1 2 UNIVERSIDADE D 9 0
COIMBRA

deer.uc
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Departamento de Engenharia Eletrotécnica e de Computadores

Projeto de Sistemas Digitais

Professor: Jorge Manuel Miranda Dias

Editor Grafico

Autor: — *Numero de Estudante:*
Gonçalo Bastos — 2020238997
Leonardo Cordeiro — 2020228071

22 Maio, 2024

1 Introdução

Este projeto destina-se ao desenvolvimento de uma interface que permitirá desenhar utilizando um mouse *PS 2* e um ecrã conectado por VGA mediante da utilização de uma placa *FPGA*. O editor faz uso da resolução total fornecida pelo módulo *VGA_sync* fornecido, `640x480`. O mouse é identificado com um cursor de *4x4 pixeis* e é possível a utilização de 8 cores diferentes.

2 Desenvolvimento

2.1 Controlador VGA

O módulo *VGA_sync* é crucial para o desenvolvimento deste trabalho, o módulo recebe os parâmetros RGB que vai imprimir no ecrã para cada pixel. Os pixeis são dados pelo *pixel_col e pixel_row* de 10 bits cada, que são dados com uma taxa de refrescamento de 60Hz, estes sinais são cruciais para controlar os posicionamento do cursor, bem como as respetivas posições de cada pixel colorido.

2.2 Memória RAM

De modo a guardar os pixeis coloridos, bem como as suas posições foi necessário implementar uma memória RAM recorrendo á livraria megafunctions da altera, especificamente o módulo *lpm_ram_dp*. A RAM foi definida com 3 bits de entrada para guardar as respetivas cores, foi necessário 16 bits para indexar os endereços de escrita de leitura, o endereço de escrita é indexado pelos 9 bits mais significativos das linhas e colunas do mouse e o endereço de leitura pelos 9 bits mais significativos da linhas e colunas provenientes do módulo VGA. A escrita para a memória pode ser ativada em 3 casos diferentes: o botão esquerdo é premido, ou seja queremos guardar o desenho, o botão direito é premido e queremos apagar e por último o botão de clear é pressionado e é feito um reset da interface.

2.3 Interface Com o Mouse

A comunicação com o mouse é assegurada pelo módulo *"mouse"* fornecido, no qual é obtida as coordenadas do mouse e ainda obter o sinal se o botâo esquerdo ou direito foram primidos. Com base nas coordenadas do mouse ee também nas coordenadas do VGA foi definido o módulo *"cursor"* que vai definir o cursor, quando as coordenadas do Mouse forem iguais às coordenadas do *pixel_row e pixel_col* provenientes do VGA é mostrado o sensor. Este módulo também é responsável por imprimir no ecrã os restantes pixeis pintados guardados na memória RAM.

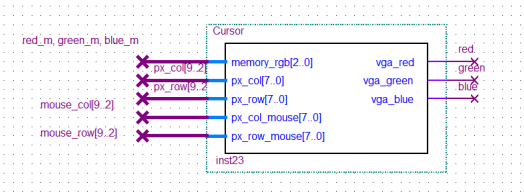


Figure 1: Módulo Mouse

```
5   ENTITY Cursor IS
6       PORT(
7           memory_rgb : IN STD_LOGIC_VECTOR(2 DOWNTO 0); -- RGB values
8           px_col : IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- VGA col
9           px_row : IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- VGA row
10          px_col_mouse : IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- Mouse col
11          px_row_mouse : IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- Mouse row
12          vga_red : OUT STD_LOGIC;
13          vga_green : OUT STD_LOGIC;
14          vga_blue : OUT STD_LOGIC
15      );
16  END ENTITY Cursor;
17
18  ARCHITECTURE Behavior OF Cursor IS
19  BEGIN
20      PROCESS(memory_rgb, px_col, px_row, px_col_mouse, px_row_mouse)
21      BEGIN
22
23          vga_red <= memory_rgb(0);
24          vga_green <= memory_rgb(1);
25          vga_blue <= memory_rgb(2);
26
27          -- Set the cursor
28          IF (px_col = px_col_mouse) AND (px_row = px_row_mouse) THEN
29              vga_red <= '1';
30              vga_green <= '1';
31              vga_blue <= '1';
32          ELSE
33              vga_red <= memory_rgb(0);
34              vga_green <= memory_rgb(1);
35              vga_blue <= memory_rgb(2);
36          END IF;
37      END PROCESS;
38
39  END ARCHITECTURE Behavior;
```

