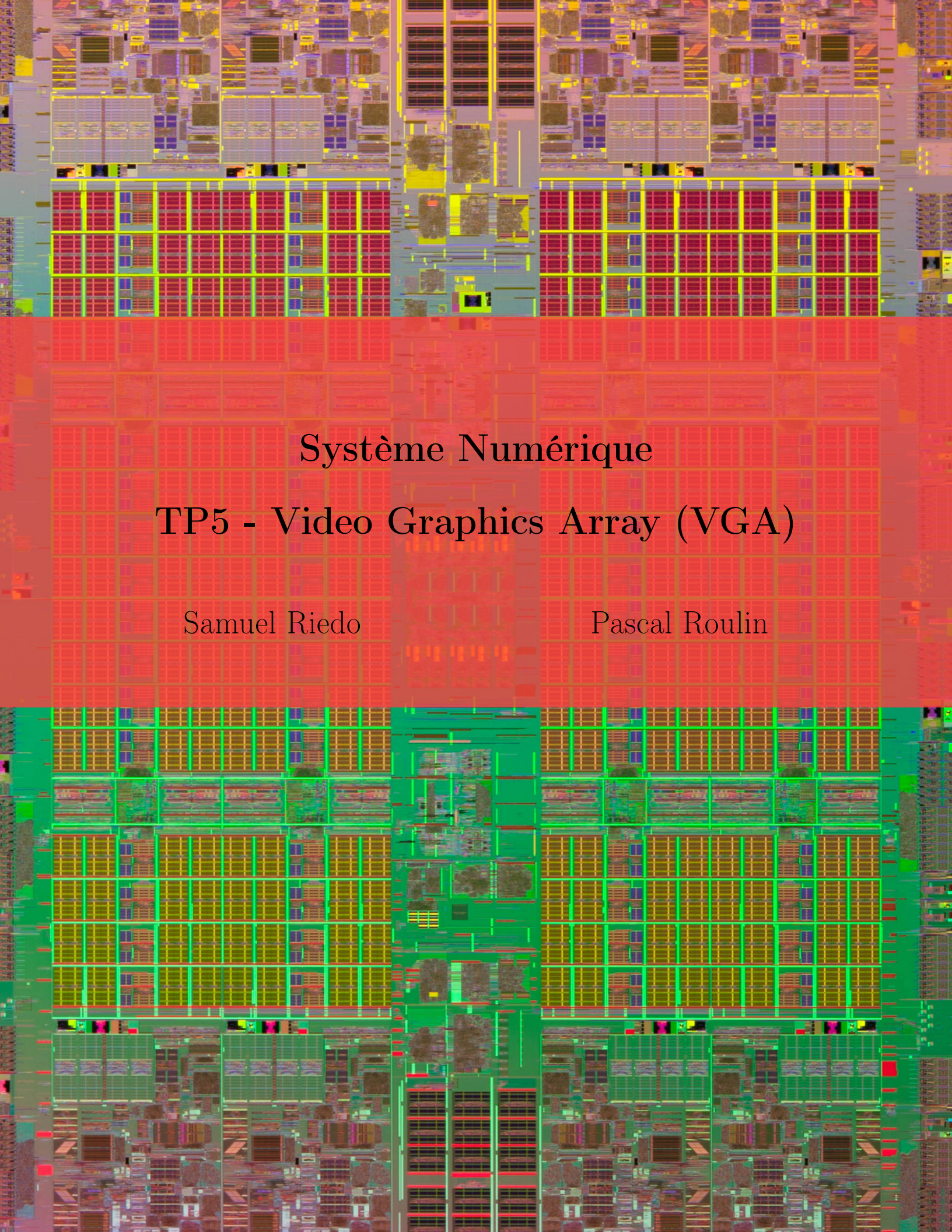

Système Numérique

TP5 - Video Graphics Array (VGA)

Samuel Riedo

Pascal Roulin

1 Introduction

VGA est un standard d'affichage développé par IBM en 1987. Il a été très largement adopté comme mode d'affichage pour PC en étant intégré sur les cartes mères des ordinateurs. Plusieurs sous-standards basés sur VGA ont été développés par après, tels que le SVGA utilisé dans ce TP qui permet une résolution supérieur à la norme originale. Il est aujourd'hui possible d'afficher des images en HD 1080p au moyen d'une connectique VGA.

