coms20001_xtime_cli

Instructions for how to setup xTIMEcomposer's toolchain without using the built-in IDE

Tested on:

- Fedora 26 x64 on both X11 and Wayland

Should work just fine on any *nix systems.

1. Download xTIMEcomposer, extract to a suitable place, you should have something like this

```
tom@kurobako ~/xTIMEcomposer/Community_14.3.2 ls -lah
total 96K
drwxr-xr-x.  18 tom tom 4.0K Sep 30 17:23 .
drwxr-xr-x.   3 tom tom 4.0K Sep 30 17:23 ..
drwxrwxrwx.   6 tom tom 4.0K Sep 30 17:23 arm_toolchain
drwxr-xr-x.   2 tom tom 4.0K Sep 30 17:23 bin
drwxr-xr-x.   3 tom tom 4.0K Sep 30 17:23 build
```

```
drwxr-xr-x.   3 tom tom 4.0K Sep 30 17:23 configs
drwxr-xr-x.   3 tom tom 4.0K Sep 30 17:23 doc
drwxr-xr-x.   4 tom tom 4.0K Sep 30 17:23 examples
drwxr-xr-x.   2 tom tom 4.0K Sep 30 17:23 icons
drwxr-xr-x.   2 tom tom 4.0K Sep 30 17:23 include
drwxr-xr-x.   4 tom tom 4.0K Sep 30 17:23 lib
drwxr-xr-x.   2 tom tom 4.0K Sep 30 17:23 libexec
drwxr-xr-x.   2 tom tom 4.0K Sep 30 17:23 license
drwxr-xr-x.   2 tom tom 4.0K Sep 30 17:23 scripts
-rwxr-xr-x.   1 tom tom 2.2K Sep 30 17:23 SetEnv
drwxr-xr-x.   4 tom tom 4.0K Sep 30 17:23 src
drwxr-xr-x.   4 tom tom 4.0K Sep 30 17:23 target
drwxr-xr-x. 301 tom tom  20K Sep 30 17:23 targets
-rwxr-xr-x.   1 tom tom  147 Sep 30 17:23 xtimecomposer
drwxr-xr-x.   8 tom tom 4.0K Nov  5 17:45 xtimecomposer_bin
```

2. **In the same directory where `SetEnv` is located**, execute it:

```
tom@kurobako  ~/xTIMEcomposer/Community_14.3.2  source ./SetEnv
```

3. Verify that your path has xTIME related variables:

   Before:

```
tom@kurobako  ~/xTIMEcomposer/Community_14.3.2  echo $PATH
/usr/local/bin:
/usr/local/sbin:
/usr/bin:
/usr/sbin:
/home/tom/bin:
/home/tom/.local/bin
```

   After:

```
 tom@kurobako  ~/xTIMEcomposer/Community_14.3.2  echo $PATH
/home/tom/xTIMEcomposer/Community_14.3.2/bin:
/home/tom/xTIMEcomposer/Community_14.3.2/xtimecomposer_bin:
/home/tom/xTIMEcomposer/Community_14.3.2/arm_toolchain/bin:
/usr/local/bin:
/usr/local/sbin:
/usr/bin:
/usr/sbin:
/home/tom/bin:
/home/tom/.local/bin
```

4. You can now build your project:

```
tom@kurobako  ~/xTIMEcomposer/Community_14.3.2  cd ~/coms20001_cw
tom@kurobako  ~/coms20001_cw1/game_of_life    master ●  xmake
Checking build modules
Using build modules: lib_i2c(3.0.0) lib_logging(2.0.0) lib_xassert(2
Creating game_of_life.xe
Build Complete
```

5. Now, to make life easier, you can create a script like so:

```
#!/bin/sh
pushd ~/xTIMEcomposer/Community_14.3.2/; source ./SetEnv;  popd
```

Place the script in your project root and run `source ./the_script.sh` before using `xmake`; add this to your `.bashrc` or `.zshrc` if you feel like XC is the next big thing /s.

Running your project

After xmake successfully builds your project, you want to run it with `xsim` (`xsim`, along with `xmake` should both be in your PATH if you have your environment setup correctly). The compiled binary will be located in `<project_root>/bin/<project_name>.xe`.

To run the binary:

```
tom@kurobako  ~/coms20001_cw1/game_of_life    master xsim bin/game_
ProcessImage: Start, size = 16x16
DataInStream: Start...
DataOutStream: Start...
Waiting for Board Tilt...
# .....
```

You may want to add some flags:

```
--warn-resources --warn-links --warn-stack --warn-registers --stats
```

The `--stats` flags is especially useful at gaging link(channel) usages.

FAQ

`xrun -l` says the following:

```
# xrun -l
ERROR: Device permissions for product 0x20b1 0xf7d4 are not set
correctly
       Please add the correct usb device rules to allow user space
access

Available XMOS Devices
----------------------

  No Available Devices Found
```

1. Verify XMOS board is connected and visible:

   ```
   # lsusb
   Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
   Bus 001 Device 017: ID 20b1:f7d4 XMOS Ltd
   Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
   ...
   ```

2. Check permission for the device:

   ```
   # ls /dev/bus/usb/001 -lah # replace 001 with the bus id from
   lsusb
   total 0
   drwxr-xr-x. 2 root root     200 Nov 14 04:13 .
   drwxr-xr-x. 6 root root     120 Nov 14 02:56 ..
   crw-rw-r--. 1 root root 189, 18 Nov 14 04:13 017
   ...
   ```

   In this case, `017` (the device id from lsusb) is our XMOS board of which is owned by root.

3. Add the current user as owner of the XMOS board:

   ```
   # cat  /etc/udev/rules.d/92-xmos-x200.rules
   SUBSYSTEM=="usb", ATTR{idVendor}=="20b1",
   ATTR{idProduct}=="f7d4", ACTION=="add", OWNER="<name>",
   MODE="0664"
   ```

   The file `92-xmos-x200.rules` will not exist so you will have to create it manually. The name of the file is not critical, the number describes the order of which the rule will be applied, the rest are for descriptive purposes. Replace `<name>` with your user name.

4. Reload udev rules:

```
# sudo udevadm control --reload-rules && udevadm trigger
```

Disconnect and reconnect the device, verify that permission of the again:

```
# ls /dev/bus/usb/001 -lah # replace 001 with the bus id from
lsusb
total 0
drwxr-xr-x. 2 root   root     200 Nov 14 04:13 .
drwxr-xr-x. 6 root   root     120 Nov 14 02:56 ..
crw-rw-r--. 1 <name> root 189, 18 Nov 14 04:14 019
...
```

The owner of the device should be `<user>`, running `xrun -l` should now see the board. Notice that the device id may change, if unclear, always check with output of `lsusb` first.

Tips

You may want to add the `-report` flag to your `Makefile` so that `xcc` can tell you more. In your `Makefile`, find the definition of `XCC_FLAGS` and append the flag, for example: `XCC_FLAGS = -report`

The output(near the end) should now look something like this:

```
Creating game_of_life.xe
Constraint check for tile[0]:
  Cores available:            8,   used:          4 .  OKAY
  Timers available:          10,   used:          4 .  OKAY
  Chanends available:        32,   used:          6 .  OKAY
  Memory available:       262144,   used:      56436 .  OKAY
    (Stack: 5996, Code: 47452, Data: 2988)
Constraints checks PASSED.
Build Complete
```

This information is very useful for debugging.