Appendice F

Listati VHDL

Listato F.1.1: nco.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity NCO is
5    port(
6         clear   : in STD_LOGIC;
7         clk           : in STD_LOGIC;
8         load    : in STD_LOGIC;
9         freq_word     : in STD_LOGIC_VECTOR (31 downto 0);
10        coseno  : out STD_LOGIC_VECTOR (11 downto 0);
11        seno    : out STD_LOGIC_VECTOR (11 downto 0)
12   );
13 end NCO;
14
15 architecture NCO of NCO is
16 ---- Signal declarations used on the diagram ----
17 signal COS           : STD_LOGIC_VECTOR (11 downto 0);
18 signal CTRL   : STD_LOGIC_VECTOR (1 downto 0);
19 signal CTRL_SEGNO_SENO : STD_LOGIC ;
20 signal FASE   : STD_LOGIC_VECTOR (12 downto 0);
21 signal SIN           : STD_LOGIC_VECTOR (11 downto 0);
22 signal THETA  : STD_LOGIC_VECTOR (31 downto 0);
23 signal THETA_TR      : STD_LOGIC_VECTOR (12 downto 0);
24 ---- Component declarations -----
25 component ACCUMULATOR
26   port (
27        clear   : in STD_LOGIC;
28        clk           : in STD_LOGIC;
29        freq_word     : in STD_LOGIC_VECTOR (31 downto 0);
30        load    : in STD_LOGIC;
31        theta   : inout STD_LOGIC_VECTOR (31 downto 0)
32   );
33 end component ;
34 component CORDIC_PIPELINED_UNROLLED
35   port (
36        clk           : in STD_LOGIC;
37        fase    : in STD_LOGIC_VECTOR (12 downto 0);
38        coseno  : out STD_LOGIC_VECTOR (11 downto 0);
39        seno    : out STD_LOGIC_VECTOR (11 downto 0)
40   );
41 end component ;
42 component COSINE_REBUILD
43   port (
44        ctrl    : in STD_LOGIC_VECTOR (1 downto 0);
45        in_cosine_rebuild : in STD_LOGIC_VECTOR (11 downto 0);
46        out_cosine_rebuild : out STD_LOGIC_VECTOR (11 downto 0)
47   );
48 end component ;
49 component DELAY_13
50   port (
51        clk             : in STD_LOGIC;
52        in_delay_13 : in STD_LOGIC;
53        out_delay_13 : out STD_LOGIC
54   );
55 end component ;
56 component SINE_REBUILD
57   port (
58        ctrl         : in STD_LOGIC;
59        in_sine_rebuild : in STD_LOGIC_VECTOR (11 downto 0);
60        out_sine_rebuild : out STD_LOGIC_VECTOR (11 downto 0)
61   );
```

```
62  end component ;
63  component TO_FIRST_QUADRANT
64    port (
65         in_tfq  : in STD_LOGIC_VECTOR (12 downto 0);
66         out_tfq  : out STD_LOGIC_VECTOR (12 downto 0)
67    );
68  end component ;
69  component TRONCATORE_12
70    port (
71         ingresso       : in STD_LOGIC_VECTOR (31 downto 0);
72         uscita  : out STD_LOGIC_VECTOR (12 downto 0)
73    );
74  end component ;
75
76  begin
77  ---- Component instantiations ----
78  U0 : CORDIC_PIPELINED_UNROLLED
79    port map(
80         clk           => clk,
81         coseno  => cos,
82         fase    => fase,
83         seno    => sin
84    );
85  U1 : DELAY_13
86    port map(
87         clk             => clk,
88         in_delay_13 => Theta_tr(11),
89         out_delay_13 => ctrl(0)
90    );
91  U2 : DELAY_13
92    port map(
93         clk             => clk,
94         in_delay_13 => Theta_tr(12),
95         out_delay_13 => ctrl(1)
96    );
97  U3 : DELAY_13
98    port map(
99         clk             => clk,
100        in_delay_13 => Theta_tr(12),
101        out_delay_13 => ctrl_segno_seno
102   );
103 U5 : TRONCATORE_12
104   port map(
105        ingresso   => Theta,
106        uscita   => Theta_tr
107   );
108 U4 : ACCUMULATOR
109   port map(
110        clear    => clear,
111        clk             => clk,
112        freq_word       => freq_word,
113        load     => load,
114        theta    => Theta
115   );
116 U6 : TO_FIRST_QUADRANT
117   port map(
118        in_tfq   => Theta_tr,
119        out_tfq  => fase
120   );
121 U8 : COSINE_REBUILD
122   port map(
123        ctrl            => ctrl,
124        in_cosine_rebuild => cos,
125        out_cosine_rebuild => coseno
126   );
127 U7 : SINE_REBUILD
128   port map(
129        ctrl            => ctrl_segno_seno,
130        in_sine_rebuild => sin,
131        out_sine_rebuild => seno
132   );
133 end NCO;
```

Listato F.1.2: accumulator.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   use ieee.std_logic_signed.all;
5
6   entity accumulator is
7         port ( freq_word          :      in std_logic_vector(31 downto 0);
8                clk, load, clear   :      in std_logic;
9                theta              : inout std_logic_vector(31 downto 0)
10               );
11  end accumulator;
12
13  architecture acc_arch of accumulator is
```

```
14          signal reg_theta    :      std_logic_vector(31 downto 0);
15          begin
16          process(load, clear, freq_word, theta)
17                  begin
18                  if load='1' then
19                          reg_theta <= freq_word;
20                  else if clear='1' then
21                          reg_theta <= "00000000000000000000000000000000";
22                  else
23                          reg_theta <= freq_word + theta;
24                  end if;
25                  end if;
26          end process;
27
28          process(clk)
29                  begin
30                  if clk'event and clk='1' then
31                          theta <= reg_theta;
32                  end if;
33          end process;
34  end acc_arch;
```

Listato F.1.3: cordic_pipelined_unrolled.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   entity cordic_pipelined_unrolled is
5     port(
6          clk            : in STD_LOGIC;
7          fase    : in STD_LOGIC_VECTOR (12 downto 0);
8          coseno  : out STD_LOGIC_VECTOR (11 downto 0);
9          seno    : out STD_LOGIC_VECTOR (11 downto 0)
10    );
11  end cordic_pipelined_unrolled;
12
13  architecture CORDIC_PIPELINED_UNROLLED of cordic_pipelined_unrolled is
14  signal A_0 : STD_LOGIC_VECTOR (12 downto 0);
15  signal A_1 : STD_LOGIC_VECTOR (12 downto 0);
16  signal A_10 : STD_LOGIC_VECTOR (12 downto 0);
17  signal A_11 : STD_LOGIC_VECTOR (12 downto 0);
18  signal A_12 : STD_LOGIC_VECTOR (12 downto 0);
19  signal A_2 : STD_LOGIC_VECTOR (12 downto 0);
20  signal A_3 : STD_LOGIC_VECTOR (12 downto 0);
21  signal A_4 : STD_LOGIC_VECTOR (12 downto 0);
22  signal A_5 : STD_LOGIC_VECTOR (12 downto 0);
23  signal A_6 : STD_LOGIC_VECTOR (12 downto 0);
24  signal A_7 : STD_LOGIC_VECTOR (12 downto 0);
25  signal A_8 : STD_LOGIC_VECTOR (12 downto 0);
26  signal A_9 : STD_LOGIC_VECTOR (12 downto 0);
27  signal X0 : STD_LOGIC_VECTOR (11 downto 0);
28  signal X1 : STD_LOGIC_VECTOR (11 downto 0);
29  signal X10 : STD_LOGIC_VECTOR (11 downto 0);
30  signal X11 : STD_LOGIC_VECTOR (11 downto 0);
31  signal X12 : STD_LOGIC_VECTOR (11 downto 0);
32  signal X13 : STD_LOGIC_VECTOR (11 downto 0);
33  signal X2 : STD_LOGIC_VECTOR (11 downto 0);
34  signal X3 : STD_LOGIC_VECTOR (11 downto 0);
35  signal X4 : STD_LOGIC_VECTOR (11 downto 0);
36  signal X5 : STD_LOGIC_VECTOR (11 downto 0);
37  signal X6 : STD_LOGIC_VECTOR (11 downto 0);
38  signal X7 : STD_LOGIC_VECTOR (11 downto 0);
39  signal X8 : STD_LOGIC_VECTOR (11 downto 0);
40  signal X9 : STD_LOGIC_VECTOR (11 downto 0);
41  signal Y0 : STD_LOGIC_VECTOR (11 downto 0);
42  signal Y1 : STD_LOGIC_VECTOR (11 downto 0);
43  signal Y10 : STD_LOGIC_VECTOR (11 downto 0);
44  signal Y11 : STD_LOGIC_VECTOR (11 downto 0);
45  signal Y12 : STD_LOGIC_VECTOR (11 downto 0);
46  signal Y13 : STD_LOGIC_VECTOR (11 downto 0);
47  signal Y2 : STD_LOGIC_VECTOR (11 downto 0);
48  signal Y3 : STD_LOGIC_VECTOR (11 downto 0);
49  signal Y4 : STD_LOGIC_VECTOR (11 downto 0);
50  signal Y5 : STD_LOGIC_VECTOR (11 downto 0);
51  signal Y6 : STD_LOGIC_VECTOR (11 downto 0);
52  signal Y7 : STD_LOGIC_VECTOR (11 downto 0);
53  signal Y8 : STD_LOGIC_VECTOR (11 downto 0);
54  signal Y9 : STD_LOGIC_VECTOR (11 downto 0);
55  signal Z0 : STD_LOGIC_VECTOR (12 downto 0);
56  signal Z1 : STD_LOGIC_VECTOR (12 downto 0);
57  signal Z10 : STD_LOGIC_VECTOR (12 downto 0);
58  signal Z11 : STD_LOGIC_VECTOR (12 downto 0);
59  signal Z12 : STD_LOGIC_VECTOR (12 downto 0);
60  signal Z2 : STD_LOGIC_VECTOR (12 downto 0);
61  signal Z3 : STD_LOGIC_VECTOR (12 downto 0);
62  signal Z4 : STD_LOGIC_VECTOR (12 downto 0);
```

```
63  signal Z5 : STD_LOGIC_VECTOR (12 downto 0);
64  signal Z6 : STD_LOGIC_VECTOR (12 downto 0);
65  signal Z7 : STD_LOGIC_VECTOR (12 downto 0);
66  signal Z8 : STD_LOGIC_VECTOR (12 downto 0);
67  signal Z9 : STD_LOGIC_VECTOR (12 downto 0);
68  ---- Component declarations -----
69  component CORDIC_BASE_J
70    generic( n : INTEGER );
71    port (
72         a_cost_j : in STD_LOGIC_VECTOR (12 downto 0);
73         clk : in STD_LOGIC;
74         in_x : in STD_LOGIC_VECTOR (11 downto 0);
75         in_y : in STD_LOGIC_VECTOR (11 downto 0);
76         in_z : in STD_LOGIC_VECTOR (12 downto 0);
77         coseno : out STD_LOGIC_VECTOR (11 downto 0);
78         fase_j : out STD_LOGIC_VECTOR (12 downto 0);
79         seno : out STD_LOGIC_VECTOR (11 downto 0)
80    );
81  end component ;
82
83  begin
84  x0   <= "010011011100" ;
85  y0   <= "000000000000" ;
86  a_0  <= "0010000000000" ;
87  a_1  <= "0001001011100" ;
88  a_2  <= "0000100111111" ;
89  a_3  <= "0000010100010" ;
90  a_4  <= "0000001010001" ;
91  a_5  <= "0000000101000" ;
92  a_6  <= "0000000010100" ;
93  a_7  <= "0000000001010" ;
94  a_8  <= "0000000000101" ;
95  a_9  <= "0000000000010" ;
96  a_10 <= "0000000000001" ;
97  a_11 <= "0000000000000" ;
98  a_12 <= "0000000000000" ;
99  ---- Component instantiations ----
100 U0 : CORDIC_BASE_J
101   generic map ( n => 0 )
102   port map(
103        a_cost_j       => a_0,
104        clk            => clk,
105        coseno  => x1,
106        fase_j  => z1,
107        in_x    => x0,
108        in_y    => y0,
109        in_z    => z0,
110        seno    => y1
111   );
112 U1 : CORDIC_BASE_J
113   generic map ( n => 1 )
114   port map(
115        a_cost_j       => a_1,
116        clk            => clk,
117        coseno  => x2,
118        fase_j  => z2,
119        in_x    => x1,
120        in_y    => y1,
121        in_z    => z1,
122        seno    => y2
123   );
124 U2 : CORDIC_BASE_J
125   generic map ( n => 2 )
126   port map(
127        a_cost_j       => a_2,
128        clk            => clk,
129        coseno  => x3,
130        fase_j  => z3,
131        in_x    => x2,
132        in_y    => y2,
133        in_z    => z2,
134        seno    => y3
135   );
136 U3 : CORDIC_BASE_J
137   generic map ( n => 3 )
138   port map(
139        a_cost_j       => a_3,
140        clk            => clk,
141        coseno  => x4,
142        fase_j  => z4,
143        in_x    => x3,
144        in_y    => y3,
145        in_z           => z3,
146        seno    => y4
147   );
148 U4 : CORDIC_BASE_J
149   generic map ( n => 4 )
150   port map(
151        a_cost_j       => a_4,
152        clk            => clk,
153        coseno  => x5,
154        fase_j  => z5,
155        in_x    => x4,
156        in_y    => y4,
157        in_z    => z4,
```

```
158         seno    => y5
159     );
160   U5 : CORDIC_BASE_J
161     generic map ( n => 5 )
162     port map(
163         a_cost_j      => a_5,
164         clk           => clk,
165         coseno  => x6,
166         fase_j  => z6,
167         in_x    => x5,
168         in_y    => y5,
169         in_z    => z5,
170         seno    => y6
171     );
172   U6 : CORDIC_BASE_J
173     generic map ( n => 6 )
174     port map(
175         a_cost_j      => a_6,
176         clk           => clk,
177         coseno  => x7,
178         fase_j  => z7,
179         in_x    => x6,
180         in_y    => y6,
181         in_z    => z6,
182         seno    => y7
183     );
184   U7 : CORDIC_BASE_J
185     generic map ( n => 7 )
186     port map(
187         a_cost_j      => a_7,
188         clk           => clk,
189         coseno  => x8,
190         fase_j  => z8,
191         in_x    => x7,
192         in_y    => y7,
193         in_z    => z7,
194         seno    => y8
195     );
196   U8 : CORDIC_BASE_J
197     generic map ( n => 8 )
198     port map(
199         a_cost_j      => a_8,
200         clk           => clk,
201         coseno  => x9,
202         fase_j  => z9,
203         in_x    => x8,
204         in_y    => y8,
205         in_z    => z8,
206         seno    => y9
207     );
208   U9 : CORDIC_BASE_J
209     generic map ( n => 9 )
210     port map(
211         a_cost_j      => a_9,
212         clk           => clk,
213         coseno  => x10,
214         fase_j  => z10,
215         in_x    => x9,
216         in_y    => y9,
217         in_z    => z9,
218         seno    => y10
219     );
220   U10 : CORDIC_BASE_J
221     generic map ( n => 10 )
222     port map(
223         a_cost_j      => a_10,
224         clk           => clk,
225         coseno  => x11,
226         fase_j  => z11,
227         in_x    => x10,
228         in_y    => y10,
229         in_z    => z10,
230         seno    => y11
231     );
232   U11 : CORDIC_BASE_J
233     generic map ( n => 11 )
234     port map(
235         a_cost_j      => a_11,
236         clk           => clk,
237         coseno  => x12,
238         fase_j  => z12,
239         in_x    => x11,
240         in_y    => y11,
241         in_z    => z11,
242         seno    => y12
243     );
244   U12 : CORDIC_BASE_J
245     generic map ( n => 12 )
246     port map(
247         a_cost_j      => a_12,
248         clk           => clk,
249         coseno  => x13,
250         in_x    => x12,
251         in_y    => y12,
252         in_z    => z12,
```

```
253         seno    => y13
254     );
255       z0  <= FASE;
256       COSENO    <= x13;
257       SENO      <= y13;
258   end CORDIC_PIPELINED_UNROLLED;
```

Listato F.1.4: cordic_base_j.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   entity cordic_base_j is
5     generic( n : integer );
6     port(
7         clk             : in STD_LOGIC;
8         a_cost_j        : in STD_LOGIC_VECTOR (12 downto 0);
9         in_x    : in STD_LOGIC_VECTOR (11 downto 0);
10        in_y    : in STD_LOGIC_VECTOR (11 downto 0);
11        in_z    : in STD_LOGIC_VECTOR (12 downto 0);
12        coseno  : out STD_LOGIC_VECTOR (11 downto 0);
13        fase_j  : out STD_LOGIC_VECTOR (12 downto 0);
14        seno    : out STD_LOGIC_VECTOR (11 downto 0)
15    );
16  end cordic_base_j;
17
18  architecture CORDIC_BASE_J of cordic_base_j is
19  signal SGN              : STD_LOGIC ;
20  signal SHIFTER_TO_ADDER_X : STD_LOGIC_VECTOR (11 downto 0);
21  signal SHIFTER_TO_ADDER_Y : STD_LOGIC_VECTOR (11 downto 0);
22  signal X_OUT_REG     : STD_LOGIC_VECTOR (11 downto 0);
23  signal Y_OUT_REG     : STD_LOGIC_VECTOR (11 downto 0);
24  signal Z_OUT_REG     : STD_LOGIC_VECTOR (12 downto 0);
25  component ADDER_12
26    port (
27        in_a    : in STD_LOGIC_VECTOR (11 downto 0);
28        in_b    : in STD_LOGIC_VECTOR (11 downto 0);
29        sgn             : in STD_LOGIC;
30        out_adder_12 : out STD_LOGIC_VECTOR (11 downto 0)
31    );
32  end component ;
33  component ADDER_13
34    port (
35        in_a    : in STD_LOGIC_VECTOR (12 downto 0);
36        in_b    : in STD_LOGIC_VECTOR (12 downto 0);
37        sgn             : in STD_LOGIC;
38        out_adder_13 : out STD_LOGIC_VECTOR (12 downto 0)
39    );
40  end component ;
41  component REG_12
42    port (
43        clk             : in STD_LOGIC;
44        in_reg_12       : in STD_LOGIC_VECTOR (11 downto 0);
45        out_reg_12 : out STD_LOGIC_VECTOR (11 downto 0)
46    );
47  end component ;
48  component REG_13
49    port (
50        clk             : in STD_LOGIC;
51        in_reg_13       : in STD_LOGIC_VECTOR (12 downto 0);
52        out_reg_13 : out STD_LOGIC_VECTOR (12 downto 0)
53    );
54  end component ;
55  component SHIFTER
56    generic( n : INTEGER );
57    port (
58        in_shifter : in STD_LOGIC_VECTOR (11 downto 0);
59        out_shifter : out STD_LOGIC_VECTOR (11 downto 0)
60    );
61  end component ;
62
63  begin
64  sgn <= not(z_out_reg(12)) ;
65  U0 : REG_12
66    port map(
67        clk             => clk,
68        in_reg_12       => in_x,
69        out_reg_12 => x_out_reg
70    );
71  U1 : REG_12
72    port map(
73        clk             => clk,
74        in_reg_12       => in_y,
75        out_reg_12 => y_out_reg
76    );
77  U2 : SHIFTER
78    generic map ( n => n )
```

```
79    port map(
80         in_shifter => x_out_reg,
81         out_shifter => shifter_to_adder_y
82    );
83  U4 : ADDER_12
84    port map(
85         in_a      => x_out_reg,
86         in_b      => shifter_to_adder_x,
87         out_adder_12 => coseno,
88         sgn              => z_out_reg(12)
89    );
90  U5 : ADDER_12
91    port map(
92         in_a      => y_out_reg,
93         in_b      => shifter_to_adder_y,
94         out_adder_12 => seno,
95         sgn              => sgn
96    );
97  U7 : ADDER_13
98    port map(
99         in_a      => z_out_reg,
100        in_b      => a_cost_j,
101        out_adder_13 => fase_j,
102        sgn              => z_out_reg(12)
103   );
104 U6 : REG_13
105   port map(
106        clk => clk,
107        in_reg_13        => in_z,
108        out_reg_13 => z_out_reg
109   );
110 U3 : SHIFTER
111   generic map ( n => n )
112   port map(
113        in_shifter => y_out_reg,
114        out_shifter => shifter_to_adder_x
115   );
116 end CORDIC_BASE_J;
```

Listato F.1.5: adder_12.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_signed.all;
4
5  entity adder_12 is
6         port(in_a , in_b      :    in std_logic_vector(11 downto 0);
7              sgn              :    in std_logic;
8              out_adder_12     :    out std_logic_vector(11 downto 0));
9  end adder_12;
10
11 architecture adder_12_arch of adder_12 is
12        begin
13        process(in_a , in_b , sgn)
14                begin
15                if sgn = '0' then
16                        out_adder_12 <= in_a - in_b ;
17                else
18                        out_adder_12 <= in_a + in_b ;
19                end if;
20        end process;
21 end adder_12_arch;
```

Listato F.1.6: adder_13.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_signed.all;
4
5  entity adder_13 is
6         port(in_a , in_b      :    in std_logic_vector(12 downto 0);
7                sgn            :    in std_logic;
8                out_adder_13 : out std_logic_vector(12 downto 0));
9  end adder_13;
10
11 architecture adder_13_arch of adder_13 is
12        begin
13        process(in_a , in_b , sgn)
14                begin
15                if sgn = '0' then
16                        out_adder_13 <= in_a - in_b ;
17                else
18                        out_adder_13 <= in_a + in_b ;
19                end if;
```

```
20        end process;
21  end adder_13_arch ;
```

Listato F.1.7: reg_12.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity reg_12 is
5         port( in_reg_12      :     in std_logic_vector(11 downto 0);
6               clk            :     in std_logic;
7               out_reg_12 : out std_logic_vector(11 downto 0));
8   end reg_12;
9
10  architecture reg_12_arch of reg_12 is
11        begin
12        process(clk , in_reg_12)
13              begin
14              if rising_edge(clk) then
15                    out_reg_12 <= in_reg_12;
16              end if;
17        end process;
18  end reg_12_arch ;
```

Listato F.1.8: reg_13.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity reg_13 is
5         port( in_reg_13       :     in std_logic_vector(12 downto 0);
6               clk             :     in std_logic;
7               out_reg_13 : out std_logic_vector(12 downto 0)
8         );
9   end reg_13;
10
11  architecture reg_13_arch of reg_13 is
12        begin
13        process(clk , in_reg_13)
14              begin
15              if rising_edge(clk) then
16                    out_reg_13 <= in_reg_13;
17              end if;
18        end process;
19  end reg_13_arch ;
```

Listato F.1.9: shifter.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity shifter is
5         generic(n:integer);
6         port( in_shifter :  in std_logic_vector(11 downto 0);
7               out_shifter : out std_logic_vector(11 downto 0));
8   end shifter;
9
10  architecture shifter_arch of shifter is
11        begin
12              process(in_shifter)
13                    variable i:integer;
14                    begin
15                          if n < 12 then
16                                out_shifter(11-n downto 0) <= in_shifter(11
                                    downto n);
17                                for i in 11 downto (11-n) loop
18                                      if in_shifter(11)='0' then
19                                            out_shifter(11 downto 11-n) <= (
                                                others=>'0');
20                                      else
21                                            out_shifter(11 downto 11-n) <= (
                                                others=>'1');
22                                      end if;
23                                end loop;
24                          else out_shifter <= "000000000000" ;
25                          end if;
26              end process;
```

```
27  end shifter_arch ;
```

Listato F.1.10: cosine_rebuild.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_signed.all;
4
5   entity cosine_rebuild is
6         port( in_cosine_rebuild    :      in  std_logic_vector(11 downto 0);
7               ctrl                 :      in  std_logic_vector(1 downto 0);
8               out_cosine_rebuild :      out std_logic_vector(11 downto 0));
9   end cosine_rebuild ;
10
11  architecture cosine_rebuild_arch of cosine_rebuild is
12  begin
13        process(in_cosine_rebuild, ctrl)
14              variable temp : std_logic_vector(11 downto 0);
15        begin
16              temp(11 downto 0) := in_cosine_rebuild ;
17              case ctrl is
18                    when "00" => null ; -- primo quadrante
19                    when "01" => temp := ("111111111111" xor temp)+1; -- secondo
                          quadrante
20                    when "10" => temp := ("111111111111" xor temp)+1; -- terzo
                          quadrante
21                    when "11" => null ; -- quarto quadrante
22                    when others => temp := "000000000000" ;
23              end case;
24              out_cosine_rebuild(11 downto 0) <= temp(11 downto 0) ;
25        end process;
26  end cosine_rebuild_arch ;
```

Listato F.1.11: delay_13.vhd

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3
4   entity delay_13 is
5           port (
6                   clk          : in STD_LOGIC;
7         in_delay_13    : in STD_LOGIC;
8         out_delay_13   : out STD_LOGIC
9                 );
10  end delay_13;
11
12  architecture delay_13_arch of delay_13 is
13    signal reg_int : STD_LOGIC_VECTOR(12 downto 0);
14  begin
15    process (clk)
16    begin
17        if clk'event and clk='1' then
18              reg_int <= in_delay_13 & reg_int(12 downto 1);
19        end if;
20    end process;
21        out_delay_13 <= reg_int(0);
22  end delay_13_arch;
```

Listato F.1.12: sine_rebuild.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_signed.all;
4
5   entity sine_rebuild is
6         port( in_sine_rebuild :    in  std_logic_vector(11 downto 0);
7               ctrl            :    in  std_logic;
8               out_sine_rebuild : out std_logic_vector(11 downto 0));
9   end sine_rebuild ;
10
11  architecture sine_rebuild_arch of sine_rebuild is
12  begin
13        process(in_sine_rebuild, ctrl)
14              variable temp : std_logic_vector(11 downto 0);
15        begin
16              temp(11 downto 0) := in_sine_rebuild ;
17              case ctrl is
```

```
18                      when '1'   => temp := ("111111111111" xor temp)+1 ;
19                      when '0'   => null ;
20                      when others => temp := "000000000000" ;
21                end case;
22                out_sine_rebuild(11 downto 0) <= temp(11 downto 0) ;
23          end process;
24    end sine_rebuild_arch ;
```

