

Getting started with the Arty board (remote 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, 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, 28 Dec 24 15:01 /dev/ttyUSB28
crw-rw---- 1 root Arty12 188, 29 Dec 24 15:01 /dev/ttyUSB29
crw-rw---- 1 root Arty12 188, 30 Dec 24 15:01 /dev/ttyUSB30
crw-rw---- 1 root Arty14 188, 31 Dec 24 15:01 /dev/ttyUSB31
crw-rw---- 1 root Arty14 188, 32 Dec 24 15:01 /dev/ttyUSB32
crw-rw---- 1 root Arty2  188, 33 Dec 24 15:01 /dev/ttyUSB33
crw-rw---- 1 root Arty2  188, 34 Dec 24 15:01 /dev/ttyUSB34
crw-rw---- 1 root Arty2  188, 35 Dec 24 15:01 /dev/ttyUSB35
crw-rw---- 1 root Arty2  188, 36 Dec 24 15:01 /dev/ttyUSB36
crw-rw---- 1 root Arty2  188, 37 Dec 24 15:01 /dev/ttyUSB37
crw-rw---- 1 root Arty2  188, 38 Dec 24 15:01 /dev/ttyUSB38
crw-rw---- 1 root Arty2  188, 39 Dec 24 15:01 /dev/ttyUSB39
crw-rw---- 1 root Arty2  188, 40 Dec 24 15:01 /dev/ttyUSB40
crw-rw---- 1 root Arty2  188, 41 Dec 24 15:01 /dev/ttyUSB41
crw-rw---- 1 root Arty2  188, 42 Dec 24 15:01 /dev/ttyUSB42
crw-rw---- 1 root Arty2  188, 43 Dec 24 15:01 /dev/ttyUSB43
crw-rw---- 1 root Arty2  188, 44 Dec 24 15:01 /dev/ttyUSB44
crw-rw---- 1 root Arty2  188, 45 Dec 24 15:01 /dev/ttyUSB45
crw-rw---- 1 root Arty2  188, 46 Dec 24 15:01 /dev/ttyUSB46
crw-rw---- 1 root Arty2  188, 47 Dec 24 15:01 /dev/ttyUSB47
crw-rw---- 1 root Arty2  188, 48 Dec 24 15:01 /dev/ttyUSB48
crw-rw---- 1 root Arty2  188, 49 Dec 24 15:01 /dev/ttyUSB49
crw-rw---- 1 root Arty2  188, 50 Dec 24 15:01 /dev/ttyUSB50
crw-rw---- 1 root Arty2  188, 51 Dec 24 15:01 /dev/ttyUSB51
crw-rw---- 1 root Arty2  188, 52 Dec 24 15:01 /dev/ttyUSB52
crw-rw---- 1 root Arty2  188, 53 Dec 24 15:01 /dev/ttyUSB53
crw-rw---- 1 root Arty2  188, 54 Dec 24 15:01 /dev/ttyUSB54
crw-rw---- 1 root Arty2  188, 55 Dec 24 15:01 /dev/ttyUSB55
crw-rw---- 1 root Arty2  188, 56 Dec 24 15:01 /dev/ttyUSB56
crw-rw---- 1 root Arty2  188, 57 Dec 24 15:01 /dev/ttyUSB57
crw-rw---- 1 root Arty2  188, 58 Dec 24 15:01 /dev/ttyUSB58
crw-rw---- 1 root Arty2  188, 59 Dec 24 15:01 /dev/ttyUSB59
crw-rw---- 1 root Arty2  188, 60 Dec 24 15:01 /dev/ttyUSB60
crw-rw---- 1 root Arty2  188, 61 Dec 24 15:01 /dev/ttyUSB61
crw-rw---- 1 root Arty2  188, 62 Dec 24 15:01 /dev/ttyUSB62
crw-rw---- 1 root Arty2  188, 63 Dec 24 15:01 /dev/ttyUSB63
crw-rw---- 1 root Arty2  188, 64 Dec 24 15:01 /dev/ttyUSB64
crw-rw---- 1 root Arty2  188, 65 Dec 24 15:01 /dev/ttyUSB65
crw-rw---- 1 root Arty2  188, 66 Dec 24 15:01 /dev/ttyUSB66
crw-rw---- 1 root Arty2  188, 67 Dec 24 15:01 /dev/ttyUSB67
