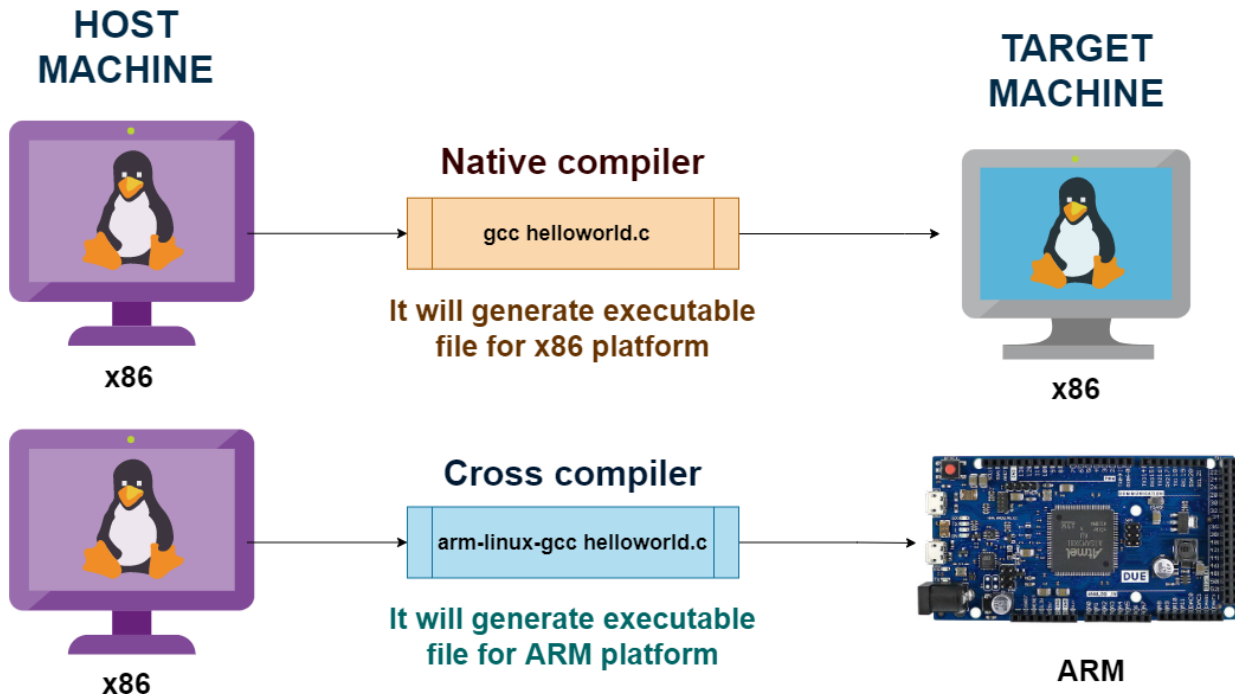


# DE10-Nano development board HPS and Linux

# Cross-compilation

- The host machine where we run the compiler (Linux-VM under WIN) and the target machine for which we make the compilation (DE10-Nano) both are different platforms: this is called cross-platform-compilation or cross-compilation



# Creating a Project Folder

- A project usually includes the design files .c/.h and a make file. These files are generally stored under the same folder.
- So, it is suggested to create a project folder where you can store your design file and make file.
- Developer can create such a folder in the “Home” directory

# Environment variables

- Some environment variables are already available:
- `osboxes@osboxes:~$ echo $SOCEDS_DEST_ROOT`