2 Analyse

2.1 Fonctionnement

Une interface VGA fonctionne selon les trois composantes RGB ainsi qu'une synchronisation horizontale et verticale (figure 2.1). Les signaux RGB décrivent la couleur des pixels composant l'image selon un balayage effectué de gauche à droite, en ligne de haut en bas. L'écran recevant ce flux RGB est capable de savoir à quel pixel il correspond selon l'instant t auquel il lit ses données dans le balayage. Néanmoins, ce n'est pas l'écran qui est chargé de sauter automatiquement à la ligne suivante lorsque chaque pixel de la ligne actuel a été traité. C'est ce à quoi sert le signal *HS*, alors que *VS* indique un retour à la première ligne.

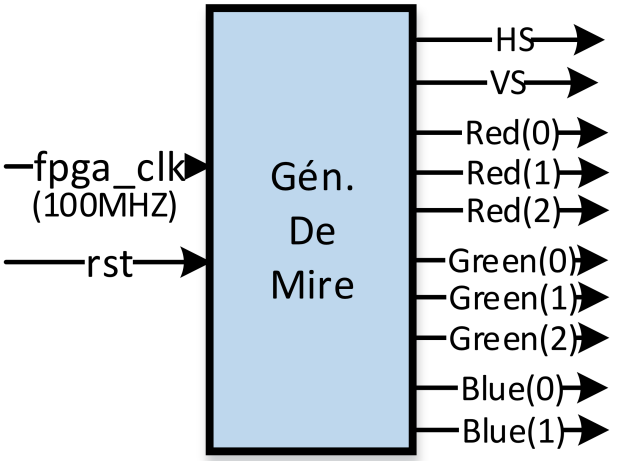

FIGURE 2.1 – Schéma bloc principal

En pratique, voici ce qu'il se passe : afin d'informer le monitor que le balayage est arrivé à la fin d'une ligne et que le prochain pixel sera le premier de la ligne suivante, le signal *HS* produit une impulsion de synchronisation. De même, lorsque le balayage est arrivé à la fin de la dernière ligne (et donc que toute une image a été transmise), le signal *VS* produit une impulsion pour indiquer un retour à la première ligne (et donc la transmission d'une nouvelle image).
Dans le cas de ce TP, le générateur de Mire (figure 2.1) sera divisé en trois sous-bocks :

- Digital Clock Management
- VGA Controller
- Circle & Rectangle Generator

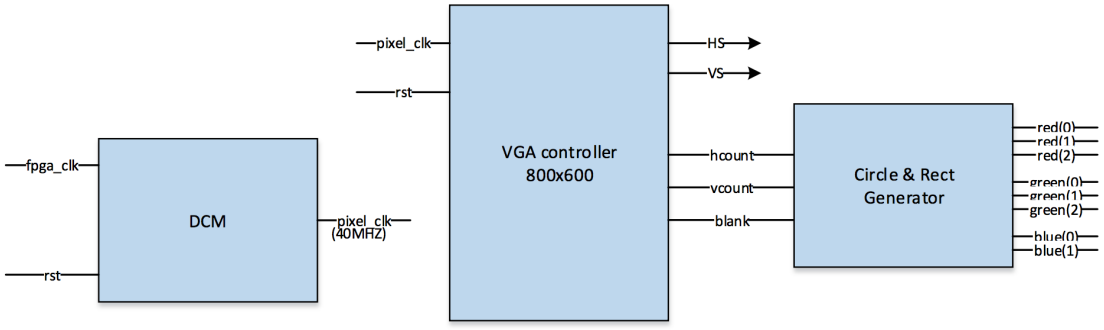

FIGURE 2.2 – Sous blocs du générateur de Mire

2.2 Digital Clock Management

La norme VGA utilise une fréquence de 40MHz pour le balayage de l'écran. Or, l'horloge intégrée à notre carte dispose d'une fréquence de 100MHz. Le bloc DCM servira simplement à créer une clock de 40MHz en sortie.
Ce type de montage étant très courant, il existe des outils pour générer un composant selon nos besoins. Le code à donc été généré en suivant la procédure de la consigne.

