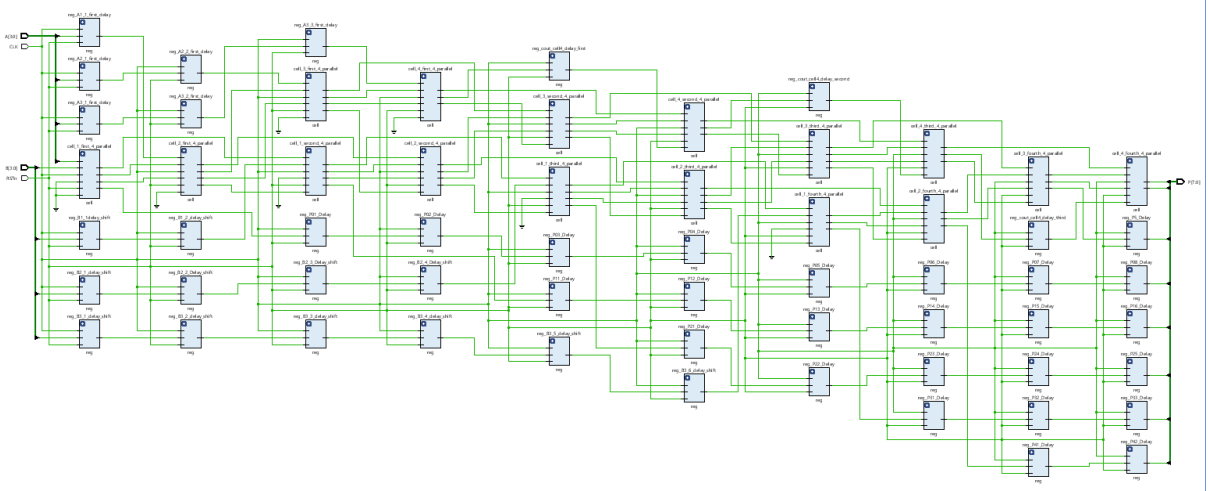


κώδικας για το cell:

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

entity cell is
    Port ( Aj : in STD_LOGIC;
           Bi : in STD_LOGIC;
           Cin : in STD_LOGIC;
           Sin : in STD_LOGIC;
           Cout : out STD_LOGIC;
           Sout : out STD_LOGIC;
           CLK : in STD_LOGIC;
           Aj_out:out STD_LOGIC;
           Bi_out:out STD_LOGIC;
           RSTn : in STD_LOGIC);
```

```
end cell;

architecture Structural of  cell is

    signal a_tmp, cout_tmp, sout_tmp : STD_LOGIC;
    signal and_out : STD_LOGIC;

  component reg is
  Port ( D : in STD_LOGIC;
           CLK : in STD_LOGIC;
           RSTn : in STD_LOGIC;
           Q : out STD_LOGIC);
end component;

component Pipelined_Full_Adder is
    Port (
        c : in STD_LOGIC;
        a : in STD_LOGIC;
        b : in STD_LOGIC;
        sum : out STD_LOGIC;
        carry_out : out STD_LOGIC
    );
end component;


begin
and_out<=Aj and Bi;
FA: Pipelined_Full_Adder port
map(c=>Cin,a=>Sin,b=>and_out,sum=>sout_tmp,carry_out=>cout_tmp);

reg_A1: reg port map(D=>Aj,CLK=>CLK,RSTn=>RSTn,Q=>a_tmp);
reg_A2: reg port map(D=>a_tmp,CLK=>CLK,RSTn=>RSTn,Q=>Aj_out);

reg_B: reg port map(D=>Bi,CLK=>CLK,RSTn=>RSTn,Q=>Bi_out);

reg_cout: reg port map(D=>cout_tmp,CLK=>CLK,RSTn=>RSTn,Q=>Cout);
reg_sout: reg port map(D=>sout_tmp,CLK=>CLK,RSTn=>RSTn,Q=>Sout);


