Le génie pour l'industrie

Architecture des systèmes ordinés et VHDL

ELE748

Travail remis à

Simon Pichette

LABORATOIRE 2

Par

Jonathan Lapointe (LAPJ05108303)
Kévin Parent Legault (PARK22049009)

Rédigé le

8 juin 2015

École de technologie supérieure
Département de Génie Électrique

Table des matières

Introduction

Dans le cadre du cours d'architecture des systèmes ordinés et VHDL, nous avons réalisé un deuxième système comportant une architecture modulaire. Cette architecture a été générée en utilisant l'utilitaire d'intégration système d'Altera, QSYS. Nous avons ensuite conçu une architecture logicielle fonctionnant sur ce système. Nous avons finalement déployé le tout sur le Cyclone V de la carte DE1 SOC d'Altera. Le système conçu est une application d'infographie simplifiée (NIOS draw) sur écran VGA qui sera contrôlée à l'aide d'une souris PS2.

Architecture du système

Description du système

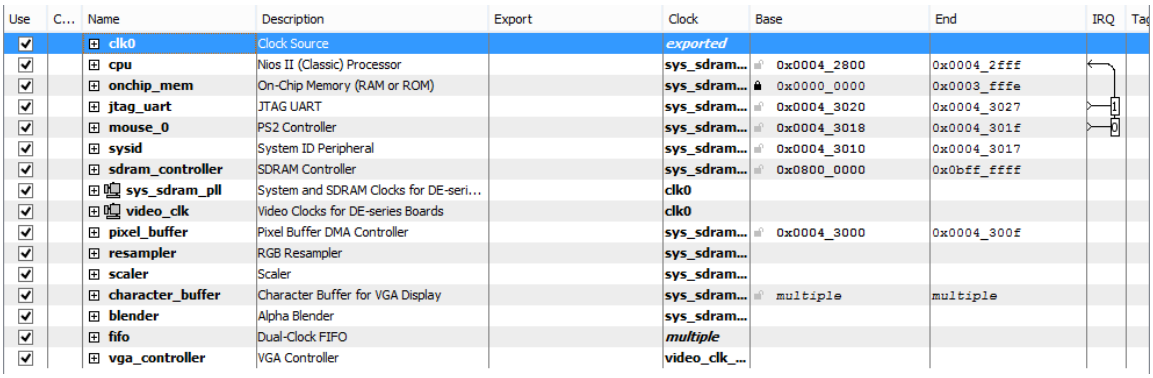
L'architecture système est composée de plusieurs modules tous générés à l'aide de QSYS. Ce dernier génère ensuite une composante VHDL qui contient les modules de notre architecture. Cette composante doit se faire connecter dans un fichier TOP au monde extérieur.

Notre architecture comprend les périphériques suivants :

Composante	Module Qsys	Description
cpu	NIOS II(classic)	Processeur RISC 32 bits
Onchip_mem	On-chip memory	Mémoire utilisé par pixel buffer (262143 octets, 32 bits)
Jtag_uart	JTAG UART	Permet d'écrire sur la console NIOS
Mouse_0	PS2 controller	Communique avec la souris PS2
Sysid	System id controller	Numéro d'identification système
Sdram_controller	SDRAM Controller	Contrôleur pour la SDRAM qui est sur la carte de développement (64Mbytes)
Sys_sram_pll	SDRAM clock	Déphasage l'horloge pour faire fonctionner la SDRAM
Video_clk	Video clock for DE board	Fourni une horloge de 25 Mhz au« vga controller »
Pixel_buffer	Pixel buffer DMA controller	Affiche les pixel sur l'écran(320X240)(background)
Resampler	RGB Resampler	Reformate l'image 320X240 à 640X480
Character_buffer	Character Buffer for VGA display	Affiche des caractères ASCII (80X60) (foreground)
blender	Alpha Blender	Mélange le background et le foreground ensemble
fifo	Dual clock FiFO	Permet faire le pont entre les 2 domaines d'horloges (50Mhz à 25Mhz)
Vga_controller	VGA Controller	Module qui gère le protocole VGA (25 Mhz) qui est connecté avec le DAC de sortie

