**High Pass Filter**

**MATLAB Code:**

function [x\_out, y\_out, r\_out, g\_out, b\_out] = hdlhpf(x\_in, y\_in, r\_in, g\_in, b\_in)

persistent OrigImg x1 x2 y1 y2

% Low Pass Filter coefficients

D = [1/6 1/6 1/6;

1/6 1/6 1/6;

1/6 1/6 1/6];

% Original RGB input

RGB = [r\_in; g\_in; b\_in];

% Low Pass Filter operation

OrigImg\_1 = D \* RGB;

% High Pass Filter operation

HPF\_RGB = RGB - OrigImg\_1;

% Output the high-pass filtered result

r\_out = HPF\_RGB(1);

g\_out = HPF\_RGB(2);

b\_out = HPF\_RGB(3);

% Pass through the coordinates

x\_out = x2;

x2 = x1;

x1 = x\_in;

y\_out = y2;

y2 = y1;

y1 = y\_in;

end

**TEST BENCH:**

WIDTH = 800;

HEIGHT = 450;

rout=zeros(450,800);

ImgData = double(imread("C:\Users\Public\lotus1.jpg"));

for y = 0:HEIGHT

for x = 0:WIDTH

if y >= 0 && y < HEIGHT && x >= 0 && x < WIDTH

b = ImgData(y+1,x+1,1);

g = ImgData(y+1,x+1,2);

r = ImgData(y+1,x+1,3);

else

b = 0;

g = 0;

r = 0;

end

[xOut, yOut,rout,gout,bout] =hdlhpf(x, y, r, g, b);

Iout(y+1,x+1,1)=bout;

Iout(y+1,x+1,2)=gout;

Iout(y+1,x+1,3)=rout;

end

end

figure(1)

imshow(uint8(ImgData));

figure(2)

imshow(uint8(Iout),[]);

**VHDL Code:**

-- -------------------------------------------------------------

--

-- File Name: C:\Users\Public\codegen\hdlhpf\hdlsrc\hdlhpf\_fixpt.vhd

-- Created: 2024-05-19 14:20:48

--

-- Generated by MATLAB 9.12, MATLAB Coder 5.4 and HDL Coder 3.20

--

--

--

-- -------------------------------------------------------------

-- Rate and Clocking Details

-- -------------------------------------------------------------

-- Design base rate: 1

--

--

-- Clock Enable Sample Time

-- -------------------------------------------------------------

-- ce\_out 1

-- -------------------------------------------------------------

--

--

-- Output Signal Clock Enable Sample Time

-- -------------------------------------------------------------

-- x\_out ce\_out 1

-- y\_out ce\_out 1

-- r\_out ce\_out 1

-- g\_out ce\_out 1

-- b\_out ce\_out 1

-- -------------------------------------------------------------

--

-- -------------------------------------------------------------

-- -------------------------------------------------------------

--

-- Module: hdlhpf\_fixpt

-- Source Path: hdlhpf\_fixpt

-- Hierarchy Level: 0

--

-- -------------------------------------------------------------

LIBRARY IEEE;

USE IEEE.std\_logic\_1164.ALL;

USE IEEE.numeric\_std.ALL;

USE work.hdlhpf\_fixpt\_pkg.ALL;

ENTITY hdlhpf\_fixpt IS

PORT( clk : IN std\_logic;

reset : IN std\_logic;

clk\_enable : IN std\_logic;

x\_in : IN std\_logic\_vector(9 DOWNTO 0);

y\_in : IN std\_logic\_vector(8 DOWNTO 0);

r\_in : IN std\_logic\_vector(7 DOWNTO 0);

g\_in : IN std\_logic\_vector(7 DOWNTO 0);

b\_in : IN std\_logic\_vector(7 DOWNTO 0);

ce\_out : OUT std\_logic;

x\_out : OUT std\_logic\_vector(9 DOWNTO 0);

y\_out : OUT std\_logic\_vector(8 DOWNTO 0);

r\_out : OUT std\_logic\_vector(8 DOWNTO 0);

g\_out : OUT std\_logic\_vector(8 DOWNTO 0);

b\_out : OUT std\_logic\_vector(8 DOWNTO 0)

);

END hdlhpf\_fixpt;

ARCHITECTURE rtl OF hdlhpf\_fixpt IS

-- Signals

SIGNAL enb : std\_logic;

SIGNAL x\_in\_unsigned : unsigned(9 DOWNTO 0);

SIGNAL x2\_reg\_reg : vector\_of\_unsigned10(0 TO 1);

SIGNAL x2 : unsigned(9 DOWNTO 0);

