Oakland University
Electrical and Computer Engineering Department
ECE 378 - Final Project

Paint Tool

Duncan Taylor
Daniel Wolfe
Cabir Yavuz

04.14.2015
Paint Tool
Datapath Circuit

TopFile.vhd

VGA_Control.vhd

mouse_top.vhd

Datapath Circuit

RGB

VS

HS

resetn

clk

ps2d

ps2c

cls

mouseoutx

mouseouty

mClick

dout_b

clock

dout_a

resetn

VGA

SW

VGA

R

G

B

HS

VS

sw

addra

addrb

din_a

resetn

RAM

we

dout_b

dout_a

hcount

vcount

resetn

clk

clk

mouseoutx

mouseouty

ps2d

ps2c

posX

posY

mClick

mClick
PS2 Mouse - Received Data Structure

Register 1

\[
\begin{array}{cccccccccc}
1 & P & y7 & y6 & y5 & y4 & y3 & y2 & y1 & y0 & 0
\end{array}
\]

Register 2

\[
\begin{array}{cccccccccc}
1 & P & x7 & x6 & x5 & x4 & x3 & x2 & x1 & x0 & 0
\end{array}
\]

Register 3

\[
\begin{array}{cccccccccc}
1 & P & yv & xv & yv & ys & xs & 1 & 0 & R & L & 0
\end{array}
\]

PS2 Input \[\rightarrow\] Four Directional Input \[\rightarrow\] Mouse Cursor Coordinates (PosX and PosY)
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity xilinx_dual_port_ram_sync is
  generic(
    ADDR_WIDTH: integer:=19;
    DATA_WIDTH: integer:=4
  );
  port(
    clk: in std_logic;
    we: in std_logic;
    addr_a: in std_logic_vector(ADDR_WIDTH-1 downto 0);
    addr_b: in std_logic_vector(ADDR_WIDTH-1 downto 0);
    din_a: in std_logic_vector(DATA_WIDTH-1 downto 0);
    dout_a: out std_logic_vector(DATA_WIDTH-1 downto 0);
    dout_b: out std_logic_vector(DATA_WIDTH-1 downto 0)
  );
end xilinx_dual_port_ram_sync;

architecture beh_arch of xilinx_dual_port_ram_sync is
  type ram_type is array (0 to 2**ADDR_WIDTH-1) of std_logic_vector (DATA_WIDTH-1 downto 0);
  signal ram: ram_type;
  signal addr_a_reg, addr_b_reg: std_logic_vector(ADDR_WIDTH-1 downto 0);
begin
  process(clk)
  begin
    if (clk'event and clk = '1') then
      if (we = '1') then
        ram(to_integer(unsigned(addr_a))) <= din_a;
      end if;
      addr_a_reg <= addr_a;
      addr_b_reg <= addr_b;
    end if;
  end process;
  dout_a <= ram(to_integer(unsigned(addr_a_reg)));
  dout_b <= ram(to_integer(unsigned(addr_b_reg)));
end beh_arch;
Assigning the RAM Addresses

Address Width = \( \log_2 (640 \times 480) = 19 \) Bits
Data Width = 3 Bits

Address “a” is used to write, so it is set to the mouse cursor

```verilog
addr_a <= conv_std_logic_vector(conv_integer(mouseposx) * 640 + conv_integer(mouseposy), 19);
```

Address “b” is used to read, so it cycles through every pixel with hcount and vcount

```verilog
addr_b <= conv_std_logic_vector(conv_integer(hcount) * 640 + conv_integer(vcount), 19);
```
VGA - MUX

```
if (hcount > (mouseposx - mousew)) and (hcount < (mouseposx + mousew)) and (vcount > (mouseposy - mouseh)) and (vcount < (mouseposy + mouseh)) then
  s := "1010";
elsif hcount <= 80 and vcount <= 51 then
  s := "1001";
elsif hcount > 80 and hcount <= 160 and vcount <= 51 then
  s := "1000";
elsif hcount > 160 and hcount <= 240 and vcount <= 51 then
  s := "0111";
elsif hcount > 240 and hcount <= 320 and vcount <= 51 then
  s := "0110";
elsif hcount > 320 and hcount <= 400 and vcount <= 51 then
  s := "0101";
elsif hcount > 400 and hcount <= 480 and vcount <= 51 then
  s := "0100";
elsif hcount > 480 and hcount <= 560 and vcount <= 51 then
  s := "0011";
elsif hcount > 560 and hcount <= 640 and vcount <= 51 then
  s := "0010";
elsif (hcount > wall_l) and (hcount < (640 - wall_l)) and (vcount > wall_t) and (vcount < (wall_b + wall_l)) then
  s := "0001";
else
  s := "0000";
end if;
```

with select
```