`/home/osboxes/intelFPGA_lite/20.1/embedded`

- `osboxes@osboxes:~$ echo $CROSS_COMPILE`

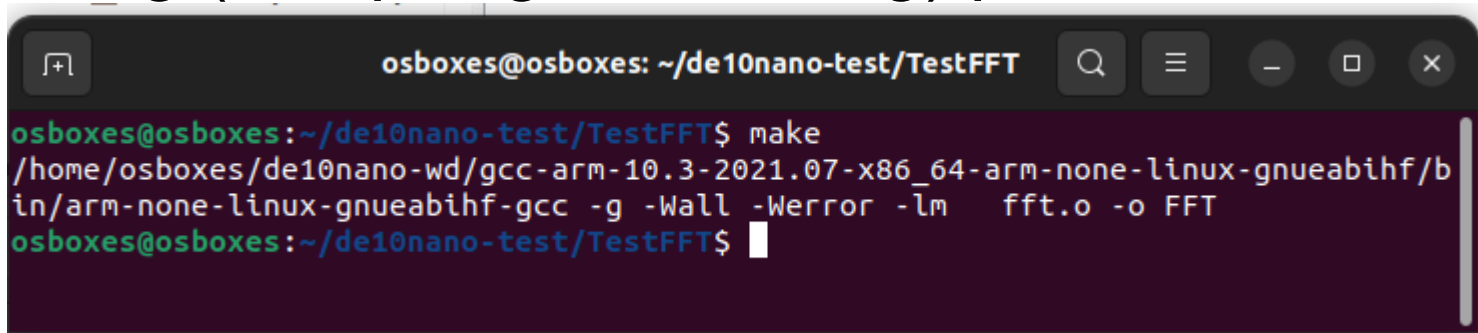
`/home/osboxes/de10nano-wd/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-`

# Make file

- A makefile is required for the Altera SoC EDS in order for it to know how to compile and link your project
- Inside the makefile, the "TARGET" variable defines the output file name.
- The makefile also specifies which compiler to use, in this case we use ARM gcc cross compiler.
- The gcc compiler parameter “-I\${SOCEDS\_DEST\_ROOT}/path-to-include” defines the searching path for the gcc including header files.
- The gcc compiler parameter “-l” defines additional library to be linked

# Execute the Make file

- In the Command Shell, please use the Linux “cd” command to change current directory to your project folder.
- Then, type the “make” command to start the building (compiling and linking) process.

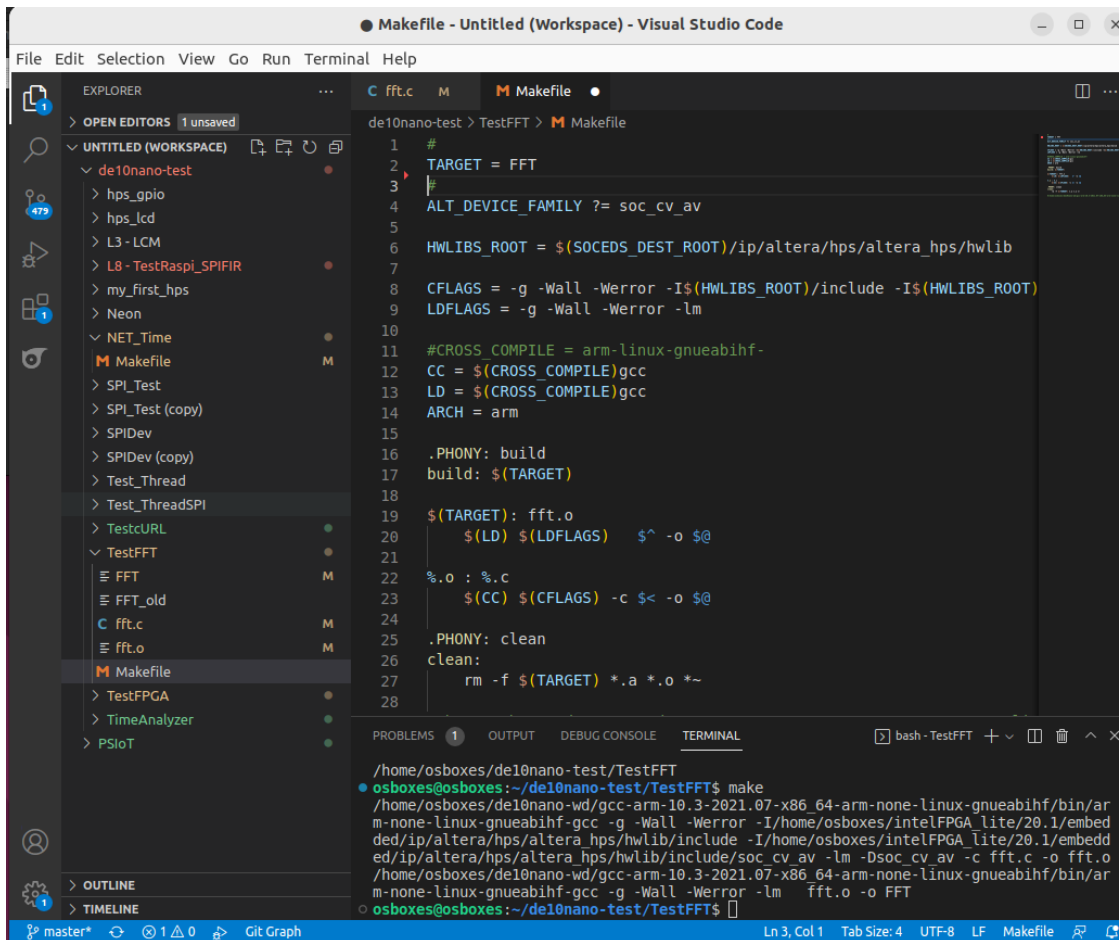
A screenshot of a terminal window with a dark background. The title bar at the top shows 'osboxes@osboxes: ~/de10nano-test/TestFFT'. The terminal text shows the user typing 'make' at the prompt 'osboxes@osboxes:~/de10nano-test/TestFFT\$'. The output shows the compiler path and flags: '/home/osboxes/de10nano-wd/gcc-arm-10.3-2021.07-x86\_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-gcc -g -Wall -Werror -lm fft.o -o FFT'. The prompt returns to 'osboxes@osboxes:~/de10nano-test/TestFFT\$' with a cursor.

```
osboxes@osboxes:~/de10nano-test/TestFFT$ make
/home/osboxes/de10nano-wd/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-gcc -g -Wall -Werror -lm  fft.o -o FFT
osboxes@osboxes:~/de10nano-test/TestFFT$
```

