Circuito Controlador VGA

Pedro Henrique Augustin - Rodrigo Dal Ri

Engenharia de Computação, Instituto de Informática
Universidade Federal do Rio Grande do Sul

I. Introdução

Esse trabalho tem como objetivo criar um circuito que apresente uma imagem estática na tela de um monitor com entrada VGA. Será feito com uma descrição em VHDL com implementação em FPGA.

O trabalho foi realizado em conjunto com o colega Pedro Henrique Augustin.

II. Descrição do projeto

No padrão VGA são utilizados 5 sinais de controle.

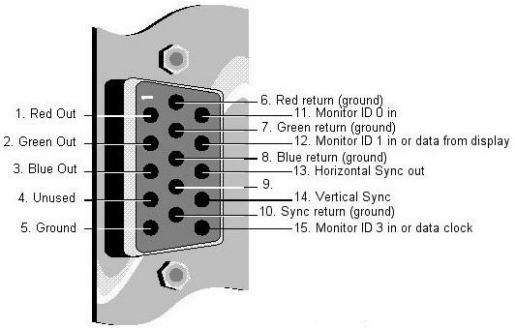


Figura 1 VGA Pinos

Dos pinos mostrados na Figura 1 somente cinco são utilizados no controle de vídeo. São os pinos: 1, 2, 3, 13 e 14. Os pinos 1, 2 e 3 são utilizados para determinar a cor (RGB) de um determinado pixel, já os pinos 13 e 14 são utilizados na sincronização do vídeo.

Um frame VGA define um conjunto de parâmetros que devem ser seguidos para sincronizar o vídeo com o mecanismo de varredura da tela.

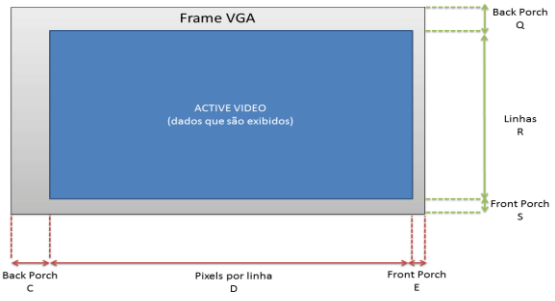


Figura 2 VGA Frame

Convém observar que além da região ativa de vídeo outros parâmetros de tempo devem ser respeitados.

Tanto na varredura horizontal quanto na vertical são estabelecidos alguns parâmetros como: Back Porch e Front Porch.

Esses parâmetros estão relacionados com os sinais Horizontal Sync (H_SYNC) e Vertical Sync (V_SYNC) que são utilizados para sincronizar a varredura da tela.

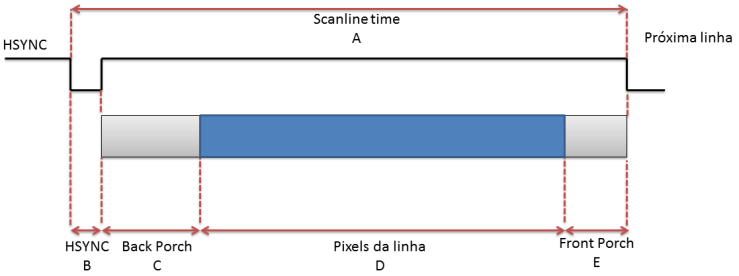


Figura 3 Sincronização horizontal

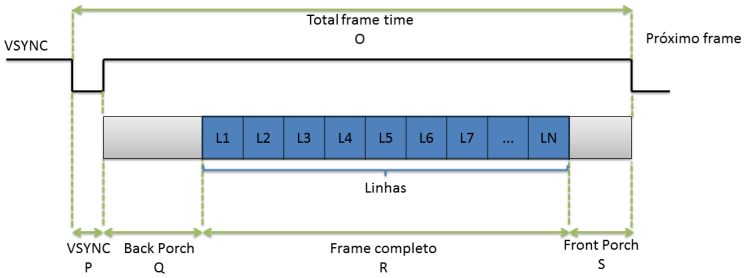


Figura 4 Sincronização vertical

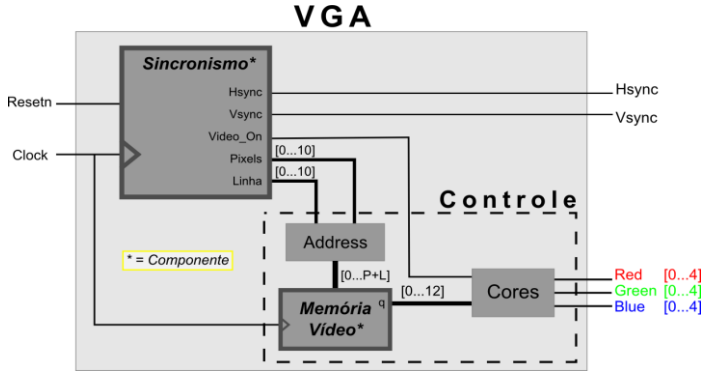


Figura 5 Modelo Controlador VGA

Este é o esquemático do controlador VGA a ser implementado em VHDL e na placa FPGA.

