ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΙΣΤΩΝ

ΨΗΦΙΑΚΑ VLSI – 8ο ΕΞΑΜΗΝΟ

Μαρκέτος Νικόδημος – ΑΜ: 03117095

Τζομάκα Αφροδίτη – ΑΜ: 03117107

Ομάδα Β2

Ζήτημα 1: Συνδυαστικός Ημιαθροιστής (Dataflow)

Στο συγκεκριμένο θέμα, καλούμαστε να υλοποιήσουμε την περιγραφή της οντότητας του δυαδικού, συνδυαστικού ημιαθροιστή με περιγραφή ροής δεδομένων. Για την υλοποίηση αυτή λοιπόν έχουμε τον παρακάτω κώδικα, μαζί με το αντίστοιχο testbench:

Code of HA

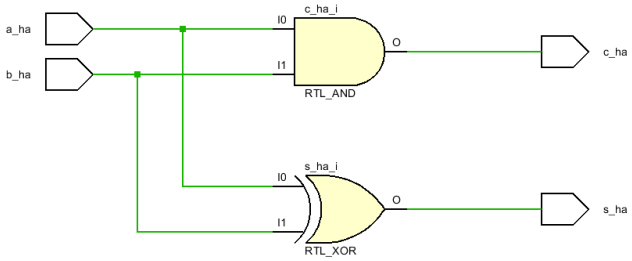
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity HA is
    Port ( a_ha: in STD_LOGIC;
           b_ha: in STD_LOGIC;
           s_ha: out STD_LOGIC;
           c_ha: out STD_LOGIC
          );
end HA;

architecture Dataflow of HA is
begin
    s_ha<= a_ha xor b_ha;
    c_ha<= a_ha and b_ha;
end Dataflow;
```

Test Bench of HA

```
library IEEE;
_LOGIC_1164.ALL;
entity HA_tb is
end HA_tb;

architecture test_bench of HA_tb is
component HA
port(
    a_ha: in STD_LOGIC;
    b_ha: in STD_LOGIC;
    s_ha: out STD_LOGIC;
    c_ha: out STD_LOGIC
    );
end component;
--Input Signals
signal a_ha: STD_LOGIC:='0';
signal b_ha: STD_LOGIC:='0';
--Output Signals
signal s_ha: STD_LOGIC;
signal c_ha: STD_LOGIC;
```

```
begin
UUT: HA port map(
    a_ha=>a_ha,
    b_ha=>b_ha,
    s_ha=>s_ha,
    c_ha=>c_ha
    );
tests: process
begin
    a_ha<='0';
    b_ha<='0';  -- expected output s=0 c=0
    wait for 10 ps;
    a_ha<='0';
    b_ha<='1'; -- expected output s=1 c=0
    wait for 10 ps;
    a_ha<='1';
    b_ha<='0'; -- expected output s=1 c=0
    wait for 10 ps;
    a_ha<='1';
    b_ha<='1';
    wait for 10 ps;
    wait; -- expected output s=0 c=1
end process;
end test_bench;
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name	1	Slice LUTs (17600)	Bonded IOB (100)
N HA		1	4

TIMING REPORT:

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	3	4	2	a_ha	c_ha	5.377	3.778	1.599	∞	input port clock		
Path 2	∞	3	4	2	a_ha	s_ha	5.351	3.752	1.599	∞	input port clock		

Από τα παραπάνω προκύπτει ότι το critical path είναι εκείνο από την είσοδο (a_ha) στο κρατούμενο (c_ha) με συνολική καθυστέρηση 5.377 ns.

Ζήτημα 2: Πλήρης Αθροιστής

α) Structural Περιγραφή

Στο συγκεκριμένο θέμα, καλούμαστε να υλοποιήσουμε την περιγραφή της οντότητας του πλήρους αθροιστή (FA), με περιγραφή ροής δομής, τόσο ως ένα ακολουθιακό όσο και ως ένα συνδυαστικό κύκλωμα. Για την υλοποίηση αυτή λοιπόν έχουμε τους παρακάτω κώδικες, μαζί με τα αντίστοιχα testbenches:

- Συνδυαστικό Κύκλωμα:

Structural Code of FA_comb

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

entity FA_comb_str is
    Port ( A     : in STD_LOGIC;
           B     : in STD_LOGIC;
           C_in  : in STD_LOGIC;
           C_out  : out STD_LOGIC;
           SUM    : out STD_LOGIC );
end FA_comb_str;

architecture Structural of FA_comb_str is
signal HA1_S: STD_LOGIC;
signal HA1_C: STD_LOGIC;
signal HA2_C: STD_LOGIC;

component HA is
    port(
        a_ha: in STD_LOGIC;
        b_ha: in STD_LOGIC;
        s_ha: out STD_LOGIC;
        c_ha: out STD_LOGIC
```

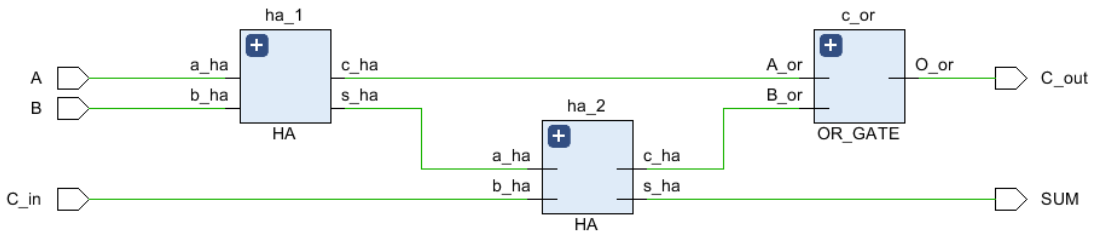
Test Bench of FA_comb

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

entity FA_comb_str_tb is
end FA_comb_str_tb;

architecture test_bench of FA_comb_str_tb is

component FA_comb_str
port(
    A: in STD_LOGIC;
    B: in STD_LOGIC;
    C_in: in STD_LOGIC;
    C_out: out STD_LOGIC;
    SUM: out STD_LOGIC
    );
end component;

--Input Signals
signal  counter: STD_LOGIC_VECTOR(2 downto 0):= (others=>
'0');
--Output Signals
```

```
        );
end component;

component OR_GATE is
    port(
        a_or: in STD_LOGIC;
        b_or: in STD_LOGIC;
        o_or: out STD_LOGIC
        )
        ;
end component;

begin
    ha_1: HA
    port map(
        a_ha => A,
        b_ha => B,
        s_ha => HA1_S,
        c_ha => HA1_C
    );
    ha_2: HA
    port map(
        a_ha => HA1_S,
        b_ha => C_in,
        s_ha => SUM,
        c_ha => HA2_C

    );
    c_or: OR_GATE port map( HA1_C , HA2_C , C_out);
end Structural;
```

```
signal SUM: STD_LOGIC;
signal C_out: STD_LOGIC;
begin
UUT: FA_comb_str port map(
    A=>counter(2),
    B=>counter(1),
    C_in=>counter(0),
    C_out=>C_out,
    SUM=>SUM
    );

tests: process
begin
    for i in 0 to 7 loop
        wait for 10 ps;
        counter <= std_logic_vector(to_unsigned(i,3));

    end loop;
    end process;
end test_bench;
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name	Slice LUTs (17600)	Bonded IOB (100)
FA_comb_str	1	5

TIMING SUMMARY:

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

Από τα παραπάνω προκύπτει ότι το critical path είναι εκείνο από την είσοδο (B) στο κρατούμενο (C_out) με συνολική καθυστέρηση 5.377 ns.

- Ακολουθιακό κύκλωμα:

Structural Code of FA _seq

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

entity FA_seq_str is
    Port ( A     : in STD_LOGIC;
           B     : in STD_LOGIC;
           C_in  : in STD_LOGIC;
           Clk   : in STD_LOGIC;
           reset : in STD_LOGIC;
           C_out  : out STD_LOGIC;
           SUM    : out STD_LOGIC );
end FA_seq_str;

architecture Structural of FA_seq_str is

signal HA1_S: STD_LOGIC;
signal HA1_C: STD_LOGIC;
signal HA2_C: STD_LOGIC;
signal temp_sum: std_logic;



component HA is
    port(
        a_ha: in STD_LOGIC;
        b_ha: in STD_LOGIC;
        s_ha: out STD_LOGIC;
        c_ha: out STD_LOGIC

        );
end component;


begin

    process(clk)
        begin
            if (rising_edge(clk)) then
                if (reset='1') then
                    c_out<= '0';
                    sum<='0';
                else
```