Listato F.1.13: to_first_quadrant.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_signed.all;
4
5   entity to_first_quadrant is
6         port( in_tfq  :      in  std_logic_vector(12 downto 0);
7                 out_tfq     :      out std_logic_vector(12 downto 0));
8   end to_first_quadrant;
9
10  architecture to_first_quadrant_arch of to_first_quadrant is
11  begin
12        process(in_tfq)
13              variable temp : std_logic_vector(10 downto 0);
14        begin
15              temp(10 downto 0) := in_tfq(10 downto 0) ;
16              case in_tfq(11) is
17                    when '1' => temp := ("11111111111" xor temp)+1 ;
18                    when '0' => null ;
19                    when others => temp := "00000000000" ;
20              end case;
21              out_tfq(10 downto 0) <= temp(10 downto 0) ;
22              out_tfq(12 downto 11) <= "00" ;
23  end process;
24  end to_first_quadrant_arch ;
```

Listato F.1.14: troncatore_12.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity troncatore_12 is
5         port( ingresso :     in     std_logic_vector(31 downto 0);
6               uscita :      out std_logic_vector(12 downto 0)
7               );
8   end troncatore_12;
9
10  architecture tro_12_arch of troncatore_12 is
11        begin
12        process(ingresso)
13              begin
14              for i in 0 to 12 loop
15                    uscita(i) <= ingresso(i+19);
16              end loop;
17        end process;
18  end tro_12_arch;
```

Listato F.2.1: polyphase_gatedClock.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   use work.SRRC_coeffs.all;
4
5   entity polyphase_gatedClock is
6     port(
7         clk           : in STD_LOGIC;
8         reset   : in STD_LOGIC;
9         to_SRRC_I     : in STD_LOGIC;
10        rate_sel      : in STD_LOGIC_VECTOR(1 downto 0);
11        clk_sync      : out STD_LOGIC;
12        poly_out      : out STD_LOGIC_VECTOR(11 downto 0)
13    );
14  end polyphase_gatedClock;
15
16  architecture polyphase_gatedClock of polyphase_gatedClock is
17  component rate_adapter
18    port (
19        clk           : in STD_LOGIC;
20        rate_sel      : in STD_LOGIC_VECTOR(1 downto 0);
```

```
21        reset   : in STD_LOGIC;
22        clk_div_n      : out STD_LOGIC;
23        coeffs_to_SRRCxN : out coeffs;
24        count_n  : out STD_LOGIC_VECTOR(2 downto 0)
25    );
26  end component;
27  component srrc_x_n
28    port (
29        SRRCxN_coeffs : in coeffs;
30        clk           : in STD_LOGIC;
31        clk_div_n     : in STD_LOGIC;
32        count_n  : in STD_LOGIC_VECTOR(2 downto 0);
33        in_fir_MSB : in STD_LOGIC;
34        reset   : in STD_LOGIC;
35        polyphase_out : out STD_LOGIC_VECTOR(11 downto 0)
36    );
37  end component;
38
39  signal clk_div_n           : STD_LOGIC;
40  signal coefficients  : coeffs;
41  signal count_n       : STD_LOGIC_VECTOR (2 downto 0);
42
43  begin
44  U1 : srrc_x_n
45    port map(
46        SRRCxN_coeffs => coefficients,
47        clk => clk,
48        clk_div_n => clk_div_n,
49        count_n => count_n,
50        in_fir_MSB => to_SRRC_I,
51        polyphase_out => poly_out,
52        reset => reset
53    );
54  U2 : rate_adapter
55    port map(
56        clk => clk,
57        clk_div_n => clk_div_n,
58        coeffs_to_SRRCxN => coefficients,
59        count_n => count_n,
60        rate_sel => rate_sel,
61        reset => reset
62    );
63        clk_sync <= clk;
64  end polyphase_gatedClock;
```

Listato F.2.2: srrc_coeffs.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_signed.all;
4
5   package SRRC_coeffs is
6         subtype coeff is std_logic_vector(11 downto 0);
7         type coeffs is array(0 to 83) of coeff;
8   end SRRC_coeffs ;
```

Listato F.2.3: rate_adapter.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   use work.SRRC_coeffs.all;
4
5   entity rate_adapter is
6     port(
7         clk           : in STD_LOGIC;
8         reset   : in STD_LOGIC;
9         rate_sel      : in STD_LOGIC_VECTOR(1 downto 0);
10        clk_div_n     : out STD_LOGIC;
11        coeffs_to_SRRCxN : out coeffs;
12        count_n  : out STD_LOGIC_VECTOR(2 downto 0)
13    );
14  end rate_adapter;
15
16  architecture rate_adapter of rate_adapter is
17  component coeffs_selector
18    port (
19        rate_sel       : in STD_LOGIC_VECTOR(1 downto 0);
20        coeffs_SRRC : out coeffs
21    );
22  end component;
23  component counter_divider_3
24    port (
25        clk     : in STD_LOGIC;
26        reset   : in STD_LOGIC;
```

```
27        clk_div_3 : out STD_LOGIC;
28        count_3  : out STD_LOGIC_VECTOR(2 downto 0)
29    );
30  end component;
31  component counter_divider_4
32    port (
33        clk     : in STD_LOGIC;
34        reset   : in STD_LOGIC;
35        clk_div_4      : out STD_LOGIC;
36        count_4  : out STD_LOGIC_VECTOR(2 downto 0)
37    );
38  end component;
39  component counter_divider_6
40    port (
41        clk           : in STD_LOGIC;
42        reset   : in STD_LOGIC;
43        clk_div_6      : out STD_LOGIC;
44        count_6  : out STD_LOGIC_VECTOR(2 downto 0)
45    );
46  end component;
47  component selector
48    port (
49        clk_div_3      : in STD_LOGIC;
50        clk_div_4      : in STD_LOGIC;
51        clk_div_6      : in STD_LOGIC;
52        count_3  : in STD_LOGIC_VECTOR(2 downto 0);
53        count_4  : in STD_LOGIC_VECTOR(2 downto 0);
54        count_6  : in STD_LOGIC_VECTOR(2 downto 0);
55        rate_sel       : in STD_LOGIC_VECTOR(1 downto 0);
56        clk_div_n      : out STD_LOGIC;
57        count_n  : out STD_LOGIC_VECTOR(2 downto 0)
58    );
59  end component;
60
61  signal SRRC_coefficients : coeffs;
62  signal to_clk_div_3 : STD_LOGIC;
63  signal to_clk_div_4 : STD_LOGIC;
64  signal to_clk_div_6 : STD_LOGIC;
65  signal count_4  : STD_LOGIC_VECTOR (2 downto 0);
66  signal to_count_3 : STD_LOGIC_VECTOR (2 downto 0);
67  signal to_count_6 : STD_LOGIC_VECTOR (2 downto 0);
68
69  begin
70  U0 : counter_divider_3
71    port map(
72        clk           => clk,
73        clk_div_3     => to_clk_div_3,
74        count_3  => to_count_3,
75        reset   => reset
76    );
77  U1 : counter_divider_4
78    port map(
79        clk           => clk,
80        clk_div_4     => to_clk_div_4,
81        count_4  => count_4,
82        reset   => reset
83    );
84  U2 : counter_divider_6
85    port map(
86        clk           => clk,
87        clk_div_6     => to_clk_div_6,
88        count_6  => to_count_6,
89        reset   => reset
90    );
91  U3 : selector
92    port map(
93        clk_div_3     => to_clk_div_3,
94        clk_div_4     => to_clk_div_4,
95        clk_div_6     => to_clk_div_6,
96        clk_div_n     => clk_div_n,
97        count_3  => to_count_3,
98        count_4  => count_4,
99        count_6  => to_count_6,
100       count_n  => count_n,
101       rate_sel      => rate_sel
102   );
103 U5 : coeffs_selector
104   port map(
105       coeffs_SRRC => SRRC_coefficients,
106       rate_sel      => rate_sel
107   );
108       coeffs_to_SRRCxN <= SRRC_coefficients;
109 end rate_adapter;
```

Listato F.2.4: coeffs_selector.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_signed.all;
```

```
4  use work.SRRC_coeffs.all;
5
6  entity coeffs_selector is
7        port(
8                rate_sel   : in std_logic_vector(1 downto 0);
9                coeffs_SRR :       out coeffs
10              );
11 end coeffs_selector;
12
13 architecture coeffs_selector_arch of coeffs_selector is
14    -- calcolati con campionamento in frequenza e scalati
15       Constant coeffs_SRRCx3 : coeffs := coeffs'(
16                                 -- coefficienti del FIR 1
17    coeff'(X"FE0") , -- 1
18    coeff'(X"020") , -- 1 inverted
19    coeff'(X"03D") , -- 2
20    coeff'(X"FC3") , -- 2 inverted
21    coeff'(X"F94") , -- 3
22    coeff'(X"06C") , -- 3 inverted
23    coeff'(X"59E") , -- 4
24    coeff'(X"A62") , -- 4 inverted
25    coeff'(X"F94") , -- 5
26    coeff'(X"06C") , -- 5 inverted
27    coeff'(X"03D") , -- 6
28    coeff'(X"FC3") , -- 6 inverted
29    coeff'(X"FE0") , -- 7
30    coeff'(X"020") , -- 7 inverted
31                                   -- coefficienti del FIR 2
32    coeff'(X"00A") , -- 8
33    coeff'(X"FF6") , -- 8 inverted
34    coeff'(X"FB0") , -- 9
35    coeff'(X"050") , -- 9 inverted
36    coeff'(X"1B1") , -- 10
37    coeff'(X"E4F") , -- 10 inverted
38    coeff'(X"461") , -- 11
39    coeff'(X"B9F") , -- 11 inverted
40    coeff'(X"F19") , -- 12
41    coeff'(X"0E7") , -- 12 inverted
42    coeff'(X"046") , -- 13
43    coeff'(X"FBA") , -- 13 inverted
44    coeff'(X"000") , -- 14
45    coeff'(X"000") , -- 14 inverted
46                                   -- coefficienti del FIR 3
47    coeff'(X"046") , -- 15
48    coeff'(X"FBA") , -- 15 inverted
49    coeff'(X"F19") , -- 16
50    coeff'(X"0E7") , -- 16 inverted
51    coeff'(X"461") , -- 17
52    coeff'(X"B9F") , -- 17 inverted
53    coeff'(X"1B1") , -- 18
54    coeff'(X"E4F") , -- 18 inverted
55    coeff'(X"FB0") , -- 19
56    coeff'(X"050") , -- 19 inverted
57    coeff'(X"00A") , -- 20
58    coeff'(X"FF6") , -- 20 inverted
59    coeff'(X"000") , -- 21
60    coeff'(X"000") , -- 21 inverted
61                                   -- coefficienti del FIR 4
62    coeff'(X"000") , -- 22
63    coeff'(X"000") , -- 22 inverted
64    coeff'(X"000") , -- 23
65    coeff'(X"000") , -- 23 inverted
66    coeff'(X"000") , -- 24
67    coeff'(X"000") , -- 24 inverted
68    coeff'(X"000") , -- 25
69    coeff'(X"000") , -- 25 inverted
70    coeff'(X"000") , -- 26
71    coeff'(X"000") , -- 26 inverted
72    coeff'(X"000") , -- 27
73    coeff'(X"000") , -- 27 inverted
74    coeff'(X"000") , -- 28
75    coeff'(X"000") , -- 28 inverted
76                                   -- coefficienti del FIR 5
77    coeff'(X"000") , -- 29
78    coeff'(X"000") , -- 29 inverted
79    coeff'(X"000") , -- 30
80    coeff'(X"000") , -- 30 inverted
81    coeff'(X"000") , -- 31
82    coeff'(X"000") , -- 31 inverted
83    coeff'(X"000") , -- 32
84    coeff'(X"000") , -- 32 inverted
85    coeff'(X"000") , -- 33
86    coeff'(X"000") , -- 33 inverted
87    coeff'(X"000") , -- 34
88    coeff'(X"000") , -- 34 inverted
89    coeff'(X"000") , -- 35
90    coeff'(X"000") , -- 35 inverted
91                                   -- coefficienti del FIR 6
92    coeff'(X"000") , -- 36
93    coeff'(X"000") , -- 36 inverted
```

```
94      coeff'(X"000") , -- 37
95      coeff'(X"000") , -- 37 inverted
96      coeff'(X"000") , -- 38
97      coeff'(X"000") , -- 38 inverted
98      coeff'(X"000") , -- 39
99      coeff'(X"000") , -- 39 inverted
100     coeff'(X"000") , -- 40
101     coeff'(X"000") , -- 40 inverted
102     coeff'(X"000") , -- 41
103     coeff'(X"000") , -- 41 inverted
104     coeff'(X"000") , -- 42
105     coeff'(X"000")  -- 42 inverted
106       );
107    -- calcolati con campionamento in frequenza e scalati
108       Constant coeffs_SRRCx4 : coeffs := coeffs'(
109                          -- coefficienti del FIR 1
110     coeff'(X"FD8") , -- 1
111     coeff'(X"028") , -- 1 inverted
112     coeff'(X"03D") , -- 2
113     coeff'(X"FC3") , -- 2 inverted
114     coeff'(X"F9B") , -- 3
115     coeff'(X"065") , -- 3 inverted
116     coeff'(X"55D") , -- 4
117     coeff'(X"AA3") , -- 4 inverted
118     coeff'(X"F9B") , -- 5
119     coeff'(X"065") , -- 5 inverted
120     coeff'(X"03D") , -- 6
121     coeff'(X"FC3") , -- 6 inverted
122     coeff'(X"FD8") , -- 7
123     coeff'(X"028") , -- 7 inverted
124                        -- coefficienti del FIR 2
125     coeff'(X"FF9") , -- 8
126     coeff'(X"007") , -- 8 inverted
127     coeff'(X"FDB") , -- 9
128     coeff'(X"025") , -- 9 inverted
129     coeff'(X"104") , -- 10
130     coeff'(X"EFC") , -- 10 inverted
131     coeff'(X"4AE") , -- 11
132     coeff'(X"B52") , -- 11 inverted
133     coeff'(X"F1A") , -- 12
134     coeff'(X"0E6") , -- 12 inverted
135     coeff'(X"051") , -- 13
136     coeff'(X"FAF") , -- 13 inverted
137     coeff'(X"000") , -- 14
138     coeff'(X"000") , -- 14 inverted
139                        -- coefficienti del FIR 3
140     coeff'(X"02C") , -- 15
141     coeff'(X"FD4") , -- 15 inverted
142     coeff'(X"F57") , -- 16
143     coeff'(X"0A9") , -- 16 inverted
144     coeff'(X"2F7") , -- 17
145     coeff'(X"D09") , -- 17 inverted
146     coeff'(X"2F7") , -- 18
147     coeff'(X"D09") , -- 18 inverted
148     coeff'(X"F57") , -- 19
149     coeff'(X"0A9") , -- 19 inverted
150     coeff'(X"02C") , -- 20
151     coeff'(X"FD4") , -- 20 inverted
152     coeff'(X"000") , -- 21
153     coeff'(X"000") , -- 21 inverted
154                        -- coefficienti del FIR 4
155     coeff'(X"051") , -- 22
156     coeff'(X"FAF") , -- 22 inverted
157     coeff'(X"F1A") , -- 23
158     coeff'(X"0E6") , -- 23 inverted
159     coeff'(X"4AE") , -- 24
160     coeff'(X"B52") , -- 24 inverted
161     coeff'(X"104") , -- 25
162     coeff'(X"EFC") , -- 25 inverted
163     coeff'(X"FDB") , -- 26
164     coeff'(X"025") , -- 26 inverted
165     coeff'(X"FF9") , -- 27
166     coeff'(X"007") , -- 27 inverted
167     coeff'(X"000") , -- 28
168     coeff'(X"000") , -- 28 inverted
169                        -- coefficienti del FIR 5
170     coeff'(X"000") , -- 29
171     coeff'(X"000") , -- 29 inverted
172     coeff'(X"000") , -- 30
173     coeff'(X"000") , -- 30 inverted
174     coeff'(X"000") , -- 31
175     coeff'(X"000") , -- 31 inverted
176     coeff'(X"000") , -- 32
177     coeff'(X"000") , -- 32 inverted
178     coeff'(X"000") , -- 33
179     coeff'(X"000") , -- 33 inverted
180     coeff'(X"000") , -- 34
181     coeff'(X"000") , -- 34 inverted
182     coeff'(X"000") , -- 35
```

```
183       coeff'(X"000") , -- 35 inverted
184                          -- coefficienti del FIR 6
185       coeff'(X"000") , -- 36
186       coeff'(X"000") , -- 36 inverted
187       coeff'(X"000") , -- 37
188       coeff'(X"000") , -- 37 inverted
189       coeff'(X"000") , -- 38
190       coeff'(X"000") , -- 38 inverted
191       coeff'(X"000") , -- 39
192       coeff'(X"000") , -- 39 inverted
193       coeff'(X"000") , -- 40
194       coeff'(X"000") , -- 40 inverted
195       coeff'(X"000") , -- 41
196       coeff'(X"000") , -- 41 inverted
197       coeff'(X"000") , -- 42
198       coeff'(X"000")  -- 42 inverted
199          );
200       -- calcolati con campionamento in frequenza e scalati
201          Constant coeffs_SRRCx6 : coeffs := coeffs'(
202                          -- coefficienti del FIR 1
203       coeff'(X"FEB") , -- 1
204       coeff'(X"015") , -- 1 inverted
205       coeff'(X"04A") , -- 2
206       coeff'(X"FB6") , -- 2 inverted
207       coeff'(X"F25") , -- 3
208       coeff'(X"0DB") , -- 3 inverted
209       coeff'(X"53D") , -- 4
210       coeff'(X"AC3") , -- 4 inverted
211       coeff'(X"06E") , -- 5
212       coeff'(X"F92") , -- 5 inverted
213       coeff'(X"00B") , -- 6
214       coeff'(X"FF5") , -- 6 inverted
215       coeff'(X"FF5") , -- 7
216       coeff'(X"00B") , -- 7 inverted
217                          -- coefficienti del FIR 2
218       coeff'(X"FED") , -- 8
219       coeff'(X"013") , -- 8 inverted
220       coeff'(X"03D") , -- 9
221       coeff'(X"FC3") , -- 9 inverted
222       coeff'(X"F91") , -- 10
223       coeff'(X"06F") , -- 10 inverted
224       coeff'(X"591") , -- 11
225       coeff'(X"A6F") , -- 11 inverted
226       coeff'(X"F91") , -- 12
227       coeff'(X"06F") , -- 12 inverted
228       coeff'(X"03D") , -- 13
229       coeff'(X"FC3") , -- 13 inverted
230       coeff'(X"FED") , -- 14
231       coeff'(X"013") , -- 14 inverted
232                          -- coefficienti del FIR 3
233       coeff'(X"FF5") , -- 15
234       coeff'(X"00B") , -- 15 inverted
235       coeff'(X"00B") , -- 16
236       coeff'(X"FF5") , -- 16 inverted
237       coeff'(X"06E") , -- 17
238       coeff'(X"F92") , -- 17 inverted
239       coeff'(X"53D") , -- 18
240       coeff'(X"AC3") , -- 18 inverted
241       coeff'(X"F25") , -- 19
242       coeff'(X"0DB") , -- 19 inverted
243       coeff'(X"04A") , -- 20
244       coeff'(X"FB6") , -- 20 inverted
245       coeff'(X"FEB") , -- 21
246       coeff'(X"015") , -- 21 inverted
247                          -- coefficienti del FIR 4
248       coeff'(X"006") , -- 22
249       coeff'(X"FFA") , -- 22 inverted
250       coeff'(X"FB9") , -- 23
251       coeff'(X"047") , -- 23 inverted
252       coeff'(X"1A7") , -- 24
253       coeff'(X"E59") , -- 24 inverted
254       coeff'(X"455") , -- 25
255       coeff'(X"BAB") , -- 25 inverted
256       coeff'(X"F1F") , -- 26
257       coeff'(X"0E1") , -- 26 inverted
258       coeff'(X"03B") , -- 27
259       coeff'(X"FC5") , -- 27 inverted
260       coeff'(X"000") , -- 28
261       coeff'(X"000") , -- 28 inverted
262                          -- coefficienti del FIR 5
263       coeff'(X"020") , -- 29
264       coeff'(X"FE0") , -- 29 inverted
265       coeff'(X"F5E") , -- 30
266       coeff'(X"0A2") , -- 30 inverted
267       coeff'(X"30B") , -- 31
268       coeff'(X"CF5") , -- 31 inverted
269       coeff'(X"30B") , -- 32
270       coeff'(X"CF5") , -- 32 inverted
271       coeff'(X"F5E") , -- 33
```

```
272     coeff'(X"0A2") , -- 33 inverted
273     coeff'(X"020") , -- 34
274     coeff'(X"FE0") , -- 34 inverted
275     coeff'(X"000") , -- 35
276     coeff'(X"000") , -- 35 inverted
277                      -- coefficienti del FIR 6
278     coeff'(X"03B") , -- 36
279     coeff'(X"FC5") , -- 36 inverted
280     coeff'(X"F1F") , -- 37
281     coeff'(X"0E1") , -- 37 inverted
282     coeff'(X"455") , -- 38
283     coeff'(X"BAB") , -- 38 inverted
284     coeff'(X"1A7") , -- 39
285     coeff'(X"E59") , -- 39 inverted
286     coeff'(X"FB9") , -- 40
287     coeff'(X"047") , -- 40 inverted
288     coeff'(X"006") , -- 41
289     coeff'(X"FFA") , -- 41 inverted
290     coeff'(X"000") , -- 42
291     coeff'(X"000")  -- 42 inverted
292       );
293 begin
294       with rate_sel select
295       coeffs_SRRC <= coeffs_SRRCx3 when "00" ,
296                            coeffs_SRRCx4 when "01" ,
297                          coeffs_SRRCx6 when others ;
298 end coeffs_selector_arch;
```

Listato F.2.5: counter_divider_3.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4
5  entity counter_divider_3 is
6        port (
7              clk          : in STD_LOGIC;
8              reset        : in STD_LOGIC;
9              count_3 : out STD_LOGIC_VECTOR (2 downto 0);
10             clk_div_3    : out STD_LOGIC
11             );
12 end counter_divider_3;
13
14 architecture counter_divider_3_arch of counter_divider_3 is
15       signal int_count_3   : STD_LOGIC_VECTOR (1 downto 0) ;
16       signal count_0_delayed : STD_LOGIC;
17 begin
18       process (clk , reset)
19       begin
20             if reset='1' then
21                   int_count_3 <= "01";
22             elsif falling_edge(clk) then
23                   int_count_3 <= int_count_3(0)&not(int_count_3(0) or
                           int_count_3(1));
24             end if;
25       end process;
26
27       process(clk)
28       begin
29             if rising_edge(clk) then
30             count_0_delayed <= int_count_3(0);
31             end if;
32       end process;
33       clk_div_3 <= int_count_3(0) nor count_0_delayed;
34
35       process(clk)
36       begin
37       if falling_edge(clk) then
38             count_3 <= '0' & int_count_3;
39       end if;
40       end process;
41 end counter_divider_3_arch;
```

Listato F.2.6: counter_divider_4.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity counter_divider_4 is
5    port(
6         clk      : in STD_LOGIC;
7         reset    : in STD_LOGIC;
```

```
8         clk_div_4 : out STD_LOGIC;
9         count_4  : out STD_LOGIC_VECTOR(2 downto 0)
10    );
11  end counter_divider_4;
12
13  architecture counter_divider_4 of counter_divider_4 is
14  component fftr
15    port (
16         clk  : in STD_LOGIC;
17         eing : in STD_LOGIC;
18         reset : in STD_LOGIC;
19         aus  : out STD_LOGIC
20    );
21  end component;
22
23  constant VCC_CONSTANT : STD_LOGIC := '1';
24  constant GND_CONSTANT : STD_LOGIC := '0';
25  signal GND : STD_LOGIC;
26  signal VCC : STD_LOGIC;
27  signal aus : STD_LOGIC_VECTOR (2 downto 0);
28
29  begin
30  U1 : fftr
31    port map(
32         aus  => aus(1),
33         clk  => clk,
34         eing => aus(0),
35         reset => reset
36    );
37  U2 : fftr
38    port map(
39         aus  => aus(0),
40         clk  => clk,
41         eing => VCC,
42         reset => reset
43    );
44          VCC <= VCC_CONSTANT;
45          GND <= GND_CONSTANT;
46          aus(2) <= GND;
47          clk_div_4 <= aus(1);
48          count_4 <= aus;
49  end counter_divider_4;
```

Listato F.2.7: fftr.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   entity fftr is
5         port (
6                reset : in std_logic;
7                clk  : in std_logic;
8                eing : in std_logic;
9                aus  : out std_logic
10        );
11  end entity;
12
13  architecture fftr_arch of fftr is
14        signal TEMP_aus: std_logic;
15  begin
16        process (clk, reset)
17        begin
18               if reset = '1' then
19                      TEMP_aus <= '0';
20               elsif falling_edge(clk) then
21                      if eing = '1' then
22                             TEMP_aus <= not TEMP_AUS;
23                      else null;
24                      end if;
25               end if;
26        end process;
27  aus <= TEMP_aus;
28  end architecture;
```

Listato F.2.8: counter_divider_6.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   use IEEE.std_logic_unsigned.all;
4
5   entity counter_divider_6 is
6         port (
7                clk          : in STD_LOGIC;
8                reset        : in STD_LOGIC;
```

```
9               count_6 : out STD_LOGIC_VECTOR (2 downto 0);
10              clk_div_6     : out STD_LOGIC
11              );
12  end counter_divider_6;
13
14  architecture counter_divider_6_arch of counter_divider_6 is
15  begin
16        process (clk, reset)
17              variable count_6_interno : STD_LOGIC_VECTOR (2 downto 0);
18        begin
19              if reset='1' then
20                    count_6_interno := "000";
21                    clk_div_6     <= '0' ;
22              else
23                    if falling_edge(clk) then
24                          if count_6_interno < 5 then
25                                count_6_interno := count_6_interno + 1;
26                                if count_6_interno = 3 then
27                                      clk_div_6 <= '1' ;
28                                else null;
29                                end if ;
30                          else
31                                count_6_interno := "000";
32                                clk_div_6     <= '0' ;
33                          end if;
34                    end if;
35              end if;
36              count_6 <= count_6_interno;
37        end process;
38  end counter_divider_6_arch;
```