Figure 2: VHDL - Módulo Cursor

2.4 Módulo de controlo

Foi desenvolvido um módulo de controlo principal nomeado por *PaintControl* nesde módulo definimos a cor que é definida mediante da utilização de 3 entradas SW (SW[2..0]). Este módulo define a cor que vai ser escrita na memória RAM, identificando também se ou o botão esquerdo ou direito é premido, esta verificação vai habilitar ou não a escrita na memória RAM. Caso o botão direito seja premido vamos escrever a cor do fundo, neste caso preto, caso seja premido o botão esquerdo o desenho é habilitado e é escrita a cor definida pela as 3 entradas, neste último caso os endereços de escrita vão ser iguais aos endereços de leitura pois queremos apagar tudo o que foi escrito no ecrã até ao momento.

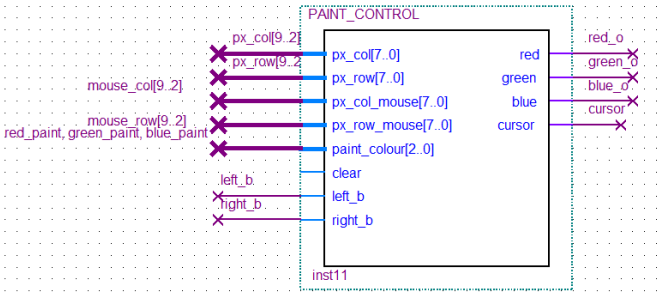


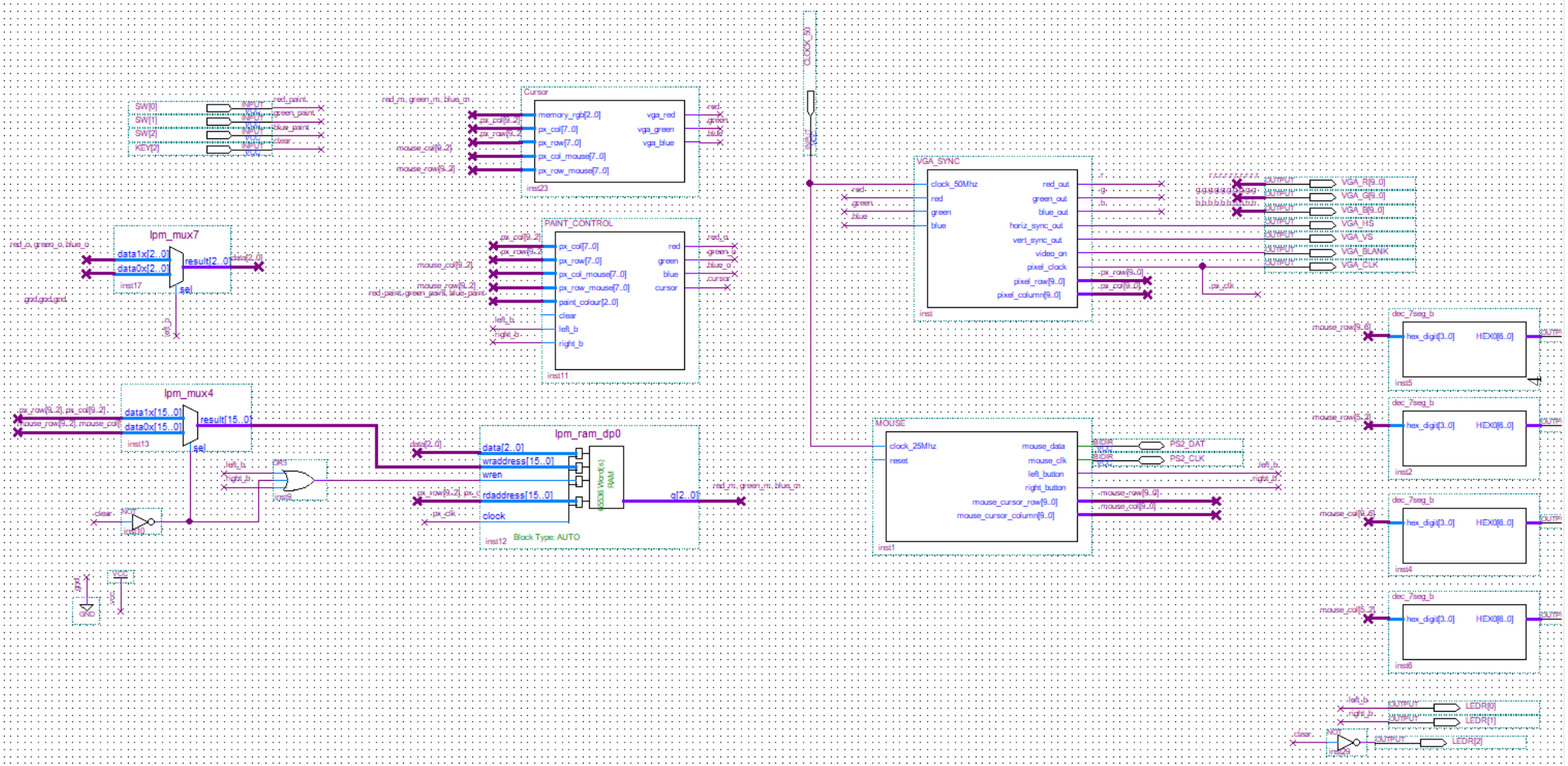
Figure 3: Módulo PaintControl

```
    BEGIN

    px_col_int := TO_INTEGER(unsigned(px_col));
    px_row_int := TO_INTEGER(unsigned(px_row));
    px_col_mouse_int := TO_INTEGER(unsigned(px_col_mouse));
    px_row_mouse_int := TO_INTEGER(unsigned(px_row_mouse));

    if(left_b = '1') then
        if((px_col_int = px_col_mouse_int) AND (px_row_int = px_row_mouse_int)) then
            red <= paint_colour(2);
            green <= paint_colour(1);
            blue <= paint_colour(0);
            cursor <= '1';
        end if;
    elsif(right_b = '1') then
        if((px_col_int = px_col_mouse_int) AND (px_row_int = px_row_mouse_int)) then
            red <= '0';
            green <= '0';
            blue <= '0';
            cursor <= '1';
        end if;
    else
        --Set the default background colour
        if((px_col_int = px_col_mouse_int) AND (px_row_int = px_row_mouse_int)) then
            red <= paint_colour(2);
            green <= paint_colour(1);
            blue <= paint_colour(0);
            cursor <= '1';
        end if;

    end if;

    END PROCESS;
END Behavior;
```

