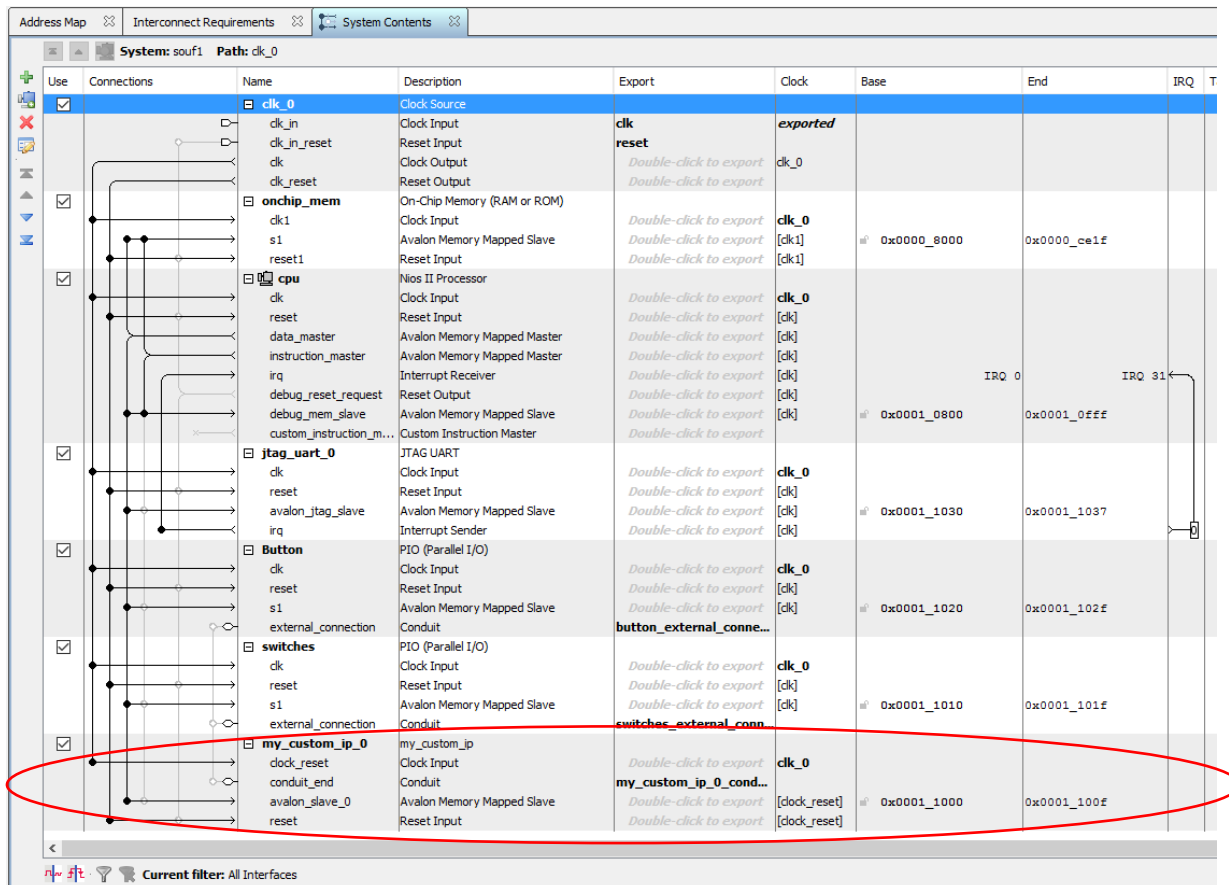
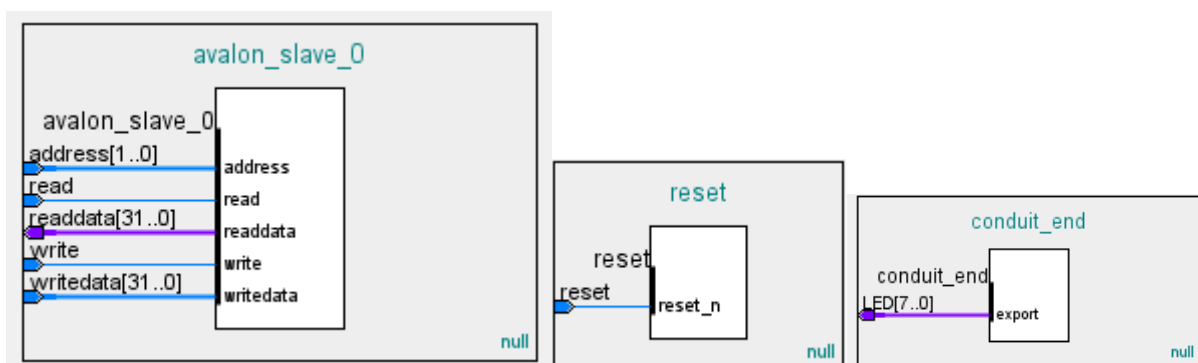