A memória foi implementada através de uma megafunção da Altera. Na memória, está um arquivo .mif(Memory Initialization File) que possui o mapeamento da imagem a ser mostrada na tela. O arquivo .mif é indicado através do MegaWizard.

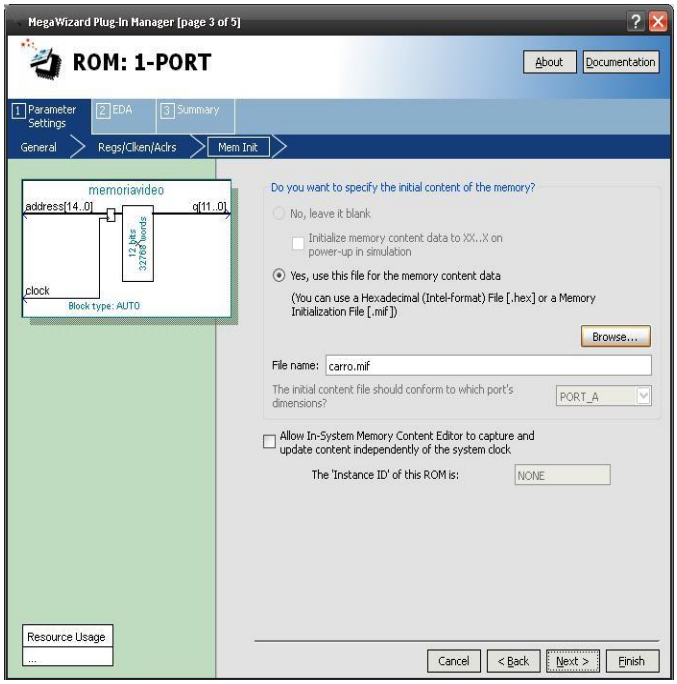


FIGURA 6 MEGAWIZARD

III. IMPLEMENTAÇÃO

Abaixo seguem os anexos com o código fonte implementado.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity tb_VGA is
end entity tb_VGA;

architecture interface of tb_VGA is
signal      clock, resetn     :  std_logic;
signal      red, green, blue  :  std_logic_vector (3 downto 0);
signal      Hsync, Vsync      :  std_logic;
signal      true              :  std_logic;

begin
uut:  entity work.VGA
port map
    (
         clock, resetn,
       red, green, blue,
       Hsync, Vsync
   );
geraclock:
process
begin
       clock <= '0';  -- Na partida, zera sinal de clock...
       wait for 10ns;
       -- Agora, comeca a oscilar:
       for cont in 0 to 3600000 loop
          clock <= not clock;
          wait for 10 ns;
       end loop;
       wait; -- Parando este processo...
end process;
comportamento:
```

FIGURA 7 CÓDIGO

```
process
    begin
       true <= '1';
       RESETn <= '0';
       wait for 50 ns;
       RESETn <= '1';
       wait for 300 us;
       true <= '0';
       wait;
end process;
end architecture interface;
```

FIGURA 8 CÓDIGO

```
library IEEE;
use  IEEE.STD_LOGIC_1164.all;
use  IEEE.STD_LOGIC_ARITH.all;
use  IEEE.STD_LOGIC_UNSIGNED.all;


entity VGA is

   port(
      clock, resetn     :  in std_logic;
      red, green, blue  :  out std_logic_vector (3 downto 0);
      Hsync, Vsync      :  out std_logic

   );
end entity VGA;

architecture comportamento of VGA is
   signal address: std_logic_vector(14 downto 0);
   signal Pixels,Linha : std_logic_vector(10 downto 0);
   signal RGB, RGB_temp: std_logic_vector(11 downto 0);
   signal video_on: std_logic;

   component memoriavideo
      PORT
      (
         address           : IN STD_LOGIC_VECTOR (14 DOWNTO 0);
         clock             : IN STD_LOGIC;
         q                 : OUT STD_LOGIC_VECTOR (11 DOWNTO 0)
      );
   end component memoriavideo;
```

FIGURA 9 CÓDIGO