Listato F.2.9: selector.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity selector is
5         port( rate_sel :     in std_logic_vector( 1 downto 0);
6                 clk_div_3 : in std_logic ;
7                 count_3     :     in std_logic_vector(2 downto 0);
8                 clk_div_4 : in std_logic ;
9                 count_4     :     in std_logic_vector(2 downto 0);
10                clk_div_6 : in std_logic ;
11                count_6     :     in std_logic_vector(2 downto 0);
12                clk_div_n : out std_logic ;
13                count_n     :     out std_logic_vector(2 downto 0)
14               );
15  end selector;
16
17  architecture selector_arch of selector is
18  begin
19        process (rate_sel,clk_div_3,clk_div_4,clk_div_6,count_3,count_4,count_6)
20        begin
21            case rate_sel is
22                  when "00" =>
23                        clk_div_n <= clk_div_3 ;
24                        count_n  <= count_3  ;
25                  when "01" =>
26                        clk_div_n <= clk_div_4 ;
27                        count_n  <= count_4  ;
28                  when "10" =>
29                       clk_div_n <= clk_div_6 ;
30                        count_n  <= count_6  ;
31                  when others =>
32                        clk_div_n <= 'X'     ;
33                        count_n  <= "XXX"    ;
34            end case;
35     end process;
36  end selector_arch;
```

Listato F.2.10: srrc_x_n.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   use work.SRRC_coeffs.all;
4
5   entity SRRC_x_N is
6     port(
7         SRRCxN_coeffs : in coeffs;
8         clk           : in STD_LOGIC;
9         clk_div_n     : in STD_LOGIC;
10        in_fir_MSB    : in STD_LOGIC;
11        reset         : in STD_LOGIC;
12        count_n       : in STD_LOGIC_VECTOR(2 downto 0);
13        polyphase_out : out STD_LOGIC_VECTOR(11 downto 0)
```

```
14    );
15  end SRRC_x_N;
16
17  architecture SRRC_x_N of SRRC_x_N is
18  component fir_1
19    port (
20         clk_div_n      : in STD_LOGIC;
21         coeff_a   : in coeff;
22         coeff_b   : in coeff;
23         coeff_c   : in coeff;
24         coeff_d   : in coeff;
25         coeff_e   : in coeff;
26         coeff_f   : in coeff;
27         coeff_g   : in coeff;
28         in_fir_MSB : in STD_LOGIC;
29         n_coeff_a      : in coeff;
30         n_coeff_b      : in coeff;
31         n_coeff_c      : in coeff;
32         n_coeff_d      : in coeff;
33         n_coeff_e      : in coeff;
34         n_coeff_f      : in coeff;
35         n_coeff_g      : in coeff;
36         reset    : in STD_LOGIC;
37         out_fir   : out STD_LOGIC_VECTOR(11 downto 0)
38    );
39  end component;
40  component mux_6
41    port (
42         clk            : in STD_LOGIC;
43         count_n   : in STD_LOGIC_VECTOR(2 downto 0);
44         in_0    : in STD_LOGIC_VECTOR(11 downto 0);
45         in_1    : in STD_LOGIC_VECTOR(11 downto 0);
46         in_2    : in STD_LOGIC_VECTOR(11 downto 0);
47         in_3    : in STD_LOGIC_VECTOR(11 downto 0);
48         in_4    : in STD_LOGIC_VECTOR(11 downto 0);
49         in_5    : in STD_LOGIC_VECTOR(11 downto 0);
50         out_mux   : out STD_LOGIC_VECTOR(11 downto 0)
51    );
52  end component;
53
54  signal coeff_1       : coeff;
55  signal coeff_2       : coeff;
56  signal coeff_3       : coeff;
57  signal coeff_4       : coeff;
58  signal coeff_5       : coeff;
59  signal coeff_6       : coeff;
60  signal coeff_7       : coeff;
61  signal coeff_8       : coeff;
62  signal coeff_9       : coeff;
63  signal coeff_10      : coeff;
64  signal coeff_11      : coeff;
65  signal coeff_12      : coeff;
66  signal coeff_13      : coeff;
67  signal coeff_14      : coeff;
68  signal coeff_15      : coeff;
69  signal coeff_16      : coeff;
70  signal coeff_17      : coeff;
71  signal coeff_18      : coeff;
72  signal coeff_19      : coeff;
73  signal coeff_20      : coeff;
74  signal coeff_21      : coeff;
75  signal coeff_22      : coeff;
76  signal coeff_23      : coeff;
77  signal coeff_24      : coeff;
78  signal coeff_25      : coeff;
79  signal coeff_26      : coeff;
80  signal coeff_27      : coeff;
81  signal coeff_28      : coeff;
82  signal coeff_29      : coeff;
83  signal coeff_30      : coeff;
84  signal coeff_31      : coeff;
85  signal coeff_32      : coeff;
86  signal coeff_33      : coeff;
87  signal coeff_34      : coeff;
88  signal coeff_35      : coeff;
89  signal coeff_36      : coeff;
90  signal coeff_37      : coeff;
91  signal coeff_38      : coeff;
92  signal coeff_39      : coeff;
93  signal coeff_40 : coeff;
94  signal coeff_41 : coeff;
95  signal coeff_42 : coeff;
96
97  signal n_coeff_1 : coeff;
98  signal n_coeff_2 : coeff;
99  signal n_coeff_3 : coeff;
100 signal n_coeff_4 : coeff;
101 signal n_coeff_5 : coeff;
102 signal n_coeff_6 : coeff;
103 signal n_coeff_7 : coeff;
104 signal n_coeff_8 : coeff;
105 signal n_coeff_9 : coeff;
106 signal n_coeff_10 : coeff;
107 signal n_coeff_11 : coeff;
108 signal n_coeff_12 : coeff;
109 signal n_coeff_13 : coeff;
110 signal n_coeff_14 : coeff;
```

```
111  signal n_coeff_15 : coeff;
112  signal n_coeff_16 : coeff;
113  signal n_coeff_17 : coeff;
114  signal n_coeff_18 : coeff;
115  signal n_coeff_19 : coeff;
116  signal n_coeff_20 : coeff;
117  signal n_coeff_21 : coeff;
118  signal n_coeff_22 : coeff;
119  signal n_coeff_23 : coeff;
120  signal n_coeff_24 : coeff;
121  signal n_coeff_25 : coeff;
122  signal n_coeff_26 : coeff;
123  signal n_coeff_27 : coeff;
124  signal n_coeff_28 : coeff;
125  signal n_coeff_29 : coeff;
126  signal n_coeff_30 : coeff;
127  signal n_coeff_31 : coeff;
128  signal n_coeff_32 : coeff;
129  signal n_coeff_33 : coeff;
130  signal n_coeff_34 : coeff;
131  signal n_coeff_35 : coeff;
132  signal n_coeff_36 : coeff;
133  signal n_coeff_37 : coeff;
134  signal n_coeff_38 : coeff;
135  signal n_coeff_39 : coeff;
136  signal n_coeff_40 : coeff;
137  signal n_coeff_41 : coeff;
138  signal n_coeff_42 : coeff;
139
140  signal to_in_0 : STD_LOGIC_VECTOR (11 downto 0);
141  signal to_in_1 : STD_LOGIC_VECTOR (11 downto 0);
142  signal to_in_2 : STD_LOGIC_VECTOR (11 downto 0);
143  signal to_in_3 : STD_LOGIC_VECTOR (11 downto 0);
144  signal to_in_4 : STD_LOGIC_VECTOR (11 downto 0);
145  signal to_in_5 : STD_LOGIC_VECTOR (11 downto 0);
146
147  begin
148  coeff_1       <= SRRCxN_coeffs(0);
149  n_coeff_1 <= SRRCxN_coeffs(1);
150  coeff_2       <= SRRCxN_coeffs(2);
151  n_coeff_2 <= SRRCxN_coeffs(3);
152  coeff_3       <= SRRCxN_coeffs(4);
153  n_coeff_3 <= SRRCxN_coeffs(5);
154  coeff_4       <= SRRCxN_coeffs(6);
155  n_coeff_4 <= SRRCxN_coeffs(7);
156  coeff_5       <= SRRCxN_coeffs(8);
157  n_coeff_5 <= SRRCxN_coeffs(9);
158  coeff_6       <= SRRCxN_coeffs(10);
159  n_coeff_6 <= SRRCxN_coeffs(11);
160  coeff_7       <= SRRCxN_coeffs(12);
161  n_coeff_7 <= SRRCxN_coeffs(13);
162  coeff_8       <= SRRCxN_coeffs(14);
163  n_coeff_8 <= SRRCxN_coeffs(15);
164  coeff_9       <= SRRCxN_coeffs(16);
165  n_coeff_9 <= SRRCxN_coeffs(17);
166  coeff_10      <= SRRCxN_coeffs(18);
167  n_coeff_10 <= SRRCxN_coeffs(19);
168  coeff_11      <= SRRCxN_coeffs(20);
169  n_coeff_11 <= SRRCxN_coeffs(21);
170  coeff_12      <= SRRCxN_coeffs(22);
171  n_coeff_12 <= SRRCxN_coeffs(23);
172  coeff_13      <= SRRCxN_coeffs(24);
173  n_coeff_13 <= SRRCxN_coeffs(25);
174  coeff_14      <= SRRCxN_coeffs(26);
175  n_coeff_14 <= SRRCxN_coeffs(27);
176  coeff_15      <= SRRCxN_coeffs(28);
177  n_coeff_15 <= SRRCxN_coeffs(29);
178  coeff_16      <= SRRCxN_coeffs(30);
179  n_coeff_16 <= SRRCxN_coeffs(31);
180  coeff_17      <= SRRCxN_coeffs(32);
181  n_coeff_17 <= SRRCxN_coeffs(33);
182  coeff_18      <= SRRCxN_coeffs(34);
183  n_coeff_18 <= SRRCxN_coeffs(35);
184  coeff_19      <= SRRCxN_coeffs(36);
185  n_coeff_19 <= SRRCxN_coeffs(37);
186  coeff_20      <= SRRCxN_coeffs(38);
187  n_coeff_20 <= SRRCxN_coeffs(39);
188  coeff_21      <= SRRCxN_coeffs(40);
189  n_coeff_21 <= SRRCxN_coeffs(41);
190  coeff_22      <= SRRCxN_coeffs(42);
191  n_coeff_22 <= SRRCxN_coeffs(43);
192  coeff_23      <= SRRCxN_coeffs(44);
193  n_coeff_23 <= SRRCxN_coeffs(45);
194  coeff_24      <= SRRCxN_coeffs(46);
195  n_coeff_24 <= SRRCxN_coeffs(47);
196  coeff_25      <= SRRCxN_coeffs(48);
197  n_coeff_25 <= SRRCxN_coeffs(49);
198  coeff_26      <= SRRCxN_coeffs(50);
199  n_coeff_26 <= SRRCxN_coeffs(51);
200  coeff_27      <= SRRCxN_coeffs(52);
201  n_coeff_27 <= SRRCxN_coeffs(53);
```

```
202  coeff_28      <= SRRCxN_coeffs(54);
203  n_coeff_28 <= SRRCxN_coeffs(55);
204  coeff_29      <= SRRCxN_coeffs(56);
205  n_coeff_29 <= SRRCxN_coeffs(57);
206  coeff_30      <= SRRCxN_coeffs(58);
207  n_coeff_30 <= SRRCxN_coeffs(59);
208  coeff_31      <= SRRCxN_coeffs(60);
209  n_coeff_31 <= SRRCxN_coeffs(61);
210  coeff_32      <= SRRCxN_coeffs(62);
211  n_coeff_32 <= SRRCxN_coeffs(63);
212  coeff_33      <= SRRCxN_coeffs(64);
213  n_coeff_33 <= SRRCxN_coeffs(65);
214  coeff_34      <= SRRCxN_coeffs(66);
215  n_coeff_34 <= SRRCxN_coeffs(67);
216  coeff_35      <= SRRCxN_coeffs(68);
217  n_coeff_35 <= SRRCxN_coeffs(69);
218  coeff_36      <= SRRCxN_coeffs(70);
219  n_coeff_36 <= SRRCxN_coeffs(71);
220  coeff_37      <= SRRCxN_coeffs(72);
221  n_coeff_37 <= SRRCxN_coeffs(73);
222  coeff_38      <= SRRCxN_coeffs(74);
223  n_coeff_38 <= SRRCxN_coeffs(75);
224  coeff_39      <= SRRCxN_coeffs(76);
225  n_coeff_39 <= SRRCxN_coeffs(77);
226  coeff_40      <= SRRCxN_coeffs(78);
227  n_coeff_40 <= SRRCxN_coeffs(79);
228  coeff_41      <= SRRCxN_coeffs(80);
229  n_coeff_41 <= SRRCxN_coeffs(81);
230  coeff_42      <= SRRCxN_coeffs(82);
231  n_coeff_42 <= SRRCxN_coeffs(83);
232
233  U1 : fir_1
234    port map(
235         clk_div_n     => clk_div_n,
236         coeff_a  => coeff_1,
237         coeff_b  => coeff_2,
238         coeff_c  => coeff_3,
239         coeff_d  => coeff_4,
240         coeff_e  => coeff_5,
241         coeff_f  => coeff_6,
242         coeff_g  => coeff_7,
243         in_fir_MSB    => in_fir_MSB,
244         n_coeff_a     => n_coeff_1,
245         n_coeff_b     => n_coeff_2,
246         n_coeff_c     => n_coeff_3,
247         n_coeff_d     => n_coeff_4,
248         n_coeff_e     => n_coeff_5,
249         n_coeff_f     => n_coeff_6,
250         n_coeff_g     => n_coeff_7,
251         out_fir  => to_in_0,
252         reset   => reset
253    );
254  U2 : fir_1
255    port map(
256         clk_div_n     => clk_div_n,
257         coeff_a  => coeff_8,
258         coeff_b  => coeff_9,
259         coeff_c  => coeff_10,
260         coeff_d  => coeff_11,
261         coeff_e  => coeff_12,
262         coeff_f  => coeff_13,
263         coeff_g  => coeff_14,
264         in_fir_MSB    => in_fir_MSB,
265         n_coeff_a     => n_coeff_8,
266         n_coeff_b     => n_coeff_9,
267         n_coeff_c     => n_coeff_10,
268         n_coeff_d     => n_coeff_11,
269         n_coeff_e     => n_coeff_12,
270         n_coeff_f     => n_coeff_13,
271         n_coeff_g     => n_coeff_14,
272         out_fir  => to_in_1,
273         reset   => reset
274    );
275  U3 : fir_1
276    port map(
277         clk_div_n     => clk_div_n,
278         coeff_a  => coeff_15,
279         coeff_b  => coeff_16,
280         coeff_c  => coeff_17,
281         coeff_d  => coeff_18,
282         coeff_e  => coeff_19,
283         coeff_f  => coeff_20,
284         coeff_g  => coeff_21,
285         in_fir_MSB    => in_fir_MSB,
286         n_coeff_a     => n_coeff_15,
287         n_coeff_b     => n_coeff_16,
288         n_coeff_c     => n_coeff_17,
289         n_coeff_d     => n_coeff_18,
290         n_coeff_e     => n_coeff_19,
291         n_coeff_f     => n_coeff_20,
292         n_coeff_g     => n_coeff_21,
293         out_fir  => to_in_2,
294         reset   => reset
295    );
```

```
296 U4 : fir_1
297   port map(
298       clk_div_n      => clk_div_n,
299       coeff_a  => coeff_22,
300       coeff_b  => coeff_23,
301       coeff_c  => coeff_24,
302       coeff_d  => coeff_25,
303       coeff_e  => coeff_26,
304       coeff_f  => coeff_27,
305       coeff_g  => coeff_28,
306       in_fir_MSB     => in_fir_MSB,
307       n_coeff_a      => n_coeff_22,
308       n_coeff_b      => n_coeff_23,
309       n_coeff_c      => n_coeff_24,
310       n_coeff_d      => n_coeff_25,
311       n_coeff_e      => n_coeff_26,
312       n_coeff_f      => n_coeff_27,
313       n_coeff_g      => n_coeff_28,
314       out_fir  => to_in_3,
315       reset   => reset
316   );
317 U5 : fir_1
318   port map(
319       clk_div_n      => clk_div_n,
320       coeff_a  => coeff_29,
321       coeff_b  => coeff_30,
322       coeff_c  => coeff_31,
323       coeff_d  => coeff_32,
324       coeff_e  => coeff_33,
325       coeff_f  => coeff_34,
326       coeff_g  => coeff_35,
327       in_fir_MSB => in_fir_MSB,
328       n_coeff_a      => n_coeff_29,
329       n_coeff_b      => n_coeff_30,
330       n_coeff_c      => n_coeff_31,
331       n_coeff_d      => n_coeff_32,
332       n_coeff_e      => n_coeff_33,
333       n_coeff_f      => n_coeff_34,
334       n_coeff_g      => n_coeff_35,
335       out_fir  => to_in_4,
336       reset   => reset
337   );
338 U6 : fir_1
339   port map(
340       clk_div_n      => clk_div_n,
341       coeff_a  => coeff_36,
342       coeff_b  => coeff_37,
343       coeff_c  => coeff_38,
344       coeff_d  => coeff_39,
345       coeff_e  => coeff_40,
346       coeff_f  => coeff_41,
347       coeff_g  => coeff_42,
348       in_fir_MSB => in_fir_MSB,
349       n_coeff_a      => n_coeff_36,
350       n_coeff_b      => n_coeff_37,
351       n_coeff_c      => n_coeff_38,
352       n_coeff_d      => n_coeff_39,
353       n_coeff_e      => n_coeff_40,
354       n_coeff_f      => n_coeff_41,
355       n_coeff_g      => n_coeff_42,
356       out_fir  => to_in_5,
357       reset   => reset
358   );
359 U7 : mux_6
360   port map(
361       clk => clk,
362       count_n => count_n,
363       in_0 => to_in_0,
364       in_1 => to_in_1,
365       in_2 => to_in_2,
366       in_3 => to_in_3,
367       in_4 => to_in_4,
368       in_5 => to_in_5,
369       out_mux => polyphase_out
370   );
371 end SRRC_x_N;
```

Listato F.2.11: fir_1.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use work.SRRC_coeffs.all;
4
5  entity fir_1 is
6    port(
7       clk_div_n : in STD_LOGIC;
8       coeff_a   : in coeff;
9       coeff_b   : in coeff;
10      coeff_c   : in coeff;
11      coeff_d   : in coeff;
12      coeff_e   : in coeff;
```

```
13        coeff_f   : in coeff;
14        coeff_g   : in coeff;
15        in_fir_MSB : in STD_LOGIC;
16        n_coeff_a : in coeff;
17        n_coeff_b : in coeff;
18        n_coeff_c : in coeff;
19        n_coeff_d : in coeff;
20        n_coeff_e : in coeff;
21        n_coeff_f : in coeff;
22        n_coeff_g : in coeff;
23        reset     : in STD_LOGIC;
24        out_fir   : out STD_LOGIC_VECTOR(11 downto 0)
25    );
26  end fir_1;
27
28  architecture fir_1 of fir_1 is
29  component adder_7
30    port (
31        clk_div_n : in STD_LOGIC;
32        in_a : in STD_LOGIC_VECTOR(11 downto 0);
33        in_b : in STD_LOGIC_VECTOR(11 downto 0);
34        in_c : in STD_LOGIC_VECTOR(11 downto 0);
35        in_d : in STD_LOGIC_VECTOR(11 downto 0);
36        in_e : in STD_LOGIC_VECTOR(11 downto 0);
37        in_f : in STD_LOGIC_VECTOR(11 downto 0);
38        in_g : in STD_LOGIC_VECTOR(11 downto 0);
39        reset : in STD_LOGIC;
40        out_adder : out STD_LOGIC_VECTOR(11 downto 0)
41    );
42  end component;
43  component fir_multiplier
44    port (
45        clk_div_n : in STD_LOGIC;
46        coeff_n  : in coeff;
47        in_a     : in STD_LOGIC;
48        n_coeff_n : in coeff;
49        reset    : in STD_LOGIC;
50        out_mult : out STD_LOGIC_VECTOR(11 downto 0)
51    );
52  end component;
53  component shift_reg
54    port (
55        clk_div_n : in STD_LOGIC;
56        in_reg   : in STD_LOGIC;
57        reset    : in STD_LOGIC;
58        out_ffd_1 : out STD_LOGIC;
59        out_ffd_2 : out STD_LOGIC;
60        out_ffd_3 : out STD_LOGIC;
61        out_ffd_4 : out STD_LOGIC;
62        out_ffd_5 : out STD_LOGIC;
63        out_ffd_6 : out STD_LOGIC
64    );
65  end component;
66  signal in_ffd_1 : STD_LOGIC;
67  signal out_ffd_1 : STD_LOGIC;
68  signal out_ffd_2 : STD_LOGIC;
69  signal out_ffd_3 : STD_LOGIC;
70  signal out_ffd_4 : STD_LOGIC;
71  signal out_ffd_5 : STD_LOGIC;
72  signal out_ffd_6 : STD_LOGIC;
73  signal to_add_a : STD_LOGIC_VECTOR (11 downto 0);
74  signal to_add_b : STD_LOGIC_VECTOR (11 downto 0);
75  signal to_add_c : STD_LOGIC_VECTOR (11 downto 0);
76  signal to_add_d : STD_LOGIC_VECTOR (11 downto 0);
77  signal to_add_e : STD_LOGIC_VECTOR (11 downto 0);
78  signal to_add_f : STD_LOGIC_VECTOR (11 downto 0);
79  signal to_add_g : STD_LOGIC_VECTOR (11 downto 0);
80
81  begin
82  U1 : shift_reg
83    port map(
84        clk_div_n => clk_div_n,
85        in_reg   => in_ffd_1,
86        out_ffd_1 => out_ffd_1,
87        out_ffd_2 => out_ffd_2,
88        out_ffd_3 => out_ffd_3,
89        out_ffd_4 => out_ffd_4,
90        out_ffd_5 => out_ffd_5,
91        out_ffd_6 => out_ffd_6,
92        reset    => reset
93    );
94  U2 : fir_multiplier
95    port map(
96        clk_div_n => clk_div_n,
97        coeff_n  => coeff_a,
98        in_a     => in_ffd_1,
99        n_coeff_n => n_coeff_a,
100       out_mult => to_add_a,
101       reset    => reset
102   );
103 U3 : fir_multiplier
104   port map(
105       clk_div_n => clk_div_n,
106       coeff_n  => coeff_b,
107       in_a     => out_ffd_1,
```

```
108         n_coeff_n => n_coeff_b,
109         out_mult => to_add_b,
110         reset    => reset
111     );
112   U4 : fir_multiplier
113     port map(
114         clk_div_n => clk_div_n,
115         coeff_n  => coeff_c,
116         in_a     => out_ffd_2,
117         n_coeff_n => n_coeff_c,
118         out_mult => to_add_c,
119         reset    => reset
120     );
121   U5 : fir_multiplier
122     port map(
123         clk_div_n => clk_div_n,
124         coeff_n  => coeff_d,
125         in_a     => out_ffd_3,
126         n_coeff_n => n_coeff_d,
127         out_mult => to_add_d,
128         reset    => reset
129     );
130   U6 : fir_multiplier
131     port map(
132         clk_div_n => clk_div_n,
133         coeff_n  => coeff_e,
134         in_a     => out_ffd_4,
135         n_coeff_n => n_coeff_e,
136         out_mult => to_add_e,
137         reset    => reset
138     );
139   U7 : fir_multiplier
140     port map(
141         clk_div_n => clk_div_n,
142         coeff_n  => coeff_f,
143         in_a     => out_ffd_5,
144         n_coeff_n => n_coeff_f,
145         out_mult => to_add_f,
146         reset    => reset
147     );
148   U8 : fir_multiplier
149     port map(
150         clk_div_n => clk_div_n,
151         coeff_n  => coeff_g,
152         in_a     => out_ffd_6,
153         n_coeff_n => n_coeff_g,
154         out_mult => to_add_g,
155         reset    => reset
156     );
157   U9 : adder_7
158     port map(
159         clk_div_n => clk_div_n,
160         in_a     => to_add_a,
161         in_b     => to_add_b,
162         in_c     => to_add_c,
163         in_d     => to_add_d,
164         in_e     => to_add_e,
165         in_f     => to_add_f,
166         in_g     => to_add_g,
167         out_adder => out_fir,
168         reset    => reset
169     );
170          in_ffd_1 <= in_fir_MSB;
171   end fir_1;
```

Listato F.2.12: adder_7.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_signed.all;
4
5   entity adder_7 is
6         port(in_a, in_b, in_c, in_d : in std_logic_vector(11 downto 0);
7            in_e, in_f, in_g      :  in std_logic_vector(11 downto 0);
8              clk_div_n , reset   :  in std_logic;
9              out_adder           :  out std_logic_vector(11 downto 0));
10  end adder_7 ;
11
12  architecture adder_7_arch of adder_7 is
13  begin
14        process(reset, clk_div_n , in_a , in_b , in_c , in_d, in_e, in_f, in_g)
15        begin
16              if reset = '1' then
17                    out_adder <= "000000000000";
18              elsif falling_edge(clk_div_n) then
19                    out_adder <= in_a + in_b + in_c + in_d + in_e + in_f + in_g;
20              end if ;
21        end process;
22  end adder_7_arch ;
```

Listato F.2.13: fir_multiplier.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_signed.all;
4  use work.SRRC_coeffs.all;
5
6  entity fir_multiplier is
7        port( in_a          :      in std_logic ;
8              reset :      in std_logic ;
9              coeff_n  :   in coeff    ;
10             n_coeff_n : in coeff    ;
11             clk_div_n :   in std_logic ;
12             out_mult    :      out std_logic_vector(11 downto 0));
13 end fir_multiplier;
14
15 architecture fir_multiplier_arch of fir_multiplier is
16 begin
17       process(in_a , coeff_n, n_coeff_n, clk_div_n, reset)
18       begin
19             if reset = '1' then
20                   out_mult <= "000000000000" ;
21             elsif falling_edge(clk_div_n) then
22                   case in_a is
23                         when '0'  => out_mult <= coeff_n    ;
24                         when '1'  => out_mult <= n_coeff_n  ;
25                         when others => out_mult <= "000000000000" ;
26                   end case;
27             end if;
28       end process;
29 end fir_multiplier_arch;
```

Listato F.2.14: shift_reg.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity shift_reg is
5        port (
6              clk_div_n : in std_logic ;
7              reset    : in std_logic ;
8              in_reg   : in std_logic ;
9              out_ffd_1 : out std_logic ;
10             out_ffd_2 : out std_logic ;
11             out_ffd_3 : out std_logic ;
12             out_ffd_4 : out std_logic ;
13             out_ffd_5 : out std_logic ;
14             out_ffd_6 : out std_logic
15       );
16 end entity;
17
18 architecture shift_reg_arch of shift_reg is
19       signal temp_out_reg : std_logic_vector(5 downto 0);
20       signal temp_go_mult : std_logic_vector(6 downto 0);
21 begin
22       process(clk_div_n, reset)
23       begin
24             if reset = '1' then
25                   temp_out_reg <= "000000";
26             elsif falling_edge(clk_div_n) then
27                   temp_out_reg <= in_reg & temp_out_reg(5 downto 1);
28             end if;
29       end process;
30       out_ffd_6 <= temp_out_reg(0);
31       out_ffd_5 <= temp_out_reg(1);
32       out_ffd_4 <= temp_out_reg(2);
33       out_ffd_3 <= temp_out_reg(3);
34       out_ffd_2 <= temp_out_reg(4);
35       out_ffd_1 <= temp_out_reg(5);
36 end architecture;
```