- After the building process is finished, type "ls" to list all the files in the current directory: the executable should be there!

# Execute the Make file

- The “terminal” tab of VS Code can be used to start the makefile as well!



The screenshot shows the Visual Studio Code interface with a workspace named 'Makefile - Untitled (Workspace)'. The Explorer panel on the left shows a project structure with a folder 'de10nano-test' containing files like 'hps\_gpio', 'hps\_lcd', 'L3-LCM', 'L8-TestRaspi\_SPIFIR', 'my\_first\_hps', 'Neon', 'NET\_Time', 'Makefile', 'SPL\_Test', 'SPL\_Test (copy)', 'SPIDev', 'SPIDev (copy)', 'Test\_Thread', 'Test\_ThreadSPI', 'TestURL', 'TestFFT', 'FFT', 'FFT\_old', 'fft.c', 'fft.o', 'TestFPGA', 'TimeAnalyzer', and 'PSIoT'. The Makefile is open in the editor, showing the following content:

```
1 #
2 TARGET = FFT
3 #
4 ALT_DEVICE_FAMILY ?= soc_cv_av
5
6 HWLIBS_ROOT = $(SOCEDS_DEST_ROOT)/ip/altera/hps/altera_hps/hwlib
7
8 CFLAGS = -g -Wall -Werror -I$(HWLIBS_ROOT)/include -I$(HWLIBS_ROOT)
9 LDFLAGS = -g -Wall -Werror -lm
10
11 #CROSS_COMPILE = arm-linux-gnueabi-
12 CC = $(CROSS_COMPILE)gcc
13 LD = $(CROSS_COMPILE)gcc
14 ARCH = arm
15
16 .PHONY: build
17 build: $(TARGET)
18
19 $(TARGET): fft.o
20     $(LD) $(LDFLAGS)  $^ -o $@
21
22 %.o : %.c
23     $(CC) $(CFLAGS) -c $< -o $@
24
25 .PHONY: clean
26 clean:
27     rm -f $(TARGET) *.a *.o *~
28
```

