Elettronica dei Sistemi Digitali Lab 03

Data-Path Elements



Gruppo: A11

15999
44728
15478

 $8~\mathrm{aprile}~2020$

Indice

1	8-bi	it Sequential RCA	2
	1.1	Spiegazione teorica	2
	1.2	Procedimento	2
	1.3	Risultati	4
	1.4	Appendice	5
2	8-bi	it Sequential Adder/Subtractor	18
	2.1	Spiegazione teorica	18
	2.2	Procedimento	18
	2.3	Risultati	19
	2.4	Appendice	20
3			
3	16-l	bit RCA, Carry-Bypass Adder and Carry-Select	
3	16-l Add	bit RCA, Carry-Bypass Adder and Carry-Select der	29
3		der	
3	\mathbf{Add}	der Spiegazione teorica	29 29
3	Ad o 3.1	der Spiegazione teorica	29 29
3	3.1 3.2	der Spiegazione teorica	29 29 29 30
3	3.1 3.2 3.3 3.4	der Spiegazione teorica	29 29 29 30 34
	3.1 3.2 3.3 3.4	der Spiegazione teorica	29 29 30 34 47
	3.1 3.2 3.3 3.4 Mu	Spiegazione teorica	29 29 30 34 47 47
	3.1 3.2 3.3 3.4 Mu 4.1	der Spiegazione teorica	29 29 30 34 47 47 47

Capitolo 1

8-bit Sequential RCA

1.1 Spiegazione teorica

Ci si propone di realizzare un ripple-carry adder "sequenziale", ovvero un RCA i cui ingressi provengono da due registri di parallelismo 8 bit e le cui uscite vengono memorizzate su elementi di memoria.

1.2 Procedimento

Si costruisce un RCA mediante connessione di 8 full-adder, come mostrato in **figura 1.1**, optando per una implementazione tramite generics al fine di garantirne la riutilizzabilità nei successivi passi dell'esercitazione. Si è scelto, inoltre, di separare il componente dall'interfaccia tramite display, la cui tavola di verità è riportata in **figura 1.2**.

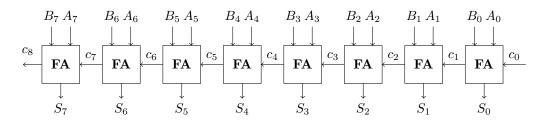


Figura 1.1: 8-bit full adder

SW(3)	SW(2)	SW(1)	SW(0)	a	b	c	d	e	f	g	HEX
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0	2
0	0	1	1	0	0	0	0	1	1	0	3
0	1	0	0	1	0	0	1	1	0	0	4
0	1	0	1	0	1	0	0	1	0	0	5
0	1	1	0	0	1	0	0	0	0	0	6
0	1	1	1	0	0	0	1	1	1	1	7
1	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	0	1	0	0	9
1	0	1	0	0	0	0	1	0	0	0	A
1	0	1	1	1	1	0	0	0	0	0	b
1	1	0	0	0	1	1	0	0	0	1	С
1	1	0	1	1	0	0	0	0	1	0	d
1	1	1	0	0	1	1	0	0	0	0	E
1	1	1	1	0	1	1	1	0	0	0	F

Figura 1.2: hex display truth table

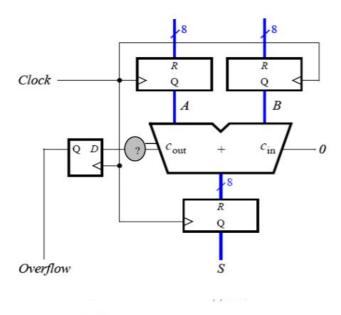


Figura 1.3: 8-bit registered-adder

1.3 Risultati

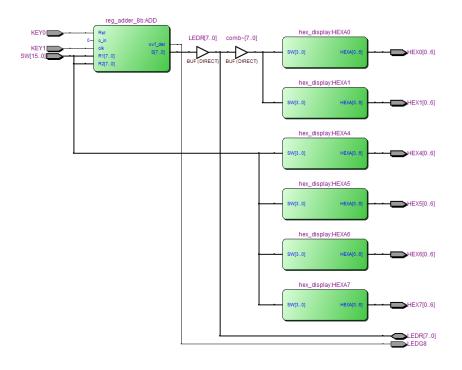


Figura 1.4: RTL viewer exercise no. 1

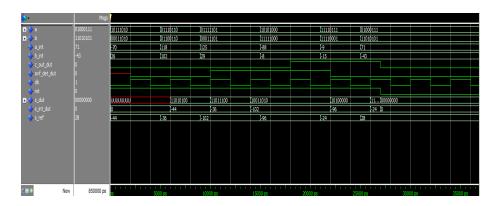
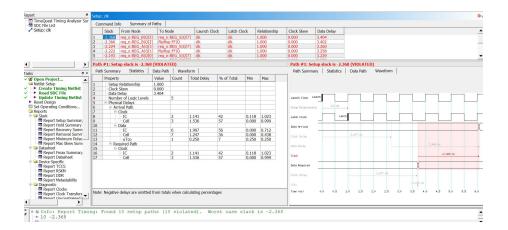


Figura 1.5: wave exercise no. 1

Di seguito, infine, è riportata un'analisi temporale dell'andamento



laddove si evincono

$$f_{max} = 286.9 \,\mathrm{MHz}$$

$$t_{del} = 3.404 \,\mathrm{ns}$$

1.4 Appendice

```
library ieee;
1
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;
5
    entity reg_adder_8b is
6
      generic (n : integer := 8);
7
8
      port ( R1,R2 : in signed ( n-1 downto 0 );
9
            c_in : in std_logic;
10
11
            clk,Rst : in std_logic;
12
            c_out : out std_logic;
            ovf_det : out std_logic;
13
            S : out signed ( n-1 downto 0 )
14
15
          );
16
17
    end reg_adder_8b;
18
19
    architecture struct of reg_adder_8b is
20
21
     component flipflop
       port (D, Clock, Resetn : in std_logic;
22
             Q : out std_logic
23
```

```
24
           );
25
      end component;
26
27
      component reg_n
28
29
        generic ( N : integer:=n);
30
31
       port (R : in signed(N-1 downto 0);
32
           Clock, Resetn : in std_logic;
           Q : out signed(N-1 downto 0)
33
34
           );
      end component;
35
36
37
38
39
        component RCA
40
41
        generic (n : integer := n);
42
        port ( A,B : in signed ( n-1 downto 0 );
43
44
              c_in : in std_logic;
              c_out : out std_logic;
45
46
              ovf_det : out std_logic;
             S : out signed ( n-1 downto 0 )
47
48
             );
        end component;
49
50
51
        signal A,B,Q : signed ( n-1 downto 0 );
52
        signal ovf : std_logic;
53
54
    begin
55
56
      REG_A: reg_n port map ( R1,clk,Rst,A);
57
      REG_B: reg_n port map ( R2,clk,Rst,B);
58
      ADD: RCA port map ( A,B,c_in,c_out,ovf,Q );
59
60
61
      FF: flipflop port map ( ovf,clk,Rst,ovf_det );
62
63
      REG_S: reg_n port map (Q,clk,Rst,S);
64
65
    end architecture;
```

Code 1.1: reg_adder_8b.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

```
entity reg_adder_8b_tb is
    end reg_adder_8b_tb;
8
9
10
    architecture struct of reg_adder_8b_tb is
11
12
      component reg_adder_8b
       generic (n : integer := 8);
13
14
       port ( R1,R2 : in signed ( n-1 downto 0 );
15
            c_in : in std_logic;
16
            clk,Rst : in std_logic;
17
18
            c_out : out std_logic;
            ovf_det : out std_logic;
19
20
            S : out signed ( n-1 downto 0 )
21
          );
22
23
      end component;
24
25
     signal A,B : signed ( 7 downto 0 );
26
     signal A_int,B_int : integer;
27
     signal c_out_dut,ovf_det_dut,clk,Rst : std_logic;
      signal S_dut : signed ( 7 downto 0 );
28
29
     signal S_int_dut,S_ref : integer;
30
31
32 begin
33
34
     process begin
35
       for i in 1 to 10 loop
36
         clk <= '0';
37
         wait for 2 ns;
38
         clk <= '1';
         wait for 2 ns;
39
40
       end loop;
41
     end process;
42
43
      Rst <= '1','0' after 27 ns;</pre>
44
      A <= "10111010", "01110110" after 5 ns, "011111101" after 9 ns,
45
          "10101000" after 15 ns,
46
          "11110111" after 21 ns, "01000111" after 25 ns;
47
     B \le "00011010", "01100110" after 5 ns, "00011101" after 9 ns,
48
          "11111000" after 15 ns,
          "11110001" after 21 ns, "11010101" after 25 ns;
49
50
51
     A_int <= to_integer(A);
      B_int <= to_integer(B);</pre>
52
```

```
53
54
55
ADD: reg_adder_8b port map (A, B, '0', clk, Rst, c_out_dut, ovf_det_dut, S_dut);
56
57
S_int_dut <= to_integer(S_dut);
58
S_ref <= to_integer(A + B);
59
60 end architecture;
```

