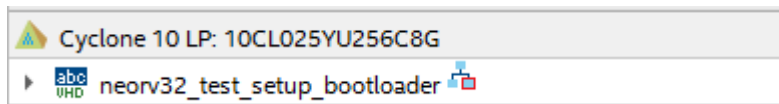


These instructions are useful to implement NEORV32 on CYC1000 FPGA board.

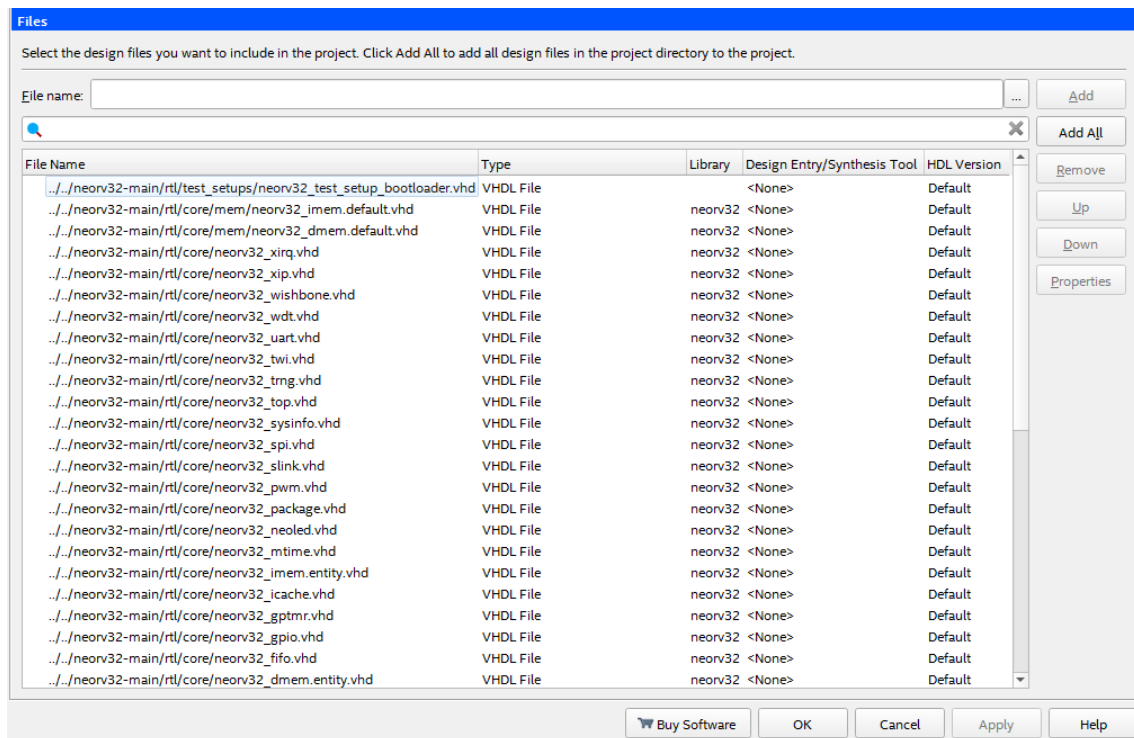
It is needed Quartus tool to synthesize the RISC-V core on the Cyclone-10 FPGA. You need the Lite edition (free) for Cyclone-10. It can be downloaded from:

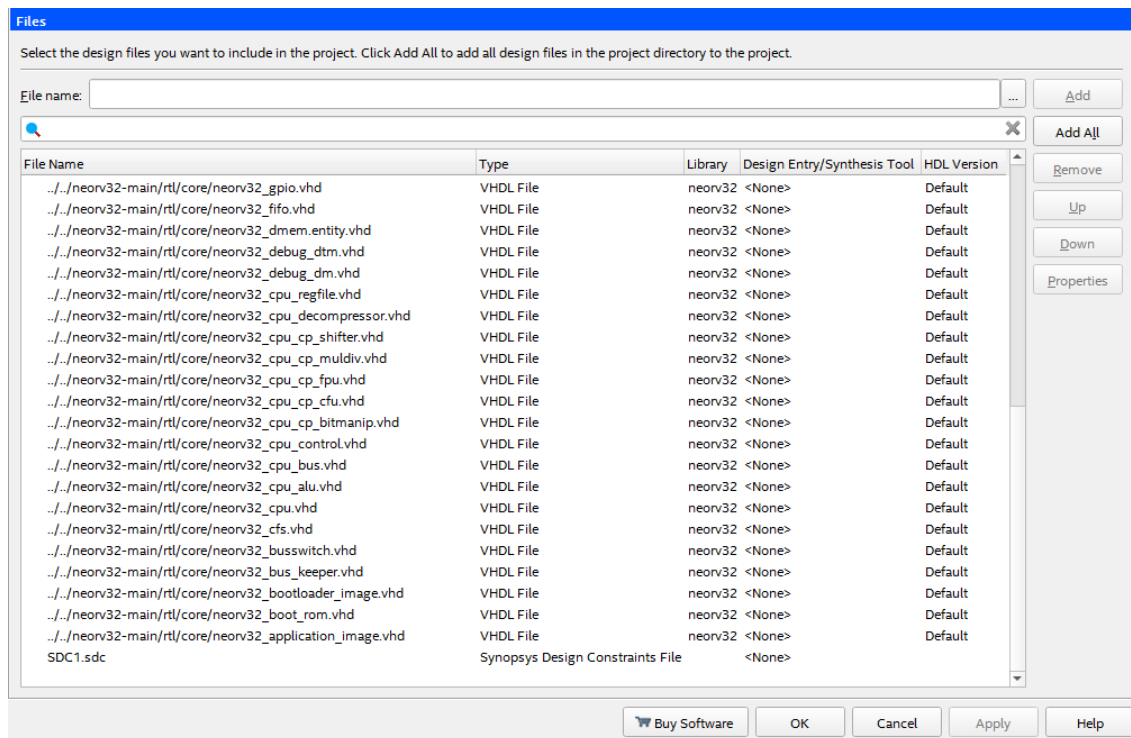
<https://www.intel.la/content/www/xl/es/software/programmable/quartus-prime/download.html>

After installation, create a project for the following FPGA:



The file neorv32-main.zip contains the NEORV32 VHDL code used. Add to Quartus the following files and assign the library neorv32 to all files except the top one (neorv32_test_setup_bootloader.vhd) as shown in the following pictures:





Add to Quartus the constraints files (SDC1.sdc) and pin assignment in QSF file. Probably you will have your own QSF file, so add to your file the following assignments:

```
set_location_assignment PIN_M2 -to clk_i
```

```
set_location_assignment PIN_T7 -to uart0_txd_o
```

```
set_location_assignment PIN_R7 -to uart0_rxd_i
```

```
set_location_assignment PIN_N6 -to rstn_i
```

```
set_location_assignment PIN_M6 -to gpio_o[0]
```

```
set_location_assignment PIN_T4 -to gpio_o[1]
```

```
set_location_assignment PIN_T3 -to gpio_o[2]
```

```
set_location_assignment PIN_R3 -to gpio_o[3]
```

```
set_location_assignment PIN_T2 -to gpio_o[4]
```

```
set_location_assignment PIN_R4 -to gpio_o[5]
```

```
set_location_assignment PIN_N5 -to gpio_o[6]
```

```
set_location_assignment PIN_N3 -to gpio_o[7]
```

```
set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to clk_i
```

```
set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to rstn_i
```

```
set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to uart0_rxd_i
```

```

set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to uart0_txd_o

set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to gpio_o[0]

set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to gpio_o[1]

set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to gpio_o[2]

set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to gpio_o[3]

set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to gpio_o[4]

set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to gpio_o[5]

set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to gpio_o[6]

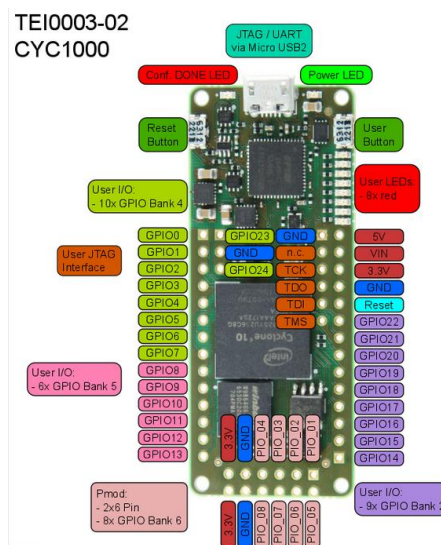
set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -to gpio_o[7]