Listato F.2.15: mux_6.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
```

```
4  entity mux_6 is
5        port(in_0, in_1, in_2, in_3, in_4, in_5 : in std_logic_vector(11 downto 0);
6                 count_n     :      in std_logic_vector(2 downto 0);
7                 clk      :   in std_logic;
8                 out_mux     :      out std_logic_vector(11 downto 0));
9  end mux_6;
10
11 architecture mux_arch of mux_6 is
12 begin
13       process (clk, count_n, in_0, in_1, in_2, in_3, in_4, in_5)
14       begin
15             if falling_edge(clk) then
16                   case count_n is
17                         when "000" =>   out_mux <= in_0 ;
18                         when "001" =>   out_mux <= in_1 ;
19                         when "010" =>   out_mux <= in_2 ;
20                         when "011" =>   out_mux <= in_3 ;
21                         when "100" =>   out_mux <= in_4 ;
22                         when "101" =>   out_mux <= in_5 ;
23                         when others =>  out_mux <= "XXXXXXXXXXXX" ;
24                   end case;
25             else null ;
26             end if;
27       end process;
28 end mux_arch ;
```

Listato F.3.1: ROM_polyphase.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity ROM_polyphase is
5    port(
6         clk     : in STD_LOGIC;
7         reset   : in STD_LOGIC;
8         to_SRRC_I : in STD_LOGIC;
9         rate_sel : in STD_LOGIC_VECTOR(1 downto 0);
10        clk_sync : out STD_LOGIC;
11        poly_out : out STD_LOGIC_VECTOR(11 downto 0)
12   );
13 end ROM_polyphase;
14
15 architecture ROM_polyphase of ROM_polyphase is
16 component counter
17   port (
18        clk     : in STD_LOGIC;
19        rate_sel : in STD_LOGIC_VECTOR(1 downto 0);
20        reset   : in STD_LOGIC;
21        clk_en  : out STD_LOGIC;
22        count   : out STD_LOGIC_VECTOR(2 downto 0)
23   );
24 end component;
25 component srrc_x_n
26   port (
27        clk       : in STD_LOGIC;
28        clk_en    : in STD_LOGIC;
29        fir_sel   : in STD_LOGIC_VECTOR(2 downto 0);
30        in_fir_MSB : in STD_LOGIC;
31        rate_sel  : in STD_LOGIC_VECTOR(1 downto 0);
32        reset     : in STD_LOGIC;
33        out_srrc  : out STD_LOGIC_VECTOR(11 downto 0)
34   );
35 end component;
36 signal clk_en  : STD_LOGIC;
37 signal count_n : STD_LOGIC_VECTOR (2 downto 0);
38
39 begin
40 U1 : counter
41   port map(
42        clk           => clk,
43        clk_en  => clk_en,
44        count   => count_n,
45        rate_sel      => rate_sel,
46        reset   => reset
47   );
48 U2 : srrc_x_n
49   port map(
50        clk           => clk,
51        clk_en  => clk_en,
52        fir_sel  => count_n,
53        in_fir_MSB => to_SRRC_I,
54        out_srrc      => poly_out,
55        rate_sel      => rate_sel,
56        reset   => reset
57   );
58 clk_sync <= clk;
59 end ROM_polyphase;
```

Listato F.3.2: counter.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4
5  entity counter is
6         port (
7               clk      : in std_logic;
8               reset    : in std_logic;
9               rate_sel : in std_logic_vector(1 downto 0);
10              count    : out std_logic_vector(2 downto 0);
11              clk_en   : out std_logic
12        );
13 end entity;
14
15 architecture counter_arch of counter is
16        type rate_table_type is array (0 to 2) of std_logic_vector (2 downto 0);
17        -- definisco un array con i valori dei rate diminuiti di 1
18        -- per migliorare l'implementazione
19        constant count_limit_table : rate_table_type := ("010","011","101");
20        signal TEMP_count    : std_logic_vector(2 downto 0);
21        signal TEMP_clk_en   : std_logic;
22 begin
23        process(clk, reset)
24        begin
25              if reset = '1' then
26                    TEMP_count <= "000";
27                    TEMP_clk_en <= '1' ;
28              elsif rising_edge(clk) then
29                    if (TEMP_count = count_limit_table(conv_integer(rate_sel))-1)
                        then
30              TEMP_clk_en <= '1' ;
31                          TEMP_count <= TEMP_count + 1;
32                    elsif (TEMP_count = count_limit_table(conv_integer(rate_sel))
                          )then
33              TEMP_clk_en <= '0' ;
34                          TEMP_count <= "000";
35                    else
36                          TEMP_clk_en <= '0' ;
37                          TEMP_count <= TEMP_count + 1;
38                    end if;
39              end if;
40        end process;
41        count  <= TEMP_count;
42        clk_en <= TEMP_clk_en;
43 end counter_arch;
```

Listato F.3.3: srrc_x_n.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity srrc_x_n is
5    port(
6         clk       : in STD_LOGIC;
7         clk_en    : in STD_LOGIC;
8         in_fir_MSB : in STD_LOGIC;
9         reset     : in STD_LOGIC;
10        fir_sel   : in STD_LOGIC_VECTOR(2 downto 0);
11        rate_sel  : in STD_LOGIC_VECTOR(1 downto 0);
12        out_srrc  : out STD_LOGIC_VECTOR(11 downto 0)
13   );
14 end srrc_x_n;
15
16 architecture srrc_x_n of srrc_x_n is
17 component demux_3x10
18   port (
19        clk       : in STD_LOGIC;
20        in_mux : in STD_LOGIC_VECTOR(9 downto 0);
21        sel    : in STD_LOGIC_VECTOR(1 downto 0);
22        out_0 : out STD_LOGIC_VECTOR(8 downto 0);
23        out_1 : out STD_LOGIC_VECTOR(8 downto 0);
24        out_2 : out STD_LOGIC_VECTOR(9 downto 0)
25   );
26 end component;
27 component mux_3x12
28   port (
29        clk    : in STD_LOGIC;
30        in_0   : in STD_LOGIC_VECTOR(11 downto 0);
31        in_1   : in STD_LOGIC_VECTOR(11 downto 0);
32        in_2   : in STD_LOGIC_VECTOR(11 downto 0);
33        sel    : in STD_LOGIC_VECTOR(1 downto 0);
34        out_mux : out STD_LOGIC_VECTOR(11 downto 0)
35   );
36 end component;
```

```
37  component romx3
38    port (
39         address : in STD_LOGIC_VECTOR(8 downto 0);
40         SRRC_out : out STD_LOGIC_VECTOR(11 downto 0)
41    );
42  end component;
43  component ROMx4
44    port (
45         address : in STD_LOGIC_VECTOR(8 downto 0);
46         SRRC_out : out STD_LOGIC_VECTOR(11 downto 0)
47    );
48  end component;
49  component ROMx6
50    port (
51         address : in STD_LOGIC_VECTOR(9 downto 0);
52         SRRC_out : out STD_LOGIC_VECTOR(11 downto 0)
53    );
54  end component;
55  component shift_reg
56    port (
57         clk      : in STD_LOGIC;
58         clk_en   : in STD_LOGIC;
59         in_reg   : in STD_LOGIC;
60         reset    : in STD_LOGIC;
61         out_ffd_1 : out STD_LOGIC;
62         out_ffd_2 : out STD_LOGIC;
63         out_ffd_3 : out STD_LOGIC;
64         out_ffd_4 : out STD_LOGIC;
65         out_ffd_5 : out STD_LOGIC;
66         out_ffd_6 : out STD_LOGIC
67    );
68  end component;
69  signal address : STD_LOGIC_VECTOR (9 downto 0);
70  signal BUS615 : STD_LOGIC_VECTOR (11 downto 0);
71  signal BUS619 : STD_LOGIC_VECTOR (11 downto 0);
72  signal BUS623 : STD_LOGIC_VECTOR (11 downto 0);
73  signal sel    : STD_LOGIC_VECTOR (1 downto 0);
74  signal to_romx3 : STD_LOGIC_VECTOR (8 downto 0);
75  signal to_romx4 : STD_LOGIC_VECTOR (8 downto 0);
76  signal to_romx6 : STD_LOGIC_VECTOR (9 downto 0);
77
78  begin
79  U1 : shift_reg
80    port map(
81         clk      => clk,
82         clk_en   => clk_en,
83         in_reg   => address(0),
84         out_ffd_1 => address(1),
85         out_ffd_2 => address(2),
86         out_ffd_3 => address(3),
87         out_ffd_4 => address(4),
88         out_ffd_5 => address(5),
89         out_ffd_6 => address(6),
90         reset    => reset
91    );
92  U2 : demux_3x10
93    port map(
94         clk    => clk,
95         in_mux => address,
96         out_0 => to_romx3,
97         out_1 => to_romx4,
98         out_2 => to_romx6,
99         sel    => sel
100   );
101 U3 : romx3
102   port map(
103        SRRC_out => BUS615,
104        address => to_romx3
105   );
106 U4 : ROMx4
107   port map(
108        SRRC_out => BUS619,
109        address => to_romx4
110   );
111 U5 : ROMx6
112   port map(
113        SRRC_out => BUS623,
114        address => to_romx6
115   );
116 U6 : mux_3x12
117   port map(
118        clk    => clk,
119        in_0   => BUS615,
120        in_1   => BUS619,
121        in_2   => BUS623,
122        out_mux => out_srrc,
123        sel    => sel
124   );
125        address(7) <= fir_sel(0);
126        address(8) <= fir_sel(1);
127        address(9) <= fir_sel(2);
128        address(0) <= in_fir_MSB;
129        sel <= rate_sel;
```

```
130 end srrc_x_n;
```

Listato F.3.4: demux_3x10.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity demux_3x10 is
5        port(in_mux     :    in std_logic_vector(9 downto 0);
6              clk     :      in std_logic;
7              sel       :    in std_logic_vector(1 downto 0);
8              out_0     :   out std_logic_vector(8 downto 0);
9              out_1     :   out std_logic_vector(8 downto 0);
10             out_2     :    out std_logic_vector(9 downto 0)
11       );
12 end demux_3x10;
13
14 architecture demux_3x10_arch of demux_3x10 is
15 begin
16       process (sel, in_mux, clk)
17       begin
18             if rising_edge(clk) then
19                   case sel is
20                         when "00" =>   out_0 <= in_mux(8 downto 0) ;
21                         when "01" =>   out_1 <= in_mux(8 downto 0) ;
22                         when "10" =>   out_2 <= in_mux ;
23                         when others => null;
24                   end case;
25             end if;
26       end process;
27 end demux_3x10_arch ;
```

Listato F.3.5: mux_3x12.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity mux_3x12 is
5        port(in_0, in_1, in_2 : in std_logic_vector(11 downto 0);
6              clk             : in std_logic ;
7              sel               :   in std_logic_vector(1 downto 0);
8              out_mux    :   out std_logic_vector(11 downto 0));
9  end mux_3x12;
10
11 architecture mux_3x12_arch of mux_3x12 is
12 begin
13       process (sel, in_0, in_1, in_2, clk)
14       begin
15             if rising_edge(clk) then
16                   case sel is
17                         when "00"  =>   out_mux <=in_0 ;
18                         when "01"  =>   out_mux <=in_1 ;
19                         when "10"  =>   out_mux <=in_2 ;
20                         when others =>  out_mux <= "XXXXXXXXXXXX" ;
21                   end case;
22             end if;
23       end process;
24 end mux_3x12_arch;
```

Listato F.3.6: ROMx3.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity ROMx3 is
6    port(address  : in STD_LOGIC_vector(8 downto 0);
7         SRRC_out  : out STD_LOGIC_VECTOR(11 downto 0)
8    );
9  end;
10
11 architecture ROMx3_arch of ROMx3 is
12 begin
13       process(address)
14             variable addr : integer ;
15       begin
16             addr := conv_integer(address) ;
17             case addr is
```

```
18          -- somme per il FIR 0
19            when  0   => SRRC_out <= X"52B";
20            when  1   => SRRC_out <= X"52B";
21            when  2   => SRRC_out <= X"517";
22            when  3   => SRRC_out <= X"517";
23            when  4   => SRRC_out <= X"5CB";
24            when  5   => SRRC_out <= X"5CB";
25            when  6   => SRRC_out <= X"5B6";
26            when  7   => SRRC_out <= X"5B6";
27            when  8   => SRRC_out <= X"1C8";
28            when  9   => SRRC_out <= X"1C8";
29            when 10   => SRRC_out <= X"1B4";
30            when 11   => SRRC_out <= X"1B4";
31            when 12   => SRRC_out <= X"268";
32            when 13   => SRRC_out <= X"268";
33            when 14   => SRRC_out <= X"253";
34            when 15   => SRRC_out <= X"253";
35            when 16   => SRRC_out <= X"C69";
36            when 17   => SRRC_out <= X"C69";
37            when 18   => SRRC_out <= X"C55";
38            when 19   => SRRC_out <= X"C55";
39            when 20   => SRRC_out <= X"D09";
40            when 21   => SRRC_out <= X"D09";
41            when 22   => SRRC_out <= X"CF4";
42            when 23   => SRRC_out <= X"CF4";
43            when 24   => SRRC_out <= X"907";
44            when 25   => SRRC_out <= X"907";
45            when 26   => SRRC_out <= X"8F2";
46            when 27   => SRRC_out <= X"8F2";
47            when 28   => SRRC_out <= X"9A6";
48            when 29   => SRRC_out <= X"9A6";
49            when 30   => SRRC_out <= X"992";
50            when 31   => SRRC_out <= X"992";
51            when 32   => SRRC_out <= X"6FA";
52            when 33   => SRRC_out <= X"6FA";
53            when 34   => SRRC_out <= X"6E6";
54            when 35   => SRRC_out <= X"6E6";
55            when 36   => SRRC_out <= X"79A";
56            when 37   => SRRC_out <= X"79A";
57            when 38   => SRRC_out <= X"785";
58            when 39   => SRRC_out <= X"785";
59            when 40   => SRRC_out <= X"397";
60            when 41   => SRRC_out <= X"397";
61            when 42   => SRRC_out <= X"383";
62            when 43   => SRRC_out <= X"383";
63            when 44   => SRRC_out <= X"437";
64            when 45   => SRRC_out <= X"437";
65            when 46   => SRRC_out <= X"422";
66            when 47   => SRRC_out <= X"422";
67            when 48   => SRRC_out <= X"E38";
68            when 49   => SRRC_out <= X"E38";
69            when 50   => SRRC_out <= X"E24";
70            when 51   => SRRC_out <= X"E24";
71            when 52   => SRRC_out <= X"ED8";
72            when 53   => SRRC_out <= X"ED8";
73            when 54   => SRRC_out <= X"EC3";
74            when 55   => SRRC_out <= X"EC3";
75            when 56   => SRRC_out <= X"AD6";
76            when 57   => SRRC_out <= X"AD6";
77            when 58   => SRRC_out <= X"AC1";
78            when 59   => SRRC_out <= X"AC1";
79            when 60   => SRRC_out <= X"B75";
80            when 61   => SRRC_out <= X"B75";
81            when 62   => SRRC_out <= X"B61";
82            when 63   => SRRC_out <= X"B61";
83            when 64   => SRRC_out <= X"49F";
84            when 65   => SRRC_out <= X"49F";
85            when 66   => SRRC_out <= X"48B";
86            when 67   => SRRC_out <= X"48B";
87            when 68   => SRRC_out <= X"53F";
88            when 69   => SRRC_out <= X"53F";
89            when 70   => SRRC_out <= X"52A";
90            when 71   => SRRC_out <= X"52A";
91            when 72   => SRRC_out <= X"13D";
92            when 73   => SRRC_out <= X"13D";
93            when 74   => SRRC_out <= X"128";
94            when 75   => SRRC_out <= X"128";
95            when 76   => SRRC_out <= X"1DC";
96            when 77   => SRRC_out <= X"1DC";
97            when 78   => SRRC_out <= X"1C8";
98            when 79   => SRRC_out <= X"1C8";
99            when 80   => SRRC_out <= X"BDE";
100           when 81   => SRRC_out <= X"BDE";
101           when 82   => SRRC_out <= X"BC9";
102           when 83   => SRRC_out <= X"BC9";
103           when 84   => SRRC_out <= X"C7D";
104           when 85   => SRRC_out <= X"C7D";
105           when 86   => SRRC_out <= X"C69";
106           when 87   => SRRC_out <= X"C69";
107           when 88   => SRRC_out <= X"87B";
108           when 89   => SRRC_out <= X"87B";
109           when 90   => SRRC_out <= X"866";
110           when 91   => SRRC_out <= X"866";
111           when 92   => SRRC_out <= X"91A";
112           when 93   => SRRC_out <= X"91A";
113           when 94   => SRRC_out <= X"906";
114           when 95   => SRRC_out <= X"906";
115           when 96   => SRRC_out <= X"66E";
```

```
116             when 97   => SRRC_out <= X"66E";
117             when 98   => SRRC_out <= X"65A";
118             when 99   => SRRC_out <= X"65A";
119             when 100  => SRRC_out <= X"70E";
120             when 101  => SRRC_out <= X"70E";
121             when 102  => SRRC_out <= X"6F9";
122             when 103  => SRRC_out <= X"6F9";
123             when 104  => SRRC_out <= X"30C";
124             when 105  => SRRC_out <= X"30C";
125             when 106  => SRRC_out <= X"2F7";
126             when 107  => SRRC_out <= X"2F7";
127             when 108  => SRRC_out <= X"3AB";
128             when 109  => SRRC_out <= X"3AB";
129             when 110  => SRRC_out <= X"397";
130             when 111  => SRRC_out <= X"397";
131             when 112  => SRRC_out <= X"DAD";
132             when 113  => SRRC_out <= X"DAD";
133             when 114  => SRRC_out <= X"D98";
134             when 115  => SRRC_out <= X"D98";
135             when 116  => SRRC_out <= X"E4C";
136             when 117  => SRRC_out <= X"E4C";
137             when 118  => SRRC_out <= X"E38";
138             when 119  => SRRC_out <= X"E38";
139             when 120  => SRRC_out <= X"A4A";
140             when 121  => SRRC_out <= X"A4A";
141             when 122  => SRRC_out <= X"A35";
142             when 123  => SRRC_out <= X"A35";
143             when 124  => SRRC_out <= X"AE9";
144             when 125  => SRRC_out <= X"AE9";
145             when 126  => SRRC_out <= X"AD5";
146             when 127  => SRRC_out <= X"AD5";
147           -- somme per il FIR 1
148             when 128  => SRRC_out <= X"52B";
149             when 129  => SRRC_out <= X"52B";
150             when 130  => SRRC_out <= X"49F";
151             when 131  => SRRC_out <= X"49F";
152             when 132  => SRRC_out <= X"6FA";
153             when 133  => SRRC_out <= X"6FA";
154             when 134  => SRRC_out <= X"66E";
155             when 135  => SRRC_out <= X"66E";
156             when 136  => SRRC_out <= X"C69";
157             when 137  => SRRC_out <= X"C69";
158             when 138  => SRRC_out <= X"BDE";
159             when 139  => SRRC_out <= X"BDE";
160             when 140  => SRRC_out <= X"E38";
161             when 141  => SRRC_out <= X"E38";
162             when 142  => SRRC_out <= X"DAD";
163             when 143  => SRRC_out <= X"DAD";
164             when 144  => SRRC_out <= X"1C8";
165             when 145  => SRRC_out <= X"1C8";
166             when 146  => SRRC_out <= X"13D";
167             when 147  => SRRC_out <= X"13D";
168             when 148  => SRRC_out <= X"397";
169             when 149  => SRRC_out <= X"397";
170             when 150  => SRRC_out <= X"30C";
171             when 151  => SRRC_out <= X"30C";
172             when 152  => SRRC_out <= X"907";
173             when 153  => SRRC_out <= X"907";
174             when 154  => SRRC_out <= X"87B";
175             when 155  => SRRC_out <= X"87B";
176             when 156  => SRRC_out <= X"AD6";
177             when 157  => SRRC_out <= X"AD6";
178             when 158  => SRRC_out <= X"A4A";
179             when 159  => SRRC_out <= X"A4A";
180             when 160  => SRRC_out <= X"5CB";
181             when 161  => SRRC_out <= X"5CB";
182             when 162  => SRRC_out <= X"53F";
183             when 163  => SRRC_out <= X"53F";
184             when 164  => SRRC_out <= X"79A";
185             when 165  => SRRC_out <= X"79A";
186             when 166  => SRRC_out <= X"70E";
187             when 167  => SRRC_out <= X"70E";
188             when 168  => SRRC_out <= X"D09";
189             when 169  => SRRC_out <= X"D09";
190             when 170  => SRRC_out <= X"C7D";
191             when 171  => SRRC_out <= X"C7D";
192             when 172  => SRRC_out <= X"ED8";
193             when 173  => SRRC_out <= X"ED8";
194             when 174  => SRRC_out <= X"E4C";
195             when 175  => SRRC_out <= X"E4C";
196             when 176  => SRRC_out <= X"268";
197             when 177  => SRRC_out <= X"268";
198             when 178  => SRRC_out <= X"1DC";
199             when 179  => SRRC_out <= X"1DC";
200             when 180  => SRRC_out <= X"437";
201             when 181  => SRRC_out <= X"437";
202             when 182  => SRRC_out <= X"3AB";
203             when 183  => SRRC_out <= X"3AB";
204             when 184  => SRRC_out <= X"9A6";
205             when 185  => SRRC_out <= X"9A6";
206             when 186  => SRRC_out <= X"91A";
207             when 187  => SRRC_out <= X"91A";
208             when 188  => SRRC_out <= X"B75";
209             when 189  => SRRC_out <= X"B75";
210             when 190  => SRRC_out <= X"AE9";
211             when 191  => SRRC_out <= X"AE9";
212             when 192  => SRRC_out <= X"517";
213             when 193  => SRRC_out <= X"517";
```

```
214             when 194  => SRRC_out <= X"48B";
215             when 195  => SRRC_out <= X"48B";
216             when 196  => SRRC_out <= X"6E6";
217             when 197  => SRRC_out <= X"6E6";
218             when 198  => SRRC_out <= X"65A";
219             when 199  => SRRC_out <= X"65A";
220             when 200  => SRRC_out <= X"C55";
221             when 201  => SRRC_out <= X"C55";
222             when 202  => SRRC_out <= X"BC9";
223             when 203  => SRRC_out <= X"BC9";
224             when 204  => SRRC_out <= X"E24";
225             when 205  => SRRC_out <= X"E24";
226             when 206  => SRRC_out <= X"D98";
227             when 207  => SRRC_out <= X"D98";
228             when 208  => SRRC_out <= X"1B4";
229             when 209  => SRRC_out <= X"1B4";
230             when 210  => SRRC_out <= X"128";
231             when 211  => SRRC_out <= X"128";
232             when 212  => SRRC_out <= X"383";
233             when 213  => SRRC_out <= X"383";
234             when 214  => SRRC_out <= X"2F7";
235             when 215  => SRRC_out <= X"2F7";
236             when 216  => SRRC_out <= X"8F2";
237             when 217  => SRRC_out <= X"8F2";
238             when 218  => SRRC_out <= X"866";
239             when 219  => SRRC_out <= X"866";
240             when 220  => SRRC_out <= X"AC1";
241             when 221  => SRRC_out <= X"AC1";
242             when 222  => SRRC_out <= X"A35";
243             when 223  => SRRC_out <= X"A35";
244             when 224  => SRRC_out <= X"5B6";
245             when 225  => SRRC_out <= X"5B6";
246             when 226  => SRRC_out <= X"52A";
247             when 227  => SRRC_out <= X"52A";
248             when 228  => SRRC_out <= X"785";
249             when 229  => SRRC_out <= X"785";
250             when 230  => SRRC_out <= X"6F9";
251             when 231  => SRRC_out <= X"6F9";
252             when 232  => SRRC_out <= X"CF4";
253             when 233  => SRRC_out <= X"CF4";
254             when 234  => SRRC_out <= X"C69";
255             when 235  => SRRC_out <= X"C69";
256             when 236  => SRRC_out <= X"EC3";
257             when 237  => SRRC_out <= X"EC3";
258             when 238  => SRRC_out <= X"E38";
259             when 239  => SRRC_out <= X"E38";
260             when 240  => SRRC_out <= X"253";
261             when 241  => SRRC_out <= X"253";
262             when 242  => SRRC_out <= X"1C8";
263             when 243  => SRRC_out <= X"1C8";
264             when 244  => SRRC_out <= X"422";
265             when 245  => SRRC_out <= X"422";
266             when 246  => SRRC_out <= X"397";
267             when 247  => SRRC_out <= X"397";
268             when 248  => SRRC_out <= X"992";
269             when 249  => SRRC_out <= X"992";
270             when 250  => SRRC_out <= X"906";
271             when 251  => SRRC_out <= X"906";
272             when 252  => SRRC_out <= X"B61";
273             when 253  => SRRC_out <= X"B61";
274             when 254  => SRRC_out <= X"AD5";
275             when 255  => SRRC_out <= X"AD5";
276           -- somme per il FIR 2
277             when 256  => SRRC_out <= X"500";
278             when 257  => SRRC_out <= X"541";
279             when 258  => SRRC_out <= X"485";
280             when 259  => SRRC_out <= X"4C6";
281             when 260  => SRRC_out <= X"5D7";
282             when 261  => SRRC_out <= X"618";
283             when 262  => SRRC_out <= X"55D";
284             when 263  => SRRC_out <= X"59E";
285             when 264  => SRRC_out <= X"9C4";
286             when 265  => SRRC_out <= X"A05";
287             when 266  => SRRC_out <= X"94A";
288             when 267  => SRRC_out <= X"98B";
289             when 268  => SRRC_out <= X"A9C";
290             when 269  => SRRC_out <= X"ADC";
291             when 270  => SRRC_out <= X"A21";
292             when 271  => SRRC_out <= X"A62";
293             when 272  => SRRC_out <= X"5D7";
294             when 273  => SRRC_out <= X"618";
295             when 274  => SRRC_out <= X"55D";
296             when 275  => SRRC_out <= X"59E";
297             when 276  => SRRC_out <= X"6AF";
298             when 277  => SRRC_out <= X"6F0";
299             when 278  => SRRC_out <= X"634";
300             when 279  => SRRC_out <= X"675";
301             when 280  => SRRC_out <= X"A9C";
302             when 281  => SRRC_out <= X"ADC";
303             when 282  => SRRC_out <= X"A21";
304             when 283  => SRRC_out <= X"A62";
305             when 284  => SRRC_out <= X"B73";
306             when 285  => SRRC_out <= X"BB4";
307             when 286  => SRRC_out <= X"AF9";
308             when 287  => SRRC_out <= X"B3A";
309             when 288  => SRRC_out <= X"485";
310             when 289  => SRRC_out <= X"4C6";
311             when 290  => SRRC_out <= X"40B";
```

