UMONS
Université de Mons

Hardware-software

Envoi de données via le protocole UART

POLYTECH MONS

But du projet:

- Envoyer 8 bits
- Baud rate: 9600 bits/s
- Protocole UART
- Envoi de valeurs entre 0 et 255

Partie software (1/3)

- Code c:

```
int main(int argc, char *argv[]) {
int i=0;
char *x=0;
int y=0;
printf("execution %s",argv[0]);
for (i=1; i<argc;i++){
 printf("\narg%d=%s",i,argv[i]);
x=argv[i];
}

printf("\nNumber Of Arguments Passed: %d\n",argc);

h2p_lw_reg3_addr=virtual_base + ( ( unsigned long  )( ALT_LWFPGASLVS_OFST + PIO_REG3_BASE ) & ( unsigned long)( HW_REGS_MASK ) ); // on vient chercher l'adresse virtuelle du registre 3

y = atoi(x);


*(uint32_t *)h2p_lw_reg3_addr = y; // on vient écrire dans le registre 3
// *(uint32_t *)h2p_lw_reg2_addr = 1;
printf( "registre %d \n", *((uint32_t *)h2p_lw_reg3_addr)); // on lit l avaleur du registre et on la print
```

Partie software (2/3)

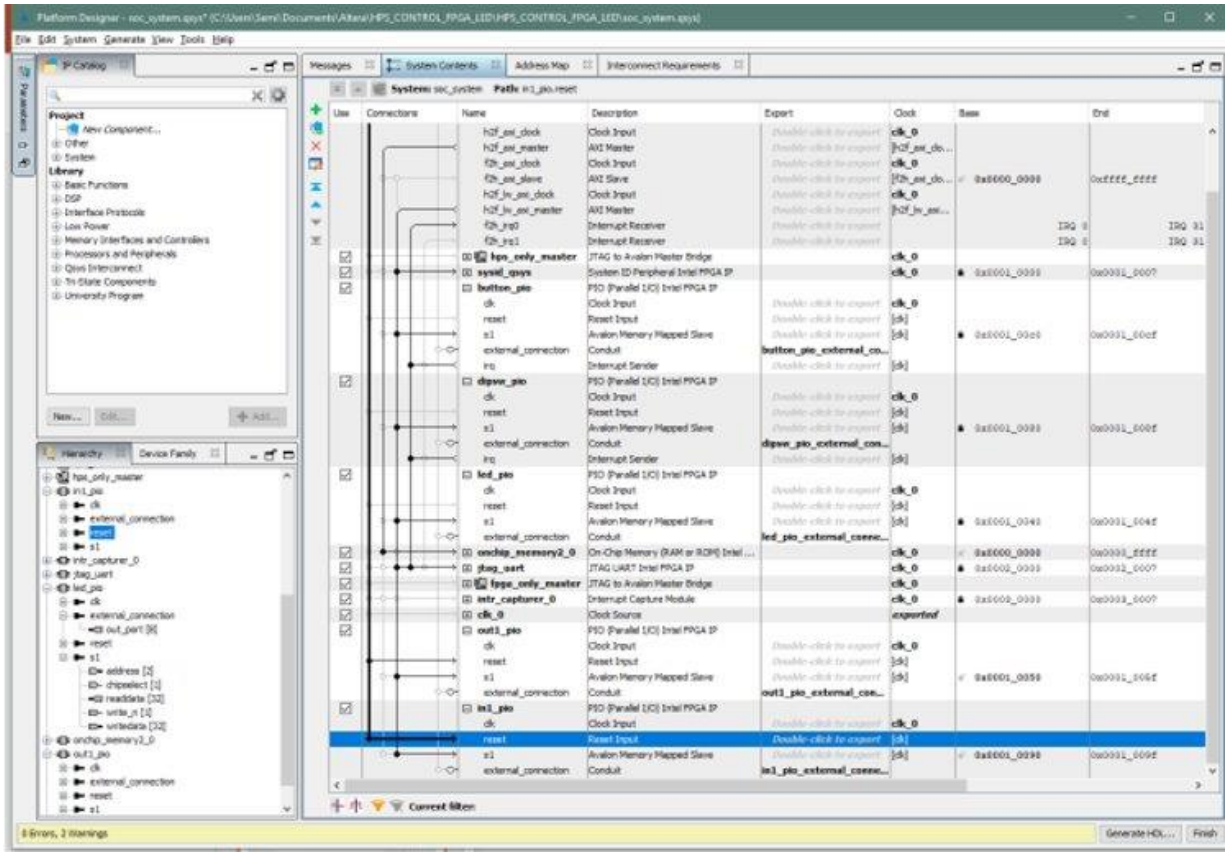
- VHDL

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use IEEE.numeric_std.all;