```

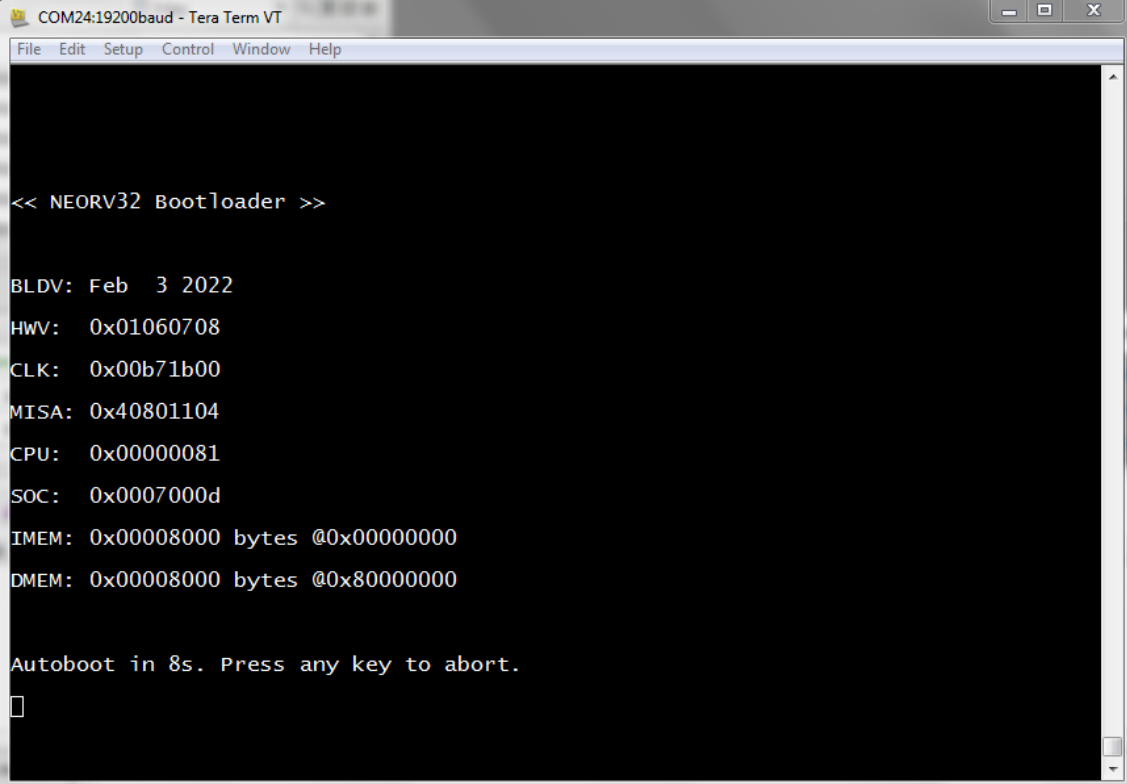
After compilation (synthesis and P&R), you will have the bitstream file. Program your FPGA by following the CYC1000 user guide instructions, available in the following link:

https://shop.trenz-electronic.de/trenzdownloads/Trenz_Electronic/Modules_and_Module_Carriers/2.5x6.15/TEI0003/REV02/Documents/CYC1000%20User%20Guide.pdf

If everything was fine, you will have a RISC-V core running in the CYC1000 board. So, the next step is to upload your code. NEORV32 contains a bootloader, so open a COM port using a terminal like [Tera Term](#) with 19200 bauds and 8N1 configuration. After that, reset the CYC1000 board by pushing the user button (the reset button program the FPGA, so the uploaded design will be lost and you will need to program the FPGA again and to reopen the COM port). The user button has been assign to the NEORV32 reset, so this button must be pressed to reset the RISC-V core.