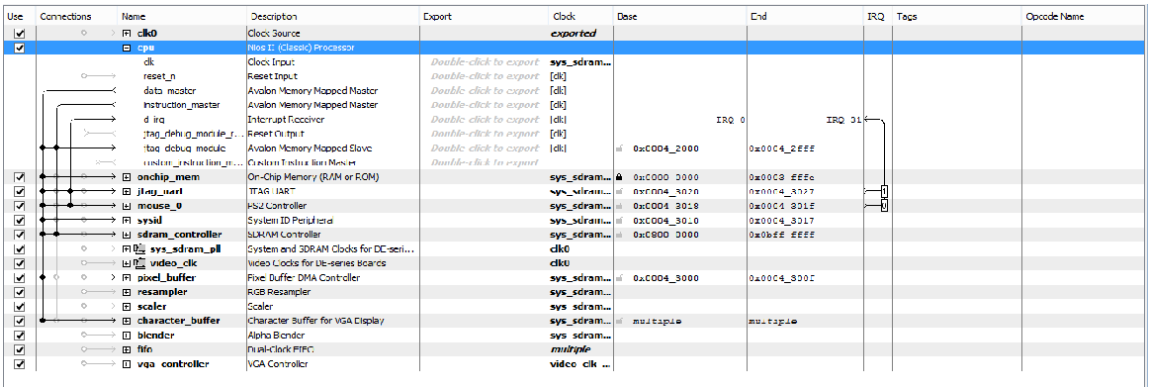
Configuration Qsys

Vue globale



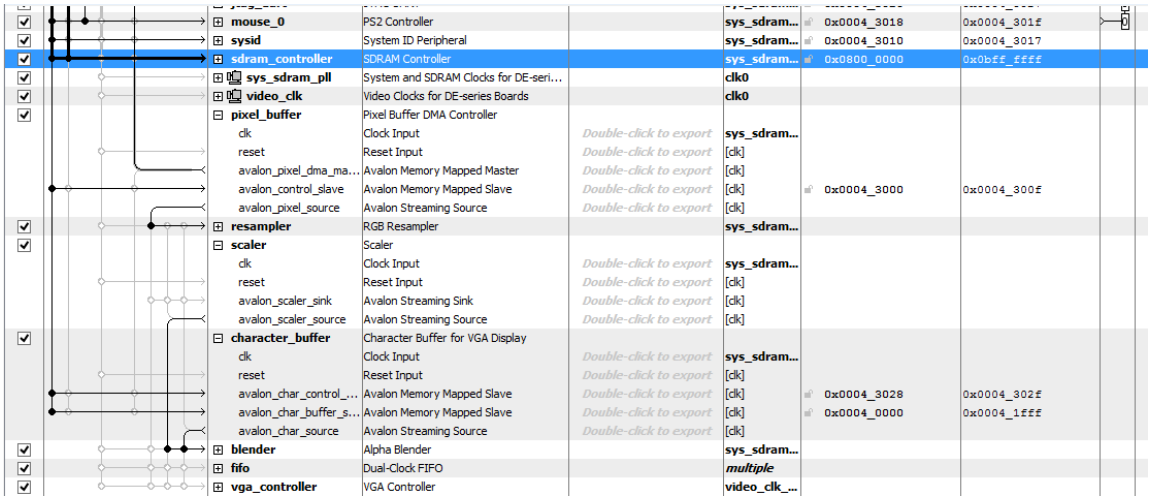
Use	C...	Name	Description	Export	Clock	Base	End	IRQ	Tag
☑		clk0	Clock Source		*exported*				
☑		cpu	Nios II (Classic) Processor		sys_sdram...	0x0004_2800	0x0004_2fff		
☑		onchip_mem	On-Chip Memory (RAM or ROM)		sys_sdram...	0x0000_0000	0x0003_fffe		
☑		jtag_uart	JTAG UART		sys_sdram...	0x0004_3020	0x0004_3027		
☑		mouse_0	PS2 Controller		sys_sdram...	0x0004_3018	0x0004_301f		
☑		sysid	System ID Peripheral		sys_sdram...	0x0004_3010	0x0004_3017		
☑		sdram_controller	SDRAM Controller		sys_sdram...	0x0800_0000	0x0bff_ffff		
☑		sys_sdram_pll	System and SDRAM Clocks for DE-seri...		clk0				
☑		video_clk	Video Clocks for DE-series Boards		clk0				
☑		pixel_buffer	Pixel Buffer DMA Controller		sys_sdram...	0x0004_3000	0x0004_300f		
☑		resampler	RGB Resampler		sys_sdram...				
☑		scaler	Scaler		sys_sdram...				
☑		character_buffer	Character Buffer for VGA Display		sys_sdram...	multiple	multiple		
☑		blender	Alpha Blender		sys_sdram...				
☑		fifo	Dual-Clock FIFO		*multiple*				
☑		vga_controller	VGA Controller		video_clk_...				

Partie Cpu



Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
☑		clk0	Clock Source		*exported*					
☑		cpu	Nios II (Classic) Processor							
		clk	Clock Input	Double-click to export	sys_sdram...					
		reset_n	Reset Input	Double-click to export	[clk]					
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]					
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]					
		d_irq	Interrupt Receiver	Double-click to export	[clk]			IRQ 0	IRQ 31	
		jtag_debug_module_r...	Reset Output	Double-click to export	[clk]					
		jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0004_2000	0x0004_27ff			
		custom_instruction_m...	Custom Instruction Master	Double-click to export						
☑		onchip_mem	On-Chip Memory (RAM or ROM)		sys_sdram...	0x0000_0000	0x0003_fffe			
☑		jtag_uart	JTAG UART		sys_sdram...	0x0004_3020	0x0004_3027			
☑		mouse_0	PS2 Controller		sys_sdram...	0x0004_3018	0x0004_301f			
☑		sysid	System ID Peripheral		sys_sdram...	0x0004_3010	0x0004_3017			
☑		sdram_controller	SDRAM Controller		sys_sdram...	0x0800_0000	0x0bff_ffff			
☑		sys_sdram_pll	System and SDRAM Clocks for DE-seri...		clk0					
☑		video_clk	Video Clocks for DE-series Boards		clk0					
☑		pixel_buffer	Pixel Buffer DMA Controller		sys_sdram...	0x0004_3000	0x0004_300f			
☑		resampler	RGB Resampler		sys_sdram...					
☑		scaler	Scaler		sys_sdram...					
☑		character_buffer	Character Buffer for VGA Display		sys_sdram...	multiple	multiple			
☑		blender	Alpha Blender		sys_sdram...					
☑		fifo	Dual-Clock FIFO		*multiple*					
☑		vga_controller	VGA Controller		video_clk_...					

Partie Vidéo



☑	mouse_0	PS2 Controller		sys_sdram...	0x0004_3018	0x0004_301f	
☑	sysid	System ID Peripheral		sys_sdram...	0x0004_3010	0x0004_3017	
☑	sdram_controller	SDRAM Controller		sys_sdram...	0x0800_0000	0x0bff_ffff	
☑	sys_sdram_pll	System and SDRAM Clocks for DE-seri...		clk0			
☑	video_clk	Video Clocks for DE-series Boards		clk0			
☑	pixel_buffer	Pixel Buffer DMA Controller					
	clk	Clock Input	Double-click to export	sys_sdram...			
	reset	Reset Input	Double-click to export	[clk]			
	avalon_pixel_dma_ma...	Avalon Memory Mapped Master	Double-click to export	[clk]			
	avalon_control_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0004_3000	0x0004_300f	
	avalon_pixel_source	Avalon Streaming Source	Double-click to export	[clk]			
☑	resampler	RGB Resampler		sys_sdram...			
☑	scaler	Scaler					
	clk	Clock Input	Double-click to export	sys_sdram...			
	reset	Reset Input	Double-click to export	[clk]			
	avalon_scaler_sink	Avalon Streaming Sink	Double-click to export	[clk]			
	avalon_scaler_source	Avalon Streaming Source	Double-click to export	[clk]			
☑	character_buffer	Character Buffer for VGA Display					
	clk	Clock Input	Double-click to export	sys_sdram...			
	reset	Reset Input	Double-click to export	[clk]			
	avalon_char_control_...	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0004_3028	0x0004_302f	
	avalon_char_buffer_s...	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0004_0000	0x0004_1fff	
	avalon_char_source	Avalon Streaming Source	Double-click to export	[clk]			
☑	blender	Alpha Blender		sys_sdram...			
☑	fifo	Dual-Clock FIFO		*multiple*			
☑	vga_controller	VGA Controller		video_clk_...			

