```
1   ----------------------------------------------------------------------------
2   -- Hardware AMBA bus IP for image scaling
3   ----------------------------------------------------------------------------
4   -- Entity:      TFG_RESIZE_EMBB
5   -- File:        TFG_RESIZE_EMBB.vhd
6   -- Author:      Luis Miguel Gonzalez Berrocal
7   ----------------------------------------------------------------------------
8   -- VHDL Standard: VHDL'93
9   ----------------------------------------------------------------------------
10  -- Naming Conventions
11  -- active low signals:                  "*_n"
12  -- clock signals:                       "clk", "clk_div#", "clk_#x"
13  -- reset signals:                       "rst", "rst_n"
14  -- generics:                            "C_*"
15  -- user defined types:                  "*_TYPE"
16  -- state machine next state:            "*_ns"
17  -- state machine current state:         "*_cs"
18  -- combinatorial signals:               "*_com"
19  -- pipelined or register delay signals: "*_d#"
20  -- counter signals:                     "*_cnt*"
21  -- clock enable signals:                "*_ce"
22  -- internal version of output port:     "*_i"
23  -- device pins:                         "*_pin"
24  -- ports:                               "Names begin with Uppercase"
25  -- processes:                           "*_PROCESS"
26  -- component instantiations:            "<ENTITY_>I_<#|FUNC>"
27  -- registers:                           "*_REG"
28  ----------------------------------------------------------------------------
29
30  library ieee;
31  use ieee.std_logic_1164.all;
32  use ieee.std_logic_unsigned.all;
33  use ieee.std_logic_arith.all;
34
35  entity TFG_RESIZE_EMBB is
36        generic (SIZE                      : integer := 11;
37                 IMSE_EXTRA_PRECISION_BITS : integer := 4;
38                 INTER_RESIZE_COEF_BITS    : integer := 11);
39        port (Clk           : in std_logic;                              -- clk
40              Rst_n         : in std_logic;                              -- active low
    reset
41              D             : in std_logic_vector (SIZE - 1 downto 0);  -- index of
    row/column
42              Shifted_scale : in std_logic_vector (31 downto 0);        --
    IMSE_RESIZE_COEF_SCALE * Src_size / Dst_size
43              Embedd_ofs    : out std_logic_vector (SIZE - 1 downto 0); -- pixel
    position using
44              Coef_even     : out std_logic_vector (15 downto 0);       --
    interpolation factor for data even
45              Coef_odd      : out std_logic_vector (15 downto 0));      --
    interpolation factor for data odd
46  end TFG_RESIZE_EMBB;
47
48  architecture Behavioral of TFG_RESIZE_EMBB is
49
50        constant UPPER : std_logic_vector (31 - INTER_RESIZE_COEF_BITS -
    IMSE_EXTRA_PRECISION_BITS - 1 downto 0) := (others => '0');
```

```
51          constant LOWER : std_logic_vector (INTER_RESIZE_COEF_BITS +
    IMSE_EXTRA_PRECISION_BITS - 1 downto 0) := (others => '0');
52          constant IMSE_RESIZE_COEF_SCALE : std_logic_vector (31 downto 0) :=  UPPER & '1'
     & LOWER;
53          signal au_com, embbed_com, embedd_ofs_i, embedd_cbuf_com : std_logic_vector (31
    downto 0);
54          signal coef_even_i, coef_odd_i : std_logic_vector (15 downto 0) := (others =>
    '0');
55          signal au2_com : std_logic_vector (31 + SIZE + 1 downto 0);
56          signal d_com : std_logic_vector (SIZE downto 0);
57
58          function saturate_short_from_int_FUNCTION (a : in std_logic_vector (31 downto 0
    )) return std_logic_vector is
59
60                variable val_com : std_logic_vector (15 downto 0);
61
62          begin
63                if (a > X"00007FFF") then
64                      val_com := X"7FFF";
65                else
66                      val_com := a (15 downto 0);
67                end if;
68                return val_com;
69          end function;
70
71    begin
72
73          combinatorial_PROCESS : process (Shifted_scale, au2_com, au_com, embbed_com,
    embedd_cbuf_com, d_com, D)
74          begin
75                d_com <= D & '0' + 1;
76                au2_com <= Shifted_scale * d_com - IMSE_RESIZE_COEF_SCALE;
77                au_com <= '0' & au2_com (31 downto 1);
78                embbed_com <= X"0000" & '0' & au_com (14 downto 0);
79                embedd_ofs_i <= X"000" & "000" & au_com (31 downto INTER_RESIZE_COEF_BITS
    + IMSE_EXTRA_PRECISION_BITS);
80                embedd_cbuf_com <= IMSE_RESIZE_COEF_SCALE - embbed_com;
81                coef_even_i <= saturate_short_from_int_FUNCTION ("0000" & embedd_cbuf_com
    (31 downto IMSE_EXTRA_PRECISION_BITS));
82                coef_odd_i <= saturate_short_from_int_FUNCTION ("0000" & embbed_com (31
    downto IMSE_EXTRA_PRECISION_BITS));
83          end process;
84
85          sequential_PROCESS : process (Clk, Rst_n)
86          begin
87                if (Rst_n = '0') then
88                      Embedd_ofs <= (others => '0');
89                      Coef_even <= (others => '0');
90                      Coef_odd <= (others => '0');
91                elsif (Clk = '1' and clk'event) then
92                      if (au2_com (31 + SIZE + 1) = '1') then               -- upscaling
    (version 2.0)
93                            Embedd_ofs <= (others => '0');
94                            Coef_even <= X"0800";
95                            Coef_odd <= (others => '0');
96                      else
97                            Embedd_ofs <= embedd_ofs_i (SIZE - 1 downto 0); -- downscaling
```

```
 98                          Coef_even <= coef_even_i;
 99                          Coef_odd <= coef_odd_i;
100                      end if;
101                  end if;
102          end process;
103
104   end Behavioral;
105
106
```