Elettronica dei Sistemi Digitali
Lab 03

Data-Path Elements

Gruppo: A11

Pronesti, Massimiliano	S245831
Oropallo, Maria Vittoria	S245999
Nicolicchia, Riccardo	S244728
Miceli, Michelangelo	S245478

8 aprile 2020

Indice

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**.

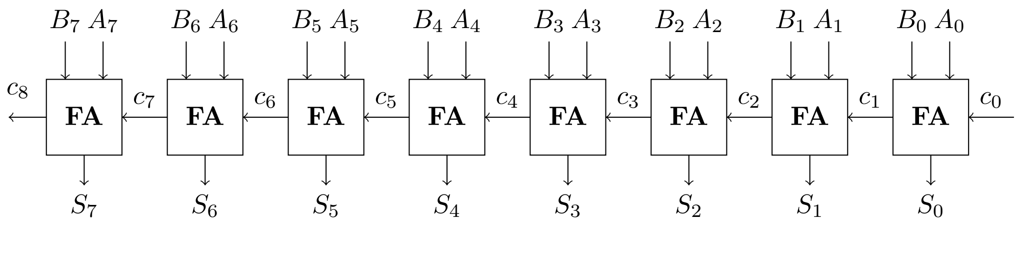


Figura 1.1: 8-bit full adder

$SW(3)$	$SW(2)$	$SW(1)$	$SW(0)$	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	C
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

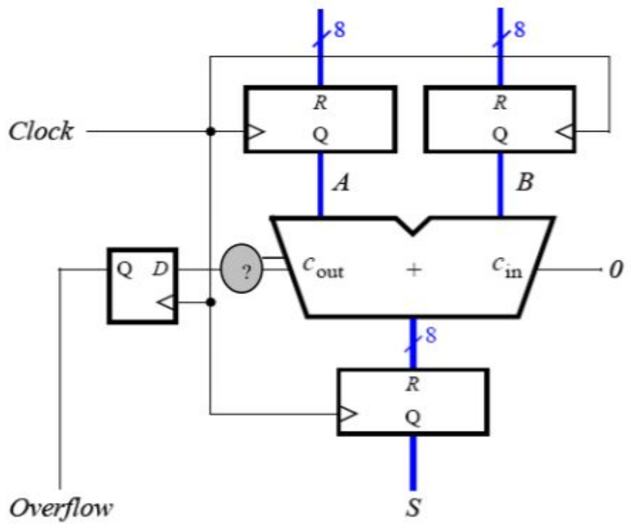


Figura 1.3: 8-bit registered-adder

1.3 Risultati

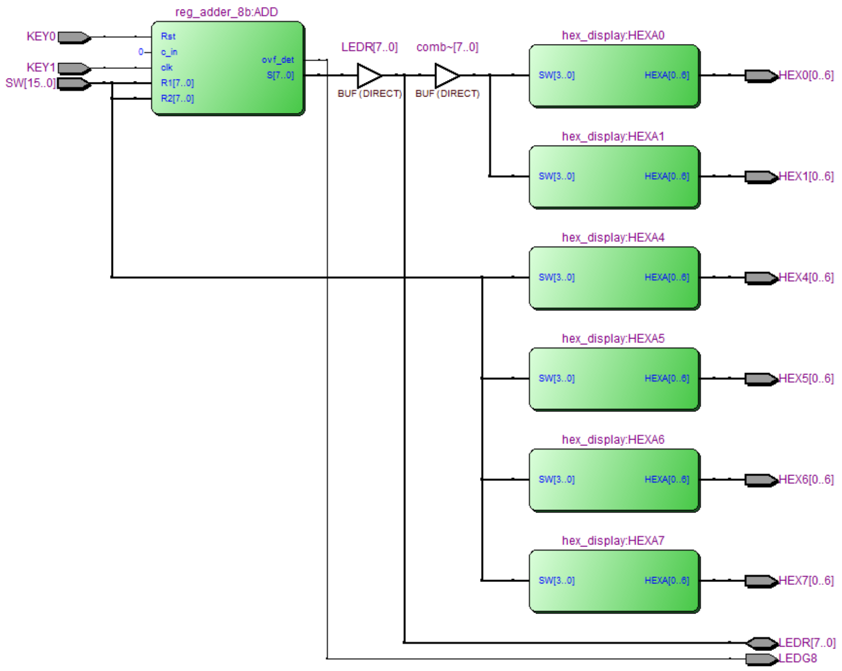


Figura 1.4: RTL viewer exercise no. 1

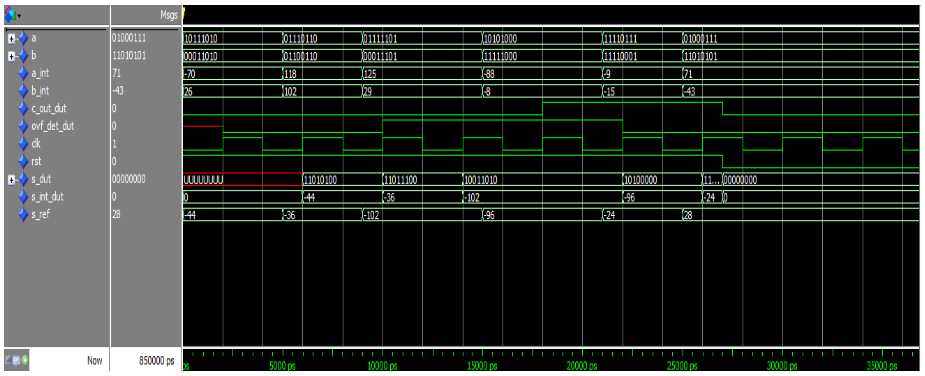
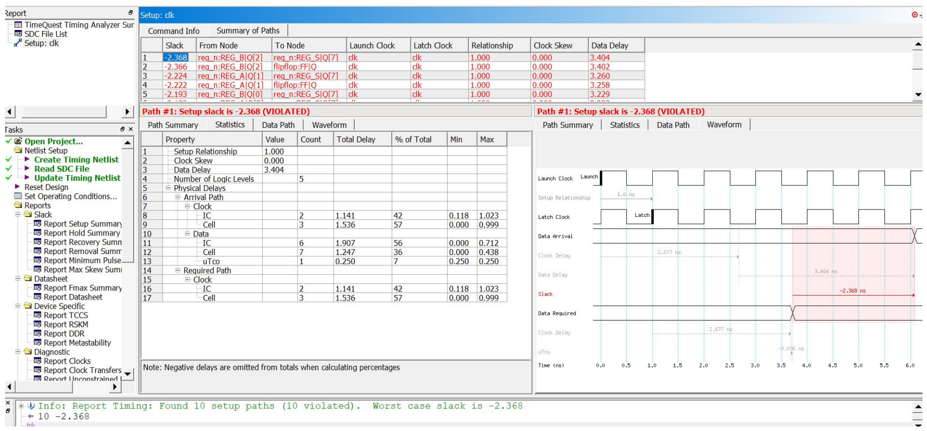


Figura 1.5: wave exercise no. 1

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



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-2.368	reg_n:REG_B|Q[2]	reg_n:REG_S|Q[7]	clk	clk	1.000	0.000	3.404
2	-2.366	reg_n:REG_B|Q[2]	flipflop:FF|Q	clk	clk	1.000	0.000	3.402
3	-2.224	reg_n:REG_A|Q[1]	reg_n:REG_S|Q[7]	clk	clk	1.000	0.000	3.260
4	-2.222	reg_n:REG_A|Q[1]	flipflop:FF|Q	clk	clk	1.000	0.000	3.258
5	-2.193	reg_n:REG_B|Q[0]	reg_n:REG_S|Q[7]	clk	clk	1.000	0.000	3.229

Path #1: Setup slack is -2.368 (VIOLATED)

	Property	Value	Count	Total Delay	% of Total	Min	Max
1	Setup Relationship	1.000					
2	Clock Skew	0.000					
3	Data Delay	3.404					
4	Number of Logic Levels		5				
5	Physical Delays						
6	Arrival Path						
7	Clock						
8	IC		2	1.141	42	0.118	1.023
9	Cell		3	1.536	57	0.000	0.999
10	Data						
11	IC		6	1.907	56	0.000	0.712
12	Cell		7	1.247	36	0.000	0.438
13	uTco		1	0.250	7	0.250	0.250
14	Required Path						
15	Clock						
16	IC		2	1.141	42	0.118	1.023
17	Cell		3	1.536	57	0.000	0.999

Note: Negative delays are omitted from totals when calculating percentages

Path #1: Setup slack is -2.368 (VIOLATED)

```
Info: Report Timing: Found 10 setup paths (10 violated).  Worst case slack is -2.368
⇐ 10 -2.368
tcl>
```

laddove si evincono

$$f_{max} = 286.9\,\text{MHz}$$
$$t_{del} = 3.404\,\text{ns}$$

1.4 Appendice

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

entity reg_adder_8b is

  generic (n : integer := 8);

  port ( R1,R2 : in signed ( n-1 downto 0 );
        c_in : in std_logic;
        clk,Rst : in std_logic;
        c_out : out std_logic;
        ovf_det : out std_logic;
        S : out signed ( n-1 downto 0 )
      );

end reg_adder_8b;

architecture struct of reg_adder_8b is

  component flipflop
    port (D, Clock, Resetn : in std_logic;
         Q : out std_logic
      );
  end component;

  component reg_n

    generic ( N : integer:=n);

    port (R : in signed(N-1 downto 0);
        Clock, Resetn : in std_logic;
        Q : out signed(N-1 downto 0)
        );
  end component;



    component RCA

    generic (n : integer := n);

     port ( A,B : in signed ( n-1 downto 0 );
          c_in : in std_logic;
          c_out : out std_logic;
          ovf_det : out std_logic;
         S : out signed ( n-1 downto 0 )
         );
    end component;

    signal A,B,Q : signed ( n-1 downto 0 );
    signal ovf : std_logic;

begin

  REG_A: reg_n port map ( R1,clk,Rst,A);
  REG_B: reg_n port map ( R2,clk,Rst,B);

  ADD: RCA port map ( A,B,c_in,c_out,ovf,Q );

  FF: flipflop port map ( ovf,clk,Rst,ovf_det );

  REG_S: reg_n port map (Q,clk,Rst,S);

end architecture;
```

Code 1.1: reg_adder_8b.vhd

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

