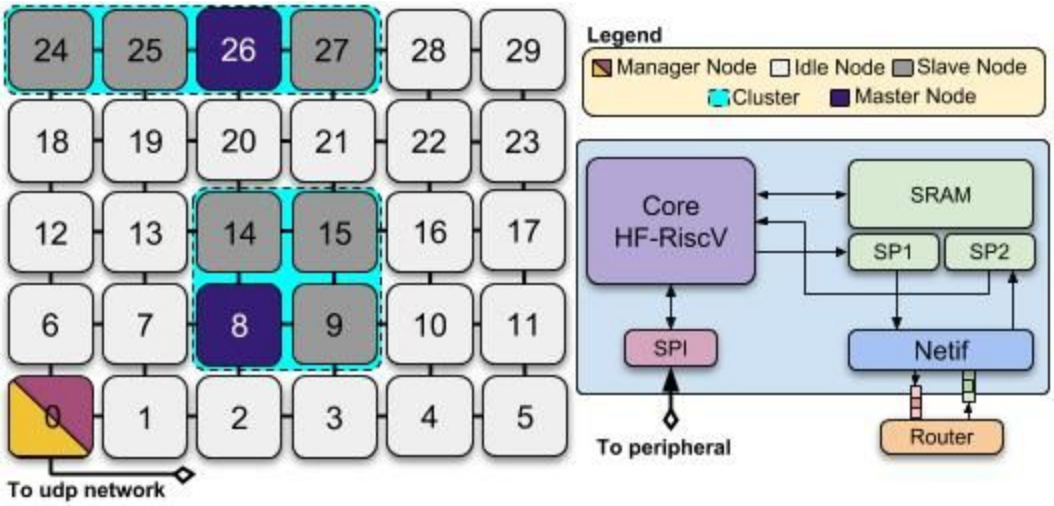
URSA - The Lazy Manual

Anderson Domingues, 11-Mar-19

1 Before Start

URSA is an API for building functional, cycle-accurate hardware emulators. Emulator are also called *functional simulators* since they mimic the behaviour of components they emule. Although URSA provides classes for building platforms, URSA is not a platform by itself. Platforms must be developed using provided libraries (similarly to OVP) and hardware models. Hardware models are written in C++ and inherit from base classes provided within URSA. For now, only models for the ORCA platforms are provided.

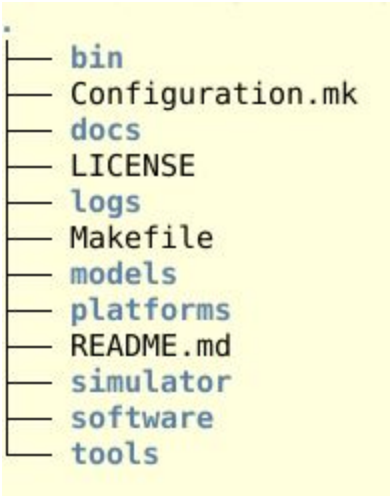
The ORCA platform is a configurable MxN MPSoC platform with focus on self-adaptation. It uses HF-RiscV processor in their processing elements, which are interconnected by a network-on-chip (NoC). The NoC is based on Hermes NoC. The image below shows an overview of the hardware for the ORCA platform. Since the MPSoC is configurable, the image reflects only a possible instance of the platform, which may vary the number and type of tiles, among other configurations.



Basically, all PE runs a copy of the HellfireOS kernel with a few modifications to the NoC driver. The kernel is responsible for managing applications execution, as well as provide the required treatment for networking packets, allowing the system to send and receive packets among the participating nodes. There are two type of nodes in the system. First, processing nodes (also called processing tiles) contain processor cores, and may have an application loaded to their memory at startup. The other type corresponds to network tiles (also called network nodes). Such nodes provide access to the MPSoC form an external source. For now this source is limited to UDP networks via sockets technology.

2 Downloading the Project

The repository (https://github.com/andersondomingues/ursa) hosts source code for both URSA and ORCA. You may download all the content from this repository. Please note that the software folder has a reference (https://github.com/andersondomingues/hellfireos) to the HellfireOS repository. You must download the HellfireOS as well. After the download of both repositories, your work directory should look like the one below. The folder *hellfireos* must reside into software folder.