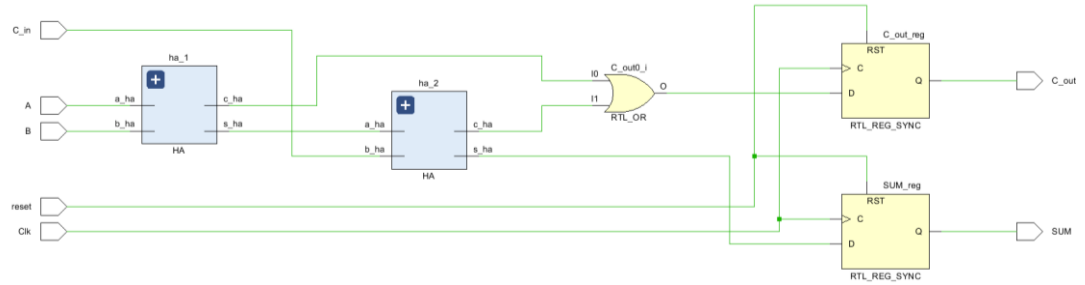
Test Bench of FA_seq

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.numeric_std.ALL;


entity FA_comb_str_tb is

end FA_comb_str_tb;


architecture test_bench of FA_comb_str_tb is


component FA_comb_str

port(

    A: in STD_LOGIC;

    B: in STD_LOGIC;

    C_in: in STD_LOGIC;

    C_out: out STD_LOGIC;

    SUM: out STD_LOGIC

    );

end component;


--Input Signals

signal  counter: STD_LOGIC_VECTOR(2 downto 0):= (others=>'0');

--Output Signals

signal SUM: STD_LOGIC;

signal C_out: STD_LOGIC;

begin
```

```
                    sum<=temp_sum;
                    c_out<=HA1_C or HA2_C;
                end if;
            end if;
    end process;

    ha_1: HA
        port map(
            a_ha => A,
            b_ha => B,
            s_ha => HA1_S,
            c_ha => HA1_C

        );
    ha_2: HA
        port map(
            a_ha => HA1_S,
            b_ha => C_in,
            s_ha => temp_SUM,
            c_ha => HA2_C

           );

end Structural;
```

```
UUT: FA_comb_str port map(

    A=>counter(2),

    B=>counter(1),

    C_in=>counter(0),

    C_out=>C_out,

    SUM=>SUM

    );


tests: process

begin

    for i in 0 to 7 loop

        wait for 10 ps;

        counter <= std_logic_vector(to_unsigned(i,3));

    end loop;

    end process;

end test_bench
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name	Slice LUTs (17600)	Slice Registers (35200)	Bonded IOB (100)	BUFGCTRL (32)
FA_seq_str	1	2	7	1

TIMING SUMMARY:

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	2	2	1	C_out_reg/C	C_out	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	B	C_out_reg/D	1.932	1.132	0.800	∞	input port clock		
Path 4	∞	2	3	2	B	SUM_reg/D	1.906	1.106	0.800	∞	input port clock		
Path 5	∞	1	2	2	reset	C_out_reg/R	1.782	0.982	0.800	∞	input port clock		
Path 6	∞	1	2	2	reset	SUM_reg/R	1.782	0.982	0.800	∞	input port clock		

Από τα παραπάνω προκύπτει ότι τα critical paths είναι εκείνα από την είσοδο των καταχωρητών κρατουμένου ή αθροίσματος (C_out_reg/SUM_reg) ως την τελική έξοδο (C_out/SUM) με συνολική καθυστέρηση 4.076 ns.

β) Behavioral Περιγραφή

Στο συγκεκριμένο θέμα, καλούμαστε να υλοποιήσουμε την περιγραφή της οντότητας του πλήρους αθροιστή (FA), με περιγραφή συμπεριφοράς, τόσο ως ένα ακολουθιακό όσο και ως ένα συνδυαστικό κύκλωμα. Για την υλοποίηση αυτή λοιπόν έχουμε τους παρακάτω κώδικες, μαζί με τα αντίστοιχα testbenches:

- Συνδυαστικό Κύκλωμα:

Behavioral Code of FA (comb)

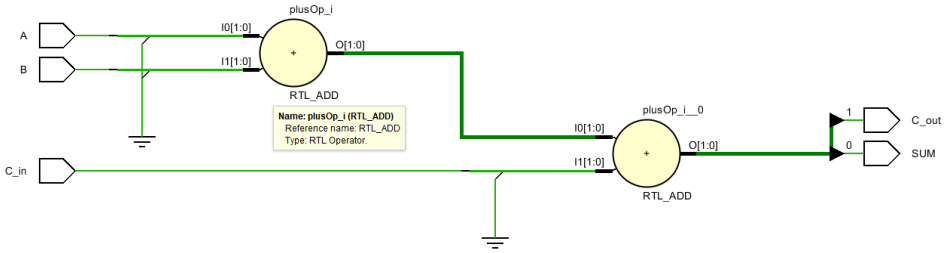
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity FA_comb_beh is
        Port ( A: in STD_LOGIC;
               B: in STD_LOGIC;
               C_in: in STD_LOGIC;
               C_out: out STD_LOGIC;
               SUM: out STD_LOGIC );
end FA_comb_beh;

architecture Behavioral of FA_comb_beh is
signal temp: std_logic_vector(1 downto 0);
begin
    process(A,B,C_in,temp)
        begin
        temp<= ('0'& a) + ('0'& b )+ ('0'& c_in);
        SUM <= temp(0);
        C_out <= temp(1);
    end process;

end Behavioral;
```

Test Bench of FA

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


entity FA_comb_beh_tb is
end FA_comb_beh_tb;

architecture test_bench of FA_comb_beh_tb is
component FA_comb_beh
port(
    A: in STD_LOGIC;
    B: in STD_LOGIC;
    C_in: in STD_LOGIC;
    C_out: out STD_LOGIC;
    SUM: out STD_LOGIC
    );
end component;
--Input Signals
signal  counter: STD_LOGIC_VECTOR(2 downto 0):= (others=>'0');
--Output Signals
signal SUM: STD_LOGIC;
signal C_out: STD_LOGIC;
begin


UUT: FA_comb_beh port map(
    A=>counter(2),
    B=>counter(1),
    C_in=>counter(0),
    C_out=>C_out,
    SUM=>SUM
    );
tests: process
```

```
begin
    for i in 0 to 7 loop
        wait for 10 ps;
        counter <= std_logic_vector(to_unsigned(i,3));
    end loop;
    end process;
end test_bench;
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name	1	Slice LUTs (17600)	Bonded IOB (100)
N FA_comb_beh		1	5

TIMING REPORT:

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	3	4	2	C_in	C_out	5.377	3.778	1.599	∞	input port clock		
Path 2	∞	3	4	2	C_in	SUM	5.351	3.752	1.599	∞	input port clock		

Από τα παραπάνω προκύπτει ότι το critical path είναι εκείνο από την είσοδο του κρατουμένου (c_in) έως το κρατούμενο εξόδου (c_out) με συνολική καθυστέρηση 5.377 ns.

- Ακολουθιακό κύκλωμα:

Behavioral Code of FA _seq

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

entity FA_seq_beh is
 Port ( A: in STD_LOGIC;
        B: in STD_LOGIC;
        C_in: in STD_LOGIC;
        CLK: in STD_LOGIC;
        reset: in STD_LOGIC;
```

Test Bench of FA_seq

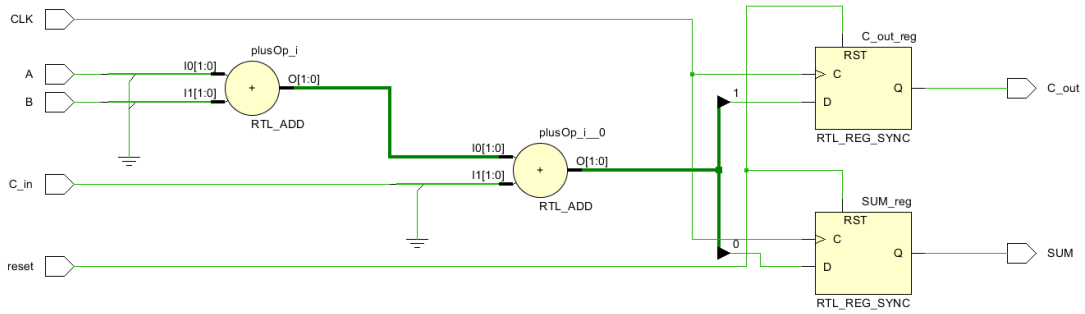
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

entity FA_seq_beh_tb is
end FA_seq_beh_tb;

architecture test_bench of FA_seq_beh_tb is
component FA_seq_beh
port(
```

```
        C_out: out STD_LOGIC;
        SUM: out STD_LOGIC
        );
end FA_seq_beh;

architecture Behavioral of FA_seq_beh is
signal addition_vector: std_logic_vector(1 downto 0);
signal a_temp : std_logic_vector(1 downto 0);
signal b_temp : std_logic_vector(1 downto 0);
signal c_in_temp : std_logic_vector(1 downto 0);
begin
        process(CLK)
            begin
                if(rising_edge(CLK)) then
                    if(reset='1') then
                        C_out<='0';
                        SUM<='0';
                    else
                        SUM <= addition_vector(0);
                        C_out <= addition_vector(1);
                    end if;
                end if;
        end process;
    addition_vector<= ('0'& a) + ('0'& b )+ ('0'& c_in);
end Behavioral;
```