entity Simpleblock is
                port (
                        clk             : in std_logic;
                        rst_n   : in std_logic;
                        reg3        : in std_logic_vector(7 downto 0);
                        GPIO_0  : out std_logic
                        |
                );
end entity;

architecture RTL of Simpleblock is
        --constant MAX : natural := 10_000;
        constant MAX : natural := 5209;
        constant INDEX_MAX : natural := 8;
        type state_type is ( s0,s1,s2,s3 );
        signal state : state_type;
        signal data : natural := 0;
```

```vhdl
begin
        process (clk,rst_n,reg3)
                variable count : natural := 0;
                variable index : natural := 0;

        variable toto : std_logic_vector(7 downto 0);

        begin

                if rst_n = '0'then
                        count := 0;
                        index := 0;
                        state <= s0;
                        GPIO_0 <= '1'; -- Etat initial

                elsif rising_edge(clk) then
                        case state is
                                when s0 =>
                                        count := 0;
                                        index := 0;
                                                GPIO_0 <= '0'; -- bit de start

                                                data <= to_integer(unsigned(reg3));
                                                state <= s1;
                                when s1 =>
                                        count := count + 1;
                                        if count >= MAX then
                                                count := 0;
                                                toto := std_logic_vector(to_unsigned(data,8));
                                                GPIO_0 <= toto(index);
                                                --GPIO_0 <= std_logic_vector(to_unsigned(data,8))(index);

                                                index := index + 1;
                                                if index >= INDEX_MAX then
                                                        index := 0;
                                                        state <= s2;
                                                end if;
                                        end if;
                                when s2 =>
                                        index := 0;
                                        count := count + 1;
                                        if count =0 then
                                           state <= s0;
                                        --end if;
                                        elsif count >= MAX then
                                                count := 0;
                                                GPIO_0 <= '1'; -- bit de stop
                                                --state <= s3;
                                        end if;


--                              when s3 =>
--                                      index := 0;
--                                      count := count + 1;
--                                      if count >= MAX then
--                                              count := 0;
--                                              GPIO_0 <= '1';
--                                              state <= s0;
--                                      end if;
                                when others =>
                                        count := 0;
                                        index := 0;
                                        state <= s0;
                                        GPIO_0 <= '1';
                        end case;
                end if;
        end process;

end RTL;
```

Partie software (3/3)

- Plateform designer



Hardware (1/4)

- Ghrd

```
//mon block
Simpleblock Myblock(
        .clk        (FPGA_CLK1_50),
        .rst_n      (KEY[0]),
        .reg3          (add_to_reg3),
        .GPIO_0    (GPIO_0[1])

);
```

Hardware (2/4)

- Test bench:

```
architecture TB of Simpleblock_TB is

                    constant PERIOD : time := 20 ns;
        signal clk : std_logic := '0';
        signal rst_n : std_logic :='0';
        signal reg3 : std_logic_vector(7 downto 0);
        signal GPIO_0 : std_logic;

        component Simpleblock is
                port (

                 clk : in std_logic;
                 rst_n : in std_logic;
                 reg3 : in std_logic_vector(7 downto 0);
                 GPIO_0 : out std_logic

                 );

        end component;
```

```
begin

        Simpleblock_I : Simpleblock
                        port map(
                                clk => clk,
                                rst_n => rst_n,
                                reg3 => reg3,
                                GPIO_0 => GPIO_0

                                 );


        rst_n_P: process --on fait juste passer le reset de 0 à 1

        begin

                rst_n <= '0';

                                                wait for (PERIOD*5209);

                --wait for PERIOD/2;

                rst_n <='1';                    wait for (PERIOD*10*5209);