```

```vhdl
entity reg_adder_8b_tb is
end reg_adder_8b_tb;



architecture struct of reg_adder_8b_tb is

  component reg_adder_8b
    generic (n : integer := 8);

    port ( R1,R2 : in signed ( n-1 downto 0 );
         c_in : in std_logic;
         clk,Rst : in std_logic;
         c_out : out std_logic;
         ovf_det : out std_logic;
         S : out signed ( n-1 downto 0 )
       );

  end component;

  signal A,B : signed ( 7 downto 0 );
  signal A_int,B_int : integer;
  signal c_out_dut,ovf_det_dut,clk,Rst : std_logic;
  signal S_dut : signed ( 7 downto 0 );
  signal S_int_dut,S_ref : integer;


begin

  process begin
    for i in 1 to 10 loop
      clk <= '0';
      wait for 2 ns;
      clk <= '1';
      wait for 2 ns;
    end loop;
  end process;

  Rst <= '1','0' after 27 ns;

  A <= "10111010", "01110110" after 5 ns, "01111101" after 9 ns,
      "10101000" after 15 ns,
       "11110111" after 21 ns, "01000111" after 25 ns;

  B <= "00011010", "01100110" after 5 ns, "00011101" after 9 ns,
      "11111000" after 15 ns,
       "11110001" after 21 ns, "11010101" after 25 ns;

  A_int <= to_integer(A);
  B_int <= to_integer(B);


  ADD: reg_adder_8b port map (A, B, '0', clk, Rst, c_out_dut,
      ovf_det_dut, S_dut);

  S_int_dut <= to_integer(S_dut);
  S_ref <= to_integer(A + B);

end architecture;
```

Code 1.2: reg_adder_tb.vhd

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

entity adder_display is

  --KEY -> clk, KEY0 -> RST, SW(15->8) -> A,
  --SW(7->0) ->B,LEDR ->S,LEDG8 -> ovf_det,
  --HEX7_6 -> A display, HEX5_4 -> B dspaly, HEX1_0 S_display
  port (
      SW : in signed ( 15 downto 0 );
      KEY1,KEY0 : in std_logic;
      LEDG8 : out std_logic;
      LEDR : buffer signed ( 7 downto 0 );
      HEX7,HEX6,HEX5,HEX4,HEX1,HEX0 : out std_logic_vector(0 to 6)
       );
end adder_display;

architecture struct of adder_display is

  component reg_adder_8b is
    generic (n : integer := 8);

    port ( R1,R2 : in signed ( n-1 downto 0 );
           c_in : in std_logic;
           clk,Rst : in std_logic;
           c_out : out std_logic;
           ovf_det : out std_logic;
           S : out signed ( n-1 downto 0 )
         );

  end component;

  component hex_display
    port(
      SW : in signed (3 downto 0);
      HEXA : out std_logic_vector ( 0 to 6)
    );
  end component;


begin

  ADD: reg_adder_8b port map(R1 => SW(15 downto 8),R2 => SW(7 downto 0),
                             c_in => '0',clk => KEY1,Rst => KEY0,
                             ovf_det => LEDG8,S => LEDR);

  --A display
  HEXA7: hex_display port map(SW(15 downto 12),HEX7);
  HEXA6: hex_display port map(SW(11 downto 8),HEX6);

  --B display
  HEXA5: hex_display port map(SW(7 downto 4),HEX5);
  HEXA4: hex_display port map(SW(3 downto 0),HEX4);

  --S display
  HEXA1: hex_display port map(LEDR(7 downto 4),HEX1);
  HEXA0: hex_display port map(LEDR(3 downto 0),HEX0);


end architecture;
```

Code 1.3: 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;


architecture struct of adder_display_tb is

  component adder_display
    port (
       SW : in signed ( 15 downto 0 );
       KEY1,KEY0 : in std_logic;
       LEDG8 : out std_logic;
       LEDR : buffer signed ( 7 downto 0 );
       HEX7,HEX6,HEX5,HEX4,HEX1,HEX0 : out std_logic_vector(0 to 6)
       );
  end component;

  signal A : signed ( 7 downto 0 );
  signal B : signed ( 7 downto 0 );
  signal A_int,B_int : integer;
  signal A_B : signed (15 downto 0);
  signal ovf_det_dut,clk,Rst : std_logic;
  signal S_dut : signed ( 7 downto 0 );
  signal S_int_dut,S_ref : integer;
  signal HEX7_dut,HEX6_dut,HEX5_dut,HEX4_dut,HEX1_dut,HEX0_dut :
      std_logic_vector(0 to 6);


begin

  process begin
    for i in 1 to 10 loop
      clk <= '0';
      wait for 2 ns;
      clk <= '1';
      wait for 2 ns;
    end loop;
  end process;

  Rst <= '1','0' after 27 ns;

  A <= "10111010", "01110110" after 5 ns, "01111101" after 9 ns,
      "10101000" after 15 ns,
       "11110111" after 21 ns, "01000111" after 25 ns;

  B <= "00011010", "01100110" after 5 ns, "00011101" after 9 ns,
      "11111000" after 15 ns,
       "11110001" after 21 ns, "11010101" after 25 ns;

  A_int <= to_integer(A);
  B_int <= to_integer(B);
  A_B <= A & B;

  ADD: adder_display port map (A_B, clk, Rst, ovf_det_dut, S_dut,
                              HEX7_dut,HEX6_dut,HEX5_dut,HEX4_dut,HEX1_dut,HEX0_dut);

  S_int_dut <= to_integer(S_dut);
  S_ref <= to_integer(A + B) ;

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
```

```
  generic (n : integer := 8);
  port ( A,B : in signed ( n-1 downto 0 );
         c_in : in std_logic;
         c_out : out std_logic;
         ovf_det : out std_logic;
         S : out signed ( n-1 downto 0 )
       );

end RCA;

architecture struct of RCA is

  component FA
   port( a,b : in std_logic;
          c_in : in std_logic;
          s,c_out: out std_logic
        );
  end component;

  signal carries : signed ( n downto 0);

  begin
  carries(0) <= c_in;

  GEN : for i in 0 to n-1 generate
    FULL_ADDERS: FA port map( A(i), B(i), carries(i), S(i),
         carries(i+1));
  end generate;

  c_out <= carries(n);
  ovf_det <= carries(n-1) xor carries(n);
end architecture;
```

Code 1.5: RCA.vhd

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

entity RCA_tb is
end RCA_tb;


architecture struct of RCA_tb is

  component RCA
    generic (n : integer := 8);

     port ( A,B : in signed ( n-1 downto 0 );
        c_in : in std_logic; --sub when add_sub == 1
        c_out : out std_logic;
        ovf_det : out std_logic;
        S : out signed ( n-1 downto 0 )
      );
  end component;

  signal A : signed ( 7 downto 0 );
  signal B : signed ( 7 downto 0 );
  signal A_int,B_int : integer;
  signal c_out_dut,ovf_det_dut : std_logic;
  signal S_dut : signed ( 7 downto 0 );
  signal S_ref,S_int_dut : integer;


begin
  process begin
   -- generating signed numbers for testcases
     for i in 2 to 4 loop

       A <= to_signed(63*i, A'length);

       for j in 0 to 2 loop
         B <= to_signed(61*j, B'length);
         wait for 5 ns;
       end loop;

     end loop;

    end process;

  A_int <= to_integer(A);
  B_int <= to_integer(B);

  ADD: RCA port map (A,B,'0',c_out_dut,ovf_det_dut,S_dut);

  S_int_dut <= to_integer(S_dut);
  S_ref <= to_integer(A + B) ;

end architecture;
```

Code 1.6: RCA_tb.vhd

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

entity FA is
  port( a,b : in std_logic;
       c_in : in std_logic;
```

```
      s,c_out: out std_logic
    );
end FA;


architecture struct of FA is

  signal p : std_logic;

  begin
   p <= a xor b;
   s <= c_in xor p;
   c_out <= b when (p='0') else c_in;

end architecture;
```

Code 1.7: FA.vhd

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


entity FA_tb is
end FA_tb;


architecture struct of FA_tb is

 component FA
  port( a,b : in std_logic;
        c_in : in std_logic;
        s,c_out: out std_logic
      );
  end component;

  signal input_string : std_logic_vector(0 to 2);
  signal s_dut,c_out_dut : std_logic;

begin

  input_string <= "000","001" after 5 ns,"010" after 10 ns,
                "011" after 15 ns,"100" after 20 ns,
                "101" after 25 ns,"110" after 30 ns, "111" after 35 ns;

  FULL_ADDER: FA port map(a => input_string(1),b => input_string(0),
                c_in => input_string(2),s => s_dut,c_out => c_out_dut);

end architecture;
```

Code 1.8: FA_tb.vhd

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


entity flipflop is
  port (
    D, Clock, Resetn : in std_logic;
    Q : out std_logic
  );
end flipflop;


architecture Behavior of flipflop is
begin

  process (Clock, Resetn)
  begin
    if (Resetn = '0') then -- asynchronous clear
      Q <= '0';
    elsif (Clock'EVENT and Clock = '1') then
      Q <= D;
    end if;
  end process;

end Behavior;
```

Code 1.9: flipflop.vhd

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


entity reg_n is
  generic ( N : integer:=8);
  port (
    R : in signed(N-1 downto 0);
    Clock, Resetn : in std_logic;
    Q : out signed(N-1 downto 0));
end reg_n;


architecture Behavior of reg_n is
begin

  process (Clock, Resetn)
  begin
    if (Resetn = '0') then
```

```
      Q <= (others => '0');
    elsif (Clock'EVENT AND Clock = '1') then
      Q <= R;
    end if;
  end process;

end Behavior;
```

Code 1.10: reg_n.vhd

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


entity reg_n_tb is
end reg_n_tb;


architecture Behavior of reg_n_tb is

  component reg_n is
    generic ( N : integer:=3);

    port (R : in signed(N-1 downto 0);
        Clock, Resetn : in std_logic;
        Q : out signed(N-1 downto 0)
        );

  end component;

    signal clk : std_logic;
    signal Rst : std_logic;
    signal input_string : signed(2 downto 0);
    signal out_dut : signed(2 downto 0);


begin

  process begin
    for i in 1 to 10 loop
      clk <= '0';
      wait for 1 ns;
      clk <= '1';
      wait for 1 ns;
    end loop;
  end process;

  Rst <= '1','0' after 13 ns;

  input_string <= "000", "001" after 4 ns,"010" after 8 ns,
                  "011" after 9 ns, "100" after 11 ns;

  REGN: reg_n port map(input_string,clk,Rst,out_dut);

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

  port( SW : in signed (3 downto 0);
        HEXA : out std_logic_vector ( 0 to 6)
      );

end hex_display;


architecture behaviour of hex_display is

begin

  with SW select
    HEXA <=
      "0000001" when "0000",
      "1001111" when "0001",
      "0010010" when "0010",
      "0000110" when "0011",
      "1001100" when "0100",
      "0100100" when "0101",
      "0100000" when "0110",
      "0001111" when "0111",
      "0000000" when "1000",
      "0000100" when "1001",
      "0001000" when "1010", -- A
      "1100000" when "1011", -- b
      "0110001" when "1100", -- C
      "1000010" when "1101", -- d
      "0110000" when "1110", -- E
      "0111000" when "1111", -- F
      "1111111" when others; --black for undefined values

end architecture;
```

