



Implantation d'un pic 16C57 dans un FPGA
Xilins spartan 3

Enseignant: Mr Moutou

Table des matières

Objectifs et présentation du projet.....	3
Définitions.....	4
FPGA XILINX SPARTAN 3.....	5
Carte d'expérimentation Spartan 3	5
PIC 16C57.....	6
Un PIC dans un FPGA !? Il est fou !.....	7
Synoptique.....	7
Description VHDL du projet	8
PIC 16C57.....	9
RAM	11
ROM	12
VGATOP.....	13
Entité et déclaration des composants.....	14
Composant Rectangle.....	16
Composant VGA_SYNC.....	16
Fichier global TopToplogic.....	18
Implantation logicielle.....	21
Améliorations possibles.....	22
Conclusion.....	22

GEII IUT de Troyes	DOSSIER Projet encadré	Session 2009
-----------------------	---	--------------

Objectifs et présentation du projet

Le projet consiste à implanter l'architecture d'un pic 16C57 de Microchip dans un FPGA Spartan 3 Xilinx. La description VHDL du pic et sa documentation ont été préalablement récupérés sur internet. Notre travail a été de l'adapter à notre application, le but étant de pouvoir faire tourner des programmes en C sur le FPGA.

Le projet s'est déroulé en plusieurs étapes:

- Récupération de la description du Pic et de la documentation
- Implantation de celui ci dans le FPGA à l'aide de Xilinx
- Interfaçage de la mémoire
- Modifications et création de nouveaux composants VHDL pour le débogage et le test du pic
- Création de nouveaux composants VHDL pour la gestion d'un écran VGA
- Développement en C sur MPLAB d'une application type « pong »

GEII IUT de Troyes	DOSSIER Projet encadré	Session 2009
-----------------------	---	--------------

Définitions

→ VHDL

VHDL est un langage de description matériel destiné à représenter le comportement ainsi que l'architecture d'un système électronique numérique. L'intérêt d'une telle description réside dans son caractère exécutable : une spécification décrite en VHDL peut être vérifiée par simulation, avant que la conception détaillée ne soit terminée. En outre, les outils de conception assistée par ordinateur permettant de passer directement d'une description fonctionnelle en VHDL à un schéma en porte logique ont révolutionné les méthodes de conception des circuits numériques, ASIC ou FPGA.

→ LANGAGE C

C est un langage de programmation impératif conçu pour la programmation système. Inventé au début des années 1970 avec UNIX, C est devenu un des langages les plus utilisés. De nombreux langages plus modernes comme C++, Java et PHP reprennent des aspects de C.

→ FPGA

Un circuit logique programmable, ou réseau logique programmable, est un circuit intégré logique qui peut être reprogrammé après sa fabrication. Il est composé de nombreuses cellules logiques élémentaires librement assemblables.

→ XILINX

Xilinx (nom complet Xilinx, Inc.) est une entreprise américaine de semi-conducteurs. Inventeur du FPGA, Xilinx fait partie des plus grandes entreprises spécialisées dans le développement et la commercialisation de composants logiques programmables, et des services associés tels que les logiciels de CAO électroniques ; blocs de propriété intellectuelle réutilisables et formation. Il s'agit aussi du logiciel de développement.

→ Microcontrôleur PIC

Les micro-contrôleurs PIC (ou PICmicro dans la terminologie du fabricant) forment une famille de micro-contrôleurs de la société Microchip. Ces micro-contrôleurs sont dérivés du PIC1650 développé à l'origine par la division microélectronique de General Instruments. Le nom PIC n'est pas officiellement un acronyme, bien que la traduction en « Peripheral Interface Controller » (contrôleur d'interface périphérique) soit généralement admise. Cependant, à l'époque du développement du PIC1650 par General Instruments, PIC était un acronyme de « Programmable Intelligent Computer » ou « Programmable Integrated Circuit ».

→ MPLAB

Environnement de développement intégré pour PIC.

FPGA XILINX SPARTAN 3

Le FPGA Spartan 3 possède 200000 portes logiques assemblables à souhait à l'aide du vhdl. Il dispose d'environ 170 entrées-sorties et fonctionne sous une tension de 1,5V.



*FPGA spartan 3 de
marque XILINX*

Carte d'expérimentation Spartan 3

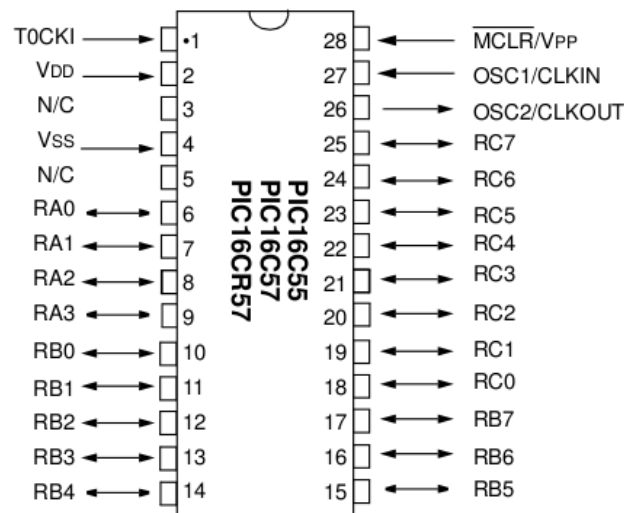
Nous avons utilisé pour le projet la carte d'expérimentation XILINX Spartan 3. Elle embarque bien sur le fpga Spartan 3. Disposant d'un grand nombre d'entrées-sorties accessible, elle dispose aussi d'un port VGA, d'un port série, d'un port usb, d'interrupteurs, de diodes ainsi qu'un afficheur 7 segments. Nous utiliserons dans notre application le port VGA pour gérer un affichage sur écran. Cette carte se programme via le port parallèle de l'ordinateur a l'aide de l'environnement de développement intégré de XILINX (XILINX ISE).



