Institut
Mines-Telecom

Paris Sorbonne
University

Using TTool with SoCLib

Daniela Genius, Ludovic Apvrille
daniela.genius@lip6.fr
ludovic.apvrille@telecom-paristech.fr

Installation Guide for the
SoCLib extension
of **TTool/AVATAR**

Requirements for Linux and MacOSX

The following describes a developer installation (to obtain a release of TTool, see `http://ttool.telecom-paristech.fr/`)

1. `git clone git@gitlab.enst.fr:mbe-tools/TTool.git`
2. The `MPSoC` subdirectory of TTool contains MutekH sources in the `mutekh` subdirectory and a SoCLib version in the `soclib` subdirectory
3. A 64 bit Linux is preferable (successfully tested)
4. Your system has to feature a gcc compiler and a gdb
   - Linux : gcc 4.7.3 was tested with success
   - MacOSX : there are some difficulties with the Xcode compiler, use macports or the homebrew gcc-4.8 compiler
5. A valid SystemC compiler to compile the SoCLib platform: `accellera.org/downloads/standards/systemc`
   - Linux : systemc-2.2.0 was tested with success
   - MacOSX : systemc-2.3.1 from Accellera was tested with success: `accellera.org/downloads/standards/systemc`

Installation under Linux

Choose a directory where you wish to install TTool : `$TTOOL_DIR`
If you have root privilege on your machine, it is recommended to install the auxiliary tools and compilers under `/opt`.
Often, this is not the case; choose a directory, example `$HOME/mydir` The crosscompiler generator script has to be adapted (see slide referring to cross compiler installation).
In the following, we refer to both as `$INSTALL_DIR`
Add in your `.bashrc`

```
export SYSTEMC=$INSTALL_DIR/systemc-2.2.0

export PATH=$INSTALL_DIR/java/bin:$SYSTEMC/bin:$INSTALL_DI
$TTOOL_DIR/MPSoC/soclib/utils/bin:$PATH
export LD_LIBRARY_PATH=$INSTALL_DIRg/mutekh/lib/:$SYSTEMC/
```

Installation under Linux (2)

We recommend to use bash.
Example .bashrc lines

```
export PATH=$PATH:$HOME/bin:$HOME/git/TTool/MPSoC/soclib/ut
export PATH=$PATH:/opt/gcc-cross-mipsel/4.3.3/bin/
export PATH=$PATH:/cxtools/gcc_mips/obj/bin/
export PS1='\u@\h \w $ '
export PATH=$HOME/mutekh/bin:$PATH
```

Installation under MacOSX

The differences are the following :

```
export SYSTEMC=$INSTALL_DIR/systemc-2.3.1
```

for the SystemC version

```
export LD_LIBRARY_PATH=$INSTALL_DIR/mutekh/lib/:$SYSTEMC/li
```

to use `lib-linux64`.

Add-ons

Some tools cannot be distributed by us. If you wish to

- Make proofs of safety properties from AVATAR models, you need to install and configure UPPAAL. Here is some help : `http://ttool.telecom-paristech.fr/installation_companion.html#uppaal`
- Make proofs of security properties from AVATAR models, you need to install and configure ProVerif. Here is some help : `http://ttool.telecom-paristech.fr/installation_companion.html#proverif`

Generating the Crosscompilers

`MPSoC/mutekh/tools/crossgen.mk` fetches and generates the crosscompilers.
The crosscompiler will serve to compile your application (task and main file produced by TTool) for the desired architecture.
To generate your crosscompilers, you have to execute `./crossgen.mk` which is part of MutekH.
This is done by typing
`./crossgen.mk toolchain`
Important : per default, the installation directory is /opt. You migght not have root privileges. In this case change the line

```
PREFIX=/opt/mutekh
```

into

```
PREFIX=$INSTALL_DIR/mutekh
```

Where `$INSTALL_DIR` is the directory you have chosen to install your crosscompilers

Generating the Crosscompilers (2)

Available targets for crosscompiler generation:
`mipsel, powerpc, arm, i686, x86_64, nios2, sparc, avr, lm32, microblaze, avr32`
SoCLib extension tested for `mipsel, powerpc`
The script will fetch everything it requires (on some machines a proxy must be set beforehand)
See also `www.soclib.fr/trac/dev/wiki/CrossCompiler`
Note: If an installation aborts, be careful to delete all files in /tmp/crossgen.

Available Processor Cores

- Currently the complete toolchain is only validated for PowerPC, but it is easy to add other CPU as MutekH allows heterogeneous processors. Validation for MIPS and Microblaze is under way.
- The topcell can potentially be used with Instruction Set Simulators for the following architectures :
  - PowerPc 405
  - Nios II
  - Mips 32
  - Arm 7
  - Sparc v8
  - LM 32
  - Microblaze
- The mapping table generation and calculation of addresses in `/src/ddtranslatorSoclib/toTopcell` of other architectures might have to be extended (ongoing work, requests to `daniela.genius@lip6.fr`)

Directories in the TTool Arborescence

Generation of POSIX code for local workstation in directory `TTool/executablecode`.
Generation of code for MPSoC platform in directory `TTool/MPSoC`.
`TTool/executablecode` and `TTool/MPSoC` both contain the following subdirectories:

- `generated_src`: generated task code for AVATAR blocks and main code spawning the POSIX threads
- `src`: the runtime for MPSoC platforms

`TTool/MPSoC` additionnally contains

- `generated_topcell`: topcell and mapping information to generate the ldscript (instructions for linker).

Directories in the TTool Arborescence (2)

In the `generated_topcell` directory :

- `config_noproc`
- `deployinfo.h` copied to `Prog/mutekh/arch/soclib`, generated by `TTool/src/ddtranslatorSoclib/toTopcell/Deplyinfo.java`
- `deployinfo_map.h` copied to `Prog/mutekh/arch/soclib`, generated by `TTool/src/ddtranslatorSoclib/toTopcell/Deplyinfo.java`
- `nbproc` contains number of CPUs, generated by `TTool/src/ddtranslatorSoclib/toTopcell/Deplyinfo.java`. This info is appended to `config_noproc`
- `procinfo.mk`

Directories in the TTool Arborescence (3)

Central Makefile `src/MPSoC/Makefile.forsoclib` which copies generated code from the `TTool` into the `MPSoC` arborescence as follows:

- `generated_src` directory:
  - `*.c` and `*.h` copied to `MPSoC/mutekh/examples/avatar`
  - `src_soclib` to `MPSoC/mutekh/libavatar`
- `generated_topcell` directory:
  - `top.cc` to `MPSoC/soclib/soclib/platform/topcells/caba-vgmn-mutekh_kernel_tutorial/top.cc`
  - `deployinfo.h` and `deployinfo_map.h` to `MPSoC/mutekh/arch/soclib`
    they are used by
    `MPSoC/mutekh/arch/soclib/ldscript.cpp`, a preprocessor generating the `ldscript`

Directories in the TTool Arborescence (4)

The `src/ddtranslatorSoclib` directory

- contains the code for analyzing the deployment diagrams
- contains subdirectories toSoclib and toTopCell generating task/main code and topcell/information for the ldscript and main, respectively

Make sure that the script `TTool/ttool.exe` contains the line

```
java -Xmx1024m -Djavax.net.ssl.trustStore=ServerKeyStore
-Djavax.net.ssl.trustStorePassword=123456  -jar ttool.jar
-config config.xml -experimental -debug -avatar -uppaal
-launcher
```

In TTool type `make all`. This should generate the class files and `TTool/bin/ttool.jar`. Then, `./ttool.exe` starts TTool.