```
    A: in STD_LOGIC;
    B: in STD_LOGIC;
    C_in: in STD_LOGIC;
    Clk: in STD_LOGIC;
    reset: in STD_LOGIC;
    C_out: out STD_LOGIC;
    SUM: out STD_LOGIC
    );
end component;
--Input Signals
signal A_B_Cin : STD_LOGIC_VECTOR(2 downto 0) := (others => '0');
signal Clk: STD_LOGIC:='0';
signal reset: STD_LOGIC:='0';
--Output Signals
signal C_out: STD_LOGIC;
signal SUM: STD_LOGIC;
--Clock
constant CLKP : time := 10ps;
begin
UUT: FA_seq_beh port map(
    A=>A_B_Cin(2),
    B=>A_B_Cin(1),
    C_in=>A_B_Cin(0),
    Clk=>Clk,
    reset=>reset,
    C_out=>C_out,
    SUM=>SUM
    );
clk_proc: process
    begin
        clk <= '0';
        wait for CLKP/2;
        clk <= '1';
        wait for CLKP/2;
    end process;
tests: process
    begin
        for i in 0 to 7 loop
            reset<='0';
            A_B_Cin <= std_logic_vector(to_unsigned(i,3));
```

```
            wait for CLKP;
        end loop;
        -- Checking if reset works
        reset<='1';
        A_B_Cin <= "111";
        wait for CLKP;
        reset<='0';
        A_B_Cin <= "111";
        wait for CLKP;
    end process;
end test_bench;
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name	Slice LUTs (17600)	Slice Registers (35200)	Bonded IOB (100)	BUFGCTRL (32)
FA_seq_beh	1	2	7	1

TIMING REPORT:

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	2	2	1	C_out_reg/C	C_out	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	C_in	C_out_reg/D	1.932	1.132	0.800	∞	input port clock		
Path 4	∞	2	3	2	C_in	SUM_reg/D	1.906	1.106	0.800	∞	input port clock		
Path 5	∞	1	2	2	reset	C_out_reg/R	1.782	0.982	0.800	∞	input port clock		
Path 6	∞	1	2	2	reset	SUM_reg/R	1.782	0.982	0.800	∞	input port clock		

Από τα παραπάνω προκύπτει ότι τα critical paths είναι εκείνα από την είσοδο των καταχωρητών κρατουμένου ή αθροίσματος (C_out_reg/SUM_reg) ως την τελική έξοδο (C_out/SUM) με συνολική καθυστέρηση 4.076 ns.

Ζήτημα 3: Παράλληλος Αθροιστής Διάδοσης Κρατουμένου 4bits (Structural)

α) Συνδυαστικό κύκλωμα

Στο συγκεκριμένο θέμα, καλούμαστε να υλοποιήσουμε την περιγραφή της οντότητας του παράλληλου αθροιστή διάδοσης κρατουμένου (RCA) 4 bits, με περιγραφή δομής. Για την υλοποίηση αυτή λοιπόν έχουμε τον παρακάτω κώδικα, μαζί με το αντίστοιχο testbench:

Code of RCA

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


entity RCA is
    Port( x: in STD_LOGIC_VECTOR(3 downto 0);
          y: in STD_LOGIC_VECTOR(3 downto 0);
          RCA_Cin: in STD_LOGIC;
          s: out STD_LOGIC_VECTOR(3 downto 0);
          RCA_Cout: out STD_LOGIC
         );
end RCA;

architecture Structural of RCA is
component FA_comb_beh is
    port( A: in STD_LOGIC;
          B: in STD_LOGIC;
          C_in : in STD_LOGIC;
          C_out: out STD_LOGIC;
          SUM: out STD_LOGIC
        );
end component;
signal c: STD_LOGIC_VECTOR(2 downto 0);

begin
fa_1 : FA_comb_beh
        port map(
            A => x(0),
            B => y(0),
            C_in => RCA_Cin,
            C_out => c(0),
            SUM => s(0)
```

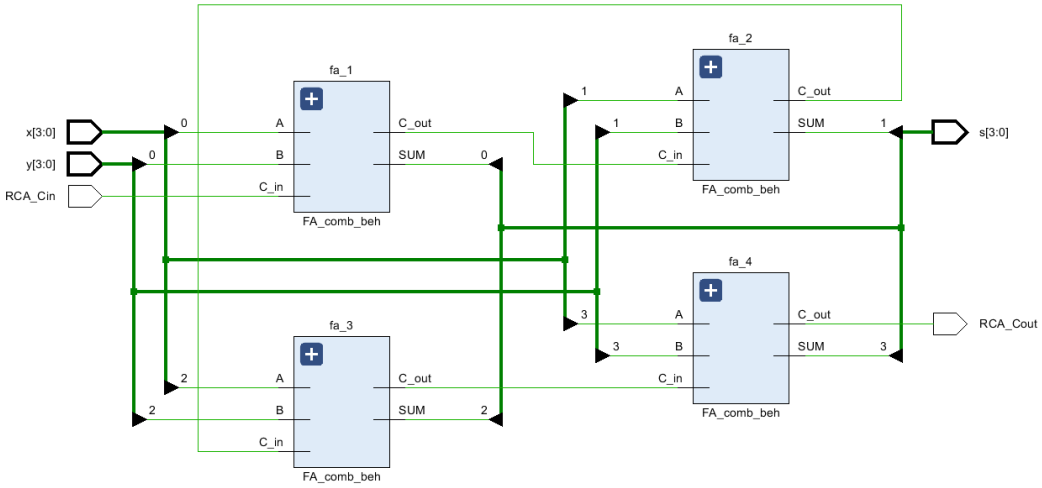
Test Bench of RCA

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

entity RCA_tb is
end RCA_tb;

architecture test_bench of RCA_tb is

component RCA
    port(
         x: in STD_LOGIC_VECTOR(3 downto 0);
         y: in STD_LOGIC_VECTOR(3 downto 0);
         RCA_Cin: in STD_LOGIC;
         s: out STD_LOGIC_VECTOR(3 downto 0);
         RCA_Cout: out STD_LOGIC
         );
end component;

-- Input Signals
signal x_tb: STD_LOGIC_VECTOR(3 downto 0);
signal y_tb: STD_LOGIC_VECTOR(3 downto 0);
signal RCA_cin_TB: STD_LOGIC;

-- Output Signals
signal s_tb: STD_LOGIC_VECTOR(3 downto 0);
signal rca_cout_tb : STD_LOGIC;
begin

UUT: RCA port
           map(
           x => x_tb,
```

```
        );

fa_2 : FA_comb_beh
        port map (
            A => x(1),
            B => y(1),
            C_in => c(0),
            C_out => c(1),
            SUM => s(1)
        );
fa_3 : FA_comb_beh
        port map (
            A => x(2),
            B => y(2),
            C_in => c(1),
            C_out => c(2),
            SUM => s(2)
        );

fa_4 : FA_comb_beh
        port map (
            A => x(3),
            B => y(3),
            C_in => c(2),
            C_out => RCA_Cout,
            SUM => s(3)
        );
end Structural;
```

```
            y => y_tb,
            RCA_Cin => RCA_Cin_tb,
            s => s_tb,
            RCA_Cout => RCA_Cout_tb
            );

tests: process
        begin
             x_tb <="0101";
             y_tb <="0000";
             RCA_Cin_tb <='0';
             wait for 10 ps;
             x_tb <="0101";
             y_tb <="0110";
             RCA_Cin_tb <='0';
             wait for 10 ps
             x_tb <="0101";
             y_tb <="0111";
             RCA_Cin_tb <='0';
             wait for 10 ps;
             x_tb <="0011";
             y_tb <="0010";
             RCA_Cin_tb <='0';
             wait for 10 ps;
             x_tb <="1010";
             y_tb <="0110";
             RCA_Cin_tb <='1';
             wait for 10 ps;
             x_tb <="1111";
             y_tb <="0101";
             RCA_Cin_tb <='0';
             wait for 10 ps;
             x_tb <="1010";
             y_tb <="0111";
             RCA_Cin_tb <='0';
             wait for 10 ps;
        end process;
end test_bench;
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name	Slice LUTs (17600)	Bonded IOB (100)
RCA	4	14

TIMING REPORT:

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

Από τα παραπάνω προκύπτει ότι τα critical paths είναι εκείνα από την είσοδο (x[0]) ως τις εξόδους s[2], s[3] με συνολική καθυστέρηση 5.970 ns.

β) Pipelined

Στο συγκεκριμένο θέμα, καλούμαστε να εφαρμόσουμε την τεχνική του Pipeline στο παραπάνω κύκλωμα. Για την υλοποίηση αυτή λοιπόν έχουμε τον παρακάτω κώδικα, μαζί με το αντίστοιχο testbench:

Code of RCA Pipelined

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity rca_pipe2 is
    Port (
    a: in std_logic_vector (3 downto 0);
    b: in std_logic_vector (3 downto 0);
    c_in: in std_logic;
```