Code VHDL

```
-- Create Date:  2015-06-02
-- Module Name: display.c
-- Project Name: lab2n
-- Target Devices: Altera cyclone V SOPC
-- Tool versions: NIOS II v14.1
-- Description: fichier de branchement de l'architecture vers l'extérieur


-- Revision: v1
-- Revision 0.01 - File Created
-- Additional Comments:


----------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
--Déclaration de l'entité TOP et des I/O
entity top is
 port (
            clock_50    : in std_logic;
            --Boutons
            KEY             : in std_logic_vector(3 downto 0);
            --écran VGA couleurs
            vga_r       : out std_logic_vector(7 downto 0);
            vga_b       : out std_logic_vector(7 downto 0);
            vga_g       : out std_logic_vector(7 downto 0);
            --écran VGA horloge
            vga_clk     : out std_logic;
            --écran VGA synchro
            vga_blank_n : out std_logic;
            vga_hs      : out std_logic;
            vga_vs      : out std_logic;
            vga_sync_n  : out std_logic;
            --broches souris
            ps2_clk     : inout std_logic;
            --ps2_clk2  : inout std_logic;
            ps2_dat     : inout std_logic;
            --ps2_dat2  : inout std_logic;
            --Broches allants vers SDRAM
            dram_clk    : out std_logic;
            dram_addr   : out std_logic_vector(12 downto 0);
            dram_dq         : inout std_logic_vector(15 downto 0);
            dram_ba         : out std_logic_vector(1 downto 0);
            dram_cas_n  : out std_logic;
            dram_cke    : out std_logic;
            dram_cs_n   : out std_logic;
            dram_we_n   : out std_logic;
            dram_udqm   : out std_logic;
            dram_ldqm   : out std_logic;
            dram_ras_n  : out std_logic
 );
 end top;
```

```
architecture structural of top is
 --PORT MAP de notre architecture
    component lab2q is
        port (
            clk_clk       : in    std_logic:= 'X';             -- clk
            vga_out_CLK   : out   std_logic;                                        -- CLK
            vga_out_HS    : out   std_logic;                                        -- HS
            vga_out_VS    : out   std_logic;                                        -- VS
            vga_out_BLANK : out   std_logic;                                        -- BLANK
            vga_out_SYNC  : out   std_logic;                                        -- SYNC
            --écran VGA couleurs
            vga_out_R     : out   std_logic_vector(7 downto 0);                     -- R
            vga_out_G     : out   std_logic_vector(7 downto 0);                     -- G
            vga_out_B     : out   std_logic_vector(7 downto 0);                     -- B
            --Broches allants vers SDRAM
            sram_addr     : out   std_logic_vector(12 downto 0);                    -- addr
            sram_ba       : out   std_logic_vector(1 downto 0);                     -- ba
            sram_cas_n    : out   std_logic;                                        -- cas_n
            sram_cke      : out   std_logic;                                        -- cke
            sram_cs_n     : out   std_logic;                                        -- cs_n
            sram_dq       : inout std_logic_vector(15 downto 0) := (others => 'X'); -- dq
            sram_dqm      : out   std_logic_vector(1 downto 0);                     -- dqm
            sram_ras_n    : out   std_logic;                                        -- ras_n
            sram_we_n     : out   std_logic;                                        -- we_n
            reset_reset_n : in    std_logic := 'X';                                 -- reset_n
            --broches souris
            mouse_CLK     : inout std_logic := 'X';                                 -- CLK
            mouse_DAT     : inout std_logic := 'X';                                 -- DAT
            --Broches allants vers SDRAM
            clk_sdram_clk : out   std_logic
        );
    end component lab2q;

  signal sdram_dqm : std_logic_vector(1 downto 0);
  begin

  nios_system : component lab2q
      port map (
          clk_clk       => clock_50 ,                    --     clk.clk
          --écran VGA horloge
          vga_out_CLK   => vga_clk,                          -- vga_out.CLK
          vga_out_HS    => vga_hs,                           --         .HS
          vga_out_VS    => vga_vs,                           --         .VS
          vga_out_BLANK => vga_blank_n,                      --         .BLANK
          vga_out_SYNC  => vga_sync_n,                       --         .SYNC
          --écran VGA couleurs
          vga_out_R     => vga_r,                            --         .R
          vga_out_G     => vga_g,                            --         .G
          vga_out_B     => vga_b,                            --         .B
          --Broches allants vers SDRAM
          sram_addr     => dram_addr,                        --    sram.addr
          sram_ba       => dram_ba,                          --         .ba
          sram_cas_n    => dram_cas_n,                       --         .cas_n
          sram_cke      => dram_cke,                         --         .cke
          sram_cs_n     => dram_cs_n,                        --         .cs_n
          sram_dq       => dram_dq,                          --         .dq
          sram_dqm      => sdram_dqm,                        --         .dqm
          sram_ras_n    => dram_ras_n,                       --         .ras_n
          sram_we_n     => dram_we_n,                        --         .we_n
          reset_reset_n => KEY(0),                           --   reset.reset_n
          --broches souris
          mouse_CLK     => ps2_clk ,                         --   mouse.CLK
```

```
--Broches allants vers SDRAM
dram_udqm <= sdram_dqm(1);
dram_ldqm <= sdram_dqm(0);



end structural;
```

Architecture logicielle

Dans ce laboratoire, l'architecture logicielle permet d'effectuer plusieurs choses, tel que :

- Recevoir, mettre en mémoire et décoder les trames envoyées par la souris PS2;
- Rafraichir les pixels et effacer l'écran VGA ;
- Envoyer les coordonnées de position et les actions de la souris à la console NIOS;
- Afficher un curseur (caractère ASCII dans notre cas) qui se déplace sur l'écran en fonction de la position de la souris;
- Afficher la position du curseur au bas de l'écran VGA.