```
312             when 291  => SRRC_out <= X"44C";
313             when 292  => SRRC_out <= X"55D";
314             when 293  => SRRC_out <= X"59E";
315             when 294  => SRRC_out <= X"4E3";
316             when 295  => SRRC_out <= X"524";
317             when 296  => SRRC_out <= X"94A";
318             when 297  => SRRC_out <= X"98B";
319             when 298  => SRRC_out <= X"8CF";
320             when 299  => SRRC_out <= X"910";
321             when 300  => SRRC_out <= X"A21";
322             when 301  => SRRC_out <= X"A62";
323             when 302  => SRRC_out <= X"9A7";
324             when 303  => SRRC_out <= X"9E8";
325             when 304  => SRRC_out <= X"55D";
326             when 305  => SRRC_out <= X"59E";
327             when 306  => SRRC_out <= X"4E3";
328             when 307  => SRRC_out <= X"524";
329             when 308  => SRRC_out <= X"634";
330             when 309  => SRRC_out <= X"675";
331             when 310  => SRRC_out <= X"5BA";
332             when 311  => SRRC_out <= X"5FB";
333             when 312  => SRRC_out <= X"A21";
334             when 313  => SRRC_out <= X"A62";
335             when 314  => SRRC_out <= X"9A7";
336             when 315  => SRRC_out <= X"9E8";
337             when 316  => SRRC_out <= X"AF9";
338             when 317  => SRRC_out <= X"B3A";
339             when 318  => SRRC_out <= X"A7E";
340             when 319  => SRRC_out <= X"ABF";
341             when 320  => SRRC_out <= X"541";
342             when 321  => SRRC_out <= X"582";
343             when 322  => SRRC_out <= X"4C6";
344             when 323  => SRRC_out <= X"507";
345             when 324  => SRRC_out <= X"618";
346             when 325  => SRRC_out <= X"659";
347             when 326  => SRRC_out <= X"59E";
348             when 327  => SRRC_out <= X"5DF";
349             when 328  => SRRC_out <= X"A05";
350             when 329  => SRRC_out <= X"A46";
351             when 330  => SRRC_out <= X"98B";
352             when 331  => SRRC_out <= X"9CC";
353             when 332  => SRRC_out <= X"ADC";
354             when 333  => SRRC_out <= X"B1D";
355             when 334  => SRRC_out <= X"A62";
356             when 335  => SRRC_out <= X"AA3";
357             when 336  => SRRC_out <= X"618";
358             when 337  => SRRC_out <= X"659";
359             when 338  => SRRC_out <= X"59E";
360             when 339  => SRRC_out <= X"5DF";
361             when 340  => SRRC_out <= X"6F0";
362             when 341  => SRRC_out <= X"731";
363             when 342  => SRRC_out <= X"675";
364             when 343  => SRRC_out <= X"6B6";
365             when 344  => SRRC_out <= X"ADC";
366             when 345  => SRRC_out <= X"B1D";
367             when 346  => SRRC_out <= X"A62";
368             when 347  => SRRC_out <= X"AA3";
369             when 348  => SRRC_out <= X"BB4";
370             when 349  => SRRC_out <= X"BF5";
371             when 350  => SRRC_out <= X"B3A";
372             when 351  => SRRC_out <= X"B7B";
373             when 352  => SRRC_out <= X"4C6";
374             when 353  => SRRC_out <= X"507";
375             when 354  => SRRC_out <= X"44C";
376             when 355  => SRRC_out <= X"48D";
377             when 356  => SRRC_out <= X"59E";
378             when 357  => SRRC_out <= X"5DF";
379             when 358  => SRRC_out <= X"524";
380             when 359  => SRRC_out <= X"564";
381             when 360  => SRRC_out <= X"98B";
382             when 361  => SRRC_out <= X"9CC";
383             when 362  => SRRC_out <= X"910";
384             when 363  => SRRC_out <= X"951";
385             when 364  => SRRC_out <= X"A62";
386             when 365  => SRRC_out <= X"AA3";
387             when 366  => SRRC_out <= X"9E8";
388             when 367  => SRRC_out <= X"A29";
389             when 368  => SRRC_out <= X"59E";
390             when 369  => SRRC_out <= X"5DF";
391             when 370  => SRRC_out <= X"524";
392             when 371  => SRRC_out <= X"564";
393             when 372  => SRRC_out <= X"675";
394             when 373  => SRRC_out <= X"6B6";
395             when 374  => SRRC_out <= X"5FB";
396             when 375  => SRRC_out <= X"63C";
397             when 376  => SRRC_out <= X"A62";
398             when 377  => SRRC_out <= X"AA3";
399             when 378  => SRRC_out <= X"9E8";
400             when 379  => SRRC_out <= X"A29";
401             when 380  => SRRC_out <= X"B3A";
402             when 381  => SRRC_out <= X"B7B";
403             when 382  => SRRC_out <= X"ABF";
404             when 383  => SRRC_out <= X"B00";
405             when OTHERS => SRRC_out <= X"000";
406             end case;
407         end process;
408 end ROMx3_arch;
```

Listato F.3.7: ROMx4.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity ROMx4 is
6    port(address  : in STD_LOGIC_vector(8 downto 0);
7         SRRC_out  : out STD_LOGIC_VECTOR(11 downto 0)
8    );
9  end;
10
11 architecture ROMx4_arch of ROMx4 is
12 begin
13        process(address)
14              variable addr : integer ;
15        begin
16              addr := conv_integer(address) ;
17              case addr is
18        -- somme per il FIR 0
19           when  0   => SRRC_out <= X"4F1";
20           when  1   => SRRC_out <= X"4F1";
21           when  2   => SRRC_out <= X"500";
22           when  3   => SRRC_out <= X"500";
23           when  4   => SRRC_out <= X"53B";
24           when  5   => SRRC_out <= X"53B";
25           when  6   => SRRC_out <= X"549";
26           when  7   => SRRC_out <= X"549";
27           when  8   => SRRC_out <= X"2EA";
28           when  9   => SRRC_out <= X"2EA";
29           when 10   => SRRC_out <= X"2F9";
30           when 11   => SRRC_out <= X"2F9";
31           when 12   => SRRC_out <= X"334";
32           when 13   => SRRC_out <= X"334";
33           when 14   => SRRC_out <= X"342";
34           when 15   => SRRC_out <= X"342";
35           when 16   => SRRC_out <= X"B95";
36           when 17   => SRRC_out <= X"B95";
37           when 18   => SRRC_out <= X"BA3";
38           when 19   => SRRC_out <= X"BA3";
39           when 20   => SRRC_out <= X"BDE";
40           when 21   => SRRC_out <= X"BDE";
41           when 22   => SRRC_out <= X"BED";
42           when 23   => SRRC_out <= X"BED";
43           when 24   => SRRC_out <= X"98E";
44           when 25   => SRRC_out <= X"98E";
45           when 26   => SRRC_out <= X"99C";
46           when 27   => SRRC_out <= X"99C";
47           when 28   => SRRC_out <= X"9D7";
48           when 29   => SRRC_out <= X"9D7";
49           when 30   => SRRC_out <= X"9E6";
50           when 31   => SRRC_out <= X"9E6";
51           when 32   => SRRC_out <= X"6BD";
52           when 33   => SRRC_out <= X"6BD";
53           when 34   => SRRC_out <= X"6CB";
54           when 35   => SRRC_out <= X"6CB";
55           when 36   => SRRC_out <= X"706";
56           when 37   => SRRC_out <= X"706";
57           when 38   => SRRC_out <= X"715";
58           when 39   => SRRC_out <= X"715";
59           when 40   => SRRC_out <= X"4B6";
60           when 41   => SRRC_out <= X"4B6";
61           when 42   => SRRC_out <= X"4C4";
62           when 43   => SRRC_out <= X"4C4";
63           when 44   => SRRC_out <= X"4FF";
64           when 45   => SRRC_out <= X"4FF";
65           when 46   => SRRC_out <= X"50E";
66           when 47   => SRRC_out <= X"50E";
67           when 48   => SRRC_out <= X"D60";
68           when 49   => SRRC_out <= X"D60";
69           when 50   => SRRC_out <= X"D6E";
70           when 51   => SRRC_out <= X"D6E";
71           when 52   => SRRC_out <= X"DAA";
72           when 53   => SRRC_out <= X"DAA";
73           when 54   => SRRC_out <= X"DB8";
74           when 55   => SRRC_out <= X"DB8";
75           when 56   => SRRC_out <= X"B59";
76           when 57   => SRRC_out <= X"B59";
77           when 58   => SRRC_out <= X"B67";
78           when 59   => SRRC_out <= X"B67";
79           when 60   => SRRC_out <= X"BA3";
80           when 61   => SRRC_out <= X"BA3";
81           when 62   => SRRC_out <= X"BB1";
82           when 63   => SRRC_out <= X"BB1";
83           when 64   => SRRC_out <= X"44F";
84           when 65   => SRRC_out <= X"44F";
85           when 66   => SRRC_out <= X"45D";
86           when 67   => SRRC_out <= X"45D";
87           when 68   => SRRC_out <= X"499";
88           when 69   => SRRC_out <= X"499";
```

```
89            when 70   => SRRC_out <= X"4A7";
90            when 71   => SRRC_out <= X"4A7";
91            when 72   => SRRC_out <= X"248";
92            when 73   => SRRC_out <= X"248";
93            when 74   => SRRC_out <= X"256";
94            when 75   => SRRC_out <= X"256";
95            when 76   => SRRC_out <= X"292";
96            when 77   => SRRC_out <= X"292";
97            when 78   => SRRC_out <= X"2A0";
98            when 79   => SRRC_out <= X"2A0";
99            when 80   => SRRC_out <= X"AF2";
100           when 81   => SRRC_out <= X"AF2";
101           when 82   => SRRC_out <= X"B01";
102           when 83   => SRRC_out <= X"B01";
103           when 84   => SRRC_out <= X"B3C";
104           when 85   => SRRC_out <= X"B3C";
105           when 86   => SRRC_out <= X"B4A";
106           when 87   => SRRC_out <= X"B4A";
107           when 88   => SRRC_out <= X"8EB";
108           when 89   => SRRC_out <= X"8EB";
109           when 90   => SRRC_out <= X"8FA";
110           when 91   => SRRC_out <= X"8FA";
111           when 92   => SRRC_out <= X"935";
112           when 93   => SRRC_out <= X"935";
113           when 94   => SRRC_out <= X"943";
114           when 95   => SRRC_out <= X"943";
115           when 96   => SRRC_out <= X"61A";
116           when 97   => SRRC_out <= X"61A";
117           when 98   => SRRC_out <= X"629";
118           when 99   => SRRC_out <= X"629";
119           when 100  => SRRC_out <= X"664";
120           when 101  => SRRC_out <= X"664";
121           when 102  => SRRC_out <= X"672";
122           when 103  => SRRC_out <= X"672";
123           when 104  => SRRC_out <= X"413";
124           when 105  => SRRC_out <= X"413";
125           when 106  => SRRC_out <= X"422";
126           when 107  => SRRC_out <= X"422";
127           when 108  => SRRC_out <= X"45D";
128           when 109  => SRRC_out <= X"45D";
129           when 110  => SRRC_out <= X"46B";
130           when 111  => SRRC_out <= X"46B";
131           when 112  => SRRC_out <= X"CBE";
132           when 113  => SRRC_out <= X"CBE";
133           when 114  => SRRC_out <= X"CCC";
134           when 115  => SRRC_out <= X"CCC";
135           when 116  => SRRC_out <= X"D07";
136           when 117  => SRRC_out <= X"D07";
137           when 118  => SRRC_out <= X"D16";
138           when 119  => SRRC_out <= X"D16";
139           when 120  => SRRC_out <= X"AB7";
140           when 121  => SRRC_out <= X"AB7";
141           when 122  => SRRC_out <= X"AC5";
142           when 123  => SRRC_out <= X"AC5";
143           when 124  => SRRC_out <= X"B00";
144           when 125  => SRRC_out <= X"B00";
145           when 126  => SRRC_out <= X"B0F";
146           when 127  => SRRC_out <= X"B0F";
147       -- somme per il FIR 1
148           when 128  => SRRC_out <= X"4F5";
149           when 129  => SRRC_out <= X"4F5";
150           when 130  => SRRC_out <= X"49D";
151           when 131  => SRRC_out <= X"49D";
152           when 132  => SRRC_out <= X"647";
153           when 133  => SRRC_out <= X"647";
154           when 134  => SRRC_out <= X"5EF";
155           when 135  => SRRC_out <= X"5EF";
156           when 136  => SRRC_out <= X"F07";
157           when 137  => SRRC_out <= X"F07";
158           when 138  => SRRC_out <= X"EAE";
159           when 139  => SRRC_out <= X"EAE";
160           when 140  => SRRC_out <= X"059";
161           when 141  => SRRC_out <= X"059";
162           when 142  => SRRC_out <= X"000";
163           when 143  => SRRC_out <= X"000";
164           when 144  => SRRC_out <= X"F07";
165           when 145  => SRRC_out <= X"F07";
166           when 146  => SRRC_out <= X"EAE";
167           when 147  => SRRC_out <= X"EAE";
168           when 148  => SRRC_out <= X"059";
169           when 149  => SRRC_out <= X"059";
170           when 150  => SRRC_out <= X"000";
171           when 151  => SRRC_out <= X"000";
172           when 152  => SRRC_out <= X"918";
173           when 153  => SRRC_out <= X"918";
174           when 154  => SRRC_out <= X"8BF";
175           when 155  => SRRC_out <= X"8BF";
176           when 156  => SRRC_out <= X"A6A";
177           when 157  => SRRC_out <= X"A6A";
178           when 158  => SRRC_out <= X"A11";
179           when 159  => SRRC_out <= X"A11";
180           when 160  => SRRC_out <= X"647";
181           when 161  => SRRC_out <= X"647";
182           when 162  => SRRC_out <= X"5EF";
183           when 163  => SRRC_out <= X"5EF";
184           when 164  => SRRC_out <= X"79A";
185           when 165  => SRRC_out <= X"79A";
186           when 166  => SRRC_out <= X"741";
```

```
187             when 167  => SRRC_out <= X"741";
188             when 168  => SRRC_out <= X"059";
189             when 169  => SRRC_out <= X"059";
190             when 170  => SRRC_out <= X"000";
191             when 171  => SRRC_out <= X"000";
192             when 172  => SRRC_out <= X"1AB";
193             when 173  => SRRC_out <= X"1AB";
194             when 174  => SRRC_out <= X"152";
195             when 175  => SRRC_out <= X"152";
196             when 176  => SRRC_out <= X"059";
197             when 177  => SRRC_out <= X"059";
198             when 178  => SRRC_out <= X"000";
199             when 179  => SRRC_out <= X"000";
200             when 180  => SRRC_out <= X"1AB";
201             when 181  => SRRC_out <= X"1AB";
202             when 182  => SRRC_out <= X"152";
203             when 183  => SRRC_out <= X"152";
204             when 184  => SRRC_out <= X"A6A";
205             when 185  => SRRC_out <= X"A6A";
206             when 186  => SRRC_out <= X"A11";
207             when 187  => SRRC_out <= X"A11";
208             when 188  => SRRC_out <= X"BBC";
209             when 189  => SRRC_out <= X"BBC";
210             when 190  => SRRC_out <= X"B63";
211             when 191  => SRRC_out <= X"B63";
212             when 192  => SRRC_out <= X"49D";
213             when 193  => SRRC_out <= X"49D";
214             when 194  => SRRC_out <= X"444";
215             when 195  => SRRC_out <= X"444";
216             when 196  => SRRC_out <= X"5EF";
217             when 197  => SRRC_out <= X"5EF";
218             when 198  => SRRC_out <= X"596";
219             when 199  => SRRC_out <= X"596";
220             when 200  => SRRC_out <= X"EAE";
221             when 201  => SRRC_out <= X"EAE";
222             when 202  => SRRC_out <= X"E55";
223             when 203  => SRRC_out <= X"E55";
224             when 204  => SRRC_out <= X"000";
225             when 205  => SRRC_out <= X"000";
226             when 206  => SRRC_out <= X"FA7";
227             when 207  => SRRC_out <= X"FA7";
228             when 208  => SRRC_out <= X"EAE";
229             when 209  => SRRC_out <= X"EAE";
230             when 210  => SRRC_out <= X"E55";
231             when 211  => SRRC_out <= X"E55";
232             when 212  => SRRC_out <= X"000";
233             when 213  => SRRC_out <= X"000";
234             when 214  => SRRC_out <= X"FA7";
235             when 215  => SRRC_out <= X"FA7";
236             when 216  => SRRC_out <= X"8BF";
237             when 217  => SRRC_out <= X"8BF";
238             when 218  => SRRC_out <= X"866";
239             when 219  => SRRC_out <= X"866";
240             when 220  => SRRC_out <= X"A11";
241             when 221  => SRRC_out <= X"A11";
242             when 222  => SRRC_out <= X"9B9";
243             when 223  => SRRC_out <= X"9B9";
244             when 224  => SRRC_out <= X"5EF";
245             when 225  => SRRC_out <= X"5EF";
246             when 226  => SRRC_out <= X"596";
247             when 227  => SRRC_out <= X"596";
248             when 228  => SRRC_out <= X"741";
249             when 229  => SRRC_out <= X"741";
250             when 230  => SRRC_out <= X"6E8";
251             when 231  => SRRC_out <= X"6E8";
252             when 232  => SRRC_out <= X"000";
253             when 233  => SRRC_out <= X"000";
254             when 234  => SRRC_out <= X"FA7";
255             when 235  => SRRC_out <= X"FA7";
256             when 236  => SRRC_out <= X"152";
257             when 237  => SRRC_out <= X"152";
258             when 238  => SRRC_out <= X"0F9";
259             when 239  => SRRC_out <= X"0F9";
260             when 240  => SRRC_out <= X"000";
261             when 241  => SRRC_out <= X"000";
262             when 242  => SRRC_out <= X"FA7";
263             when 243  => SRRC_out <= X"FA7";
264             when 244  => SRRC_out <= X"152";
265             when 245  => SRRC_out <= X"152";
266             when 246  => SRRC_out <= X"0F9";
267             when 247  => SRRC_out <= X"0F9";
268             when 248  => SRRC_out <= X"A11";
269             when 249  => SRRC_out <= X"A11";
270             when 250  => SRRC_out <= X"9B9";
271             when 251  => SRRC_out <= X"9B9";
272             when 252  => SRRC_out <= X"B63";
273             when 253  => SRRC_out <= X"B63";
274             when 254  => SRRC_out <= X"B0B";
275             when 255  => SRRC_out <= X"B0B";
276           -- somme per il FIR 2
277             when 256  => SRRC_out <= X"4F1";
278             when 257  => SRRC_out <= X"4F1";
279             when 258  => SRRC_out <= X"44F";
280             when 259  => SRRC_out <= X"44F";
281             when 260  => SRRC_out <= X"6BD";
282             when 261  => SRRC_out <= X"6BD";
283             when 262  => SRRC_out <= X"61A";
284             when 263  => SRRC_out <= X"61A";
```

```
285             when 264  => SRRC_out <= X"B95";
286             when 265  => SRRC_out <= X"B95";
287             when 266  => SRRC_out <= X"AF2";
288             when 267  => SRRC_out <= X"AF2";
289             when 268  => SRRC_out <= X"D60";
290             when 269  => SRRC_out <= X"D60";
291             when 270  => SRRC_out <= X"CBE";
292             when 271  => SRRC_out <= X"CBE";
293             when 272  => SRRC_out <= X"2EA";
294             when 273  => SRRC_out <= X"2EA";
295             when 274  => SRRC_out <= X"248";
296             when 275  => SRRC_out <= X"248";
297             when 276  => SRRC_out <= X"4B6";
298             when 277  => SRRC_out <= X"4B6";
299             when 278  => SRRC_out <= X"413";
300             when 279  => SRRC_out <= X"413";
301             when 280  => SRRC_out <= X"98E";
302             when 281  => SRRC_out <= X"98E";
303             when 282  => SRRC_out <= X"8EB";
304             when 283  => SRRC_out <= X"8EB";
305             when 284  => SRRC_out <= X"B59";
306             when 285  => SRRC_out <= X"B59";
307             when 286  => SRRC_out <= X"AB7";
308             when 287  => SRRC_out <= X"AB7";
309             when 288  => SRRC_out <= X"53B";
310             when 289  => SRRC_out <= X"53B";
311             when 290  => SRRC_out <= X"499";
312             when 291  => SRRC_out <= X"499";
313             when 292  => SRRC_out <= X"706";
314             when 293  => SRRC_out <= X"706";
315             when 294  => SRRC_out <= X"664";
316             when 295  => SRRC_out <= X"664";
317             when 296  => SRRC_out <= X"BDE";
318             when 297  => SRRC_out <= X"BDE";
319             when 298  => SRRC_out <= X"B3C";
320             when 299  => SRRC_out <= X"B3C";
321             when 300  => SRRC_out <= X"DAA";
322             when 301  => SRRC_out <= X"DAA";
323             when 302  => SRRC_out <= X"D07";
324             when 303  => SRRC_out <= X"D07";
325             when 304  => SRRC_out <= X"334";
326             when 305  => SRRC_out <= X"334";
327             when 306  => SRRC_out <= X"292";
328             when 307  => SRRC_out <= X"292";
329             when 308  => SRRC_out <= X"4FF";
330             when 309  => SRRC_out <= X"4FF";
331             when 310  => SRRC_out <= X"45D";
332             when 311  => SRRC_out <= X"45D";
333             when 312  => SRRC_out <= X"9D7";
334             when 313  => SRRC_out <= X"9D7";
335             when 314  => SRRC_out <= X"935";
336             when 315  => SRRC_out <= X"935";
337             when 316  => SRRC_out <= X"BA3";
338             when 317  => SRRC_out <= X"BA3";
339             when 318  => SRRC_out <= X"B00";
340             when 319  => SRRC_out <= X"B00";
341             when 320  => SRRC_out <= X"500";
342             when 321  => SRRC_out <= X"500";
343             when 322  => SRRC_out <= X"45D";
344             when 323  => SRRC_out <= X"45D";
345             when 324  => SRRC_out <= X"6CB";
346             when 325  => SRRC_out <= X"6CB";
347             when 326  => SRRC_out <= X"629";
348             when 327  => SRRC_out <= X"629";
349             when 328  => SRRC_out <= X"BA3";
350             when 329  => SRRC_out <= X"BA3";
351             when 330  => SRRC_out <= X"B01";
352             when 331  => SRRC_out <= X"B01";
353             when 332  => SRRC_out <= X"D6E";
354             when 333  => SRRC_out <= X"D6E";
355             when 334  => SRRC_out <= X"CCC";
356             when 335  => SRRC_out <= X"CCC";
357             when 336  => SRRC_out <= X"2F9";
358             when 337  => SRRC_out <= X"2F9";
359             when 338  => SRRC_out <= X"256";
360             when 339  => SRRC_out <= X"256";
361             when 340  => SRRC_out <= X"4C4";
362             when 341  => SRRC_out <= X"4C4";
363             when 342  => SRRC_out <= X"422";
364             when 343  => SRRC_out <= X"422";
365             when 344  => SRRC_out <= X"99C";
366             when 345  => SRRC_out <= X"99C";
367             when 346  => SRRC_out <= X"8FA";
368             when 347  => SRRC_out <= X"8FA";
369             when 348  => SRRC_out <= X"B67";
370             when 349  => SRRC_out <= X"B67";
371             when 350  => SRRC_out <= X"AC5";
372             when 351  => SRRC_out <= X"AC5";
373             when 352  => SRRC_out <= X"549";
374             when 353  => SRRC_out <= X"549";
375             when 354  => SRRC_out <= X"4A7";
376             when 355  => SRRC_out <= X"4A7";
377             when 356  => SRRC_out <= X"715";
378             when 357  => SRRC_out <= X"715";
379             when 358  => SRRC_out <= X"672";
380             when 359  => SRRC_out <= X"672";
381             when 360  => SRRC_out <= X"BED";
382             when 361  => SRRC_out <= X"BED";
```

```
383             when 362  => SRRC_out <= X"B4A";
384             when 363  => SRRC_out <= X"B4A";
385             when 364  => SRRC_out <= X"DB8";
386             when 365  => SRRC_out <= X"DB8";
387             when 366  => SRRC_out <= X"D16";
388             when 367  => SRRC_out <= X"D16";
389             when 368  => SRRC_out <= X"342";
390             when 369  => SRRC_out <= X"342";
391             when 370  => SRRC_out <= X"2A0";
392             when 371  => SRRC_out <= X"2A0";
393             when 372  => SRRC_out <= X"50E";
394             when 373  => SRRC_out <= X"50E";
395             when 374  => SRRC_out <= X"46B";
396             when 375  => SRRC_out <= X"46B";
397             when 376  => SRRC_out <= X"9E6";
398             when 377  => SRRC_out <= X"9E6";
399             when 378  => SRRC_out <= X"943";
400             when 379  => SRRC_out <= X"943";
401             when 380  => SRRC_out <= X"BB1";
402             when 381  => SRRC_out <= X"BB1";
403             when 382  => SRRC_out <= X"B0F";
404             when 383  => SRRC_out <= X"B0F";
405           -- somme per il FIR 3
406             when 384  => SRRC_out <= X"4BC";
407             when 385  => SRRC_out <= X"50D";
408             when 386  => SRRC_out <= X"443";
409             when 387  => SRRC_out <= X"493";
410             when 388  => SRRC_out <= X"586";
411             when 389  => SRRC_out <= X"5D6";
412             when 390  => SRRC_out <= X"50D";
413             when 391  => SRRC_out <= X"55D";
414             when 392  => SRRC_out <= X"A02";
415             when 393  => SRRC_out <= X"A53";
416             when 394  => SRRC_out <= X"989";
417             when 395  => SRRC_out <= X"9D9";
418             when 396  => SRRC_out <= X"ACC";
419             when 397  => SRRC_out <= X"B1C";
420             when 398  => SRRC_out <= X"A53";
421             when 399  => SRRC_out <= X"AA3";
422             when 400  => SRRC_out <= X"586";
423             when 401  => SRRC_out <= X"5D6";
424             when 402  => SRRC_out <= X"50D";
425             when 403  => SRRC_out <= X"55D";
426             when 404  => SRRC_out <= X"64F";
427             when 405  => SRRC_out <= X"6A0";
428             when 406  => SRRC_out <= X"5D6";
429             when 407  => SRRC_out <= X"627";
430             when 408  => SRRC_out <= X"ACC";
431             when 409  => SRRC_out <= X"B1C";
432             when 410  => SRRC_out <= X"A53";
433             when 411  => SRRC_out <= X"AA3";
434             when 412  => SRRC_out <= X"B95";
435             when 413  => SRRC_out <= X"BE6";
436             when 414  => SRRC_out <= X"B1C";
437             when 415  => SRRC_out <= X"B6D";
438             when 416  => SRRC_out <= X"443";
439             when 417  => SRRC_out <= X"493";
440             when 418  => SRRC_out <= X"3CA";
441             when 419  => SRRC_out <= X"41A";
442             when 420  => SRRC_out <= X"50D";
443             when 421  => SRRC_out <= X"55D";
444             when 422  => SRRC_out <= X"494";
445             when 423  => SRRC_out <= X"4E4";
446             when 424  => SRRC_out <= X"989";
447             when 425  => SRRC_out <= X"9D9";
448             when 426  => SRRC_out <= X"910";
449             when 427  => SRRC_out <= X"960";
450             when 428  => SRRC_out <= X"A53";
451             when 429  => SRRC_out <= X"AA3";
452             when 430  => SRRC_out <= X"9DA";
453             when 431  => SRRC_out <= X"A2A";
454             when 432  => SRRC_out <= X"50D";
455             when 433  => SRRC_out <= X"55D";
456             when 434  => SRRC_out <= X"494";
457             when 435  => SRRC_out <= X"4E4";
458             when 436  => SRRC_out <= X"5D6";
459             when 437  => SRRC_out <= X"627";
460             when 438  => SRRC_out <= X"55D";
461             when 439  => SRRC_out <= X"5AD";
462             when 440  => SRRC_out <= X"A53";
463             when 441  => SRRC_out <= X"AA3";
464             when 442  => SRRC_out <= X"9DA";
465             when 443  => SRRC_out <= X"A2A";
466             when 444  => SRRC_out <= X"B1C";
467             when 445  => SRRC_out <= X"B6D";
468             when 446  => SRRC_out <= X"AA3";
469             when 447  => SRRC_out <= X"AF3";
470             when 448  => SRRC_out <= X"50D";
471             when 449  => SRRC_out <= X"55D";
472             when 450  => SRRC_out <= X"493";
473             when 451  => SRRC_out <= X"4E4";
474             when 452  => SRRC_out <= X"5D6";
475             when 453  => SRRC_out <= X"626";
476             when 454  => SRRC_out <= X"55D";
477             when 455  => SRRC_out <= X"5AD";
478             when 456  => SRRC_out <= X"A53";
479             when 457  => SRRC_out <= X"AA3";
480             when 458  => SRRC_out <= X"9D9";
```

```
481           when 459  => SRRC_out <= X"A2A";
482           when 460  => SRRC_out <= X"B1C";
483           when 461  => SRRC_out <= X"B6C";
484           when 462  => SRRC_out <= X"AA3";
485           when 463  => SRRC_out <= X"AF3";
486           when 464  => SRRC_out <= X"5D6";
487           when 465  => SRRC_out <= X"626";
488           when 466  => SRRC_out <= X"55D";
489           when 467  => SRRC_out <= X"5AD";
490           when 468  => SRRC_out <= X"6A0";
491           when 469  => SRRC_out <= X"6F0";
492           when 470  => SRRC_out <= X"627";
493           when 471  => SRRC_out <= X"677";
494           when 472  => SRRC_out <= X"B1C";
495           when 473  => SRRC_out <= X"B6C";
496           when 474  => SRRC_out <= X"AA3";
497           when 475  => SRRC_out <= X"AF3";
498           when 476  => SRRC_out <= X"BE6";
499           when 477  => SRRC_out <= X"C36";
500           when 478  => SRRC_out <= X"B6D";
501           when 479  => SRRC_out <= X"BBD";
502           when 480  => SRRC_out <= X"493";
503           when 481  => SRRC_out <= X"4E4";
504           when 482  => SRRC_out <= X"41A";
505           when 483  => SRRC_out <= X"46B";
506           when 484  => SRRC_out <= X"55D";
507           when 485  => SRRC_out <= X"5AD";
508           when 486  => SRRC_out <= X"4E4";
509           when 487  => SRRC_out <= X"534";
510           when 488  => SRRC_out <= X"9D9";
511           when 489  => SRRC_out <= X"A2A";
512           when 490  => SRRC_out <= X"960";
513           when 491  => SRRC_out <= X"9B1";
514           when 492  => SRRC_out <= X"AA3";
515           when 493  => SRRC_out <= X"AF3";
516           when 494  => SRRC_out <= X"A2A";
517           when 495  => SRRC_out <= X"A7A";
518           when 496  => SRRC_out <= X"55D";
519           when 497  => SRRC_out <= X"5AD";
520           when 498  => SRRC_out <= X"4E4";
521           when 499  => SRRC_out <= X"534";
522           when 500  => SRRC_out <= X"627";
523           when 501  => SRRC_out <= X"677";
524           when 502  => SRRC_out <= X"5AD";
525           when 503  => SRRC_out <= X"5FE";
526           when 504  => SRRC_out <= X"AA3";
527           when 505  => SRRC_out <= X"AF3";
528           when 506  => SRRC_out <= X"A2A";
529           when 507  => SRRC_out <= X"A7A";
530           when 508  => SRRC_out <= X"B6D";
531           when 509  => SRRC_out <= X"BBD";
532           when 510  => SRRC_out <= X"AF3";
533           when 511  => SRRC_out <= X"B44";
534           when OTHERS => SRRC_out <= X"000";
535         end case;
536       end process;
537 end ROMx4_arch;
```