Homework 1

J'ai commencé par crée My_custom_ip.sv afin de crée un nouveau composant dans QSYS



Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	T
<input checked="" type="checkbox"/>		clk_0	Clock Source						
		clk_in	Clock Input	clk	exported				
		clk_in_reset	Reset Input	reset					
		clk	Clock Output	Double-click to export	clk_0				
		clk_reset	Reset Output	Double-click to export					
<input checked="" type="checkbox"/>		onchip_mem	On-Chip Memory (RAM or ROM)						
		clk1	Clock Input	Double-click to export	clk_0				
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	# 0x0000_8000	0x0000_ce1f		
		reset1	Reset Input	Double-click to export	[clk1]				
<input checked="" type="checkbox"/>		cpu	Nios II Processor						
		clk	Clock Input	Double-click to export	clk_0				
		reset	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]				
		irq	Interrupt Receiver	Double-click to export	[clk]			IRQ 0	IRQ 31
		debug_reset_request	Reset Output	Double-click to export	[clk]				
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	# 0x0001_0800	0x0001_0fff		
		custom_instruction_m...	Custom Instruction Master	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART						
		clk	Clock Input	Double-click to export	clk_0				
		reset	Reset Input	Double-click to export	[clk]				
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	# 0x0001_1030	0x0001_1037		
		irq	Interrupt Sender	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		Button	PIO (Parallel I/O)						
		clk	Clock Input	Double-click to export	clk_0				
		reset	Reset Input	Double-click to export	[clk]				
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	# 0x0001_1020	0x0001_102f		
		external_connection	Conduit	button_external_conne...					
<input checked="" type="checkbox"/>		switches	PIO (Parallel I/O)						
		clk	Clock Input	Double-click to export	clk_0				
		reset	Reset Input	Double-click to export	[clk]				
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	# 0x0001_1010	0x0001_101f		
		external_connection	Conduit	switches_external_conn...					
<input checked="" type="checkbox"/>		my_custom_ip_0	my_custom_ip						
		clock_reset	Clock Input	Double-click to export	clk_0				
		conduit_end	Conduit	my_custom_ip_0_cond...	[clock_reset]	# 0x0001_1000	0x0001_100f		
		avalon_slave_0	Avalon Memory Mapped Slave	Double-click to export	[clock_reset]				
		reset	Reset Input	Double-click to export					



