Animație VGA

Proiect PSN

31.05.2023

Lihaciu Oana-Marcela

Grupa 30215

Îndrumător: Fleger Dan

Cuprins

Contents

1. Rezumat

Rețelele neuronale artificiale au fost concepute pentru a ajuta calculatoarele să proceseze informațiile într-un mod asemănător cu cel al creierului uman.

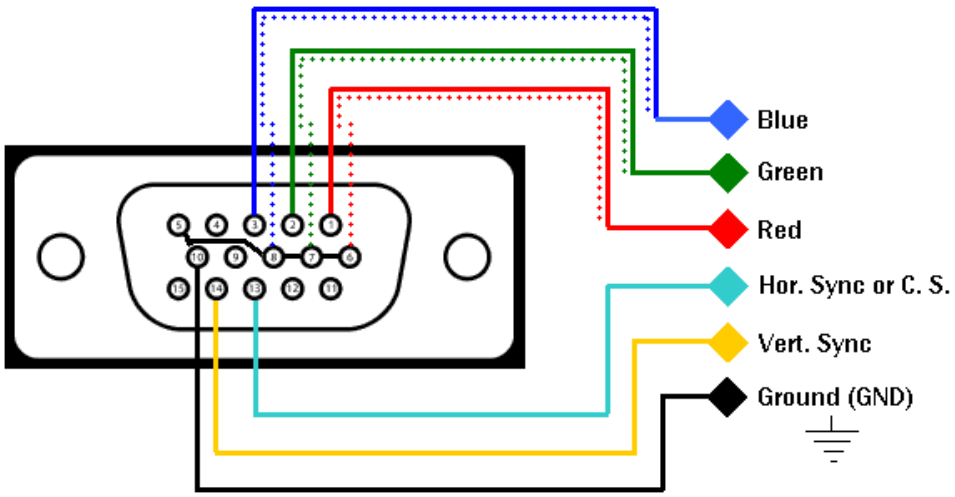
Unul din rezultatele acestui concept este plăcuța FPGA. Programarea sa este diferită față de programarea obișnuită deoarece intrucțiunile sunt implementate ca circuite digitale. Scrierea codului se face în mod tradiționanl cu ajutorul limbajelor de descriere (VHDL, Verilog). Se pot folosi însă și alte limbaje (C, C++).

Obiectivul proiectului este de a crea, folosind plăcuța FPGA, un program care se antrenează singur și care afișează pe ecranul VGA o serie de imagini pentru a crea o animație.

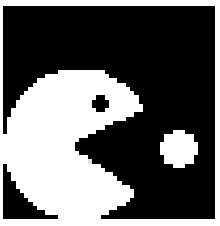
2. Introducere

Monitoarele VGA (“video graphics array) au fost dezvoltate de către IBM și au fost introduse în 1987. Deși nu transmite semnale audio, poate susține o rezoluție de până la 2048 x 1536 și o frecvență de 60Hz.

Cablul în sine conține 15 pini, dintre care se remarcă: R, G, B, HSYNC, VSYNC:



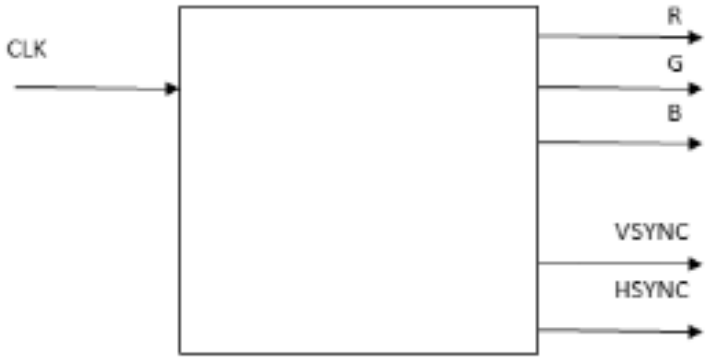
După cum scrie și mai sus, proiectul constă într-o animație pe un ecran VGA. M-am folosit de 23 de variabile de tip matrice care conțin numere de 1 si 0 pentru a reprezenta imagini de 40x40 de pixeli de culoare alb-negru:



După cele 23 de imagini care alcătuiesc animația, urmează încă 9 imagini negre (ecranul este complet negru).

