



Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Ψηφιακά Συστήματα VLSI

3η Εργαστηριακή Άσκηση

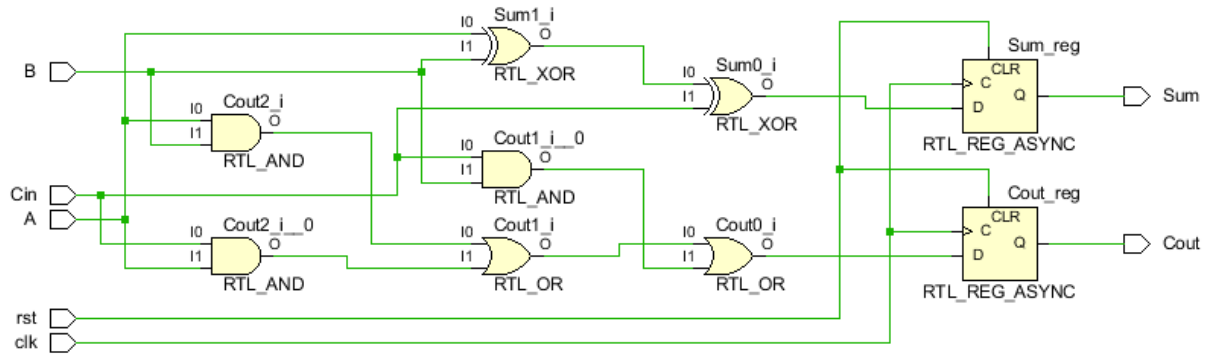
Σχεδίαση Μονάδων Υλικού με την Τεχνική Pipelining

Καραμπίνας Παναγιώτης (03116170)
Αλικάρης Αντώνιος Δημήτριος (03118062)

1. Σύγχρονος Πλήρης Αθροιστής

Κώδικας και δομικό διάγραμμα (RTL)

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity Synchronous_Full_Adder_Behavioral is
6      port(
7          clk : in std_logic;
8          rst : in std_logic;
9          A : in std_logic;
10         B : in std_logic;
11         Cin : in std_logic;
12         Sum : out std_logic;
13         Cout : out std_logic
14     );
15 end entity; -- Synchronous Full Adder
16
17 architecture behavioral_arch of Synchronous_Full_Adder_Behavioral is
18     --In the below implementation of the Synchronous adder we observe that only when
19     --↪ we have a logic switch
20     --of our clock from 0 -> 1, there happens a change in our FA variables.
21     --Another point of interest is that the use of variables other than Sum and Cout
22     --↪ inside the if(of the CLK event)
23     --triggers an unwanted behavior. So we understand that for every variable used
24     --↪ inside the if (CLK event) our compiler
25     --creates a register, therefore it is logical to use a minimal number of
26     --↪ variables inside the if statement.
27 begin
28     FA_LOGIC : process(rst,clk)
29     begin
30         if(rst='1') then
31             Sum <='0';
32             Cout<='0';
33         elsif (clk'event and clk='1') then
34             Sum <= A XOR B XOR Cin ;
35             Cout <= (A AND B) OR (Cin AND A) OR (Cin AND B) ;
36         end if;
37     end process;
38 end architecture ; -- arch
```



Κώδικας testbench και Κυματομορφή εξόδου

```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY Synchronous_Full_Adder_Behavioral_tb IS
5  END Synchronous_Full_Adder_Behavioral_tb;
6
7  ARCHITECTURE behavior OF Synchronous_Full_Adder_Behavioral_tb IS
8
9      -- Component Declaration for the Unit Under Test (UUT)
10
11     COMPONENT Synchronous_Full_Adder_Behavioral
12     PORT(
13     A : IN std_logic;
14     B : IN std_logic;
15     Cin : IN std_logic;
16     clk : in std_logic;
17     rst : in std_logic;
18     Sum : OUT std_logic;
19     Cout : OUT std_logic
20     );
21     END COMPONENT;
22
23     --constants
24     constant T : time := 10ns; --clock period
25     -- Inputs
26     signal A : std_logic := '0';
27     signal B : std_logic := '0';
28     signal Cin : std_logic := '0';
29     signal clk : std_logic := '0';
30     signal rst : std_logic := '0';
31
32     -- Outputs
33     signal Sum : std_logic;

```

```

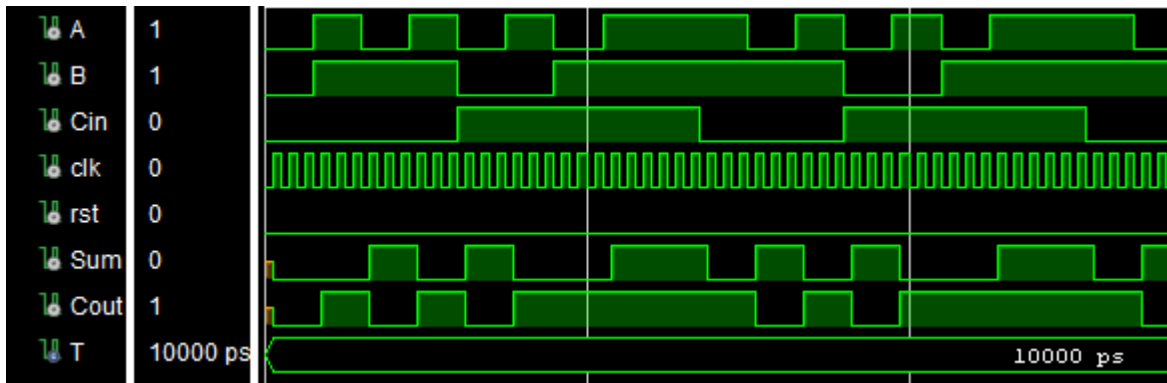
34  signal Cout : std_logic;
35
36  BEGIN
37
38      -- Instantiate the Unit Under Test (UUT)
39      uut: Synchronous_Full_Adder_Behavioral PORT MAP (
40          A => A,
41          B => B,
42          Cin => Cin,
43          clk => clk,
44          rst => rst,
45          Sum => Sum,
46          Cout => Cout
47      );
48
49      clk_gen: process begin
50          clk <= '0';
51          wait for T/2;
52          clk <= '1';
53          wait for T/2;
54      end process;
55
56      -- Stimulus process
57      stim_proc: process
58      begin
59          -- hold reset state for 100 ns.
60          wait for 30 ns;
61
62          -- insert stimulus here
63          A <= '1';
64          B <= '1';
65          Cin <= '0';
66          wait for 30 ns;
67
68          A <= '0';
69          B <= '1';
70          Cin <= '0';
71          wait for 30 ns;
72
73          A <= '1';
74          B <= '1';
75          Cin <= '0';
76          wait for 30 ns;
77
78          A <= '0';
79          B <= '0';
80          Cin <= '1';
81          wait for 30 ns;

```

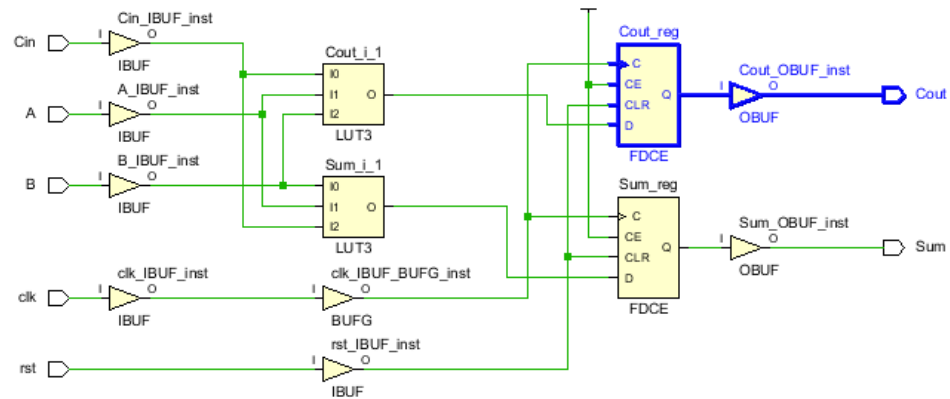
```