On voit en bleu les inputs et en mauve les outputs

Le code nécessaire à la création de notre composant se trouve ci-dessous

```
1 `timescale 1 ps / 1 ps
2 module my_custom_ip(
3     output reg [7:0] LED,
4     input wire clock,
5     input wire reset,
6     input wire [7:0] address,
7     input wire read,
8     output reg [31:0] readdata,
9     input wire write,
10    input wire [31:0] writedata
11 );
12
13    logic enable;
14    logic [3:0] duty_cycle;
15    logic [31:0] period;
16    logic [31:0] compteur;
17
18    always_ff @(posedge clock)
19    begin
20        if (write) begin //recuperation des data
21            if (address == 8'd0) enable <= writedata[0];
22            else if (address == 8'd1) duty_cycle <= writedata[3:0];
23            else if (address == 8'd2) period <= writedata[31:0];
24        end
25    end
26
27    always_ff @(posedge clock) // envoie du duty_cycle ce qui nous permettra de le lire
28    if(read) readdata <= {28'b0, duty_cycle};
29    else readdata <= 32'b0;
30
31    always_ff @(posedge clock)
32    begin
33        if(!reset) begin //remise a 0 lorsqu'on appuie sur key[0]
34            LED <= 8'b0;
35            compteur <= 32'b0;
36        end
37
38        else begin
39            if (enable) begin // lorsque KEY[1] relacher alors on applique le duty_cycle sur les led
40                if (compteur <= period) compteur <= compteur + 1;
41                else compteur <= 32'b0;
42                if (compteur <= duty_cycle*period/15) LED <= 8'b11111111;
43                else LED <= 8'b0;
44            end
45
46            else LED <= 8'b0; //sinon les leds sont eteints
47
48        end
49    end
50 end
51
52
53 endmodule
54
```

Une fois ce composant créé on ajoute Comme au labo la mémoire, le cpu, le jtag_uart, button, switches et le composant qu'on vient de créer my_custom_ip_0 (voir figure1)

Et on génère nos fichiers avec QSYS

Puis on crée le code C qui va permettre d'envoyer et de recevoir les infos c'est l'interface entre tous nos composants. Ça va aussi nous permettre d'afficher le duty cycle sur la fenêtre de commande.

Le code C ci-dessus

```

#include <stdio.h>
#include "system.h"

int main(void)
{

    volatile int * mycustom    = (int*) MY_CUSTOM_IP_0_BASE;
    volatile int * button      = (int*) BUTTON_BASE;
    volatile int * switches    = (int*) SWITCHES_BASE;

    *(mycustom) = 1; //enable
    *(mycustom+2) = 100000; //periode

    while(1)
    {
        *(mycustom+1) = *switches; //duty cycle

        if(!*button)
        {
            *(mycustom) = 0; // mise du enable a 0
            printf(" Le duty cycle est de %d/%d \n", *(mycustom+1), 15); //lecture du duty cycle
            while(!*button)
            {} // temporisateur temps qu'on reste appuyer
            *(mycustom) = 1; //remise du enable a 1
        }
    }

    return 0;
}

```

Observation des résultats

```

Le duty cycle est de 0/15
Le duty cycle est de 1/15
Le duty cycle est de 3/15
Le duty cycle est de 7/15
Le duty cycle est de 15/15

```

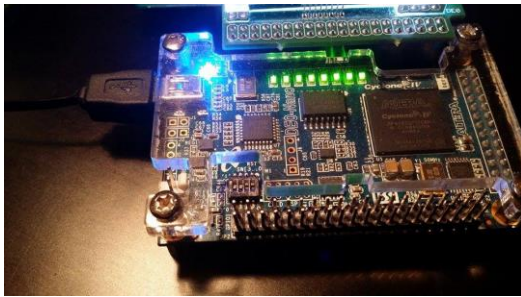
0/15



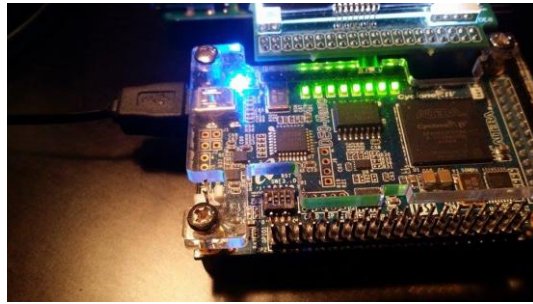
1/15



3/15



7/15



15/15