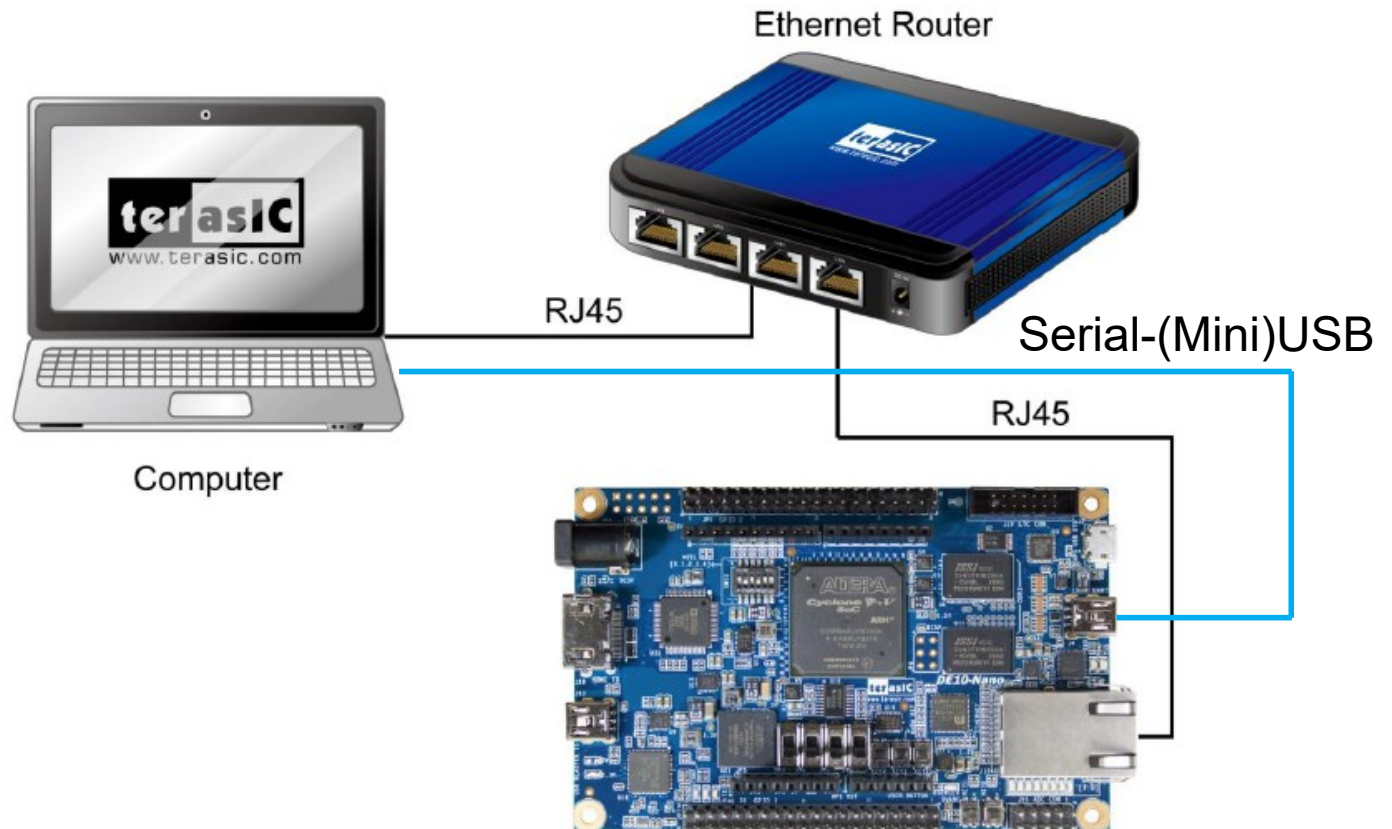
The Terminal panel at the bottom shows the command being executed:

```
/home/osboxes/de10nano-test/TestFFT
osboxes@osboxes:~/de10nano-test/TestFFT$ make
/home/osboxes/de10nano-wd/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-gcc -g -Wall -Werror -I/home/osboxes/intelFPGA_lite/20.1/embedded/ip/altera/hps/altera_hps/hwlib/include -I/home/osboxes/intelFPGA_lite/20.1/embedded/ip/altera/hps/altera_hps/hwlib/include/soc_cv_av -lm -Dsoc_cv_av -c fft.c -o fft.o
/home/osboxes/de10nano-wd/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-gcc -g -Wall -Werror -lm  fft.o -o FFT
osboxes@osboxes:~/de10nano-test/TestFFT$
```

Check that the current folder is the right one (type «pwd»)