82
83 A <= '1';
84 B <= '0';
85 Cin <= '1';
86 wait for 30 ns;
87
88 A <= '0';
89 B <= '1';
90 Cin <= '1';
91 wait for 30 ns;
92
93 A <= '1';
94 B <= '1';
95 Cin <= '1';
96 wait for 30 ns;
97
98 end process;
99
100 END;

```



Critical Path και Συνολική Καθυστέρηση



Timing											
Unconstrained Paths - NONE - NONE - Setup											
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	
Path 1	∞	2	2	1	Cout_reg/C	Cout	4.076	3.276	0.800	∞	
Path 2	∞	2	2	1	Sum_reg/C	Sum	4.076	3.276	0.800	∞	
Path 3	∞	2	3	2	A	Sum_reg/D	1.932	1.132	0.800	∞	
Path 4	∞	2	3	2	B	Cout_reg/D	1.906	1.106	0.800	∞	
Path 5	∞	1	2	2	rst	Cout_reg/CLR	1.782	0.982	0.800	∞	
Path 6	∞	1	2	2	rst	Sum_reg/CLR	1.782	0.982	0.800	∞	

Όπως φαίνεται από την παραπάνω εικόνα, το critical path είναι από τον τελευταίο καταχωρητή στο Cout και η συνολική καθυστέρηση είναι 4.076ns.

2. Σύγχρονος Αθροιστής Διάδοσης κρατούμενου 4ων bit

Κώδικας και δομικό διάγραμμα (RTL)

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity Synchronous_4bit_Adder_Structural is
6      port(
7          clk : in  std_logic;
8          rst : in  std_logic;
9          A   : in  std_logic_vector(3 downto 0);
10         B   : in  std_logic_vector(3 downto 0);
11         Cin  : in  std_logic;
12         Sum  : out std_logic_vector(4 downto 0);
13         Cout : out std_logic
14     );
15 end entity; -- Synchronous_4bit_Adder_Structural
16
17 architecture structural_arch of Synchronous_4bit_Adder_Structural is
18
19     -----
20     -- Declarations of lower level components
21     -- used in this level of hierarchy
22
23     --Our Implementation is only based on building blocks of the
24     -- "Synchronous_Full_Adder_Behavioral", we also use
25     -- flip - flops but not in a structural way. The VLSI design was based on the
26     -- diagram given below.
27     component Synchronous_Full_Adder_Behavioral is
28         port(
29             clk : in  std_logic;
30             rst : in  std_logic;
31             A   : in  std_logic;
32             B   : in  std_logic;
33             Cin  : in  std_logic;
34             Sum  : out std_logic;
35             Cout : out std_logic
36         );
37     end component;
38
39     -----
40     -- Declarations of internal signals
41     -- used in this level of hierarchy
42     signal C, S : std_logic_vector(3 downto 0);
43     signal R01, R02, R03, R10a, R10b, R12, R13, R20a, R20b, R21a, R21b, R23, R30a,
44     -- R30b, R31a, R31b, R32a, R32b: std_logic := '0';
```

```

42
43 --4 stages of synchronous full adders according to our diagram
44 begin
45 -----
46 -- LEVEL 0 component instantiation --
47 -----
48 Synchronous_Full_Adder_Behavioral_INSTANCE_0 : Synchronous_Full_Adder_Behavioral
49     port map (
50         clk => clk,
51         rst => rst,
52         A  => A(0),
53         B  => B(0),
54         Cin => Cin,
55         Sum => S(0),
56         Cout => C(0)
57     );
58 -----
59 -- LEVEL 1 component instantiation --
60 -----
61
62
63 Synchronous_Full_Adder_Behavioral_INSTANCE_1 : Synchronous_Full_Adder_Behavioral
64     port map (
65         clk => clk,
66         rst => rst,
67         A  => R10a,
68         B  => R10b,
69         Cin => C(0),
70         Sum => S(1),
71         Cout => C(1)
72     );
73 -----
74 -- LEVEL 2 component instantiation --
75 -----
76 Synchronous_Full_Adder_Behavioral_INSTANCE_2 : Synchronous_Full_Adder_Behavioral
77     port map (
78         clk => clk,
79         rst => rst,
80         A  => R21a,
81         B  => R21b,
82         Cin => C(1),
83         Sum => S(2),
84         Cout => C(2)
85     );
86 -----
87 -- LEVEL 3 component instantiation --
88 -----
89 Synchronous_Full_Adder_Behavioral_INSTANCE_3 : Synchronous_Full_Adder_Behavioral

```



```

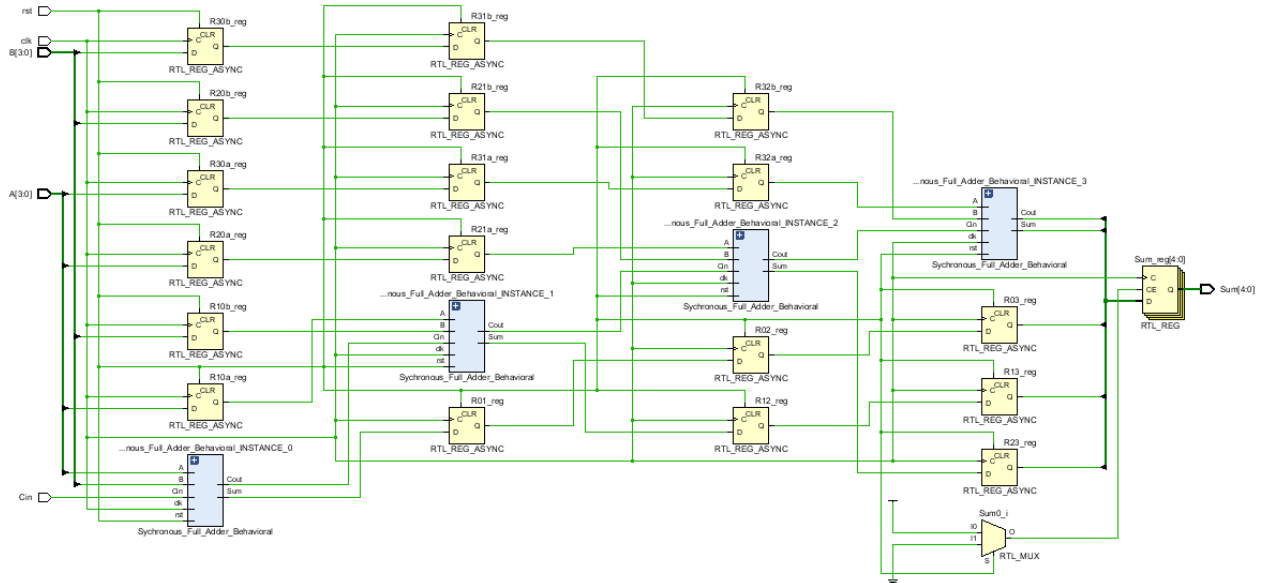
90  port map (
91      clk => clk,
92      rst => rst,
93      A   => R32a,
94      B   => R32b,
95      Cin => C(2),
96      Sum => S(3),
97      Cout => C(3)
98  );
99
100  --pipelining procedure:
101  -- We made a single if statement for every single flip - flop we want to
102  -- create. Not the best procedure but it works fine.
103  -- By creating a flip - flop structure the code would be more organized
104  -- and easy to read. Comments in the below statement
105  -- explain the use of our flip - flops in respect to the given diagram.
106  pipeline_proc: process(clk, rst)
107  begin
108      if (rst = '1') then
109          R01 <= '0'; R02 <= '0'; R03 <= '0';
110          R10a <= '0'; R10b <= '0'; R12 <= '0'; R13 <= '0';
111          R20a <= '0'; R20b <= '0'; R21a <= '0'; R21b <= '0';
112          R23 <= '0'; R30a <= '0'; R30b <= '0';
113          R31a <= '0'; R31b <= '0'; R32a <= '0'; R32b <= '0';
114
115      elsif (clk'event and clk='1') then
116          --delays for the s0 column
117          --After FA
118          Sum(0) <= R03; R03 <= R02; R02 <= R01; R01 <= S(0);
119
120
121          --delays for the Sum1 column
122          --After FA
123          Sum(1) <= R13; R13 <= R12; R12 <= S(1);
124          --Before FA
125          R10a <= A(1);
126          R10b <= B(1);
127
128
129          --delays for the Sum2 column
130          --After FA
131          Sum(2) <= R23; R23 <= S(2);
132          --Before FA
133          R21a <= R20a; R20a <= A(2);
134          R21b <= R20b; R20b <= B(2);
135

