µCore for the RZ-EASY Cyclone IV prototyping board

To experiment with the prototyping board, you need:

1. the **RZ-EASY Cyclone IV OMDAZZ** prototyping board @
   https://de.aliexpress.com/item/1005004903110811.html?spm=a2g0o.detail.1000060.2.73db4ab3pwoFez&gps-id=pcDetailBottomMoreThisSeller&scm=1007.13339.291025.0&scm_id=1007.13339.291025.0&scm-url=1007.13339.291025.0&pvid=34ac4770-05fa-4b0c-a387-f1fe196a98b9&_t=gps-id%3ApcDetailBottomMoreThisSeller%2Cscm-url%3A1007.13339.291025.0%2Cpvid%3A34ac4770-05fa-4b0c-a387-f1fe196a98b9%2Ctpp_buckets%3A668%232846%238115%232000&pdp_ext_f=%7B%22sku_id%22%3A%2212000030964042423%22%2C%22sceneId%22%3A%223339%22%7D&pdp_npi=2%40dis%21USD%2154.26%2154.26%21%21%21%21%21%402101d1bc16707072536501776e194c%2112000030964042423%21rec&gatewayAdapt=glo2deu
   It comes with the Intel USB-Blaster JTAG programming dongle.

2. a 5V Power supply. Either from a supply with a battery plug or a USB cabel.

3. the Quartus II (currently version 22.1std) development environment
   Windows:
   @ https://www.intel.com/content/www/us/en/software-kit/660905/intel-quartus-prime-standard-edition-design-software-version-20-1-1-for-windows.html?
   Linux/MacOS:
   @ https://www.intel.com/content/www/us/en/software-kit/660903/intel-quartus-prime-standard-edition-design-software-version-20-1-1-for-linux.html?

4. a USB <-> RS232 serial interface dongle. (Prolific is a lot faster then the ubiquituous FTI based dongles, because of lower transmit delays.)

5. a gforth_0.6.2 system as the basis for the uForth cross-compiler.
   Windows:
   Up until Win7, use the installer @ https://github.com/microCore-VHDL/gforth-for-Win7-hosts. Currently, there is no solution for Win10 and above.
   Linux/MacOS:
   You must install docker from https://www.docker.com/products/docker-desktop/. After installation, **docker pull microcore/gforth_062** will pull a gforth_0.6.2 system into your host. Please note that the docker image requires an RS232 serial interface.

6. the µCore and µForth co-design environment for the **RZ-EASY Cyclone IV prototyping board**, written in VHDL and Forth from github.
   If you have e.g. github CLI installed, **git clone** https://github.com/microCore-VHDL/uCore-for-Cyclone-IV will pull a customized version for the board into your host. If not, you can download all files as a .ZIP-file @ https://github.com/microCore-VHDL/uCore-for-Cyclone-IV. Klick on the green **Code** button (on the right) and select **Download ZIP**.
   Documentation on the board is available in the **documents/board** directory. During code development, ModelSim has been used for simulation and Synplify for synthesis.
   documents/uCore_overview.pdf is a six page introduction to µCore.

7. to configure the RS232 interface:
   Windows:
   Check the device-manager menue for the COM interface that is actually used by your RS232 dongle. Un-comment the line
   `include rs232_windows.fs B115200 Umbilical: COM3` in **umbilical.fs**. This will load gforth's windows rs232 interface using COM3 @ 115200 baud.
   Linux/MacOS:
   Check **/dev/ttyUSB*** for the interface number used by your RS232 dongle. Then open the script **uCore-for-Cyclone-IV/gforth062.sh** in a text editor and set the device mapping

`--device=/dev/ttyUSB0:/dev/ttyUSB0` by modifying the first /dev/ttyUSB0 accodingly. Un-comment the line `include rs232_linux.fs B115200 Umbilical: /dev/ttyUSB0` in **umbilical.fs**. This will load gforth's linux rs232 interface using ttyUSB0 @ 115200 baud. ttyUSB0 will always be valid, because you established the actual mapping in the **gforth062.sh** script.

FPGA Configuration

First connect the USB-Blaster to the JTAG and the RS232 dongle to the UART connector on the board. Switch power on. Start the Quartus programmer and open project **./Quartus/EP4CE6_Altera.qpf**.

Now the FPGA has to be configured. In the **Tasks** window, double click on **Program Device**. This will open the programmer window, which should be set for programming both the FPGA (as well as the serial flash device). Use FPGA configuration file **../vhdl/architectures/32b/uCore_32b.sof** and tick the **Program/Configure** check box. Pressing the **Start** button will configure the FPGA.

Make sure that the files **architecture_pkg.vhd**, **architecture_pkg_sim.vhd**, and **uDatacache.vhd** from **../vhdl/architecture/32b/** have been copied into the **../vhdl** directory. This will set µForth for the 32b (byte-addressed) architecture as well.

When you want to experiment with a different architecture, use the files of one of the prepared architectures accordingly.

Programming the serial flash device using an **<architecture>.jic** file will configure the FPGA automatically after power on.

Running coretest.fs

Windows:
Open a cmd window in the **software** directory. Executing gforth should now open gforth 0.6.2.

Linux/MacOS:
Open a terminal in the **uCore-for-Cyclone-IV** project directory and execute **./gforth062.sh**. This will start gforth_0.6.2 with a focus on the **software** directory.

Execute `include load_core.fs`, which will load the entire cross-compiler and the code for the core test. `microCross version 1310_32b` should be displayed and the last line should read `#3295 instructions compiled ok`.

Now execute `run`. If all goes well, `HANDSHAKE` will be displayed followed by the `uCore>` prompt on the next line.

Execute `coretest`. If all goes well, `0 ok` will be displayed. Any other number is an error code, which can be searched for in **coretest.fs**.

Executing `memtest` will do a simple test of the SDRAM. If it returns 0, everything is fine. Otherwise the address of the first error will be on the stack.

`bye` will bring you back into the gforth system on the host. `ucore` will return into the target environment.