Description de logiciel

Voici un tableau qui décrit plus en détail les librairies conçues pour ce projet :

Librairie	Nom des Fonctions	Description
Main (.c)	main	Routine principale
Display(.c/.h)	InitDisplay	Initialise pixel buffer et char buffer
	DrawPixel	Rafraichit la couleur d'un pixel selon sa position x et y
	DrawPixelColored	Affiche un pixel avec couleur en alternance selon x,y
	CleanDrawZone	Efface la zone de dessin
	UpdateCursorPosition	Rafraichit les valeurs de la position du curseur à l'écran
	NiosDrawApp	Routine principale de rafraichissement de l'écran
	DisplayCoordinate	Affiche les coordonnées au bas de l'écran
	SendTelemetry	Rafraichit les valeurs de télémétrie envoyées à la console Nios
Mouse(.c/.h)	ps2_isr	Routine d'interruption : mise en mémoire de X,Y,SW des évènements souris dans des FIFO
	mouseInit	Initialise la souris et les interruptions
	mouseGetNbEvent	Retourne le nombre d'évènement souris reçu
	mouseGetX	Extrait une valeur de X du fifo
	mouseGetY	Extrait une valeur de Y du fifo
	mouseGetSwL	Extrait une valeur de bouton gauche du fifo
	mouseGetSwR	Extrait une valeur de bouton droit du fifo
	mousePtrOutInc	Incrémente le pointeur de sortie du fifo et décrémente «NbEvent »; doit être appelé après avoir fait tous les « get »
JtagUart(.c/.h)	jUartSendString	Envoie sur console Nios une chaîne de caractère ou un tableau se terminant par un «NULL»
	jUartSendVar	Envoie une valeur décimale sur la console Nios
Hardware (.h)		Définition des types et inclusion des librairies utilisées dans le projet (BSP)

