library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use ieee.std\_logic\_unsigned.all;

entity part\_2 is

generic(SAM\_SIZE:integer:=8;

OUT\_SIZE:integer:=16;

FAC\_SIZE:integer:=8);

Port ( p\_0, p\_1:in std\_logic\_vector(SAM\_SIZE-1 downto 0) ;

alpha:in std\_logic\_vector(FAC\_SIZE-1 downto 0);

p\_f:out std\_logic\_vector(OUT\_SIZE-1 downto 0);

clk:in std\_logic

);

end part\_2;

architecture Behavioral of part\_2 is

--CLK WIZARD

component clk\_wiz\_0

port

(-- Clock in ports

-- Clock out ports

clk\_out1 : out std\_logic;

clk\_out2 :out std\_logic;

-- Status and control signals

locked : out std\_logic;

clk\_in1 : in std\_logic

);

end component;

signal s0, s1 :std\_logic\_vector(SAM\_SIZE-1 downto 0);

signal blend:std\_logic\_vector(OUT\_SIZE-1 downto 0):="0000000000000000";

signal locked,clk1x,clk2x:std\_logic;

begin

u1:clk\_wiz\_0 port map(clk\_out1=>clk1x, clk\_out2=>clk2x, locked=>locked, clk\_in1=>clk);

process(clk1x, blend)

begin

if rising\_edge(clk1x) then

s0 <=p\_0;

s1<=p\_1;

end if;

end process;

process(clk2x)

begin

if clk2x'event and clk2x='1' then

blend<=s0\*alpha+0;

else

blend<=blend+(1-alpha)\*s1;

-- blend<="0000000000000011";

end if;

end process;

process(clk1x, blend)

begin

if rising\_edge(clk1x) then

p\_f<=blend;

end if;

end process;

end Behavioral;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;library IEEE;

use ieee.numeric\_std.all;use IEEE.STD\_LOGIC\_1164.ALL;

use ieee.std\_logic\_unsigned.all;

entity tb\_part\_1 is

end tb\_part\_1;

architecture Behavioral of tb\_part\_1 is

component part\_1

Port ( p\_0, p\_1:in std\_logic\_vector(7 downto 0) ;

alpha:in std\_logic\_vector(7 downto 0);

p\_f:out std\_logic\_vector(15 downto 0));

end component;

component part\_2

Port ( p\_0, p\_1:in std\_logic\_vector(7 downto 0) ;

alpha:in std\_logic\_vector(7 downto 0);

p\_f:out std\_logic\_vector(15 downto 0);

clk:in std\_logic);

end component;

constant clock\_period:time:=20 ns;

signal clk:std\_logic;

signal p\_0, p\_1:std\_logic\_vector(7 downto 0);

signal p\_f :std\_logic\_vector(15 downto 0);

signal alpha:std\_logic\_vector(7 downto 0);

begin

clock:process

begin

clk<='0';

wait for clock\_period/2;

clk<= not clk;

wait for clock\_period/2;

end process;

sine\_wave\_inputs :process

begin

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(0,8));

p\_1 <= std\_logic\_vector(to\_signed(0,8));

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(-126,8));

p\_1 <= std\_logic\_vector(to\_signed(-252,8));

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(32,8));

p\_1 <= std\_logic\_vector(to\_signed(64,8));

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(118,8));

p\_1 <= std\_logic\_vector(to\_signed(235,8));

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(-62,8));

p\_1 <= std\_logic\_vector(to\_signed(-125,8));

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(-102,8));

p\_1 <= std\_logic\_vector(to\_signed(-203,8));

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(88,8));

p\_1 <= std\_logic\_vector(to\_signed(177,8));

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(79,8));

p\_1 <= std\_logic\_vector(to\_signed(158,8));

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(-108,8));

p\_1 <= std\_logic\_vector(to\_signed(-217,8));

wait for 12.5 ns;

p\_0 <= std\_logic\_vector(to\_signed(-51,8));

p\_1 <= std\_logic\_vector(to\_signed(-102,8));

end process;

alpha\_input:process

begin

alpha <= std\_logic\_vector(to\_signed(0,8));

wait for 500 ns;

alpha <= std\_logic\_vector(to\_signed(1,8));

wait for 500 ns;

end process;

--u1:part\_1 port map(p\_0=>p\_0, p\_1=>p\_1, alpha=>alpha, p\_f=>p\_f);

u2:part\_2 port map(p\_0=>p\_0, p\_1=>p\_1, alpha=>alpha, p\_f=>p\_f, clk=>clk);

end Behavioral;