3. Proiectare

- Schema Bloc



Proiectul începe cu crearea noului clock de o secundă pornind de la clock-ul plăcuței Nexys 4 care este de 10ns (100Mhz). Pentru asta avem nevoie de o constanta, pe care am denumit-o counter, de tip integer care numără de la 0 la 50.000.000.

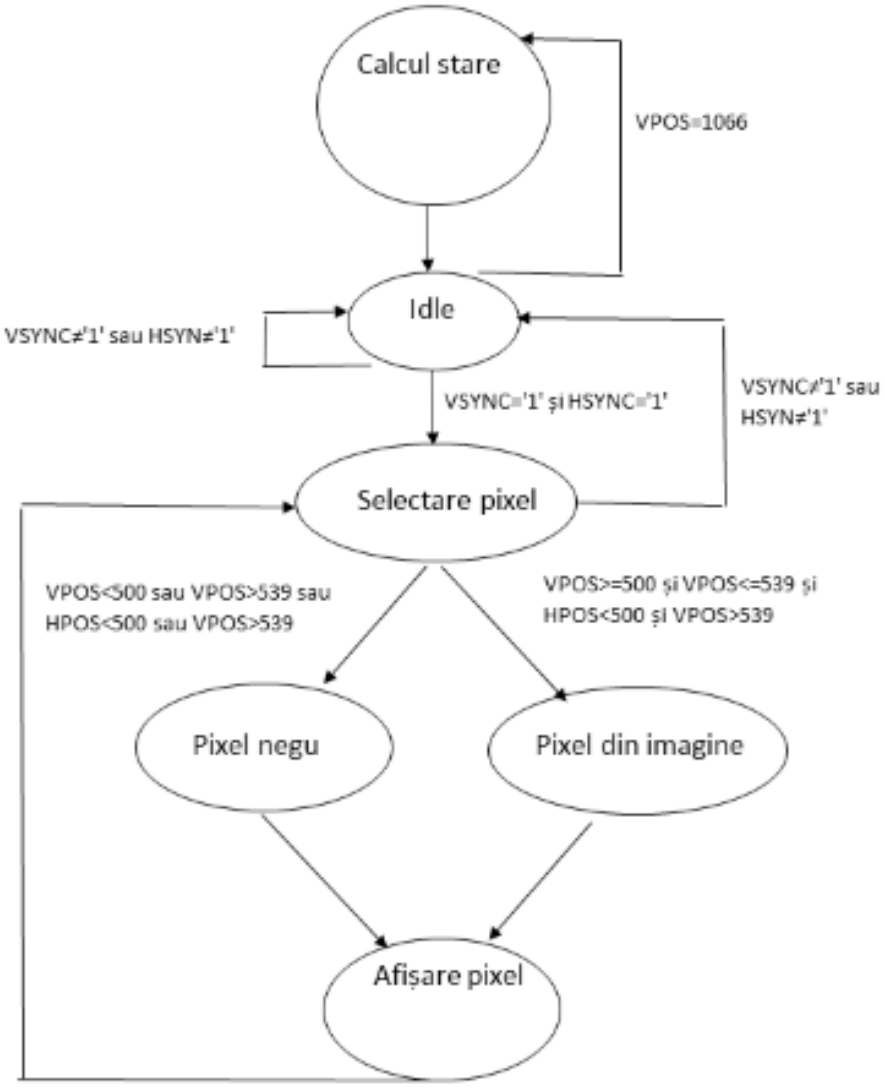
Odată cu noul clock se actualizează și variabila “stare”. Aceasta este de tip std_logic_vector și are în total 5 biți. Scopul variabilei este de a ține minte ce imagine se afișează pe ecran. De la 0 la 22 avem imagini cu pacman iar de la 23 până la 31 avem imagini negre, adică o pauză.

Imaginea va fi desenată începând cu coordonatele (500,500) până la (539, 539), deci când HPOS>=500 și HPOS<=539 și VPOS>=500 și VPOS<=539.

x și y au un rol asemănător cu al lui HPOS și VPOS, doar că nu reprezintă coordonatele ecranului în pixeli, ci coordonatele din matricea logică a imaginilor

Dacă VSYNC și HSYNC sunt active simultan, înseamnă că suntem în zona vizibilă a ecranului și se poate realiza “desenarea” lui.