# Copy the executable

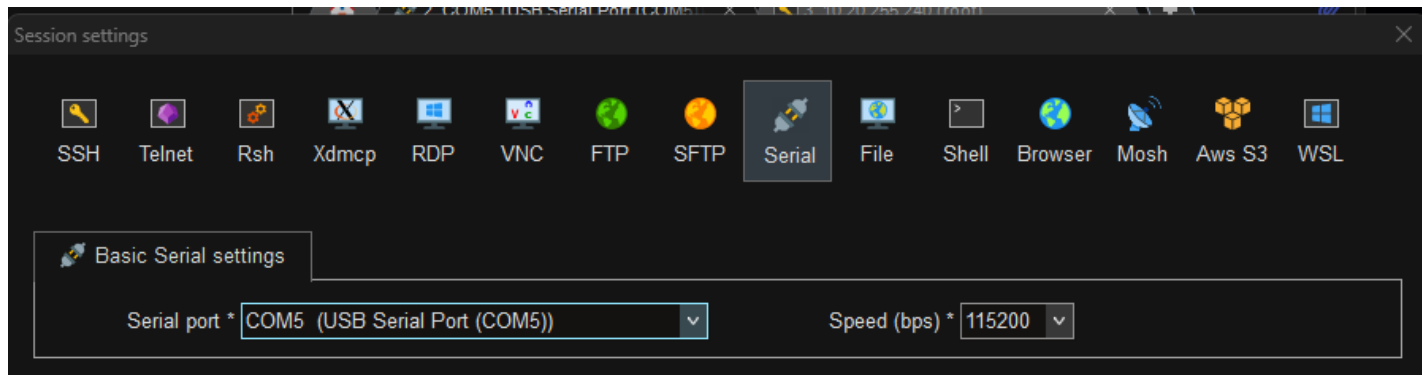
## ■ Setup





# Copy the executable

- Discover the IP address: use Moba Xterm to create a «Serial» session: 115200 bps



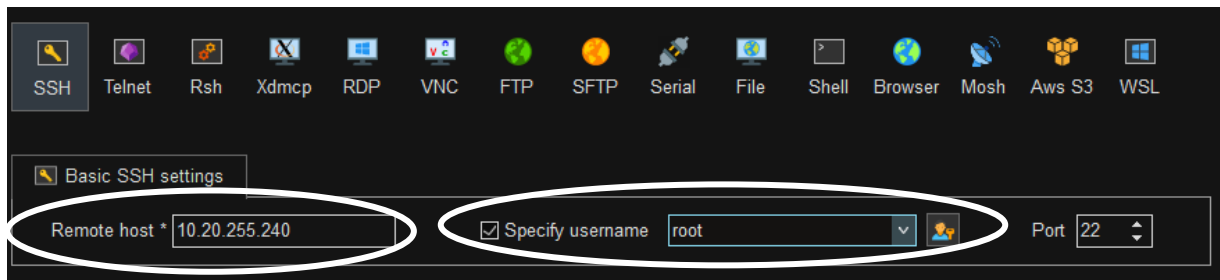
- Run «ifconfig»:

```
root@socfpga:~# ifconfig
eth0      Link encap:Ethernet  HWaddr c2:61:91:09:02:ea
          inet addr:10.20.255.240  Bcast:10.20.255.255  Mask:255.255.0.0
```

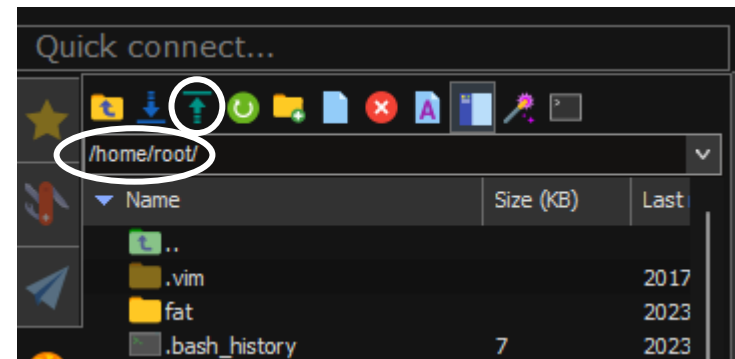
- If needed, exec «udhcpc» to make a DHCP request

# Copy the executable

- Use Moba Xterm to start a «SSH» session:  
Remote host: the IP-addr previously find  
Username: root
- You are redirected to: /home/root