crw-rw---- 1 root Arty2  188, 68 Dec 24 15:01 /dev/ttyUSB68
crw-rw---- 1 root Arty2  188, 69 Dec 24 15:01 /dev/ttyUSB69
crw-rw---- 1 root Arty2  188, 70 Dec 24 15:01 /dev/ttyUSB70
crw-rw---- 1 root Arty2  188, 71 Dec 24 15:01 /dev/ttyUSB71
crw-rw---- 1 root Arty2  188, 72 Dec 24 15:01 /dev/ttyUSB72
crw-rw---- 1 root Arty2  188, 73 Dec 24 15:01 /dev/ttyUSB73
crw-rw---- 1 root Arty2  188, 74 Dec 24 15:01 /dev/ttyUSB74
crw-rw---- 1 root Arty2  188, 75 Dec 24 15:01 /dev/ttyUSB75
crw-rw---- 1 root Arty2  188, 76 Dec 24 15:01 /dev/ttyUSB76
crw-rw---- 1 root Arty2  188, 77 Dec 24 15:01 /dev/ttyUSB77
crw-rw---- 1 root Arty2  188, 78 Dec 24 15:01 /dev/ttyUSB78
crw-rw---- 1 root Arty2  188, 79 Dec 24 15:01 /dev/ttyUSB79
crw-rw---- 1 root Arty2  188, 80 Dec 24 15:01 /dev/ttyUSB80
crw-rw---- 1 root Arty2  188, 81 Dec 24 15:01 /dev/ttyUSB81
crw-rw---- 1 root Arty2  188, 82 Dec 24 15:01 /dev/ttyUSB82
crw-rw---- 1 root Arty2  188, 83 Dec 24 15:01 /dev/ttyUSB83
crw-rw---- 1 root Arty2  188, 84 Dec 24 15:01 /dev/ttyUSB84
crw-rw---- 1 root Arty2  188, 85 Dec 24 15:01 /dev/ttyUSB85
crw-rw---- 1 root Arty2  188, 86 Dec 24 15:01 /dev/ttyUSB86
crw-rw---- 1 root Arty2  188, 87 Dec 24 15:01 /dev/ttyUSB87
crw-rw---- 1 root Arty2  188, 88 Dec 24 15:01 /dev/ttyUSB88
crw-rw---- 1 root Arty2  188, 89 Dec 24 15:01 /dev/ttyUSB89
crw-rw---- 1 root Arty2  188, 90 Dec 24 15:01 /dev/ttyUSB90
crw-rw---- 1 root Arty2  188, 91 Dec 24 15:01 /dev/ttyUSB91
crw-rw---- 1 root Arty2  188, 92 Dec 24 15:01 /dev/ttyUSB92
crw-rw---- 1 root Arty2  188, 93 Dec 24 15:01 /dev/ttyUSB93
crw-rw---- 1 root Arty2  188, 94 Dec 24 15:01 /dev/ttyUSB94
crw-rw---- 1 root Arty2  188, 95 Dec 24 15:01 /dev/ttyUSB95
crw-rw---- 1 root Arty2  188, 96 Dec 24 15:01 /dev/ttyUSB96
crw-rw---- 1 root Arty2  188, 97 Dec 24 15:01 /dev/ttyUSB97
crw-rw---- 1 root Arty2  188, 98 Dec 24 15:01 /dev/ttyUSB98
crw-rw---- 1 root Arty2  188, 99 Dec 24 15:01 /dev/ttyUSB99
```

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_template_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      IO_STANDARD 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      IO_STANDARD LVCMOS33 } [get_ports {
  ↳ uart_rxd_out }];
#IO_L19N_T3_VREF_16 Sch=uart_rxd_out
set_property -dict { PACKAGE_PIN A9      IO_STANDARD LVCMOS33 } [get_ports {
  ↳ uart_txd_in }];
#IO_L14N_T2_SRCC_16 Sch=uart_txd_in

```

Connnecting 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 transmitter 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