library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

use IEEE.MATH\_REAL.ALL;

use ieee.numeric\_std.all;

entity sort\_1 is

port( rst\_n : in std\_logic:='0';

Mat0 : in std\_logic\_vector(7 downto 0);

Mat1 : in std\_logic\_vector(7 downto 0);

Mat2 : in std\_logic\_vector(7 downto 0);

out\_pixel : out std\_logic\_vector(7 downto 0):=(others=>'0');

temp1 : out std\_logic\_vector(7 downto 0));

end sort\_1;

architecture behavioural of sort\_1 is

type matrix is array (0 to 2) of std\_logic\_vector (7 downto 0);

signal mat\_temp : matrix := (others => "00000000");

begin

mat\_temp(0) <= Mat0;

mat\_temp(1) <= Mat1;

mat\_temp(2) <= Mat2;

process(rst\_n, mat\_temp)

variable temp\_mat : matrix := (others => "00000000");

variable temp : std\_logic\_vector(7 downto 0):=(others =>'0');

begin

case rst\_n is

when '0' =>

temp\_mat := (others =>"00000000");

temp := (others =>'0');

out\_pixel <= "01000010";

out\_pixel <= (others =>'1');

when '1' =>

temp\_mat := mat\_temp;

loop1:for i in 0 to 2 loop

loop2: for j in 0 to 7 loop

if(temp\_mat(j+1) > temp\_mat(j)) then

temp := temp\_mat(j);

temp\_mat(j) := temp\_mat(j+1);

temp\_mat(j+1) := temp;

else

NULL;

end if;

end loop loop2;

end loop loop1;

out\_pixel <= temp\_mat(1);

--outpixel gives median value

when others => null;

end case;

temp1 <=temp;

end process;

end behavioural;