TP3 Integration of the VGA Controller

1) At the hardware level :

Nous créons d'abord un nouveau périphérique qui gèrera la sortie vidéo vga. Ce périphérique « ahblite_vga » contient à l'intérieur tous les composants nécessaires pour la gestion de cette tâche :

```
13  ----------------------------------------------------
14  entity ahblite_vga is
15  port (
16      HRESETn     : in  std_logic;
17      HCLK        : in  std_logic;
18
19      HSEL        : in  std_logic;
20      HREADY      : in  std_logic;
21
22      btnR : in  STD_LOGIC;
23      Hsync : out  STD_LOGIC;
24      Vsync : out  STD_LOGIC;
25      vgaRed : out  STD_LOGIC_VECTOR (3 downto 0);
26      vgaGreen : out  STD_LOGIC_VECTOR (3 downto 0);
27      vgaBlue : out  STD_LOGIC_VECTOR (3 downto 0);
28
29      -- AHB-Lite interface
30      AHBLITE_IN  : in  AHBLite_master_vector;
31      AHBLITE_OUT : out AHBLite_slave_vector);
32  end;
33
34
```

Ensuite, nous rajoutons un registre background dans notre périphérique « vga » qui nous permettra de contrôler par le software la couleur du fond d'écran dans la suite du TP :

```
111             Background <= (others => '0');
112             --Reg2 <= (others => '0');
113             --Reg3 <= (others => '0');
114             --Reg4 <= (others => '0');
115
116         ---------------------------------
117         elsif rising_edge(HCLK) then
118             -- Error management
119             SlaveOut.HREADYOUT <= not invalid;
120             SlaveOut.HRESP <= invalid or not SlaveOut.HREADYOUT;
121
122             -- Performe write if requested last cycle and no error occured
123             if SlaveOut.HRESP = '0' and lastwr = '1' then
124                 case lastaddr is
125                     when x"00" => Background    <= SlaveIn.HWDATA;
126                     --when x"01" => Reg2    <= SlaveIn.HWDATA;
127                     --when x"02" => Reg3    <= SlaveIn.HWDATA;
128                     --when x"03" => Reg4    <= SlaveIn.HWDATA;
129                     when others =>
130                 end case;
131             end if;
132
133             -- Check for transfer
134             if transfer = '1' and invalid = '0' then
135                 -- Read operation: retrieve data and fill empty spaces with '0'
136                 if SlaveIn.HWRITE = '0' then
137                     SlaveOut.HRDATA <= (others => '0');
138                     case address is
139                         when x"00" => SlaveOut.HRDATA    <= Background;
140                         --when x"01" => SlaveOut.HRDATA    <= Reg2;
```

Une fois que nous avons fini de remonter les signaux venant du périphérique jusqu'au top, nous modifions le fichier xdc pour qu'il soit en accord avec la nomenclature que nous avons donné aux signaux du périphérique vga.

```
set_property PACKAGE_PIN G19 [get_ports {vgaRed[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[0]}]
set_property PACKAGE_PIN H19 [get_ports {vgaRed[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[1]}]
set_property PACKAGE_PIN J19 [get_ports {vgaRed[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[2]}]
set_property PACKAGE_PIN N19 [get_ports {vgaRed[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[3]}]
```

Et en fin nous mettons une adresse dans le fichier decoder_mcu.vhd :

```
when x"24" & "00" => sel <= CID_VGA;
```

2) At the software level :

Nous declarons arch_vga.h :

td-icobs-light-project-master > lib > arch > C arch_vga.h > ...

```
#ifndef __ARCH_VGA_H__
#define __ARCH_VGA_H__

typedef struct
{
    volatile unsigned int background;
}VGA_t;

#endif
```

Et dans arch.h :

```
//VGA
#define VGA                    (*(VGA_t*)VGA_BASE)
```

Puis, une fois les modifications réalisées, il ne nous reste plus qu'a relier dans le software notre registre background au périphérique GPIOA qui est chargé de gérer les switches dans la carte :

```
while (1)
{
    VGA.background = GPIOA.IDR;
```

3) Sprite control

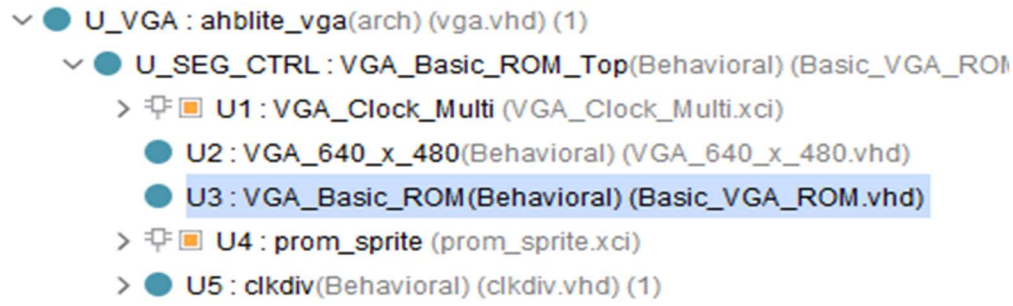
Dans cette partie nous rajoutons encore deux registres qui nous servirons à relier la position de notre Sprite aux boutons btn de notre carte en passant par le software :

```
55      signal Background    : std_logic_vector(31 downto 0);
56      signal X1_Position    : std_logic_vector(31 downto 0);
57      signal Y1_Position    : std_logic_vector(31 downto 0);
58      --signal Reg4    : std_logic_vector(31 downto 0);
```