Code 1.2: reg_adder_tb.vhd

```
1 library ieee;
   use ieee.std_logic_1164.all;
3
   use ieee.numeric_std.all;
4
    entity adder_display is
     --KEY -> clk, KEYO -> RST, SW(15->8) -> A,
7
     --SW(7->0) ->B,LEDR ->S,LEDG8 -> ovf_det,
8
     --HEX7_6 -> A display, HEX5_4 -> B dspaly, HEX1_0 S_display
9
     port (
10
         SW : in signed ( 15 downto 0 );
11
12
         KEY1,KEY0 : in std_logic;
         LEDG8 : out std_logic;
13
14
         LEDR : buffer signed ( 7 downto 0 );
15
         HEX7,HEX6,HEX5,HEX4,HEX1,HEX0 : out std_logic_vector(0 to 6)
16
          );
17
    end adder_display;
18
19
    architecture struct of adder_display is
20
21
     component reg_adder_8b is
22
       generic (n : integer := 8);
23
       port ( R1,R2 : in signed ( n-1 downto 0 );
24
25
              c_in : in std_logic;
26
              clk,Rst : in std_logic;
27
              c_out : out std_logic;
28
              ovf_det : out std_logic;
29
              S : out signed ( n-1 downto 0 )
30
            );
31
32
     end component;
33
34
     component hex_display
35
       port(
         SW : in signed (3 downto 0);
36
37
         HEXA : out std_logic_vector ( 0 to 6)
```

```
38
         );
     end component;
39
40
41
42
   begin
43
44
     ADD: reg_adder_8b port map(R1 => SW(15 downto 8),R2 => SW(7 downto 0),
                               c_in => '0',clk => KEY1,Rst => KEY0,
45
                                ovf_det => LEDG8,S => LEDR);
46
47
     --A display
48
     HEXA7: hex_display port map(SW(15 downto 12), HEX7);
49
     HEXA6: hex_display port map(SW(11 downto 8),HEX6);
50
51
      --B display
52
53
     HEXA5: hex_display port map(SW(7 downto 4),HEX5);
     HEXA4: hex_display port map(SW(3 downto 0), HEX4);
54
55
56
     --S display
57
     HEXA1: hex_display port map(LEDR(7 downto 4), HEX1);
     HEXAO: hex_display port map(LEDR(3 downto 0), HEXO);
58
59
60
61
   end architecture;
```

Code 1.3: adder_display.vhd

```
1 library ieee;
   use ieee.std_logic_1164.all;
   use ieee.numeric_std.all;
5
    entity adder_display_tb is
    end adder_display_tb;
6
8
    architecture struct of adder_display_tb is
9
10
11
     component adder_display
12
       port (
13
          SW : in signed ( 15 downto 0 );
          KEY1,KEY0 : in std_logic;
14
15
          LEDG8 : out std_logic;
          LEDR : buffer signed ( 7 downto 0 );
16
17
          HEX7,HEX6,HEX5,HEX4,HEX1,HEX0 : out std_logic_vector(0 to 6)
18
          );
19
     end component;
20
21
     signal A : signed ( 7 downto 0 );
22
     signal B : signed ( 7 downto 0 );
```

```
23
      signal A_int,B_int : integer;
24
     signal A_B : signed (15 downto 0);
25
      signal ovf_det_dut,clk,Rst : std_logic;
26
      signal S_dut : signed ( 7 downto 0 );
27
      signal S_int_dut,S_ref : integer;
28
      signal HEX7_dut,HEX6_dut,HEX5_dut,HEX4_dut,HEX1_dut,HEX0_dut :
          std_logic_vector(0 to 6);
29
30
31
    begin
32
33
     process begin
34
       for i in 1 to 10 loop
35
         clk <= '0';
         wait for 2 ns;
36
37
         clk <= '1';
38
         wait for 2 ns;
39
       end loop;
40
      end process;
41
42
      Rst <= '1','0' after 27 ns;</pre>
43
      A <= "10111010", "01110110" after 5 ns, "011111101" after 9 ns,
44
          "10101000" after 15 ns,
          "11110111" after 21 ns, "01000111" after 25 ns;
45
46
      B <= "00011010", "01100110" after 5 ns, "00011101" after 9 ns,
47
          "11111000" after 15 ns,
48
          "11110001" after 21 ns, "11010101" after 25 ns;
49
50
      A_int <= to_integer(A);
51
      B_int <= to_integer(B);</pre>
52
     A_B <= A & B;
53
54
      ADD: adder_display port map (A_B, clk, Rst, ovf_det_dut, S_dut,
55
                                  HEX7_dut,HEX6_dut,HEX5_dut,HEX4_dut,HEX1_dut,HEX0_dut);
56
57
     S_int_dut <= to_integer(S_dut);</pre>
58
     S_ref <= to_integer(A + B) ;</pre>
59
    end architecture;
```

Code 1.4: adder_display_tb.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity RCA is
```

```
6
      generic (n : integer := 8);
 7
      port ( A,B : in signed ( n-1 downto 0 );
 8
             c_in : in std_logic;
 9
             c_out : out std_logic;
10
            ovf_det : out std_logic;
11
            S : out signed ( n-1 downto 0 )
12
13
    end RCA;
14
15
    architecture struct of RCA is
16
17
      component FA
18
19
       port( a,b : in std_logic;
20
             c_in : in std_logic;
21
             s,c_out: out std_logic
22
           );
23
      end component;
24
25
      signal carries : signed ( n downto 0);
26
27
      begin
28
      carries(0) <= c_in;</pre>
29
      GEN : for i in 0 to n-1 generate
30
       FULL_ADDERS: FA port map( A(i), B(i), carries(i), S(i),
31
            carries(i+1));
32
      end generate;
33
34
      c_out <= carries(n);</pre>
35
      ovf_det <= carries(n-1) xor carries(n);</pre>
    end architecture;
```

Code 1.5: RCA.vhd

```
library ieee;
    use ieee.std_logic_1164.all;
3
    use ieee.numeric_std.all;
5
    entity RCA_tb is
    end RCA_tb;
6
7
8
    architecture struct of RCA_tb is
10
11
      component RCA
12
        generic (n : integer := 8);
13
        port ( A,B : in signed ( n-1 downto 0 );
14
```

```
15
             c_{in} : in std_logic; --sub when add_sub == 1
16
             c_out : out std_logic;
17
             ovf_det : out std_logic;
18
            S : out signed ( n-1 downto 0 )
19
           );
20
      end component;
21
22
      signal A : signed ( 7 downto 0 );
      signal B : signed ( 7 downto 0 );
23
      signal A_int,B_int : integer;
24
      signal c_out_dut,ovf_det_dut : std_logic;
25
      signal S_dut : signed ( 7 downto 0 );
26
27
      signal S_ref,S_int_dut : integer;
28
29
30 begin
31
      process begin
32
       -- generating signed numbers for testcases
33
          for i in 2 to 4 loop
34
35
            A <= to_signed(63*i, A'length);
36
37
           for j in 0 to 2 loop
             B <= to_signed(61*j, B'length);</pre>
38
             wait for 5 ns;
39
           end loop;
40
41
42
          end loop;
43
        end process;
44
45
46
      A_int <= to_integer(A);
47
      B_int <= to_integer(B);</pre>
48
49
      ADD: RCA port map (A,B,'0',c_out_dut,ovf_det_dut,S_dut);
50
51
      S_int_dut <= to_integer(S_dut);</pre>
52
      S_ref <= to_integer(A + B) ;</pre>
53
54
    end architecture;
```

Code 1.6: RCA_tb.vhd

```
s,c_out: out std_logic
          );
 8
 9
    end FA;
10
11
12 architecture struct of FA is
13
       signal p : std_logic;
14
15
       begin
16
17
        p <= a xor b;</pre>
        s <= c_in xor p;
18
        c_out <= b when (p='0') else c_in;</pre>
19
20
21
    end architecture;
```

Code 1.7: FA.vhd

