library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

--use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity stream\_interleaver\_pt3 is

Generic ( alpha : signed(7 downto 0) := "01111111");

Port ( p0 : in STD\_LOGIC\_VECTOR (7 downto 0);

p1 : in STD\_LOGIC\_VECTOR (7 downto 0);

clk : in STD\_LOGIC;

pf : out STD\_LOGIC\_VECTOR (16 downto 0));

end stream\_interleaver\_pt3;

architecture Behavioral of stream\_interleaver\_pt3 is

--Stage 1

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

signal clk\_temp, clken: std\_logic;

signal SI, SO: std\_logic\_vector(7 downto 0);

signal clk\_temp2, we: std\_logic;

signal a: std\_logic\_vector(5 downto 0);

signal di, do : std\_logic\_vector(15 downto 0);

--Stage 2

signal s0\_reg, s1\_reg : signed(7 downto 0);

signal alpha\_reg, alphaN\_reg : signed(7 downto 0);

signal mux\_A, mux\_B : signed(7 downto 0);

--Stage 3

signal QA, QB: signed(7 downto 0);

signal prod : signed(15 downto 0);

--Stage 4

signal mux\_out, blend : signed(16 downto 0);

--Instantiate Clock Wizard

component clk\_wiz\_0

port ( clk\_in1: in std\_logic;

clk\_out1 : out std\_logic;

clk\_out2: out std\_logic);

end component;

--SRL

component shift\_registers\_0

port ( clk\_temp : in std\_logic;

clken : in std\_logic;

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

SO : out std\_logic\_vector(7 downto 0));

end component;

--BRAM

component ram\_dist

port(

clk\_temp2 : in std\_logic;

we : in std\_logic;

a : in std\_logic\_vector(5 downto 0);

di : in std\_logic\_vector(15 downto 0);

do : out std\_logic\_vector(15 downto 0)

);

end component;

begin

UUT : clk\_wiz\_0

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

UUT2: shift\_registers\_0

port map (clk\_temp => clk\_temp, clken => clken, SI => SI, SO => SO);

UUT3: ram\_dist

port map (clk\_temp2 => clk\_temp, we => we, a => a, di => di, do => do);

--Store into regesters and select

data\_in: process(clk\_1x)

begin

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

s0\_reg <= signed(p0);

s1\_reg <= signed(p1);

alpha\_reg <= alpha;

alphaN\_reg <= 127 - alpha;

end if;

end process data\_in;

--MUX selection to load for product

mux\_A <= s0\_reg when clk\_1x = '1' else s1\_reg;

mux\_B <= alpha\_reg when clk\_1x = '1' else alphaN\_reg;

--Multiplication process

product: process(clk\_2x)

begin

if(rising\_edge(clk\_2x)) then

QA <= mux\_A;

QB <= mux\_B;

prod <= QA \* QB;

end if;

end process product;

--MUX selection to load for acummluate

mux\_out <= (others => '0') when clk\_2x = '0' else blend;

--Addition process

add: process(clk\_2x)

begin

if(clk\_2x = '1') then

blend <= mux\_out + (prod(15) & prod);

end if;

end process add;

--Addition process

data\_out: process(clk\_1x)

begin

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

pf <= std\_logic\_vector(blend);

end if;

end process data\_out;

end Behavioral;

-- Single-Port RAM with Asynchronous Read (Distributed RAM)

-- File: rams\_dist.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity rams\_dist is

port(

clk\_temp2 : in std\_logic;

we : in std\_logic;

a : in std\_logic\_vector(5 downto 0);

di : in std\_logic\_vector(15 downto 0);

do : out std\_logic\_vector(15 downto 0)

);

end rams\_dist;

architecture syn of rams\_dist is

type ram\_type is array (63 downto 0) of std\_logic\_vector(15 downto 0);

signal RAM : ram\_type;

begin

process(clk\_temp2)

begin

if (clk\_temp2'event and clk\_temp2 = '1') then

if (we = '1') then

RAM(conv\_integer(a)) <= di;

end if;

end if;

end process;

do <= RAM(conv\_integer(a));

end syn;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity tb\_stream\_interleaver\_pt2 is

-- Port ( );

end tb\_stream\_interleaver\_pt2;

architecture Behavioral of tb\_stream\_interleaver\_pt2 is

component stream\_interleaver\_pt2 is

Generic(alpha : signed(7 downto 0) := "01111111");

Port ( p0 : in STD\_LOGIC\_VECTOR (7 downto 0);

p1 : in STD\_LOGIC\_VECTOR (7 downto 0);

clk : in STD\_LOGIC;

pf : out STD\_LOGIC\_VECTOR (16 downto 0));

end component;

signal p0\_reg, p1\_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\_pt2

port map(p0 => p0\_reg, p1 => p1\_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;

p0\_reg <= std\_logic\_vector(to\_signed(0,8));

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

wait for clk\_pulse;

p0\_reg <= std\_logic\_vector(to\_signed(75,8));

p1\_reg <= std\_logic\_vector(to\_signed(37,8));

wait for clk\_pulse;

p0\_reg <= std\_logic\_vector(to\_signed(121,8));

p1\_reg <= std\_logic\_vector(to\_signed(60,8));

wait for clk\_pulse;

p0\_reg <= std\_logic\_vector(to\_signed(121,8));

p1\_reg <= std\_logic\_vector(to\_signed(60,8));

wait for clk\_pulse;

p0\_reg <= std\_logic\_vector(to\_signed(75,8));

p1\_reg <= std\_logic\_vector(to\_signed(37,8));

wait for clk\_pulse;

p0\_reg <= std\_logic\_vector(to\_signed(0,8));

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

wait for clk\_pulse;

p0\_reg <= std\_logic\_vector(to\_signed(-121,8));

p1\_reg <= std\_logic\_vector(to\_signed(-60,8));

wait for clk\_pulse;

p0\_reg <= std\_logic\_vector(to\_signed(-121,8));

p1\_reg <= std\_logic\_vector(to\_signed(-60,8));

wait for clk\_pulse;

p0\_reg <= std\_logic\_vector(to\_signed(-74,8));

p1\_reg <= std\_logic\_vector(to\_signed(-37,8));

wait for clk\_pulse;

p0\_reg <= std\_logic\_vector(to\_signed(0,8));

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

end process;

alpha\_input: process

begin

alpha <= std\_logic\_vector(to\_signed(1,8)); wait for 1000 ns;

alpha <= std\_logic\_vector(to\_signed(0,8)); wait for 1000 ns;

end process alpha\_input;

end Behavioral;