Le carte d'expérimentation

PIC 16C57

Le pic 16C57 de MICROCHIP est un petit micro-contrôleur de 33 instructions qui embarque dans sa version boitier 2000 octets de rom et seulement 72 octets de ram. Il dispose de trois ports A, B et C ainsi que d'un timer TMR0.



PIC 16C57 dans sa version boitier

Un PIC dans un FPGA !? Il est fou !

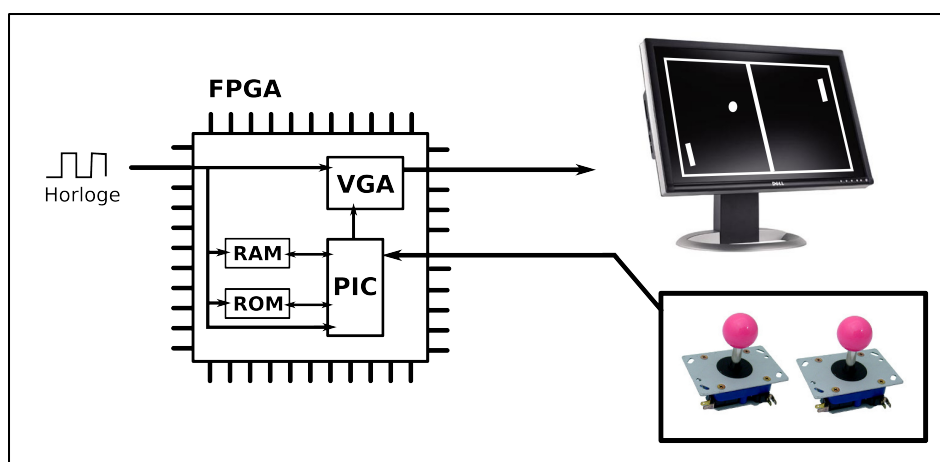
Il ne s'agit pas vous vous en doutez de faire rentrer un pic de force dans un malheureux fpga. Il s'agit de copier le façon dont est faite un PIC et de le programmer dans un FPGA. On pourra ensuite programmer et utiliser le FPGA de la même façon qu'un PIC. En effet un PIC n'est qu'un assemblage assez complexe de portes logiques mis dans un boitier. Un FPGA est quant-a-lui un composant rempli de portes logiques en attente d'être reliées. Le VHDL pouvant être vu comme un langage de description de schémas logiques il est aisé à comprendre que si on dispose de la description VHDL du pic, il est possible de le programmer dans un FPGA

« Oui mais pourquoi transformer un FPGA en PIC alors qu'un FPGA coûte beaucoup plus cher que le PIC en question ? »

C'est une bonne question... En fait, le PIC va prendre seulement une petite partie du FPGA, ce qui va laisser de la place pour y mettre d'autres composants. On peut par exemple y mettre de la mémoire supplémentaire pour étendre les possibilités du pic, ou encore rajouter des ports d'entrées-sorties. On peut même imaginer de mettre plusieurs PIC dans le même FPGA. Toute fonction réalisée par ces composants annexes est une tâche en moins pour le PIC. De plus, une fois celui-ci intégré, on peut programmer notre application en langage C.

Synoptique

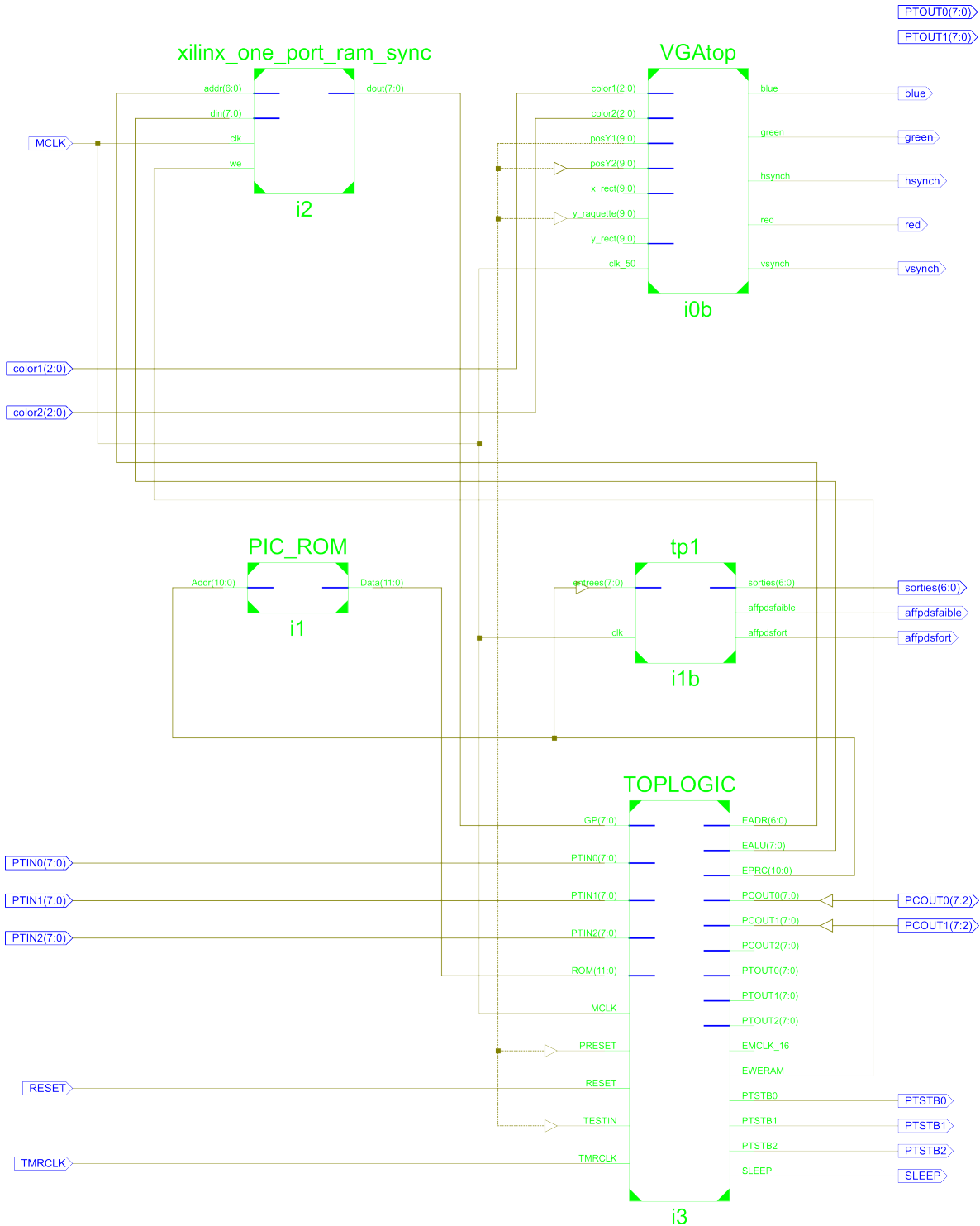
Dans notre cas le projet VHDL sera constitué de 4 composants principaux : le PIC, de la mémoire Ram, de la mémoire Rom et d'un composant qui va gérer l'affichage sur un écran VGA.