After NEORV32 reset, the following message from the bootloader will be presented in the terminal. Press any key to upload your code.



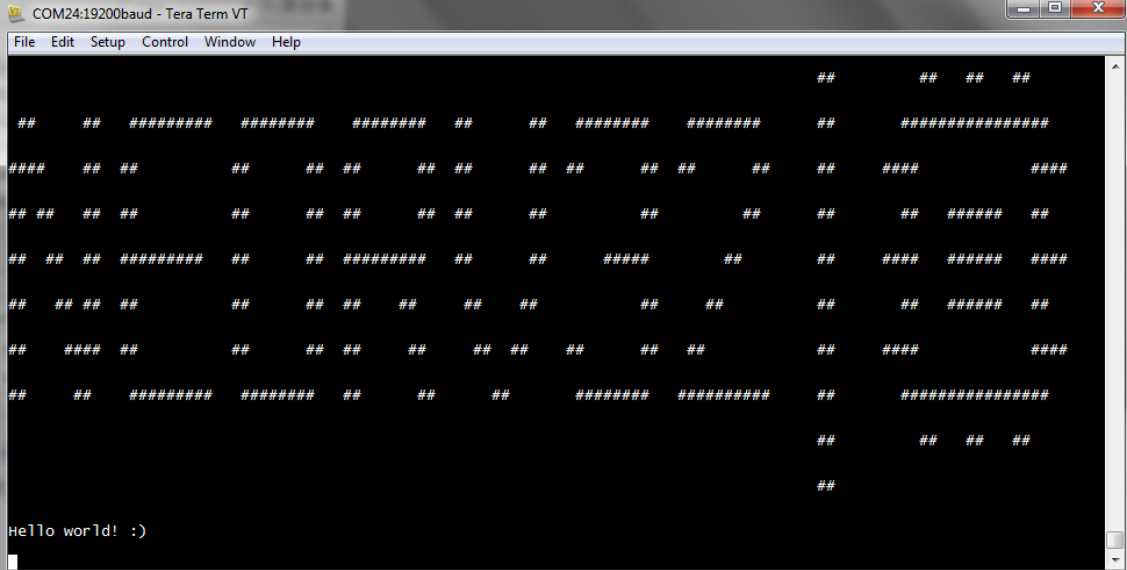
```
COM24:19200baud - Tera Term VT
File Edit Setup Control Window Help

<< NEORV32 Bootloader >>

BLDV: Feb  3 2022
HWV:  0x01060708
CLK:  0x00b71b00
MISA: 0x40801104
CPU:   0x00000081
SOC:   0x0007000d
IMEM: 0x00008000 bytes @0x00000000
DMEM: 0x00008000 bytes @0x80000000

Autoboot in 8s. Press any key to abort.
█
```

In the repository you will find a binary code for the hello world example. Press u to upload this file by using “sent file” in Tera Term (activate the binary option). After upload your code, execute the code and you will see the NEORV32 hello world.



```
COM24:19200baud - Tera Term VT
File Edit Setup Control Window Help

#####
##  ##  #####  #####  ##  ##  #####  #####  ##  #####
####  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ####  ####
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  #####  ##
##  ##  ##  #####  ##  ##  #####  ##  ##  #####  ##  ##  #####  #####
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  #####  ##
##  #####  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  #####  #####
##  ##  #####  #####  ##  ##  ##  #####  #####  ##  #####
#####
##  ##  ##  ##
##

hello world! :)
█
```

The link for the prebuilt RISC-V GCC toolchains to compile your own code is available here:

<https://github.com/stnolting/riscv-gcc-prebuilt>

Enjoy RISC-V NEORV32 core!

;~)