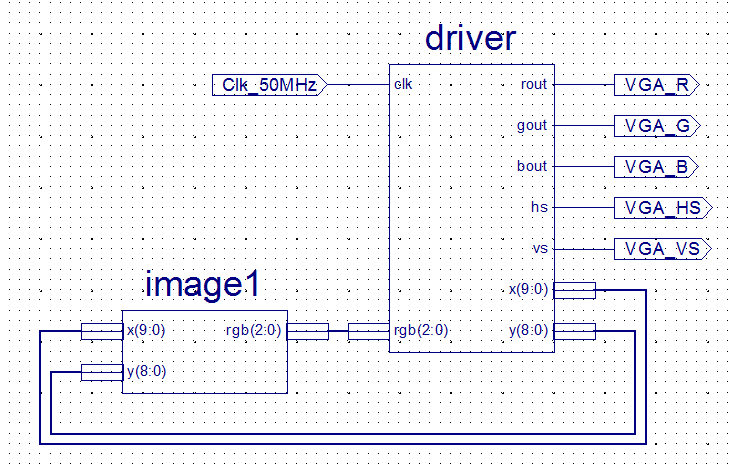
Układy cyfrowe i systemy wbudowane 1 – laboratorium mgr inż. Antoni Sterna Pt 1245 - 1500

**SPRAWOZDANIE**

W czasie zajęć naszym zadaniem było napisanie sterownika dla monitora podłączonego za pomocą portu VGA do układu Spartan. Sterownik miał generować sygnał akceptowalny przez monitor i wyświetlać różne obrazy w zależności od ustawienia przełączników na układzie. Ćwiczenie zrealizowaliśmy w programie Xilnix ISE Project Navigator.

1. **Sterownik monitora VGA**

Budowę układu rozpoczęliśmy od stworzenia modułu sterownika. Posiada on wejście dla zegara 50 MHz, który wewnątrz modułu dzielony jest w ten sposób, że powstaje sygnał 25 MHz. Wyjściami modułu są linie odpowiadające podstawowym kolorom RGB oraz linie synchronizacji poziomej i pionowej. Ponadto wyprowadziliśmy linie przekazujące współrzędne aktualnie rysowanego piksela obrazu, co było niezbędne do wyświetlenia obrazka na monitorze (aby rysowanie odbywało się we właściwym miejscu ekranu). Za generowanie obrazka odpowiedzialny był kolejny moduł, który na podstawie podanych przez sterownik współrzędnych przesyłał do sterownika informacje na temat ustawień kolorów dla danego punktu. Obraz generowany przez układ charakteryzował się rozdzielczością 640x480 punktów przy odświeżaniu 60 Hz. Przy projektowaniu układu sterownika bardzo pomocna okazała się dokumentacja układu Spartan zamieszczona na stronach firmy Xilinx. Znaleźliśmy tam informacje ile cykli zegara powinny trwać poszczególne sygnały i w których momentach następują zmiany stanów linii synchronizacji, dzięki czemu implementacja okazała się bezproblemowa. Przed zaprogramowaniem naszej realizacji na układzie, sprawdziliśmy jej działanie w symulatorze, w którym mogliśmy prześledzić kluczowe momenty zmian i ogólną poprawność działania sterownika. Spartan może wyświetlać 8 kolorów (kombinacja 0 i 1 dla linii R,G,B). Pełny schemat układu przedstawiał się następująco:



A oto kod sterownika:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity driver is

port

(

rgb : in std\_logic\_vector(2 downto 0);

clk : in std\_logic;

rout : out std\_logic;

gout : out std\_logic;

bout : out std\_logic;

hs : out std\_logic;

vs : out std\_logic;

x : out std\_logic\_vector(9 downto 0);

y : out std\_logic\_vector(8 downto 0)

);

end driver;

architecture driver\_arch of driver is

signal clk\_2 : std\_logic;

signal xcounter : integer range 0 to 799;

signal ycounter : integer range 0 to 520;

begin

process (clk)

begin

if rising\_edge(clk) then

clk\_2 <= not clk\_2;

end if;

end process;

process (clk\_2)

begin

if rising\_edge(clk\_2) then

if xcounter = 799 then

xcounter <= 0;

if ycounter = 520 then

ycounter <= 0;

else

ycounter <= ycounter + 1;

end if;

else

xcounter <= xcounter + 1;

end if;

end if;

end process;

hs <= '0' when xcounter >= 16 and xcounter < 112 else '1';

vs <= '0' when ycounter >= 10 and ycounter < 12 else '1';

process (xcounter, ycounter, rgb)

begin

if xcounter < 160 or ycounter < 41 then

rout <= '0';

gout <= '0';

bout <= '0';

else

rout <= rgb(2);

gout <= rgb(1);

bout <= rgb(0);

end if;

end process;

process (xcounter)

begin

if xcounter >= 160 then

x <= std\_logic\_vector(to\_unsigned(xcounter - 160, x'length));

else

x <= std\_logic\_vector(to\_unsigned(640, x'length));

end if;

end process;

process (ycounter)

begin

if ycounter >= 41 then

y <= std\_logic\_vector(to\_unsigned(ycounter - 41, y'length));

else

y <= std\_logic\_vector(to\_unsigned(480, y'length));

end if;

end process;

end driver\_arch;

**Kod układu generującego obrazek (cztery kwadraty w kolorach niebieskim, czerwonym, żółtym i białym):**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity image1 is

port

(

x : in std\_logic\_vector(9 downto 0);

y : in std\_logic\_vector(8 downto 0);

rgb : out std\_logic\_vector(2 downto 0)

);

end image1;

architecture image1\_arch of image1 is

signal xint : integer range 0 to 640;

signal yint : integer range 0 to 480;

begin

process (x, y)

begin

if xint < 320 then

if yint < 240 then

rgb(2) <= '0';

rgb(1) <= '0';

rgb(0) <= '1';

else

rgb(2) <= '1';

rgb(1) <= '1';

rgb(0) <= '0';

end if;

else

if yint < 240 then

rgb(2) <= '1';

rgb(1) <= '0';

rgb(0) <= '0';

else

rgb(2) <= '1';

rgb(1) <= '1';

rgb(0) <= '1';

end if;

end if;

end process;

xint <= to\_integer(unsigned(x));

yint <= to\_integer(unsigned(y));

end image1\_arch;

**Wnioski:**

Dzięki tym zajęciom mogliśmy poznać i zrozumieć specyfikę funkcjonowania portu VGA oraz jak generowany jest obraz, który później widzimy na ekranie monitora. Dzięki dobrej dokumentacji układu Spartan, a także kilku radom prowadzącego, bez większych problemów udało nam się napisać działający sterownik ekranu. Bardzo ważną sprawą w czasie realizacji okazało się precyzyjne wyliczenie cykli zegarowych, ponieważ monitor pracuje w określonych trybach, w których dane sygnały muszą pojawić się w ściśle określonych momentach, aby obraz został wyświetlony poprawnie. Tutaj z pomocą przyszedł nam symulator, dzięki któremu mogliśmy zobaczyć przebiegi poszczególnych sygnałów oraz krytyczne momenty i porównać je ze specyfikacją. Dzięki temu, po zaprogramowaniu naszej realizacji na układzie, na ekranie monitora zobaczyliśmy zadany obraz.