Synoptique

Description VHDL du projet

Le projet VHDL global est constitué de 5 composants, le tout rassemblé grace au fichier TopToplogic. Tp1 est un composant de test, il ne sera pas décrit ici .

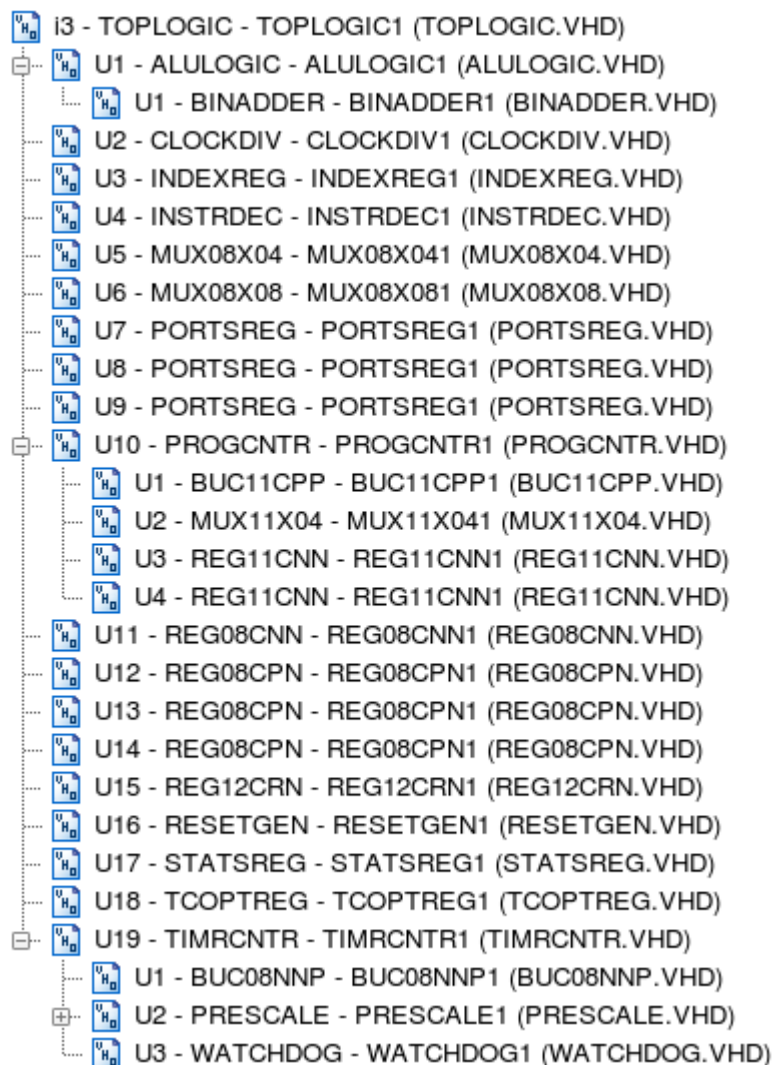


Architecture et composants de TopTopLogic

PIC 16C57

La description VHDL du pic 16C57 a été récupérée sur internet. Le programme initial s'articule autour du fichier TOPLOGIC . Il utilise différents composants décrits dans la documentation (nous ne les détaillerons pas ici). Voici la liste des ces composants qui composent le microcontrôleur et de leur architecture:

Architecture



Entité

entity TOPLOGIC **is**

```

port (
  EADR:    out std_logic_vector( 6 downto 0 );
  EALU:    out std_logic_vector( 7 downto 0 );
  EMCLK_16: out std_logic;
  EPRC:    out std_logic_vector( 10 downto 0 );
  EWERAM:  out std_logic;
  GP:      in  std_logic_vector( 7 downto 0 );
  MCLK:    in  std_logic;
  PCOUT0:  out std_logic_vector( 7 downto 0 );
  PCOUT1:  out std_logic_vector( 7 downto 0 );
  PCOUT2:  out std_logic_vector( 7 downto 0 );
  PTIN0:   in  std_logic_vector( 7 downto 0 );
  PTIN1:   in  std_logic_vector( 7 downto 0 );
  PTIN2:   in  std_logic_vector( 7 downto 0 );
  PTOUT0:  out std_logic_vector( 7 downto 0 );
  PTOUT1:  out std_logic_vector( 7 downto 0 );
  PTOUT2:  out std_logic_vector( 7 downto 0 );
  PTSTB0:  out std_logic;
  PTSTB1:  out std_logic;
  PTSTB2:  out std_logic;
  PRESET:  in  std_logic;
  RESET:   in  std_logic;
  ROM:     in  std_logic_vector( 11 downto 0 );
  SLEEP:   out std_logic;
  TESTIN:  in  std_logic;
  TMRCLK:  in  std_logic
);

```

end entity TOPLOGIC;

RAM

La façon dont les mémoires sont implantées dans les FPGA dépendent énormément du constructeur. Le plus souvent, des espaces dédiés contenant de la véritable mémoire sont définis dans ceux-ci. Pour une question de portabilité, les personnes qui sont à l'origine du projet VHDL ne les ont donc pas incluses dans le programme. Cependant ils fournissent des exemple pour différents constructeurs dont Xilinx. On s'est donc inspirés de cet exemple pour interfacer notre RAM.