```
component sincronismo
   PORT
      (
            Clock, Resetn           :  in std_logic;
            Video_on                :  out std_logic;
            Pixels,Linha            : out std_logic_vector(10 downto 0);
            Horiz_sync,Vert_sync    : out std_logic
      );
end component sincronismo;

begin

   memoriavideo_inst : memoriavideo PORT MAP (
         address   => address,
         clock  => clock,
         q   => RGB_temp
      );

   sincronismo_inst : sincronismo port map (
         clock => clock,
         Resetn => resetn,
         Video_on => Video_on,
         Pixels => Pixels,
         Linha => Linha,
         Horiz_sync => Hsync,
         Vert_sync => Vsync
   );
```

FIGURA 10 CÓDIGO

```
Cores:process (clock,RGB,Video_on)
   begin
      if (Video_on = '0') Then
         red   <= "0000";
         green <= "0000";
         blue <= "0000";

      else
         red   <= RGB(11 downto 8);
         green <= RGB( 7 downto 4);
         blue  <= RGB( 3 downto 0);
      end if;
   end Process Cores;

   Enderecamento: process(Linha,Pixels,clock)
      begin

         Address <= NOT Linha(6 downto 0) & Pixels(7 downto 0);
         -- Enviando enderecos para a memoria:
         if (Pixels <= 255) and (Linha <= 127) then
         -- utilizo pixels de 8 ate 1 descartandando o ultimo bit.
         -- isso divide a frequencia por 2 de forma com que o clock seja semelh
            RGB <= RGB_temp;
         else
            RGB <= "000000000000";
         end if;
      end Process Enderecamento;

end comportamento;
```

FIGURA 11 CÓDIGO

```
library IEEE;
use  IEEE.STD_LOGIC_1164.all;
use  IEEE.STD_LOGIC_ARITH.all;
use  IEEE.STD_LOGIC_UNSIGNED.all;


entity sincronismo is

   Generic(ADDR_WIDTH: integer := 12; DATA_WIDTH: integer := 1);

   port( signal Clock, Resetn : in std_logic;
         signal video_on : out std_logic;
         signal Pixels,Linha : out std_logic_vector(10 downto 0);
         signal Horiz_sync,Vert_sync : out std_logic);

end sincronismo;

architecture comportamento of sincronismo is

   signal cont_x,cont_y: std_logic_vector(10 Downto 0);
   signal RESET : std_logic;
   constant H_max : std_logic_vector(10 Downto 0) := CONV_STD_LOGIC_VECTOR(1588,11);
   --valor maximo da variavel cont_x, valor encontrado a partir da analise dos tempos do sincronismo horizontal
   constant V_max : std_logic_vector(10 Downto 0) := CONV_STD_LOGIC_VECTOR(528,11);
   --valor maximo da variavel cont_y, valor encontrado a partir da analise dos tempos do sincronismo vertical
   signal video_on_H, video_on_V: std_logic;
```

FIGURA 12 CÓDIGO

```
begin

   RESET <= NOT(resetn);
   video_on <= video_on_H and video_on_V;

   --Generate Horizontal and Vertical Timing Signals for Video Signal
   VIDEO_DISPLAY: Process
      Begin

      Wait until(Clock'Event) and (Clock='1');

      If
         Reset = '1' Then
         cont_x <= CONV_STD_LOGIC_VECTOR(0,11);
         cont_y <= CONV_STD_LOGIC_VECTOR(0,11);
         Video_on_H <= '0';
         Video_on_V <= '0';
      Else

         -- cont_x conta os pixels (espaco utilizado+espaco nao utilizado+tempo extra para o sinal de sincr
         --
         --  Contagem de Pixels:
         --                                          <-H Sync->
         --   ------------------------------------_________
         --   0        511 -espaco nao utilizado-  1400


          If (cont_x >= H_max) then
            cont_x <= "00000000000";
          Else
            cont_x <= cont_x + "00000000001";
          End if;

         -- O Horiz_Sync deve permanecer em nivel logico alto por 27,06 us
         -- entao em baixo por 3,77 us
```

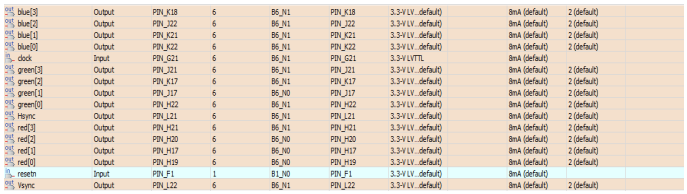
FIGURA 13 CÓDIGO

```
          If (cont_x <= 1494) and (cont_x >= 1306) Then
               Horiz_Sync <= '0';
          ELSE
               Horiz_Sync <= '1';
          End if;

         -- Ajusta o tempo do Video_on_H
          If (cont_x <= 1258) Then
            video_on_H <= '1';
          ELSE
            video_on_H <= '0';
          End if;

         --  Contagem de linhas...
         --Linha conta as linhas de pixels (127 + tempo extra para sinais de sincronismo)
         --
         --  <--128 linhas utilizadas -->                  ->V Sync<-
         --  ------------------------------------------------_______-------------
         --  0                           127           495-494               528
         --

