library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity stream\_interleaver is

port (clk : in std\_logic;

p1, p2 : in std\_logic\_vector(7 downto 0);

pf : out std\_logic\_vector(16 downto 0));

end stream\_interleaver;

architecture Behavioral of stream\_interleaver is

--Enable DSPs

attribute use\_dsp: string;

attribute use\_dsp of pf: signal is "yes";

--Signal Declarations

signal temp1, temp2 : std\_logic\_vector(15 downto 0); --Part 1 variables

signal alpha : signed (7 downto 0) := "00000000"; --Part 2 variables

signal clk\_1x, clk\_2x, locked, clk\_in1 : std\_logic;

signal p1\_reg, p2\_reg, alpha\_reg : std\_logic\_vector(7 downto 0);

signal blend : std\_logic\_vector(16 downto 0) := "00000000000000000";

signal mux\_a, mux\_b : std\_logic\_vector(7 downto 0);

--signal s0, s1 : std\_logic\_vector(7 downto 0);

--Instantiate Clock Wizard

component clock\_manager

port(clk\_in1 : in std\_logic;

clk\_out1 : out std\_logic; --50 Mhz

clk\_out2 : out std\_logic; --100 Mhz

locked : out std\_logic);

end component;

begin

--

UUT: clock\_manager

port map(clk\_out1 => clk\_1x, clk\_out2 => clk\_2x, locked => locked, clk\_in1 => clk);

----|Part 1: Direct implementation|----

--[pf <= alpha \* p1 + (1 - alpha) \* p2]--

-- temp1 <= std\_logic\_vector(signed(p1) \* alpha);

-- temp2 <= std\_logic\_vector(signed(p2) \* (127 - alpha));

-- pf <= std\_logic\_vector(signed(temp1(15) & temp1) + signed(temp2(15) & temp2));

----|Part 2: Optimized Implementation|----

--Step 1: Register Design

step1: process(clk\_1x)

begin

if(rising\_edge(clk\_1x)) then

p1\_reg <= p1;

p2\_reg <= p2;

alpha\_reg <= std\_logic\_vector(alpha);

end if;

end process step1;

--Step 2 : MUX Design for CLK\_1X

mux\_a <=

p1\_reg when clk\_1x = '0' else

p2\_reg when clk\_1x = '1' else

mux\_a;

mux\_b <=

std\_logic\_vector(alpha) when clk\_1x = '0' else

std\_logic\_vector(127 - signed(alpha)) when clk\_1x = '1' else

mux\_a;

--Step 3: MUX Design for CLK\_2X

step3: process(clk\_2x)

begin

if(clk\_2x'event and clk\_2x = '1') then

--blend <= std\_logic\_vector(signed(p1\_reg) \* alpha + 0);

blend <= std\_logic\_vector(signed(p1\_reg) \* alpha + 0); --select half of blend (P1 with 0)

else

--blend <= std\_logic\_vector(signed(blend) + (127 - alpha) \* signed(p2\_reg));

blend <= std\_logic\_vector(signed(blend) + (127 - alpha) \* signed(p2\_reg)); --select blend

end if;

end process step3;

--Step 4: Output Design of CLK\_1X

step4: process(clk\_1x)

begin

if(rising\_edge(clk\_1x)) then

pf <= blend;

--else?

end if;

end process step4;

end Behavioral;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity tb\_signal is

-- Port ( );

end tb\_signal;

architecture Behavioral of tb\_signal is

--Instantiate component for testing

--component stream\_interleaver is

-- port (p1, p2 : in std\_logic\_vector(7 downto 0);

-- pf : out std\_logic\_vector(16 downto 0));

--end component;

component stream\_interleaver is

port (clk : in std\_logic;

p1, p2 : in std\_logic\_vector(7 downto 0);

pf : out std\_logic\_vector(16 downto 0));

end component;

signal p1\_reg, p2\_reg, alpha: std\_logic\_vector(7 downto 0);

signal pf\_reg : std\_logic\_vector(16 downto 0);

signal clk : std\_logic := '0';

constant clk\_pulse : time := 20 ns;

begin

UUT: stream\_interleaver

port map(p1 => p1\_reg, p2 => p2\_reg, pf => pf\_reg, clk => clk);

clock: process(clk)

begin

clk <= not clk after clk\_pulse / 2;

end process clock;

--8 Mhz normal amplitude p0

interleave: process

begin

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(0,8));

p2\_reg <= std\_logic\_vector(to\_signed(0,8));

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(75,8));

p2\_reg <= std\_logic\_vector(to\_signed(37,8));

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(121,8));

p2\_reg <= std\_logic\_vector(to\_signed(60,8));

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(121,8));

p2\_reg <= std\_logic\_vector(to\_signed(60,8));

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(75,8));

p2\_reg <= std\_logic\_vector(to\_signed(37,8));

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(0,8));

p2\_reg <= std\_logic\_vector(to\_signed(0,8));

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(-121,8));

p2\_reg <= std\_logic\_vector(to\_signed(-60,8));

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(-121,8));

p2\_reg <= std\_logic\_vector(to\_signed(-60,8));

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(-74,8));

p2\_reg <= std\_logic\_vector(to\_signed(-37,8));

wait for clk\_pulse;

p1\_reg <= std\_logic\_vector(to\_signed(0,8));

p2\_reg <= std\_logic\_vector(to\_signed(0,8));

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 alpha\_input;

end Behavioral;