```

```

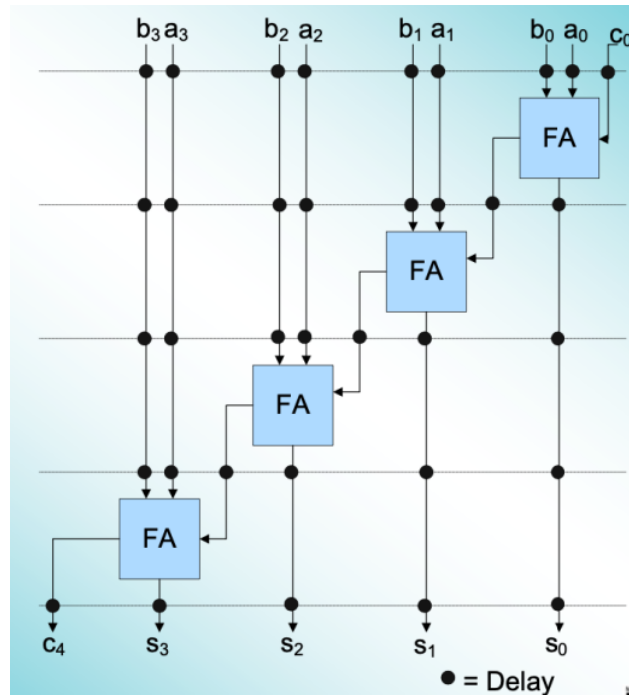
136
137
138      --delays for the Sum3 column
139      --After FA
140      Sum(3) <= S(3);
141      --Before FA
142      R32a <= R31a; R31a <= R30a; R30a <= A(3);
143      R32b <= R31b; R31b <= R30b; R30b <= B(3);
144
145      --delays for the Sum4 column
146      Sum(4) <= C(3);
147
148      end if;
149      end process;
150
151      end architecture ; -- arch

```



Προσθέσαμε επιπλέον καταχωρητές πριν και μετά τα full adder blocks, τόσους ώστε τα δεδομένα να είναι στο ίδιο επίπεδο την ίδια χρονική στιγμή. Για παράδειγμα, όταν ο πρώτος αθροιστής υπολογίσει το αποτέλεσμα, αυτό το εμφανιστεί στην έξοδο μετά από 3 καθυστερήσεις και αυτό διότι ο τελευταίος αθροιστής θα υπολογίσει τον ίδιο αριθμό 3 κύκλους μετά τον 1ο.

Πιο συγκεκριμένα, οι καθυστερήσεις σε κάθε στάδιο έγιναν με βάση το παρακάτω διάγραμμα, όπου στα σημεία που υπάρχουν κουκκίδες προστέθηκε μία καθυστέρηση δηλαδή ένας καταχωρητής.



Κώδικας testbench και Κυματομορφή εξόδου

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.numeric_std.ALL;
4  USE IEEE.std_logic_unsigned.ALL;
5
6
7  entity Synchronous_4bit_Adder_Structural_tb is
8  -- Port ( );
9  end entity;
10
11  architecture Bench of Synchronous_4bit_Adder_Structural_tb is
12
13  COMPONENT Synchronous_4bit_Adder_Structural is
14      Port (
15          clk : in  std_logic;
16          rst : in  std_logic;
17          A   : in  std_logic_vector(3 downto 0);
18          B   : in  std_logic_vector(3 downto 0);
19          Cin  : in  std_logic;
20          Sum   : out std_logic_vector(4 downto 0)
21      );
22  end COMPONENT;
23
24  SIGNAL A, B : STD_LOGIC_VECTOR(3 downto 0);
25  SIGNAL Sum : STD_LOGIC_VECTOR(4 downto 0);
26  SIGNAL Cin : STD_LOGIC := '0';

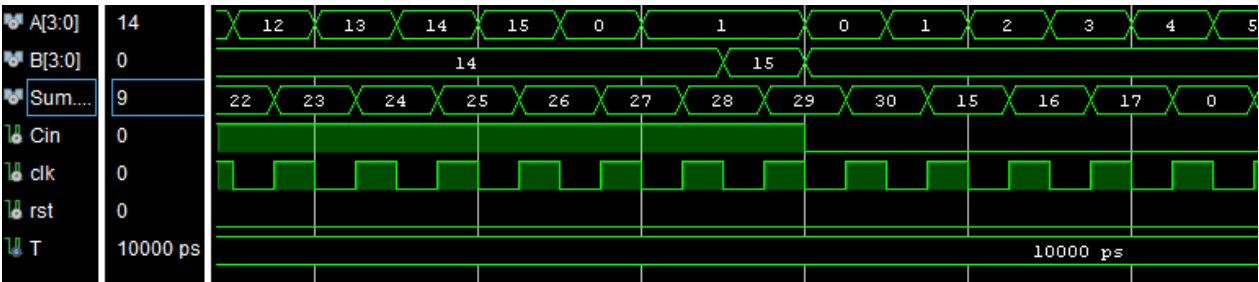
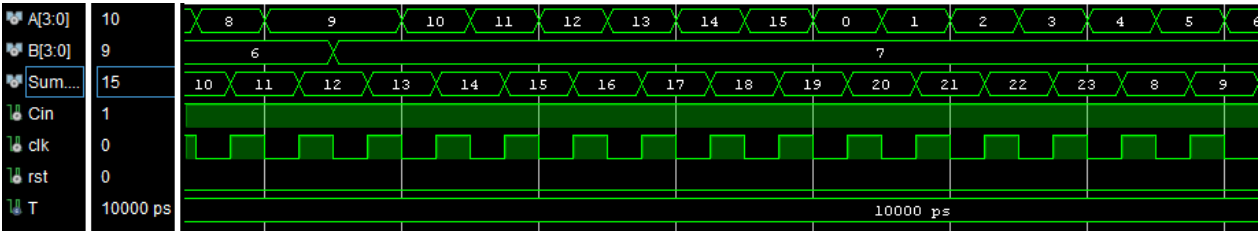
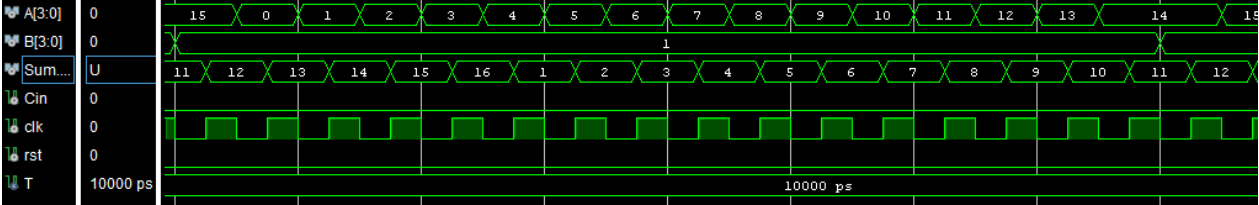
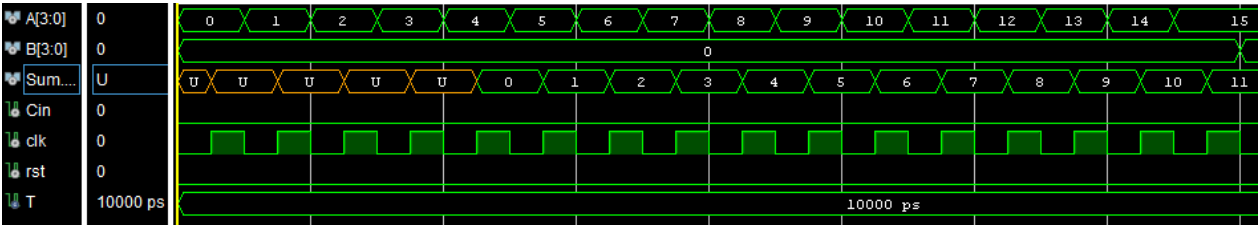
```