Code 1.12: hex_display.vhd

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

entity hex_display_tb is
end hex_display_tb;


architecture behaviour of hex_display_tb is

  component hex_display
    port(
      SW : in signed (3 downto 0);
      HEXA : out std_logic_vector ( 0 to 6)
      );
  end component;

  signal n_in : signed (3 downto 0);
  signal out_dut : std_logic_vector (0 to 6);

begin
  n_in <= "0000", "0001" after 5 ns, "0010" after 10 ns, "0011" after 15 ns,
                  "0100" after 20 ns, "0101" after 25 ns, "0110" after 30 ns,
                  "0111" after 35 ns, "1000" after 40 ns, "1001" after 45 ns,
                  "1010" after 50 ns, "1011" after 55 ns, "1100" after 60 ns,
                  "1101" after 65 ns, "1110" after 70 ns, "1111" after 75 ns;

  DISPLAY: hex_display port map (n_in,out_dut);

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

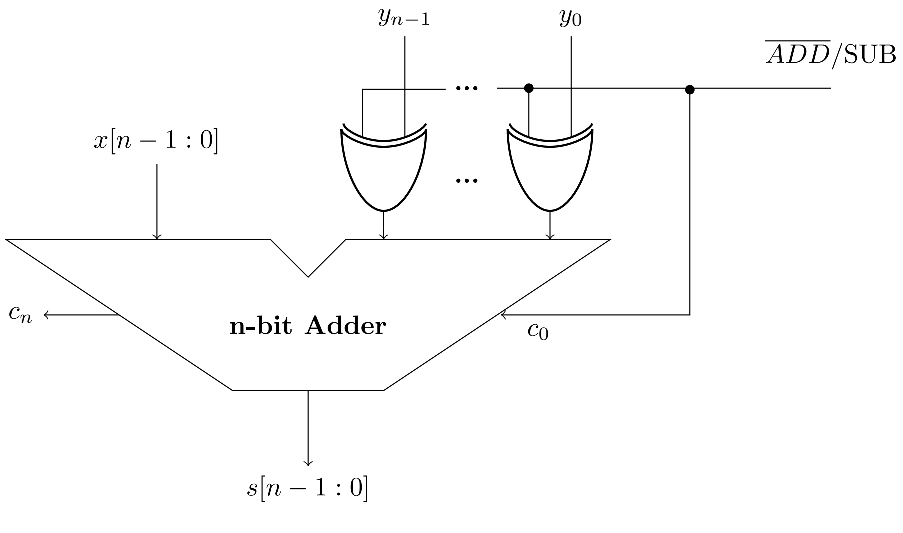


Figura 2.1: 2's n-bit adder/subtractor

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

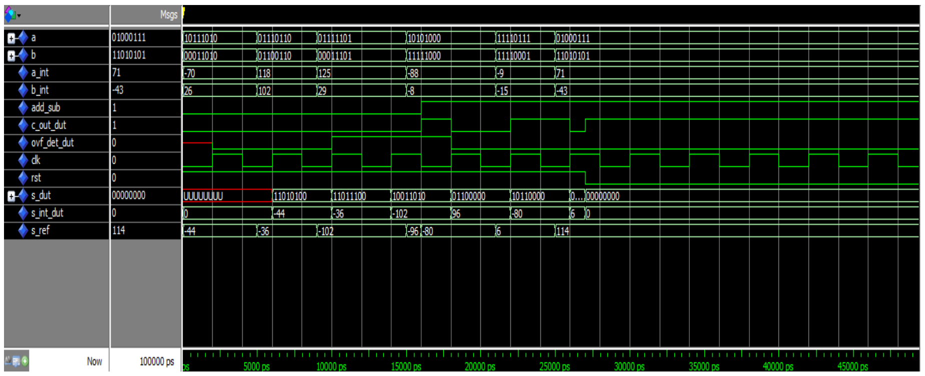


Figura 2.2: wave exercise no. 2

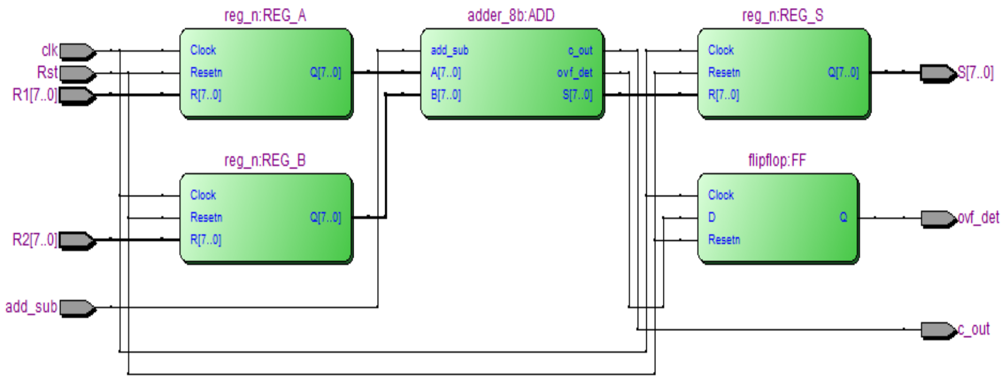
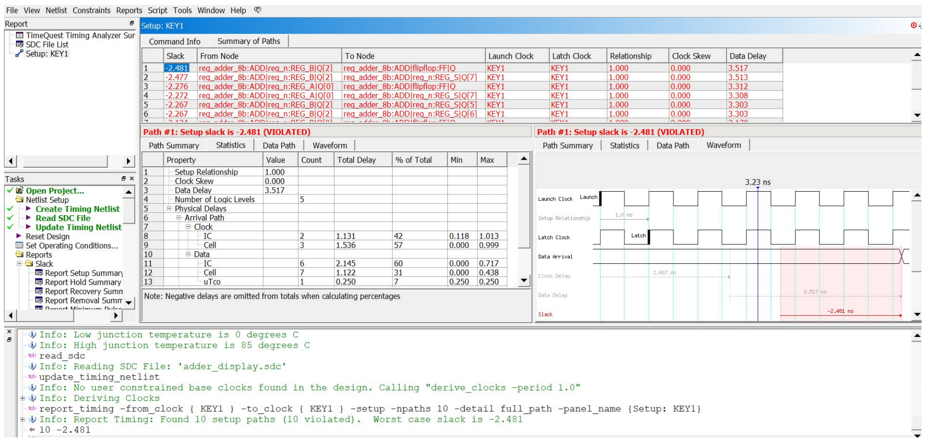


Figura 2.3: RTL viewer exercise no. 2

Di seguito, infine, è riportata l'analisi temporale


Setup: KEY1

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-2.481	reg_adder_8b:ADD|reg_n:REG_B|Q[2]	reg_adder_8b:ADD|flipflop:FF|Q	KEY1	KEY1	1.000	0.000	3.517
2	-2.477	reg_adder_8b:ADD|reg_n:REG_B|Q[2]	reg_adder_8b:ADD|reg_n:REG_S|Q[7]	KEY1	KEY1	1.000	0.000	3.513
3	-2.276	reg_adder_8b:ADD|reg_n:REG_A|Q[0]	reg_adder_8b:ADD|flipflop:FF|Q	KEY1	KEY1	1.000	0.000	3.312
4	-2.272	reg_adder_8b:ADD|reg_n:REG_A|Q[0]	reg_adder_8b:ADD|reg_n:REG_S|Q[7]	KEY1	KEY1	1.000	0.000	3.308
5	-2.267	reg_adder_8b:ADD|reg_n:REG_B|Q[2]	reg_adder_8b:ADD|reg_n:REG_S|Q[5]	KEY1	KEY1	1.000	0.000	3.303
6	-2.267	reg_adder_8b:ADD|reg_n:REG_B|Q[2]	reg_adder_8b:ADD|reg_n:REG_S|Q[6]	KEY1	KEY1	1.000	0.000	3.303

Path #1: Setup slack is -2.481 (VIOLATED)

	Property	Value	Count	Total Delay	% of Total	Min	Max
1	Setup Relationship	1.000					
2	Clock Skew	0.000					
3	Data Delay	3.517					
4	Number of Logic Levels		5				
5	Physical Delays						
6	Arrival Path						
7	Clock						
8	IC		2	1.131	42	0.118	1.013
9	Cell		3	1.536	57	0.000	0.999
10	Data						
11	IC		6	2.145	60	0.000	0.717
12	Cell		7	1.122	31	0.000	0.438
13	uTco		1	0.250	7	0.250	0.250

Note: Negative delays are omitted from totals when calculating percentages

```
Info: Low junction temperature is 0 degrees C
Info: High junction temperature is 85 degrees C
read_sdc
Info: Reading SDC File: 'adder_display.sdc'
update_timing_netlist
Info: No user constrained base clocks found in the design. Calling "derive_clocks -period 1.0"
Info: Deriving Clocks
report_timing -from_clock { KEY1 } -to_clock { KEY1 } -setup -npaths 10 -detail full_path -panel_name {Setup: KEY1}
Info: Report Timing: Found 10 setup paths (10 violated).  Worst case slack is -2.481
10 -2.481
```

laddove si evince

$$f_{max} = 287.3\,\text{MHz}$$
$$t_{del} = 3.517\,\text{ns}$$

2.4 Appendice

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

entity reg_adder_8b is
  generic (n : integer := 8);

  port ( R1,R2 : in signed ( n-1 downto 0 );
         add_sub : in std_logic;
         clk,Rst : in std_logic;
         c_out : out std_logic;
         ovf_det : out std_logic;
         S : out signed ( n-1 downto 0 )
       );
end reg_adder_8b;


architecture struct of reg_adder_8b is

  component flipflop
    port (D, Clock, Resetn : in std_logic;
          Q : out std_logic
         );
  end component;

  component reg_n
    generic ( N : integer:=n);

    port (R : in signed(N-1 downto 0);
        Clock, Resetn : in std_logic;
        Q : out signed(N-1 downto 0)
        );
  end component;



    component adder_8b
    generic (n : integer := n);

     port ( A,B : in signed ( n-1 downto 0 );
           add_sub : in std_logic;
           c_out : out std_logic;
           ovf_det : out std_logic;
          S : out signed ( n-1 downto 0 )
          );
    end component;

    signal A,B,Q : signed ( n-1 downto 0 );
    signal ovf : std_logic;

begin

  REG_A: reg_n port map ( R1,clk,Rst,A);
  REG_B: reg_n port map ( R2,clk,Rst,B);