Test Bench of RCA Pipelined

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

entity rca_pipe2_tb is
end rca_pipe2_tb;

architecture test_bench of rca_pipe2_tb is

component rca_pipe2
Port (
```

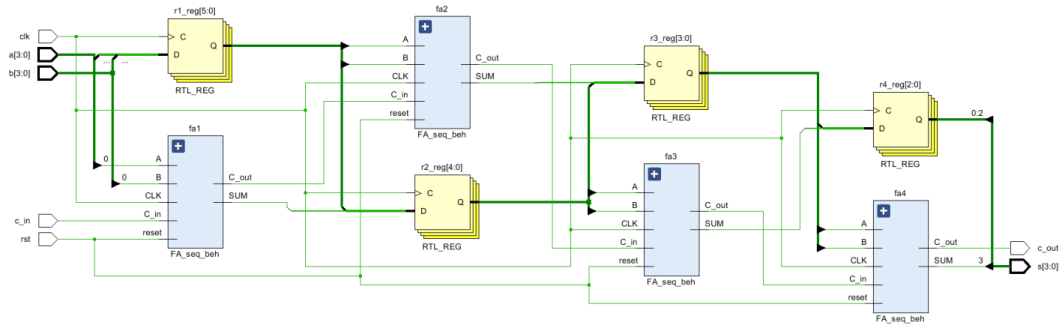
```
    clk: in std_logic;
    rst: in std_logic;
    s: out std_logic_vector (3 downto 0);
    c_out: out std_logic
     );
end rca_pipe2;

architecture Behavioral of rca_pipe2 is

component FA_seq_beh is
Port ( A     : in STD_LOGIC;
       B     : in STD_LOGIC;
       C_in  : in STD_LOGIC;
       Clk   : in STD_LOGIC;
       reset : in STD_LOGIC;
       C_out  : out STD_LOGIC;
       SUM    : out STD_LOGIC );
end component;

signal r1: std_logic_vector(5 downto 0) := (others=>'0');
signal r2: std_logic_vector(4 downto 0) := (others=>'0');
signal r3: std_logic_vector(3 downto 0) := (others=>'0');
signal r4: std_logic_vector(2 downto 0) := (others=>'0');

signal c1,c2,c3,s1,s2,s3,s4,c4 : std_logic := '0';

begin

reg: process(clk)
begin
    if (rising_edge(clk)) then
        r4 <= r3(1 downto 0) & s3 ;
          -- r4(s1,s2,s3)
        r3 <= r2(4) & r2(2) & r2(0) & s2;
          -- r3(a3,b3,s1,s2)
        r2 <= r1(5 downto 4) & r1(2 downto 1) & s1;
          -- r2(a3,a2,b3,b2,s1)
        r1 <= a(3 downto 1) & b(3 downto 1) ;
          -- r1(a3,a2,a1,b3,b2,b1)
     end if;
end process;
```

```
    a: in std_logic_vector (3 downto 0);
    b: in std_logic_vector (3 downto 0);
    c_in: in std_logic;
    clk: in std_logic;
    rst: in std_logic;
    s: out std_logic_vector (3 downto 0);
    c_out: out std_logic
     );
end component;

--Input Signals
signal a: std_logic_vector (3 downto 0):= (others=>'0');
signal b: std_logic_vector (3 downto 0):= (others=>'0');
signal c_in: std_logic:= '0';
signal clk: std_logic:='0';
signal rst: std_logic:='0';

--Output Signals
signal s: std_logic_vector(3 downto 0);
signal c_out: std_logic := '0';

--Clock
constant CLKP : time := 1ns;

begin
    UUT: rca_pipe2 port map(
        a=>a,
        b=>b,
        c_in=>c_in,
        clk=>clk,
        rst=>rst,
        s=>s,
        c_out=>c_out
        );
    clk_proc: process
        begin
            clk <= '0';
            wait for CLKP/2;
            clk <= '1';
            wait for CLKP/2;
        end process;
    tests: process
    begin
        rst<='1';
        a<="0010";
        b<="0010";
        c_in<='0';
        wait for CLKP;
        rst<='0';
        a<="0010";
        b<="0001";
        wait for CLKP;
        a<="0100";
        b<="0100";
        rst<='0';
```

```
    s <= s4 & r4(0) & r4(1) & r4(2);
    c_out <= c4;

fa1: fa_seq_beh port map(a(0), b(0), c_in,   clk, rst, c1, s1);
fa2: fa_seq_beh port map(r1(3), r1(0), c1,   clk, rst, c2, s2);
fa3: fa_seq_beh port map(r2(3), r2(1), c2,   clk, rst, c3, s3);
fa4: fa_seq_beh port map(r3(3), r3(2), c3,   clk, rst, c4, s4);

end Behavioral;
```

```
        wait for CLKP;
        a<="1111";
        b<="1111";
        wait for CLKP;
        a<="1100";
        b<="0001";
        wait for CLKP;
        a<="1100";
        b<="1011";
        wait for CLKP;
        a<="0100";
        b<="0001";
        wait for CLKP;
        a<="0110";
        b<="1011";
        wait;
    end process;
end test_bench;
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name	Slice LUTs (17600)	Slice Registers (35200)	Bonded IOB (100)	BUFGCTRL (32)
rca_pipe2	5	24	16	1
fa1 (FA_seq_beh)	1	2	0	0
fa2 (FA_seq_beh_0)	1	2	0	0
fa3 (FA_seq_beh_1)	1	2	0	0
fa4 (FA_seq_beh_2)	1	2	0	0

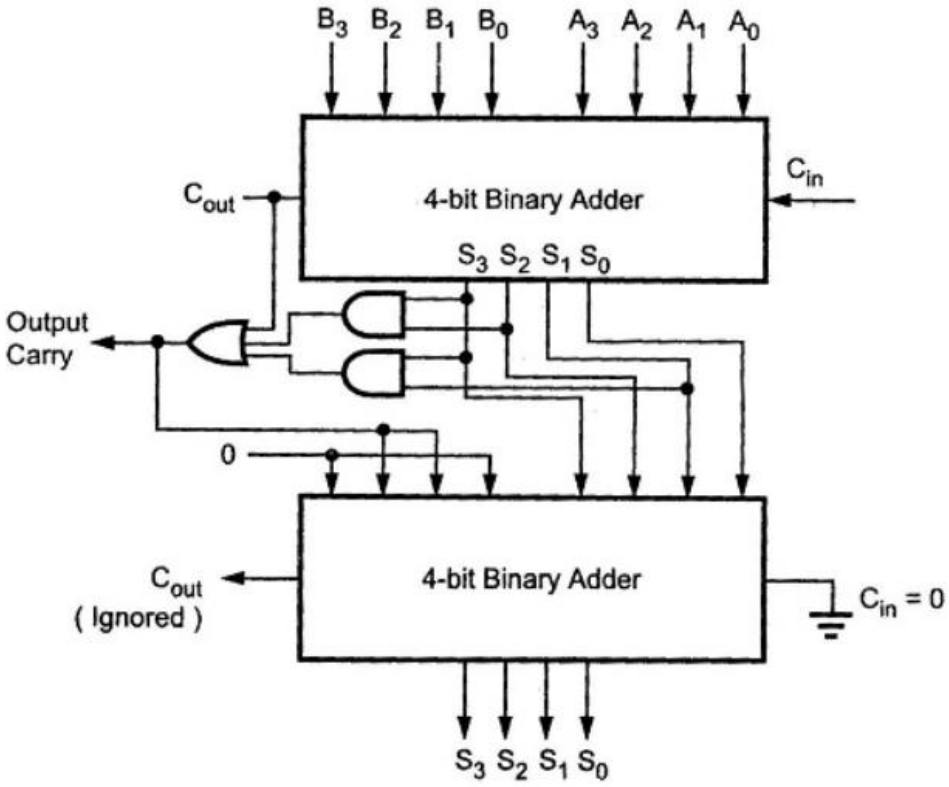
TIMING REPORT:

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	2	2	1	fa4/C_out_reg/C	c_out	4.076	3.276	0.800	∞			
Path 2	∞	2	2	1	r4_reg[1]/C	s[1]	4.076	3.276	0.800	∞			
Path 3	∞	2	2	1	r4_reg[0]/C	s[2]	4.076	3.276	0.800	∞			
Path 4	∞	2	2	1	fa4/SUM_reg/C	s[3]	4.076	3.276	0.800	∞			
Path 5	∞	2	2	1	r4_reg[2]/C	s[0]	4.058	3.258	0.800	∞			
Path 6	∞	2	3	2	c_in	fa1/C_out_reg/D	1.932	1.132	0.800	∞	input port clock		
Path 7	∞	2	3	2	c_in	fa1/SUM_reg/D	1.906	1.106	0.800	∞	input port clock		
Path 8	∞	1	2	8	rst	fa1/C_out_reg/R	1.782	0.982	0.800	∞	input port clock		
Path 9	∞	1	2	8	rst	fa1/SUM_reg/R	1.782	0.982	0.800	∞	input port clock		
Path 10	∞	1	2	8	rst	fa2/C_out_reg/R	1.782	0.982	0.800	∞	input port clock		

Από τα παραπάνω προκύπτει ότι τα critical paths είναι: από το τελικό FA component (fa4) προς τα τελικά c_out και s[3] και από τον τελικό καταχωρητή του Pipeline (r4) προς τις εξόδους s[1], s[1] με συνολική καθυστέρηση 4.076 ns το καθένα.

