Getting started with the Arty board (reomote lab)

Introduction Arty Board

Quick introduction to FPGAs

An FPGA is an integrated circuit designed to be configured multiple times after being placed onto an electronic board; normally it is not able to do nothing useful before the programming step. Our FPGA needs to be programmed at every power-up. To do this we need to generate a special file called **bitstream** (.bit extension in our case)

Two set of files should be written by the user:

1. HDL files (vhdl in our case, .vhd extension)
2. pin mapping file (tcl file in our case, .tcl extension

It is **mandatory** that the pin names in the pin mapping file match the port name of the top level entity used in the same project.

Remote Lab

How to find your board

15 Arty7 Boards (once per group) are connected to the server *lxilinx1*; each board exposes **two** USB devices to the server, hence if you try to list the USB connections you get the following output.

The command:

```
groups
```

```
Arty2
```

tell you at which groups you belong: you may find to belong to ArtyN N = 1..14

```
ls -l /dev/ttyUSB*
```

shows all the USB connections (devices).

```
crw-rw---- 1 root Arty1  188,  1 Dec 24 15:01 /dev/ttyUSB1
crw-rw---- 1 root Arty13 188, 10 Dec 24 15:01 /dev/ttyUSB10
```

```
crw-rw---- 1 root Arty13 188, 11 Dec 24 15:01 /dev/ttyUSB11
crw-rw---- 1 root Arty3  188, 12 Dec 24 15:01 /dev/ttyUSB12
crw-rw---- 1 root Arty3  188, 13 Dec 24 15:01 /dev/ttyUSB13
crw-rw---- 1 root Arty8  188, 14 Dec 24 15:01 /dev/ttyUSB14
crw-rw---- 1 root Arty8  188, 15 Dec 24 15:01 /dev/ttyUSB15
crw-rw---- 1 root Arty9  188, 16 Dec 24 15:01 /dev/ttyUSB16
crw-rw---- 1 root Arty9  188, 17 Dec 24 15:01 /dev/ttyUSB17
crw-rw---- 1 root Arty10 188, 18 Dec 24 15:01 /dev/ttyUSB18
crw-rw---- 1 root Arty10 188, 19 Dec 24 15:01 /dev/ttyUSB19
crw-rw---- 1 root Arty11 188,  2 Dec 24 15:01 /dev/ttyUSB2
crw-rw---- 1 root Arty4  188, 20 Dec 24 15:01 /dev/ttyUSB20
crw-rw---- 1 root Arty4  188, 21 Dec 24 15:01 /dev/ttyUSB21
crw-rw---- 1 root Arty5  188, 22 Dec 24 15:01 /dev/ttyUSB22
crw-rw---- 1 root Arty5  188, 23 Dec 24 15:01 /dev/ttyUSB23
crw-rw---- 1 root Arty6  188, 24 Dec 24 15:01 /dev/ttyUSB24
crw-rw---- 1 root Arty6  188, 25 Dec 24 15:01 /dev/ttyUSB25
crw-rw---- 1 root Arty7  188, 26 Dec 24 15:01 /dev/ttyUSB26
crw-rw---- 1 root Arty7  188, 27 Dec 24 15:01 /dev/ttyUSB27
crw-rw---- 1 root Arty11 188,  3 Dec 24 15:01 /dev/ttyUSB3
crw-rw---- 1 root Arty12 188,  4 Dec 24 15:01 /dev/ttyUSB4
crw-rw---- 1 root Arty12 188,  5 Dec 24 15:01 /dev/ttyUSB5
crw-rw---- 1 root Arty14 188,  6 Dec 24 15:01 /dev/ttyUSB6
crw-rw---- 1 root Arty14 188,  7 Dec 24 15:01 /dev/ttyUSB7
crw-rw---- 1 root Arty2  188,  9 Dec 30 15:47 /dev/ttyUSB9
```

You can gain the access only for the devices belonging to your group

Normally **two** of these are associated to a single group: one device is devoted to *download* the bitstream, the other is devoted to connect to the board serial port.

One of these two "disappear" when you try to configure the board the first time: you have to use the other as **serial port connection** (see below)

Template project

The project template is downloadable here:

https://baltig.infn.it/abergnol/arty_template_project/-/archive/a100/arty_template_project-a100.zip

(only for groups Arty1 and Arty15 the link is the following: https://baltig.infn.it/abergnol/arty_templat e_project/-/archive/a100/arty_template_project-a35.zip)

The significant files are ready available (top.vhd, pins.tcl). The pins.tcl file contains all the available pin specification of the board, most of these are commented out. If you want to use them in your project just uncomment the corresponding directive and pay attention to use the same pin name in the top level entity port.

if you are logged in the *lxilinx1* server just type

```
wget https://baltig.infn.it/abergnol/arty_template_project/-/archive/master/arty_template_project-master.zip

unzip arty_template_project-master.zip

cd arty_template_project-master
```

in order to create the correct development environment variables setup type:

```
source  /tools/Xilinx/Vivado/2018.3/settings64.sh
```

to generate the bitstream type:

```
make
```

to program the FPGA type:

```
make program_fpga
```

the simple *.vhd* file placed in the template make the project ready to be used but simply connect the uart output to the uart input with a *loop wire* :

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

entity top is
  port (
    CLK100MHZ    : in  std_logic;
    uart_rxd_out : out std_logic;
    uart_txd_in  : in  std_logic);
end entity top;

architecture str of top is
```

```
begin
  process(CLK100MHZ) is
  begin
    if rising_edge(CLK100MHZ) then
      uart_rxd_out <= uart_txd_in;
    end if;
  end process;

end architecture str;
```

clock pin directives need to be uncommented:

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports {
↪  CLK100MHZ }]; #IO_L12P_T1_MRCC_35 Sch=gclk[100]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5}
↪  [get_ports { CLK100MHZ }];
```

uart (both TX and RX) pin directives need to be uncommented

```
## USB-UART Interface
set_property -dict { PACKAGE_PIN D10   IOSTANDARD LVCMOS33 } [get_ports {
↪  uart_rxd_out }]; #IO_L19N_T3_VREF_16 Sch=uart_rxd_out
set_property -dict { PACKAGE_PIN A9    IOSTANDARD LVCMOS33 } [get_ports {
↪  uart_txd_in }]; #IO_L14N_T2_SRCC_16 Sch=uart_txd_in
```

Conncecting to the board via USB-serial terminal

screen is an utility that allows to connect to the serial port, send and receive characters:

look for the USB device that belongs to your group and type:

```
screen /dev/ttyUSB<your number> 115200
```

You may see an *echo like* behaviour: every character sent with the keyboard is sent back into the screen

TIPS: to exit from screen, type *C-a k* (control-a k)

https://www.gnu.org/software/screen/manual/html_node/Default-Key-Bindings.html

Implement an uart transmitter

you can change the vhd file(s) of the template project with your own code. Try to implement an UART transmitter placing your source code in the *src* directory.

Some example code could be picked up from here:

wget https://baltig.infn.it/abergnol/vivado_non_prj_simple/-/raw/master/src/top.vhd

This top level block send the “hello” string one character per second to the UART triansmitter block.

Add your transmitter, and create the *one_second_generator* entity, modify the pin.tcl accordingly.

Other Useful tool

The **micro** editor is installed in the server. https://micro-editor.github.io/

type

```
micro your-file-name.vhd
```

to open and edit a file.

Type:

1. C-q to exit
2. C-e to enter the command help