Code C

Display.c

```
Company: École de technologie supérieur
Engineer:
      Kevin Parent Legault
      Jonathan Lapointe
Create Date:  2015-06-02
Module Name: display.c
Project Name: lab2n
Target Devices: Altera cyclone V SOPC
Tool versions: NIOS II v14.1
Description:
   Module d'affichage permettant l'exécution de l'application NiosDraw, le déplacement d'un curseur sur un écran VGA d'une
   résolution de 320x240 redimensionné pour 640x480 et exécuté un tracé de couleur à l'écran.
   Le module possède également une fonction permettant d'afficher sur la console des informations télémétriques tel que :
            - Détection d'un clique droit et coordonnées x-y lors du clique.
            - Détection d'un clique gauche et coordonnées x-y lors du clique
   Le module affiche à l'écran la position du curseur.

   Possède les fonctions suivantes :
      1. void InitDisplay(void);
      2. void NiosDrawApp(void);
      3. U8 DrawPixelColored(U16 x,U16 y)
      4. void CleanDrawZone(void)
      5. U8 UpdateCursorPosition(U16 x, U16 y)
      6. void DisplayCoordinate(U16 x, U16 y)
      7. void SendTelemetry(U16 x, U16 y, U8 FlagR, U8 FlagL)


Revision: v1
```

```
/*
 * @fn   void InitDisplay(void)
 * @des  Ouvre les périphérique pixel buffer et char buffer
 * @arg  N'a besoin d'aucun argument
 * @ret  Retourne rien.
 */
void InitDisplay(void)
{
    pixelptr=alt_up_pixel_buffer_dma_open_dev(PIXEL_BUFFER_NAME); // Initialise pixel buffer
    charbuff = alt_up_char_buffer_open_dev("/dev/character_buffer"); // Initialise char buffer
}
```

```
/*
 * @fn   DrawPixel(U16 x, U16 y,U32 color)
 * @des  Affiche un pixel de la couleur désiré aux coordonnés x-y spécifiez
 * @arg  U16 x , U16 y, U32 color
 * @ret  Retourne 1 si la zone de dessin est dépassé.
 */
U8 DrawPixelColored(U16 x,U16 y)
{
    U16 val=0;
    if(colorInc > 3)colorInc = 0; // réinitialise la boucle à zéro
    if(y < DRAW_LIMIT_HEIGHT_MIN  || y > DRAW_LIMIT_HEIGHT_MAX )
    {
        val = DRAW_LIMIT_EXCEEDED;
    }
    else
    {
        val = OK;
        alt_up_pixel_buffer_dma_draw(pixelptr,colorSwap[colorInc++],x,y); // Active un pixel aux coordonnées x-y.
    }
return val;
}
```

```
/*
 * @fn   CleanDrawZone
 * @des  Nettoie la zone de dessin en remplissant la zone de la couleur BACKGROUND.
 * @arg  Ne prend aucun argument.
 * @ret  Ne retourne aucune valeur.
 */
void CleanDrawZone(void)
{
    alt_up_pixel_buffer_dma_draw_box(pixelptr,DISPLAY_RES_ORIGIN,DRAW_LIMIT_HEIGHT_MIN, DISPLAY_RES_WIDTH, DRAW_LIMIT_HEIGHT_MAX,BACKGROUND, 0);
}
```

```
/*
 * @fn   UpdateCursorPosition(U16 x,U16 y)
 * @des  Met à jour l'emplacement du curseur en effaçant l'ancienne emplacement
 * @arg U16 x : Coordonnées en x. U16 y : Coordonnées en y
 * @ret Retourne 1 si les limites de la zone de dessin sont dépassés.
 */
U8 UpdateCursorPosition(U16 x,U16 y)
{
    U16 val=0;
    if(y < DRAW_LIMIT_HEIGHT_MIN  || y > DRAW_LIMIT_HEIGHT_MAX )val = DRAW_LIMIT_EXCEEDED;
    else
    {
        val = OK;
        // Efface l'ancien curseur et mise à l'échelle de la position x-y du curseur en fonction de la résolution de l'écran
        alt_up_char_buffer_draw(charbuff,ERASE_LAST,xLastValue>>2,yLastValue>>2);
        // Dessine le curseur à la nouvelle position et mise à l'échelle de la positio xy du curseur en fonction de la résolution de l'écran.
        alt_up_char_buffer_draw(charbuff,CURSOR,x>>2,y>>2);

        xLastValue = x;
        yLastValue = y;
    }
return val;
}
```

```
* @fn   void NiosDrawApp(void)
* @des  Exécute l'application NiosDrawv0.1.
*                                      - Clique gauche: Active le tracé de couleur (Rouge,Bleu,Jaune et Vert).
*                                      - Clique droite: Efface la zone de dessin
*                                      - Le fond d'écran est modifiable par le define: BACKGROUND (choix entre : Rouge,Bleu,Jaune et Vert).
*                                      - Cycle de rafraichissement de l'écran à 60 cycles par seconde.
*                                      - Envoie de la télémétrie sur le port Jtag. (Clique gauche, clique droit, coordonées x-y lors du clique)
*
* @arg  Ne prend aucun argument
* @ret  Ne retourne pas de valeur
*/
void NiosDrawApp(void)
{
    //Initialisation du panneau de dessin.
    if(init)
    {
        alt_up_char_buffer_clear(charbuff);
        alt_up_pixel_buffer_dma_draw_box(pixelptr,DISPLAY_RES_ORIGIN,DRAW_LIMIT_HEIGHT_MIN, DISPLAY_RES_WIDTH, DRAW_LIMIT_HEIGHT_MAX,BACKGROUND, 0);
        // Envoie des caractères pour la bannière du haut avec les defines associés.
        alt_up_char_buffer_string(charbuff,LABEL,LABEL_POSITION_WIDTH_TOP, LABEL_POSITION_HEIGHT_TOP);
        // Envoie des caractères pour la bannière en bas à droite avec les defines associés.
        alt_up_char_buffer_string(charbuff,X,COORDINATE_X_POSITION_WIDTH,COORDINATE_X_POSITION_HEIGHT);
        alt_up_char_buffer_string(charbuff,Y,COORDINATE_Y_POSITION_WIDTH,COORDINATE_Y_POSITION_HEIGHT);
        alt_up_pixel_buffer_dma_swap_buffers(pixelptr); // Rotation du buffer. Attente du cycle de rafraichissement
    init = 0;
    }
```

```
    // Vérifie que le buffer estb complétement vide. L'écran VGA est donc rafraichie.
    if(alt_up_pixel_buffer_dma_check_swap_buffers_status(pixelptr) == 0)
    {
// Tourne tant que le FIFO de la souris n'est pas vide.
(FIFO : Zone tampon stockant les valeurs de déplacement de la souris. Cela nous permet de dessiné plus de point)
        while(mouseGetNbEvent())
        {
                updatePositionFlag = 1;
                xValue = mouseGetX(); // Stockage de la coordonnée de la souris en x
                yValue = mouseGetY(); // Stockage de la coordonnée de la souris en y
                xTempo=xValue;
                yTempo=yValue;
                if(mouseGetSWL()) //Vérifie si le bouton gauche est appuyé.
                {
                    DrawPixelColored(xValue,yValue); // Dessine un pixel d'une couleur.
                    SendTelemetry(xTempo,yTempo,0,TelemetryFlagButtonL);//Envoie des données télémétriques lorsque le flag est a 1.
                    TelemetryFlagButtonL=0; // Le flag boutton gauche à 0.
                }
                else if(mouseGetSWR())
                {
                    if(TelemetryFlagButtonR)CleanDrawZone(); // Nettoyage de la zone de dessin.
                    SendTelemetry(xTempo,yTempo,TelemetryFlagButtonR,0); // Envoie des données télémètriques lorsque le flag est a 1
                    TelemetryFlagButtonR=0;
                }
                if(mouseGetSWL()==0)TelemetryFlagButtonL=1; //Lève le Flag pour envoie de la télémétrie du bouton gauche
                if(mouseGetSWR()==0)TelemetryFlagButtonR=1; //Lève le Flag pour envoie de la télémétrie du bouton droit
                mousePtrOutInc(); // Incrément de l'index du FIFO souris.
                alt_up_pixel_buffer_dma_swap_buffers(pixelptr); // rotation du pixel buffer.Délai de rafraichissement de l'écran.
        }
```

```
        if(updatePositionFlag == 1) // Mis à jour des coordonnées et déplacement du curseur.
        {
            DisplayCoordinate(xTempo, yTempo);
            UpdateCursorPosition(xValue,yValue);
            updatePositionFlag = 0;
        }
    }
}
```

```
 * @fn   DisplayCoordinate(U16 x, U16 y)
 * @des  Affiche à l'écran les coordonnées du pointeur aux positions suivantes :
 *       Coordonnée en x : Width = de 58 à 62 et Height = 58
 *       Coordonnée en y : Width = de 63 à 67 et Height = 58
 * @arg  U16 x : Coordonnées en x et U16 y : Coordonnées en Y.
 * @ret  Ne retourne rien
 */
void DisplayCoordinate(U16 x, U16 y)
{
    U8 valueConv[4];
    U16 xt;
    U16 yt;
    S8 i=0;
    valueConv[3]=0;
    //Mise à l'échelle pour une résolution de 640x480
    xt = x<<1;
    yt = y<<1;
        for(i=2;i>=0;i--)//Conversion des valeurs U8 en caractère ascii correspondant pour x
        {
            valueConv[i]=(xt %10)+0x30;
            xt=xt/10;
        }
        alt_up_char_buffer_string(charbuff,valueConv,(COORDINATE_X_POSITION_WIDTH + X_SHIFT),COORDINATE_X_POSITION_HEIGHT);
        //Envoie de la Conversion des valeurs U8 en caractère ascii correspondant pour y
        for(i=2;i>=0;i--)
        {
            valueConv[i]=(yt %10)+0x30;
            yt=yt/10;
        }
        //envoie chaîne de caractère avec fonction Hal.
        alt_up_char_buffer_string(charbuff,valueConv,COORDINATE_Y_POSITION_WIDTH+Y_SHIFT, COORDINATE_Y_POSITION_HEIGHT);
}
```

```
/*
 * @fn   SendTelemetry(U16 x, U16 y, U8 FlagR,U8 FlagL)
 * @des  Envoie les données télémétriques tel que : Lorsqu'un bouton est appuyé et les coordonnées au moment de l'action.
 * @arg  U16 x : Coordonnés en x à envoyer.
 *       U16 y : Coordonnés en y à envoyer.
 *       U8  FlagR : Permet l'envoie sur le port jtag, lorsque FlagR = 1.
 *       U8  FlagL : Permet l'envoie sur le port jtag, lorsque FlagL = 1.
 * @ret  Ne retourne aucune donnée.
 */
void SendTelemetry(U16 x, U16 y, U8 FlagR,U8 FlagL)
{

    //Variables locaux
    U8 valueConvx[4],valueConvy[4];
    S8 i=0;
    valueConvx[3]=0;
    valueConvy[3]=0;
    U16 xt;
    U16 yt;

    //Mise à l'échelle pour une résoltion de 640x480
    xt = (x << 1)+2;
    yt = (y << 1)+2;

    //Conversion des valeurs en x en caractère ascii.
    for(i=2;i>=0;i--)
    {
        valueConvx[i]=(xt %10)+0x30;
        xt=xt/10;
    }
```

```
    //Conversion des valeurs en y en caractère ascii.
    for(i=2;i>=0;i--)
    {
        valueConvy[i]=(yt %10)+0x30;
        yt=yt/10;
    }
    //Envoie sur le port Jtag des informations d'état concernant le bouton de droite.
    if(FlagR)
    {

        jUartSendString("Right button pressed!\n"); // Envoie une chaine de caractère

        jUartSendString("X :  ");
        jUartSendString(valueConvx); // Envoie des coordonnées en x
        jUartSendString(" ");        // Espacement

        jUartSendString("Y :  ");
        jUartSendString(valueConvy); // Envoie des coordonnées en y
        jUartSendString("\n ");      // Saut de ligne

    }
```

```
    //Envoie sur le port Jtag des informations d'état concernant le bouton de gauche.
    if(FlagL)
    {

        jUartSendString("Left button pressed!\n");
        jUartSendString("X :  ");
        jUartSendString(valueConvx);    // Envoie des coordonnées en x
        jUartSendString(" ");           // Espacement

        jUartSendString("Y :  ");
        jUartSendString(valueConvy);    // Envoie des coordonées en y
        jUartSendString("\n ");         // Saut de ligne

    }
}
```

