

Homework 2 : Accelerometer and Custom Instruction

1) QSYS

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags
<input checked="" type="checkbox"/>		clk_0	Clock Source		exported				
		clk_in	Clock Input	clk	clk_0				
		clk_in_reset	Reset Input	reset					
		clk	Clock Output						
		clk_reset	Reset Output						
<input checked="" type="checkbox"/>		onchip_mem	On-Chip Memory (RAM or ROM)						
		clk1	Clock Input		clk_0 [clk1]				
		s1	Avalon Memory Mapped Slave			0x0000_8000	0x0000_ce1f		
		reset1	Reset Input						
<input checked="" type="checkbox"/>		cpu	Nios II Processor						
		clk	Clock Input		clk_0 [clk]				
		reset	Reset Input						
		data_master	Avalon Memory Mapped Master						
		instruction_master	Avalon Memory Mapped Master						
		irq	Interrupt Receiver					IRQ 0	IRQ 31
		debug_reset_request	Reset Output						
		debug_mem_slave	Avalon Memory Mapped Slave			0x0001_0800	0x0001_0fff		
		custom_instruction_m...	Custom Instruction Master						
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART						
		clk	Clock Input		clk_0 [clk]				
		reset	Reset Input						
		avalon_jtag_slave	Avalon Memory Mapped Slave			0x0001_1460	0x0001_1467		
		irq	Interrupt Sender						
<input checked="" type="checkbox"/>		button	PIO (Parallel I/O)						
		clk	Clock Input		clk_0 [clk]				
		reset	Reset Input						
		s1	Avalon Memory Mapped Slave			0x0001_1450	0x0001_145f		
		external_connection	Conduit	button_external_conne...					
<input checked="" type="checkbox"/>		switches	PIO (Parallel I/O)						
		clk	Clock Input		clk_0 [clk]				
		reset	Reset Input						
		s1	Avalon Memory Mapped Slave			0x0001_1440	0x0001_144f		
		external_connection	Conduit	switches_external_conn...					
<input checked="" type="checkbox"/>		my_custom_ip_0	my_custom_ip						
		clock_reset	Clock Input		clk_0 [clock_reset]				
		reset	Reset Input						
		conduit_end	Conduit	my_custom_ip_0_cond...					
		avalon_slave_0	Avalon Memory Mapped Slave			0x0001_1000	0x0001_13ff		
<input checked="" type="checkbox"/>		acc	PIO (Parallel I/O)						
		clk	Clock Input		clk_0 [clk]				
		reset	Reset Input						
		s1	Avalon Memory Mapped Slave			0x0001_1430	0x0001_143f		
		external_connection	Conduit	acc_external_connection					
		irq	Interrupt Sender						
<input checked="" type="checkbox"/>		timer_0	Interval Timer						
		clk	Clock Input		clk_0 [clk]				
		reset	Reset Input						
		s1	Avalon Memory Mapped Slave			0x0001_1400	0x0001_141f		
		irq	Interrupt Sender						
<input checked="" type="checkbox"/>		TERASIC_SPI_3WIR...	TERASIC_SPI_3WIRE						
		clock_reset	Clock Input		clk_0 [clock_reset]				
		clock_reset_reset	Reset Input						
		slave	Avalon Memory Mapped Slave			0x0000_0000	0x0000_003f		
		conduit_end	Conduit	terasic_spi_3wire_0_co...					
<input checked="" type="checkbox"/>		my_custom_inst_0	my_custom_inst						
		ncs_cis0	Custom Instruction Slave						

Par rapport au premier devoir , j'ai ajouté un accéléromètres qui est un PIO avec le TERASIC_SPI_3WIRE, le timer et le my_custom_inst

J'ai laissé les switches mais ce n'est pas nécessaire pour ce devoir car le pwm ne sera plus donnée par les switches mais pas l'accéléromètre en fonction de l'inclinaison qu'aura notre FPGA.