```
1 library ieee;
   use ieee.std_logic_1164.all;
4
5
    entity FA_tb is
6
    end FA_tb;
7
8
9
    architecture struct of FA_tb is
10
11
    component FA
     port( a,b : in std_logic;
12
          c_in : in std_logic;
13
           s,c_out: out std_logic
14
15
         );
     end component;
16
17
18
     signal input_string : std_logic_vector(0 to 2);
19
     signal s_dut,c_out_dut : std_logic;
20
21
    begin
22
      input_string <= "000","001" after 5 ns,"010" after 10 ns,
23
                     "011" after 15 ns, "100" after 20 ns,
24
25
                    "101" after 25 ns, "110" after 30 ns, "111" after 35 ns;
26
      FULL_ADDER: FA port map(a => input_string(1),b => input_string(0),
27
                    c_in => input_string(2),s => s_dut,c_out => c_out_dut);
28
29
30
    end architecture;
```

Code 1.8: FA_tb.vhd

```
1 library ieee;
   use ieee.std_logic_1164.all;
    entity flipflop is
5
     port (
6
       D, Clock, Resetn : in std_logic;
7
       Q : out std_logic
     );
9
10
    end flipflop;
11
12
    architecture Behavior of flipflop is
13
    begin
15
16
     process (Clock, Resetn)
17
      begin
        if (Resetn = '0') then -- asynchronous clear
18
         Q <= '0';
19
20
       elsif (Clock'EVENT and Clock = '1') then
21
         Q <= D;
22
       end if;
23
      end process;
24
25
    end Behavior;
```

Code 1.9: flipflop.vhd

```
1 library ieee;
    use ieee.std_logic_1164.all;
3
    use ieee.numeric_std.all;
5
6
    entity reg_n is
7
     generic ( N : integer:=8);
8
     port (
       R : in signed(N-1 downto 0);
9
10
       Clock, Resetn : in std_logic;
       Q : out signed(N-1 downto 0));
11
12
    end reg_n;
13
14
15
    architecture Behavior of reg_n is
16
    begin
17
18
     process (Clock, Resetn)
19
      begin
        if (Resetn = '0') then
20
```

```
21     Q <= (others => '0');
22     elsif (Clock'EVENT AND Clock = '1') then
23     Q <= R;
24     end if;
25     end process;
26
27     end Behavior;</pre>
```

Code 1.10: reg_n.vhd

```
library ieee;
1
    use ieee.std_logic_1164.all;
3
    use ieee.numeric_std.all;
4
5
6
    entity reg_n_tb is
    end reg_n_tb;
9
10
   architecture Behavior of reg_n_tb is
11
      component reg_n is
12
       generic ( N : integer:=3);
13
14
15
       port (R : in signed(N-1 downto 0);
16
           Clock, Resetn : in std_logic;
           Q : out signed(N-1 downto 0)
17
18
           );
19
      end component;
20
21
22
       signal clk : std_logic;
       signal Rst : std_logic;
23
        signal input_string : signed(2 downto 0);
24
       signal out_dut : signed(2 downto 0);
25
26
27
28
    begin
29
30
     process begin
31
       for i in 1 to 10 loop
         clk <= '0';
32
33
         wait for 1 ns;
34
         clk <= '1';
35
         wait for 1 ns;
36
       end loop;
37
      end process;
38
39
      Rst <= '1','0' after 13 ns;</pre>
```

```
40
41 input_string <= "000", "001" after 4 ns,"010" after 8 ns,
42 "011" after 9 ns, "100" after 11 ns;
43
44 REGN: reg_n port map(input_string,clk,Rst,out_dut);
45
46 end Behavior;
```

Code 1.11: reg_n_tb.vhd

```
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;
    entity hex_display is
5
6
 7
      port( SW : in signed (3 downto 0);
           HEXA : out std_logic_vector ( 0 to 6)
8
9
         );
10
11
    end hex_display;
12
13
14
    architecture behaviour of hex_display is
15
16
    begin
17
18
     with SW select
19
       HEXA <=
20
         "0000001" when "0000",
         "1001111" when "0001",
21
22
         "0010010" when "0010",
23
         "0000110" when "0011",
24
         "1001100" when "0100",
25
         "0100100" when "0101",
26
         "0100000" when "0110",
         "0001111" when "0111",
27
         "0000000" when "1000",
28
         "0000100" when "1001",
29
30
         "0001000" when "1010", -- A
31
         "1100000" when "1011", -- b
         "0110001" when "1100", -- C
32
33
         "1000010" when "1101", -- d
         "0110000" when "1110", -- E
34
         "0111000" when "1111", -- F
35
         "1111111" when others; --black for undefined values
36
37
    end architecture;
```

Code 1.12: hex_display.vhd

```
1 library ieee;
   use ieee.std_logic_1164.all;
   use ieee.numeric_std.all;
    entity hex_display_tb is
    end hex_display_tb;
6
8
9
    architecture behaviour of hex_display_tb is
10
      component hex_display
11
12
       port(
         SW : in signed (3 downto 0);
13
         HEXA : out std_logic_vector ( 0 to 6)
14
15
         );
      end component;
16
17
      signal n_in : signed (3 downto 0);
18
19
      signal out_dut : std_logic_vector (0 to 6);
20
21
     n_in <= "0000", "0001" after 5 ns, "0010" after 10 ns, "0011" after 15
22
          ns,
                      "0100" after 20 ns, "0101" after 25 ns, "0110" after
23
                          30 ns,
                      "0111" after 35 ns, "1000" after 40 ns, "1001" after
24
                          45 ns,
                      "1010" after 50 ns, "1011" after 55 ns, "1100" after
25
26
                      "1101" after 65 ns, "1110" after 70 ns, "1111" after
                          75 ns;
27
      DISPLAY: hex_display port map (n_in,out_dut);
28
29
    end architecture;
```

Code 1.13: hex_display_tb.vhd

Capitolo 2

8-bit Sequential Adder/Subtractor

2.1 Spiegazione teorica

Si modifica il precedente RCA in modo che esso permetta di svolgere anche la sottrazione.

2.2 Procedimento

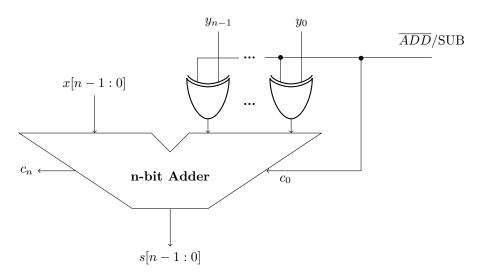


Figura 2.1: 2's n-bit adder/subtractor

Si realizza lo schema a blocchi mostrato in **figura 2.1**. L'aggiunta delle porte XOR, se il segnale di SUB è attivo, complementa il secondo ingresso (vettoriale) eseguendo così una sottrazione. Viceversa, i due ingressi sono addizionati.

2.3 Risultati

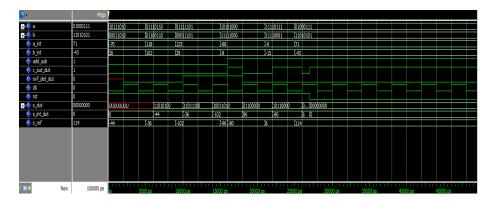


Figura 2.2: wave exercise no. 2

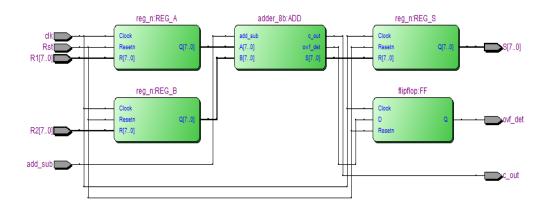
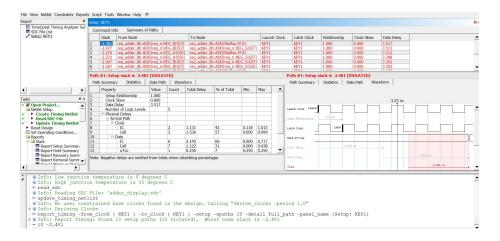


Figura 2.3: RTL viewer exercise no. 2

Di seguito, infine, è riportata l'analisi temporale



laddove si evince

$$f_{max} = 287.3 \,\mathrm{MHz}$$

$$t_{del} = 3.517 \,\mathrm{ns}$$

2.4 Appendice