Ζήτημα 4: Συνδυαστικός BCD Πλήρης Αθροιστής 4 bits (Structural)

Στο συγκεκριμένο θέμα, καλούμαστε να υλοποιήσουμε την περιγραφή της οντότητας του συνδυαστικού πλήρους αθροιστή BCD - 4 bits, με περιγραφή δομής. Για την υλοποίηση αυτή Βασιστήκαμε στο ακόλουθο σχήμα από την γνωστή θεωρία:



(όπου 4-bit Binary Adder ο RCA που υλοποιήσαμε στο προηγούμενο ερώτημα). Παραθέτουμε λοιπόν παρακάτω τον αντίστοιχο κώδικα, μαζί με το αντίστοιχο testbench:

Code of BCD_FA

```
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);
          BCD_FA_Cin: in STD_LOGIC;
          S: out STD_LOGIC_VECTOR(3 downto 0);
          BCD_FA_Cout: out STD_LOGIC
```

Test Bench of BCD

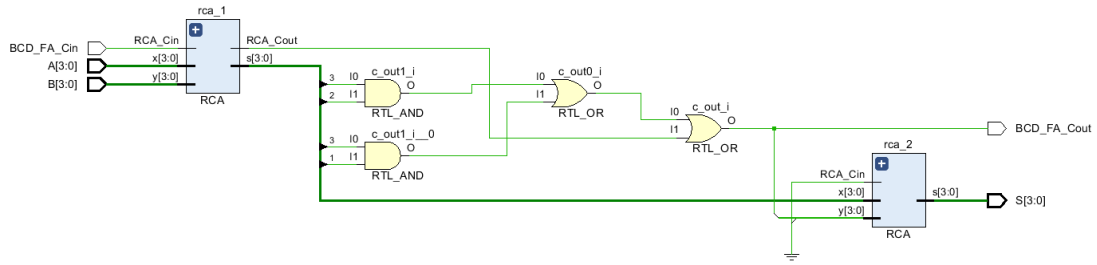
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

entity BCD_FA_tb is
end BCD_FA_tb;

architecture test_bench of BCD_FA_tb is
    component BCD_FA
```

```
        );
end BCD_FA;

architecture Structural of BCD_FA is
    component RCA is
        port(   x: in STD_LOGIC_VECTOR(3 downto 0);
                y: in STD_LOGIC_VECTOR(3 downto 0);
                RCA_Cin: in STD_LOGIC;
                s: out STD_LOGIC_VECTOR(3 downto 0);
                RCA_Cout: out STD_LOGIC
            );
    end component;

    signal s_tmp,offset:STD_LOGIC_VECTOR(3 downto 0);
    signal c_1, c_out: STD_LOGIC;

begin
    rca_1: RCA
            port map(
                x => A,
                y => B,
                RCA_Cin => BCD_FA_Cin,
                s => s_tmp,
                RCA_Cout => c_1
            );
    c_out <= (s_tmp(3) and s_tmp(2)) or (s_tmp(3) and
 s_tmp(1)) or c_1;
    BCD_FA_Cout <= c_out;
    offset <= '0' & c_out & c_out & '0';
    rca_2: RCA
            port map(
                x => s_tmp,
                y => offset,
                RCA_Cin => '0',
                s => S
            );
end Structural; -- Structural
```

```
        port(
            A: in STD_LOGIC_VECTOR(3 downto 0);
            B: in STD_LOGIC_VECTOR(3 downto 0);
            BCD_FA_Cin: in STD_LOGIC;
            S: out STD_LOGIC_VECTOR(3 downto 0);
            BCD_FA_Cout: out STD_LOGIC
            );
    end component;

    signal A_tb, B_tb: STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal BCD_FA_Cin_tb: STD_LOGIC;
    signal S_tb: STD_LOGIC_VECTOR(3 downto 0);
    signal BCD_FA_Cout_tb: STD_LOGIC;

begin
    UUT: BCD_FA port map(
                        A => A_tb,
                        B => B_tb,
                        BCD_FA_Cin => BCD_FA_Cin_tb,
                        S => S_tb,
                        BCD_FA_Cout => BCD_FA_Cout_tb
                        );
    test:process
         begin
            A_tb <="0101"; -- 5
            B_tb <="0000"; -- 0
            BCD_FA_Cin_tb <='0';
            wait for 10 ps; -- sum = 5 = 0000 0101

            A_tb <="0101"; -- 5
            B_tb <="0110"; -- 6
            wait for 10 ps; -- sum = 11 = 0001 0001

            A_tb <="0101"; -- 5
            B_tb <="0111"; -- 7
            wait for 10 ps; -- sum = 12 = 0001 0010

            A_tb <="1001"; -- 9
            B_tb <="0110"; -- 6
            wait for 10 ps; -- sum = 15 = 0001 0101
```

```
        A_tb <="1001"; -- 9
        B_tb <="1001"; -- 9
        wait for 10 ps; -- sum = 18 = 0001 1000


    end process;
end test_bench ; -- test_bench
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name	Slice LUTs (17600)	Slice (4400)	LUT as Logic (17600)	Bonded IOB (100)
BCD_FA	6	2	6	14

TIMING REPORT:

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	4	3	4	A[0]	S[3]	8.293	4.027	4.266	∞	input port clock		
Path 2	∞	4	3	4	A[0]	S[2]	8.270	3.792	4.477	∞	input port clock		
Path 3	∞	4	3	4	A[0]	BCD_FA_Cout	7.966	4.052	3.914	∞	input port clock		
Path 4	∞	4	3	4	A[0]	S[1]	7.607	4.033	3.573	∞	input port clock		
Path 5	∞	3	2	3	A[0]	S[0]	6.820	3.681	3.139	∞	input port clock		

Από τα παραπάνω προκύπτει ότι το critical path είναι εκείνο από την είσοδο (A[0]) έως την έξοδο (S[3]) με συνολική καθυστέρηση 8.293 ns.

Ζήτημα 5: Παράλληλος BCD Αθροιστής 4 ψηφίων (Structural)

Στο συγκεκριμένο θέμα, καλούμαστε να υλοποιήσουμε την περιγραφή της οντότητας του συνδυαστικού πλήρους αθροιστή BCD - 4 bits, με περιγραφή δομής. Για την υλοποίηση αυτή λοιπόν έχουμε τον παρακάτω κώδικα, μαζί με το αντίστοιχο testbench:

Code of BCD_4PA

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity BCD_4PA is
    Port( A: in STD_LOGIC_VECTOR(15 downto 0);
```

Test Bench

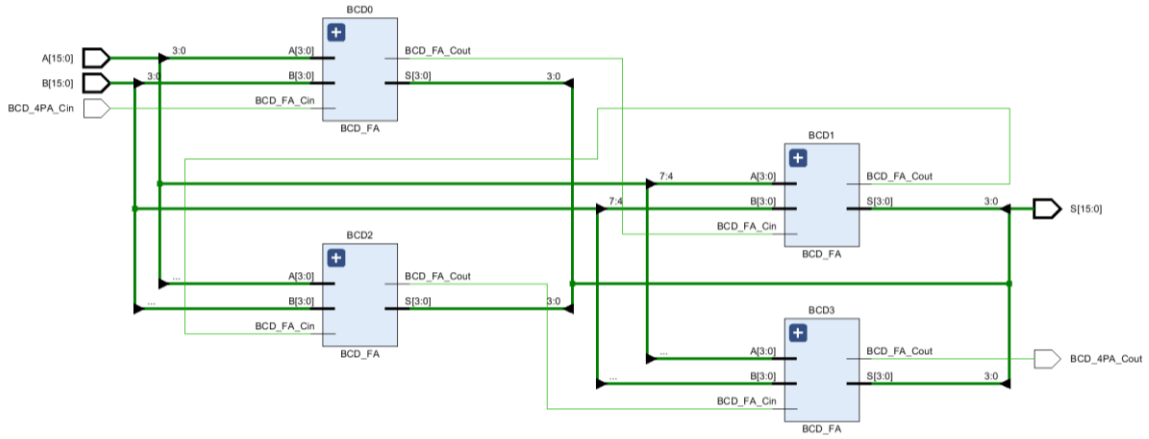
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
```

```
          B: in STD_LOGIC_VECTOR(15 downto 0);
          BCD_4PA_Cin: in STD_LOGIC;
          S: out STD_LOGIC_VECTOR(15 downto 0);
          BCD_4PA_Cout: out STD_LOGIC
         );
end BCD_4PA;

architecture Structural of BCD_4PA is
    component BCD_FA is
        port(   A: in STD_LOGIC_VECTOR(3 downto 0);
                B: in STD_LOGIC_VECTOR(3 downto 0);
                BCD_FA_Cin: in STD_LOGIC;
                S: out STD_LOGIC_VECTOR(3 downto 0);
                BCD_FA_Cout: out STD_LOGIC
            );
    end component;

    signal c0, c1, c2: STD_LOGIC;

begin
    BCD0 : BCD_FA port map(
                        A => A(3 downto 0),
                        B => B(3 downto 0),
                        BCD_FA_Cin => BCD_4PA_Cin,
                        S => S(3 downto 0),
                        BCD_FA_Cout => c0
                        );
    BCD1 : BCD_FA port map(
                        A => A(7 downto 4),
                        B => B(7 downto 4),
                        BCD_FA_Cin => c0,
                        S => S(7 downto 4),
                        BCD_FA_Cout => c1
                        );
    BCD2 : BCD_FA port map(
                        A => A(11 downto 8),
                        B => B(11 downto 8),
                        BCD_FA_Cin => c1,
                        S => S(11 downto 8),
                        BCD_FA_Cout => c2
```

```
entity BCD_4PA_tb is
end BCD_4PA_tb;

architecture test_bench of BCD_4PA_tb is
    component BCD_4PA
        port(
            A: in STD_LOGIC_VECTOR(3 downto 0);
            B: in STD_LOGIC_VECTOR(3 downto 0);
            BCD_4PA_Cin: in STD_LOGIC;
            S: out STD_LOGIC_VECTOR(3 downto 0);
            BCD_4PA_Cout: out STD_LOGIC
            );
    end component;

    signal A_tb, B_tb: STD_LOGIC_VECTOR(15 downto 0)
                                        := "0000000000000000";
    signal BCD_4PA_Cin_tb: STD_LOGIC;
    signal S_tb: STD_LOGIC_VECTOR(15 downto 0);
    signal BCD_4PA_Cout_tb: STD_LOGIC;

begin
    UUT: BCD_4PA port map(
                        A => A_tb,
                        B => B_tb,
                        BCD_4PA_Cin => BCD_4PA_Cin_tb,
                        S => S_tb,
                        BCD_4PA_Cout => BCD_4PA_Cout_tb
                        );
    test:process
         begin
            A_tb <="0001000000001111"; -- 1009
            B_tb <="0000010100110001"; -- 531
            BCD_4PA_Cin_tb <='0';
            wait for 10 ps; -- sum = 1540 = 0001 0101 0100 0000

            A_tb <="0101000001010000"; -- 5050
            wait for 10 ps; -- sum = 5581 = 0101 0101 1000 0001

            A_tb <="0001001000010001"; -- 1211
            B_tb <="0000000100000001"; -- 101
```

```
                        );                                          wait for 10 ps; -- sum = 1312 = 0001 0011 0001 0010
    BCD3 : BCD_FA port map(
                        A => A(15 downto 12),                   end process;
                        B => B(15 downto 12),           end test_bench ; -- test_bench
                        BCD_FA_Cin => c2,
                        S => S(15 downto 12),
                        BCD_FA_Cout => BCD_4PA_Cout
                        );
end Structural ; -- Structural
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name 1	Slice LUTs (17600)	Bonded IOB (100)
N BCD_4PA	24	50

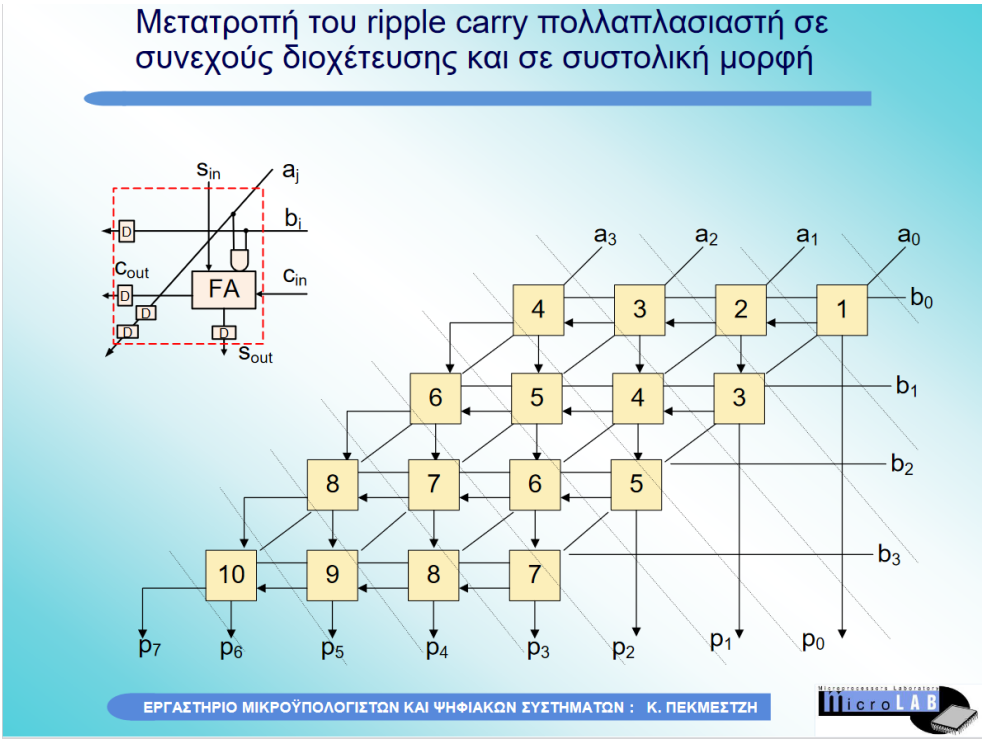
TIMING REPORT:

Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	10	11	7	A[0]	S[13]	9.505	4.608	4.897	∞	input port clock		
Path 2	∞	10	11	7	A[0]	BCD_4PA_Cout	9.499	4.602	4.897	∞	input port clock		
Path 3	∞	10	11	7	A[0]	S[14]	9.499	4.602	4.897	∞	input port clock		
Path 4	∞	10	11	7	A[0]	S[15]	9.499	4.602	4.897	∞	input port clock		
Path 5	∞	9	10	7	A[0]	S[10]	8.908	4.484	4.424	∞	input port clock		
Path 6	∞	9	10	7	A[0]	S[12]	8.908	4.484	4.424	∞	input port clock		
Path 7	∞	9	10	7	A[0]	S[9]	8.908	4.484	4.424	∞	input port clock		
Path 8	∞	8	9	7	A[0]	S[11]	8.307	4.360	3.947	∞	input port clock		
Path 9	∞	7	8	7	A[0]	S[5]	7.729	4.242	3.487	∞	input port clock		
Path 10	∞	7	8	7	A[0]	S[6]	7.729	4.242	3.487	∞	input port clock		

Από τα παραπάνω προκύπτει ότι το critical path είναι εκείνο από την είσοδο (A[0]) έως την έξοδο (S[13]) με συνολική καθυστέρηση 9.505 ns.

Ζήτημα 6: Συστολικός Πολλαπλασιαστής Διάδοσης Κρατουμένου 4 bits

Στο συγκεκριμένο θέμα, καλούμαστε να υλοποιήσουμε την περιγραφή της οντότητας του συστολικού πολλαπλασιαστή διάδοσης κρατουμένων (SYS_MUL) 4 bits. Για την υλοποίηση αυτή βασιστήκαμε στο παρακάτω θεωρητικό σχήμα.



Έχουμε λοιπόν αρχικά την περιγραφή της οντότητας κάθε τετραγώνου που είναι ουσιαστικά το σχήμα πάνω αριστερά (ένας full adder με μία πύλη and). Το στοιχειώδες αυτό component το ονομάσαμε MUL_CELL και ο κώδικάς του δίνεται παρακάτω. Επίσης, για την υλοποίηση του pipeline βασιστήκαμε στους χρονισμούς που υπάρχουν πάνω στο σχήμα, βάζοντας καταχωρητές ουσιαστικά στις διαγώνιες περιοχές που ορίζουν οι διαγώνιες γραμμές. Πιο συγκεκριμένα, κάθε καταχωρητής Pipeline «γεμίζει» με όσα bits πληροφορίας θα χρειαστούν στα επόμενα επίπεδα. Η λογική αυτή, όπως θα φανεί και αργότερα στην RTL περιγραφή, όχι μόνο υλοποιεί ορθά την τεχνική το pipeline αλλά ρίχνει ικανοποιητικά το resource utilization κάνοντας την υλοποίησή μας πολύ αποδοτική. Ακολουθούν λοιπόν οι κώδικες, το σχηματικό και τα αντίστοιχα reports:

Code of Mul_Cell

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


entity MUL_CELL is
  port (
    clk: in std_logic;
    a: in std_logic;
    b: in std_logic;
    cin: in std_logic;
```

```
    sin: in std_logic;
    cout: out std_logic;
    sout: out std_logic
  ) ;
end MUL_CELL;

architecture Structural of MUL_CELL is
    component FA_seq_beh is
        port(
            A: in std_logic;
            B: in std_logic;
            C_in: in std_logic;
            CLK: in std_logic;
            reset: in std_logic;
            C_out: out std_logic;
            SUM: out std_logic
        );
    end component;

    signal tmp: std_logic;

begin
    tmp <= a and b;
    fa: FA_seq_beh
            port map(
                A => tmp,
                B => sin,
                C_in => cin,
                CLK => clk,
                reset => '0',
                C_out => cout,
                SUM => sout
            );
end Structural ; -- Structural
```