2.3 VGA Controller

L'écran recevant les signaux VGA a besoin des signaux de synchronisation horizontale et verticale (voir point 2.1). Ces derniers sont générés par le contrôleur VGA. Ce composant n'est pas chargé directement d'afficher des données à l'écran, mais il permet au moniteur et au générateur de rectangle/cercle de le faire.
Ce dernier composant reçoit lui, les coordonnées x et y (hcount & vcount) du pixel actuellement traité par le moniteur. De plus, lorsque la norme VGA fut établie, les écrans étaient de type CRT et le balayage du spot sur l'écran était réalisé par un oscillateur à relaxation. Ce dispositif impliquait que le spot s'éteigne arrivé en fin de ligne à droite, traverse l'écran pour se rendre au début de la ligne à gauche, puis reparte vers la droite. Pour laisser un temps suffisant à la réalisation de cette opération, un signal *blank* indique à l'écran que le balayage actuel est en dehors de la zone visible.

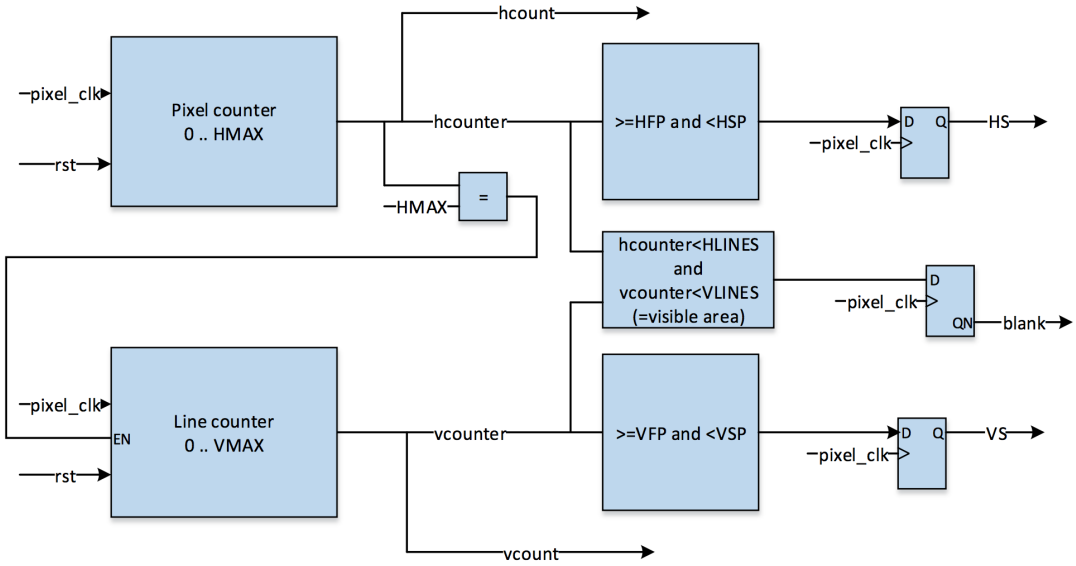

FIGURE 2.3 – Schéma bloc de VGA Controller

2.4 Circle & Rectangle Generator

Le composant *VGA Controller* ne fait que de fournir des signaux indispensables à l'affichage de donnée via VGA, mais sans jamais lui même envoyer des données utiles au moniteur. Afin de tester nos divers composants, le générateur de cercle et rectangle affiche un rectangle et un cercle tous les deux noirs sur un font blanc.