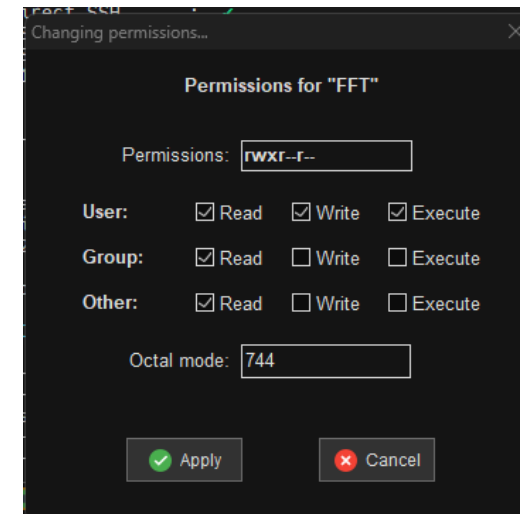
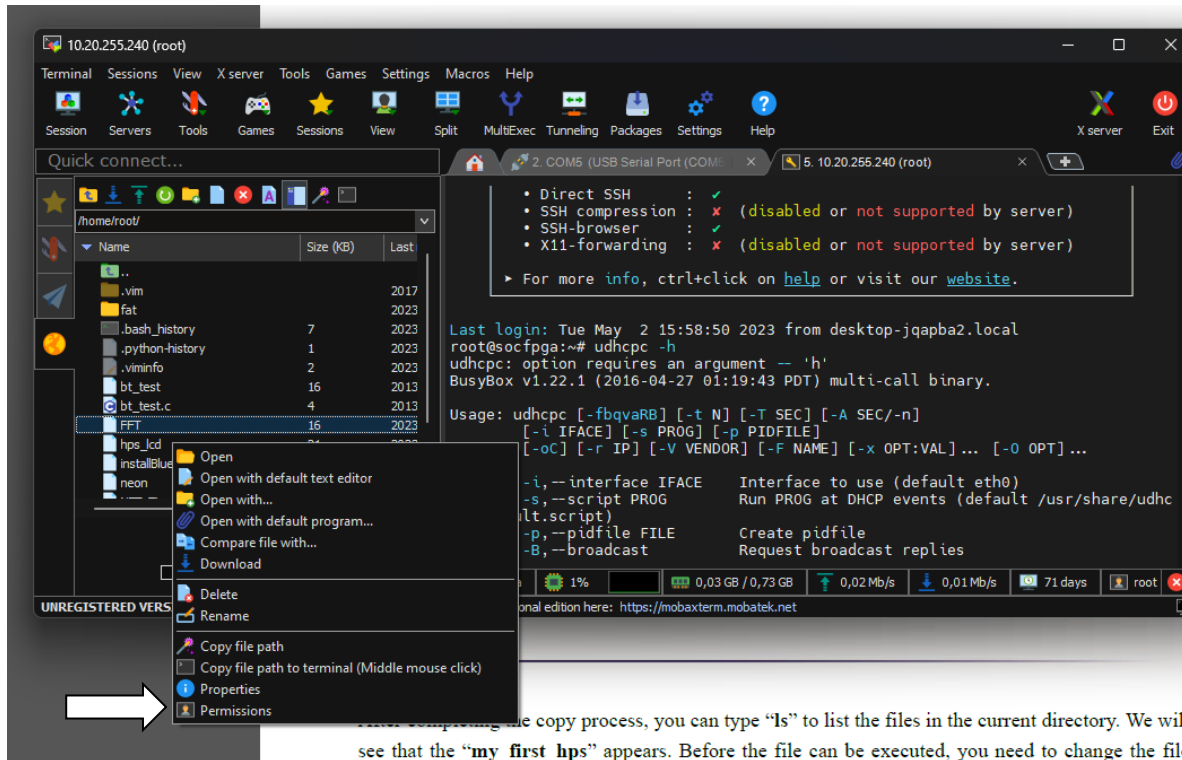


- In the «Quick connect» choose «Upload Current Folder»



# Copy the executable

- Upload the executable and provide «exec» permission (instead of executing «chmod 744 *executable\_name*»)