Code of SYS_MUL

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

```vhdl
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity SYS_MUL is
  port (
    A: in STD_LOGIC_VECTOR(3 downto 0);
    B: in STD_LOGIC_VECTOR(3 downto 0);
    clk: in STD_LOGIC;
    P: out STD_LOGIC_VECTOR(7 downto 0)
  );
end SYS_MUL;


architecture Structural of SYS_MUL is
    component MUL_CELL is
        port(
            clk: in std_logic;
            a: in std_logic;
            b: in std_logic;
            cin: in std_logic;
            sin: in std_logic;
            cout: out std_logic;
            sout: out std_logic
        );
    end component;
    -- Pipeline Registers
    --6 bits
    signal r9: std_logic_vector(5 downto 0);
    --7 bits
    signal r7: std_logic_vector(6 downto 0);
    --8 bits
    signal r0,r8: std_logic_vector(7 downto 0);
    --9 bits
    signal r1,r2,r3,r5,r6: std_logic_vector(8 downto 0);
    --10 bits
    signal r4: std_logic_vector(9 downto 0);

    --In-Between Carrys/Sums/Products
    signal c00,c01,c02,c03,c10,c11,c12,c13,c20,c21,c22,c23,c30,c31,c32,
           s01,s02,s03,s11,s12,s13,s21,s22,s23,
           p0,p1,p2,p3,p4,p5,p6,p7: std_logic;
begin
```

```
mc00: MUL_CELL
        port map(clk => clk, a => A(0), b => B(0), cin => '0', sin => '0', cout => c00, sout => p0);
mc01: MUL_CELL
        port map(clk => clk, a => r0(1), b => r0(4), cin => c00, sin => '0', cout => c01, sout => s01);
mc02: MUL_CELL
        port map(clk => clk, a => r1(2), b => r1(4), cin => c01, sin => '0', cout => c02, sout => s02);
mc03: MUL_CELL
        port map(clk => clk, a => r2(3), b => r2(4), cin => c02, sin => '0', cout => c03, sout => s03);
mc10: MUL_CELL
        port map(clk => clk, a => r1(0), b => r1(5), cin => '0', sin => s01, cout => c10, sout => p1);
mc11: MUL_CELL
        port map(clk => clk, a => r2(1), b => r2(5), cin => c10, sin => s02, cout => c11, sout => s11);
mc12: MUL_CELL
        port map(clk => clk, a => r3(2), b => r3(4), cin => c11, sin => s03, cout => c12, sout => s12);
mc13: MUL_CELL
        port map(clk => clk, a => r4(3), b => r4(4), cin => c12, sin => r4(9), cout => c13, sout => s13);
mc20: MUL_CELL
        port map(clk => clk, a => r3(0), b => r3(5), cin => '0', sin => s11, cout => c20, sout => p2);
mc21: MUL_CELL
        port map(clk => clk, a => r4(1), b => r4(5), cin => c20, sin => s12, cout => c21, sout => s21);
mc22: MUL_CELL
        port map(clk => clk, a => r5(2), b => r5(4), cin => c21, sin => s13, cout => c22, sout => s22);
mc23: MUL_CELL
        port map(clk => clk, a => r6(2), b => r6(3), cin => c22, sin => r6(8), cout => c23, sout => s23);
mc30: MUL_CELL
        port map(clk => clk, a => r5(0), b => r5(5), cin => '0', sin => s21, cout => c30, sout => p3);
mc31: MUL_CELL
        port map(clk => clk, a => r6(0), b => r6(4), cin => c30, sin => s22, cout => c31, sout => p4);
mc32: MUL_CELL
        port map(clk => clk, a => r7(0), b => r7(2), cin => c31, sin => s23, cout => c32, sout => p5);
mc33: MUL_CELL
        port map(clk => clk, a => r8(0), b => r8(1), cin => c32, sin => r8(7), cout => p7, sout => p6);
  next_clk : process( clk )
  begin
    if rising_edge(clk) then
      r9 <= p5 & r8(6 downto 2); --6bits                      r9(p0,p1,p2,p3,p4,p5)
      r8 <= c23 & p4 & r7(6 downto 1); --8bits                r8(a3,b3,p0,p1,p2,p3,p4,c23)
      r7 <= p3 & r6(7 downto 4) & r6(2 downto 1); --7bits     r7(a2,a3,b3,p0,p1,p2,p3)
      r6 <= c13 & r5(8 downto 1); --9bits                     r6(a1,a2,a3,b2,b3,p0,p1,p2,c13)
      r5 <= p2 & r4(8 downto 5) & r4(3 downto 0); --9bits     r5(a0,a1,a2,a3,b2,b3,p0,p1,p2)
```

```
      r4 <= c03 & r3(8 downto 0); --10bits                    r4(a0,a1,a2,a3,b1,b2,b3,p0,p1,c03)
      r3 <= p1 & r2(8 downto 5) & r2(3 downto 0); --9bits     r3(a0,a1,a2,a3,b1,b2,b3,p0,p1)
      r2 <= r1(8 downto 0); --9bits                           r2(a0,a1,a2,a3,b0,b1,b2,b3,p0)
      r1 <= p0 & r0(7 downto 0); --9bits                      r1(a0,a1,a2,a3,b0,b1,b2,b3,p0)
      r0 <= B(3 downto 0) & A(3 downto 0) ; --8bits           r0(a0,a1,a2,a3,b0,b1,b2,b3)
    end if;
  end process ; -- next_clk
  P <= p7 & p6 & r9(5 downto 0);
end Structural ; -- Structural
```

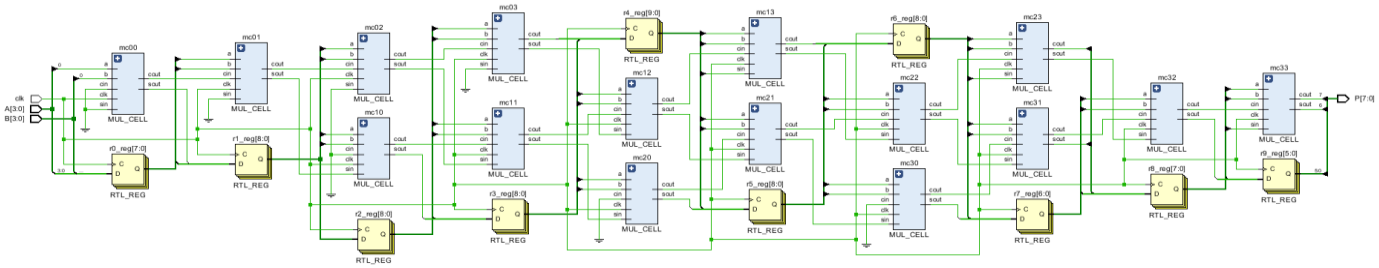
Test Bench of SYS_MUL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity SYS_MUL_tb is
end SYS_MUL_tb;

architecture test_bench of SYS_MUL_tb is
    component SYS_MUL
        port (
                A: in std_logic_vector (3 downto 0);
                B: in std_logic_vector (3 downto 0);
                clk: in std_logic;
                P: out std_logic_vector (7 downto 0)
                );
    end component;
--Input Signals
    signal A_tb, B_tb: std_logic_vector (3 downto 0):= (others=>'0');
    signal clk: std_logic:='0';
--Output Signals
    signal P_tb: std_logic_vector(7 downto 0);
--Clock
    constant CLKP : time := 10ps;
begin
UUT: SYS_MUL port map(
                    A => A_tb,
                    B => B_tb,
                    clk => clk,
                    P => P_tb (7 downto 0)
```