Description VHDL de la RAM

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity xilinx_one_port_ram_sync is
  generic(
    ADDR_WIDTH: integer:=7;
    DATA_WIDTH: integer:=8
  );
  port(
    clk: in std_logic;
    we: in std_logic;
    addr: in std_logic_vector(ADDR_WIDTH-1 downto 0);
    din: in std_logic_vector(DATA_WIDTH-1 downto 0);
    dout: out std_logic_vector(DATA_WIDTH-1 downto 0)
  );
end xilinx_one_port_ram_sync;

architecture beh_arch of xilinx_one_port_ram_sync is
  type ram_type is array (2**ADDR_WIDTH-1 downto 0)
    of std_logic_vector (DATA_WIDTH-1 downto 0);
  signal ram: ram_type;
  --signal addr_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))) <= din;
        end if;
        --addr_reg <= addr;
        end if;
    end process;
    --dout <= ram(to_integer(unsigned(addr_reg)));
    dout <= ram(to_integer(unsigned(addr)));
  end beh_arch;
```

ROM

La ROM contient le programme du micro-contrôleur. On utilise un programme qui convertit les fichiers hexadécimaux directement en VHDL. Cela permet de développer le programme du μC en C et de l'intégrer dans le VHDL du projet. Ce logiciel génère des fichiers VHDL de ce type :

Description VHDL de la rom (exemple)

```
library IEEE;
use IEEE.std_logic_1164.all;
--use IEEE.std_logic_arith.all;
use IEEE.numeric_std.all;

entity PIC_ROM is
  port (
    Addr  : in  std_logic_vector(10 downto 0);
    Data  : out std_logic_vector(11 downto 0));
end PIC_ROM;

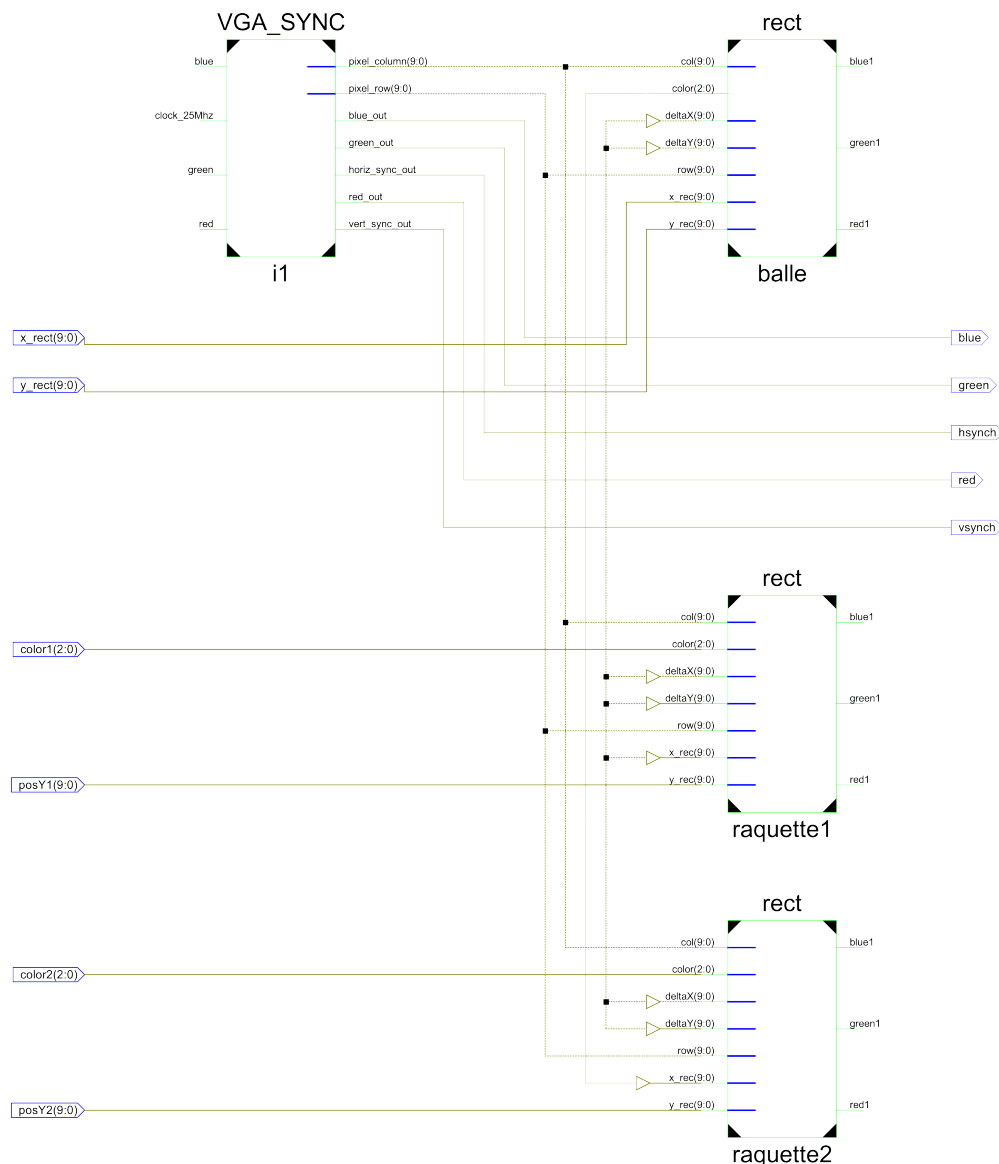
architecture first of PIC_ROM is
begin
  Data <=
    "000000100101" When to_integer(unsigned(Addr)) = 0000 Else
    "000001100100" When to_integer(unsigned(Addr)) = 0001 Else
    "010111100100" When to_integer(unsigned(Addr)) = 0002 Else
    "110000001010" When to_integer(unsigned(Addr)) = 0003 Else
    "000000100100" When to_integer(unsigned(Addr)) = 0004 Else
    "110000001100" When to_integer(unsigned(Addr)) = 0005 Else
    :
    :
    :
    "101000111110" When to_integer(unsigned(Addr)) = 1020 Else
    "010011000011" When to_integer(unsigned(Addr)) = 1021 Else
    "010010100011" When to_integer(unsigned(Addr)) = 1022 Else
    "101000000010" When to_integer(unsigned(Addr)) = 1023 Else
    "000000000000";
end first;
```

VGATOP

Le composant VGATOP va s'occuper de l'affichage sur écran VGA. Il gère la synchronisation avec l'écran et affiche des carrés sur celui-ci. Leur coordonnées, leur taille ainsi que leur couleur peuvent être soit :

- Fixées directement dans le composant
- Définies à l'aide d'une entrée sur le composant

La couleur des raquettes est modifiable grâce aux interrupteurs de la carte. Leur position en Y est pilotée par le micro-contrôleur et leur position en X reste fixe. Les coordonnées en X et en Y de la balle sont gérées par le micro-contrôleur. Les valeurs des autres rectangles (contours, filet...) restent fixes. VGATOP est constitué de 2 composants : VGA_SYNC et RECT.



Architecture et composants de VGATOP

Entité et déclaration des composants