Listato F.3.8: ROMx6.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity ROMx6 is
6    port(address  : in STD_LOGIC_vector(9 downto 0);
7         SRRC_out  : out STD_LOGIC_VECTOR(11 downto 0)
8    );
9  end;
10
11 architecture ROMx6_arch of ROMx6 is
12 begin
13        process(address)
14              variable addr : integer ;
15        begin
16              addr := conv_integer(address) ;
17              case addr is
18        -- somme per il FIR 0
19           when  0   => SRRC_out <= X"515";
20           when  1   => SRRC_out <= X"515";
21           when  2   => SRRC_out <= X"509";
22           when  3   => SRRC_out <= X"509";
23           when  4   => SRRC_out <= X"5A3";
24           when  5   => SRRC_out <= X"5A3";
25           when  6   => SRRC_out <= X"597";
26           when  7   => SRRC_out <= X"597";
27           when  8   => SRRC_out <= X"1C7";
28           when  9   => SRRC_out <= X"1C7";
29           when 10   => SRRC_out <= X"1BB";
30           when 11   => SRRC_out <= X"1BB";
31           when 12   => SRRC_out <= X"254";
32           when 13   => SRRC_out <= X"254";
```

```
33            when 14   => SRRC_out <= X"248";
34            when 15   => SRRC_out <= X"248";
35            when 16   => SRRC_out <= X"C6C";
36            when 17   => SRRC_out <= X"C6C";
37            when 18   => SRRC_out <= X"C60";
38            when 19   => SRRC_out <= X"C60";
39            when 20   => SRRC_out <= X"CF9";
40            when 21   => SRRC_out <= X"CF9";
41            when 22   => SRRC_out <= X"CED";
42            when 23   => SRRC_out <= X"CED";
43            when 24   => SRRC_out <= X"91D";
44            when 25   => SRRC_out <= X"91D";
45            when 26   => SRRC_out <= X"911";
46            when 27   => SRRC_out <= X"911";
47            when 28   => SRRC_out <= X"9AA";
48            when 29   => SRRC_out <= X"9AA";
49            when 30   => SRRC_out <= X"99F";
50            when 31   => SRRC_out <= X"99F";
51            when 32   => SRRC_out <= X"6D8";
52            when 33   => SRRC_out <= X"6D8";
53            when 34   => SRRC_out <= X"6CC";
54            when 35   => SRRC_out <= X"6CC";
55            when 36   => SRRC_out <= X"765";
56            when 37   => SRRC_out <= X"765";
57            when 38   => SRRC_out <= X"759";
58            when 39   => SRRC_out <= X"759";
59            when 40   => SRRC_out <= X"389";
60            when 41   => SRRC_out <= X"389";
61            when 42   => SRRC_out <= X"37D";
62            when 43   => SRRC_out <= X"37D";
63            when 44   => SRRC_out <= X"416";
64            when 45   => SRRC_out <= X"416";
65            when 46   => SRRC_out <= X"40A";
66            when 47   => SRRC_out <= X"40A";
67            when 48   => SRRC_out <= X"E2E";
68            when 49   => SRRC_out <= X"E2E";
69            when 50   => SRRC_out <= X"E23";
70            when 51   => SRRC_out <= X"E23";
71            when 52   => SRRC_out <= X"EBC";
72            when 53   => SRRC_out <= X"EBC";
73            when 54   => SRRC_out <= X"EB0";
74            when 55   => SRRC_out <= X"EB0";
75            when 56   => SRRC_out <= X"AE0";
76            when 57   => SRRC_out <= X"AE0";
77            when 58   => SRRC_out <= X"AD4";
78            when 59   => SRRC_out <= X"AD4";
79            when 60   => SRRC_out <= X"B6D";
80            when 61   => SRRC_out <= X"B6D";
81            when 62   => SRRC_out <= X"B61";
82            when 63   => SRRC_out <= X"B61";
83            when 64   => SRRC_out <= X"49F";
84            when 65   => SRRC_out <= X"49F";
85            when 66   => SRRC_out <= X"493";
86            when 67   => SRRC_out <= X"493";
87            when 68   => SRRC_out <= X"52C";
88            when 69   => SRRC_out <= X"52C";
89            when 70   => SRRC_out <= X"520";
90            when 71   => SRRC_out <= X"520";
91            when 72   => SRRC_out <= X"150";
92            when 73   => SRRC_out <= X"150";
93            when 74   => SRRC_out <= X"144";
94            when 75   => SRRC_out <= X"144";
95            when 76   => SRRC_out <= X"1DD";
96            when 77   => SRRC_out <= X"1DD";
97            when 78   => SRRC_out <= X"1D2";
98            when 79   => SRRC_out <= X"1D2";
99            when 80   => SRRC_out <= X"BF6";
100           when 81   => SRRC_out <= X"BF6";
101           when 82   => SRRC_out <= X"BEA";
102           when 83   => SRRC_out <= X"BEA";
103           when 84   => SRRC_out <= X"C83";
104           when 85   => SRRC_out <= X"C83";
105           when 86   => SRRC_out <= X"C77";
106           when 87   => SRRC_out <= X"C77";
107           when 88   => SRRC_out <= X"8A7";
108           when 89   => SRRC_out <= X"8A7";
109           when 90   => SRRC_out <= X"89B";
110           when 91   => SRRC_out <= X"89B";
111           when 92   => SRRC_out <= X"934";
112           when 93   => SRRC_out <= X"934";
113           when 94   => SRRC_out <= X"928";
114           when 95   => SRRC_out <= X"928";
115           when 96   => SRRC_out <= X"661";
116           when 97   => SRRC_out <= X"661";
117           when 98   => SRRC_out <= X"656";
118           when 99   => SRRC_out <= X"656";
119           when 100  => SRRC_out <= X"6EF";
120           when 101  => SRRC_out <= X"6EF";
121           when 102  => SRRC_out <= X"6E3";
122           when 103  => SRRC_out <= X"6E3";
123           when 104  => SRRC_out <= X"313";
124           when 105  => SRRC_out <= X"313";
125           when 106  => SRRC_out <= X"307";
126           when 107  => SRRC_out <= X"307";
127           when 108  => SRRC_out <= X"3A0";
128           when 109  => SRRC_out <= X"3A0";
129           when 110  => SRRC_out <= X"394";
130           when 111  => SRRC_out <= X"394";
```

```
131             when 112  => SRRC_out <= X"DB8";
132             when 113  => SRRC_out <= X"DB8";
133             when 114  => SRRC_out <= X"DAC";
134             when 115  => SRRC_out <= X"DAC";
135             when 116  => SRRC_out <= X"E45";
136             when 117  => SRRC_out <= X"E45";
137             when 118  => SRRC_out <= X"E39";
138             when 119  => SRRC_out <= X"E39";
139             when 120  => SRRC_out <= X"A69";
140             when 121  => SRRC_out <= X"A69";
141             when 122  => SRRC_out <= X"A5D";
142             when 123  => SRRC_out <= X"A5D";
143             when 124  => SRRC_out <= X"AF7";
144             when 125  => SRRC_out <= X"AF7";
145             when 126  => SRRC_out <= X"AEB";
146             when 127  => SRRC_out <= X"AEB";
147           -- somme per il FIR 1
148             when 128  => SRRC_out <= X"513";
149             when 129  => SRRC_out <= X"513";
150             when 130  => SRRC_out <= X"4D3";
151             when 131  => SRRC_out <= X"4D3";
152             when 132  => SRRC_out <= X"656";
153             when 133  => SRRC_out <= X"656";
154             when 134  => SRRC_out <= X"616";
155             when 135  => SRRC_out <= X"616";
156             when 136  => SRRC_out <= X"EFC";
157             when 137  => SRRC_out <= X"EFC";
158             when 138  => SRRC_out <= X"EBD";
159             when 139  => SRRC_out <= X"EBD";
160             when 140  => SRRC_out <= X"040";
161             when 141  => SRRC_out <= X"040";
162             when 142  => SRRC_out <= X"000";
163             when 143  => SRRC_out <= X"000";
164             when 144  => SRRC_out <= X"EFC";
165             when 145  => SRRC_out <= X"EFC";
166             when 146  => SRRC_out <= X"EBD";
167             when 147  => SRRC_out <= X"EBD";
168             when 148  => SRRC_out <= X"040";
169             when 149  => SRRC_out <= X"040";
170             when 150  => SRRC_out <= X"000";
171             when 151  => SRRC_out <= X"000";
172             when 152  => SRRC_out <= X"8E6";
173             when 153  => SRRC_out <= X"8E6";
174             when 154  => SRRC_out <= X"8A6";
175             when 155  => SRRC_out <= X"8A6";
176             when 156  => SRRC_out <= X"A29";
177             when 157  => SRRC_out <= X"A29";
178             when 158  => SRRC_out <= X"9EA";
179             when 159  => SRRC_out <= X"9EA";
180             when 160  => SRRC_out <= X"656";
181             when 161  => SRRC_out <= X"656";
182             when 162  => SRRC_out <= X"616";
183             when 163  => SRRC_out <= X"616";
184             when 164  => SRRC_out <= X"79A";
185             when 165  => SRRC_out <= X"79A";
186             when 166  => SRRC_out <= X"75A";
187             when 167  => SRRC_out <= X"75A";
188             when 168  => SRRC_out <= X"040";
189             when 169  => SRRC_out <= X"040";
190             when 170  => SRRC_out <= X"000";
191             when 171  => SRRC_out <= X"000";
192             when 172  => SRRC_out <= X"183";
193             when 173  => SRRC_out <= X"183";
194             when 174  => SRRC_out <= X"143";
195             when 175  => SRRC_out <= X"143";
196             when 176  => SRRC_out <= X"040";
197             when 177  => SRRC_out <= X"040";
198             when 178  => SRRC_out <= X"000";
199             when 179  => SRRC_out <= X"000";
200             when 180  => SRRC_out <= X"183";
201             when 181  => SRRC_out <= X"183";
202             when 182  => SRRC_out <= X"143";
203             when 183  => SRRC_out <= X"143";
204             when 184  => SRRC_out <= X"A29";
205             when 185  => SRRC_out <= X"A29";
206             when 186  => SRRC_out <= X"9EA";
207             when 187  => SRRC_out <= X"9EA";
208             when 188  => SRRC_out <= X"B6D";
209             when 189  => SRRC_out <= X"B6D";
210             when 190  => SRRC_out <= X"B2D";
211             when 191  => SRRC_out <= X"B2D";
212             when 192  => SRRC_out <= X"4D3";
213             when 193  => SRRC_out <= X"4D3";
214             when 194  => SRRC_out <= X"493";
215             when 195  => SRRC_out <= X"493";
216             when 196  => SRRC_out <= X"616";
217             when 197  => SRRC_out <= X"616";
218             when 198  => SRRC_out <= X"5D7";
219             when 199  => SRRC_out <= X"5D7";
220             when 200  => SRRC_out <= X"EBD";
221             when 201  => SRRC_out <= X"EBD";
222             when 202  => SRRC_out <= X"E7D";
223             when 203  => SRRC_out <= X"E7D";
224             when 204  => SRRC_out <= X"000";
225             when 205  => SRRC_out <= X"000";
226             when 206  => SRRC_out <= X"FC0";
227             when 207  => SRRC_out <= X"FC0";
228             when 208  => SRRC_out <= X"EBD";
```

```
229             when 209  => SRRC_out <= X"EBD";
230             when 210  => SRRC_out <= X"E7D";
231             when 211  => SRRC_out <= X"E7D";
232             when 212  => SRRC_out <= X"000";
233             when 213  => SRRC_out <= X"000";
234             when 214  => SRRC_out <= X"FC0";
235             when 215  => SRRC_out <= X"FC0";
236             when 216  => SRRC_out <= X"8A6";
237             when 217  => SRRC_out <= X"8A6";
238             when 218  => SRRC_out <= X"866";
239             when 219  => SRRC_out <= X"866";
240             when 220  => SRRC_out <= X"9EA";
241             when 221  => SRRC_out <= X"9EA";
242             when 222  => SRRC_out <= X"9AA";
243             when 223  => SRRC_out <= X"9AA";
244             when 224  => SRRC_out <= X"616";
245             when 225  => SRRC_out <= X"616";
246             when 226  => SRRC_out <= X"5D7";
247             when 227  => SRRC_out <= X"5D7";
248             when 228  => SRRC_out <= X"75A";
249             when 229  => SRRC_out <= X"75A";
250             when 230  => SRRC_out <= X"71A";
251             when 231  => SRRC_out <= X"71A";
252             when 232  => SRRC_out <= X"000";
253             when 233  => SRRC_out <= X"000";
254             when 234  => SRRC_out <= X"FC0";
255             when 235  => SRRC_out <= X"FC0";
256             when 236  => SRRC_out <= X"143";
257             when 237  => SRRC_out <= X"143";
258             when 238  => SRRC_out <= X"104";
259             when 239  => SRRC_out <= X"104";
260             when 240  => SRRC_out <= X"000";
261             when 241  => SRRC_out <= X"000";
262             when 242  => SRRC_out <= X"FC0";
263             when 243  => SRRC_out <= X"FC0";
264             when 244  => SRRC_out <= X"143";
265             when 245  => SRRC_out <= X"143";
266             when 246  => SRRC_out <= X"104";
267             when 247  => SRRC_out <= X"104";
268             when 248  => SRRC_out <= X"9EA";
269             when 249  => SRRC_out <= X"9EA";
270             when 250  => SRRC_out <= X"9AA";
271             when 251  => SRRC_out <= X"9AA";
272             when 252  => SRRC_out <= X"B2D";
273             when 253  => SRRC_out <= X"B2D";
274             when 254  => SRRC_out <= X"AED";
275             when 255  => SRRC_out <= X"AED";
276           -- somme per il FIR 2
277             when 256  => SRRC_out <= X"515";
278             when 257  => SRRC_out <= X"515";
279             when 258  => SRRC_out <= X"49F";
280             when 259  => SRRC_out <= X"49F";
281             when 260  => SRRC_out <= X"6D8";
282             when 261  => SRRC_out <= X"6D8";
283             when 262  => SRRC_out <= X"661";
284             when 263  => SRRC_out <= X"661";
285             when 264  => SRRC_out <= X"C6C";
286             when 265  => SRRC_out <= X"C6C";
287             when 266  => SRRC_out <= X"BF6";
288             when 267  => SRRC_out <= X"BF6";
289             when 268  => SRRC_out <= X"E2E";
290             when 269  => SRRC_out <= X"E2E";
291             when 270  => SRRC_out <= X"DB8";
292             when 271  => SRRC_out <= X"DB8";
293             when 272  => SRRC_out <= X"1C7";
294             when 273  => SRRC_out <= X"1C7";
295             when 274  => SRRC_out <= X"150";
296             when 275  => SRRC_out <= X"150";
297             when 276  => SRRC_out <= X"389";
298             when 277  => SRRC_out <= X"389";
299             when 278  => SRRC_out <= X"313";
300             when 279  => SRRC_out <= X"313";
301             when 280  => SRRC_out <= X"91D";
302             when 281  => SRRC_out <= X"91D";
303             when 282  => SRRC_out <= X"8A7";
304             when 283  => SRRC_out <= X"8A7";
305             when 284  => SRRC_out <= X"AE0";
306             when 285  => SRRC_out <= X"AE0";
307             when 286  => SRRC_out <= X"A69";
308             when 287  => SRRC_out <= X"A69";
309             when 288  => SRRC_out <= X"5A3";
310             when 289  => SRRC_out <= X"5A3";
311             when 290  => SRRC_out <= X"52C";
312             when 291  => SRRC_out <= X"52C";
313             when 292  => SRRC_out <= X"765";
314             when 293  => SRRC_out <= X"765";
315             when 294  => SRRC_out <= X"6EF";
316             when 295  => SRRC_out <= X"6EF";
317             when 296  => SRRC_out <= X"CF9";
318             when 297  => SRRC_out <= X"CF9";
319             when 298  => SRRC_out <= X"C83";
320             when 299  => SRRC_out <= X"C83";
321             when 300  => SRRC_out <= X"EBC";
322             when 301  => SRRC_out <= X"EBC";
323             when 302  => SRRC_out <= X"E45";
324             when 303  => SRRC_out <= X"E45";
325             when 304  => SRRC_out <= X"254";
326             when 305  => SRRC_out <= X"254";
```

```
327             when 306  => SRRC_out <= X"1DD";
328             when 307  => SRRC_out <= X"1DD";
329             when 308  => SRRC_out <= X"416";
330             when 309  => SRRC_out <= X"416";
331             when 310  => SRRC_out <= X"3A0";
332             when 311  => SRRC_out <= X"3A0";
333             when 312  => SRRC_out <= X"9AA";
334             when 313  => SRRC_out <= X"9AA";
335             when 314  => SRRC_out <= X"934";
336             when 315  => SRRC_out <= X"934";
337             when 316  => SRRC_out <= X"B6D";
338             when 317  => SRRC_out <= X"B6D";
339             when 318  => SRRC_out <= X"AF7";
340             when 319  => SRRC_out <= X"AF7";
341             when 320  => SRRC_out <= X"509";
342             when 321  => SRRC_out <= X"509";
343             when 322  => SRRC_out <= X"493";
344             when 323  => SRRC_out <= X"493";
345             when 324  => SRRC_out <= X"6CC";
346             when 325  => SRRC_out <= X"6CC";
347             when 326  => SRRC_out <= X"656";
348             when 327  => SRRC_out <= X"656";
349             when 328  => SRRC_out <= X"C60";
350             when 329  => SRRC_out <= X"C60";
351             when 330  => SRRC_out <= X"BEA";
352             when 331  => SRRC_out <= X"BEA";
353             when 332  => SRRC_out <= X"E23";
354             when 333  => SRRC_out <= X"E23";
355             when 334  => SRRC_out <= X"DAC";
356             when 335  => SRRC_out <= X"DAC";
357             when 336  => SRRC_out <= X"1BB";
358             when 337  => SRRC_out <= X"1BB";
359             when 338  => SRRC_out <= X"144";
360             when 339  => SRRC_out <= X"144";
361             when 340  => SRRC_out <= X"37D";
362             when 341  => SRRC_out <= X"37D";
363             when 342  => SRRC_out <= X"307";
364             when 343  => SRRC_out <= X"307";
365             when 344  => SRRC_out <= X"911";
366             when 345  => SRRC_out <= X"911";
367             when 346  => SRRC_out <= X"89B";
368             when 347  => SRRC_out <= X"89B";
369             when 348  => SRRC_out <= X"AD4";
370             when 349  => SRRC_out <= X"AD4";
371             when 350  => SRRC_out <= X"A5D";
372             when 351  => SRRC_out <= X"A5D";
373             when 352  => SRRC_out <= X"597";
374             when 353  => SRRC_out <= X"597";
375             when 354  => SRRC_out <= X"520";
376             when 355  => SRRC_out <= X"520";
377             when 356  => SRRC_out <= X"759";
378             when 357  => SRRC_out <= X"759";
379             when 358  => SRRC_out <= X"6E3";
380             when 359  => SRRC_out <= X"6E3";
381             when 360  => SRRC_out <= X"CED";
382             when 361  => SRRC_out <= X"CED";
383             when 362  => SRRC_out <= X"C77";
384             when 363  => SRRC_out <= X"C77";
385             when 364  => SRRC_out <= X"EB0";
386             when 365  => SRRC_out <= X"EB0";
387             when 366  => SRRC_out <= X"E39";
388             when 367  => SRRC_out <= X"E39";
389             when 368  => SRRC_out <= X"248";
390             when 369  => SRRC_out <= X"248";
391             when 370  => SRRC_out <= X"1D2";
392             when 371  => SRRC_out <= X"1D2";
393             when 372  => SRRC_out <= X"40A";
394             when 373  => SRRC_out <= X"40A";
395             when 374  => SRRC_out <= X"394";
396             when 375  => SRRC_out <= X"394";
397             when 376  => SRRC_out <= X"99F";
398             when 377  => SRRC_out <= X"99F";
399             when 378  => SRRC_out <= X"928";
400             when 379  => SRRC_out <= X"928";
401             when 380  => SRRC_out <= X"B61";
402             when 381  => SRRC_out <= X"B61";
403             when 382  => SRRC_out <= X"AEB";
404             when 383  => SRRC_out <= X"AEB";
405         -- somme per il FIR 3
406             when 384  => SRRC_out <= X"505";
407             when 385  => SRRC_out <= X"52F";
408             when 386  => SRRC_out <= X"471";
409             when 387  => SRRC_out <= X"49B";
410             when 388  => SRRC_out <= X"6BB";
411             when 389  => SRRC_out <= X"6E6";
412             when 390  => SRRC_out <= X"628";
413             when 391  => SRRC_out <= X"652";
414             when 392  => SRRC_out <= X"A8A";
415             when 393  => SRRC_out <= X"AB4";
416             when 394  => SRRC_out <= X"9F6";
417             when 395  => SRRC_out <= X"A21";
418             when 396  => SRRC_out <= X"C41";
419             when 397  => SRRC_out <= X"C6B";
420             when 398  => SRRC_out <= X"BAD";
421             when 399  => SRRC_out <= X"BD7";
422             when 400  => SRRC_out <= X"429";
423             when 401  => SRRC_out <= X"453";
424             when 402  => SRRC_out <= X"395";
```