Display.h

```
Company: École de technologie supérieur
Engineer:
       Kevin Parent Legault
       Jonathan Lapointe
Create Date:  2015-06-02
Module Name: display.h
Project Name: lab2n
Target Devices: Altera cyclone V SOPC
Tool versions: NIOS II v14.1
Description:


Revision: v1
Revision 0.01 - File Created
Additional Comments:


**********************************************************************************************************/
#ifndef DISPLAY_H_
#define DISPLAY_H_

#include "hardware.h"
#include "jtagUart.h"
#include "mouse.h"
#include "altera_up_avalon_video_pixel_buffer_dma.h"
#include "altera_up_avalon_video_character_buffer_with_dma.h"
#include "altera_up_avalon_video_character_buffer_with_dma_regs.h"
#include "system.h"
```

```
//Pointeur périphérique pixel buffer ext char buffer
alt_up_pixel_buffer_dma_dev* pixelptr;
alt_up_char_buffer_dev* charbuff;

// Variables globales
U16 xLastValue;
U16 yLastValue;


/*
 * @fn   void InitDisplay(void)
 * @des  Ouvre les périphérique pixel buffer et char buffer
 * @arg  N'a besoin d'aucun argument
 * @ret  Retourne rien.
 */
void InitDisplay(void);

/*
 * @fn   void NiosDrawApp(void)
 * @des  Exécute l'application NiosDrawv0.1.
 *                                      - Clique gauche: Active le tracé de couleur (Rouge,Bleu,Jaune et Vert).
 *                                      - Clique droite: Efface la zone de dessin
 *                                      - Le fond d'écran est modifiable par le define: BACKGROUND (choix entre : Rouge,Bleu,Jaune et Vert).
 *                                      - Cycle de rafraichissement de l'écran à 60 cycles par seconde.
 *                                      - Envoie de la télémétrie sur le port Jtag. (Clique gauche, clique droit, coordonées x-y lors du clique)
 *
 * @arg  Ne prend aucun argument
 * @ret  Ne retourne pas de valeur
 */
void NiosDrawApp(void);

#endif /* DISPLAY_H_ */
```