```
                    );


clk_proc: process
    begin
        clk <= '0';
        wait for CLKP/2;
        clk <= '1';
        wait for CLKP/2;
    end process;
tests: process
begin


        A_tb<="0001";   --1
        B_tb<="0011";   --3
        wait for CLKP;  --3
        A_tb<="0010";   --2
        B_tb<="0100";   --4
        wait for CLKP;  --8
        A_tb<="0010";   --2
        B_tb<="1111";   --15
        wait for CLKP;  --30
        A_tb<="0000";
        B_tb<="0001";
        wait for CLKP;
        A_tb<="1111";
        B_tb<="1111";
        wait for CLKP;
        A_tb<="0001";
        B_tb<="0001";
        wait for CLKP;
        A_tb<="0100"; --4
        B_tb<="1011"; --11
        wait for CLKP;
        A_tb<="0100"; --4
        B_tb<="1001"; --9
        wait for CLKP;
        A_tb<="0000";
        B_tb<="1011";
        wait for CLKP;
        A_tb<="0011"; --3
```

```
        B_tb<="1000"; --8
        wait for CLKP;
        A_tb<="0100";
        B_tb<="0001";
        wait for CLKP;
        A_tb<="0000";
        B_tb<="1011";
        wait;

end process;

end test_bench;
```

RTL SCHEMATIC:



RESOURCE UTILIZATION:

Name	Slice LUTs (17600)	Slice Registers (35200)	Bonded IOB (100)	BUFGCTRL (32)
N SYS_MUL	23	79	17	1
mc01 (MUL_CELL)	1	1	0	0
mc02 (MUL_CELL_0)	1	1	0	0
mc03 (MUL_CELL_1)	1	1	0	0
mc10 (MUL_CELL_2)	1	1	0	0
mc11 (MUL_CELL_3)	1	2	0	0
mc12 (MUL_CELL_4)	2	2	0	0
mc13 (MUL_CELL_5)	0	2	0	0
mc20 (MUL_CELL_6)	1	1	0	0
mc21 (MUL_CELL_7)	1	2	0	0
mc22 (MUL_CELL_8)	1	2	0	0
mc23 (MUL_CELL_9)	1	2	0	0
mc30 (MUL_CELL_10)	1	1	0	0
mc31 (MUL_CELL_11)	1	1	0	0
mc32 (MUL_CELL_12)	1	2	0	0
mc33 (MUL_CELL_13)	1	2	0	0

TIMING REPORT:

Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	2	2	1	r9_reg[5]/C	P[5]	4.076	3.276	0.800	∞			
Path 2	∞	2	2	1	mc33/fa/SUM_reg/C	P[6]	4.076	3.276	0.800	∞			
Path 3	∞	2	2	1	mc33/fa/C_out_reg/C	P[7]	4.076	3.276	0.800	∞			
Path 4	∞	2	2	1	r9_reg[0]/C	P[0]	4.058	3.258	0.800	∞			
Path 5	∞	2	2	1	r9_reg[1]/C	P[1]	4.058	3.258	0.800	∞			
Path 6	∞	2	2	1	r9_reg[2]/C	P[2]	4.058	3.258	0.800	∞			
Path 7	∞	2	2	1	r9_reg[3]/C	P[3]	4.058	3.258	0.800	∞			
Path 8	∞	2	2	1	r9_reg[4]/C	P[4]	4.058	3.258	0.800	∞			
Path 9	∞	2	3	2	B[0]	r8_reg[2]_srl9/D	2.239	1.106	1.133	∞	input port clock		
Path 10	∞	2	2	3	r1_reg[5]/C	r8_reg[3]_srl7/D	1.843	0.751	1.092	∞			

Από τα παραπάνω προκύπτει ότι τα critical paths είναι

- από την είσοδο του τελικού καταχωρητή (r9_reg[5]) έως την έξοδο (P[5])
- από το τελικό mul_cell (m33) και το SUM_reg αυτού μέχρι την έξοδο (P[6])
- από το τελικό mul_cell (m33) και το C_out_reg αυτού μέχρι την έξοδο (P[7])

με συνολική καθυστέρηση 4.076 ns το καθένα.

Ερωτήσεις Θεωρίας

1) Η διαφορά μεταξύ σύγχρονων και ασύγχρονων σημάτων ελέγχου σε ένα ακολουθιακό κύκλωμα είναι η εξής: τα πρώτα, επιβάλλουν τον υπολογισμό των σημάτων εξόδου στην θετική/αρνητική ακμή του ρολογιού ενώ στα δεύτερα ο υπολογισμός αυτός γίνεται άμεσα.

2) Η διαδικασία της σύνθεσης λαμβάνει ως είσοδο ένα αρχείο σε VHDL το οποίο περιγράφει το κύκλωμά μας, τους περιορισμούς που θέτει ο σχεδιαστής καθώς και τα διαθέσιμα components του FPGA. Το αποτέλεσμα της σύνθεσης είναι το netlist, η λίστα διασυνδέσεων του κυκλώματος σε επίπεδο λογικών πυλών. Επομένως είναι σημαντικό, να έχουμε εκ των προτέρων ελέγξει την ορθή λειτουργία του κυκλώματος μας με κάποιο testbench σε RT επίπεδο. Εκεί είναι ευκολότερη η ανίχνευση λογικών λαθών πριν "σπαταλήσουμε" πόρους της σύνθεσης

3) Η βασική διαφορά ενός testbench για ένα ακολουθιακό και ένα συνδυαστικό κύκλωμα είναι η ύπαρξη ή μη (αντίστοιχα) του ρολογιού. Πιο συγκεκριμένα, όπως ειπώθηκε και στην πρώτη ερώτηση, τα ακολουθιακά κυκλώματα σημαίνουν χρήση ρολογιού το οποίο θα πρέπει κ εκείνο να προσομοιωθεί. Επομένως στα testbenches που προορίζονται για ένα ακολουθιακό κύκλωμα είναι απαραίτητη η ύπαρξη και ενός process το οποίο να ορίζει την συμπεριφορά του ρολογιού. Στα testbenches που προορίζονται για συνδυαστικά κυκλώματα από την άλλη, εκείνο που έχουμε να κάνουμε είναι να προσθέσουμε μία μικρή καθυστέρηση μέσω ενός wait για την παραγωγή του αποτελέσματος μεταξύ διαφορετικών εισόδων.

4) Όσον αφορά στα αποτελέσματα για τα resources και το timing των RCAs που υλοποιήθηκαν στο ερώτημα 3, έχουμε να επισημάνουμε τα εξής:

- Στο συνδυαστικό κύκλωμα, παρατηρούμε μεγαλύτερο delay στο critical path (5.970ns) και λιγότερο utilization στα resources. Αυτό είναι λογικό εφόσον αφενός για τον καθορισμό ενός path (διαδρομή μεταξύ δύο σημείων όπου το αποτέλεσμα μπορεί να αποθηκευτεί) σε ένα συνδυαστικό κύκλωμα πρέπει να «μετρήσουμε» από μία είσοδο έως μία έξοδο και αφετέρου δεν έχουμε την χρήση επιπλέον υλικών (όπως registers) αλλά μόνο τα απαραίτητα components (full adders).
- Στο ακολουθιακό κύκλωμα από την άλλη, παρατηρούμε μικρότερο delay στο critical path (4.076ns) και περισσότερο utilization στα resources. Αντίστοιχα με πριν τα αποτελέσματα αυτά είναι αναμενόμενα καθώς το critical path μπορεί να βρεθεί είτε μεταξύ εισόδου –

ενδιάμεσου καταχωρητή, ενδιάμεσου καταχωρητή – εξόδου(όπως στην δική μας περίπτωση), ή και μεταξύ δύο ενδιάμεσων καταχωρητών (μικρότερη απόσταση από το είσοδος – έξοδος). Επίσης η χρήση των καταχωρητών του pipeline αυξάνει προφανώς την χρησιμοποίηση του υλικού.

Επομένως οι δύο αυτές υλοποιήσεις είναι ένα καλό παράδειγμα του trade-off (γρηγορότερο ή πιο οικονομικό από άποψη υλικού κύκλωμα;) το οποίο καλείται να λάβει υπόψη ο σχεδιαστής, ανάλογα με την εφαρμογή που θέλει να αναπτύξει.