  ADD: adder_8b port map ( A,B,add_sub,c_out,ovf,Q );

  FF: flipflop port map ( ovf,clk,Rst,ovf_det );

  REG_S: reg_n port map (Q,clk,Rst,S);

end architecture;
```

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;


```

```

architecture struct of reg_adder_8b_tb is

  component reg_adder_8b
    generic (n : integer := 8);

    port ( R1,R2 : in signed ( n-1 downto 0 );
         add_sub : in std_logic;
         clk,Rst : in std_logic;
         c_out : out std_logic;
         ovf_det : out std_logic;
         S : out signed ( n-1 downto 0 )
       );
  end component;

  signal A,B : signed ( 7 downto 0 );
  signal A_int,B_int : integer;
  signal add_sub,c_out_dut,ovf_det_dut,clk,Rst : std_logic;
  signal S_dut : signed ( 7 downto 0 );
  signal S_int_dut,S_ref : integer;


begin

  process begin
    for i in 1 to 10 loop
      clk <= '0';
      wait for 2 ns;
      clk <= '1';
      wait for 2 ns;
    end loop;
  end process;

  Rst <= '1','0' after 27 ns;

  add_sub <= '0', '1' after 16 ns;

  A <= "10111010", "01110110" after 5 ns, "01111101" after 9 ns,
      "10101000" after 15 ns,
       "11110111" after 21 ns, "01000111" after 25 ns;

  B <= "00011010", "01100110" after 5 ns, "00011101" after 9 ns,
      "11111000" after 15 ns,
       "11110001" after 21 ns, "11010101" after 25 ns;

  A_int <= to_integer(A);
  B_int <= to_integer(B);


  ADD: reg_adder_8b port map (A, B, add_sub, clk, Rst, c_out_dut,
      ovf_det_dut, S_dut);

  S_int_dut <= to_integer(S_dut);

  with add_sub select
    S_ref <=
      to_integer(A + B) when '0',
      to_integer(A - B) when others;

end architecture;
```

Code 2.2: reg_adder_8b_tb.vhd

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

entity adder_8b is
  generic (n : integer := 8);

  port ( A,B : in signed ( n-1 downto 0 );
         add_sub : in std_logic; --sub when add_sub == 1
         c_out : out std_logic;
         ovf_det : out std_logic;
         S : out signed ( n-1 downto 0 )
       );

end adder_8b;

architecture struct of adder_8b is

  component FA
   port( a,b : in std_logic;
          c_in : in std_logic;
          s,c_out: out std_logic
        );
  end component;

  signal carries : signed ( n downto 0);
  signal b_pos_neg : signed ( n-1 downto 0);


begin
  carries(0) <= add_sub;

  GEN: for i in 0 to n-1 generate

    b_pos_neg(i) <= add_sub xor B(i);
    FULL_ADDERS: FA port map( A(i), b_pos_neg(i), carries(i), S(i),
        carries(i+1));

    end generate;

  c_out <= carries(n);
  ovf_det <= carries(n-1) xor carries(n);

end architecture;
```

Code 2.3: adder_8b.vhd

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

entity adder_8b_tb is
end adder_8b_tb;


architecture struct of adder_8b_tb is

  component adder_8b
    generic (n : integer := 8);

     port ( A,B : in signed ( n-1 downto 0 );
         add_sub : in std_logic; --sub when add_sub == 1
         c_out : out std_logic;
         ovf_det : out std_logic;
         S : out signed ( n-1 downto 0 )
       );
  end component;

  signal A : signed ( 7 downto 0 );
  signal B : signed ( 7 downto 0 );
  signal A_int,B_int : integer;
  signal add_sub : std_logic;
  signal c_out_dut,ovf_det_dut : std_logic;
  signal S_dut : signed ( 7 downto 0 );
  signal S_ref,S_int_dut : integer;



begin

  add_sub <= '0', '1' after 16 ns;

  process begin
      for i in 2 to 4 loop

       A <= to_signed(63*i, A'length);

       for j in 0 to 2 loop

         B <= to_signed(61*j, B'length);
         wait for 2 ns;

       end loop;

     end loop;

    end process;  A_int <= to_integer(A);
  B_int <= to_integer(B);

  ADD: adder_8b port map (A,B,add_sub,c_out_dut,ovf_det_dut,S_dut);

  S_int_dut <= to_integer(S_dut);

  with add_sub select
    S_ref <=
      to_integer(A + B) when '0',
      to_integer(A - B) when others;

end architecture;
```

Code 2.4: adder_8b_tb.vhd

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

entity adder_display is

  --KEY -> clk, KEY0 -> RST, SW(16) -> add_sub,SW(15->8) -> A,
  --SW(7->0) ->B,LEDR ->S,LEDG8 -> ovf_det,
  --HEX7_6 -> A display, HEX5_4 -> B dspaly, HEX1_0 S_display
  port ( SW : in signed ( 16 downto 0 );
         KEY1,KEY0 : in std_logic;
         LEDG8 : out std_logic;
         LEDR : buffer signed ( 7 downto 0 );
         HEX7,HEX6,HEX5,HEX4,HEX1,HEX0 : out std_logic_vector(0 to 6)

       );
end adder_display;

architecture struct of adder_display is

  component reg_adder_8b
    generic (n : integer := 8);

    port ( R1,R2 : in signed ( n-1 downto 0 );
         add_sub : in std_logic;
         clk,Rst : in std_logic;
         c_out : out std_logic;
         ovf_det : out std_logic;
         S : out signed ( n-1 downto 0 )
       );
  end component;

  component hex_display
    port( SW : in signed (3 downto 0);
        HEXA : out std_logic_vector ( 0 to 6)
      );
  end component;

  signal c_out : std_logic;


begin

  ADD: reg_adder_8b port map(SW(15 downto 8), SW(7 downto 0), SW(16),
      KEY1, KEY0, c_out, LEDG8, LEDR);

  --A display
  HEXA7: hex_display port map(SW(15 downto 12),HEX7);
  HEXA6: hex_display port map(SW(11 downto 8),HEX6);

  --B display
  HEXA5: hex_display port map(SW(7 downto 4),HEX5);
  HEXA4: hex_display port map(SW(3 downto 0),HEX4);

  --S display
  HEXA1: hex_display port map(LEDR(7 downto 4),HEX1);
  HEXA0: hex_display port map(LEDR(3 downto 0),HEX0);


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;


```

```vhdl
architecture struct of adder_display_tb is

  component adder_display
    port (
         SW : in signed ( 16 downto 0 );
         KEY1,KEY0 : in std_logic;
         LEDG8 : out std_logic;
         LEDR : buffer signed ( 7 downto 0 );
         HEX7,HEX6,HEX5,HEX4,HEX1,HEX0 : out std_logic_vector(0 to 6)
       );
  end component;

  signal A : signed ( 7 downto 0 );
  signal B : signed ( 7 downto 0 );
  signal A_int,B_int : integer;
  signal A_B : signed (16 downto 0);
  signal add_sub,ovf_det_dut,clk,Rst : std_logic;
  signal S_dut : signed ( 7 downto 0 );
  signal S_int_dut,S_ref : integer;
  signal HEX7_dut,HEX6_dut,HEX5_dut,HEX4_dut,HEX1_dut,HEX0_dut :
      std_logic_vector(0 to 6);


begin

  process
  begin
    for i in 1 to 10 loop
      clk <= '0';
      wait for 2 ns;
      clk <= '1';
      wait for 2 ns;
    end loop;
  end process;

  add_sub <= '0', '1' after 16 ns;
  Rst <= '1','0' after 27 ns;

  A <= "10111010", "01110110" after 5 ns, "01111101" after 9 ns,
      "10101000" after 15 ns,
       "11110111" after 21 ns, "01000111" after 25 ns;

  B <= "00011010", "01100110" after 5 ns, "00011101" after 9 ns,
      "11111000" after 15 ns,
       "11110001" after 21 ns, "11010101" after 25 ns;

  A_int <= to_integer(A);
  B_int <= to_integer(B);

  A_B <= add_sub & A & B;

  ADD: adder_display port map (A_B, clk, Rst, ovf_det_dut, S_dut,
                               HEX7_dut,HEX6_dut,HEX5_dut,HEX4_dut,HEX1_dut,HEX0_dut);

  S_int_dut <= to_integer(S_dut);

  with add_sub select
    S_ref <=
      to_integer(A + B) when '0',
      to_integer(A - B) when others;

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**.

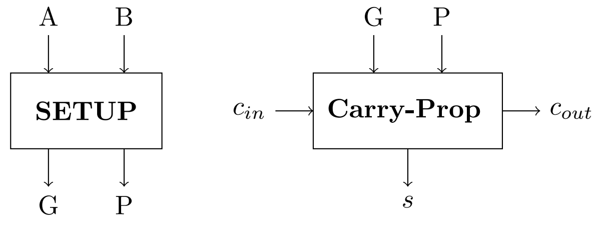
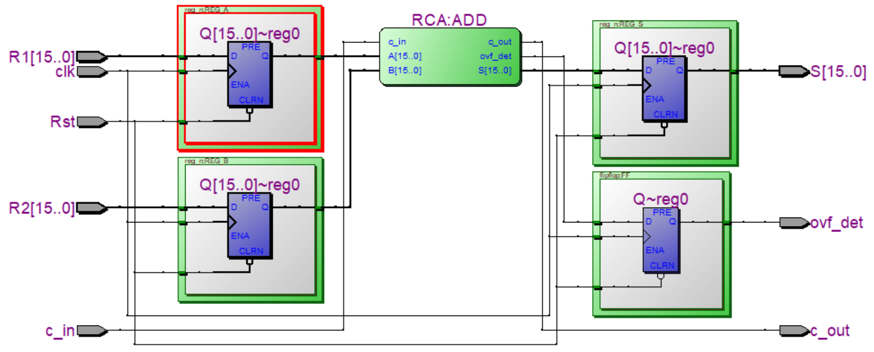


Figura 3.1: setup and carry-prop blocks

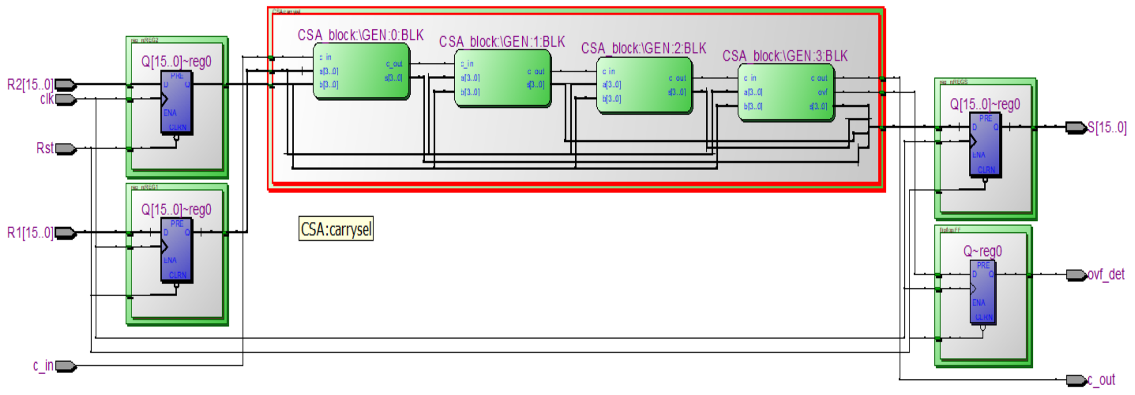
Nel caso del CSA, vengono usati due carry propagation che ricevono gli stessi ingressi e forniscono risultati diversi in virtù del c_{in}.