```

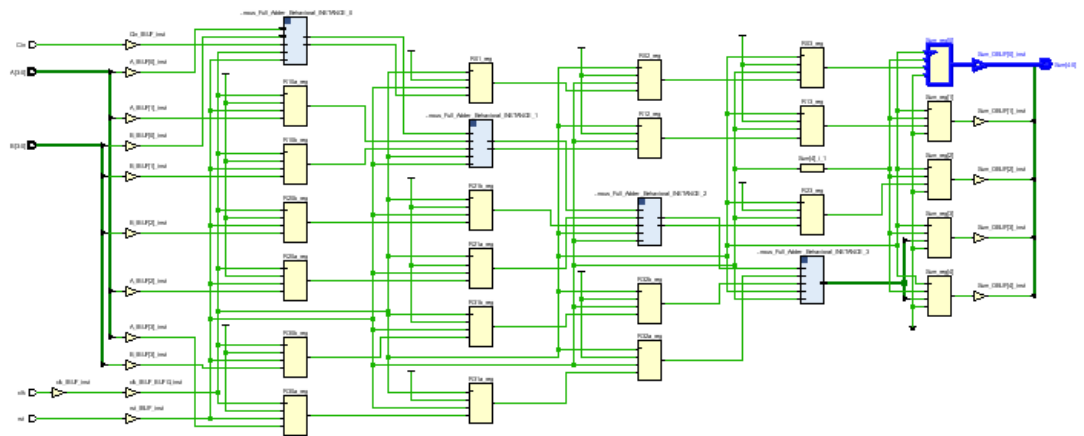
27 SIGNAL clk : std_logic;
28 SIGNAL rst : std_logic := '0';
29
30 CONSTANT T : TIME := 10 ns;
31
32 begin
33
34 uut: Synchronous_4bit_Adder_Structural PORT MAP (
35     clk => clk,
36     rst => rst,
37     A => A,
38     B => B,
39     Cin => Cin,
40     Sum => Sum);
41
42
43 stimuli: PROCESS
44 begin
45     A <= "0000";
46     B <= "0000";
47     WAIT FOR T;
48
49     FOR j IN 1 TO 15 LOOP
50         FOR i IN 1 TO 15 LOOP
51             A <= A + 1;
52             WAIT FOR T;
53         end LOOP;
54         B <= B + 1;
55         WAIT FOR T;
56     END LOOP;
57
58     IF Cin = '0' THEN
59         Cin <= '1';
60     ELSE
61         Cin <= '0';
62     END IF;
63
64 end PROCESS;
65
66 clk_gen: process begin
67     clk <= '0';
68     wait for T/2;
69     clk <= '1';
70     wait for T/2;
71 end process;
72
73
74

```

75 `end Bench;`



Critical Path και Συνολική Καθυστέρηση



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Reqd
Path 1	∞	2	2	1	Sum_reg[0]/C	Sum[0]	4.076	3.276	0.800	
Path 2	∞	2	2	1	Sum_reg[1]/C	Sum[1]	4.076	3.276	0.800	
Path 3	∞	2	2	1	Sum_reg[2]/C	Sum[2]	4.076	3.276	0.800	
Path 4	∞	2	2	1	Sum_reg[3]/C	Sum[3]	4.076	3.276	0.800	
Path 5	∞	2	2	1	Sum_reg[4]/C	Sum[4]	4.076	3.276	0.800	
Path 6	∞	2	3	27	rst	Sum_reg[0]/CE	2.407	1.106	1.301	

Όπως φαίνεται από την παραπάνω εικόνα, το critical path είναι από τον τελευταίο καταχωρητή στο Sum[0] και η συνολική καθυστέρηση είναι 4.076 ns.

Ο Παράλληλος Αθροιστής της 2ης Εργαστηριακής Άσκησης είχε ως καθυστέρηση, και άρα ως κύκλο ρολογιού θεωρώντας ότι είναι σύγχρονος, 5.942ns. Άρα, με την μέθοδο pipeline μειώσαμε την καθυστέρηση στο 1/3 της αρχικής. Όσον, αφορά την κατανάλωση πόρων, η μέθοδος pipeline χρησιμοποιεί σαφώς περισσότερο υλικό. Πιο συγκεκριμένα χρησιμοποιεί 3 επιπλέον καταχωρητές σε κάθε στάδιο, σε αντίθεση με τον προηγούμενο που δεν χρησιμοποιεί κανέναν. Όμως, τα πλεονεκτήματα του pipelining ξεπερνούν το παραπάνω μειονέκτημα καθώς το σύστημά μας γίνεται αρκετά πιο γρήγορο και καταναλώνει λιγότερη ισχύ.

3. Συστολικός Πολλαπλασιαστής διάδοσης κρατουμένου 4ων bit

Κώδικας και δομικό διάγραμμα (RTL)

D Flip Flop

```
1  Library IEEE;
2  USE IEEE.Std_logic_1164.all;
3
4  entity D_Flip_Flop_behavioral is
5      port(
6          Q : out std_logic;
7          Clk : in std_logic;
8          rst : in std_logic;
9          D :in  std_logic
10     );
11 end D_Flip_Flop_behavioral;
12 architecture Behavioral of D_Flip_Flop_behavioral is
13 begin
14     process(Clk)
15     begin
16         if(rst='1') then
17             Q<='0';
18         elsif(rising_edge(Clk)) then
19             Q <= D;
20         end if;
21     end process;
22 end Behavioral;
```

Building Block

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  --The following is the implementation of the Building block that we will use for
6  --We can observe the diagram of the building block at the upper left corner of
7  --the Multiplication diagram we quoted below.
8  entity Building_Block is
9      port(
10         clk : in std_logic;
11         rst : in std_logic;
12         Sin : in std_logic;
13         Ain  : in std_logic;
14         Bin  : in std_logic;
15         Cin : in std_logic;
```

```

15     Sout: out std_logic;
16     Cout: out std_logic;
17     Aout: out std_logic;
18     Bout: out std_logic
19 );
20 end entity; -- Building Block
21
22 architecture structural_arch of Building_Block is
23
24     -----
25     -- Declarations of lower level components
26     -- used in this level of hierarchy
27     component Synchronous_Full_Adder_Behavioral is
28     port(
29         clk : in std_logic;
30         rst : in std_logic;
31         A : in std_logic;
32         B : in std_logic;
33         Cin : in std_logic;
34         Sum : out std_logic;
35         Cout : out std_logic
36     );
37 end component;
38
39 component D_Flip_Flop_behavioral is
40 port(
41     Q : out std_logic;
42     Clk : in std_logic;
43     rst : in std_logic;
44     D : in std_logic
45 );
46 end component;
47
48     -----
49     -- Declarations of internal signals
50     -- used in this level of hierarchy
51 signal C1,C3 : std_logic; -- // C1 -> input FA // C3 -> between D's //
52 begin
53     C1<=Ain AND Bin;
54
55     -----
56     -- LEVEL 0 component instantiation --
57     -----
58     Synchronous_Full_Adder_Behavioral_INSTANCE_0 : Synchronous_Full_Adder_Behavioral
59     port map (
60         clk => clk,
61         rst => rst,
62         A => C1, --result of AND
63         B => Sin, --result of above addition
64         Cin => Cin,