          If (cont_y >= V_max) and (cont_x >= 736) then
               cont_y <= "00000000000";
          Elsif (cont_x = H_Max) Then
                cont_y <= cont_y + "00000000001";
          End if;

         -- Generate Vertical Sync Signal
          If (cont_y <= 496) and (cont_y >= 495) Then
            Vert_Sync <= '0';
          ELSE
            Vert_Sync <= '1';
          End if;
```

FIGURA 14 CÓDIGO

```
         -- Ajusta o tempo do Video_on_V
         If (cont_y <= 479) Then
            video_on_V <= '1';
         ELSE
            video_on_V <= '0';
         End if;

      End if; -- Termina o IF do Reset

      Linha    <= cont_y;
      Pixels   <= "0" & cont_x(10 downto 1);
            -- Utilizo cont_x descartandando o ultimo bit para dividir por 2 a frequencia
            -- De forma com que o clock seja semelhante ao do monitor.

      end process VIDEO_DISPLAY;

end comportamento;
```

FIGURA 15 CÓDIGO



blue[3]	Output	PIN_K18	6	B6_N1	PIN_K18	3.3-V LV...default)	8mA (default)	2 (default)
blue[2]	Output	PIN_J22	6	B6_N1	PIN_J22	3.3-V LV...default)	8mA (default)	2 (default)
blue[1]	Output	PIN_K21	6	B6_N1	PIN_K21	3.3-V LV...default)	8mA (default)	2 (default)
blue[0]	Output	PIN_K22	6	B6_N1	PIN_K22	3.3-V LV...default)	8mA (default)	2 (default)
clock	Input	PIN_G21	6	B6_N1	PIN_G21	3.3-V LVTTL	8mA (default)	
green[3]	Output	PIN_J21	6	B6_N1	PIN_J21	3.3-V LV...default)	8mA (default)	2 (default)
green[2]	Output	PIN_K17	6	B6_N1	PIN_K17	3.3-V LV...default)	8mA (default)	2 (default)
green[1]	Output	PIN_J17	6	B6_N0	PIN_J17	3.3-V LV...default)	8mA (default)	2 (default)
green[0]	Output	PIN_H22	6	B6_N1	PIN_H22	3.3-V LV...default)	8mA (default)	2 (default)
Hsync	Output	PIN_L21	6	B6_N1	PIN_L21	3.3-V LV...default)	8mA (default)	2 (default)
red[3]	Output	PIN_H21	6	B6_N1	PIN_H21	3.3-V LV...default)	8mA (default)	2 (default)
red[2]	Output	PIN_H20	6	B6_N0	PIN_H20	3.3-V LV...default)	8mA (default)	2 (default)
red[1]	Output	PIN_H17	6	B6_N0	PIN_H17	3.3-V LV...default)	8mA (default)	2 (default)
red[0]	Output	PIN_H19	6	B6_N0	PIN_H19	3.3-V LV...default)	8mA (default)	2 (default)
resetn	Input	PIN_F1	1	B1_N0	PIN_F1	3.3-V LV...default)	8mA (default)	
Vsync	Output	PIN_L22	6	B6_N1	PIN_L22	3.3-V LV...default)	8mA (default)	2 (default)

FIGURA 16 PINAGEM

IV. RESULTADOS PRÁTICOS

Flow Summary	
Flow Status	Successful - Tue Nov 17 15:59:06 2015
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	VGA
Top-level Entity Name	VGA
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	114 / 15,408 (< 1 %)
Total combinational functions	110 / 15,408 (< 1 %)
Dedicated logic registers	49 / 15,408 (< 1 %)
Total registers	49
Total pins	16 / 347 (5 %)
Total virtual pins	0
Total memory bits	393,216 / 516,096 (76 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	0 / 4 (0 %)

FIGURA 17 RESULTADO DA ANÁLISE E SÍNTESE

FIGURA 18 TESTE REALIZADO EM LABORATÓRIO

V. CONCLUSÕES FINAIS

Neste trabalho realizamos a implementação de um circuito controlador VGA descrito em VHDL e implementado na FPGA. Tivemos problemas apenas em conseguir imagens no formato .mif, no final acabamos criando um arquivo e escrevendo as cores de cada pixel manualmente.

REFERÊNCIAS

[1] (2015) VGA Controller (VHDL) [Online] Available: https://eewiki.net/pages/viewpage.action?pageId=15925278

[2] (2015) Fpga para Todos [Online] Available: http://fpgaparatodos.com.br/exemplos-de-aplicacao/exemplos-com-fpga/24-vga-para-fpga.html

[3] (2015) Stack Overflow [Online] Available: http://stackoverflow.com/questions/26388004/vga-controller-with-vhdl