```
425             when 403  => SRRC_out <= X"3BF";
426             when 404  => SRRC_out <= X"5DF";
427             when 405  => SRRC_out <= X"60A";
428             when 406  => SRRC_out <= X"54B";
429             when 407  => SRRC_out <= X"576";
430             when 408  => SRRC_out <= X"9AE";
431             when 409  => SRRC_out <= X"9D8";
432             when 410  => SRRC_out <= X"91A";
433             when 411  => SRRC_out <= X"944";
434             when 412  => SRRC_out <= X"B64";
435             when 413  => SRRC_out <= X"B8F";
436             when 414  => SRRC_out <= X"AD1";
437             when 415  => SRRC_out <= X"AFB";
438             when 416  => SRRC_out <= X"4EF";
439             when 417  => SRRC_out <= X"519";
440             when 418  => SRRC_out <= X"45B";
441             when 419  => SRRC_out <= X"485";
442             when 420  => SRRC_out <= X"6A5";
443             when 421  => SRRC_out <= X"6D0";
444             when 422  => SRRC_out <= X"611";
445             when 423  => SRRC_out <= X"63C";
446             when 424  => SRRC_out <= X"A74";
447             when 425  => SRRC_out <= X"A9E";
448             when 426  => SRRC_out <= X"9E0";
449             when 427  => SRRC_out <= X"A0A";
450             when 428  => SRRC_out <= X"C2A";
451             when 429  => SRRC_out <= X"C55";
452             when 430  => SRRC_out <= X"B96";
453             when 431  => SRRC_out <= X"BC1";
454             when 432  => SRRC_out <= X"413";
455             when 433  => SRRC_out <= X"43D";
456             when 434  => SRRC_out <= X"37F";
457             when 435  => SRRC_out <= X"3A9";
458             when 436  => SRRC_out <= X"5C9";
459             when 437  => SRRC_out <= X"5F4";
460             when 438  => SRRC_out <= X"535";
461             when 439  => SRRC_out <= X"560";
462             when 440  => SRRC_out <= X"998";
463             when 441  => SRRC_out <= X"9C2";
464             when 442  => SRRC_out <= X"904";
465             when 443  => SRRC_out <= X"92E";
466             when 444  => SRRC_out <= X"B4E";
467             when 445  => SRRC_out <= X"B79";
468             when 446  => SRRC_out <= X"ABA";
469             when 447  => SRRC_out <= X"AE5";
470             when 448  => SRRC_out <= X"51B";
471             when 449  => SRRC_out <= X"546";
472             when 450  => SRRC_out <= X"487";
473             when 451  => SRRC_out <= X"4B2";
474             when 452  => SRRC_out <= X"6D2";
475             when 453  => SRRC_out <= X"6FC";
476             when 454  => SRRC_out <= X"63E";
477             when 455  => SRRC_out <= X"668";
478             when 456  => SRRC_out <= X"AA0";
479             when 457  => SRRC_out <= X"ACB";
480             when 458  => SRRC_out <= X"A0C";
481             when 459  => SRRC_out <= X"A37";
482             when 460  => SRRC_out <= X"C57";
483             when 461  => SRRC_out <= X"C81";
484             when 462  => SRRC_out <= X"BC3";
485             when 463  => SRRC_out <= X"BED";
486             when 464  => SRRC_out <= X"43F";
487             when 465  => SRRC_out <= X"46A";
488             when 466  => SRRC_out <= X"3AB";
489             when 467  => SRRC_out <= X"3D6";
490             when 468  => SRRC_out <= X"5F6";
491             when 469  => SRRC_out <= X"620";
492             when 470  => SRRC_out <= X"562";
493             when 471  => SRRC_out <= X"58C";
494             when 472  => SRRC_out <= X"9C4";
495             when 473  => SRRC_out <= X"9EF";
496             when 474  => SRRC_out <= X"930";
497             when 475  => SRRC_out <= X"95B";
498             when 476  => SRRC_out <= X"B7B";
499             when 477  => SRRC_out <= X"BA5";
500             when 478  => SRRC_out <= X"AE7";
501             when 479  => SRRC_out <= X"B11";
502             when 480  => SRRC_out <= X"505";
503             when 481  => SRRC_out <= X"52F";
504             when 482  => SRRC_out <= X"471";
505             when 483  => SRRC_out <= X"49C";
506             when 484  => SRRC_out <= X"6BC";
507             when 485  => SRRC_out <= X"6E6";
508             when 486  => SRRC_out <= X"628";
509             when 487  => SRRC_out <= X"652";
510             when 488  => SRRC_out <= X"A8A";
511             when 489  => SRRC_out <= X"AB5";
512             when 490  => SRRC_out <= X"9F6";
513             when 491  => SRRC_out <= X"A21";
514             when 492  => SRRC_out <= X"C41";
515             when 493  => SRRC_out <= X"C6B";
516             when 494  => SRRC_out <= X"BAD";
517             when 495  => SRRC_out <= X"BD7";
518             when 496  => SRRC_out <= X"429";
519             when 497  => SRRC_out <= X"453";
520             when 498  => SRRC_out <= X"395";
521             when 499  => SRRC_out <= X"3BF";
522             when 500  => SRRC_out <= X"5DF";
```

```
523             when 501  => SRRC_out <= X"60A";
524             when 502  => SRRC_out <= X"54C";
525             when 503  => SRRC_out <= X"576";
526             when 504  => SRRC_out <= X"9AE";
527             when 505  => SRRC_out <= X"9D8";
528             when 506  => SRRC_out <= X"91A";
529             when 507  => SRRC_out <= X"945";
530             when 508  => SRRC_out <= X"B65";
531             when 509  => SRRC_out <= X"B8F";
532             when 510  => SRRC_out <= X"AD1";
533             when 511  => SRRC_out <= X"AFB";
534          -- somme per il FIR 4
535             when 512  => SRRC_out <= X"508";
536             when 513  => SRRC_out <= X"52D";
537             when 514  => SRRC_out <= X"48E";
538             when 515  => SRRC_out <= X"4B4";
539             when 516  => SRRC_out <= X"5E5";
540             when 517  => SRRC_out <= X"60B";
541             when 518  => SRRC_out <= X"56C";
542             when 519  => SRRC_out <= X"591";
543             when 520  => SRRC_out <= X"9E5";
544             when 521  => SRRC_out <= X"A0B";
545             when 522  => SRRC_out <= X"96B";
546             when 523  => SRRC_out <= X"991";
547             when 524  => SRRC_out <= X"AC2";
548             when 525  => SRRC_out <= X"AE8";
549             when 526  => SRRC_out <= X"A49";
550             when 527  => SRRC_out <= X"A6F";
551             when 528  => SRRC_out <= X"5E5";
552             when 529  => SRRC_out <= X"60B";
553             when 530  => SRRC_out <= X"56C";
554             when 531  => SRRC_out <= X"591";
555             when 532  => SRRC_out <= X"6C3";
556             when 533  => SRRC_out <= X"6E9";
557             when 534  => SRRC_out <= X"649";
558             when 535  => SRRC_out <= X"66F";
559             when 536  => SRRC_out <= X"AC2";
560             when 537  => SRRC_out <= X"AE8";
561             when 538  => SRRC_out <= X"A49";
562             when 539  => SRRC_out <= X"A6F";
563             when 540  => SRRC_out <= X"BA0";
564             when 541  => SRRC_out <= X"BC6";
565             when 542  => SRRC_out <= X"B26";
566             when 543  => SRRC_out <= X"B4C";
567             when 544  => SRRC_out <= X"48E";
568             when 545  => SRRC_out <= X"4B4";
569             when 546  => SRRC_out <= X"414";
570             when 547  => SRRC_out <= X"43A";
571             when 548  => SRRC_out <= X"56C";
572             when 549  => SRRC_out <= X"591";
573             when 550  => SRRC_out <= X"4F2";
574             when 551  => SRRC_out <= X"518";
575             when 552  => SRRC_out <= X"96B";
576             when 553  => SRRC_out <= X"991";
577             when 554  => SRRC_out <= X"8F1";
578             when 555  => SRRC_out <= X"917";
579             when 556  => SRRC_out <= X"A49";
580             when 557  => SRRC_out <= X"A6F";
581             when 558  => SRRC_out <= X"9CF";
582             when 559  => SRRC_out <= X"9F5";
583             when 560  => SRRC_out <= X"56C";
584             when 561  => SRRC_out <= X"591";
585             when 562  => SRRC_out <= X"4F2";
586             when 563  => SRRC_out <= X"518";
587             when 564  => SRRC_out <= X"649";
588             when 565  => SRRC_out <= X"66F";
589             when 566  => SRRC_out <= X"5D0";
590             when 567  => SRRC_out <= X"5F5";
591             when 568  => SRRC_out <= X"A49";
592             when 569  => SRRC_out <= X"A6F";
593             when 570  => SRRC_out <= X"9CF";
594             when 571  => SRRC_out <= X"9F5";
595             when 572  => SRRC_out <= X"B26";
596             when 573  => SRRC_out <= X"B4C";
597             when 574  => SRRC_out <= X"AAD";
598             when 575  => SRRC_out <= X"AD3";
599             when 576  => SRRC_out <= X"52D";
600             when 577  => SRRC_out <= X"553";
601             when 578  => SRRC_out <= X"4B4";
602             when 579  => SRRC_out <= X"4DA";
603             when 580  => SRRC_out <= X"60B";
604             when 581  => SRRC_out <= X"631";
605             when 582  => SRRC_out <= X"591";
606             when 583  => SRRC_out <= X"5B7";
607             when 584  => SRRC_out <= X"A0B";
608             when 585  => SRRC_out <= X"A30";
609             when 586  => SRRC_out <= X"991";
610             when 587  => SRRC_out <= X"9B7";
611             when 588  => SRRC_out <= X"AE8";
612             when 589  => SRRC_out <= X"B0E";
613             when 590  => SRRC_out <= X"A6F";
614             when 591  => SRRC_out <= X"A94";
615             when 592  => SRRC_out <= X"60B";
616             when 593  => SRRC_out <= X"631";
617             when 594  => SRRC_out <= X"591";
618             when 595  => SRRC_out <= X"5B7";
619             when 596  => SRRC_out <= X"6E9";
620             when 597  => SRRC_out <= X"70F";
```

```
621             when 598  => SRRC_out <= X"66F";
622             when 599  => SRRC_out <= X"695";
623             when 600  => SRRC_out <= X"AE8";
624             when 601  => SRRC_out <= X"B0E";
625             when 602  => SRRC_out <= X"A6F";
626             when 603  => SRRC_out <= X"A94";
627             when 604  => SRRC_out <= X"BC6";
628             when 605  => SRRC_out <= X"BEC";
629             when 606  => SRRC_out <= X"B4C";
630             when 607  => SRRC_out <= X"B72";
631             when 608  => SRRC_out <= X"4B4";
632             when 609  => SRRC_out <= X"4DA";
633             when 610  => SRRC_out <= X"43A";
634             when 611  => SRRC_out <= X"460";
635             when 612  => SRRC_out <= X"591";
636             when 613  => SRRC_out <= X"5B7";
637             when 614  => SRRC_out <= X"518";
638             when 615  => SRRC_out <= X"53E";
639             when 616  => SRRC_out <= X"991";
640             when 617  => SRRC_out <= X"9B7";
641             when 618  => SRRC_out <= X"917";
642             when 619  => SRRC_out <= X"93D";
643             when 620  => SRRC_out <= X"A6F";
644             when 621  => SRRC_out <= X"A94";
645             when 622  => SRRC_out <= X"9F5";
646             when 623  => SRRC_out <= X"A1B";
647             when 624  => SRRC_out <= X"591";
648             when 625  => SRRC_out <= X"5B7";
649             when 626  => SRRC_out <= X"518";
650             when 627  => SRRC_out <= X"53E";
651             when 628  => SRRC_out <= X"66F";
652             when 629  => SRRC_out <= X"695";
653             when 630  => SRRC_out <= X"5F5";
654             when 631  => SRRC_out <= X"61B";
655             when 632  => SRRC_out <= X"A6F";
656             when 633  => SRRC_out <= X"A94";
657             when 634  => SRRC_out <= X"9F5";
658             when 635  => SRRC_out <= X"A1B";
659             when 636  => SRRC_out <= X"B4C";
660             when 637  => SRRC_out <= X"B72";
661             when 638  => SRRC_out <= X"AD3";
662             when 639  => SRRC_out <= X"AF8";
663         -- somme per il FIR 5
664             when 640  => SRRC_out <= X"505";
665             when 641  => SRRC_out <= X"51B";
666             when 642  => SRRC_out <= X"4EF";
667             when 643  => SRRC_out <= X"505";
668             when 644  => SRRC_out <= X"429";
669             when 645  => SRRC_out <= X"43F";
670             when 646  => SRRC_out <= X"413";
671             when 647  => SRRC_out <= X"429";
672             when 648  => SRRC_out <= X"A8A";
673             when 649  => SRRC_out <= X"AA0";
674             when 650  => SRRC_out <= X"A74";
675             when 651  => SRRC_out <= X"A8A";
676             when 652  => SRRC_out <= X"9AE";
677             when 653  => SRRC_out <= X"9C4";
678             when 654  => SRRC_out <= X"998";
679             when 655  => SRRC_out <= X"9AE";
680             when 656  => SRRC_out <= X"6BB";
681             when 657  => SRRC_out <= X"6D2";
682             when 658  => SRRC_out <= X"6A5";
683             when 659  => SRRC_out <= X"6BC";
684             when 660  => SRRC_out <= X"5DF";
685             when 661  => SRRC_out <= X"5F6";
686             when 662  => SRRC_out <= X"5C9";
687             when 663  => SRRC_out <= X"5DF";
688             when 664  => SRRC_out <= X"C41";
689             when 665  => SRRC_out <= X"C57";
690             when 666  => SRRC_out <= X"C2A";
691             when 667  => SRRC_out <= X"C41";
692             when 668  => SRRC_out <= X"B64";
693             when 669  => SRRC_out <= X"B7B";
694             when 670  => SRRC_out <= X"B4E";
695             when 671  => SRRC_out <= X"B65";
696             when 672  => SRRC_out <= X"471";
697             when 673  => SRRC_out <= X"487";
698             when 674  => SRRC_out <= X"45B";
699             when 675  => SRRC_out <= X"471";
700             when 676  => SRRC_out <= X"395";
701             when 677  => SRRC_out <= X"3AB";
702             when 678  => SRRC_out <= X"37F";
703             when 679  => SRRC_out <= X"395";
704             when 680  => SRRC_out <= X"9F6";
705             when 681  => SRRC_out <= X"A0C";
706             when 682  => SRRC_out <= X"9E0";
707             when 683  => SRRC_out <= X"9F6";
708             when 684  => SRRC_out <= X"91A";
709             when 685  => SRRC_out <= X"930";
710             when 686  => SRRC_out <= X"904";
711             when 687  => SRRC_out <= X"91A";
712             when 688  => SRRC_out <= X"628";
713             when 689  => SRRC_out <= X"63E";
714             when 690  => SRRC_out <= X"611";
715             when 691  => SRRC_out <= X"628";
716             when 692  => SRRC_out <= X"54B";
717             when 693  => SRRC_out <= X"562";
718             when 694  => SRRC_out <= X"535";
```

```
719             when 695  => SRRC_out <= X"54C";
720             when 696  => SRRC_out <= X"BAD";
721             when 697  => SRRC_out <= X"BC3";
722             when 698  => SRRC_out <= X"B96";
723             when 699  => SRRC_out <= X"BAD";
724             when 700  => SRRC_out <= X"AD1";
725             when 701  => SRRC_out <= X"AE7";
726             when 702  => SRRC_out <= X"ABA";
727             when 703  => SRRC_out <= X"AD1";
728             when 704  => SRRC_out <= X"52F";
729             when 705  => SRRC_out <= X"546";
730             when 706  => SRRC_out <= X"519";
731             when 707  => SRRC_out <= X"52F";
732             when 708  => SRRC_out <= X"453";
733             when 709  => SRRC_out <= X"46A";
734             when 710  => SRRC_out <= X"43D";
735             when 711  => SRRC_out <= X"453";
736             when 712  => SRRC_out <= X"AB4";
737             when 713  => SRRC_out <= X"ACB";
738             when 714  => SRRC_out <= X"A9E";
739             when 715  => SRRC_out <= X"AB5";
740             when 716  => SRRC_out <= X"9D8";
741             when 717  => SRRC_out <= X"9EF";
742             when 718  => SRRC_out <= X"9C2";
743             when 719  => SRRC_out <= X"9D8";
744             when 720  => SRRC_out <= X"6E6";
745             when 721  => SRRC_out <= X"6FC";
746             when 722  => SRRC_out <= X"6D0";
747             when 723  => SRRC_out <= X"6E6";
748             when 724  => SRRC_out <= X"60A";
749             when 725  => SRRC_out <= X"620";
750             when 726  => SRRC_out <= X"5F4";
751             when 727  => SRRC_out <= X"60A";
752             when 728  => SRRC_out <= X"C6B";
753             when 729  => SRRC_out <= X"C81";
754             when 730  => SRRC_out <= X"C55";
755             when 731  => SRRC_out <= X"C6B";
756             when 732  => SRRC_out <= X"B8F";
757             when 733  => SRRC_out <= X"BA5";
758             when 734  => SRRC_out <= X"B79";
759             when 735  => SRRC_out <= X"B8F";
760             when 736  => SRRC_out <= X"49B";
761             when 737  => SRRC_out <= X"4B2";
762             when 738  => SRRC_out <= X"485";
763             when 739  => SRRC_out <= X"49C";
764             when 740  => SRRC_out <= X"3BF";
765             when 741  => SRRC_out <= X"3D6";
766             when 742  => SRRC_out <= X"3A9";
767             when 743  => SRRC_out <= X"3BF";
768             when 744  => SRRC_out <= X"A21";
769             when 745  => SRRC_out <= X"A37";
770             when 746  => SRRC_out <= X"A0A";
771             when 747  => SRRC_out <= X"A21";
772             when 748  => SRRC_out <= X"944";
773             when 749  => SRRC_out <= X"95B";
774             when 750  => SRRC_out <= X"92E";
775             when 751  => SRRC_out <= X"945";
776             when 752  => SRRC_out <= X"652";
777             when 753  => SRRC_out <= X"668";
778             when 754  => SRRC_out <= X"63C";
779             when 755  => SRRC_out <= X"652";
780             when 756  => SRRC_out <= X"576";
781             when 757  => SRRC_out <= X"58C";
782             when 758  => SRRC_out <= X"560";
783             when 759  => SRRC_out <= X"576";
784             when 760  => SRRC_out <= X"BD7";
785             when 761  => SRRC_out <= X"BED";
786             when 762  => SRRC_out <= X"BC1";
787             when 763  => SRRC_out <= X"BD7";
788             when 764  => SRRC_out <= X"AFB";
789             when 765  => SRRC_out <= X"B11";
790             when 766  => SRRC_out <= X"AE5";
791             when 767  => SRRC_out <= X"AFB";
792             when OTHERS => SRRC_out <= X"000";
793             end case;
794         end process;
795 end ROMx6_arch;
```

Listato F.3.9: shift_reg.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity shift_reg is
5         port (
6                clk       : in std_logic ;
7                clk_en    : in std_logic ;
8                reset     : in std_logic ;
9                in_reg    : in std_logic ;
10               out_ffd_1 : out std_logic ;
11               out_ffd_2 : out std_logic ;
12               out_ffd_3 : out std_logic ;
```

```
13              out_ffd_4 : out std_logic ;
14              out_ffd_5 : out std_logic ;
15              out_ffd_6 : out std_logic
16       );
17 end entity;
18
19 architecture shift_reg_arch of shift_reg is
20       signal temp_out_reg : std_logic_vector(5 downto 0);
21 begin
22       process(clk, clk_en, reset)
23       begin
24              if reset = '1' then
25                     temp_out_reg <= "000000";
26              elsif rising_edge(clk) then
27                     if clk_en = '1' then
28                            temp_out_reg <= in_reg & temp_out_reg(5 downto 1);
29                     end if;
30              end if;
31       end process;
32       out_ffd_6 <= temp_out_reg(0);
33       out_ffd_5 <= temp_out_reg(1);
34       out_ffd_4 <= temp_out_reg(2);
35       out_ffd_3 <= temp_out_reg(3);
36       out_ffd_2 <= temp_out_reg(4);
37       out_ffd_1 <= temp_out_reg(5);
38 end architecture;
```

Listato F.4.1: Modulator_BlockRAM.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity Modulator_BlockRAM is
5    port(
6        clk            : in STD_LOGIC;
7        reset   : in STD_LOGIC;
8        to_SRRC_I      : in STD_LOGIC;
9        to_SRRC_Q      : in STD_LOGIC;
10       rate_sel       : in STD_LOGIC_VECTOR(1 downto 0);
11       clk_2_analyzer : out STD_LOGIC;
12       QPSK_out       : out STD_LOGIC_VECTOR(11 downto 0)
13   );
14 end Modulator_BlockRAM;
15
16 architecture Modulator_BlockRAM of Modulator_BlockRAM is
17 component adder_I_Q
18   port (
19       QPSK_I  : in STD_LOGIC_VECTOR(11 downto 0);
20       QPSK_Q  : in STD_LOGIC_VECTOR(11 downto 0);
21       clk            : in STD_LOGIC;
22       reset   : in STD_LOGIC;
23       QPSK_out       : out STD_LOGIC_VECTOR(11 downto 0)
24   );
25 end component;
26 component counter
27   port (
28       clk            : in STD_LOGIC;
29       rate_sel       : in STD_LOGIC_VECTOR(1 downto 0);
30       reset   : in STD_LOGIC;
31       clk_en  : out STD_LOGIC;
32       count   : out STD_LOGIC_VECTOR(2 downto 0)
33   );
34 end component;
35 component multiplier_I_Q
36   port (
37       clk            : in STD_LOGIC;
38       from_NCO       : in STD_LOGIC_VECTOR(1 downto 0);
39       from_SRRC      : in STD_LOGIC_VECTOR(11 downto 0);
40       out_mult       : out STD_LOGIC_VECTOR(11 downto 0)
41   );
42 end component;
43 component NCO_basic
44   port (
45       clk            : in STD_LOGIC;
46       reset   : in STD_LOGIC;
47       cosine  : out STD_LOGIC_VECTOR(1 downto 0);
48       sine    : out STD_LOGIC_VECTOR(1 downto 0)
49   );
50 end component;
51 component srrc_x_n
52   port (
53       clk            : in STD_LOGIC;
54       clk_en  : in STD_LOGIC;
55       fir_sel  : in STD_LOGIC_VECTOR(2 downto 0);
56       in_fir_MSB : in STD_LOGIC;
57       rate_sel       : in STD_LOGIC_VECTOR(1 downto 0);
58       reset   : in STD_LOGIC;
59       out_srrc       : out STD_LOGIC_VECTOR(11 downto 0)
```

```
60    );
61  end component;
62  ---- Signal declarations used on the diagram ----
63  signal clk_en  : STD_LOGIC;
64  signal cosine  : STD_LOGIC_VECTOR (1 downto 0);
65  signal count_n  : STD_LOGIC_VECTOR (2 downto 0);
66  signal QPSK_I  : STD_LOGIC_VECTOR (11 downto 0);
67  signal QPSK_Q  : STD_LOGIC_VECTOR (11 downto 0);
68  signal sine   : STD_LOGIC_VECTOR (1 downto 0);
69  signal to_mult_I     : STD_LOGIC_VECTOR (11 downto 0);
70  signal to_mult_Q     : STD_LOGIC_VECTOR (11 downto 0);
71
72  begin
73  U1 : srrc_x_n
74    port map(
75         clk           => clk,
76         clk_en  => clk_en,
77         fir_sel  => count_n,
78         in_fir_MSB => to_SRRC_I,
79         out_srrc      => to_mult_I,
80         rate_sel      => rate_sel,
81         reset   => reset
82    );
83  U2 : srrc_x_n
84    port map(
85         clk           => clk,
86         clk_en  => clk_en,
87         fir_sel  => count_n,
88         in_fir_MSB => to_SRRC_Q,
89         out_srrc      => to_mult_Q,
90         rate_sel      => rate_sel,
91         reset   => reset
92    );
93  U3 : counter
94    port map(
95         clk           => clk,
96         clk_en  => clk_en,
97         count   => count_n,
98         rate_sel      => rate_sel,
99         reset   => reset
100   );
101 U4 : multiplier_I_Q
102   port map(
103        clk           => clk,
104        from_NCO      => sine,
105        from_SRRC     => to_mult_Q,
106        out_mult      => QPSK_Q
107   );
108 U5 : multiplier_I_Q
109   port map(
110        clk           => clk,
111        from_NCO      => cosine,
112        from_SRRC     => to_mult_I,
113        out_mult      => QPSK_I
114   );
115 U6 : NCO_basic
116   port map(
117        clk           => clk,
118        cosine  => cosine,
119        reset   => reset,
120        sine    => sine
121   );
122 U7 : adder_I_Q
123   port map(
124        QPSK_I  => QPSK_I,
125        QPSK_Q  => QPSK_Q,
126        QPSK_out      => QPSK_out,
127        clk           => clk,
128        reset   => reset
129   );
130       clk_2_analyzer <= clk;
131 end Modulator_BlockRAM;
```

Listato F.4.2: adder_I_Q.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_signed.all;
4
5  entity adder_I_Q is
6        port(QPSK_I, QPSK_Q  :      in std_logic_vector(11 downto 0);
7             clk , reset  :      in std_logic;
8             QPSK_out           :      out std_logic_vector(11 downto 0));
9  end adder_I_Q ;
10
11 architecture adder_I_Q_arch of adder_I_Q is
12       begin
13       process(reset, clk , QPSK_I , QPSK_Q)
14             begin
15             if falling_edge(clk) then
```

```
16                      if reset = '1' then
17                              QPSK_out <= "000000000000";
18                      else QPSK_out <= QPSK_I + QPSK_Q ;
19                      end if;
20              else null ;
21              end if ;
22      end process;
23  end adder_I_Q_arch ;
```

Listato F.4.3: counter.vhd

vedi listato(F.3.2)

Listato F.4.4: multiplier_I_Q.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_signed.all;
4
5   entity multiplier_I_Q is
6       port( from_SRRC :    in std_logic_vector(11 downto 0);
7             from_NCO  :  in std_logic_vector( 1 downto 0);
8             clk       :  in std_logic ;
9             out_mult     :     out std_logic_vector(11 downto 0));
10  end multiplier_I_Q;
11
12  architecture multiplier_I_Q_arch of multiplier_I_Q is
13  begin
14      process(clk, from_SRRC ,from_NCO)
15          variable temp : std_logic_vector(11 downto 0);
16      begin
17          if falling_edge(clk) then
18              temp(11 downto 0) := from_SRRC ;
19              case from_NCO is
20                  when "11"     => temp := ("111111111111" xor temp)+1 ;
21                  when "01"     => null ;
22                  when others => temp := "000000000000" ;
23              end case;
24              out_mult(11 downto 0) <= temp(11 downto 0) ;
25          else null;
26          end if;
27      end process;
28  end multiplier_I_Q_arch;
```

Listato F.4.5: NCO_basic.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_signed.all;
4
5   entity NCO_basic is
6       port( clk    :  in std_logic ;
7             reset  :  in std_logic ;
8             cosine :     out std_logic_vector(1 downto 0);
9             sine   :     out std_logic_vector(1 downto 0)
10             );
11  end NCO_basic;
12
13  architecture NCO_basic_arch of NCO_basic is
14      signal count_4 : STD_LOGIC_VECTOR (1 downto 0) ;
15  begin
16      process (clk, reset)
17      begin
18          if reset='1' then
19              count_4 <= "00";
20          else
21              if falling_edge(clk) then
22                  count_4 <= count_4 + 1 ;
23              else null ;
24              end if;
25          end if;
26      end process;
27
28      process (clk, count_4)
```

```
29          begin
30                  if falling_edge(clk) then
31                          case count_4 is
32                                  when "00" =>
33                                          cosine <= "01" ;
34                                          sine  <= "00" ;
35                                  when "01" =>
36                                          cosine <= "00" ;
37                                          sine  <= "11" ;
38                                  when "10" =>
39                                          cosine <= "11" ;
40                                          sine  <= "00" ;
41                                  when "11" =>
42                                          cosine <= "00" ;
43                                          sine  <= "01" ;
44                                  when others =>
45                                          cosine <= "XX" ;
46                                          sine  <= "XX" ;
47                          end case;
48                  else null;
49                  end if;
50          end process;
51  end NCO_basic_arch;
```