```
library ieee;
1
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;
5
    entity reg_adder_8b is
      generic (n : integer := 8);
6
 7
      port ( R1,R2 : in signed ( n-1 downto 0 );
8
            add_sub : in std_logic;
9
10
            clk,Rst : in std_logic;
11
            c_out : out std_logic;
12
            ovf_det : out std_logic;
            S : out signed ( n-1 downto 0 )
13
14
          );
15
    end reg_adder_8b;
16
17
    architecture struct of reg_adder_8b is
18
19
20
      component flipflop
21
       port (D, Clock, Resetn : in std_logic;
22
             Q : out std_logic
           );
23
      end component;
24
```

```
25
26
      component reg_n
27
        generic ( N : integer:=n);
28
29
       port (R : in signed(N-1 downto 0);
30
           Clock, Resetn : in std_logic;
31
           Q : out signed(N-1 downto 0)
32
           );
      end component;
33
34
35
36
        component adder_8b
37
38
        generic (n : integer := n);
39
40
        port ( A,B : in signed ( n-1 downto 0 );
              add_sub : in std_logic;
41
42
              c_out : out std_logic;
43
              ovf_det : out std_logic;
44
             S : out signed ( n-1 downto 0 )
             );
45
        end component;
46
47
        signal A,B,Q : signed ( n-1 downto 0 );
48
        signal ovf : std_logic;
49
50
51
    begin
52
53
      REG_A: reg_n port map ( R1,clk,Rst,A);
54
      REG_B: reg_n port map ( R2,clk,Rst,B);
55
56
      ADD: adder_8b port map ( A,B,add_sub,c_out,ovf,Q );
57
58
      FF: flipflop port map ( ovf,clk,Rst,ovf_det );
59
      REG_S: reg_n port map (Q,clk,Rst,S);
60
61
    end architecture;
62
```

Code 2.1: reg_adder_8b.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reg_adder_8b_tb is
end reg_adder_8b_tb;

8
```

```
9
10
    architecture struct of reg_adder_8b_tb is
11
12
      component reg_adder_8b
13
        generic (n : integer := 8);
14
15
       port ( R1,R2 : in signed ( n-1 downto 0 );
            add_sub : in std_logic;
16
            clk,Rst : in std_logic;
17
            c_out : out std_logic;
18
            ovf_det : out std_logic;
19
            S : out signed ( n-1 downto 0 )
20
21
          );
22
      end component;
23
24
     signal A,B : signed ( 7 downto 0 );
25
     signal A_int,B_int : integer;
26
      signal add_sub,c_out_dut,ovf_det_dut,clk,Rst : std_logic;
27
      signal S_dut : signed ( 7 downto 0 );
28
      signal S_int_dut,S_ref : integer;
29
30
31 begin
32
33
     process begin
34
       for i in 1 to 10 loop
35
         clk <= '0';
36
         wait for 2 ns;
37
         clk <= '1';
38
         wait for 2 ns;
39
       end loop;
40
      end process;
41
42
      Rst <= '1','0' after 27 ns;</pre>
43
      add_sub <= '0', '1' after 16 ns;
44
45
      A <= "10111010", "01110110" after 5 ns, "011111101" after 9 ns,
46
          "10101000" after 15 ns,
47
          "11110111" after 21 ns, "01000111" after 25 ns;
48
     B \le "00011010", "01100110" after 5 ns, "00011101" after 9 ns,
49
          "11111000" after 15 ns,
50
          "11110001" after 21 ns, "11010101" after 25 ns;
51
52
     A_int <= to_integer(A);
53
      B_int <= to_integer(B);</pre>
54
55
      ADD: reg_adder_8b port map (A, B, add_sub, clk, Rst, c_out_dut,
56
```

```
ovf_det_dut, S_dut);
57
58
      S_int_dut <= to_integer(S_dut);</pre>
59
60
      with add_sub select
61
        S_ref <=
62
          to_integer(A + B) when '0',
63
          to_integer(A - B) when others;
64
65
    end architecture;
```

Code 2.2: reg_adder_8b_tb.vhd

```
1 library ieee;
 2
    use ieee.std_logic_1164.all;
3
    use ieee.numeric_std.all;
5
    entity adder_8b is
     generic (n : integer := 8);
 6
 7
      port ( A,B : in signed ( n-1 downto 0 );
8
            add_sub : in std_logic; --sub when add_sub == 1
9
            c_out : out std_logic;
10
11
            ovf_det : out std_logic;
12
            S : out signed (n-1 downto 0)
13
          );
14
15
    end adder_8b;
16
    architecture struct of adder_8b is
17
18
19
      component FA
20
      port( a,b : in std_logic;
21
             c_in : in std_logic;
             s,c_out: out std_logic
22
23
           );
24
      end component;
25
26
      signal carries : signed ( n downto 0);
27
      signal b_pos_neg : signed ( n-1 downto 0);
28
29
30
    begin
31
      carries(0) <= add_sub;</pre>
32
33
      GEN: for i in 0 to n-1 generate
34
        b_pos_neg(i) <= add_sub xor B(i);</pre>
35
       FULL_ADDERS: FA port map( A(i), b_pos_neg(i), carries(i), S(i),
36
```

```
carries(i+1));
37
38    end generate;
39
40    c_out <= carries(n);
41    ovf_det <= carries(n-1) xor carries(n);
42
43    end architecture;</pre>
```

Code 2.3: adder_8b.vhd

```
library ieee;
1
 2
    use ieee.std_logic_1164.all;
3
    use ieee.numeric_std.all;
4
5
    entity adder_8b_tb is
    end adder_8b_tb;
9
    architecture struct of adder_8b_tb is
10
      component adder_8b
11
        generic (n : integer := 8);
12
13
14
        port ( A,B : in signed ( n-1 downto 0 );
15
            add_sub : in std_logic; --sub when add_sub == 1
16
            c_out : out std_logic;
17
            ovf_det : out std_logic;
18
            S : out signed ( n-1 downto 0 )
          );
19
20
      end component;
21
      signal A : signed ( 7 downto 0 );
22
      signal B : signed ( 7 downto 0 );
23
     signal A_int,B_int : integer;
24
25
      signal add_sub : std_logic;
26
      signal c_out_dut,ovf_det_dut : std_logic;
27
      signal S_dut : signed ( 7 downto 0 );
28
      signal S_ref,S_int_dut : integer;
29
30
31
32
   begin
33
34
     add_sub <= '0', '1' after 16 ns;
35
36
     process begin
37
         for i in 2 to 4 loop
38
```

```
39
            A <= to_signed(63*i, A'length);
40
            for j in 0 to 2 loop
41
42
43
              B <= to_signed(61*j, B'length);</pre>
44
             wait for 2 ns;
45
            end loop;
46
47
48
          end loop;
49
50
        end process; A_int <= to_integer(A);</pre>
51
      B_int <= to_integer(B);</pre>
52
53
      ADD: adder_8b port map (A,B,add_sub,c_out_dut,ovf_det_dut,S_dut);
54
55
      S_int_dut <= to_integer(S_dut);</pre>
56
57
      with add_sub select
58
        S_ref <=
59
          to_integer(A + B) when '0',
          to_integer(A - B) when others;
60
61
62
    end architecture;
```

Code 2.4: adder_8b_tb.vhd

```
1 library ieee;
2 use ieee.std_logic_1164.all;
   use ieee.numeric_std.all;
5
   entity adder_display is
6
     --KEY -> clk, KEYO -> RST, SW(16) -> add_sub,SW(15->8) -> A,
7
     --SW(7->0) ->B,LEDR ->S,LEDG8 -> ovf_det,
8
     --HEX7_6 -> A display, HEX5_4 -> B dspaly, HEX1_0 S_display
9
10
     port ( SW : in signed ( 16 downto 0 );
            KEY1,KEY0 : in std_logic;
11
12
            LEDG8 : out std_logic;
13
            LEDR : buffer signed ( 7 downto 0 );
14
            HEX7,HEX6,HEX5,HEX4,HEX1,HEX0 : out std_logic_vector(0 to 6)
15
16
          );
17
   end adder_display;
18
19
   architecture struct of adder_display is
20
21
     component reg_adder_8b
22
       generic (n : integer := 8);
```

```
23
24
       port ( R1,R2 : in signed ( n-1 downto 0 );
25
            add_sub : in std_logic;
            clk,Rst : in std_logic;
26
27
            c_out : out std_logic;
28
            ovf_det : out std_logic;
29
            S : out signed ( n-1 downto 0 )
30
          );
31
      end component;
32
      component hex_display
33
       port( SW : in signed (3 downto 0);
34
           HEXA : out std_logic_vector ( 0 to 6)
35
36
37
      end component;
38
39
      signal c_out : std_logic;
40
41
42
    begin
43
      ADD: reg_adder_8b port map(SW(15 downto 8), SW(7 downto 0), SW(16),
44
          KEY1, KEY0, c_out, LEDG8, LEDR);
45
      --A display
46
      HEXA7: hex_display port map(SW(15 downto 12), HEX7);
47
48
     HEXA6: hex_display port map(SW(11 downto 8), HEX6);
49
50
      --B display
51
      HEXA5: hex_display port map(SW(7 downto 4), HEX5);
52
      HEXA4: hex_display port map(SW(3 downto 0), HEX4);
53
54
      --S display
      HEXA1: hex_display port map(LEDR(7 downto 4), HEX1);
55
56
     HEXAO: hex_display port map(LEDR(3 downto 0), HEXO);
57
58
59
    end architecture;
```