                -- wait for PERIOD/2;

        end process;


        clk_P : process -- on fait passer aussi la clk de 0 à 1

        begin

                clk <= '0';

                wait for PERIOD/2;

                clk <= '1';

                wait for PERIOD/2;

        end process;
```

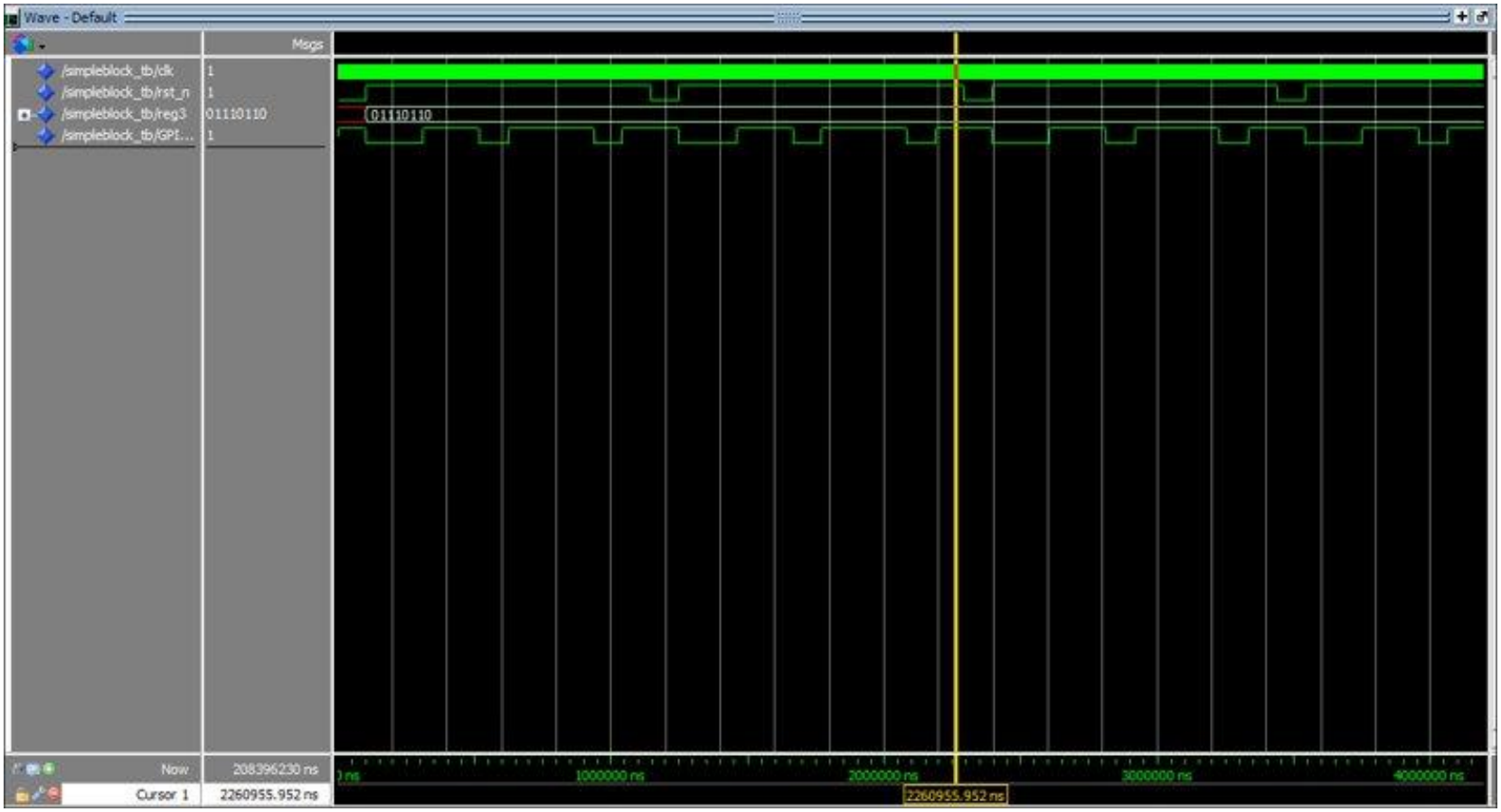
Hardware (3/4)

- Test bench:

