ALMOS distribution for the TSAR many-core

Summary

Running ALMOS on the TSAR virtual prototype requires the installation of several open-source technologies like a GCC cross-compiler for Mips (el), ?SocLib virtual prototyping library, ?TSAR related components, and ?SystemCASS. To simplify the task of building and configuring a correct development environment, ALMOS comes with a stand-alone and ready-to-use distribution.

Mainly, this distribution enables you to:

- Port your own applications and libraries to ALMOS.
- Run these applications on TSAR using several configurations ranging from 4 to 1024 cores.
- Analyse the performance of your applications or the totality of the software-stack on a large-scale many-core.

Upon your needs you can also use this distribution to:

- Validate and evaluate any kernel new features or updates.
- Experiment and develop new parallel programming libraries and run-times for a large-scale single-chip many-core.
- Consolidate/build your own educational materials in the field of operating systems and parallel programming.
- Validate and evaluate any hardware evolution/development in a TSAR based architecture.

Compatibility

The distribution package contains some binary executables (e.g. GCC cross-compiler, TSAR simulator). These programs can run on a Linux based distribution. They were tested on Ubuntu 10.04/12.04 and on Scientific Linux 6.2. Both 64 and 32 bits host machines are supported.

Setup

Download the latest stable distribution from this link. Decompress the .tbz2 file:

```
$ tar jxf almos-tsar-mipsel.tbz2
```

Now you have a sub-directory named *almos-tsar-mipsel*. In the reset of this page, we refer to the absolute path to this directory as *DISTRIB*.

Start the simulation

Before any use of the distribution package you need to set some environment variables:

```
$ cd DISTRIB
$ source ./SourceMe
```

Note: by default you have to source this file from its local directory, that is, you have to be in the *DISTRIB* directory.

Now, lets go to *DISTRIB/test/pf1* and run *make sim1*:

```
$ cd DISTRIB/test/pf1
$ make sim1
```

This will launch the TSAR full-system simulator with its 4 tty terminals and one frame-buffer window. You will end by having a shell prompt on the tty1. That is it ... you are done !!

The tty0 and tty3 are reserved for the kernel trace and information messages. For a user application, the tty1 represent the stdin and stdout while the tty2 represent the stderr.

Sim1 rule means the simulator of TSAR is configured to one cluster, that is, 4 cores. Using *sim4*, *sim16*, *sim64* or *sim128* lets you start the simulator with (respectively) 4, 16, 64 or 128 clusters of 4 cores each.

Note: although the hardware configuration can be changed at each simulation, there is no need to recompile or regenerate the kernel. The kernel of ALMOS detects the hardware resources at each boot. A user application can get the number of online cores using the standard *sysconf* call (man sysconf).

Now lets take a look inside the *DISTRIB/test/pf1* directory.

```
$ ls DISTRIB/test/pf1
arch-info.bin bootloader.bin hdd-img.bin kernel-soclib.bin Makefile
```

After each *make simN* command (where N is 1, 4, 16, 64 or 128) you will have at least 4 .bin files. These bins are required by the simulator. In the TSAR simulated platform, there is a ROM component and a H.D.D controller. The ROM will be filled with the contents of *arch-info.bin*, *bootloader.bin* and *kernel-soclib.bin* before starting the simulation. The *hdd-img.bin* is a FAT32 file system image sought by the H.D.D controller at each sector transfer.

The *kernel-soclib.bin* file is the ALMOS kernel while the *bootloader.bin* file is the ALMOS boot-loader for TSAR. The *arch-info.bin* file contains the description of the TSAR hardware resources. This file is regenerated (copied from *DISTRIB/test/arch-bins*) at each *make simN* command.

You can take a look to *DISTRIB/test/pf1/Makefile*. This file lets you customize some parameters like the used ALMOS kernel revision and the number of simulator threads. If you want to run simultaneously several configurations, lets say 4, you can create 3 additional *DISTRIB/test/pf[2-4]* directories each of which with its own customized Makefile. Than in each of these *pf* directories you can type your *make simN* command.

Note: by default, the location from where the corresponding kernel revision is looked for is *DISTRIB/test/kern-bins*. The number of simulator's threads should not exceed the number of the physical CPUs (not the logical one) of your host machine. Finally, the hdd-imge.bin is a symbolic link to *DISTRIB/test/misc/hdd-img.bin* so be careful to provide an *hdd-img.bin* regular file per *pf* directory when you run simultaneously a multiple instances of the simulator.

Your first application

Running without interactive mode