Code 2.5: adder_display.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity adder_display_tb is
end adder_display_tb;

8
```

```
architecture struct of adder_display_tb is
10
11
     component adder_display
12
       port (
13
            SW : in signed ( 16 downto 0 );
14
            KEY1,KEY0 : in std_logic;
15
            LEDG8 : out std_logic;
            LEDR : buffer signed ( 7 downto 0 );
16
            HEX7,HEX6,HEX5,HEX4,HEX1,HEX0 : out std_logic_vector(0 to 6)
17
          );
18
19
     end component;
20
     signal A : signed ( 7 downto 0 );
21
22
     signal B : signed ( 7 downto 0 );
     signal A_int,B_int : integer;
23
24
     signal A_B : signed (16 downto 0);
25
     signal add_sub,ovf_det_dut,clk,Rst : std_logic;
26
     signal S_dut : signed ( 7 downto 0 );
27
     signal S_int_dut,S_ref : integer;
28
     signal HEX7_dut,HEX6_dut,HEX5_dut,HEX4_dut,HEX1_dut,HEX0_dut :
          std_logic_vector(0 to 6);
29
30
31
   begin
32
33
     process
34
     begin
35
       for i in 1 to 10 loop
36
         clk <= '0';
37
         wait for 2 ns;
38
         clk <= '1';
39
         wait for 2 ns;
40
       end loop;
     end process;
41
42
     add_sub <= '0', '1' after 16 ns;
43
     Rst <= '1','0' after 27 ns;</pre>
44
45
      A <= "10111010", "01110110" after 5 ns, "011111101" after 9 ns,
46
          "10101000" after 15 ns,
          "11110111" after 21 ns, "01000111" after 25 ns;
47
48
49
     B \le "00011010", "01100110" after 5 ns, "00011101" after 9 ns,
          "11111000" after 15 ns,
50
          "11110001" after 21 ns, "11010101" after 25 ns;
51
52
     A_int <= to_integer(A);
53
     B_int <= to_integer(B);</pre>
54
55
     A_B <= add_sub & A & B;
```

```
56
57
     ADD: adder_display port map (A_B, clk, Rst, ovf_det_dut, S_dut,
58
                                 HEX7_dut,HEX6_dut,HEX5_dut,HEX4_dut,HEX1_dut,HEX0_dut);
59
      S_int_dut <= to_integer(S_dut);</pre>
60
61
     with add_sub select
62
63
       S_ref <=
64
         to_integer(A + B) when '0',
65
         to_integer(A - B) when others;
66
67
    end architecture;
```

Code 2.6: adder_display_tb.vhd

Capitolo 3

16-bit RCA, Carry-Bypass Adder and Carry-Select Adder

3.1 Spiegazione teorica

Per ogni sezione di questo esercizio si è sostituito l'8-bit Adder usato nell'esercizio 1 con un carry-bypass adder per la sezione 2 e con un carry-select adder per la sezione 3.

3.2 Procedimento

Per la sezione 1 è stato riutilizzato il codice del primo esercizio settando il generic n=16. Per ciò che concerne, invece, le sezioni 2 e 3, sono stati usati i blocchi di setup e carry propagation mostrato in figura 3.1 e in Code 3.10 - 3.11.

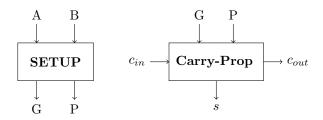
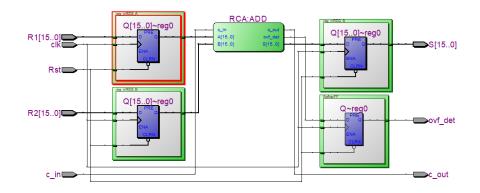


Figura 3.1: setup and carry-prop blocks

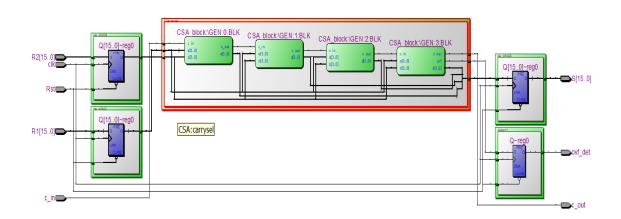
Nel caso del CSA, vengono usati due carry propagation che ricevono gli stessi ingressi e forniscono risultati diversi in virtù del c_{in} .

3.3 Risultati

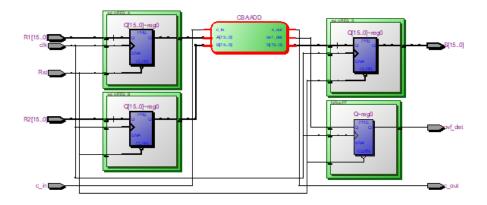
Di seguito sono mostrati i risultati prodotti da simulazione e sintesi.



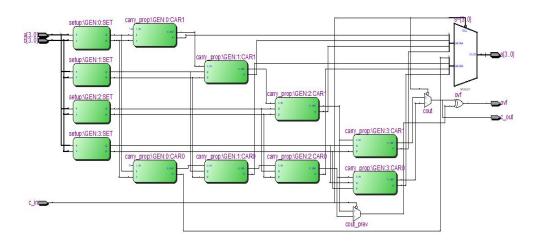
(a) RTL viewer RCA 16 bit



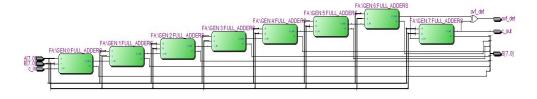
(b) RTL viewer CSA 16 bit



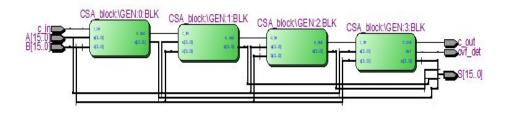
(c) RTL viewer BCA 16 bit



(d) RTL viewer CSA_block

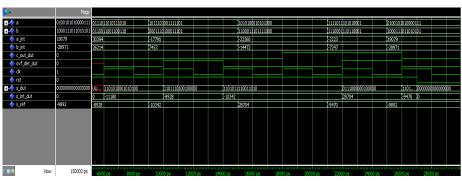


(e) RTL viewer RCA adder

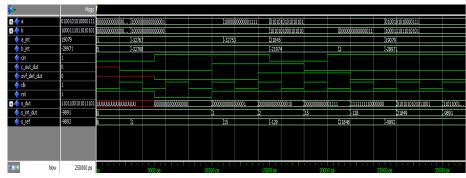


(f) RTL viewer CSA adder

Figura 3.2: RTL viewer exercise no. 3



(a) wave CBA

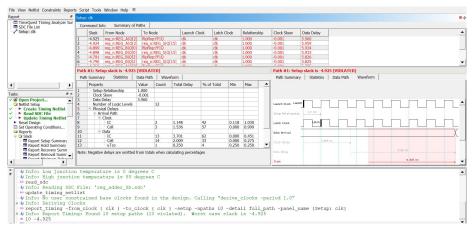


(b) wave CSA

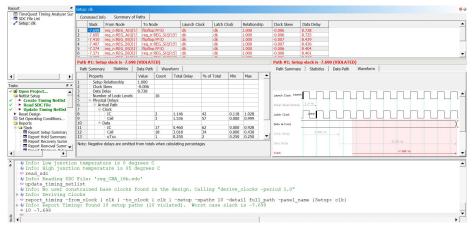
Figura 3.3: waves exercise no. 3

Laddove si è riportata solo la sintesi dell'RCA a 16 bit poichè la simulazione ha lo stesso andamento di quella mostrata al capitolo 1, differendo solo del parallelismo.

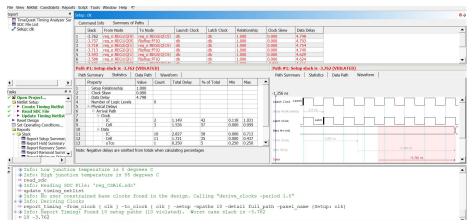
Di seguito, infine, i risultati della time analysis per i 3 sommatori in esame



(a) time analysis RCA 16 bit



(b) time analysis CBA 16 bit



(c) time analysis CSA 16 bit

Figura 3.4: time analysis exercise no. 3

Laddove si evincono, rispettivamente

$$f_{max} = 168.8 \, \text{MHz}$$
 $t_{del} = 5.96 \, \text{ns}$
 $f_{max} = 114.97 \, \text{MHz}$ $t_{del} = 8.73 \, \text{ns}$
 $f_{max} = 210 \, \text{MHz}$ $t_{del} = 4.80 \, \text{ns}$

3.4 Appendice