end Structural;
```

Παρουσίαση δομικού διαγράμματος (RTL schematic) του) Πολλαπλασιαστή διάδοσης κρατουμένων των 4 bits.



Δημιουργία testbench, με το οποίο γίνεται ο έλεγχος ορθής λειτουργίας του κυκλώματος.

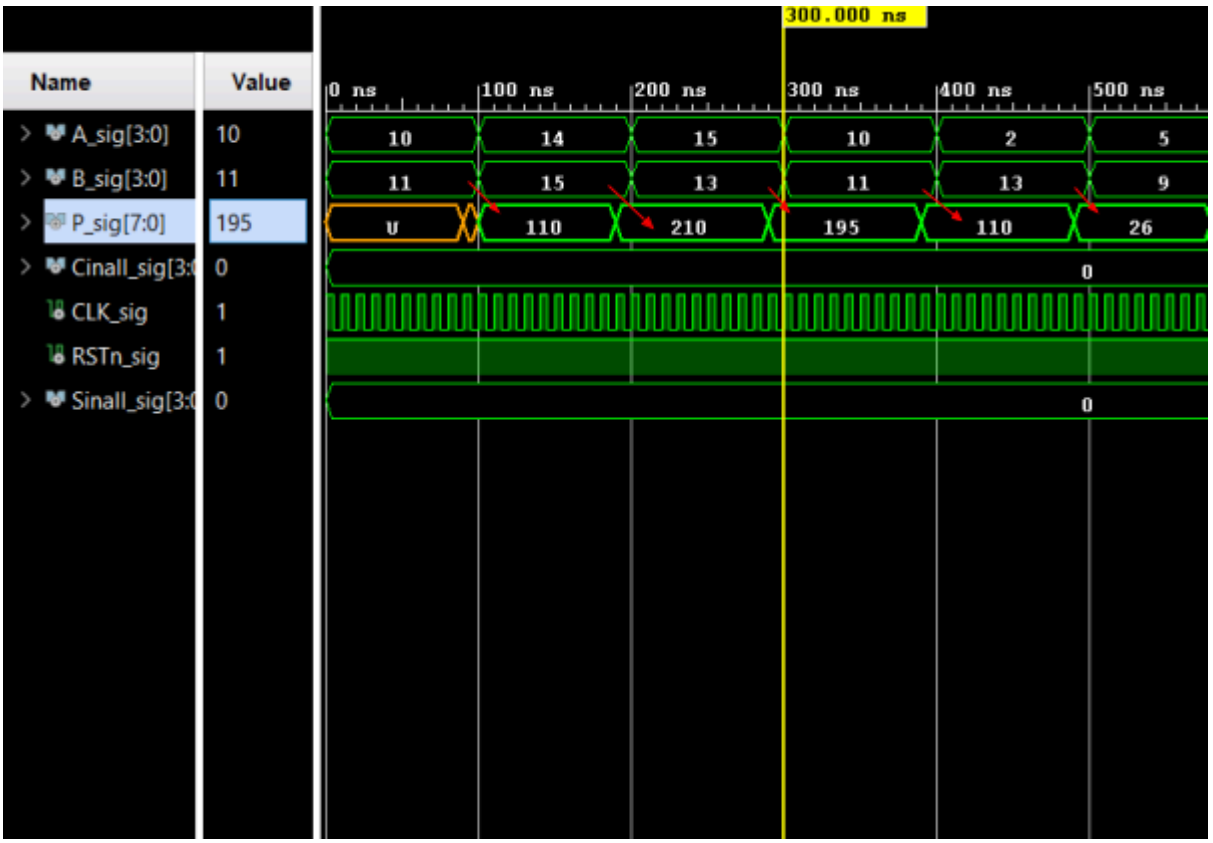
κώδικας testbench αρχείου:

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

entity pipeline_4bit_mult_tb is
end pipeline_4bit_mult_tb;

architecture Behavioral of pipeline_4bit_mult_tb is
    -- Component declaration
    component pipeline_4bit_mult
        Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
               B : in STD_LOGIC_VECTOR (3 downto 0);
               P : out STD_LOGIC_VECTOR (7 downto 0);
               --Cinall : in STD_LOGIC_VECTOR (3 downto 0);
               CLK : in STD_LOGIC;
               RSTn : in STD_LOGIC
              -- Sinall : in STD_LOGIC_VECTOR (3 downto 0)
              );

    end component;

    -- Signals declaration
    signal A_sig : STD_LOGIC_VECTOR (3 downto 0);
    signal B_sig : STD_LOGIC_VECTOR (3 downto 0);
    signal P_sig : STD_LOGIC_VECTOR (7 downto 0);
```