```

Architecture VHDL de VGATOP
library IEEE;
use IEEE.STD_LOGIC_1164.all;

ENTITY VGATop IS
  PORT (clk_50 : in STD_LOGIC;
          x_rect, y_rect, y_raquette: IN STD_LOGIC_VECTOR(9 DOWNTO 0);
          color1, color2 :IN STD_LOGIC_VECTOR(2 DOWNTO 0);
          posY1, posY2 :IN STD_LOGIC_VECTOR(9 DOWNTO 0);
          hsynch,vsynch,red,green,blue : out STD_LOGIC);
END VGATop;

ARCHITECTURE atop of VGATop is

COMPONENT VGA_SYNC IS
  PORT( clock_25Mhz, red, green, blue           : IN      STD_LOGIC;
          red_out, green_out, blue_out, horiz_sync_out, vert_sync_out : OUT   STD_LOGIC;
          pixel_row, pixel_column: OUT STD_LOGIC_VECTOR(9 DOWNTO 0));
END COMPONENT;

COMPONENT rect IS PORT(
  row,col,x_rec,y_rec :in STD_LOGIC_VECTOR(9 DOWNTO 0);
  deltaX,deltaY : in std_logic_vector(9 DOWNTO 0);
  color: in std_logic_vector(2 DOWNTO 0);
  red1,green1,blue1 : out std_logic);
END component;

signal clk_25,sred,sgreen,sblue,
          sred1,sgreen1,sblue1,
          sred2,sgreen2,sblue2,
          sred3,sgreen3,sblue3,
          sred4,sgreen4,sblue4,
          sred5,sgreen5,sblue5,
          sred6,sgreen6,sblue6,
          sred7,sgreen7,sblue7,
          sred8,sgreen8,sblue8 : std_logic;
signal srow,scol : STD_LOGIC_VECTOR(9 DOWNTO 0);

begin

sred <= sred1 or sred2 or sred3 or sred4 or sred5 or sred6 or sred7 or sred8;
sgreen <= sgreen1 or sgreen2 or sgreen3 or sgreen4 or sgreen5 or sgreen6 or sgreen7 or sgreen8;
sblue <= sblue1 or sblue2 or sblue3 or sblue4 or sblue5 or sblue6 or sblue7 or sblue8;

process(clk_50) begin
  if clk_50'event and clk_50='1' then
    clk_25 <= not clk_25;
  end if;
end process;
i1:vga_sync port map(clock_25Mhz =>clk_25,red_out=>red,green_out=>green,blue_out=>blue,
                    horiz_sync_out=>hsynch,vert_sync_out=>vsynch,red=>sred,green=>sgreen,
                    blue=>sblue,pixel_row=>srow,pixel_column=>scol);

-- Pour modifier les dimensions de la balle et de la raquette : modifier deltaX et deltaY
-- dans les port map
-- Pour modifier les couleurs : modifier le mot color

balle:rect port map(row=>srow,col=>scol,red1=>sred1,green1=>sgreen1,blue1=>sblue1,
                    x_rec => x_rect,
                    y_rec => y_rect,

```

```

                                deltaX =>"0000010000",
                                deltaY=> "0000010000",
                                color => "111");

raquette1:rect port map(row=>srow,col=>scol,red1=>sred2,green1=>sgreen2,blue1=>sblue2,
                        x_rec => "0000010110",
                        y_rec => posY1, -- pilotage
                        deltaX =>"0000010000",
                        deltaY=> "0001000000",
                        color => color1);

raquette2:rect port map(row=>srow,col=>scol,red1=>sred3,green1=>sgreen3,blue1=>sblue3,
                        x_rec => "1001011010",
                        y_rec => posY2, -- pilotage
                        deltaX =>"0000010000",
                        deltaY=> "0001000000",
                        color => color2);

filet:rect port map(row=>srow,col=>scol,red1=>sred4,green1=>sgreen4,blue1=>sblue4,
                    x_rec => "0100111100",
                    y_rec => "0000000000",
                    deltaX =>"0000001000",
                    deltaY=> "0111100000",
                    color => "111");