Figure 4: VHDL - PaintControl Behavior

2.5 Conclusão

O objetivo geral do trabalho foi alcançado, ou seja implementámos com sucesso um editor grafico de desenho com o rato com possibilidade de mudança de cores, e com possibilidade de apagar partes do desenho ou o desenho todo. Na figura abaixo está o esquema completo do módulo de mais alto nível do trabalho.


Cursor
memory_rgb[2..0]
px_col[7..0]
px_row[7..0]
px_col_mouse[7..0]
px_row_mouse[7..0]
vga_red
vga_green
vga_blue
inst23
PAINT_CONTROL
px_col[7..0]
px_row[7..0]
px_col_mouse[7..0]
px_row_mouse[7..0]
paint_colour[2..0]
clear
left_b
right_b
red
green
blue
cursor
inst11
VGA_SYNC
clock_50Mhz
red
green
blue
red_out
green_out
blue_out
horiz_sync_out
vert_sync_out
video_on
pixel_clock
pixel_row[9..0]
pixel_column[9..0]
inst
VGA_R[9..0]
VGA_G[9..0]
VGA_B[9..0]
VGA_HS
VGA_VS
VGA_BLANK
VGA_CLK
lpm_mux7
data1x[2..0]
data0x[2..0]
result[2..0]
sel
inst17
lpm_mux4
data1x[15..0]
data0x[15..0]
result[15..0]
sel
inst13
lpm_ram_dp0
data[2..0]
wraddress[15..0]
wren
rdaddress[15..0]
clock
q[2..0]
65536 Word(s) RAM
inst12
Block Type: AUTO
MOUSE
clock_25Mhz
reset
mouse_data
mouse_clk
left_button
right_button
mouse_cursor_row[9..0]
mouse_cursor_column[9..0]
inst1
PS2_DAT
PS2_CLK
dec_7seg_b
hex_digit[3..0]
HEX0[6..0]
inst5
inst2
inst4
inst6
LEDR[0]
LEDR[1]
LEDR[2]
SW[0]
SW[1]
SW[2]
KEY[2]
GND
VCC