- Organigrama



4. Bibliografie

1. https://towardsdatascience.com/neural-network-inference-on-fpgas-d1c20c479e84
2. https://www.computerhope.com/jargon/v/vga.htm
3. https://aws.amazon.com/what-is/neural-network

5. Anexă

➢ VGA_animație_pacman:

- Entitate

```
entity VGA_animatie_pacman is
Port ( signal clk:in std_logic;
       signal R:out std_logic_vector(3 downto 0);
       signal G:out std_logic_vector(3 downto 0);
       signal B:out std_logic_vector(3 downto 0);
       signal VSYNC:out std_logic;
       signal HSYNC:out std_logic);
end VGA_animatie_pacman;
```

- Procedură

```
architecture Behavioral of VGA_animatie_pacman is
type matrice is array(39 downto 0) of std_logic_vector(39 downto 0);
signal clock_60hz:std_logic;
signal counter:integer:=0;
signal stare:std_logic_vector(4 downto 0); --22 de imagini<32=2^5
signal HPOS:integer:=0;
signal VPOS:integer:=0;
```

```
signal x:integer:=0;
signal y:integer:=0;


begin
  ----------divizorul de frecventa de la 100Mhz la 60Hz----------
   process(clk)
   begin
   if clk='1' and clk'event then
   if counter<=49999999 then
     counter<=counter+1;
   else
     counter<=0;
     clock_60hz<=not(clock_60hz);
     stare<=stare+1;  --putea fi pus la un loc cu  x<=0, y<=0?
   end if;
   end if;
   end process;

   ----------imagine in functie de stare----------
   process(clock_60hz)
   begin
   if clock_60hz='1' and clock_60hz'event then
```

```
case stare is
 when "00000" => img<=pacman_0;
 when "00001" => img<=pacman_1;
 when "00010" => img<=pacman_2;
 when "00011" => img<=pacman_3;
 when "00100" => img<=pacman_4;
 when "00101" => img<=pacman_5;
 when "00110" => img<=pacman_6;
 when "00111" => img<=pacman_7;
 when "01000" => img<=pacman_8;
 when "01001" => img<=pacman_9;
 when "01010" => img<=pacman_10;
 when "01011" => img<=pacman_11;
 when "01100" => img<=pacman_12;
 when "01101" => img<=pacman_13;
 when "01110" => img<=pacman_14;
 when "01111" => img<=pacman_15;
 when "10000" => img<=pacman_16;
 when "10001" => img<=pacman_17;
 when "10010" => img<=pacman_18;
 when "10011" => img<=pacman_19;
 when "10100" => img<=pacman_20;
 when "10101" => img<=pacman_21;
 when "10110" => img<=pacman_22;
```

```
 when others  => img<=img;
end case;
end if;
end process;


----------parcurgere ecran----------

process(clk)
begin

if clk='1' and clk'event then

    if HPOS<1688 then
      HPOS<=HPOS+1;   --daca nu s-a parcurs o linie, se incrementeaza HPOS
    else
      HPOS<=0;        --daca o s-a finalizat parcurgerea unei linii, se incepe de la 0 de la urmatoarea linie
      if VPOS<1066 then
        VPOS<=VPOS+1;
      else
        VPOS<=0;    --se reseteazaa ecranul
      end if;
```

```
    end if;

end if;
end process;


------------culorile de pe ecran-------------------------
process(clk, HPOS, VPOS)
begin

if clk='1' and clk'event then

    if HPOS>=500 and HPOS<=539 and VPOS>=500 and VPOS<=539
then

      --se actualizeaza coordonatele matricei
      if x<40 then
        if y<40 then
          y<=y+1;       --se printeaza linia actuala
        else
          y<=0;         --se trece la urmatoarea linie
          x<=x+1;       --OBS: x si y vor trebui resetate la 0 cand se
schimba imaginea
        end if;
```

```
  else
      x<=0;
  end if;

  --se printeaza imaginea--
  if img(x)(y)='0' then
    --culoare neagra--
    R<="0000";
    G<="0000";
    B<="0000";

  else
    --culoare alba--
    R<="1111";
    G<="1111";
    B<="1111";
  end if;

else
    --culoare default = neagra--
    R<="0000";
    G<="0000";
    B<="0000";
end if;
```

```
    end if;
    end process;

    ------------perioada de sincronizare----------------------

    process(clk, HPOS, VPOS)
    begin

    if HPOS>=48 and HPOS<160 then
       HSYNC<='0';
    else
       HSYNC<='1';
    end if;

    if VPOS>=1 and VPOS<4 then
       VSYNC<='0';
    else
       VSYNC<='1';
    end if;

    end process;
end Behavioral;
```