```



```

63         Sum => Sout,
64         Cout => Cout
65     );
66 D_Flip_Flop_behavioral_INSTANCE_0 : D_Flip_Flop_behavioral
67     port map (
68         Q => Bout,
69         Clk => clk,
70         rst => rst,
71         D => Bin
72     );
73 -----
74 -- LEVEL 1 component instantiation --
75 -----
76 D_Flip_Flop_behavioral_INSTANCE_1 : D_Flip_Flop_behavioral
77     port map (
78         Q => C3,
79         Clk => clk,
80         rst => rst,
81         D => Ain
82     );
83 -----
84 -- LEVEL 2 component instantiation --
85 -----
86 D_Flip_Flop_behavioral_INSTANCE_2 : D_Flip_Flop_behavioral
87     port map (
88         Q => Aout,
89         Clk => clk,
90         rst => rst,
91         D => C3
92     );
93
94 end architecture ; -- arch

```

Σημειώνεται ότι το Building Block φαίνεται στην άνω δεξιά γωνία της φωτογραφίας που ακολουθεί και που δείχνει την διαγραμματική υλοποίηση του πολλαπλασιαστή.

Πολλαπλασιαστής

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  --Our design is based on the diagram given below the code part.
5  --Regarding the delays we use code as the one given below, to create delays via
6  --  ↪ the use of flip - flops(D):
7  --
8  -- delay_n:process (clk)                --n clock cycles
9  -- begin
10 --     if rising_edge(clk) then

```

```

10  --      delay0 <= delay1;
11  --      ....
12  --      ....
13  --      ....
14  --      delayn-1 <= A(x);
15  --      end if;
16  --      end process;
17
18  entity Synchronous_Systolic_4bit_Multiplier is
19      Port ( clk : in std_logic;
20            rst : in std_logic;
21            A  : in STD_LOGIC_VECTOR (3 downto 0);
22            B  : in STD_LOGIC_VECTOR (3 downto 0);
23            Cin : in STD_LOGIC;
24            Product : out STD_LOGIC_VECTOR (7 downto 0));
25  end Synchronous_Systolic_4bit_Multiplier;
26
27  architecture Structural of Synchronous_Systolic_4bit_Multiplier is
28      --Our single compoment used 16 times is the Synchronous FA
29      component Building_Block is
30      port(
31          clk : in std_logic;
32          rst : in std_logic;
33          Sin : in std_logic;
34          Ain : in std_logic;
35          Bin : in std_logic;
36          Cin : in std_logic;
37          Sout: out std_logic;
38          Cout: out std_logic;
39          Aout: out std_logic;
40          Bout: out std_logic
41      );
42  end component;
43  --About Delays:
44  --We have 4 categories:
45  --1)Delays for the Cin of each FA
46  --2)Delays for the "straight"(according to the diagram) inputs ,B
47  --3)Delays for the "diagonal"(according to the diagram) inputs ,A
48  --4)Delays for the Outputs of the last Full Addders-counting from top to
49      ↪ bottom-(giving us the multiplication result)
50  --
51  --In the following code we distinguish every type of delay from each other
52      ↪ according to the above enumeration.
53
54  --Signals regarding subtotals
55  signal P0 : std_logic_vector(9 downto 0):="0000000000"; --ready
56  signal P1 : std_logic_vector(8 downto 0):="0000000000"; --ready

```

```

56  signal P2 : std_logic_vector(7 downto 0) := "00000000"; --ready
57  signal P3 : std_logic_vector(6 downto 0) := "0000000"; --ready
58  signal P4 : std_logic_vector(4 downto 0) := "00000"; --ready
59  signal P5 : std_logic_vector(2 downto 0) := "000"; --ready
60  signal P6 : std_logic := '0'; --ready
61  signal P7 : std_logic := '0'; --ready
62
63  --signals regarding internal transports
64  signal C0 : std_logic_vector(3 downto 0) := "0000"; --Cout each gate
65  signal C1 : std_logic_vector(3 downto 0) := "0000"; --Cout each gate
66  signal C2 : std_logic_vector(3 downto 0) := "0000"; --Cout each gate
67  signal C3 : std_logic_vector(3 downto 0) := "0000"; --Cout each gate
68  signal A0 : std_logic_vector(3 downto 0) := "0000"; --diagonal
69  signal A1 : std_logic_vector(3 downto 0) := "0000"; --diagonal
70  signal A2 : std_logic_vector(3 downto 0) := "0000"; --diagonal
71  signal A3 : std_logic_vector(3 downto 0) := "0000"; --diagonal
72  signal B0 : std_logic_vector(3 downto 0) := "0000"; --straight
73  signal B1 : std_logic_vector(3 downto 0) := "0000"; --straight
74  signal B2 : std_logic_vector(3 downto 0) := "0000"; --straight
75  signal B3 : std_logic_vector(3 downto 0) := "0000"; --straight
76
77  --delays for A
78  signal delayA1 : std_logic := '0';
79  signal delayA2,delayA3 : std_logic := '0';
80  signal delayA4,delayA5,delayA6 : std_logic := '0';
81
82  --delays for B
83  signal delayB1,delayB2 : std_logic := '0';
84  signal delayB3,delayB4 : std_logic := '0';
85  signal delayB5,delayB6,delayB7 : std_logic := '0';
86  signal delayB8,delayB9,delayB10 : std_logic := '0';
87  signal delayB11,delayB12,delayB13,delayB14 : std_logic := '0';
88  --delays for C
89  signal delayC1,delayC3 : std_logic := '0';
90  signal delayC2,delayC4 : std_logic := '0';
91  signal delayC5,delayC6 : std_logic := '0';
92  begin
93  --
94  --
95  --
96  --
97  --
98  --
99  --
100
101  Building_Block_INSTANCE_0: Building_Block
102      port map ( clk => clk,
103                rst => rst,

```

```

104         Sin => '0',
105         Ain => A(0),
106         Bin => B(0),
107         Cin => '0',
108         Sout => P0(0),
109         Cout => C0(0),
110         Aout => A0(0),
111         Bout => B0(0));
112
113 --Type 4
114 delay_P0:process (clk)                                --10 clock cycles
115 begin
116     if rising_edge(clk) then
117         Product(0) <= P0(9);
118         P0(9) <= P0(8);
119         P0(8) <= P0(7);
120         P0(7) <= P0(6);
121         P0(6) <= P0(5);
122         P0(5) <= P0(4);
123         P0(4) <= P0(3);
124         P0(3) <= P0(2);
125         P0(2) <= P0(1);
126         P0(1) <= P0(0);
127     end if;
128 end process;
129 --Type 3
130 delay_1:process (clk)
131 begin
132     if rising_edge(clk) then
133         delayA1 <= A(1);
134     end if;
135 end process;
136
137 Building_Block_INSTANCE_1: Building_Block
138     port map ( clk => clk,
139               rst => rst,
140               Sin => '0',
141               Ain => delayA1,
142               Bin => B0(0),
143               Cin => C0(0),
144               Sout => P1(0),
145               Cout => C0(1),
146               Aout => A1(0),
147               Bout => B0(1));
148 --Type 3
149 delay_2:process (clk)                                --2 clock cycles
150 begin
151     if rising_edge(clk) then

```

