MutekH quick start guide for SoCLib platform

SoCLib simulator allow easy experimentation with advanced multi-processor programming.

This guide explains how to run MutekH on a SoCLib hardware simulator with native processor heterogeneity support.

The SoCLib simulator used here is easy to use but has a complex internal design due to dynamic processors model instanciation. This is really convenient if you want to experiment with different processors without modifying the simulator. This simulator allows processor heterogeneity.

If you are interested in learning SoCLib hardware simulator, or plan to use SoCLib to model your own platform, you have better reading the MutekH/SocCLib tutorial first.

You are **highly encouraged** to first follow the MutekH as Unix process quick start guide which introduce more basic concepts.

The SoCLib platform

Getting SoCLib

We now need to have a working SoCLib install. SoCLib installation is explained here: ?soclib:InstallationNotes

Moreover, you'll need the MutekH source tree and its prerequisites. See InstallationNotes

SoCLib platform description

The SoCLib source tree contains a platform dedicated to this tutorial:
`soclib/soclib/platform/topcells/caba-vgmn-mutekh_kernel_tutorial/`.

The MutekH part

Getting the sources

```
svn co https://www-asim.lip6.fr/svn/mutekh/trunk/mutekh
```

Writing the example source code

The MutekH kernel source code is fully configurable and can be tweaked to adapt hardware platform and application needs. Configuration is handled by a dedicated tool which check dependencies and other relationships between the large set of available configuration tokens.

What you need to do:

- Write the source code in `hello.c`
- Write the `Makefile`
- Write the `platform-mips+arm.dts` to describe hardware, see FlattenedDeviceTree for details.
- Write the source configuration file, see BuildSystem for details.

Note: This example is available directly from `examples/hello_het` directory in source tree: trunk/mutekh/examples/hello_het

Getting the cross-compilers

You can rely on the `tools/crossgen.mk` script which comes along with MutekH to build some GNU cross-toolchains:

```
$ tools/crossgen.mk
$ tools/crossgen.mk all TARGET=mipsel-unknown-elf
$ tools/crossgen.mk all TARGET=arm-unknown-elf
```

Compiling the application along with MutekH

```
$ cd path/to/mutekh
$ make kernel-het CONF=examples/hello_het/config BUILD=ph-het EACH=soclib-arm:soclib-mips32el
```

This will build the MutekH kernel along with the application. The simulator can then be built using:

```
$ cd path/to/soclib/soclib/platform/topcells/caba-vgmn-mutekh_kernel_tutorial
$ make system.x
```

Execution

The simulator needs the MutekH executable file name and the processor type and the number of processors of this type:

```
$ cd path/to/soclib/soclib/platform/topcells/caba-vgmn-mutekh_kernel_tutorial
$ ./system.x mips32el:2 path/to/mutekh/kernel-mipsel.het.out arm:2 path/to/mutekh/kernel-arm.het
```

You may want to refer to other articles and documents available from the main page to go further with MutekH.

The BuildingExamples article explain how to build other sample applications.

Other more advanced topics and guides are available from the Main page.