```
        begin

        wait until rst_n = '1'; --
                              --reg3 <= "00000000";
       -- reg3 <= std_logic_vector(to_unsigned(0,8)); -- on ecrit dans le registre ; le 8 donne la taile du vecteur ici 8 bits,
                       --le 0 est la valeur que l'on ecrit dans le vecteur : 00000000

                              -- wait for PERIOD;
       -- wait for (PERIOD*5209);

                       reg3 <= "01010101";

        --reg3 <= std_logic_vector(to_unsigned(1,8)); -- 00000001

        wait for (PERIOD*10*5209);
                     -- reg3 <= "11111111";

        --reg3 <= std_logic_vector(to_unsigned(255,8)); --11111111

       -- wait for (PERIOD*5209);


end process;


end TB;
```

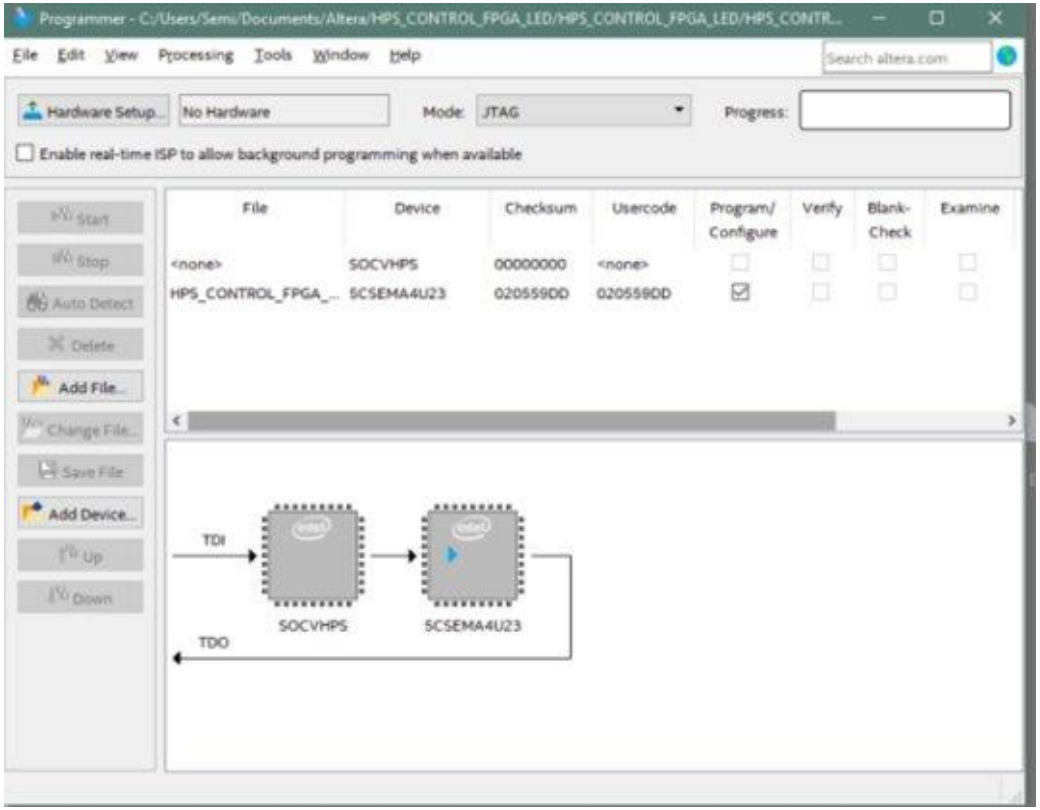
Hardware (4/4)

- Test bench:



Running the program step by step (1/4)

- Télécharger la partie Hardware dans le FPGA



Running the program step by step (2/4)