```

152     delayA2 <= delayA3;
153     delayA3 <= A(2);
154 end if;
155 end process;
156
157 Building_Block_INSTANCE_2: Building_Block
158     port map ( clk => clk,
159               rst => rst,
160               Sin => '0',
161               Ain => delayA2,
162               Bin => B0(1),
163               Cin => C0(1),
164               Sout => P2(0),
165               Cout => C0(2),
166               Aout => A2(0),
167               Bout => B0(2));
168
169 --Type 3
170 delay_3:process (clk)                                --3 clock cycles
171 begin
172     if rising_edge(clk) then
173         delayA4 <= delayA5;
174         delayA5 <= delayA6;
175         delayA6 <= A(3);
176     end if;
177 end process;
178
179 Building_Block_INSTANCE_3: Building_Block
180     port map ( clk => clk,
181               rst => rst,
182               Sin => '0',
183               Ain => delayA4,
184               Bin => B0(2),
185               Cin => C0(1),
186               Sout => P3(0),
187               Cout => C0(3),
188               Aout => A3(0),
189               Bout => B0(3));
190
191 --Type 2
192 delay_4:process (clk)                                --2 clock cycles
193 begin
194     if rising_edge(clk) then
195         delayB1 <= delayB2;
196         delayB2 <= B(1);
197     end if;
198 end process;
199
200 Building_Block_INSTANCE_4: Building_Block
201     port map ( clk => clk,

```

```

200         rst => rst,
201         Sin => P1(0),
202         Ain => A0(0),
203         Bin => delayB1,
204         Cin => '0',
205         Sout =>P1(1),
206         Cout => C1(0),
207         Aout => A0(1),
208         Bout => B1(0));
209
210 --Type 4
211 delay_P1:process (clk) --8 clock cycles
212 begin
213     if rising_edge(clk) then
214         Product(1) <= P1(8);
215         P1(8) <= P1(7);
216         P1(7) <= P1(6);
217         P1(6) <= P1(5);
218         P1(5) <= P1(4);
219         P1(4) <= P1(3);
220         P1(3) <= P1(2);
221         P1(2) <= P1(1);
222     end if;
223 end process;
224
225 Building_Block_INSTANCE_5: Building_Block
226     port map ( clk => clk,
227               rst => rst,
228               Sin => P2(0),
229               Ain => A1(0),
230               Bin => B1(0),
231               Cin => C1(0),
232               Sout =>P2(1),
233               Cout => C1(1),
234               Aout => A1(1),
235               Bout => B1(1));
236
237 Building_Block_INSTANCE_6: Building_Block
238     port map ( clk => clk,
239               rst => rst,
240               Sin => P3(0),
241               Ain => A2(0),
242               Bin => B1(1),
243               Cin => C1(1),
244               Sout =>P3(1),
245               Cout => C1(2),
246               Aout => A2(1),
247               Bout => B1(2));

```

```

248  --Type 1
249  delay_5:process (clk)                                --5 clock cycles
250  begin
251      if rising_edge(clk) then
252          delayC1 <= C0(3);
253          --delayC2 <= C0(3);
254      end if;
255  end process;
256
257  Building_Block_INSTANCE_7: Building_Block
258      port map ( clk => clk,
259                  rst => rst,
260                  Sin => delayC1,
261                  Ain => A3(0),
262                  Bin => B1(2),
263                  Cin => C1(2),
264                  Sout => P4(0),
265                  Cout => C1(3),
266                  Aout => A3(1),
267                  Bout => B1(3));
268  --Type 2
269  delay_6:process (clk)                                --5 clock cycles
270  begin
271      if rising_edge(clk) then
272          delayB3 <= delayB4;
273          delayB4 <= delayB5;
274          delayB5 <= delayB6;
275          delayB6 <= B(2);
276      end if;
277  end process;
278
279  Building_Block_INSTANCE_8: Building_Block
280      port map ( clk => clk,
281                  rst => rst,
282                  Sin => P2(1),
283                  Ain => A0(1),
284                  Bin => delayB3,
285                  Cin => '0',
286                  Sout => P2(2),
287                  Cout => C2(0),
288                  Aout => A0(2),
289                  Bout => B2(0));
290  --Type 4
291  delay_P2:process (clk)                                --7 clock cycles
292  begin
293      if rising_edge(clk) then
294          Product(2) <= P2(7);
295          P2(7) <= P2(6);

```

```

296     P2(6) <= P2(5);
297     P2(5) <= P2(4);
298     P2(4) <= P2(3);
299     P2(3) <= P2(2);
300     end if;
301 end process;
302
303 Building_Block_INSTANCE_9: Building_Block
304     port map ( clk => clk,
305                rst => rst,
306                Sin => P3(1),
307                Ain => A1(1),
308                Bin => B2(0),
309                Cin => C2(0),
310                Sout => P3(2),
311                Cout => C2(1),
312                Aout => A1(2),
313                Bout => B2(1));
314
315 Building_Block_INSTANCE_10: Building_Block
316     port map ( clk => clk,
317                rst => rst,
318                Sin => P4(0),
319                Ain => A2(1),
320                Bin => B2(1),
321                Cin => C2(1),
322                Sout => P4(1),
323                Cout => C2(2),
324                Aout => A2(2),
325                Bout => B2(2));
326 --Type 1
327 delay_7:process (clk)                                --2 clock cycles
328 begin
329     if rising_edge(clk) then
330         delayC3 <= C1(3);
331         --delayC4 <= C1(3);
332     end if;
333 end process;
334
335 Building_Block_INSTANCE_11: Building_Block
336     port map ( clk => clk,
337                rst => rst,
338                Sin => delayC3,
339                Ain => A3(1),
340                Bin => B2(2),
341                Cin => C2(2),
342                Sout => P5(0),
343                Cout => C2(3),

```



```

344         Aout => A3(2),
345         Bout => B2(3));
346 --Type 2
347 delay_8:process (clk)                                --6 clock cycles
348 begin
349     if rising_edge(clk) then
350         delayB7 <= delayB8;
351         delayB8 <= delayB9;
352         delayB9 <= delayB10;
353         delayB10 <= delayB11;
354         delayB11 <= delayB12;
355         delayB12 <= B(3);
356     end if;
357 end process;
358
359 Building_Block_INSTANCE_12: Building_Block
360     port map ( clk => clk,
361                rst => rst,
362                Sin => P3(2),
363                Ain => A0(2),
364                Bin => delayB7,
365                Cin => '0',
366                Sout => P3(3),
367                Cout => C3(0),
368                Aout => A0(3),
369                Bout => B3(0));
370 --Type 4
371 delay_P3:process (clk)                                --4 clock cycles
372 begin
373     if rising_edge(clk) then
374         Product(3) <= P3(6);
375         P3(6) <= P3(5);
376         P3(5) <= P3(4);
377         P3(4) <= P3(3);
378     end if;
379 end process;
380
381 Building_Block_INSTANCE_13: Building_Block
382     port map ( clk => clk,
383                rst => rst,
384                Sin => P4(1),
385                Ain => A1(2),
386                Bin => B3(0),
387                Cin => C3(0),
388                Sout => P4(2),
389                Cout => C3(1),
390                Aout => A1(3),
391                Bout => B3(1));

```