SIGNAL y\_in\_unsigned : unsigned(8 DOWNTO 0);

SIGNAL y2\_reg\_reg : vector\_of\_unsigned9(0 TO 1);

SIGNAL y2 : unsigned(8 DOWNTO 0);

SIGNAL r\_in\_unsigned : unsigned(7 DOWNTO 0);

SIGNAL g\_in\_unsigned : unsigned(7 DOWNTO 0);

SIGNAL b\_in\_unsigned : unsigned(7 DOWNTO 0);

SIGNAL RGB : vector\_of\_unsigned8(0 TO 2);

SIGNAL c : vector\_of\_unsigned11(0 TO 2);

SIGNAL c\_1 : vector\_of\_unsigned11(0 TO 2);

SIGNAL OrigImg\_1 : vector\_of\_unsigned7(0 TO 2);

SIGNAL HPF\_RGB : vector\_of\_signed9(0 TO 2);

SIGNAL p10HPF\_RGB\_sub\_cast : vector\_of\_signed11(0 TO 2);

SIGNAL p10HPF\_RGB\_sub\_cast\_1 : vector\_of\_signed11(0 TO 2);

SIGNAL p10HPF\_RGB\_sub\_temp : vector\_of\_signed11(0 TO 2);

SIGNAL r\_out\_tmp : signed(8 DOWNTO 0);

SIGNAL g\_out\_tmp : signed(8 DOWNTO 0);

SIGNAL b\_out\_tmp : signed(8 DOWNTO 0);

BEGIN

x\_in\_unsigned <= unsigned(x\_in);

enb <= clk\_enable;

-- Pass through the coordinates

x2\_reg\_process : PROCESS (clk, reset)

BEGIN

IF reset = '1' THEN

x2\_reg\_reg <= (OTHERS => to\_unsigned(16#000#, 10));

ELSIF clk'EVENT AND clk = '1' THEN

IF enb = '1' THEN

x2\_reg\_reg(0) <= x\_in\_unsigned;

x2\_reg\_reg(1) <= x2\_reg\_reg(0);

END IF;

END IF;

END PROCESS x2\_reg\_process;

x2 <= x2\_reg\_reg(1);

x\_out <= std\_logic\_vector(x2);

y\_in\_unsigned <= unsigned(y\_in);

y2\_reg\_process : PROCESS (clk, reset)

BEGIN

IF reset = '1' THEN

y2\_reg\_reg <= (OTHERS => to\_unsigned(16#000#, 9));

ELSIF clk'EVENT AND clk = '1' THEN

IF enb = '1' THEN

y2\_reg\_reg(0) <= y\_in\_unsigned;

y2\_reg\_reg(1) <= y2\_reg\_reg(0);

END IF;

END IF;

END PROCESS y2\_reg\_process;

y2 <= y2\_reg\_reg(1);

y\_out <= std\_logic\_vector(y2);

r\_in\_unsigned <= unsigned(r\_in);

g\_in\_unsigned <= unsigned(g\_in);

b\_in\_unsigned <= unsigned(b\_in);

-- Original RGB input

--

-- Low Pass Filter coefficients

--

-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

--

-- %

--

-- Generated by MATLAB 9.12 and Fixed-Point Designer 7.4 %

--

-- %

--

-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

RGB(0) <= r\_in\_unsigned;

RGB(1) <= g\_in\_unsigned;

RGB(2) <= b\_in\_unsigned;

-- Low Pass Filter operation

c <= (OTHERS => to\_unsigned(16#000#, 11));

-- HDL code generation from MATLAB function: hdlhpf\_fixpt

p4\_output : PROCESS

VARIABLE c1 : vector\_of\_unsigned11(0 TO 2);

BEGIN

FOR l IN 0 TO 2 LOOP

FOR k IN 0 TO 2 LOOP

c1(l) := to\_unsigned(16#000#, 11);

END LOOP;

c\_1(l) <= c1(l);

END LOOP;

WAIT;

END PROCESS p4\_output;

OrigImg\_1\_gen: FOR t\_0 IN 0 TO 2 GENERATE

OrigImg\_1(t\_0) <= c\_1(t\_0)(6 DOWNTO 0);

END GENERATE OrigImg\_1\_gen;

-- High Pass Filter operation

HPF\_RGB\_gen: FOR t\_01 IN 0 TO 2 GENERATE

p10HPF\_RGB\_sub\_cast(t\_01) <= signed(resize(RGB(t\_01), 11));

p10HPF\_RGB\_sub\_cast\_1(t\_01) <= signed(resize(OrigImg\_1(t\_01), 11));

p10HPF\_RGB\_sub\_temp(t\_01) <= p10HPF\_RGB\_sub\_cast(t\_01) - p10HPF\_RGB\_sub\_cast\_1(t\_01);