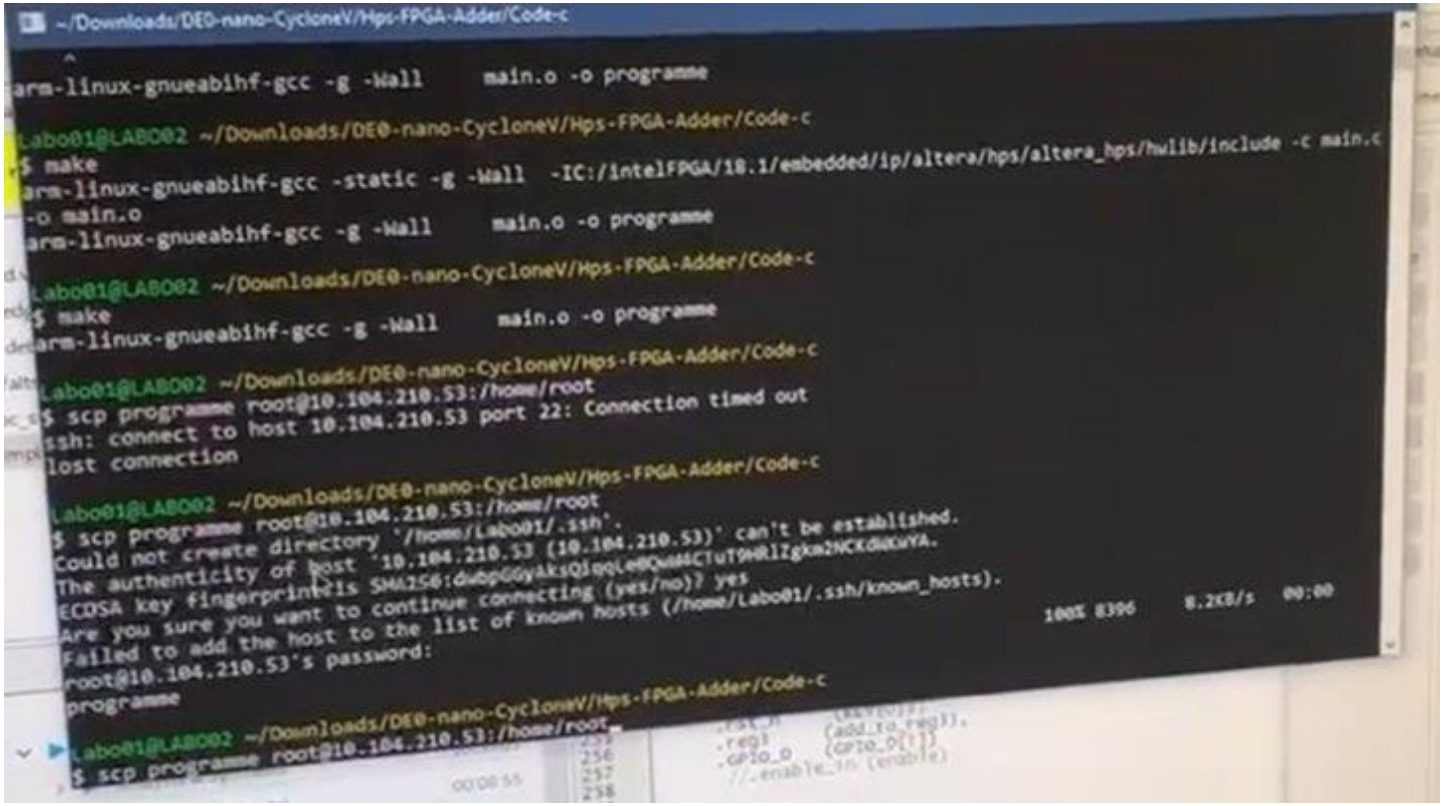
- Connexion à la carte pour récupérer son adresse via Putty:

COM5 - PuTTY

```
Poky 8.0 (Yocto Project 1.3 Reference Distro) 1.3
 ttyS0

socfpga login: root
root@socfpga:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 96:06:5d:05:15:68
          inet addr:10.104.210.48  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:50 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8950 (8.7 KiB)  TX bytes:1670 (1.6 KiB)
          Interrupt:152

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@socfpga:~#
```

Running the program step by step (3/4)

- Téléchargement de l'exécutable via la fenêtre SOC shell:



Running the program step by step (4/4)

Vérification que l'exécutable est bien sur la carte plus exécution du programme et observation des résultats:

```
root@socfpga:~# ./programme 0
execution ./programme
arg1=0
Number Of Arguments Passed: 2
registre 0
root@socfpga:~#
```