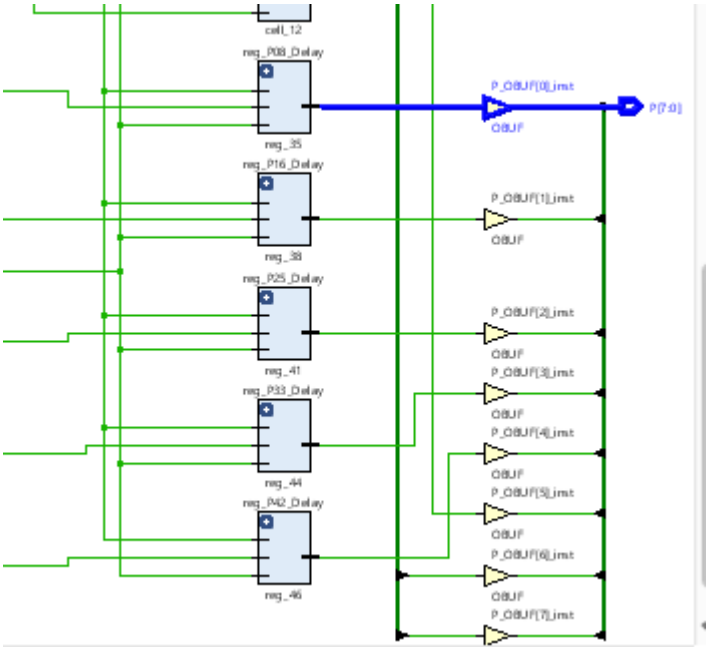
```
    signal Cinall_sig : STD_LOGIC_VECTOR (3 downto 0);
    signal CLK_sig : STD_LOGIC := '0'; -- Clock signal
    signal RSTn_sig : STD_LOGIC := '1'; -- Reset signal
    signal Sinall_sig : STD_LOGIC_VECTOR (3 downto 0);

begin
    -- Instantiate the pipeline_4bit_mult module
    UUT : pipeline_4bit_mult port map(
        A => A_sig,
        B => B_sig,
        P => P_sig,
      --  Cinall => Cinall_sig,
        CLK => CLK_sig,
        RSTn => RSTn_sig
       -- Sinall => Sinall_sig
    );

    -- Clock process
    CLK_process: process
    begin
        while now < 1000 ns loop
            CLK_sig <= not CLK_sig; -- Toggle clock
            wait for 5 ns; -- Clock period
        end loop;
        wait;
    end process;

    -- Stimulus process
    stimulus_process: process
    begin
        -- Reset initialization

        RSTn_sig <= '1';

     -- Initial values
A_sig <= "1010"; B_sig <= "1011"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

-- Next values
A_sig <= "1110"; B_sig <= "1111"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

-- Next values
A_sig <= "1111"; B_sig <= "1101"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;
```

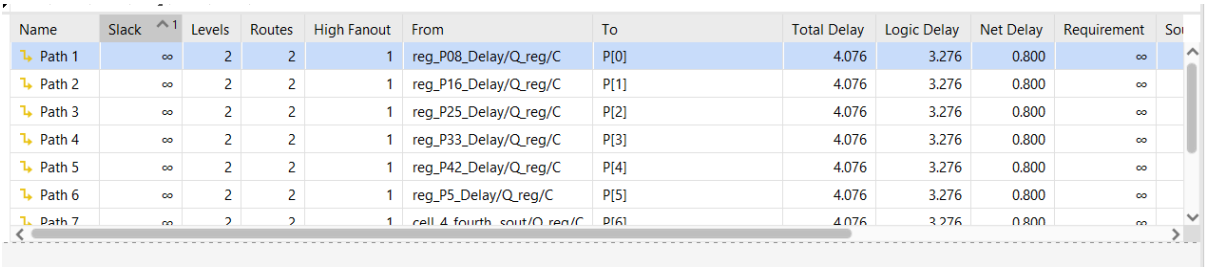
```
-- Next values
A_sig <= "1010"; B_sig <= "1011"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

-- Additional values
A_sig <= "0010"; B_sig <= "1101"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

A_sig <= "0101"; B_sig <= "1001"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

A_sig <= "0001"; B_sig <= "1110"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

A_sig <= "0110"; B_sig <= "0101"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

-- Next values
A_sig <= "1010"; B_sig <= "1011"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

-- Next values
A_sig <= "1110"; B_sig <= "1111"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

-- Next values
A_sig <= "1111"; B_sig <= "1101"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

-- Next values
A_sig <= "1010"; B_sig <= "1011"; Cinall_sig <= "0000"; Sinall_sig <=
"0000"; wait for 100 ns;

wait;
    end process;

end Behavioral;
```

Από τον παραπάνω κώδικα προκύπτει το ακόλουθο simulation το οποίο συμφωνεί με τον πίνακα αλήθειας ενός Πολλαπλασιαστή διάδοσης κρατουμένων των 4 bits.



Εύρεση του κρίσιμου μονοπατιού (critical path), καθώς και της χρονικής του καθυστέρησης.





Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	So
Path 1	∞	2	2	1	reg_P08_Delay/Q_reg/C	P[0]	4.076	3.276	0.800	∞	
Path 2	∞	2	2	1	reg_P16_Delay/Q_reg/C	P[1]	4.076	3.276	0.800	∞	
Path 3	∞	2	2	1	reg_P25_Delay/Q_reg/C	P[2]	4.076	3.276	0.800	∞	
Path 4	∞	2	2	1	reg_P33_Delay/Q_reg/C	P[3]	4.076	3.276	0.800	∞	
Path 5	∞	2	2	1	reg_P42_Delay/Q_reg/C	P[4]	4.076	3.276	0.800	∞	
Path 6	∞	2	2	1	reg_P5_Delay/Q_reg/C	P[5]	4.076	3.276	0.800	∞	
Path 7	∞	2	2	1	cell_4_fourth_sout/Q_reg/C	P[6]	4.076	3.276	0.800	∞	

Το κρίσιμο μονοπάτι είναι οι διαδρομές Path1-Path7 όπως φαίνεται στο παραπάνω screenshot με total delay 4.076.