-- CONTOURS -----
haut:rect port map(row=>srow,col=>scol,red1=>sred5,green1=>sgreen5,blue1=>sblue5,
                  x_rec => "0000000000",
                  y_rec => "0000000000",
                  deltaX =>"1010000000",
                  deltaY=> "0000000110",
                  color => "111");

bas:rect port map(row=>srow,col=>scol,red1=>sred6,green1=>sgreen6,blue1=>sblue6,
                 x_rec => "0000000000",
                 y_rec => "0111011010",
                 deltaX =>"1010000000",
                 deltaY=> "0000000110",
                 color => "111");

gauche:rect port map(row=>srow,col=>scol,red1=>sred7,green1=>sgreen7,blue1=>sblue7,
                    x_rec => "0000000000",
                    y_rec => "0000000000",
                    deltaX =>"0000000110",
                    deltaY=> "0111100000",
                    color => "111");

droite:rect port map(row=>srow,col=>scol,red1=>sred8,green1=>sgreen8,blue1=>sblue8,
                    x_rec => "1001111010",
                    y_rec => "0000000000",
                    deltaX =>"0000000110",
                    deltaY=> "0111100000",
                    color => "111");

end atop;

```

Composant Rectangle

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--use ieee.numeric_std.all;

ENTITY rect IS PORT(
row,col,x_rec,y_rec :in STD_LOGIC_VECTOR(9 DOWNTO 0);
deltaX,deltaY : in std_logic_vector(9 DOWNTO 0);
color: in std_logic_vector(2 DOWNTO 0);
red1,green1,blue1 : out std_logic);
END rect;

ARCHITECTURE arect of rect is begin
PROCESS(row,col,x_rec,y_rec) BEGIN
if row > y_rec and row < y_rec+deltaY then
if col > x_rec and col < x_rec+deltaX then
red1 <= color(0);
green1 <= color(1);
blue1 <= color(2);
else
red1 <= '0';
green1 <= '0';
blue1 <= '0';
end if;
else
red1 <= '0';
green1 <= '0';
blue1 <= '0';
end if;
end process;
end arect;

```

Composant VGA_SYNC

```

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

ENTITY VGA_SYNC IS
PORT( clock_25Mhz, red, green, blue : IN STD_LOGIC;
red_out, green_out, blue_out, horiz_sync_out, vert_sync_out : OUT STD_LOGIC;
pixel_row, pixel_column: OUT STD_LOGIC_VECTOR(9 DOWNTO 0));
END VGA_SYNC;
ARCHITECTURE a OF VGA_SYNC IS
SIGNAL horiz_sync, vert_sync : STD_LOGIC;
SIGNAL video_on, video_on_v, video_on_h : STD_LOGIC;
SIGNAL h_count, v_count :STD_LOGIC_VECTOR(9 DOWNTO 0);

BEGIN
--Generate Horizontal and Vertical Timing Signals for Video Signal
-- H_count counts pixels (640 + extra time for sync signals)
--
-- Horiz_sync -----
-- H_count 0 640 659 755 799
--
gestion_H_Count:PROCESS(clock_25Mhz) BEGIN
IF(clock_25Mhz'EVENT) AND (clock_25Mhz='1') THEN
IF (h_count = 799) THEN

```

```

        h_count <= (others =>'0');
    ELSE
        h_count <= h_count + 1;
    END IF;
END IF;
END PROCESS;
gestion_Horiz_sync: PROCESS(clock_25Mhz,h_count) BEGIN
--Generate Horizontal Sync Signal using H_count
IF(clock_25Mhz'EVENT) AND (clock_25Mhz='0') THEN
    IF (h_count <= 755) AND (h_count >= 659) THEN
        horiz_sync <= '0';
    ELSE
        horiz_sync <= '1';
    END IF;
END IF;
END PROCESS;
--V_count counts rows of pixels (480 + extra time for sync signals)
--
-- Vert_sync -----
-- V_count      0          480  493-494      524
--
gestion_V_Count: PROCESS(clock_25Mhz,h_count) BEGIN
IF(clock_25Mhz'EVENT) AND (clock_25Mhz='1') THEN
    IF (v_count >= 524) AND (h_count >= 699) THEN
        v_count <= (others =>'0');
    ELSIF (h_count = 699) THEN
        v_count <= v_count + 1;
    END IF;
END IF;
END PROCESS;
gestion_Vertical_sync:PROCESS(clock_25Mhz,v_count) BEGIN
IF(clock_25Mhz'EVENT) AND (clock_25Mhz='0') THEN
-- Generate Vertical Sync Signal using V_count
IF (v_count <= 494) AND (v_count >= 493) THEN
    vert_sync <= '0';
ELSE
    vert_sync <= '1';
END IF;
END IF;
END PROCESS;
PROCESS(h_count,v_count) BEGIN
-- Generate Video on Screen Signals for Pixel Data
IF (h_count <= 639) THEN
    video_on_h <= '1';
ELSE
    video_on_h <= '0';
END IF;

IF (v_count <= 479) THEN
    video_on_v <= '1';
ELSE
    video_on_v <= '0';
END IF;
END PROCESS;
-- video_on is high only when RGB data is displayed
video_on <= video_on_H AND video_on_V;
pixel_column <= h_count;
pixel_row <= v_count;
-- Put all video signals through DFFs to eliminate any delays that cause a blurry image
red_out <= red AND video_on;
green_out <= green AND video_on;
blue_out <= blue AND video_on;
horiz_sync_out <= horiz_sync;
vert_sync_out <= vert_sync;

END a;
```