```
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;
    entity reg_CBA_16b is
5
      port ( R1,R2 : in signed ( 15 downto 0 );
6
 7
            c_in : in std_logic;
8
            clk,Rst : in std_logic;
9
            c_out : out std_logic;
            ovf_det : out std_logic;
10
            S : out signed ( 15 downto 0 )
11
12
13
    end reg_CBA_16b;
14
15
    architecture struct of reg_CBA_16b is
16
17
      component flipflop
       port (D, Clock, Resetn : in std_logic;
18
```

```
19
             Q : out std_logic
20
           );
21
      end component;
22
23
24
      component reg_n
25
        generic ( N : integer:=16);
26
27
       port (R : in signed(N-1 downto 0);
           Clock, Resetn : in std_logic;
28
           Q : out signed(N-1 downto 0)
29
           );
30
31
      end component;
32
33
34
35
        component CBA
36
         port(
             A,B : in signed( 15 downto 0 );
37
38
             c_in : in std_logic;
39
             S : out signed( 15 downto 0 );
             c_out,ovf_det : out std_logic
40
41
           );
        end component;
42
43
        signal A,B,Q : signed ( 15 downto 0 );
44
45
        signal ovf : std_logic;
46
47
    begin
48
49
      REG_A: reg_n port map ( R1,clk,Rst,A);
50
     REG_B: reg_n port map ( R2,clk,Rst,B);
51
52
     ADD: CBA port map ( A,B,c_in,Q ,c_out,ovf);
53
      FF: flipflop port map ( ovf,clk,Rst,ovf_det );
54
55
56
      REG_S: reg_n port map (Q,clk,Rst,S);
57
58
    end architecture;
```

Code 3.1: reg_CBA_16b.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reg_CBA_16b_tb is
end reg_CBA_16b_tb;
```

```
7
8
9
10
   architecture struct of reg_CBA_16b_tb is
11
12
     component reg_CBA_16b
13
14
       port ( R1,R2 : in signed ( 15 downto 0 );
15
16
            c_in : in std_logic;
17
            clk,Rst : in std_logic;
            c_out : out std_logic;
18
19
            ovf_det : out std_logic;
20
            S : out signed ( 15 downto 0 )
21
          );
22
23
     end component;
24
25
     signal A,B : signed ( 15 downto 0 );
26
     signal A_int,B_int : integer;
27
     signal c_out_dut,ovf_det_dut,clk,Rst : std_logic;
28
     signal S_dut : signed ( 15 downto 0 );
29
     signal S_int_dut,S_ref : integer;
30
31
32
   begin
33
34
     process begin
35
       for i in 1 to 10 loop
36
         clk <= '0';
37
         wait for 2 ns;
38
         clk <= '1';
39
         wait for 2 ns;
40
       end loop;
41
     end process;
42
43
     Rst <= '1','0' after 27 ns;</pre>
44
     A <= "1011101010111010", "0111011010111010" after 5 ns,
45
          "1011101001111101" after 9 ns, "1010100010101000" after 15 ns,
          "1111011101010001" after 21 ns, "0100101010000111" after 25 ns;
46
47
48
     B <= "0001100110011010", "011001100110" after 5 ns,
          "0001110100011101" after 9 ns, "1100011101111000" after 15 ns,
          "1110001110110001" after 21 ns, "1000111011010101" after 25 ns;
49
50
51
     A_int <= to_integer(A);
     B_int <= to_integer(B);</pre>
52
53
54
```

Code 3.2: reg_CBA_16b_tb.vhd

```
library ieee;
1
    use ieee.std_logic_1164.all;
3
    use ieee.numeric_std.all;
4
5
6
    entity CBA is
     port(
8
           A,B : in signed( 15 downto 0 );
9
           c_in : in std_logic;
10
           S : out signed( 15 downto 0 );
           c_out,ovf_det : out std_logic
11
         );
12
    end CBA;
13
14
15
16
    architecture behaviour of CBA is
17
18
     component setup
19
       port( a,b : in std_logic;
           p,g : out std_logic
20
         );
21
22
     end component;
23
24
25
      component carry_prop
26
       port( p,g,c_in : in std_logic;
27
           s,c_out : out std_logic
28
         );
29
      end component;
30
31
     signal P,G : std_logic_vector ( 15 downto 0 );
32
      signal carries : std_logic_vector ( 20 downto 0 );
33
      signal sel : std_logic_vector ( 3 downto 0 );
34
35
36
    begin
37
      carries(0) <= c_in;</pre>
38
39
```

```
40
                          GEN: for i in 0 to 3 generate
41
42
                           -- whole CBA ovf check
43
                          HDHDH: for j in 0 to 3 generate
44
45
                          SET: setup port map ( A(4*i+j), B(4*i+j), P(4*i+j), G(4*i+j));
46
47
                          \texttt{CA\_PROP: carry\_prop port map (P(4*i+j),G(4*i+j),carries(5*i+j),S(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j),G(4*i+j)
                                              +j),carries(5*i+j+1));
48
                                   end generate;
49
50
51
                               -- check on propagate value
52
                                  sel(i) \le P(4*i) \text{ and } P(4*i+1) \text{ and } P(4*i+2) \text{ and } P(4*i+3);
                                   carries(i*5 + 5) <= carries(i*5+4) when (sel(i) = '0') else
53
                                                      carries(i*5);
54
55
                          end generate;
56
57
                          c_out <= carries(20);</pre>
58
                          ovf_det <= carries(18) xor carries(19);</pre>
59
60
                 end architecture;
```

Code 3.3: CBA.vhd

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
   use ieee.numeric_std.all;
5
   entity CBA_tb is
   end CBA_tb;
6
8
9
   architecture struct of CBA_tb is
10
11
12
     component CBA
13
      port(
14
         A,B : in signed( 15 downto 0 );
15
        c_in : in std_logic;
16
        S : out signed( 15 downto 0 );
17
        c_out,ovf_det : out std_logic
18
          );
19
     end component;
20
     signal A : signed ( 15 downto 0 );
21
     signal B : signed ( 15 downto 0 );
22
     signal A_int,B_int : integer;
23
```

```
24
      signal c_out_dut,ovf_det_dut : std_logic;
25
      signal S_dut : signed ( 15 downto 0 );
26
     signal S_ref,S_int_dut : integer;
27
28
   begin
29
     process begin
30
         for i in 2 to 4 loop
31
32
           A <= to_signed(63*i, A'length);
33
           for j in 0 to 2 loop
34
35
             B <= to_signed(61*j, B'length);</pre>
36
37
             wait for 2 ns;
38
39
           end loop;
40
41
         end loop;
42
43
        end process;
44
45 A_int <= to_integer(A);
46 B_int <= to_integer(B);
   BCA: CBA port map (A,B,'0',S_dut,c_out_dut,ovf_det_dut);
47
49 S_int_dut <= to_integer(S_dut);</pre>
50 S_ref <= to_integer(A + B);
51
   end architecture;
```

Code 3.4: CBA_tb.vhd