HPF\_RGB(t\_01) <= p10HPF\_RGB\_sub\_temp(t\_01)(8 DOWNTO 0);

END GENERATE HPF\_RGB\_gen;

-- Output the high-pass filtered result

r\_out\_tmp <= HPF\_RGB(0);

r\_out <= std\_logic\_vector(r\_out\_tmp);

g\_out\_tmp <= HPF\_RGB(1);

g\_out <= std\_logic\_vector(g\_out\_tmp);

b\_out\_tmp <= HPF\_RGB(2);

b\_out <= std\_logic\_vector(b\_out\_tmp);

ce\_out <= clk\_enable;

END rtl;

**Verilog Code:**  
// -------------------------------------------------------------

//

// File Name: C:\Users\Public\codegen\hdlhpf\hdlsrc\hdlhpf\_fixpt.v

// Created: 2024-05-19 14:22:57

//

// Generated by MATLAB 9.12, MATLAB Coder 5.4 and HDL Coder 3.20

//

//

//

// -- -------------------------------------------------------------

// -- Rate and Clocking Details

// -- -------------------------------------------------------------

// Design base rate: 1

//

//

// Clock Enable Sample Time

// -- -------------------------------------------------------------

// ce\_out 1

// -- -------------------------------------------------------------

//

//

// Output Signal Clock Enable Sample Time

// -- -------------------------------------------------------------

// x\_out ce\_out 1

// y\_out ce\_out 1

// r\_out ce\_out 1

// g\_out ce\_out 1

// b\_out ce\_out 1

// -- -------------------------------------------------------------

//

// -------------------------------------------------------------

// -------------------------------------------------------------

//

// Module: hdlhpf\_fixpt

// Source Path: hdlhpf\_fixpt

// Hierarchy Level: 0

//

// -------------------------------------------------------------

`timescale 1 ns / 1 ns

module hdlhpf\_fixpt

(clk,

reset,

clk\_enable,

x\_in,

y\_in,

r\_in,

g\_in,

b\_in,

ce\_out,

x\_out,

y\_out,

r\_out,

g\_out,

b\_out);

input clk;

input reset;

input clk\_enable;

input [9:0] x\_in;

input [8:0] y\_in;

input [7:0] r\_in;

input [7:0] g\_in;

input [7:0] b\_in;

output ce\_out;

output [9:0] x\_out;

output [8:0] y\_out;

output signed [8:0] r\_out;

output signed [8:0] g\_out;

output signed [8:0] b\_out;

wire enb;

reg [9:0] x2\_reg\_reg [0:1];

wire [9:0] x2\_reg\_reg\_next [0:1];

wire [9:0] x2;

reg [8:0] y2\_reg\_reg [0:1];

wire [8:0] y2\_reg\_reg\_next [0:1];

wire [8:0] y2;

wire [7:0] RGB [0:2];

wire [10:0] c [0:2];

reg [10:0] c\_1 [0:2];

wire [10:0] c\_2 [0:2];

wire [6:0] OrigImg\_1 [0:2];

wire signed [8:0] HPF\_RGB [0:2];

wire signed [10:0] p10HPF\_RGB\_sub\_temp [0:2];

wire signed [10:0] p10HPF\_RGB\_1 [0:2];

wire signed [10:0] p10HPF\_RGB\_2 [0:2];

reg signed [31:0] p4\_l;

reg [10:0] p4\_c [0:2];

reg signed [31:0] p4\_k;

assign enb = clk\_enable;

// Pass through the coordinates

always @(posedge clk or posedge reset)

begin : x2\_reg\_process

if (reset == 1'b1) begin

x2\_reg\_reg[0] <= 10'b0000000000;

x2\_reg\_reg[1] <= 10'b0000000000;

end

else begin

if (enb) begin

x2\_reg\_reg[0] <= x2\_reg\_reg\_next[0];

x2\_reg\_reg[1] <= x2\_reg\_reg\_next[1];

end

end

end

assign x2 = x2\_reg\_reg[1];

assign x2\_reg\_reg\_next[0] = x\_in;

assign x2\_reg\_reg\_next[1] = x2\_reg\_reg[0];

always @(posedge clk or posedge reset)

begin : y2\_reg\_process

if (reset == 1'b1) begin

y2\_reg\_reg[0] <= 9'b000000000;

y2\_reg\_reg[1] <= 9'b000000000;

end

else begin

if (enb) begin

y2\_reg\_reg[0] <= y2\_reg\_reg\_next[0];

y2\_reg\_reg[1] <= y2\_reg\_reg\_next[1];

end

end

end

assign y2 = y2\_reg\_reg[1];

assign y2\_reg\_reg\_next[0] = y\_in;

assign y2\_reg\_reg\_next[1] = y2\_reg\_reg[0];

