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

--

-- FileName: hw\_image\_generator.vhd

-- Dependencies: none

-- Design Software: Quartus II 64-bit Version 12.1 Build 177 SJ Full Version

--

-- HDL CODE IS PROVIDED "AS IS." DIGI-KEY EXPRESSLY DISCLAIMS ANY

-- WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT

-- LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A

-- PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL DIGI-KEY

-- BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL

-- DAMAGES, LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF

-- PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS

-- BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF),

-- ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.

--

-- Version History

-- Version 1.0 05/10/2013 Scott Larson

-- Initial Public Release

--

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

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.math\_real.all;

ENTITY hw\_image\_generator IS

-- GENERIC(

-- Y\_TOP : INTEGER := 478;

-- X\_TOP : INTEGER := 600;

-- Y\_BOTTOM : INTEGER := 578;

-- X\_BOTTOM : INTEGER := 800);

PORT(

ps2\_code : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

clk1 : IN STD\_LOGIC;

CLK : IN STD\_LOGIC;

disp\_ena : IN STD\_LOGIC; --display enable ('1' = display time, '0' = blanking time)

row : IN INTEGER; --row pixel coordinate

column : IN INTEGER; --column pixel coordinate

red : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) := (OTHERS => '0'); --red magnitude output to DAC

green : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) := (OTHERS => '0'); --green magnitude output to DAC

blue : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) := (OTHERS => '0')); --blue magnitude output to DAC

END hw\_image\_generator;

ARCHITECTURE behavioral OF hw\_image\_generator IS

--RANGE 0 TO 26

--RANGE 0 TO 47

SIGNAL COUNT : INTEGER RANGE 0 TO 50000000 := 0;

SIGNAL XPOS :INTEGER := 520;

SIGNAL YPOS : INTEGER := 920;

SIGNAL DIRECTION : INTEGER RANGE 0 TO 3 := 0;

SIGNAL XFOOD : INTEGER RANGE 0 TO 47 := 5;

SIGNAL YFOOD : INTEGER RANGE 0 TO 26 := 5;

SIGNAL EAT : STD\_LOGIC := '0';

SIGNAL LENG : INTEGER := 1;

type xSnake is array (0 to 19) of integer;

type ySnake is array (0 to 19) of integer;

signal xBody : xSnake;

signal yBody : ySnake;

BEGIN

PROCESS(CLK)

VARIABLE foodx : integer := xFOOD\*40;

VARIABLE foody : integer := yFOOD\*40;

BEGIN

IF(CLK'EVENT AND CLK = '1')THEN

COUNT <= COUNT + 1;

IF(COUNT = 15000000)THEN

COUNT <= 0;

--XPOS <= XPOS + 100;

if(direction = 0) then

-- points(xSnake(0),ySnake(0)) <= '0';

-- points(xSnake(0)+1,ySnake(0)) <= '1';

-- xSnake(0) <= xSnake(0) + 1;

if(ypos=1920-40)then

ypos<=0;

else

yPOS<= yPOS + 40;

end if;

elsif(direction = 1) then

-- points(xSnake(0),ySnake(0)) <= '0';

-- points(xSnake(0)-1,ySnake(0)) <= '1';

-- xSnake(0) <= xSnake(0) - 1;

if(ypos=0)then

ypos<=1920-40;

else

yPOS<= yPOS - 40;

end if;

elsif(direction = 2) then

-- points(xSnake(0),ySnake(0)) <= '0';

-- points(xSnake(0),ySnake(0)+1) <= '1';

-- ySnake(0) <= ySnake(0) + 1;

if(xpos=0)then

xpos<=1080-40;

else

xPOS<= xPOS - 40;

end if;

elsif(direction = 3) then

-- points(xSnake(0),ySnake(0)) <= '0';

-- points(xSnake(0),ySnake(0)-1) <= '1';

-- ySnake(0) <= ySnake(0) - 1;

if(xpos=1080-40)then

xpos<=0;

else

xPOS<= xPOS + 40;

end if;

end if;

-- if(xpos=foodx and ypos=foody)then

-- eat <= '1';

-- end if;

END IF;

if(xpos/40 = xfood and ypos/40 = yfood)then

LENG <= LENG + 1;

xFOOD <= (xFood+5)\*7;

yFOOD <= (yFood+5)\*7;

if(xFood > 47)then

xfood <= xfood/10;

end if;

if(yfood>26)theN

yfood <= yfood/10;

end if;

end if;

END IF;

END PROCESS;

process(ps2\_code) --changing direction

begin

if(ps2\_code = "00100011" and direction /= 1) then --right

direction <= 0;

elsif(ps2\_code = "00011100" and direction /= 0) then --left

direction <= 1;

elsif(ps2\_code = "00011101" and direction /= 3) then --up

direction <= 2;

elsif(ps2\_code = "00011011" and direction /= 2) then --down

direction <= 3;

end if;

end process;

PROCESS(disp\_ena, row, column, xpos, ypos)

VARIABLE foodx : integer := xFOOD\*40;

VARIABLE foody : integer := yFOOD\*40;

BEGIN

IF(disp\_ena = '1') THEN --display time

IF(row > yPOS AND row < yPOS+40 and column > xPOS AND column < xPOS+40) THEN

red <= (OTHERS => '0');

green <= (OTHERS => '0');

blue <= (OTHERS => '1');

ELSE

red <= (OTHERS => '0');

green <= (OTHERS => '0');

blue <= (OTHERS => '0');

END IF;

ELSE --blanking time

red <= (OTHERS => '0');

green <= (OTHERS => '0');

blue <= (OTHERS => '0');

END IF;

IF(row > foodx AND row < foodx+40 and column > foody AND column < foody+40) THEN

red <= (OTHERS => '1');

green <= (OTHERS => '1');

blue <= (OTHERS => '1');

END IF;

END PROCESS;

-- PROCESS(eat)

-- BEGIN

-- if(eat = '1')then

-- xFOOD <= (xFood+5)\*7;

-- yFOOD <= (yFood+5)\*7;

-- if(xFood > 47)then

-- xfood <= xfood/10;

-- end if;

-- if(yfood>26)theN

-- yfood <= yfood/10;

-- end if;

-- --eat <= '0';

-- end if;

-- END PROCESS;

END behavioral;