```
1 library ieee;
   use ieee.std_logic_1164.all;
3
   use ieee.numeric_std.all;
 5
    entity reg_CSA16 is
     generic (n : integer := 16);
 6
 7
     port (
 8
       R1,R2 : in signed ( n-1 downto 0 );
9
       c_in : in std_logic;
10
       clk,Rst : in std_logic;
11
       c_out : out std_logic;
12
       ovf_det : out std_logic;
13
       S : out signed ( n-1 downto 0 )
     );
14
    end reg_CSA16;
15
16
17
```

```
architecture behavior of reg_CSA16 is
18
19
20
      component flipflop
21
        port (
22
         D, Clock, Resetn : in std_logic;
23
         Q : out std_logic
24
       );
25
      end component;
26
27
      component reg_n
        generic ( N : integer:=n);
28
29
       port (
         R : in signed(N-1 downto 0);
30
31
         Clock, Resetn : in std_logic;
32
         Q : out signed(N-1 downto 0)
33
       );
34
      end component;
35
36
      component CSA
37
         port(
38
           A,B : in signed( 15 downto 0 );
           c_in : in std_logic;
39
40
           S : out signed( 15 downto 0 );
           c_out,ovf_det : out std_logic
41
         );
42
43
      end component;
44
45
      signal a,b,q : signed(n-1 downto 0);
46
      signal ovfl : std_logic;
47
48
     begin
49
50
       REG1: reg_n port map (R1, clk, rst, a);
51
       REG2: reg_n port map (R2, clk, rst, b);
52
        carrysel: CSA port map (a, b, c_in, q, c_out, ovfl);
        FF: flipflop port map (ovfl, clk, rst, ovf_det);
53
       REGS: reg_n port map(q, clk, rst, S);
54
55
56
    end behavior;
```

Code 3.5: reg_CSA16.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reg_csa16_tb is
end reg_csa16_tb;
```

```
architecture test of reg_csa16_tb is
9
10
     component reg_CSA16 is
     generic (n : integer := 16);
11
12
     port (
13
       R1,R2 : in signed ( n-1 downto 0 );
14
       c_in : in std_logic;
       clk,Rst : in std_logic;
15
16
       c_out : out std_logic;
       ovf_det : out std_logic;
17
       S : out signed ( n-1 downto 0 )
18
19
     );
20
    end component;
21
     signal A,B : signed ( 15 downto 0 );
22
23
     signal A_int,B_int : integer;
     signal cin, c_out_dut,ovf_det_dut,clk,Rst : std_logic;
24
25
     signal S_dut : signed ( 15 downto 0 );
26
     signal s_int_dut, S_ref : integer;
27
28
29 begin
30
31
     process begin
       for i in 1 to 10 loop
32
         clk <= '0';
33
34
         wait for 2 ns;
35
         clk <= '1';
36
         wait for 2 ns;
37
       end loop;
38
     end process;
39
40
     rst <= '1', '0' after 5 ns, '1' after 10 ns;
41
42
     --input values
     A <= "000000000000000001", "100000000000001" after 3 ns,
43
          "100000000001111" after 11 ns,
      "0101010101010101" after 15 ns, "01010101010101" after 21 ns,
44
          "0100101010000111" after 25 ns;
45
     B \le "0000000000000000101", "1000000000000000" after 3 ns,
46
          "1000000000000000" after 11 ns,
47
      "1010101000101010" after 15 ns, "000000000000011" after 21 ns,
          "1000111011010101" after 25 ns;
48
49
     cin <= '0', '1' after 5 ns, '1' after 10 ns, '0' after 15 ns, '1'
          after 20 ns;
50
51
     DUT: reg_CSA16 port map (A, B, cin, clk, rst, c_out_dut, ovf_det_dut,
          s_dut);
```

```
52
53  A_int <= to_integer(A);
54  B_int <= to_integer(B);
55
56  S_int_dut <= to_integer(S_dut);
57  S_ref <= to_integer(A + B);
58
59  end test;
```

Code 3.6: reg_CSA16_tb.vhd

```
library ieee;
1
 2
    use ieee.std_logic_1164.all;
3
    use ieee.numeric_std.all;
4
5
    entity CSA is
7
     port(
           A,B : in signed( 15 downto 0 );
8
9
           c_in : in std_logic;
           S : out signed( 15 downto 0 );
10
           c_out,ovf_det : out std_logic
11
         );
12
13
    end CSA;
14
15
16
17
    architecture behaviour of CSA is
18
19
     component CSA_block
20
       port(
21
           a,b : in signed( 3 downto 0 );
22
           c_in : in std_logic;
           s : out signed( 3 downto 0 );
23
           c_out, ovf : out std_logic
24
25
         );
26
      end component;
27
28
      signal carries : std_logic_vector ( 4 downto 0 );
29
      signal ovfs : std_logic_vector(3 downto 0);
30
31
    begin
32
33
      carries(0) <= c_in;</pre>
34
35
      GEN: for i in 0 to 3 generate
36
37
       BLK: CSA_block port map (A((i*4+3) downto i*4), B( (i*4+3) downto
            i*4), carries(i),
```

Code 3.7: CSA.vhd

```
library ieee;
1
2
   use ieee.std_logic_1164.all;
3
   use ieee.numeric_std.all;
4
5
   entity csa_tb is
   end csa_tb;
8
   architecture test of csa_tb is
9
     component CSA
10
        port(
11
           A,B : in signed( 15 downto 0 );
12
           c_in : in std_logic;
13
           S : out signed( 15 downto 0 );
14
15
           c_out,ovf_det : out std_logic
16
         );
       end component;
17
18
19
       signal in1, in2, sum, sum_ref : signed(15 downto 0);
20
       signal cin, cout, cout_ref, ovf, ovf_ref : std_logic;
21
22
       begin
         --input values
23
         in1 <= "0000000000000000001", "100000000000001" after 5 ns,
24
              "000000000001111" after 10 ns,
         "0101010101010101" after 15 ns, "01010101010101" after 20 ns;
25
26
27
         in2 <= "000000000000000101", "100000000000000" after 5 ns,
              "10000000000000000" after 10 ns,
         "1010101000101010" after 15 ns, "00000000000011" after 20 ns;
28
29
         cin <= '0', '1' after 5 ns, '1' after 10 ns, '0' after 15 ns, '1'
30
             after 20 ns;
31
32
         --expected outputs
         sum_ref <= "000000000000000110", "0000000000000010" after 5 ns,
33
              "1000000000010000" after 10 ns,
         "1111111111111" after 15 ns, "01010101010101" after 20 ns;
34
```

Code 3.8: CSA_tb.vhd

```
1
2 library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;
6
7
    entity CSA_block is
8
     port(
           a,b : in signed( 3 downto 0 );
9
           c_in : in std_logic;
10
11
           s : out signed( 3 downto 0 );
12
           c_out, ovf : out std_logic
13
         );
14
    end CSA_block;
15
16
    architecture behaviour of CSA_block is
17
18
      component setup
19
       port( a,b : in std_logic;
20
           p,g : out std_logic
         );
21
22
      end component;
23
24
25
     component carry_prop
26
        port( p,g, c_in : in std_logic;
27
           s,c_out : out std_logic
28
         );
29
      end component;
30
31
     signal p,g : std_logic_vector ( 3 downto 0 );
32
33
     signal carries_0, carries_1 : std_logic_vector ( 4 downto 0 );
34
      signal sum_0,sum_1 : signed ( 3 downto 0 );
      signal cout, cout_prev : std_logic;
35
36
```

```
37
    begin
38
39
      carries_0(0) <= '0';
      carries_1(0) <= '1';
40
41
42
      GEN: for i in 0 to 3 generate
43
44
        SET: setup port map ( a(i), b(i), p(i), g(i));
        CARO: carry_prop port map ( p(i), g(i), carries_0(i), sum_0(i),
45
            carries_0(i+1));
        CAR1: carry_prop port map ( p(i), g(i), carries_1(i), sum_1(i),
46
            carries_1(i+1));
47
48
      end generate;
49
50
      cout <= carries_0(4) when (c_in = '0') else carries_1(4);</pre>
      cout_prev <= carries_0(3) when (c_in = '0') else carries_1(3);</pre>
51
52
53
      c_out <= cout;</pre>
54
      ovf <= cout xor cout_prev;</pre>
55
      s <= sum_0 when (c_in = '0') else sum_1;</pre>
56
57
58
    end architecture;
```

Code 3.9: CSA_block.vhd

```
1 library ieee;
 2 use ieee.std_logic_1164.all;
   use ieee.numeric_std.all;
4
5
    entity carry_prop is
     port( p,g,c_in : in std_logic;
 7
           s,c_out : out std_logic
         );
8
9
    end carry_prop;
10
11
    architecture logic of carry_prop is
12
13
   begin
14
15
     s <= c_in xor p;
      c_out <= g or ( p and c_in );</pre>
16
17
18
    end architecture;
```

Code 3.10: carry_prop.vhd

1

```
2 library ieee;
3 use ieee.std_logic_1164.all;
4 use ieee.numeric_std.all;
6
   entity setup is
7
     port( a,b : in std_logic;
8
          p,g : out std_logic
9
10
   end setup;
11
   architecture logic of setup is
12
13
14
   begin
15
16
     g \le a and b;
17
     p <= a xor b;
18
19 end architecture;
```

Code 3.11: setup.vhd

Capitolo 4

Multiplier

4.1 Spiegazione teorica

In quest'ultimo capitolo, ci proponiamo di implementare un moltiplicatore di ingressi unsigned di parallelismo 4, per la realizzazione del quale ci si è serviti dell' RCA implementato nel secondo esercizio, modificando opportunamente i tipi (da signed ad unsigned).