Listato F.4.6: srrc_x_n.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   entity srrc_x_n is
5     port(
6          clk       : in STD_LOGIC;
7          clk_en    : in STD_LOGIC;
8          in_fir_MSB : in STD_LOGIC;
9          reset     : in STD_LOGIC;
10         fir_sel   : in STD_LOGIC_VECTOR(2 downto 0);
11         rate_sel  : in STD_LOGIC_VECTOR(1 downto 0);
12         out_srrc  : out STD_LOGIC_VECTOR(11 downto 0)
13    );
14  end srrc_x_n;
15
16  architecture srrc_x_n of srrc_x_n is
17  component shift_reg
18    port (
19         clk             : in STD_LOGIC;
20         clk_en   : in STD_LOGIC;
21         in_reg   : in STD_LOGIC;
22         reset    : in STD_LOGIC;
23         out_ffd_1       : out STD_LOGIC;
24         out_ffd_2       : out STD_LOGIC;
25         out_ffd_3       : out STD_LOGIC;
26         out_ffd_4       : out STD_LOGIC;
27         out_ffd_5       : out STD_LOGIC;
28         out_ffd_6       : out STD_LOGIC
29    );
30  end component;
31  component ram
32    port (
33         addr     : in STD_LOGIC_VECTOR(11 downto 0);
34         clk              : in STD_LOGIC;
35         dout     : out STD_LOGIC_VECTOR(11 downto 0)
36    );
37  end component;
38  signal address   : STD_LOGIC_VECTOR (11 downto 0);
39
40  begin
41  U1 : shift_reg
42    port map(
43         clk             => clk,
44         clk_en   => clk_en,
45         in_reg   => address(0),
46         out_ffd_1       => address(1),
47         out_ffd_2       => address(2),
48         out_ffd_3       => address(3),
49         out_ffd_4       => address(4),
50         out_ffd_5       => address(5),
51         out_ffd_6       => address(6),
52         reset    => reset
53    );
54  U3 : ram
55    port map(
56         addr     => address,
57         clk              => clk,
58         dout     => out_srrc
59    );
60          address(7)    <= fir_sel(0);
61          address(8)    <= fir_sel(1);
62          address(9)    <= fir_sel(2);
63          address(0)    <= in_fir_MSB;
64          address(10) <= rate_sel(0);
65          address(11) <= rate_sel(1);
```

```
66  end srrc_x_n;
```

Listato F.4.7: ram.vhd

```
1   -- synopsys translate_off
2   LIBRARY ieee;
3   USE ieee.std_logic_1164.ALL;
4
5   Library XilinxCoreLib;
6   ENTITY ram IS
7         port (
8         addr   : IN std_logic_VECTOR(11 downto 0);
9         clk    : IN std_logic;
10        dout   : OUT std_logic_VECTOR(11 downto 0));
11  END ram;
12
13  ARCHITECTURE ram_a OF ram IS
14  component wrapped_ram
15        port (
16        addr   : IN std_logic_VECTOR(11 downto 0);
17        clk    : IN std_logic;
18        dout   : OUT std_logic_VECTOR(11 downto 0));
19  end component;
20  -- Configuration specification
21        for all : wrapped_ram
22         use entity XilinxCoreLib.blkmemsp_v3_1(behavioral)
23               generic map(
24                     c_reg_inputs  => 0,
25                     c_addr_width  => 12,
26                     c_has_sinit   => 0,
27                     c_has_rdy           => 0,
28                     c_width             => 12,
29                     c_has_en            => 0,
30                     c_mem_init_file     => "ram.mif",
31                     c_depth             => 4096,
32                     c_has_nd            => 0,
33                     c_has_default_data => 0,
34                     c_default_data  => "0",
35                     c_limit_data_pitch => 8,
36                     c_pipe_stages  => 0,
37                     c_has_rfd           => 0,
38                     c_has_we            => 0,
39                     c_sinit_value  => "0",
40                     c_has_limit_data_pitch => 0,
41                     c_enable_rlocs  => 0,
42                     c_has_din           => 0,
43                     c_write_mode  => 0);
44  BEGIN
45  U0 : wrapped_ram
46               port map (
47                     addr                => addr,
48                     clk                 => clk,
49                     dout                => dout);
50  END ram_a;
51  -- synopsys translate_on
```

Listato F.4.8: shift_reg.vhd

vedi listato(F.3.9)

Listato F.5.1: FIFO_RAM_ThinModulator.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   library unisim;
5   use unisim.vcomponents.all;
6
7   entity FIFO_RAM_ThinModulator is
8     port(
9         clk                  : in STD_LOGIC;
10        count_clk_en_reset : in STD_LOGIC;
```

```
11        data_in_I             : in STD_LOGIC;
12        data_in_Q             : in STD_LOGIC;
13        data_in_clk    : in std_ulogic;
14        reset          : in STD_LOGIC;
15        rate_sel              : in STD_LOGIC_VECTOR(1 downto 0);
16        data_in_clk_sync : out STD_LOGIC;
17        data_out_clk   : out STD_LOGIC;
18        modulator_out  : out STD_LOGIC_VECTOR(11 downto 0)
19    );
20  end FIFO_RAM_ThinModulator;
21
22  architecture FIFO_RAM_ThinModulator of FIFO_RAM_ThinModulator is
23  component counter
24    port (
25        clk                   : in STD_LOGIC;
26        rate_sel              : in STD_LOGIC_VECTOR(1 downto 0);
27        reset          : in STD_LOGIC;
28        clk_en         : out STD_LOGIC;
29        count          : out STD_LOGIC_VECTOR(2 downto 0)
30    );
31  end component;
32  component Data_Source_Interface
33    port (
34        I_to_fifo             : in STD_LOGIC;
35        Q_to_fifo             : in STD_LOGIC;
36        clk                   : in STD_LOGIC;
37        clk_en         : in STD_LOGIC;
38        data_in_clk    : in STD_LOGIC;
39        reset          : in STD_LOGIC;
40        I_from_fifo    : out STD_LOGIC;
41        Q_from_fifo    : out STD_LOGIC
42    );
43  end component;
44  component ThinModulator
45    port (
46        I_in_fir_MSB   : in STD_LOGIC;
47        Q_in_fir_MSB   : in STD_LOGIC;
48        clk                   : in STD_LOGIC;
49        clk_en         : in STD_LOGIC;
50        fir_sel        : in STD_LOGIC_VECTOR(2 downto 0);
51        rate_sel              : in STD_LOGIC_VECTOR(1 downto 0);
52        reset          : in STD_LOGIC;
53        QPSK_out              : out STD_LOGIC_VECTOR(11 downto 0)
54    );
55  end component;
56  component IBUFG
57    port (
58        I : in std_ulogic;
59        O : out std_ulogic
60    );
61  end component;
62  signal clk_en         : STD_LOGIC;
63  signal clk_to_fifo    : STD_LOGIC;
64  signal I_data_from_fifo : STD_LOGIC;
65  signal Q_data_from_fifo : STD_LOGIC;
66  signal fir_sel        : STD_LOGIC_VECTOR (2 downto 0);
67  for U4 : IBUFG use entity virtex.IBUFG;
68
69  begin
70  U1 : Data_Source_Interface
71    port map(
72        I_from_fifo    => I_data_from_fifo,
73        I_to_fifo             => data_in_I,
74        Q_from_fifo    => Q_data_from_fifo,
75        Q_to_fifo             => data_in_Q,
76        clk                   => clk,
77        clk_en         => clk_en,
78        data_in_clk    => clk_to_fifo,
79        reset          => reset
80    );
81  U2 : ThinModulator
82    port map(
83        I_in_fir_MSB   => I_data_from_fifo,
84        QPSK_out              => modulator_out,
85        Q_in_fir_MSB   => Q_data_from_fifo,
86        clk                   => clk,
87        clk_en         => clk_en,
88        fir_sel        => fir_sel,
89        rate_sel              => rate_sel,
90        reset          => reset
91    );
92  U3 : counter
93    port map(
94        clk                   => clk,
95        clk_en         => clk_en,
96        count          => fir_sel,
97        rate_sel              => rate_sel,
98        reset          => count_clk_en_reset
99    );
100 U4 : IBUFG
101   port map(
102       I                     => data_in_clk,
103       O                     => clk_to_fifo
104   );
105       data_in_clk_sync    <= clk_en;
```

```
106         data_out_clk  <= clk;
107 end FIFO_RAM_ThinModulator;
```

Listato F.5.2: counter.vhd

vedi listato(F.3.2)

Listato F.5.3: Data_Source_Interface.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity Data_Source_Interface is
5    port(
6         I_to_fifo             : in STD_LOGIC;
7         Q_to_fifo             : in STD_LOGIC;
8         clk                   : in STD_LOGIC;
9         clk_en          : in STD_LOGIC;
10        data_in_clk     : in STD_LOGIC;
11        reset           : in STD_LOGIC;
12        I_from_fifo     : out STD_LOGIC;
13        Q_from_fifo     : out STD_LOGIC
14   );
15 end Data_Source_Interface;
16
17 architecture Data_Source_Interface of Data_Source_Interface is
18 component ffd
19   port (
20        clk                   : in STD_LOGIC;
21        in_ffd          : in STD_LOGIC;
22        reset           : in STD_LOGIC;
23        out_ffd         : out STD_LOGIC
24   );
25 end component;
26 component ffd_en
27   port (
28        clk                   : in STD_LOGIC;
29        clk_en          : in STD_LOGIC;
30        in_ffd          : in STD_LOGIC;
31        reset           : in STD_LOGIC;
32        out_ffd         : out STD_LOGIC
33   );
34 end component;
35 component ffs
36   port (
37        S                     : in std_ulogic;
38        clk                   : in std_ulogic;
39        reset           : in std_ulogic;
40        Q                     : out std_ulogic
41   );
42 end component;
43 component asynch_fifo_2x15
44   port (
45        ainit           : in STD_LOGIC;
46        din                   : in STD_LOGIC_VECTOR(1 downto 0);
47        rd_clk          : in STD_LOGIC;
48        rd_en           : in STD_LOGIC;
49        wr_clk          : in STD_LOGIC;
50        wr_en           : in STD_LOGIC;
51        almost_empty    : out STD_LOGIC;
52        dout            : out STD_LOGIC_VECTOR(1 downto 0);
53        empty           : out STD_LOGIC;
54        full            : out STD_LOGIC
55   );
56 end component;
57 constant VCC_CONSTANT       : STD_LOGIC := '1';
58 signal data_in_clk_to_fifo  : STD_LOGIC;
59 signal enable_reading       : STD_LOGIC;
60 signal fifo_out_enable      : STD_LOGIC;
61 signal out_enable           : STD_LOGIC;
62 signal read_enable          : STD_LOGIC;
63 signal start_rd_en          : STD_LOGIC;
64 signal VCC                        : STD_LOGIC;
65 signal data_from_fifo       : STD_LOGIC_VECTOR (1 downto 0);
66 signal data_to_fifo         : STD_LOGIC_VECTOR (1 downto 0);
67
68 begin
69 U1 : asynch_fifo_2x15
```

```
70    port map(
71         ainit                => reset,
72         almost_empty         => enable_reading,
73         din                       => data_to_fifo,
74         dout                 => data_from_fifo,
75         rd_clk               => clk,
76         rd_en                => read_enable,
77         wr_clk               => data_in_clk_to_fifo,
78         wr_en                => VCC
79    );
80  U11 : ffs
81    port map(
82         Q                         => start_rd_en,
83         S                         => enable_reading,
84         clk                       => clk,
85         reset                => reset
86    );
87  U14 : ffd_en
88    port map(
89         clk                       => clk,
90         clk_en               => clk_en,
91         in_ffd               => start_rd_en,
92         out_ffd              => out_enable,
93         reset                => reset
94    );
95  U15 : ffd
96    port map(
97         clk                       => clk,
98         in_ffd               => data_from_fifo(1),
99         out_ffd              => Q_from_fifo,
100        reset                => fifo_out_enable
101   );
102 U2 : ffd
103   port map(
104        clk                       => clk,
105        in_ffd               => data_from_fifo(0),
106        out_ffd              => I_from_fifo,
107        reset                => fifo_out_enable
108   );
109 fifo_out_enable                 <= not(out_enable);
110 read_enable                <= start_rd_en and clk_en;
111 VCC                             <= VCC_CONSTANT;
112        data_to_fifo(0)          <= I_to_fifo;
113        data_to_fifo(1)          <= Q_to_fifo;
114        data_in_clk_to_fifo  <= data_in_clk;
115 end Data_Source_Interface;
```

Listato F.5.4: asynch_fifo_2x15.vhd

```
1   -- synopsys translate_off
2   LIBRARY ieee;
3   USE ieee.std_logic_1164.ALL;
4   Library XilinxCoreLib;
5   ENTITY asynch_fifo_2x15 IS
6         port (
7         din           : IN std_logic_VECTOR(1 downto 0);
8         wr_en         : IN std_logic;
9         wr_clk : IN std_logic;
10        rd_en         : IN std_logic;
11        rd_clk : IN std_logic;
12        ainit         : IN std_logic;
13        dout          : OUT std_logic_VECTOR(1 downto 0);
14        full          : OUT std_logic;
15        empty         : OUT std_logic;
16        almost_empty: OUT std_logic);
17  END asynch_fifo_2x15;
18  ARCHITECTURE asynch_fifo_2x15_a OF asynch_fifo_2x15 IS
19  component wrapped_asynch_fifo_2x15
20        port (
21        din           : IN std_logic_VECTOR(1 downto 0);
22        wr_en         : IN std_logic;
23        wr_clk : IN std_logic;
24        rd_en         : IN std_logic;
25        rd_clk : IN std_logic;
26        ainit         : IN std_logic;
27        dout          : OUT std_logic_VECTOR(1 downto 0);
28        full          : OUT std_logic;
29        empty         : OUT std_logic;
30        almost_empty: OUT std_logic);
31  end component;
32  -- Configuration specification
33        for all : wrapped_asynch_fifo_2x15
34         use entity XilinxCoreLib.async_fifo_v4_0(behavioral)
35               generic map(
36                     c_wr_count_width    => 2,
37                     c_has_rd_err  => 0,
38                     c_data_width  => 2,
39                     c_has_almost_full => 0,
40                     c_rd_err_low  => 0,
41                     c_has_wr_ack  => 0,
```

```
                    c_wr_ack_low  => 0,
                    c_fifo_depth  => 15,
                    c_rd_count_width    => 2,
                    c_has_wr_err  => 0,
                    c_has_almost_empty => 1,
                    c_rd_ack_low  => 0,
                    c_has_wr_count  => 0,
                    c_use_blockmem  => 1,
                    c_has_rd_ack  => 0,
                    c_has_rd_count  => 0,
                    c_wr_err_low  => 0,
                    c_enable_rlocs  => 0);
BEGIN
U0 : wrapped_asynch_fifo_2x15
             port map (
                    din                 => din,
                    wr_en         => wr_en,
                    wr_clk        => wr_clk,
                    rd_en         => rd_en,
                    rd_clk        => rd_clk,
                    ainit         => ainit,
                    dout                => dout,
                    full                => full,
                    empty         => empty,
                    almost_empty  => almost_empty);
END asynch_fifo_2x15_a;
-- synopsys translate_on
```

Listato F.5.5: ffd.vhd

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

entity ffd is
       port(in_ffd          :      in std_logic ;
              clk           :      in std_logic ;
              reset :      in std_logic ;
              out_ffd       :      out std_logic );
end ffd;

architecture ffd_arch of ffd is
begin
       process (clk, reset)
       begin
             if reset='1' then
             out_ffd <= '0';
             elsif rising_edge(clk) then
             out_ffd <= in_ffd;
             end if;
       end process;
end ffd_arch ;
```

Listato F.5.6: ffd_en.vhd

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

entity ffd_en is
       port(in_ffd          :      in std_logic ;
              clk           :      in std_logic ;
              clk_en        :      in std_logic ;
              reset :      in std_logic ;
              out_ffd       :      out std_logic );
end ffd_en;

architecture ffd_en_arch of ffd_en is
begin
       process (clk, clk_en, reset)
       begin
             if reset='1' then
             out_ffd <= '0';
             elsif rising_edge(clk) then
                    if clk_en = '1' then
                    out_ffd <= in_ffd;
                    end if;
             end if;
       end process;
end ffd_en_arch ;
```

Listato F.5.7: ffs.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  entity ffs is
4    port (S, clk, reset : in std_ulogic;
5          Q : out std_ulogic);
6  end entity ffs;
7
8  architecture ffs_arch of ffs is
9    signal state : std_ulogic;
10 begin
11       process (clk, reset) is
12     begin
13       if (reset = '1') then
14         state <= '0';
15       elsif rising_edge(clk) then
16         case S is
17               when '0'  => state <= '1';
18               when others => null;
19         end case;
20          end if;
21     end process ;
22       Q <= state;
23 end ffs_arch;
```

Listato F.5.8: ThinModulator.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity ThinModulator is
5    port(
6         I_in_fir_MSB : in STD_LOGIC;
7         Q_in_fir_MSB : in STD_LOGIC;
8         clk       : in STD_LOGIC;
9         clk_en    : in STD_LOGIC;
10        reset     : in STD_LOGIC;
11        fir_sel   : in STD_LOGIC_VECTOR(2 downto 0);
12        rate_sel          : in STD_LOGIC_VECTOR(1 downto 0);
13        QPSK_out          : out STD_LOGIC_VECTOR(11 downto 0)
14   );
15 end ThinModulator;
16
17 architecture ThinModulator of ThinModulator is
18 component mult_C2_adder
19   port (
20        clk               : in STD_LOGIC;
21        in_mca    : in STD_LOGIC_VECTOR(11 downto 0);
22        reset     : in STD_LOGIC;
23        out_mca   : out STD_LOGIC_VECTOR(11 downto 0)
24   );
25 end component;
26 component mux_2x7
27   port (
28        clk       : in STD_LOGIC;
29        in_I      : in STD_LOGIC_VECTOR(6 downto 0);
30        in_Q      : in STD_LOGIC_VECTOR(6 downto 0);
31        reset     : in STD_LOGIC;
32        out_mux   : out STD_LOGIC_VECTOR(6 downto 0)
33   );
34 end component;
35 component shift_reg
36   port (
37        clk               : in STD_LOGIC;
38        clk_en            : in STD_LOGIC;
39        in_reg            : in STD_LOGIC;
40        reset             : in STD_LOGIC;
41        out_ffd_1         : out STD_LOGIC;
42        out_ffd_2         : out STD_LOGIC;
43        out_ffd_3         : out STD_LOGIC;
44        out_ffd_4         : out STD_LOGIC;
45        out_ffd_5         : out STD_LOGIC;
46        out_ffd_6         : out STD_LOGIC
47   );
48 end component;
49 component ram_12x4096_rising_registered
50   port (
51        addr      : in STD_LOGIC_VECTOR(11 downto 0);
52        clk       : in STD_LOGIC;
53        dout      : out STD_LOGIC_VECTOR(11 downto 0)
54   );
55 end component;
56 signal address         : STD_LOGIC_VECTOR (11 downto 0);
57 signal I_address : STD_LOGIC_VECTOR (6 downto 0);
58 signal out_ram         : STD_LOGIC_VECTOR (11 downto 0);
59 signal Q_address       : STD_LOGIC_VECTOR (6 downto 0);
60
61 begin
```

```
62  U1 : shift_reg
63    port map(
64         clk             => clk,
65         clk_en   => clk_en,
66         in_reg   => I_address(0),
67         out_ffd_1 => I_address(1),
68         out_ffd_2 => I_address(2),
69         out_ffd_3 => I_address(3),
70         out_ffd_4 => I_address(4),
71         out_ffd_5 => I_address(5),
72         out_ffd_6       => I_address(6),
73         reset    => reset
74    );
75  U2 : shift_reg
76    port map(
77         clk             => clk,
78         clk_en   => clk_en,
79         in_reg   => Q_address(0),
80         out_ffd_1       => Q_address(1),
81         out_ffd_2       => Q_address(2),
82         out_ffd_3       => Q_address(3),
83         out_ffd_4       => Q_address(4),
84         out_ffd_5       => Q_address(5),
85         out_ffd_6       => Q_address(6),
86         reset    => reset
87    );
88  U3 : ram_12x4096_rising_registered
89    port map(
90         addr     => address,
91         clk             => clk,
92         dout     => out_ram
93    );
94  U4 : mux_2x7
95    port map(
96         out_mux(0) => address(0),
97         out_mux(1) => address(1),
98         out_mux(2) => address(2),
99         out_mux(3) => address(3),
100        out_mux(4) => address(4),
101        out_mux(5) => address(5),
102        out_mux(6) => address(6),
103        clk              => clk,
104        in_I     => I_address,
105        in_Q     => Q_address,
106        reset    => reset
107   );
108 U6 : mult_C2_adder
109   port map(
110        clk             => clk,
111        in_mca   => out_ram,
112        out_mca  => QPSK_out,
113        reset    => reset
114   );
115        I_address(0) <= I_in_fir_MSB;
116        Q_address(0) <= Q_in_fir_MSB;
117        address(7)     <= fir_sel(0);
118        address(8)     <= fir_sel(1);
119        address(9)     <= fir_sel(2);
120        address(10) <= rate_sel(0);
121        address(11) <= rate_sel(1);
122 end ThinModulator;
```

Listato F.5.9: mult_C2_adder.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_signed.all;
4
5  entity mult_C2_adder is
6         port(in_mca  :       in std_logic_vector(11 downto 0);
7                reset   :  in std_logic ;
8                clk     :  in std_logic ;
9                out_mca :     out std_logic_vector(11 downto 0));
10 end mult_C2_adder;
11
12 architecture mult_C2_adder_arch of mult_C2_adder is
13        signal TEMP_count : std_logic_vector(1 downto 0);
14 begin
15        process(clk, reset)
16        begin
17               if reset = '1' then
18                      TEMP_count <= "11";
19               elsif rising_edge(clk) then
20                      TEMP_count <= TEMP_count + 1;
21               end if;
22        end process;
23        process(clk, reset, TEMP_count(1), in_mca)
```

```
24          begin
25                  if reset = '1' then
26                          out_mca <= x"000" ;
27                  elsif rising_edge(clk) then
28                          out_mca <= in_mca ;
29                          if TEMP_count(1) = '1' then
30                                  out_mca <= ( x"FFF" xor in_mca) + 1 ;
31                          end if;
32                  end if;
33          end process;
34  end mult_C2_adder_arch ;
```

Listato F.5.10: mux_2x7.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity mux_2x7 is
5           port(in_I, in_Q :    in std_logic_vector(6 downto 0);
6                   reset    : in std_logic ;
7                   clk      : in std_logic ;
8                   out_mux  :    out std_logic_vector(6 downto 0));
9   end mux_2x7;
10
11  architecture mux_2x7_arch of mux_2x7 is
12          signal sel : std_logic;
13  begin
14          -- processo che genera il segnale di selezione alternato
15          process (clk, reset)
16          begin
17                  if reset = '1' then
18                          sel <= '1';
19                  elsif rising_edge(clk) then
20                          sel <= not sel;
21                  end if;
22          end process;
23          out_mux <= in_I when (sel = '0') else in_Q ;
24  end mux_2x7_arch ;
```

Listato F.5.11: ram_12x4096_rising_registered.vhd

```
1   -- synopsys translate_off
2   LIBRARY ieee;
3   USE ieee.std_logic_1164.ALL;
4
5   Library XilinxCoreLib;
6   ENTITY ram_12x4096_rising_registered IS
7           port (
8           addr    : IN std_logic_VECTOR(11 downto 0);
9           clk     : IN std_logic;
10          dout    : OUT std_logic_VECTOR(11 downto 0));
11  END ram_12x4096_rising_registered;
12
13  ARCHITECTURE ram_12x4096_rising_registered_a OF ram_12x4096_rising_registered IS
14  component wrapped_ram_12x4096_rising_registered
15          port (
16          addr: IN std_logic_VECTOR(11 downto 0);
17          clk: IN std_logic;
18          dout: OUT std_logic_VECTOR(11 downto 0));
19  end component;
20          for all : wrapped_ram_12x4096_rising_registered
21                  use entity XilinxCoreLib.blkmemsp_v4_0(behavioral)
22                  generic map(
23                          c_reg_inputs  => 1,
24                          c_addr_width  => 12,
25                          c_has_sinit   => 0,
26                          c_ysinit_is_high     => 1,
27                          c_has_rdy            => 0,
28                          c_width              => 12,
29                          c_has_en             => 0,
30                          c_ymake_bmm   => 0,
31                          c_yen_is_high  => 1,
32                          c_yprimitive_type => "4kx1",
33                          c_yhierarchy  => "hierarchy1",
34                          c_mem_init_file      => "ram_12x4096_rising_registered.mif",
35                          c_ywe_is_high  => 1,
36                          c_yuse_single_primitive => 1,
37                          c_depth              => 4096,
38                          c_has_nd             => 0,
39                          c_has_default_data => 0,
40                          c_default_data   => "0",
41                          c_ytop_addr   => "1024",
42                          c_limit_data_pitch => 8,
43                          c_pipe_stages  => 1,
```

```
44                      c_ybottom_addr  => "0",
45                      c_has_rfd           => 0,
46                      c_yclk_is_rising    => 1,
47                      c_has_we            => 0,
48                      c_sinit_value  => "0",
49                      c_has_limit_data_pitch => 0,
50                      c_enable_rlocs  => 0,
51                      c_has_din           => 0,
52                      c_write_mode  => 0);
53  BEGIN
54  U0 : wrapped_ram_12x4096_rising_registered
55                  port map (
56                      addr                => addr,
57                      clk                 => clk,
58                      dout                => dout);
59  END ram_12x4096_rising_registered_a;
60  -- synopsys translate_on
```

Listato F.5.12: shift_reg.vhd

vedi listato(F.3.9)