➢ Nexys4:

- Entitate

```
entity Nexys4 is
Port ( signal clk:in std_logic;
signal VGA_R:out std_logic_vector(3 downto 0);
signal VGA_G:out std_logic_vector(3 downto 0);
signal VGA_B:out std_logic_vector(3 downto 0);
signal VGA_HS:out std_logic;
signal VGA_VS:out std_logic
);
end Nexys4;
```

- Procedură

```
architecture Behavioral of Nexys4 is
begin
display:entity WORK.VGA_animatie_pacman port map
(clk=>clk,
HSYNC=>VGA_HS,
VSYNC=>VGA_VS,
R=>VGA_R,
G=>VGA_G,
B=>VGA_B);
```

end Behavioral;

➢ Constraints:

```
set_property -dict { PACKAGE_PIN A3    IOSTANDARD LVCMOS33 }
[get_ports { VGA_R[0] }]; #IO_L8N_T1_AD14N_35 Sch=vga_r[0]

set_property -dict { PACKAGE_PIN B4    IOSTANDARD LVCMOS33 }
[get_ports { VGA_R[1] }]; #IO_L7N_T1_AD6N_35 Sch=vga_r[1]

set_property -dict { PACKAGE_PIN C5    IOSTANDARD LVCMOS33 }
[get_ports { VGA_R[2] }]; #IO_L1N_T0_AD4N_35 Sch=vga_r[2]

set_property -dict { PACKAGE_PIN A4    IOSTANDARD LVCMOS33 }
[get_ports { VGA_R[3] }]; #IO_L8P_T1_AD14P_35 Sch=vga_r[3]


set_property -dict { PACKAGE_PIN C6    IOSTANDARD LVCMOS33 }
[get_ports { VGA_G[0] }]; #IO_L1P_T0_AD4P_35 Sch=vga_g[0]

set_property -dict { PACKAGE_PIN A5    IOSTANDARD LVCMOS33 }
[get_ports { VGA_G[1] }]; #IO_L3N_T0_DQS_AD5N_35 Sch=vga_g[1]

set_property -dict { PACKAGE_PIN B6    IOSTANDARD LVCMOS33 }
[get_ports { VGA_G[2] }]; #IO_L2N_T0_AD12N_35 Sch=vga_g[2]

set_property -dict { PACKAGE_PIN A6    IOSTANDARD LVCMOS33 }
[get_ports { VGA_G[3] }]; #IO_L3P_T0_DQS_AD5P_35 Sch=vga_g[3]


set_property -dict { PACKAGE_PIN B7    IOSTANDARD LVCMOS33 }
[get_ports { VGA_B[0] }]; #IO_L2P_T0_AD12P_35 Sch=vga_b[0]

set_property -dict { PACKAGE_PIN C7    IOSTANDARD LVCMOS33 }
[get_ports { VGA_B[1] }]; #IO_L4N_T0_35 Sch=vga_b[1]
```

```
set_property -dict { PACKAGE_PIN D7    IOSTANDARD LVCMOS33 } [get_ports { VGA_B[2] }]; #IO_L6N_T0_VREF_35 Sch=vga_b[2]
set_property -dict { PACKAGE_PIN D8    IOSTANDARD LVCMOS33 } [get_ports { VGA_B[3] }]; #IO_L4P_T0_35 Sch=vga_b[3]

set_property -dict { PACKAGE_PIN B11   IOSTANDARD LVCMOS33 } [get_ports { VGA_HS }]; #IO_L4P_T0_15 Sch=vga_hs
set_property -dict { PACKAGE_PIN B12   IOSTANDARD LVCMOS33 } [get_ports { VGA_VS }]; #IO_L3N_T0_DQS_AD1N_15 Sch=vga_vs
```