Ensuite nous rajoutons les composants nous permettant d'introduire et de gérer notre Sprite dans la rom de notre cpu :

```
64  component VGA_Basic_ROM_Top is
65      Port ( clk : in  STD_LOGIC;
66             btnR : in  STD_LOGIC;
67             Hsync : out  STD_LOGIC;
68             Vsync : out  STD_LOGIC;
69             sw: in  STD_LOGIC_VECTOR (11 downto 0);
70             X1_Pos : in std_logic_vector(9 downto 0);
71             Y1_Pos : in std_logic_vector(9 downto 0);
72             vgaRed : out  STD_LOGIC_VECTOR (3 downto 0);
73             vgaGreen : out  STD_LOGIC_VECTOR (3 downto 0);
74             vgaBlue : out  STD_LOGIC_VECTOR (3 downto 0));
75  end component VGA_Basic_ROM_Top;
76  ----------------------------------------------------------
77  begin
78
79      test <= "0000000000";
80      U_SEG_CTRL: VGA_Basic_ROM_Top
81          Port map (
82              clk => HCLK,
83              btnR => RST,
84              Hsync => Hsync,
85              Vsync => Vsync,
86              --X1_Pos => test,
87              --Y1_Pos => test,
88              X1_Pos => X1_Position(9 downto 0),
89              Y1_Pos => Y1_Position(9 downto 0),
90              sw => Background (11 downto 0),
91              vgaRed => vgaRed,
```

Une fois que nous avons rajouté tous les fichiers, notre arborescence ressemblera à ça :



Et finalement dans basic_rom nous mettons la valeur des registres à C et R, qui gèrent l'affichage du Sprite dans l'écran :

```
            rom_addr4: out std_logic_vector(15 downto 0); --j'ai modifié ça pour rectangle, rechanger à 3 pour revenir en arriere
            M: std_logic_vector(11 downto 0);
            red : out STD_LOGIC_VECTOR (3 downto 0);
            green : out STD_LOGIC_VECTOR (3 downto 0);
            blue : out STD_LOGIC_VECTOR (3 downto 0);
            C: in STD_LOGIC_VECTOR (9 downto 0);
            R: in STD_LOGIC_VECTOR (9 downto 0));
end VGA_Basic_ROM;

architecture Behavioral of VGA_Basic_ROM is
signal spriteon: STD_LOGIC;
constant hbp: unsigned(9 downto 0)      := "0010010000"; -- horizontal back porch = 128 + 16 = 144 ou 96 + 48
constant vbp: unsigned(9 downto 0)      := "0000011111"; -- vertical back porch = 2 + 29 = 31

constant w: unsigned(9 downto 0)        := to_unsigned(240, 10); --largeur du sprit adaptée
constant h: unsigned(9 downto 0)        := to_unsigned(160, 10); --meme hauteur que l'image d'origine


signal xpix, ypix, R1, C1: unsigned(9 downto 0);
signal rom_addr_s: std_logic_vector(19 downto 0);

begin

--R1 <= unsigned(sw(11 downto 6) & "0000"); --ligne désirée pour notre image
--C1 <= unsigned(sw(5 downto 0) & "0000"); --colonne désirée
R1<=unsigned(R);--TP3 RESTE A METTRE SA DANS L4ORDRE
C1<=unsigned(C);
xpix <= unsigned(hc) - (hbp + C1);
ypix <= unsigned(vc) - (vbp + R1);
```

Puis dans le software nous rajoutons les registres comme précédemment :

demo-icobs-light-project > lib > arch > C arch_vga.h > ...

```
#ifndef __ARCH_VGA_H__
#define __ARCH_VGA_H__

typedef struct
{
    volatile unsigned int background;
    volatile unsigned int X1_Pos;
    volatile unsigned int Y1_Pos;
}VGA_t;

#endif
```

4) Final step

Then finally, we create a code that allow us to change the background color as well as making mouve the Sprite through the buttoms of the card :

```
81      while (1)
82      {
83          VGA.background = GPIOA.IDR;
84          if (count<3000)
85          {
86              count++;
87          }else count =0;
88          if (ct_print<300000)
89          {
90              ct_print++;
91          }else
92          {
93              ct_print =0;
94              myprintf("valeur de x1_pos %d\n",VGA.X1_Pos);
95              myprintf("valeur de y1_pos %d\n",VGA.Y1_Pos);
96          }
97
98
99          VGA.background = GPIOA.IDR;
100
101         if ((VGA.X1_Pos< SCREEN_WIDTH - w) && (VGA.X1_Pos>=0))
102         {
103             VGA.X1_Pos += dx;
104         }
105         if ((VGA.Y1_Pos>=0) && (VGA.Y1_Pos< SCREEN_HEIGHT - h))
106         {
107             VGA.Y1_Pos += dy;
108         }
109
110
111         dx=0;
112         dy=0;
```

```
111         dx=0;
112         dy=0;
113         do{
114
115             if (BTNU && count==0) //upper
116                 {
117                     //dx=0;
118                     dy=-vitesse;
119                 }
120             if (BTND && count==0) //down
121                 {
122                     //dx=0;
123                     dy=vitesse;
124                 }
125             if (BTNL && count==0)  //left
126                 {
127                     dx=-vitesse;
128                     //dy=0;
129                 }
130             if (BTNR && count==0)  //right
131                 {
132                     dx=vitesse;
133                     //dy=0;
134                 }
135
136         }while(!TIMER_FLAG);
137     }
138     return 0;
139 }
```