Mouse.c

```
/************************************************************************************************
 Company: École de technologie supérieur
 Engineer:
        Kevin Parent Legault
        Jonathan Lapointe
 Create Date: 2015-06-02
 Module Name: mouse.c
 Project Name: lab2n
 Target Devices: Altera cyclone V SOPC
 Tool versions: NIOS II v14.1
 Description: fonctions de reception des données envoyées par la souris mis dans des buffer circulaire
                FIFO, fonctions de recuperations des données une a une GET
 Revision: v1
 Revision 0.01 - File Created
 Additional Comments:
************************************************************************************************/

#include "hardware.h"
#include "mouse.h"
#include "sys/alt_irq.h"
#include "altera_up_avalon_ps2.h"
#include "jtagUart.h"

#define MOUSE_Y_MAX 229
#define MOUSE_Y_MIN 10
#define MOUSE_X_MAX 319
#define MOUSE_EVENT  1
#define MOUSE_IDLE   0
#define MSG_Q_SIZE  10
#define MOUSE_X_SIGN_MASK  0x10
#define MOUSE_Y_SIGN_MASK  0x20
#define MOUSE_START_BYTE_MASK 0x08
```

```
#define MOUSE_BUFFER_SIZE 300

#define MOUSE_SW_L_MASK  0x01
#define MOUSE_SW_R_MASK  0x02
#define MOUSE_SW_L_SIGN_DEL   0
#define MOUSE_SW_R_SIGN_DEL   1
#define MOUSE_ERROR_FLAG "PS2 device not found!"
#define MOUSE_FLAG "PS2 device found!"
//type Variable contexte non utilisée
typedef struct message_queue {
    volatile unsigned char messages[MSG_Q_SIZE][3];     //Tableau de messages
    volatile unsigned char read_addr;                   //Addresse de lecture
    volatile unsigned char write_addr;                  //Addresse d'ecriture
}message_queue_t;
//Declaration de la structure du contenant de variables de contexte
//type Variable contexte non utilisée
typedef struct context {
    volatile unsigned char reset_flag, ready;   //Drapeaux de statut du PS/2
    message_queue_t *message_q;                 //Pointeur vers le tableau de message
    alt_up_ps2_dev *ps2;                        //Pointeur vers le port PS/2
} context_t;
context_t ps2_context;

//états de lectures des données envoyé par la souris
typedef enum eMouseState
{
    START_BYTE_STATE,
    X_BYTE_STATE,
    Y_BYTE_STATE
}tMouseState;
```

```
//DECLARATIONS DES VARIABLES POUR LES FIFO
//FIFOs de position
U8  yPosBuffer[MOUSE_BUFFER_SIZE];
U16 xPosBuffer[MOUSE_BUFFER_SIZE];
//FIFOS d'états des boutons
U8  swLeft[MOUSE_BUFFER_SIZE];
U8  swRight[MOUSE_BUFFER_SIZE];
//pointeurs in et out
U16  ptrIn=0;
U16 ptrOut=0;

U16  nbBytes=0;
//
U8 lastSwitchRight=0;
//fanion d'écriture aux fifo
U8 writeTobuffer=1;

//variables machine a état
static volatile U8 mouseState=START_BYTE_STATE;
alt_up_ps2_dev *ps2Inst;

//variables lecture des valeurs souris
static volatile S8 signX;
static volatile S8 signY;
static volatile U8 swLeftState;
static volatile U8 swRightState;

//compteur de position x et y
static volatile U16 mousePosX;
static volatile U16 mousePosY;
```

```
 * @fn   static void ps2_isr(void *context, alt_u32 id)
 * @des  routine d'interruption reception données souris
 * @arg  *context non utilisé, U32 id
 * @ret  Retourne rien.
 */
static void ps2_isr(void *context, U32 id)
{
    U8 data=0;
    if(alt_up_ps2_read_data_byte(ps2Inst, &data)==0 && nbBytes<MOUSE_BUFFER_SIZE)
    {
        switch(mouseState)
        {
        //START_STATE
        default:
        //synchronisation avec le premier octet
         if(data&MOUSE_START_BYTE_MASK)
         {
             signX=(data&MOUSE_X_SIGN_MASK);
             signY=(data&MOUSE_Y_SIGN_MASK);
             swRightState =(data&MOUSE_SW_R_MASK)>>MOUSE_SW_R_SIGN_DEL;
             swLeftState=(data&MOUSE_SW_L_MASK)>>MOUSE_SW_L_SIGN_DEL;
             if((lastSwitchRight==swRightState)&&(swRightState==MOUSE_SW_R_MASK))
             {
                 writeTobuffer=0;
             }
             else
             {
                 writeTobuffer=1;
             }
             mouseState=X_BYTE_STATE;
             lastSwitchRight=swRightState;
         }
        break;
```

```
//lecture du byte X
case X_BYTE_STATE:

    if(data!=0)
    {
        //effectue une soustraction si valeur signée
        if(signX)
        {
            data = (255-data);
            if(data>mousePosX)
            mousePosX=0;
            else

            mousePosX-=data;

        }
        else
        {
            mousePosX+=data;
        }
        if(mousePosX>MOUSE_X_MAX)
            mousePosX=MOUSE_X_MAX;
    }
    mouseState=Y_BYTE_STATE;
break;
```

```
//lecture du byte Y
case Y_BYTE_STATE:

    if(data!=0)
    {
        //effectue une soustraction si valeur signée
        if(signY)
        {
            data = (255-data);
            mousePosY+=data;

        }
        else
        {
            if(data>mousePosY)
                mousePosY=0;
            else
                mousePosY-=data;

        }
        if(mousePosY>MOUSE_Y_MAX)
                    mousePosY=MOUSE_Y_MAX;
        if(mousePosY<MOUSE_Y_MIN)
                    mousePosY=MOUSE_Y_MIN;
    }
    mouseEvent=MOUSE_EVENT;

    //Place les données dans des FIFO
    if(writeTobuffer)
    {
        //stocke les valeurs dans les fifos
        yPosBuffer[ptrIn]=mousePosY;
        xPosBuffer[ptrIn]=mousePosX;
        swLeft[ptrIn]=swLeftState;
        swRight[ptrIn]=swRightState;
        ptrIn++;
        nbBytes++;

        if(ptrIn>MOUSE_BUFFER_SIZE-1)
            ptrIn=0;
    }
    //reinitialise la machine a etat à start state
    mouseState=START_BYTE_STATE;
break;
    }
}
```

```
/*
 * @fn   void mouseInit(void)
 * @des  init de la souris et des interruption souris
 * @arg  void
 * @ret  Retourne rien.
 */
void mouseInit(void)
{

    ps2Inst=alt_up_ps2_open_dev(MOUSE_0_NAME);

    alt_irq_register(MOUSE_0_IRQ_INTERRUPT_CONTROLLER_ID,(void *) (&ps2_context),ps2_isr);
    alt_up_ps2_enable_read_interrupt(ps2Inst);

    if(ps2Inst == 0)jUartSendString(MOUSE_ERROR_FLAG);
    else if(ps2Inst != 0)jUartSendString(MOUSE_FLAG);

}
```

```
/*
 * @fn   U8 mouseGetNbEvent(void)
 * @des  retourne le nb de trames disponibles
 * @arg  void
 * @ret  U8 nBEvent.
 */
U8 mouseGetNbEvent(void)
{
    return nbBytes;
}
/*
 * @fn   void mousePtrOutInc(void)
 * @des  incremente le ptr de sortie du fifo
        décrémente le nb de trames disponibles
        doit etre appeler apres avoir récupéré
        toutes les données souris
 */
void mousePtrOutInc(void)
{
    ptrOut++;
    nbBytes--;
    if(ptrOut>MOUSE_BUFFER_SIZE-1)
        ptrOut=0;
}
```

```
U16 mouseGetX(void)
{
    return xPosBuffer[ptrOut];
}
/*
 * @fn   U16 mouseGetY(void)
 * @des  retourne la donnée de position Y
 */
U16 mouseGetY(void)
{
    return (U16)yPosBuffer[ptrOut];
}
/*
 * @fn   U8 mouseGetSWL(void)
 * @des  retourne la donnée de bouton gauche
 */
U8 mouseGetSWL(void)
{
    return swLeft[ptrOut];
}
/*
 * @fn   U8 mouseGetSWR(void)
 * @des  retourne la donnée de bouton droit
 */
U8 mouseGetSWR(void)
{
    return swRight[ptrOut];
}
```

Mouse.h

```
/**************************************************************************************************
 Company: École de technologie supérieur
 Engineer:
       Kevin Parent Legault
       Jonathan Lapointe
 Create Date: 2015-06-02
 Module Name: mouse.h
 Project Name: lab2n
 Target Devices: Altera cyclone V SOPC
 Tool versions: NIOS II v14.1
 Description:


 Revision: v1
 Revision 0.01 - File Created
 Additional Comments:


**************************************************************************************************/
#include"hardware.h"


/*
 * @fn   void mouseInit(void)
 * @des  init de la souris et des interruption souris
 * @arg  void
 * @ret  Retourne rien.
 */
void mouseInit(void);


/*
 * @fn   U8 mouseGetNbEvent(void)
 * @des  retourne le nb de trames disponibles
 * @arg  void
 * @ret  U8 nBEvent.
 */
U8 mouseGetNbEvent(void);
/*
 * @fn   U16 mouseGetY(void)
 * @des  retourne la donnée de position Y
 */
U16 mouseGetY(void);


/*
 * @fn   U16 mouseGetX(void)
 * @des  retourne la donnée de position X
 */
U16 mouseGetX(void);

/*
 * @fn   U8 mouseGetSWL(void)
 * @des  retourne la donnée de bouton gauche
 */
U8 mouseGetSWL(void);

/*
 * @fn   U8 mouseGetSWR(void)
 * @des  retourne la donnée de bouton droit
 */
U8 mouseGetSWR(void);
```

```
/*
 * @fn    void mousePtrOutInc(void)
 * @des   incremente le ptr de sortie du fifo
          décrémente le nb de trames disponibles
          doit etre appeler apres avoir récupéré
          toutes les données souris
 */
void mousePtrOutInc(void);
```