3.3 Risultati

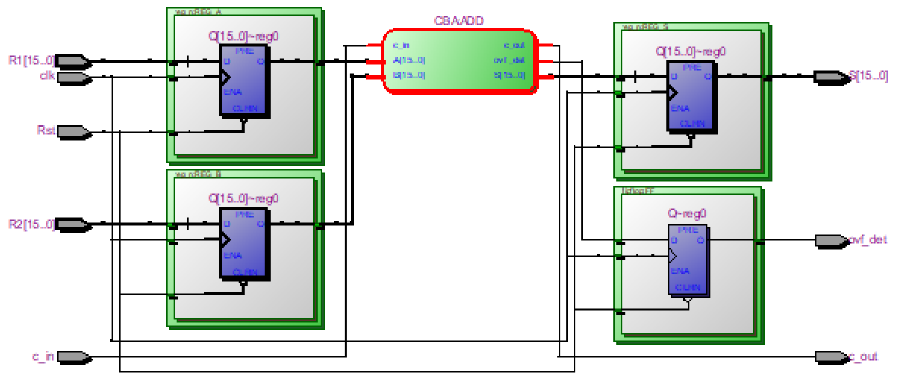
Di seguito sono mostrati i risultati prodotti da simulazione e sintesi.



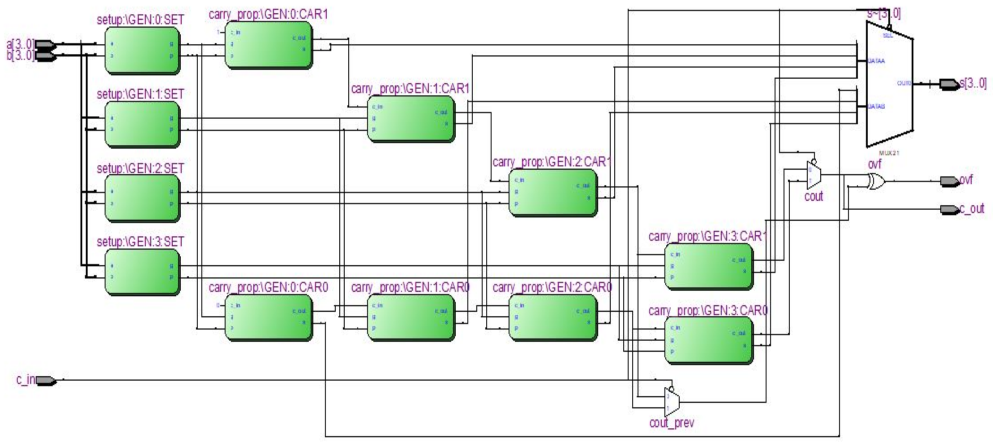
(a) RTL viewer RCA 16 bit



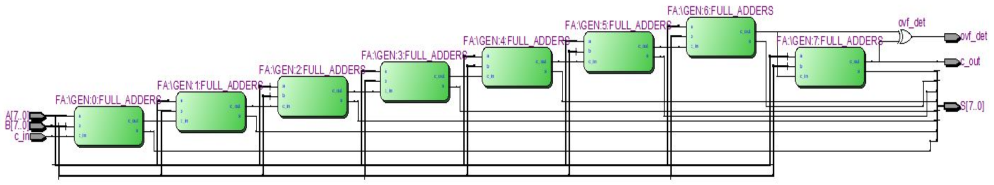
(b) RTL viewer CSA 16 bit



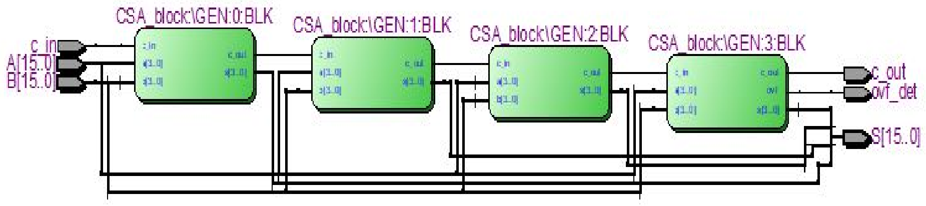
(c) RTL viewer BCA 16 bit



(d) RTL viewer CSA_block

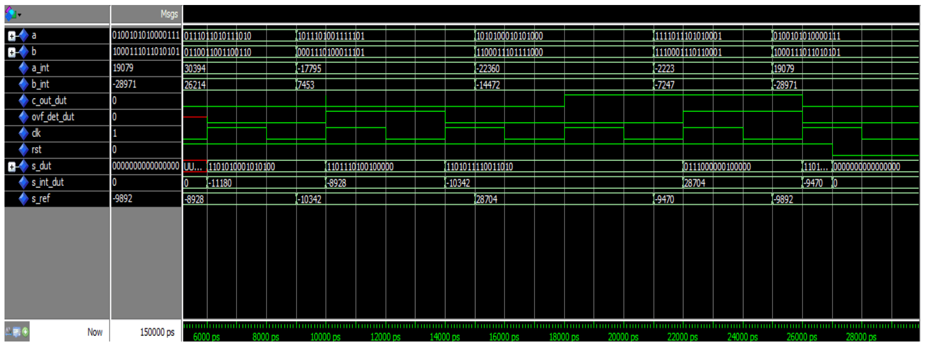


(e) RTL viewer RCA adder

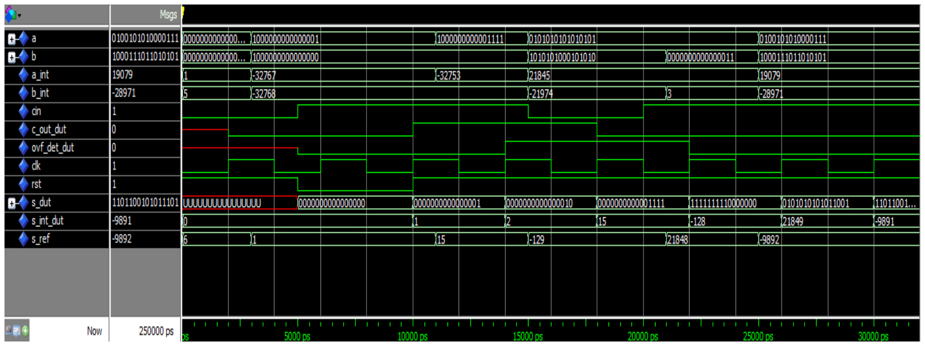


(f) RTL viewer CSA adder

Figura 3.2: RTL viewer exercise no. 3



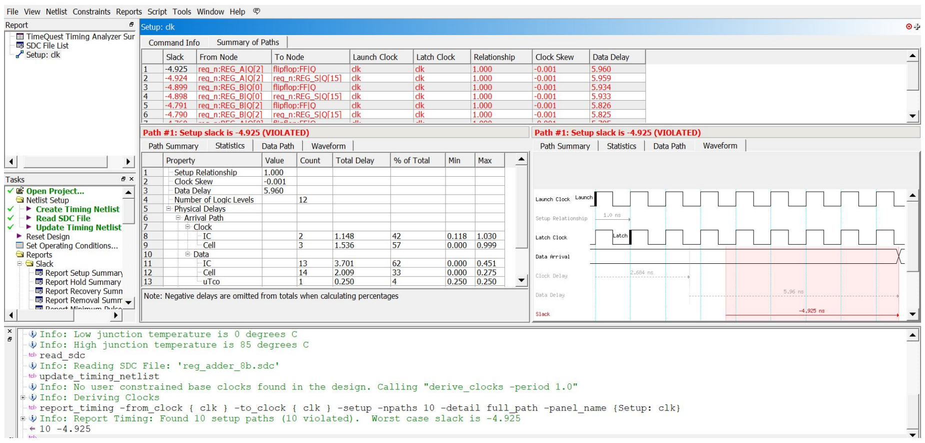
(a) wave CBA



(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



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-4.925	reg_n:REG_A|Q[2]	flipflop:FF|Q	clk	clk	1.000	-0.001	5.960
2	-4.924	reg_n:REG_A|Q[2]	reg_n:REG_S|Q[15]	clk	clk	1.000	-0.001	5.959
3	-4.899	reg_n:REG_B|Q[0]	flipflop:FF|Q	clk	clk	1.000	-0.001	5.934
4	-4.898	reg_n:REG_B|Q[0]	reg_n:REG_S|Q[15]	clk	clk	1.000	-0.001	5.933
5	-4.791	reg_n:REG_B|Q[2]	flipflop:FF|Q	clk	clk	1.000	-0.001	5.826
6	-4.790	reg_n:REG_B|Q[2]	reg_n:REG_S|Q[15]	clk	clk	1.000	-0.001	5.825