// Original RGB input

//

// Low Pass Filter coefficients

//

// %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//

// %

//

// Generated by MATLAB 9.12 and Fixed-Point Designer 7.4 %

//

// %

//

// %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

assign RGB[0] = r\_in;

assign RGB[1] = g\_in;

assign RGB[2] = b\_in;

// Low Pass Filter operation

assign c[0] = 11'b00000000000;

assign c[1] = 11'b00000000000;

assign c[2] = 11'b00000000000;

// HDL code generation from MATLAB function: hdlhpf\_fixpt

initial begin

for(p4\_l = 32'sd0; p4\_l <= 32'sd2; p4\_l = p4\_l + 32'sd1) begin

for(p4\_k = 32'sd0; p4\_k <= 32'sd2; p4\_k = p4\_k + 32'sd1) begin

p4\_c[p4\_l] = 11'b00000000000;

end

c\_1[p4\_l] = p4\_c[p4\_l];

end

end

assign c\_2[0] = c\_1[0];

assign c\_2[1] = c\_1[1];

assign c\_2[2] = c\_1[2];

assign OrigImg\_1[0] = c\_2[0][6:0];

assign OrigImg\_1[1] = c\_2[1][6:0];

assign OrigImg\_1[2] = c\_2[2][6:0];

// High Pass Filter operation

assign p10HPF\_RGB\_1[0] = {3'b0, RGB[0]};

assign p10HPF\_RGB\_2[0] = {4'b0, OrigImg\_1[0]};

assign p10HPF\_RGB\_sub\_temp[0] = p10HPF\_RGB\_1[0] - p10HPF\_RGB\_2[0];

assign HPF\_RGB[0] = p10HPF\_RGB\_sub\_temp[0][8:0];

assign p10HPF\_RGB\_1[1] = {3'b0, RGB[1]};

assign p10HPF\_RGB\_2[1] = {4'b0, OrigImg\_1[1]};

assign p10HPF\_RGB\_sub\_temp[1] = p10HPF\_RGB\_1[1] - p10HPF\_RGB\_2[1];

assign HPF\_RGB[1] = p10HPF\_RGB\_sub\_temp[1][8:0];

assign p10HPF\_RGB\_1[2] = {3'b0, RGB[2]};

assign p10HPF\_RGB\_2[2] = {4'b0, OrigImg\_1[2]};

assign p10HPF\_RGB\_sub\_temp[2] = p10HPF\_RGB\_1[2] - p10HPF\_RGB\_2[2];

assign HPF\_RGB[2] = p10HPF\_RGB\_sub\_temp[2][8:0];

// Output the high-pass filtered result

assign r\_out = HPF\_RGB[0];

assign g\_out = HPF\_RGB[1];

assign b\_out = HPF\_RGB[2];

assign ce\_out = clk\_enable;

assign x\_out = x2;

assign y\_out = y2;

endmodule // hdlhpf\_fixpt

**Package:**

LIBRARY IEEE;

USE IEEE.std\_logic\_1164.ALL;

USE IEEE.numeric\_std.ALL;

PACKAGE hdlhpf\_fixpt\_pkg IS

TYPE vector\_of\_unsigned10 IS ARRAY (NATURAL RANGE <>) OF unsigned(9 DOWNTO 0);

TYPE vector\_of\_unsigned9 IS ARRAY (NATURAL RANGE <>) OF unsigned(8 DOWNTO 0);

TYPE vector\_of\_unsigned8 IS ARRAY (NATURAL RANGE <>) OF unsigned(7 DOWNTO 0);

TYPE vector\_of\_unsigned11 IS ARRAY (NATURAL RANGE <>) OF unsigned(10 DOWNTO 0);

TYPE vector\_of\_unsigned7 IS ARRAY (NATURAL RANGE <>) OF unsigned(6 DOWNTO 0);

TYPE vector\_of\_signed9 IS ARRAY (NATURAL RANGE <>) OF signed(8 DOWNTO 0);

TYPE vector\_of\_signed11 IS ARRAY (NATURAL RANGE <>) OF signed(10 DOWNTO 0);

END hdlhpf\_fixpt\_pkg;

**Input Image:**



**Output Image:**