2) Choix de Led et contrôle de LED

- my_custom_instr.sv

Ce module va nous fournir un registre avec la led qui va s'allumer et celle qui seront éteinte. Il reçoit en entrée une data, qui provient de l'accéléromètres auquel on a ajouté un offset pour éviter d'avoir un nombre négatif (voir code c). En fonction de cette valeur elle retourne la led à allumer

```
11 `timescale 1 ps / 1 ps
12 module my_custom_instr (
13     input wire [31:0] ncs_cis0_dataaa, // ncs_cis0.dataaa
14     output wire [31:0] ncs_cis0_result // .result
15 );
16
17 // TODO: Auto-generated HDL template
18
19 logic [7:0] led;
20
21 always_comb begin
22     if (ncs_cis0_dataaa < 32'd66) led <= 8'b10000000;
23     else if (ncs_cis0_dataaa < 32'd132) led <= 8'b01000000;
24     else if (ncs_cis0_dataaa < 32'd198) led <= 8'b00100000;
25     else if (ncs_cis0_dataaa < 32'd264) led <= 8'b00010000;
26     else if (ncs_cis0_dataaa < 32'd330) led <= 8'b00001000;
27     else if (ncs_cis0_dataaa < 32'd396) led <= 8'b00000100;
28     else if (ncs_cis0_dataaa < 32'd462) led <= 8'b00000010;
29     else led <= 8'b00000001;
30 end
31
32 assign ncs_cis0_result = {24'b0,led};
33
34 endmodule
```

- module my_custom_ip

Celui-ci est resté quasi le même que la dernière fois, il contrôle les leds. Il va allumer et éteindre les LEDs.

Il fallait adapter le code afin qu'il puisse lire l'adresse dans lequel on a mis la led à allumer et les leds à éteindre. Qu'on connaît grâce à notre custom instruction.

```

begin
  if (write) begin //recuperation des data
    if (address == 8'd1) duty_cycle <= writedata[31:0];
    else if (address == 8'd2) period <= writedata[31:0];
    else if (address == 8'd3) led <= writedata[31:0];
  end
end

always_ff @(posedge clock) // envoie du duty_cycle ce qui nous permettra de le lire
if(read) readdata <= {28'b0, duty_cycle};
else readdata <= 32'b0;

always_ff @(posedge clock)
begin
  if(!reset) begin //remise a 0 lorsque on appuie sur key[0]
    LED <= 8'b0;
    compteur <= 32'b0;
  end

  else begin

    if (compteur <= period) compteur <= compteur + 1;
    else compteur <= 32'b0;
    if (compteur <= duty_cycle*period/15) LED <= led[7:0];
    else LED <= 8'b0;
  end
end
end

```

3) Gestion du PWM et du bouton pause

Comme la dernière fois, la gestion du pwm est fait dans le software ainsi que le bouton pause. voir le code c (commenté).

utilisation du custom instruction dans le software :

```

ADXL345_SPI_XYZ_Read(TERASIC_SPI_3WIRE_0_BASE,g_XYZ);
*(mycustom+3) = ALT_CI_MY_CUSTOM_INST_0(g_XYZ[0]+280,0);

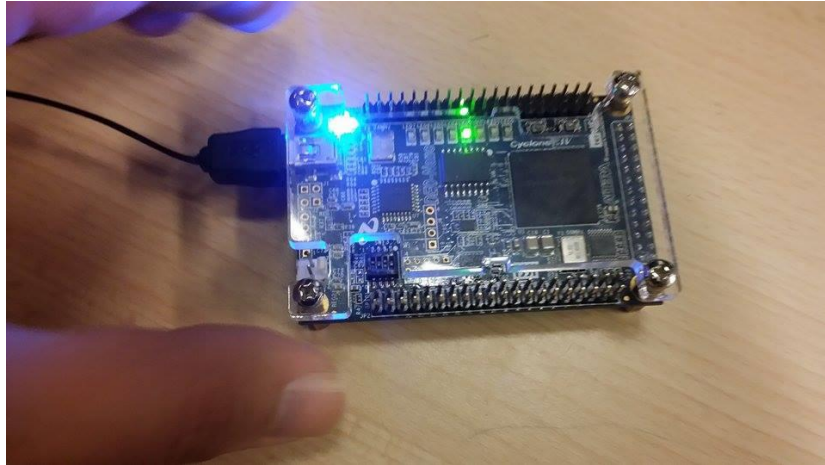
```