JtagUart.c

```
/**************************************************************************************************
    Company: École de technologie supérieur
    Engineer:
           Kevin Parent Legault
           Jonathan Lapointe
    Create Date:    2015-05-17
    Design Name: Lab2
    Module Name: jtagUart
    Project Name: Lab2
    Target Devices: Altera Cyclone V SOPC
    Tool versions: NIOSII V14.2
    Description:
    Module de communication jtagUart pour la transmission de trame entre le port console NIOSII console et le
    board.
    Contient les fonctions suivantes :

    1. void jUartSendString(U8 *ptrStr)
        - Tant que le tableau contenant le message n'est pas vide, on envoie une donnée sur le port jtagUart
        - Vérifie le registre controle et autorise la transmission sur le port si celui-ci est en mode lecture.

    2. void jUartSendVar(U16 value)
        -Converti les valeurs numériques en leur valeur ascii correspondante.
        - Fait appel à la fonction JuartSendString pour l'envoie des données


    Revision: v1
    Revision 0.01 - File Created
    Additional Comments:


**************************************************************************************************/
```

```
#include "jtagUart.h"

#define JUART_DATA_REG_OFT       0           // offset du data register
#define JUART_CTRL_REG_OFT       1


/*
 * @fn   void jUartSendString(U8 *ptrStr)
 * @des  Permet d'envoyé des chaines de caractère sur le port console jtagUart
 * @arg  Pointeur sur un tableau contenant le message à envoyer
 */

void jUartSendString(U8 *ptrStr)
{
    U32 data32;
    while(*ptrStr != '\0')
    {
            data32 = (U32) *ptrStr;
            if (IORD(JTAG_UART_BASE,JUART_CTRL_REG_OFT) != 0) {
                IOWR(JTAG_UART_BASE,JUART_DATA_REG_OFT, data32);
                ptrStr++;
            }
    }
}
```

```
/*
 * @fn   void jUartSendVar(U16 value)
 * @des  Permet l'envoie de variable décimale
 * @arg  U16 valeur décimale entre 0 et 1023.
 */

void jUartSendVar(U16 value)
{
    S8 i;
    U8 valueDigit[7];
    //segmentation des digits
    for(i=3;i>=0;i--)
    {
        valueDigit[i]=(value %10)+0x30; // conversion en caractère ascii équivalent.
        value=value/10;
    }
    valueDigit[4]=0;
    jUartSendString(valueDigit); // appel de fonction.

}
```

jtagUart.h

```
//*jtagUart.c pour description de fichier.
#ifndef __JTAG_UART_H__
#define __JTAG_UART_H__

#include "hardware.h"


/*
 * @fn   void jUartSendString(U8 *ptrStr)
 * @des  Permet d'envoyé des chaines de caractère sur le port console jtagUart
 * @arg  Pointeur sur un tableau contenant le message à envoyer
 */
void jUartSendString(U8 *ptrStr);

/*
 * @fn   void jUartSendVar(U16 value)
 * @des  Permet l'envoie de variable décimale
 * @arg  U16 valeur décimale entre 0 et 1023.
 */

void jUartSendVar(U16 value);

#endif
```

Hardware.h

```
#ifndef HARDWARE_H
#define HARDWARE_H

// Inclusion des librairies bsp
#include "io.h"
#include "alt_types.h"
#include "system.h"
#include "sys/alt_sys_init.h"



//DECLARATION DE TYPES
#define U8  alt_u8
#define S8  alt_8
#define U32 alt_u32
#define U16 alt_u16


void hDInitHardware(void);


#endif
```

Main.c

```
/**************************************************************************************************************

 Company: École de technologie supérieur
 Engineer:
       Kevin Parent Legault
       Jonathan Lapointe
 Create Date:  2015-05-26
 Module Name: main.c
 Project Name: lab2n
 Target Devices: Altera cyclone V SOPC
 Tool versions: NIOS II v14.1
 Description: Programme principale appelant l'application NiosDrawv0.1


 Revision: v1
 Revision 0.01 - File Created
 Additional Comments:


**************************************************************************************************************/

//Function declaration


#include "hardware.h"
#include "jtagUart.h"
#include "system.h"
#include "display.h"
#include "mouse.h"
```

```
/*
 * @fn   int main(void)
 * @des  Programme principale
 * @arg  void
 * @ret retourne 0
 */
int main(void)
{
//Init des périphériques
    InitDisplay();
    mouseInit();
    for(;;)
    {

        NiosDrawApp(); // Exécute l'app niosdraw.

    }
return 0;
}
```

Discussion

Ce laboratoire, servant à nous initier avec les périphériques avancés de même que les librairies HAL, a été difficile à réaliser sur plusieurs aspects. En effet, lors de l'implantation de l'architecture physique, un des fichiers Qsys était corrompu ce qui nous a obligé à préparer un environnement de développement sur nos ordinateurs personnels ayant ainsi un impact considérable sur l'échéancier fixé préalablement. De plus, la mise à jour des librairies HAL et les documents référentiels pas à jour, à augmenter de façon considérable le niveau de difficulté concernant la réalisation du projet. De plus, lors de notre phase de validation on a remarqué que la souris ne s'activait pas tout le temps, et ce, même si le périphérique est bel et bien initialisé. Ce problème persiste et on n'a toujours pas trouvé la cause de cette défaillance. En contrepartie, la plupart des spécifications exigées dans le devis ont été rencontrées. En effet, on pense que notre système répond de façon honnête aux exigences. Finalement, on a pu observer qu'avec les modules« IP cores» il était possible d'intégrer et de rapidement concevoir un système complexe.

Conclusion

En terminant, ce laboratoire n'a pas été facile à réaliser à cause des outils de développement. Nous avons, cependant, beaucoup appris à les connaître et surtout à nous débrouiller par nous même pour les faire marcher. Il s'agissait d'une expérience enrichissante. Ce laboratoire nous permettra probablement d'être plus à l'aise avec les outils pour réaliser le projet qui suivra.