3 Codes

```
----------------------------------------------------------------------------------
-- Company: HES-SO
-- Engineer: Samuel Riedo & Pascal Roulin
-- Create Date:     09:20:02 03/02/2017
-- Design Name:     dcm.vhd
-- Project Name: Super Mario World - FPGA Edition
-- Target Devices: Digilent NEXYS 3 (Xilinx Spartan 6 XC6LX16-CS324)
-- Description: Input: 100MHz clock
--              Output: 40MHz clock
-- Revision 0.01 - File Created
--          1.00 - First functionnal version
----------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use ieee.numeric_std.all;

library unisim;
use unisim.vcomponents.all;

entity dcm is
  port
   (                                       -- Clock in ports
    CLK_IN1   : in  std_logic;
    -- Clock out ports
    CLK_OUT1 : out std_logic;
    -- Status and control signals
    RESET     : in  std_logic;
    LOCKED    : out std_logic
     );
end dcm;

architecture xilinx of dcm is
  attribute CORE_GENERATION_INFO            : string;
  attribute CORE_GENERATION_INFO of xilinx : architecture is "DCM,clk_wiz_v3_6,
    {component_name=DCM,use_phase_alignment=true,use_min_o_jitter=false,
     use_max_i_jitter=false,use_dyn_phase_shift=false,use_inclk_switchover=false,
     use_dyn_reconfig=false, feedback_source=FDBK_AUTO,primtype_sel=DCM_SP,
     num_out_clk=1,clkin1_period=10.0,clkin2_period=10.0,use_power_down=false,
     use_reset=true,use_locked=true,use_inclk_stopped=false,use_status=false,
     use_freeze=false,use_clk_valid=false,feedback_type=SINGLE,clock_mgr_type=AUTO,
     manual_override=false}";
  -- Input clock buffering / unused connectors
  signal clkin1                             : std_logic;
  -- Output clock buffering
  signal clkfb                              : std_logic;
  signal clk0                               : std_logic;
  signal clkfx                              : std_logic;
  signal clkfbout                           : std_logic;
  signal locked_internal                    : std_logic;
  signal status_internal                    : std_logic_vector(7 downto 0);
begin


  -- Input buffering
  -------------------------------------
  clkin1_buf : IBUFG
    port map
    (O => clkin1,
     I => CLK_IN1);


  -- Clocking primitive
  -------------------------------------
```

```
  -- Instantiation of the DCM primitive
  --    * Unused inputs are tied off
  --    * Unused outputs are labeled unused
  dcm_sp_inst : DCM_SP
    generic map
    (CLKDV_DIVIDE          => 2.500,
     CLKFX_DIVIDE          => 5,
     CLKFX_MULTIPLY        => 2,
     CLKIN_DIVIDE_BY_2     => false,
     CLKIN_PERIOD          => 10.0,
     CLKOUT_PHASE_SHIFT    => "NONE",
     CLK_FEEDBACK          => "1X",
     DESKEW_ADJUST         => "SYSTEM_SYNCHRONOUS",
     PHASE_SHIFT           => 0,
     STARTUP_WAIT          => false)
    port map
    -- Input clock
    (CLKIN     => clkin1,
     CLKFB     => clkfb,
     -- Output clocks
     CLK0      => clk0,
     CLK90     => open,
     CLK180    => open,
     CLK270    => open,
     CLK2X     => open,
     CLK2X180  => open,
     CLKFX     => clkfx,
     CLKFX180  => open,
     CLKDV     => open,
     -- Ports for dynamic phase shift
     PSCLK     => '0',
     PSEN      => '0',
     PSINCDEC  => '0',
     PSDONE    => open,
     -- Other control and status signals
     LOCKED    => locked_internal,
     STATUS    => status_internal,
     RST       => RESET,
     -- Unused pin, tie low
     DSSEN     => '0');

  LOCKED <= locked_internal;



  -- Output buffering
  -------------------------------------
  clkf_buf : BUFG
    port map
    (O => clkfb,
     I => clk0);


  clkout1_buf : BUFG
    port map
    (O => CLK_OUT1,
     I => clkfx);

end xilinx;
```

```
----------------------------------------------------------------------------------
-- Company: HES-SO
-- Engineer: Samuel Riedo & Pascal Roulin
-- Create Date:     09:20:02 03/02/2017
-- Design Name:     display.vhd
-- Project Name: Super Mario World - FPGA Edition
-- Target Devices: Digilent NEXYS 3 (Xilinx Spartan 6 XC6LX16-CS324)
-- Description: Print a rectangle and a circle, only used as a vga demo
-- Revision 0.01 - File Created
--          1.00 - First functionnal version
----------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library work;
use work.marioPackage.all;

entity display is
  port (
    blank   : in  std_logic;              -- If 1, video output must be null