```

392  --Type 4
393  delay_P4:process (clk)                                --3 clock cycles
394  begin
395      if rising_edge(clk) then
396          Product(4) <= P4(4);
397          P4(4) <= P4(3);
398          P4(3) <= P4(2);
399      end if;
400  end process;
401
402  Building_Block_INSTANCE_14: Building_Block
403      port map ( clk => clk,
404                  rst => rst,
405                  Sin => P5(0),
406                  Ain => A2(2),
407                  Bin => B3(1),
408                  Cin => C3(1),
409                  Sout => P5(1),
410                  Cout => C3(2),
411                  Aout => A2(3),
412                  Bout => B3(2));
413  --Type 4
414  delay_P5:process (clk)                                --2 clock cycles
415  begin
416      if rising_edge(clk) then
417          Product(5) <= P5(2);
418          P5(2) <= P5(1);
419      end if;
420  end process;
421  --Type 1
422  delay_9:process (clk)                                --2 clock cycles
423  begin
424      if rising_edge(clk) then
425          delayC5 <= C2(3);
426          --delayC6 <= C2(3);
427      end if;
428  end process;
429
430  Building_Block_INSTANCE_15: Building_Block
431      port map ( clk => clk,
432                  rst => rst,
433                  Sin => delayC5,
434                  Ain => A3(2),
435                  Bin => B3(2),
436                  Cin => C3(2),
437                  Sout => P6,
438                  Cout => P7,
439                  Aout => A3(3),

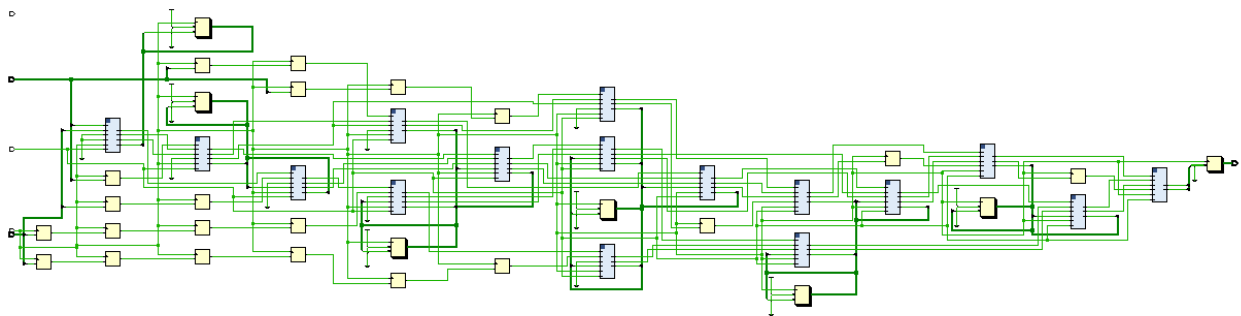
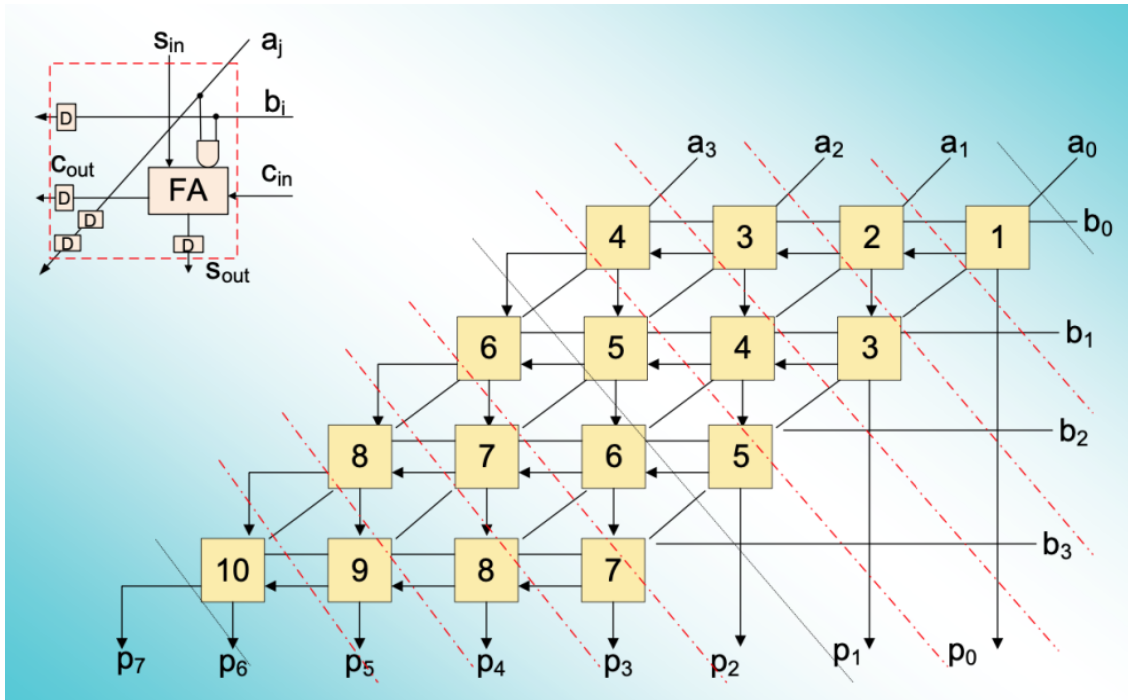
```