# Run the program

- That's all!

```
root@socfpga:~# ./FFT
Orig (dim=8):  0.12, 0.00   0.09, 0.00   0.00, 0.00  -0.09, 0.00  -0.12, 0.00  -0.09, 0.00  -0.00, 0.00   0.09, 0.00
FFT (dim=8):  -0.00, 0.00   0.50,-0.00   0.00, 0.00   0.00, 0.00  -0.00, 0.00   0.00,-0.00   0.00, 0.00   0.50, 0.00
iFFT (dim=8):  1.00, 0.00   0.71, 0.00   0.00,-0.00  -0.71, 0.00  -1.00, 0.00  -0.71, 0.00  -0.00, 0.00   0.71, 0.00
Orig (dim=8):  0.30,-0.00   0.21,-0.21   0.00,-0.30  -0.21,-0.21  -0.30,-0.00  -0.21, 0.21  -0.00, 0.30   0.21, 0.21
FFT (dim=8):  -0.00,-0.00   0.00,-0.00   0.00,-0.00   0.00, 0.00  -0.00,-0.00   0.00,-0.00   0.00,-0.00   2.40, 0.00
iFFT (dim=8):  2.40, 0.00   1.70,-1.70   0.00,-2.40  -1.70,-1.70  -2.40,-0.00  -1.70, 1.70  -0.00, 2.40   1.70, 1.70
root@socfpga:~# █
```

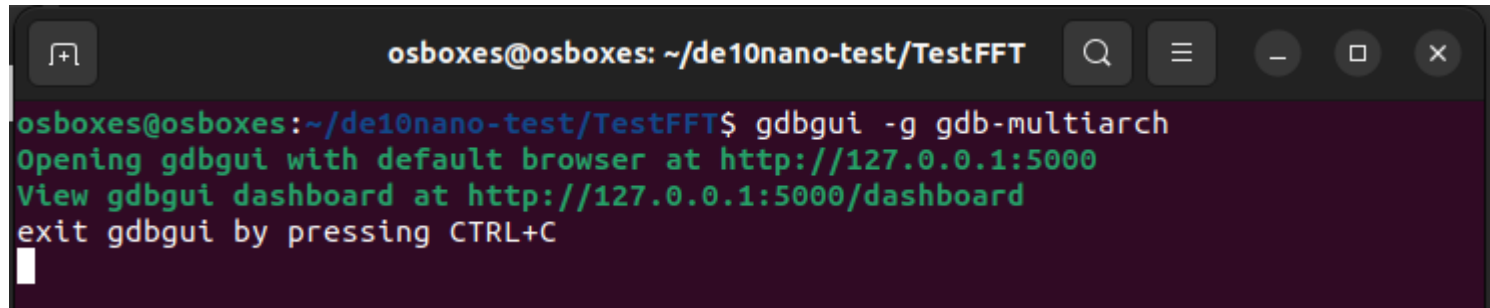
- What about debugging? Let's try remotely use the «gdbserver» and «gdbgui»

# Remote debugging

- On the remote system, launch the gdbserver:  
`gdbserver RemoteIPaddr.Port# file_name`

```
root@socfpga:~# gdbserver 10.20.255.240:8888 FFT
Process FFT created; pid = 10824
Listening on port 8888
```