    hcount : in  std_logic_vector(10 downto 0);  -- Pixel x coordinate
    vcount : in  std_logic_vector(10 downto 0);  -- Pixel y coordinate
    red     : out std_logic_vector(2 downto 0);   -- Red color output
    green   : out std_logic_vector(2 downto 0);   -- Green color output
    blue    : out std_logic_vector(1 downto 0));  -- Blue color output
end entity display;

architecture logic of display is

  constant xc      : integer := 100;                 -- circle x center
  constant yc      : integer := 100;                 -- circle y center
  constant r2      : integer := 100;                 -- circle rayon
  signal color     : std_logic_vector(7 downto 0); -- Color output
  signal vcounter : integer range 0 to VMAX;
  signal hcounter : integer range 0 to HMAX;

begin

  vcounter <= to_integer(unsigned(vcount));
  hcounter <= to_integer(unsigned(hcount));

  red   <= color(7 downto 5) when blank = '0' else "000";
  green <= color(4 downto 2) when blank = '0' else "000";
  blue  <= color(1 downto 0) when blank = '0' else "00";

  process(hcounter, vcounter)
    variable temp : integer;
  begin
  -- cercle
    color <= "11111111";
    temp  := (((hcounter-xc)*(hcounter-xc))+((vcounter-yc)*(vcounter-yc)));
-- equation du cercle
    if temp = r2 then
      color <= "00000000";
    end if;

    --rectangle
    if(((vcounter = 200) or (vcounter = 400)) and ((hcounter > 200) and (hcounter < 600)))
      or
      (((hcounter = 200) or (hcounter = 600)) and ((vcounter > 200) and (vcounter < 400)))
    then
      color <= "00000000";
    end if;
  end process;
end architecture;
```

```
----------------------------------------------------------------------------------
-- Company: HES-SO
-- Engineer: Samuel Riedo & Pascal Roulin
-- Create Date:    09:20:02 03/02/2017
-- Design Name: vga.vhd
-- Project Name: Super Mario World - FPGA Edition
-- Target Devices: Digilent NEXYS 3 (Xilinx Spartan 6 XC6LX16-CS324)
-- Description: Video Graphics Array
-- Revision 0.01 - File Created
--          1.00 - First functionnal version
----------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

use work.marioPackage.all;

entity vga is
  port (
    pixel_clk : in  std_logic;          -- 40MHz
    rst       : in  std_logic;          -- active high
    hs        : out std_logic;          -- Horizontale synchonization impulsion
    vs        : out std_logic;          -- Vertical synchonization impulsion
    blank     : out std_logic;          -- If 1, video output must be null
    hcount    : out std_logic_vector(10 downto 0);  -- Pixel x coordinate
    vcount    : out std_logic_vector(10 downto 0));  -- Pixel y coordinate
end entity vga;

architecture logic of vga is

  signal hcounter : integer range 0 to HMAX;
  signal vcounter : integer range 0 to VMAX;
  signal en       : std_logic;

begin

  hcount <= std_logic_vector(to_unsigned(hcounter, 11));
  vcount <= std_logic_vector(to_unsigned(vcounter, 11));
  en     <= '1' when (hcounter = HMAX) else '0';

  -- Processus: Pixel Counter
  process(rst, pixel_clk)
  begin
    if rst = '1' then
      hcounter <= 0;
    elsif rising_edge(pixel_clk) then
     if hcounter = HMAX then
      hcounter <= 0;
    else
      hcounter <= hcounter + 1;
    end if;
    end if;
  end process;

-- Processus: Line Counter
  process(rst, pixel_clk, en)
  begin
      if rst = '1' then
        vcounter <= 0;
      elsif rising_edge(pixel_clk) then
    if en = '1' then
      if vcounter = VMAX then
      vcounter <= 0;
      else
        vcounter <= vcounter + 1;
      end if;
      end if;
```

```
    end if;
  end process;

  process (pixel_clk, rst) is
  begin
    if rst = '1' then                   -- asynchronous reset (active low)
      hs <= '0';
    elsif rising_edge(pixel_clk) then   -- rising clock edge
      if (hcounter >= HFP and hcounter < HSP) then
        hs <= '1';
      else
        hs <= '0';
      end if;
    end if;
  end process;

  process (pixel_clk, rst) is
  begin
    if rst = '1' then                   -- asynchronous reset (active low)
      vs <= '0';
    elsif rising_edge(pixel_clk) then   -- rising clock edge
      if (vcounter >= VFP and vcounter < VSP) then
        vs <= '1';
      else
        vs <= '0';
      end if;
    end if;
  end process;

  process (pixel_clk, rst) is
  begin
    if rst = '1' then                   -- asynchronous reset (active low)