```

440      Bout => B3(3));
441  --Type 4
442  delay_P6_P7:process (clk)                --2 clock cycles
443  begin
444      if rising_edge(clk) then
445          Product(7) <= P7;
446          Product(6) <= P6;
447      end if;
448  end process;
449
450  end Structural;

```

Παρακάτω δίνεται το διάγραμμα το οποίο παρουσιάζει οπτικά την υλοποίηση που πραγματοποιήσαμε καθώς και το rtl σχηματικό.



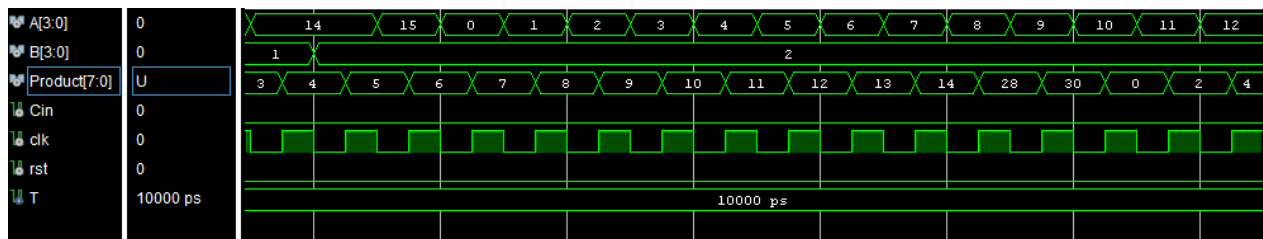
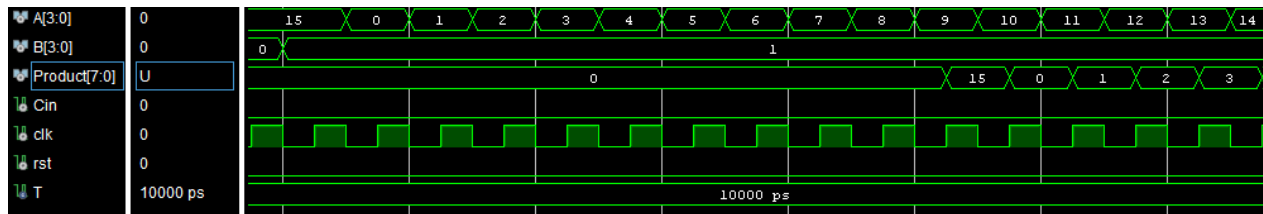
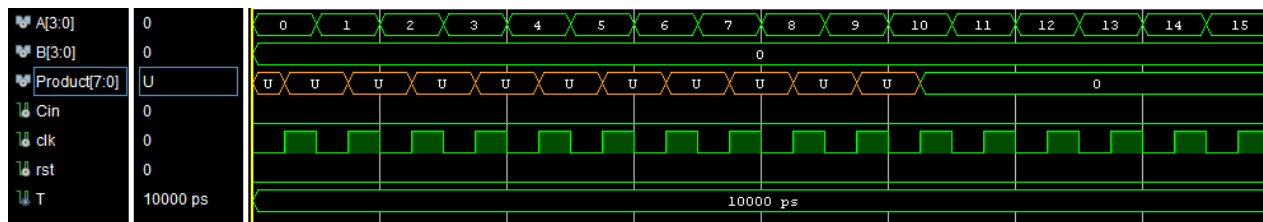
Κώδικας testbench και Κυματομορφή εξόδου

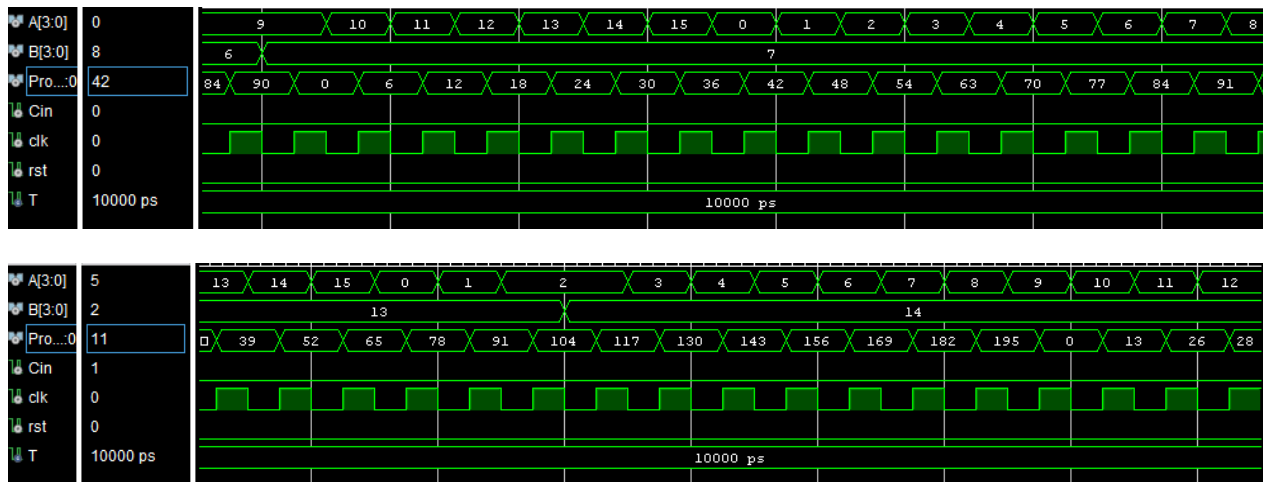
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.numeric_std.ALL;
4  USE IEEE.std_logic_unsigned.ALL;
5
6
7  entity Synchronous_Systolic_4bit_Multiplier_tb is
8  -- Port ( );
9  end entity;
10
11 architecture Bench of Synchronous_Systolic_4bit_Multiplier_tb is
12
13 COMPONENT Synchronous_Systolic_4bit_Multiplier is
14     Port ( clk : in std_logic;
15           rst : in std_logic;
16           A : in STD_LOGIC_VECTOR (3 downto 0);
17           B : in STD_LOGIC_VECTOR (3 downto 0);
18           Cin : in STD_LOGIC;
19           Product : out STD_LOGIC_VECTOR (7 downto 0));
20 end COMPONENT;
21
22 SIGNAL A, B : STD_LOGIC_VECTOR(3 downto 0);
23 SIGNAL Product : STD_LOGIC_VECTOR(7 downto 0);
24 SIGNAL Cin : STD_LOGIC := '0';
25 SIGNAL clk : std_logic;
26 SIGNAL rst : std_logic := '0';
27
28 CONSTANT T : TIME := 10 ns;
29
30 begin
31
32 uut: Synchronous_Systolic_4bit_Multiplier PORT MAP (
33     clk => clk,
34     rst => rst,
35     A => A,
36     B => B,
37     Cin => Cin,
38     Product => Product);
39
40
41 stimuli: PROCESS
42 begin
43     A <= "0000";
44     B <= "0000";
45     WAIT FOR T;
46
```

```

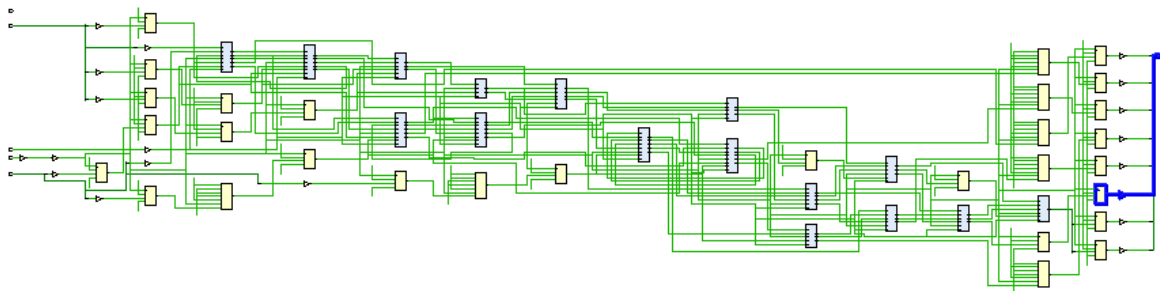
47   FOR j IN 1 TO 15 LOOP
48       FOR i IN 1 TO 15 LOOP
49           A <= A + 1;
50           WAIT FOR T;
51       end LOOP;
52       B <= B + 1;
53       WAIT FOR T;
54   END LOOP;
55
56   IF Cin = '0' THEN
57       Cin <= '1';
58   ELSE
59       Cin <= '0';
60   END IF;
61
62 end PROCESS;
63
64 clk_gen: process begin
65     clk <= '0';
66     wait for T/2;
67     clk <= '1';
68     wait for T/2;
69 end process;
70
71
72
73 end Bench;

```





Critical Path και Συνολική Καθυστερήση



ns Timing x											
Unconstrained Paths - NONE - NONE - Setup											
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Req	
Path 1	∞	2	2	1	Product_reg[5]/C	Product[5]	4.076	3.276	0.800		
Path 2	∞	2	2	1	Product_reg[6]/C	Product[6]	4.076	3.276	0.800		
Path 3	∞	2	2	1	Product_reg[7]/C	Product[7]	4.076	3.276	0.800		
Path 4	∞	2	2	1	Product_reg[0]/C	Product[0]	4.058	3.258	0.800		
Path 5	∞	2	2	1	Product_reg[1]/C	Product[1]	4.058	3.258	0.800		
Path 6	∞	2	2	1	Product_reg[2]/C	Product[2]	4.058	3.258	0.800		

Όπως φαίνεται από την παραπάνω εικόνα, το critical path είναι από τον τελευταίο καταχωρητή στο Product[5] και η συνολική καθυστέρηση είναι 4.076ns.