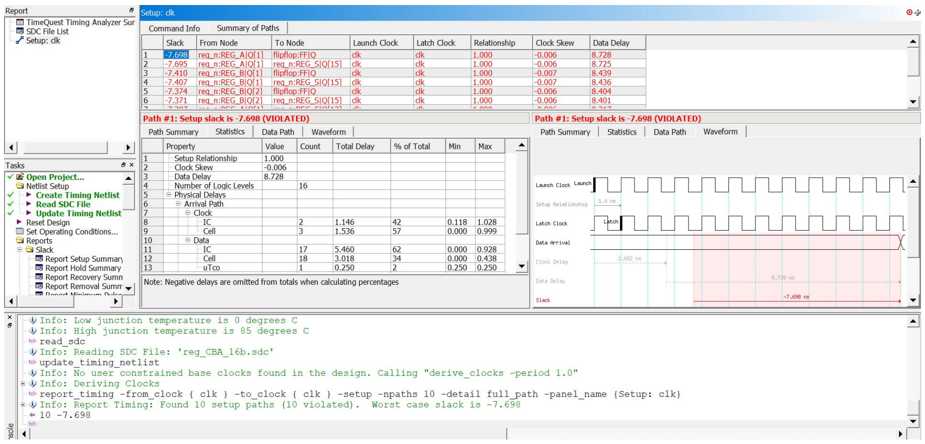
Path #1: Setup slack is -4.925 (VIOLATED)

	Property	Value	Count	Total Delay	% of Total	Min	Max
1	Setup Relationship	1.000					
2	Clock Skew	-0.001					
3	Data Delay	5.960					
4	Number of Logic Levels		12				
5	Physical Delays						
6	Arrival Path						
7	Clock						
8	IC		2	1.148	42	0.118	1.030
9	Cell		3	1.536	57	0.000	0.999
10	Data						
11	IC		13	3.701	62	0.000	0.451
12	Cell		14	2.009	33	0.000	0.275
13	uTco		1	0.250	4	0.250	0.250

Note: Negative delays are omitted from totals when calculating percentages

```
Info: Low junction temperature is 0 degrees C
Info: High junction temperature is 85 degrees C
read_sdc
Info: Reading SDC File: 'reg_adder_8b.sdc'
update_timing_netlist
Info: No user constrained base clocks found in the design. Calling "derive_clocks -period 1.0"
Info: Deriving Clocks
report_timing -from_clock { clk } -to_clock { clk } -setup -npaths 10 -detail full_path -panel_name {Setup: clk}
Info: Report Timing: Found 10 setup paths (10 violated).  Worst case slack is -4.925
10 -4.925
```

(a) time analysis RCA 16 bit



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-7.698	reg_n:REG_A|Q[1]	flipflop:FF|Q	clk	clk	1.000	-0.006	8.728
2	-7.695	reg_n:REG_A|Q[1]	reg_n:REG_S|Q[15]	clk	clk	1.000	-0.006	8.725
3	-7.410	reg_n:REG_B|Q[1]	flipflop:FF|Q	clk	clk	1.000	-0.007	8.439
4	-7.407	reg_n:REG_B|Q[1]	reg_n:REG_S|Q[15]	clk	clk	1.000	-0.007	8.436
5	-7.374	reg_n:REG_B|Q[2]	flipflop:FF|Q	clk	clk	1.000	-0.006	8.404
6	-7.371	reg_n:REG_B|Q[2]	reg_n:REG_S|Q[15]	clk	clk	1.000	-0.006	8.401

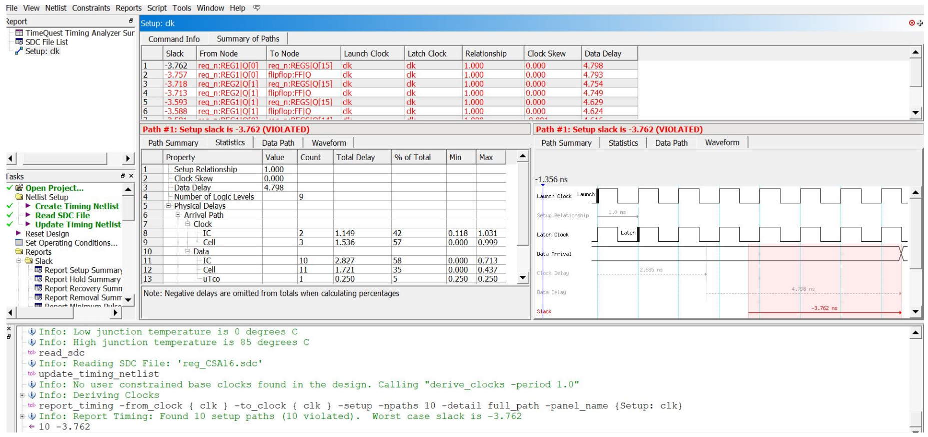
Path #1: Setup slack is -7.698 (VIOLATED)

	Property	Value	Count	Total Delay	% of Total	Min	Max
1	Setup Relationship	1.000					
2	Clock Skew	-0.006					
3	Data Delay	8.728					
4	Number of Logic Levels		16				
5	Physical Delays						
6	Arrival Path						
7	Clock						
8	IC		2	1.146	42	0.118	1.028
9	Cell		3	1.536	57	0.000	0.999
10	Data						
11	IC		17	5.460	62	0.000	0.928
12	Cell		18	3.018	34	0.000	0.438
13	uTco		1	0.250	2	0.250	0.250

Note: Negative delays are omitted from totals when calculating percentages

```
Info: Low junction temperature is 0 degrees C
Info: High junction temperature is 85 degrees C
read_sdc
Info: Reading SDC File: 'reg_CBA_16b.sdc'
update_timing_netlist
Info: No user constrained base clocks found in the design. Calling "derive_clocks -period 1.0"
Info: Deriving Clocks
report_timing -from_clock { clk } -to_clock { clk } -setup -npaths 10 -detail full_path -panel_name {Setup: clk}
Info: Report Timing: Found 10 setup paths (10 violated).  Worst case slack is -7.698
10 -7.698
```

(b) time analysis CBA 16 bit


Setup: clk

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-3.762	reg_n:REG1|Q[0]	reg_n:REGS|Q[15]	clk	clk	1.000	0.000	4.798
2	-3.757	reg_n:REG1|Q[0]	flipflop:FF|Q	clk	clk	1.000	0.000	4.793
3	-3.718	reg_n:REG2|Q[1]	reg_n:REGS|Q[15]	clk	clk	1.000	0.000	4.754
4	-3.713	reg_n:REG2|Q[1]	flipflop:FF|Q	clk	clk	1.000	0.000	4.749
5	-3.593	reg_n:REG1|Q[1]	reg_n:REGS|Q[15]	clk	clk	1.000	0.000	4.629
6	-3.588	reg_n:REG1|Q[1]	flipflop:FF|Q	clk	clk	1.000	0.000	4.624

Path #1: Setup slack is -3.762 (VIOLATED)

	Property	Value	Count	Total Delay	% of Total	Min	Max
1	Setup Relationship	1.000					
2	Clock Skew	0.000					
3	Data Delay	4.798					
4	Number of Logic Levels		9				
5	Physical Delays						
6	Arrival Path						
7	Clock						
8	IC		2	1.149	42	0.118	1.031
9	Cell		3	1.536	57	0.000	0.999
10	Data						
11	IC		10	2.827	58	0.000	0.713
12	Cell		11	1.721	35	0.000	0.437
13	uTco		1	0.250	5	0.250	0.250

Note: Negative delays are omitted from totals when calculating percentages

```
Info: Low junction temperature is 0 degrees C
Info: High junction temperature is 85 degrees C
read_sdc
Info: Reading SDC File: 'reg_CSA16.sdc'
update_timing_netlist
Info: No user constrained base clocks found in the design. Calling "derive_clocks -period 1.0"
Info: Deriving Clocks
report_timing -from_clock { clk } -to_clock { clk } -setup -npaths 10 -detail full_path -panel_name {Setup: clk}
Info: Report Timing: Found 10 setup paths (10 violated).  Worst case slack is -3.762
10 -3.762
```

(c) time analysis CSA 16 bit

Figura 3.4: time analysis exercise no. 3

Laddove si evincono, rispettivamente

$$
\begin{aligned}
f_{max} &= 168.8\,\text{MHz} \qquad & t_{del} &= 5.96\,\text{ns} \\
f_{max} &= 114.97\,\text{MHz} \qquad & t_{del} &= 8.73\,\text{ns} \\
f_{max} &= 210\,\text{MHz} \qquad & t_{del} &= 4.80\,\text{ns}
\end{aligned}
$$

3.4 Appendice

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

entity reg_CBA_16b is
  port ( R1,R2 : in signed ( 15 downto 0 );
         c_in : in std_logic;
         clk,Rst : in std_logic;
         c_out : out std_logic;
         ovf_det : out std_logic;
         S : out signed ( 15 downto 0 )
       );
end reg_CBA_16b;

architecture struct of reg_CBA_16b is

  component flipflop
    port (D, Clock, Resetn : in std_logic;
          Q : out std_logic
        );
  end component;


  component reg_n
    generic ( N : integer:=16);

    port (R : in signed(N-1 downto 0);
        Clock, Resetn : in std_logic;
        Q : out signed(N-1 downto 0)
        );
  end component;



    component CBA
      port(
          A,B : in signed( 15 downto 0 );
          c_in : in std_logic;
          S : out signed( 15 downto 0 );
          c_out,ovf_det : out std_logic
        );
    end component;

    signal A,B,Q : signed ( 15 downto 0 );
    signal ovf : std_logic;

begin

  REG_A: reg_n port map ( R1,clk,Rst,A);
  REG_B: reg_n port map ( R2,clk,Rst,B);

  ADD: CBA port map ( A,B,c_in,Q ,c_out,ovf);

  FF: flipflop port map ( ovf,clk,Rst,ovf_det );

  REG_S: reg_n port map (Q,clk,Rst,S);

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;
```

```vhdl



architecture struct of reg_CBA_16b_tb is

  component reg_CBA_16b


    port ( R1,R2 : in signed ( 15 downto 0 );
         c_in : in std_logic;
         clk,Rst : in std_logic;
         c_out : out std_logic;
         ovf_det : out std_logic;
         S : out signed ( 15 downto 0 )
       );

  end component;

  signal A,B : signed ( 15 downto 0 );
  signal A_int,B_int : integer;
  signal c_out_dut,ovf_det_dut,clk,Rst : std_logic;
  signal S_dut : signed ( 15 downto 0 );
  signal S_int_dut,S_ref : integer;


begin

  process begin
    for i in 1 to 10 loop
      clk <= '0';
      wait for 2 ns;
      clk <= '1';
      wait for 2 ns;
    end loop;
  end process;

  Rst <= '1','0' after 27 ns;

  A <= "1011101010111010", "0111011010111010" after 5 ns, "1011101001111101" after 9 ns, "1010100010101000" after 15 ns,
       "1111011101010001" after 21 ns, "0100101010000111" after 25 ns;