Fichier global TopToplogic

```

library ieee;
use ieee.std_logic_1164.all;
library unisim;
use unisim.vcomponents.all;
ENTITY TopTopLogic IS
port (

    MCLK:    in std_logic;
    --PORT0:  inout std_logic_vector( 7 downto 0 );
    --PORT1:  inout std_logic_vector( 7 downto 0 );
    --PORT2:  inout std_logic_vector( 7 downto 0 );
    PCOUT0:  out std_logic_vector( 7 downto 0 );
    PCOUT1:  out std_logic_vector( 7 downto 0 );
    PCOUT2:  out std_logic_vector( 7 downto 0 );
    PTIN0:   in std_logic_vector( 7 downto 0 );
    PTIN1:   in std_logic_vector( 7 downto 0 );
    PTIN2:   in std_logic_vector( 7 downto 0 );
    PTOU0:   out std_logic_vector( 7 downto 0 );
    PTOU1:   out std_logic_vector( 7 downto 0 );
    PTOU2:   out std_logic_vector( 7 downto 0 );
    PTSTB0:  out std_logic;
    PTSTB1:  out std_logic;
    PTSTB2:  out std_logic;
    RESET:   in std_logic;
    SLEEP:   out std_logic;
    --TESTIN: in std_logic;
    TMRCLK:  in std_logic;

    affpdsfaible,affpdsfort : out std_logic;
    sorties : out std_logic_vector(6 downto 0);
    color1: in std_logic_vector(2 downto 0);
    color2: in std_logic_vector(2 downto 0);
    hsynch,vsynch,red,green,blue : out STD_LOGIC

);
END TopTopLogic;

ARCHITECTURE aTopTopLogic OF TopTopLogic IS
signal EADR: std_logic_vector( 6 downto 0 );
signal EALU: std_logic_vector( 7 downto 0 );
signal EMCLK_16: std_logic;
signal EPRC: std_logic_vector( 10 downto 0 );
signal EWERAM: std_logic;
signal GP: std_logic_vector( 7 downto 0 );
signal PRESET: std_logic;
signal MCLK_16 : std_logic;
signal ROM: std_logic_vector( 11 downto 0 );

COMPONENT VGAtop IS
    PORT (clk_50 : in STD_LOGIC;
          x_rect, y_rect: IN STD_LOGIC_VECTOR(9 DOWNT0 0);
          posY1, posY2 :IN STD_LOGIC_VECTOR(9 DOWNT0 0);
          color1, color2 :IN STD_LOGIC_VECTOR(2 DOWNT0 0);
          hsynch,vsynch,red,green,blue : out STD_LOGIC);
END COMPONENT VGAtop;

component CLOCKDIV
port(
    MCLK:    in std_logic;
    MCLK_4:  out std_logic;
    MCLK_16: out std_logic;
    TEST:   in std_logic
);
end component;
component xilinx_one_port_ram_sync
generic(

```

```

ADDR_WIDTH: integer:=7;
DATA_WIDTH: integer:=8
);
port(
  clk: in std_logic;
  we: in std_logic;
  addr: in std_logic_vector(ADDR_WIDTH-1 downto 0);
  din: in std_logic_vector(DATA_WIDTH-1 downto 0);
  dout: out std_logic_vector(DATA_WIDTH-1 downto 0)
);
end component xilinx_one_port_ram_sync;

COMPONENT PIC_ROM
port(
  addr: in std_logic_vector(10 downto 0);
  data: out std_logic_vector(11 downto 0)
);
end COMPONENT PIC_ROM;

component TOPLOGIC
port (
  EADR:    out std_logic_vector( 6 downto 0 );
  EALU:    out std_logic_vector( 7 downto 0 );
  EMCLK_16: out std_logic;
  EPRC:    out std_logic_vector( 10 downto 0 );
  EWERAM:  out std_logic;
  GP:      in  std_logic_vector( 7 downto 0 );
  MCLK:    in  std_logic;
  PCOUT0:  out std_logic_vector( 7 downto 0 );
  PCOUT1:  out std_logic_vector( 7 downto 0 );
  PCOUT2:  out std_logic_vector( 7 downto 0 );
  PTIN0:   in  std_logic_vector( 7 downto 0 );
  PTIN1:   in  std_logic_vector( 7 downto 0 );
  PTIN2:   in  std_logic_vector( 7 downto 0 );
  PTOUT0:  out std_logic_vector( 7 downto 0 );
  PTOUT1:  out std_logic_vector( 7 downto 0 );
  PTOUT2:  out std_logic_vector( 7 downto 0 );
  PTSTB0:  out std_logic;
  PTSTB1:  out std_logic;
  PTSTB2:  out std_logic;
  PRESET:  in  std_logic;
  RESET:   in  std_logic;
  ROM:     in  std_logic_vector( 11 downto 0 );
  SLEEP:   out std_logic;
  TESTIN:  in  std_logic;
  TMRCLK:  in  std_logic
);
end component;
component tp1
port (
  clk : in std_logic;
  entrees : in std_logic_vector(7 downto 0);
  sorties : out std_logic_vector(6 downto 0);
  affpdsfaible,affpdsfort : out std_logic);
end component;

--SIGNAUX X
signal s_PORTA : std_logic_vector(9 downto 0);

--SIGNAUX Y
signal s_PORTB : std_logic_vector(9 downto 0);

--SIGNAUX POSITION RAQUETTES
signal posY1 : std_logic_vector(7 downto 0);
signal posY2 : std_logic_vector(7 downto 0);

signal s_affpdsfaible : std_logic;