Dans cette ligne de code on voit bien qu'on lit la valeur de la gravité dans l'axe x additionner de 280 afin d'éviter les nombres négatifs. J'ai choisi 280 car le minimum était au alentour des -270.

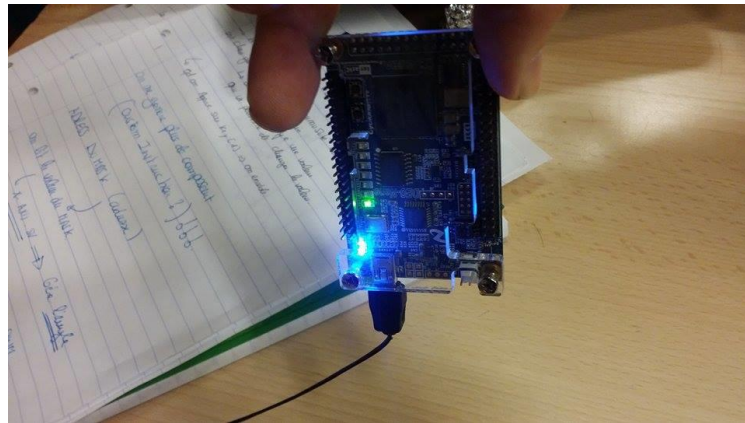
Puis cette valeur est envoyé dans le input du custom instruction, qui à son tour va nous renvoyé les valeur des led, qui seront écritent dans l'adresse 3 et qui sera plus tard utilisé dans le my_custom_ip.

4) Bon fonctionnement

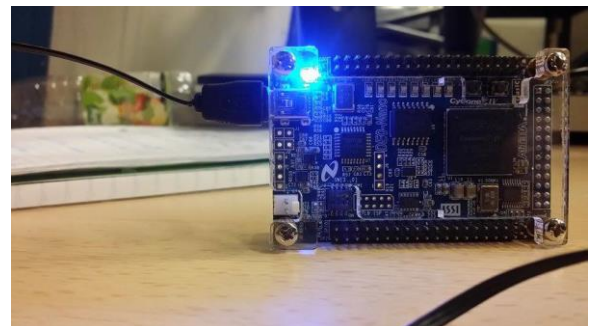
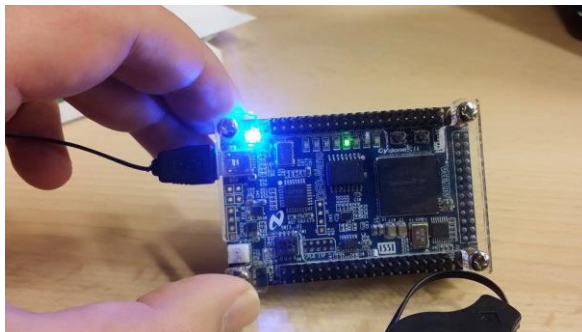
La plaque déposer sur la table, donc on est à luminosité maximum et c'est la led4 qui s'allume



Déplacement de la lumière dans le sens de la pente



On observe que la luminosité de la led diminue lorsqu'on soulève la plaque de cette manière. Jusqu'à totalement s'éteindre lorsque la plaque est à 90°



Lorsqu'on appuie sur KEY[1] alors on met le système sur pause donc on le voit bien sur cette photo.

Le FPGA est sur la table mais la LED1 est allumer au lieu de la LED4 car j'ai appuyé sur le bouton lorsque il était penché.