  B <= "0001100110011010", "0110011001100110" after 5 ns, "0001110100011101" after 9 ns, "1100011101111000" after 15 ns,
       "1110001110110001" after 21 ns, "1000111011010101" after 25 ns;

  A_int <= to_integer(A);
  B_int <= to_integer(B);


  ADD: reg_CBA_16b port map (A, B, '0', clk, Rst, c_out_dut,
      ovf_det_dut, S_dut);

  S_int_dut <= to_integer(S_dut);
  S_ref <= to_integer(A + B);

end architecture;
```

Code 3.2: reg_CBA_16b_tb.vhd

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


entity CBA is
  port(
        A,B : in signed( 15 downto 0 );
        c_in : in std_logic;
        S : out signed( 15 downto 0 );
        c_out,ovf_det : out std_logic
      );
end CBA;


architecture behaviour of CBA is

  component setup
    port( a,b : in std_logic;
        p,g : out std_logic
      );
  end component;


  component carry_prop
    port( p,g,c_in : in std_logic;
        s,c_out : out std_logic
      );
  end component;

  signal P,G : std_logic_vector ( 15 downto 0 );
  signal carries : std_logic_vector ( 20 downto 0 );
  signal sel : std_logic_vector ( 3 downto 0 );


begin

  carries(0) <= c_in;

  GEN: for i in 0 to 3 generate

  -- whole CBA ovf check
  HDHDH: for j in 0 to 3 generate

  SET: setup port map ( A(4*i+j),B(4*i+j),P(4*i+j),G(4*i+j) );

  CA_PROP: carry_prop port map ( P(4*i+j),G(4*i+j),carries(5*i+j),S(4*i
      +j),carries(5*i+j+1) );

    end generate;

   -- check on propagate value
    sel(i) <= P(4*i) and P(4*i+1) and P(4*i+2) and P(4*i+3);
    carries(i*5 + 5) <= carries(i*5+4) when (sel(i) = '0') else
        carries(i*5);

  end generate;

  c_out <= carries(20);
  ovf_det <= carries(18) xor carries(19);

end architecture;
```

Code 3.3: CBA.vhd

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

entity CBA_tb is
end CBA_tb;



architecture struct of CBA_tb is

  component CBA
   port(
      A,B : in signed( 15 downto 0 );
     c_in : in std_logic;
     S : out signed( 15 downto 0 );
     c_out,ovf_det : out std_logic
       );
  end component;

  signal A : signed ( 15 downto 0 );
  signal B : signed ( 15 downto 0 );
  signal A_int,B_int : integer;
```

```vhdl
  signal c_out_dut,ovf_det_dut : std_logic;
  signal S_dut : signed ( 15 downto 0 );
  signal S_ref,S_int_dut : integer;

begin
  process begin
      for i in 2 to 4 loop

        A <= to_signed(63*i, A'length);

        for j in 0 to 2 loop

          B <= to_signed(61*j, B'length);
          wait for 2 ns;

        end loop;

      end loop;

    end process;

A_int <= to_integer(A);
B_int <= to_integer(B);
BCA: CBA port map (A,B,'0',S_dut,c_out_dut,ovf_det_dut);

S_int_dut <= to_integer(S_dut);
S_ref <= to_integer(A + B) ;

end architecture;
```

Code 3.4: CBA_tb.vhd

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

entity reg_CSA16 is
  generic (n : integer := 16);
  port (
    R1,R2 : in signed ( n-1 downto 0 );
    c_in : in std_logic;
    clk,Rst : in std_logic;
    c_out : out std_logic;
    ovf_det : out std_logic;
    S : out signed ( n-1 downto 0 )
  );
end reg_CSA16;


```

```vhdl
architecture behavior of reg_CSA16 is

  component flipflop
    port (
      D, Clock, Resetn : in std_logic;
      Q : out std_logic
    );
  end component;

  component reg_n
    generic ( N : integer:=n);
    port (
      R : in signed(N-1 downto 0);
      Clock, Resetn : in std_logic;
      Q : out signed(N-1 downto 0)
    );
  end component;

  component CSA
      port(
        A,B : in signed( 15 downto 0 );
        c_in : in std_logic;
        S : out signed( 15 downto 0 );
        c_out,ovf_det : out std_logic
      );
  end component;

  signal a,b,q : signed(n-1 downto 0);
  signal ovfl : std_logic;

  begin

    REG1: reg_n port map (R1, clk, rst, a);
    REG2: reg_n port map (R2, clk, rst, b);
    carrysel: CSA port map (a, b, c_in, q, c_out, ovfl);
    FF: flipflop port map (ovfl, clk, rst, ovf_det);
    REGS: reg_n port map(q, clk, rst, S);

end behavior;
```

Code 3.5: reg_CSA16.vhd

```vhdl
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

  component reg_CSA16 is
  generic (n : integer := 16);
  port (
    R1,R2 : in signed ( n-1 downto 0 );
    c_in : in std_logic;
    clk,Rst : in std_logic;
    c_out : out std_logic;
    ovf_det : out std_logic;
    S : out signed ( n-1 downto 0 )
  );
 end component;

  signal A,B : signed ( 15 downto 0 );
  signal A_int,B_int : integer;
  signal cin, c_out_dut,ovf_det_dut,clk,Rst : std_logic;
  signal S_dut : signed ( 15 downto 0 );
  signal s_int_dut, S_ref : integer;


begin

  process begin
    for i in 1 to 10 loop
      clk <= '0';
      wait for 2 ns;
      clk <= '1';
      wait for 2 ns;
    end loop;
  end process;

  rst <= '1', '0' after 5 ns, '1' after 10 ns;

  --input values
  A <= "0000000000000001", "1000000000000001" after 3 ns,
      "1000000000001111" after 11 ns,
  "0101010101010101" after 15 ns, "0101010101010101" after 21 ns,
      "0100101010000111" after 25 ns;

  B <= "0000000000000101", "1000000000000000" after 3 ns,
      "1000000000000000" after 11 ns,
  "1010101000101010" after 15 ns, "0000000000000011" after 21 ns,
      "1000111011010101" after 25 ns;

  cin <= '0', '1' after 5 ns, '1' after 10 ns, '0' after 15 ns, '1'
      after 20 ns;

  DUT: reg_CSA16 port map (A, B, cin, clk, rst, c_out_dut, ovf_det_dut,
      s_dut);

  A_int <= to_integer(A);
  B_int <= to_integer(B);

  S_int_dut <= to_integer(S_dut);
  S_ref <= to_integer(A + B);

end test;
```

Code 3.6: reg_CSA16_tb.vhd

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


entity CSA is
  port(
        A,B : in signed( 15 downto 0 );
        c_in : in std_logic;
        S : out signed( 15 downto 0 );
        c_out,ovf_det : out std_logic
      );
end CSA;



architecture behaviour of CSA is

 component CSA_block
    port(
        a,b : in signed( 3 downto 0 );
        c_in : in std_logic;
        s : out signed( 3 downto 0 );
        c_out, ovf : out std_logic
      );
  end component;

  signal carries : std_logic_vector ( 4 downto 0 );
  signal ovfs : std_logic_vector(3 downto 0);

begin

  carries(0) <= c_in;

  GEN: for i in 0 to 3 generate

    BLK: CSA_block port map (A((i*4+3) downto i*4), B( (i*4+3) downto
        i*4), carries(i),
```

```vhdl
                            s((i*4+3) downto i*4),carries(i+1), ovfs(i));

  end generate;

  c_out <= carries(4);
  ovf_det <= ovfs(3);

end architecture;
```

Code 3.7: CSA.vhd

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

entity csa_tb is
end csa_tb;

architecture test of csa_tb is

  component CSA
     port(
        A,B : in signed( 15 downto 0 );
        c_in : in std_logic;
        S : out signed( 15 downto 0 );
        c_out,ovf_det : out std_logic
      );
    end component;

    signal in1, in2, sum, sum_ref : signed(15 downto 0);
    signal cin, cout, cout_ref, ovf, ovf_ref : std_logic;

    begin
      --input values
      in1 <= "0000000000000001", "1000000000000001" after 5 ns,
          "0000000000001111" after 10 ns,
      "0101010101010101" after 15 ns, "0101010101010101" after 20 ns;

      in2 <= "0000000000000101", "1000000000000000" after 5 ns,
          "1000000000000000" after 10 ns,
      "1010101000101010" after 15 ns, "0000000000000011" after 20 ns;

      cin <= '0', '1' after 5 ns, '1' after 10 ns, '0' after 15 ns, '1'
          after 20 ns;

      --expected outputs
      sum_ref <= "0000000000000110", "0000000000000010" after 5 ns,
          "1000000000010000" after 10 ns,
      "1111111101111111" after 15 ns, "0101010101011001" after 20 ns;

      cout_ref <= '0', '1' after 5 ns, '0' after 10 ns, '0' after 15 ns,
          '0' after 20 ns;

      ovf_ref <= '0', '1' after 5 ns, '0' after 10 ns, '0' after 15 ns,
          '0' after 20 ns;

      DUT: CSA port map (in1, in2, cin, sum, cout, ovf);

  end test;
```

Code 3.8: CSA_tb.vhd

```

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


entity CSA_block is
  port(
       a,b : in signed( 3 downto 0 );
       c_in : in std_logic;
       s : out signed( 3 downto 0 );
       c_out, ovf : out std_logic
     );
end CSA_block;

architecture behaviour of CSA_block is

  component setup
    port( a,b : in std_logic;
       p,g : out std_logic
     );
  end component;


  component carry_prop
    port( p,g, c_in : in std_logic;
       s,c_out : out std_logic
     );
  end component;


  signal p,g : std_logic_vector ( 3 downto 0 );
  signal carries_0,carries_1 : std_logic_vector ( 4 downto 0 );
  signal sum_0,sum_1 : signed ( 3 downto 0 );
  signal cout, cout_prev : std_logic;

```

```
begin

  carries_0(0) <= '0';
  carries_1(0) <= '1';

  GEN: for i in 0 to 3 generate

    SET: setup port map ( a(i), b(i), p(i), g(i) );
    CAR0: carry_prop port map ( p(i), g(i), carries_0(i), sum_0(i),
        carries_0(i+1));
    CAR1: carry_prop port map ( p(i), g(i), carries_1(i), sum_1(i),
        carries_1(i+1));

  end generate;

  cout <= carries_0(4) when (c_in = '0') else carries_1(4);
  cout_prev <= carries_0(3) when (c_in = '0') else carries_1(3);

  c_out <= cout;
  ovf <= cout xor cout_prev;

  s <= sum_0 when (c_in = '0') else sum_1;

end architecture;
```