Folder explanation:

- **bin** : contains all generated files, including generated libraries and executable files.
- **Configuration.mk** : includes the design time configuration of the platform.
- **docs** : includes this file and pictures from the MD file
- **logs**: logs for the emulated platform
- **Makefile**: compilation script for the whole platform
- **models**: hardware models
- **platforms**: top-level modules for each platform
- **simulator**: URSA's source
- **software**: kernel, middleware and libraries
- **tools**: several helper scripts and files

3 Running simulations

3.1 Requirements

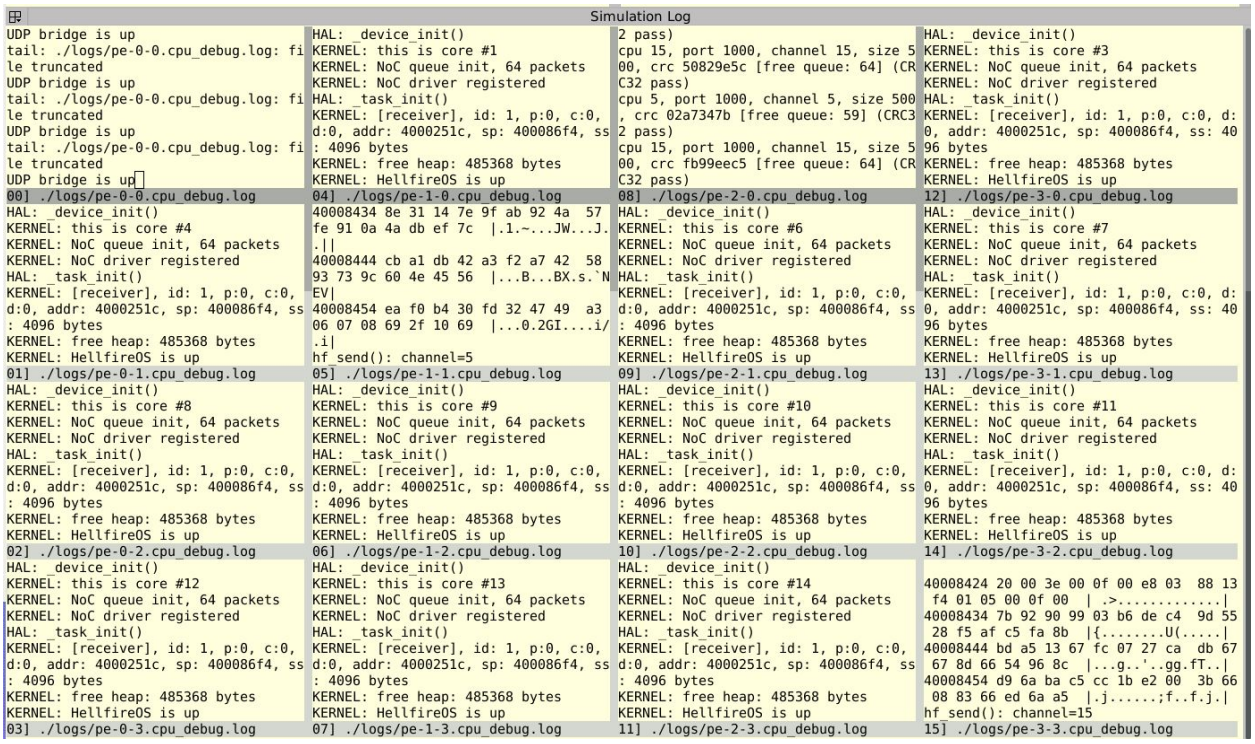
- Compile URSA and platforms using GCC version 6.3.0 (Debian 6.3.0-18+deb9u1). No guarantees for other versions. `$make`
- Compile RiscV-compatible applications with riscv32 toolchain.

- ❏ Gaph users: `$module load riscv32-elf`
- ❏ Other users: use script from `$./tools/riscv32toolchain.sh` and add the generated toolchain to the path.

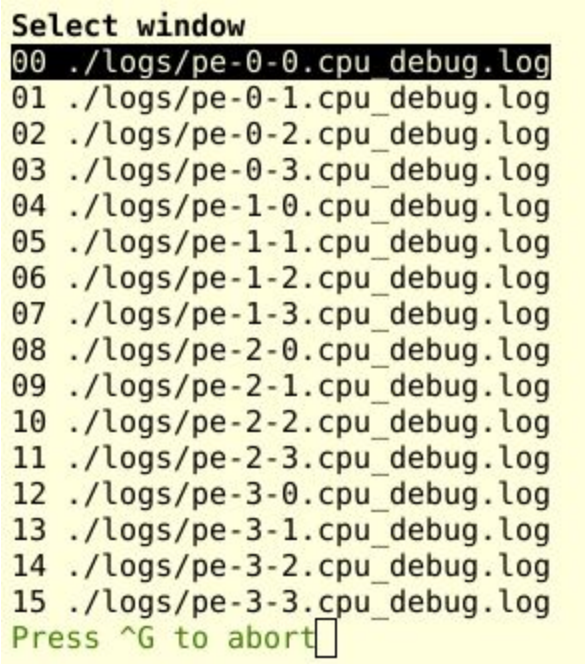
3.2 How to Run

1) `$cd ./platforms/sulphane-generic`
2) `$../tools/multitail.sh` #requires multitail and terminator
3) `$make app` #to compile the kernel + application
4) `$make clean; make` #to compile and run the platforms

Your screen should look like the below picture.



Press B to bring up the window selection menu.



Select one of the shown files to open the log file for one of the PE. The key "Q" closes the log file.