      blank <= '1';
    elsif rising_edge(pixel_clk) then   -- rising clock edge
      if (hcounter < HLINES and vcounter < VLINES) then
        blank <= '0';
      else
        blank <= '1';
      end if;
    end if;
  end process;

end architecture logic;
```

```
----------------------------------------------------------------------------------
-- Company: HES-SO
-- Engineer: Samuel Riedo & Pascal Roulin
-- Create Date:    09:20:02 03/02/2017
-- Design Name: vgaDisplayTopModule.vhd
-- Project Name: Super Mario World - FPGA Edition
-- Target Devices: Digilent NEXYS 3 (Xilinx Spartan 6 XC6LX16-CS324)
-- Description: Top module for vga demo
-- Revision 0.01 - File Created
--          1.00 - First functionnal version
----------------------------------------------------------------------------------
library IEEE;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity vgagenerator is
  port (
    fpga_clk : in  std_logic;
    rst      : in  std_logic;
    HS       : out std_logic;
    VS       : out std_logic;
    red      : out std_logic_vector(2 downto 0);
    green    : out std_logic_vector(2 downto 0);
    blue     : out std_logic_vector(1 downto 0));
end entity;  -- vgagenerator

architecture behav of vgagenerator is
  signal pixel_clk : std_logic;
  signal hcount    : std_logic_vector(10 downto 0);
  signal vcount    : std_logic_vector(10 downto 0);
  signal blank     : std_logic;
  signal locked    : std_logic;

  component display is
    port (
      hcount : in  std_logic_vector(10 downto 0);  -- Pixel x coordinate
      vcount : in  std_logic_vector(10 downto 0);  -- Pixel y coordinate
      blank  : in  std_logic;                      -- If 1, video output must be null
      red    : out std_logic_vector(2 downto 0);   -- Red color output
      green  : out std_logic_vector(2 downto 0);   -- Green color output
      blue   : out std_logic_vector(1 downto 0));  -- Blue color output
  end component;

  component dcm is
    port
     (                                    -- Clock in ports
      CLK_IN1  : in  std_logic;
      -- Clock out ports
      CLK_OUT1 : out std_logic;
      -- Status and control signals
      RESET    : in  std_logic;
      LOCKED   : out std_logic
      );
  end component;

  component vga is
    port (
      pixel_clk : in  std_logic;         -- 40MHz
      rst       : in  std_logic;         -- active low
      hs        : out std_logic;         -- Horizontale synchonization impulsion
      vs        : out std_logic;         -- Vertical synchonization impulsion
      blank     : out std_logic;         -- If 1,  video output must be null
      hcount    : out std_logic_vector(10 downto 0);   -- Pixel x coordinate
      vcount    : out std_logic_vector(10 downto 0));  -- Pixel y coordinate
  end component;
begin
```

```
  dcm_map : dcm
    port map
    (
      CLK_IN1  => fpga_clk,
      CLK_OUT1 => pixel_clk,
      RESET    => rst,
      LOCKED   => LOCKED);

  vga_map : vga
    port map (
      pixel_clk => pixel_clk,
      rst       => rst,
      hs        => hs,
      vs        => vs,
      blank     => blank,
      hcount    => hcount,
      vcount    => vcount);

  display_map : display
    port map(
      blank  => blank,
      hcount => hcount,
      vcount => vcount,
      red    => red,
      green  => green,
      blue   => blue);

end architecture;
```

```
----------------------------------------------------------------------------------
-- Company: HES-SO
-- Engineer: Samuel Riedo & Pascal Roulin
-- Create Date:     09:20:02 03/02/2017
-- Design Name:     marioPackage.vhd
-- Project Name: Super Mario World - FPGA Edition
-- Target Devices: Digilent NEXYS 3 (Xilinx Spartan 6 XC6LX16-CS324)
-- Description: All constants for Super Mario World - FPGA Edition project
-- Revision 0.01 - File Created
--          0.01 - vga controller constants added
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

package marioPackage is
  -- vga constants
  constant HMAX   : integer := 1056;
  constant VMAX   : integer := 628;
  constant HLINES : integer := 800;
  constant VLINES : integer := 600;
  constant HSP    : integer := 968;
  constant HFP    : integer := 840;
  constant VFP    : integer := 601;
  constant VSP    : integer := 605;
  -- file constants
end marioPackage;
```

Fribourg, le 15 mars 2017.

Samuel Riedo

Pascal Roulin