Code 3.9: CSA_block.vhd

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

entity carry_prop is
  port( p,g,c_in : in std_logic;
        s,c_out : out std_logic
     );
end carry_prop;

architecture logic of carry_prop is

begin

  s <= c_in xor p;
  c_out <= g or ( p and c_in );

end architecture;
```

Code 3.10: carry_prop.vhd

```

```

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

entity setup is
  port( a,b : in std_logic;
        p,g : out std_logic
      );
end setup;

architecture logic of setup is

begin

  g <= a and b;
  p <= a xor b;

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)".

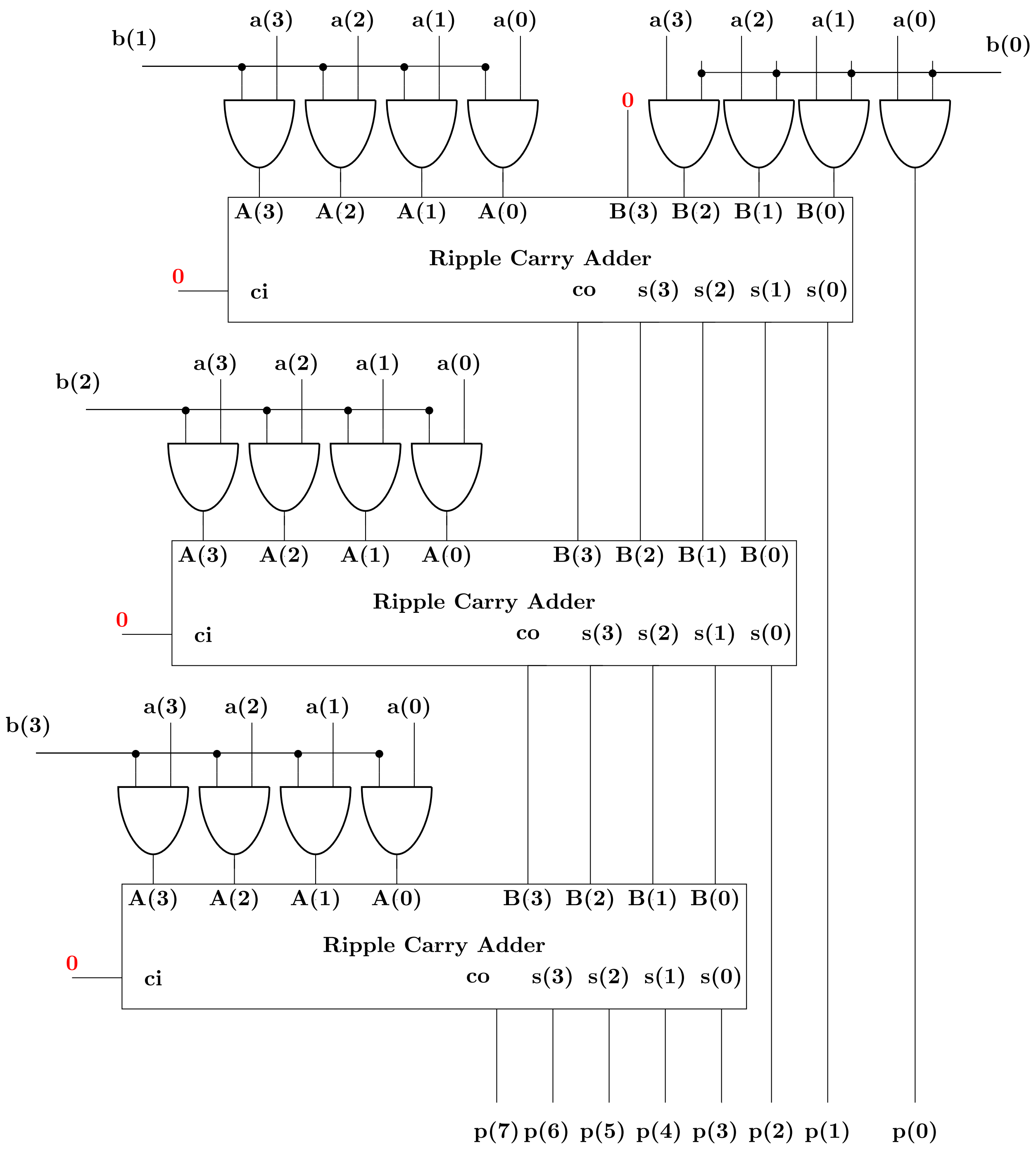


Figura 4.1: 4bit-Multiplier block scheme

4.3 Risultati

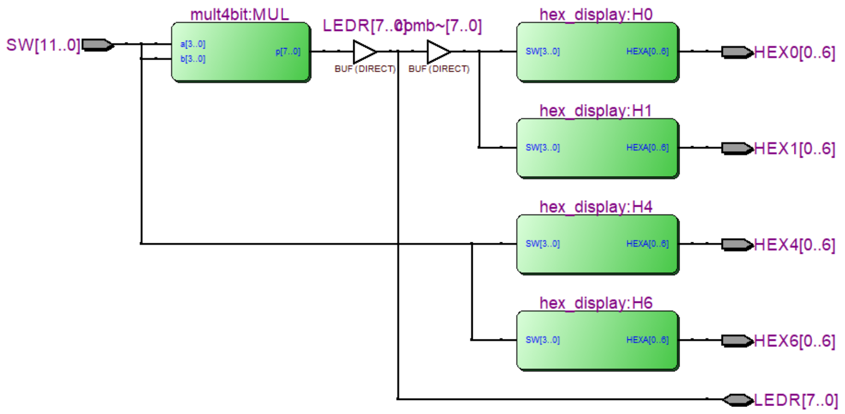


Figura 4.2: RTL viewer exercise no. 4

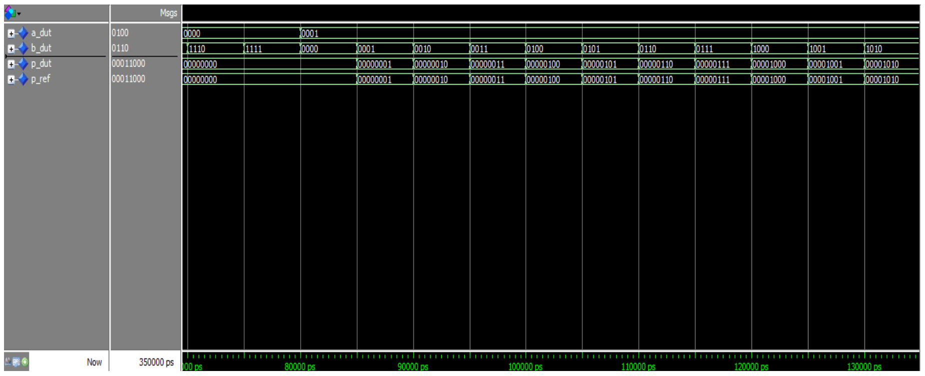


Figura 4.3: wave exercise no. 4

4.4 Appendice

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

entity mult4bit is
  port(
    a, b : in unsigned(3 downto 0);
    p : out unsigned (7 downto 0)
  );
end mult4bit;


architecture behaviour of mult4bit is
  component RCA
    port(
      a,b : in unsigned(3 downto 0);
      s : out unsigned(3 downto 0);
      cout : out std_logic
    );
  end component;

  signal AND0, AND1, AND2 : unsigned (3 downto 0); -- ANDs input to RCAs
  signal B0, B1, B2 : unsigned (3 downto 0); -- intermediate signals
      between RCAs
  begin


  B0 <= ('0', a(3) and b(0), a(2) and b(0), a(1) and b(0));

-- bitwise AND generating correct dimension range
-- for R values of AND operand
  AND0 <= a and (AND0'range => b(1));
  AND1 <= a and (AND1'range => b(2));
  AND2 <= a and (AND2'range => b(3));


 -- evaluating outputs
  p(0) <= a(0) and b(0);
  RCA0: RCA port map (
          a => AND0,
          b => B0,
          cout => B1(3),
          s(3 downto 1) => B1(2 downto 0),
          s(0) => p(1)
      );

  RCA1: RCA port map (
          a => AND1,
          b => B1,
          cout => B2(3),
          s(3 downto 1) => B2(2 downto 0),
          s(0) => p(2)
      );

 RCA2: RCA port map (
          a => AND2,
          b => B2,
          cout => p(7), -- last bit = carry out
          s => p(6 downto 3)
      );

end architecture;
```

Code 4.1: mult4bit.vhd

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

entity mult4bit_tb is
end mult4bit_tb;


architecture testcase of mult4bit_tb is
  component mult4bit is
    port(
       a, b : in unsigned(3 downto 0);
       p : out unsigned (7 downto 0)
    );
  end component;

  signal a_dut, b_dut : unsigned(3 downto 0);
  signal p_dut, p_ref : unsigned(7 downto 0);



  begin
    UUT : mult4bit port map (a_dut, b_dut, p_dut);

    -- test cases (all combinations of two 4 bit wide inputs)
    process begin
      for i in 0 to 15 loop

        a_dut <= to_unsigned(i, a_dut'length);

        for j in 0 to 15 loop
          b_dut <= to_unsigned(j, b_dut'length);
          wait for 0 ns;
          p_ref <= a_dut * b_dut; -- works bc they're both unsigned !!
          wait for 5 ns;
        end loop;

      end loop;

    end process;

end architecture;
```

Code 4.2: mult4bit_tb.vhd

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

entity mult4bit_synth is
  port(
     SW : in unsigned (11 downto 0);
     LEDR : buffer unsigned (7 downto 0);
     HEX6 : out std_logic_vector(0 to 6);
     HEX4 : out std_logic_vector(0 to 6);
     HEX1 : out std_logic_vector(0 to 6);
     HEX0 : out std_logic_vector(0 to 6)
  );
end mult4bit_synth;

architecture behaviour of mult4bit_synth is

  component mult4bit is
    port(
       a, b : in unsigned(3 downto 0);
       p : out unsigned (7 downto 0)
    );
  end component;

  component hex_display is
    port(
        SW : in unsigned (3 downto 0);
        HEXA : out std_logic_vector ( 0 to 6)
        );
  end component;

  begin
    MUL : mult4bit port map(SW(11 downto 8), SW(3 downto 0), LEDR);

    H6 : hex_display port map(SW(11 downto 8), HEX6);
    H4 : hex_display port map(SW(3 downto 0), HEX4);
    H1 : hex_display port map(LEDR (7 downto 4), HEX1); -- last 4 bits
        on HEX1
    H0 : hex_display port map(LEDR(3 downto 0), HEX0); -- first 4 bits
        on HEX0
end architecture;
```

Code 4.3: mult4b_synth.vhd