```

```

--signal MCLK_256: std_logic;
BEGIN
i0: CLOCKDIV PORT MAP (MCLK=>MCLK,MCLK_16=>MCLK_16,TEST=>'0');
-- i01: CLOCKDIV PORT MAP (MCLK=>MCLK_16,MCLK_16=>MCLK_256,TEST=>'0');

i0b: VGAtop PORT MAP (clk_50 => MCLK,hsynch=>hsynch,vsynch=>vsynch,
red=>red,green=>green,blue=>blue,
x_rect=>s_PORTA,
color1=>color1,
color2=>color2,
-- POSITION RAQUETTE
posY1(8 downto 1)=>posY1,
posY1(9)=>'0',
posY1(0)=>'0',
posY2(8 downto 1)=>posY2,
posY2(9)=>'0',
posY2(0)=>'0',
y_rect=>s_PORTB);

i1: PIC_ROM PORT MAP (addr =>EPRC,data => ROM);

i1b:tp1 PORT MAP(clk=>MCLK,entrees=>EPRC(7 DOWNTO 0),sorties=>sorties,
affpdsfaible=>s_affpdsfaible,affpdsfort=>affpdsfort);
affpdsfaible <= s_affpdsfaible ;
i2:xilinx_one_port_ram_sync PORT MAP(
clk=>MCLK,
we=>EWERAM,
addr=>EADR,
din=>EALU ,
dout=>GP );

i3: TOPLOGIC PORT MAP (
TESTIN=>'0',
EADR=>EADR,
EALU=>EALU,

EMCLK_16=>EMCLK_16,
EPRC=>EPRC,
EWERAM=>EWERAM,
GP=>GP,
MCLK=>MCLK,
PCOUT0(1 DOWNTO 0)=>s_PORTA(9 DOWNTO 8),
PCOUT0(7 DOWNTO 2)=> PCOUT0(7 DOWNTO 2),
PCOUT1(1 DOWNTO 0)=>s_PORTB(9 DOWNTO 8),
PCOUT1(7 DOWNTO 2)=> PCOUT1(7 DOWNTO 2),

PCOUT2=> posY1,
PTIN0=>PTIN0,
PTIN1=>PTIN1,
PTIN2=>PTIN2,
PTOUT0=>s_PORTA(7 DOWNTO 0),
PTOUT1=>s_PORTB(7 DOWNTO 0),
PTOUT2=> posY2,
PTSTB0=>PTSTB0,
PTSTB1=>PTSTB1,
PTSTB2=>PTSTB2,
PRESET=>'0',--PRESET,
RESET=>RESET,
ROM=>ROM,
SLEEP=>SLEEP,

TMRCLK => TMRCLK

);
PTOUT1 <= s_PORTB(7 downto 0);
PTOUT0 <= s_PORTA(7 downto 0);

END aTopTopLogic ;

```

Implantation logicielle

Une fois la structure interne du FPGA finie, nous avons développé un programme en C pour le PIC. Le programme en C est compilé à l'aide de MPLab. Le fichier hexadécimal obtenu est à son tour converti en mémoire VHDL à l'aide d'un petit logiciel qui se nomme HEXtoVHDL. Ce fichier est ensuite importé dans xilinx. Il ne reste ensuite qu'à programmer le FPGA et l'application est prête.

Nous avons choisi pour notre exemple une application type « pong ». Voici code source:

```
#include <pic16F5x.h>
#define xmin 0
#define xmax 600
#define ymin 0
#define ymax 480
char incx=1, incy=1;
unsigned int posx=100, posy=100;

void setX(unsigned int x);
void setY(unsigned int y);
void wait(int tempo);

void main(){
    while(1){

        if(posx<xmin && incx<0) {incx=-incx;}
        if(posx>xmax && incx>0) {incx=-incx;}
        if(posy<ymin && incy<0) {incy=-incy;}
        if(posy>ymax && incy>0) {incy=-incy;}

        posx+=incx;
        posy+=incy;

        setX(posx);
        setY(posy);

        wait(30000);
        wait(30000);
    }
}

void setX(unsigned int x){
    //unsigned char i;
    PORTA=x;
    TRISA=x>>8;
}

void setY(unsigned int y){
    //unsigned char i;
    PORTB=y;
    TRISB=y>>8;
}

void wait(int tempo){
    int i;
    for(i=tempo;i>0;i--);
}
```

GEII IUT de Troyes	DOSSIER Projet encadré	Session 2009
-----------------------	---	--------------

Améliorations possibles

A la fin des séances de TR nous avons une application fonctionnelle. L'implantation du pic ainsi que sa programmation ont été une réussite. Le projet n'étant pas parfait, on peut citer quelques pistes d'amélioration :

- Utilisation d'un PIC plus performant : En effet le 16C57 est assez limité en mémoire ce qui pose problème lors de la programmation d'application plus complexes. Le FPGA étant utilisé seulement à 1 quart de ses capacités, il serait facile de remplacer le PIC par un plus performant
- Rendre le Pic programmable directement sans re-flasher le FPGA à chaque fois. Cela est possible mais nécessite une quantité de travail non négligeable. En effet il faudrait pour cela optimiser l'architecture du PIC pour le spartan3 et aussi développer les applications pour sa programmation depuis un PC.

Conclusion

Ce projet était très particulier puisqu'il nécessitait de travailler sur deux langages fondamentalement différents : le VHDL et le C. Nous avons de nombreuses questions : Comment le micro-processeur allait-il se comporter au sein du FPGA, un milieu qui n'est à la base pas le sien ? Quels sont les choses à modifier pour le faire fonctionner correctement ? Comment l'interfacer avec les entrées et les sorties du FPGA ? Comment le programmer, lui qui est à l'intérieur même du FPGA ? Toutes ces questions ainsi que les nombreux problèmes rencontrés tout au long du projet nous ont obligés à nous pencher sur le programme VHDL du micro-processeur et à comprendre son fonctionnement au sein du FPGA. De cette expérience est née une meilleure compréhension de son architecture ainsi que de la différence fondamentale entre celui-ci et les composants de logique programmables. Nous avons acquis aussi une certaine aisance dans la compréhension et l'utilisation du VHDL que seule l'expérience peut donner.