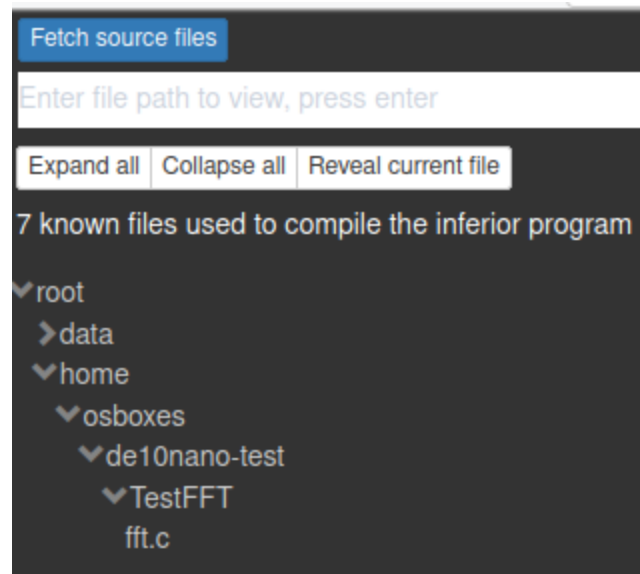
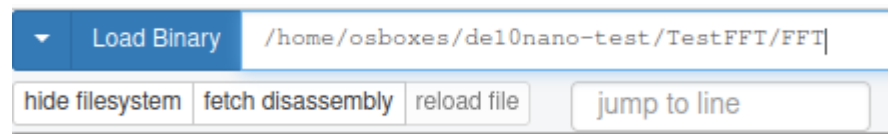
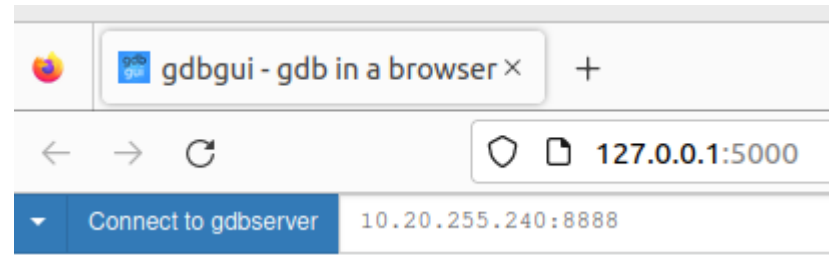
- On the host system, open a web browser and launch the web gui:  
`gdbgui -g gdb-multiarch`



```
osboxes@osboxes: ~/de10nano-test/TestFFT
osboxes@osboxes:~/de10nano-test/TestFFT$ gdbgui -g gdb-multiarch
Opening gdbgui with default browser at http://127.0.0.1:5000
View gdbgui dashboard at http://127.0.0.1:5000/dashboard
exit gdbgui by pressing CTRL+C
```

# Remote debugging

- Connect to the remote server:
- Load the binary file on the host:
- Fetch the source files:

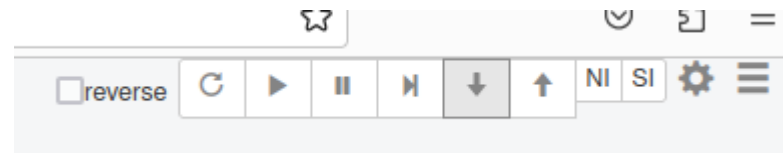


# Remote debugging

- Insert a breakpoint:

```
122     for(k=0; k<N; k++) {
123         v[k].Re = 0.125*cos(2*PI*k/(double)N);
124         v[k].Im = 0; //0.125*sin(2*PI*k/(double)N);
125         v1[k].Re = 0.3*cos(2*PI*k/(double)N);
126         v1[k].Im = -0.3*sin(2*PI*k/(double)N);
127     }
128
129     /* FFT, iFFT of v[]: */
130     print_vector("Orig", v, N);
131     fft( v, N, scratch );
132     print_vector(" FFT", v, N);
133     ifft( v, N, scratch );
134     print_vector("iFFT", v, N);
135 }
```

- Execute step by step:
- And check:



```
root@socfpga:~# ./FFT
Orig (dim=8):  0.12, 0.00  0.09, 0.00  0.00, 0.00 -0.09, 0.00 -0.12, 0.00 -0.09, 0.00 -0.00, 0.00  0.09, 0.00
FFT (dim=8):  -0.00, 0.00  0.50,-0.00  0.00, 0.00  0.00, 0.00 -0.00, 0.00 -0.00, 0.00  0.00,-0.00  0.00, 0.00  0.50, 0.00
iFFT (dim=8):  1.00, 0.00  0.71, 0.00  0.00,-0.00 -0.71, 0.00 -1.00, 0.00 -0.71, 0.00 -0.00, 0.00 -0.00, 0.00  0.71, 0.00
Orig (dim=8):  0.30,-0.00  0.21,-0.21  0.00,-0.30 -0.21,-0.21 -0.30,-0.00 -0.21, 0.21 -0.00, 0.30  0.21, 0.21
FFT (dim=8):  -0.00,-0.00  0.00,-0.00  0.00,-0.00  0.00, 0.00 -0.00,-0.00  0.00,-0.00  0.00,-0.00  2.40, 0.00
iFFT (dim=8):  2.40, 0.00  1.70,-1.70  0.00,-2.40 -1.70,-1.70 -2.40,-0.00 -1.70, 1.70 -0.00, 2.40  1.70, 1.70
root@socfpga:~#
```