4.2 Procedimento

Lo schema a blocchi a partire dal quale si è implementato il circuito è mostrato in **figura 4.1**, nel quale si evidenzia la struttura globale del circuito. Rispetto alla notazione utilizzata in **Code 4.1**, per questioni grafiche, i segnali intermedi sono stati chiamati genericamente "A(i)" e "B(i)".

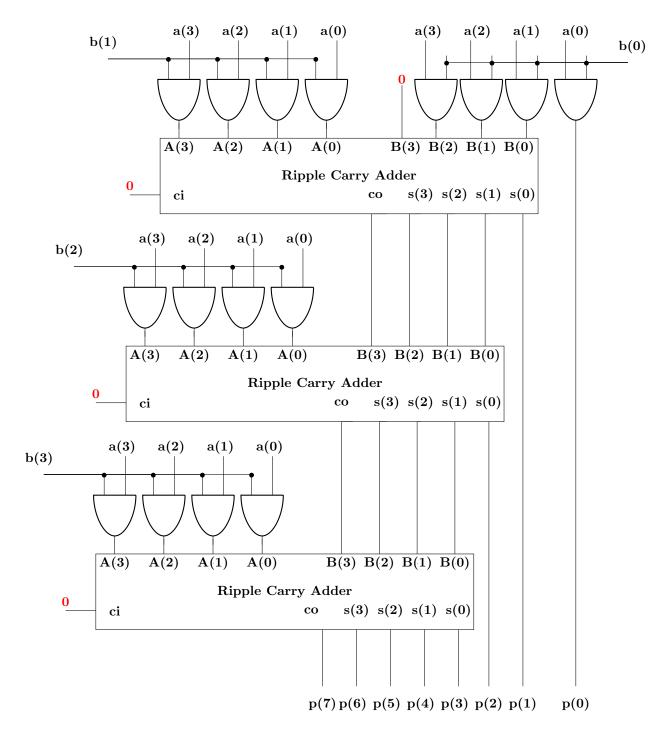


Figura 4.1: 4bit-Multiplier block scheme

4.3 Risultati

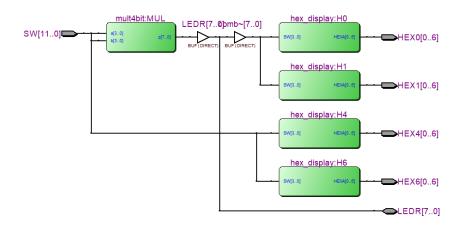


Figura 4.2: RTL viewer exercise no. 4



Figura 4.3: wave exercise no. 4

4.4 Appendice

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity mult4bit is
6 port(
```

```
a, b : in unsigned(3 downto 0);
         p : out unsigned (7 downto 0)
 8
 9
      );
10
    end mult4bit;
11
12
13
    architecture behaviour of mult4bit is
14
      component RCA
15
        port(
16
           a,b : in unsigned(3 downto 0);
           s : out unsigned(3 downto 0);
17
           cout : out std_logic
18
19
        );
20
      end component;
21
22
      signal ANDO, AND1, AND2 : unsigned (3 downto 0); -- ANDs input to RCAs
      signal B0, B1, B2 : unsigned (3 downto 0); -- intermediate signals
23
           between RCAs
24
      begin
25
26
27
      B0 <= ('0', a(3) and b(0), a(2) and b(0), a(1) and b(0));
28
    -- bitwise AND generating correct dimension range
29
    -- for R values of AND operand
30
31
      ANDO <= a and (ANDO'range => b(1));
32
      AND1 <= a and (AND1'range => b(2));
33
      AND2 <= a and (AND2'range => b(3));
34
35
36
     -- evaluating outputs
37
      p(0) \le a(0) \text{ and } b(0);
38
      RCAO: RCA port map (
39
                a \Rightarrow ANDO,
40
                b \Rightarrow B0,
41
                cout \Rightarrow B1(3),
42
                s(3 \text{ downto } 1) \Rightarrow B1(2 \text{ downto } 0),
                s(0) \Rightarrow p(1)
43
44
            );
45
46
      RCA1: RCA port map (
47
                a \Rightarrow AND1,
48
                b \Rightarrow B1,
49
                cout \Rightarrow B2(3),
50
                s(3 \text{ downto } 1) \Rightarrow B2(2 \text{ downto } 0),
51
                s(0) => p(2)
52
            );
53
54
     RCA2: RCA port map (
                a \Rightarrow AND2,
55
```

```
56 b => B2,

57 cout => p(7), -- last bit = carry out

58 s => p(6 downto 3)

59 );

60

61 end architecture;
```

Code 4.1: mult4bit.vhd

```
library ieee;
    use ieee.std_logic_1164.all;
 3
    use ieee.numeric_std.all;
 4
 5
    entity mult4bit_tb is
 6
    end mult4bit_tb;
 7
 9
    architecture testcase of mult4bit_tb is
10
      component mult4bit is
       port(
11
12
           a, b : in unsigned(3 downto 0);
13
          p : out unsigned (7 downto 0)
       );
14
15
      end component;
16
17
      signal a_dut, b_dut : unsigned(3 downto 0);
18
      signal p_dut, p_ref : unsigned(7 downto 0);
19
20
21
22
      begin
        UUT : mult4bit port map (a_dut, b_dut, p_dut);
23
24
        -- test cases (all combinations of two 4 bit wide inputs)
25
26
       process begin
27
          for i in 0 to 15 loop
28
29
           a_dut <= to_unsigned(i, a_dut'length);</pre>
30
31
           for j in 0 to 15 loop
32
             b_dut <= to_unsigned(j, b_dut'length);</pre>
33
             wait for 0 ns;
             p_ref <= a_dut * b_dut; -- works bc they're both unsigned !!</pre>
34
35
             wait for 5 ns;
36
           end loop;
37
38
          end loop;
39
        end process;
40
```

```
41 42 end architecture;
```

Code 4.2: mult4bit_tb.vhd

```
library ieee;
   use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;
5
    entity mult4bit_synth is
6
     port(
        SW : in unsigned (11 downto 0);
8
        LEDR : buffer unsigned (7 downto 0);
9
        HEX6 : out std_logic_vector(0 to 6);
10
        HEX4 : out std_logic_vector(0 to 6);
11
        HEX1 : out std_logic_vector(0 to 6);
12
        HEX0 : out std_logic_vector(0 to 6)
13
     );
14
    end mult4bit_synth;
15
    architecture behaviour of mult4bit_synth is
16
17
     component mult4bit is
18
19
       port(
20
          a, b : in unsigned(3 downto 0);
21
          p : out unsigned (7 downto 0)
       );
22
23
     end component;
24
25
      component hex_display is
26
       port(
27
           SW : in unsigned (3 downto 0);
28
           HEXA : out std_logic_vector ( 0 to 6)
29
           );
     end component;
30
31
32
     begin
33
       MUL : mult4bit port map(SW(11 downto 8), SW(3 downto 0), LEDR);
34
35
       H6 : hex_display port map(SW(11 downto 8), HEX6);
36
       H4 : hex_display port map(SW(3 downto 0), HEX4);
37
       H1: hex_display port map(LEDR (7 downto 4), HEX1); -- last 4 bits
            on HEX1
38
       HO : hex_display port map(LEDR(3 downto 0), HEXO); -- first 4 bits
            on HEXO
    end architecture;
39
```

Code 4.3: mult4b_synth.vhd