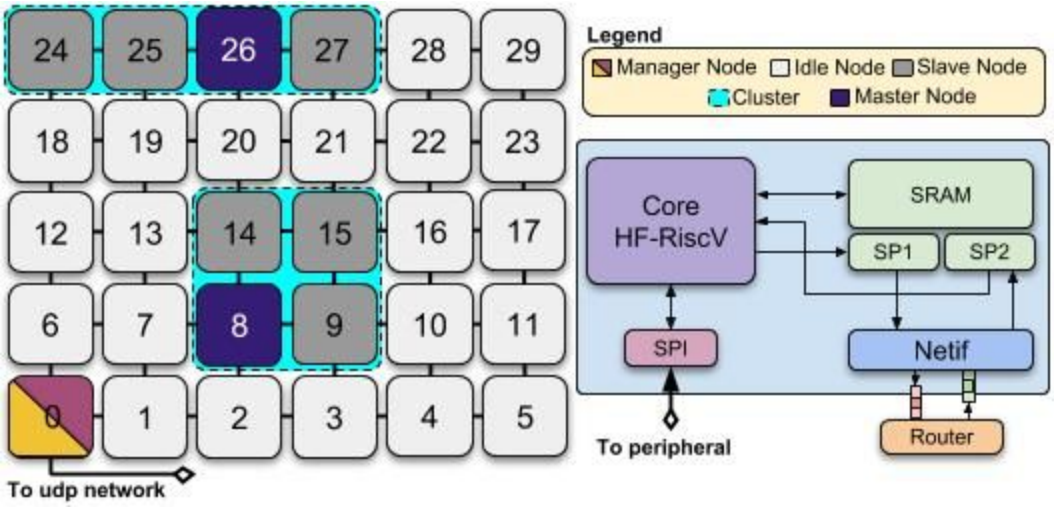
URSA/Sulphane - The Lazy Manual

Anderson Domingues, 24-Oct-18

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 simulating and build 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 Sulphane platforms are provided.

Sulphane (which chemical symbol is H_2S) stands for Hermes-Hellfire SoC. As the name says, the platform is a SoC comprising an instance of Hermes NoC in which each of the PE is equipped with one Core HF-RiscV. The following picture illustrate the platform.



Basically, all the PE runs a copy of the HellfireOS kernel with a few modifications to the NoC driver. The network interface (NI) is quite different from the one used in Hemps. First, the NI cannot access the main memory directly. The sending/receiving of packages is done through two auxiliary memory modules. Once a packet arrives at the NI input port, data is copied to the memory SP2 and the core is interrupted at the end of the copy. Then, the core copies the received data from the memory SP2 into the main memory. The sending process is very similar but for the core copying data into memory SP1 and then activating the NI. The arrows in the above picture represent the data flow between components.

Another important fact about the platform is the SPI interface hooked to the core. For the PE-0, specifically, an UDP module (not shown in the picture) is **to be** attached the SPI interface. Other peripheral can be attached to other cores if they comply with SPI.

2 Downloading the Project

The repository (https://github.com/andersondomingues/ursa) hosts both URSA and Sulphane source code. Please notice that the software folder is empty and you must clone the repository for HellFireOS (https://github.com/andersondomingues/hellfireos) into it. Your work directory must be similar to the one bellow.

There are four folder at the root of the project: *libs*, *platforms*, *software* and *tools*.

- *libs*: contains source code for the simulation core (libsim) and hardware models of Sulphane platform (libmod). For other hardware models a new library must be created inside into this folder.
- *platforms*: holds projects for some test benches (tb_ prefix) and a generic Sulphane platform. New platforms must be deployed to this folder.
- *software*: stores code and libraries for software-only components. The repository of HellfireOS must be clone inside if working with Sulphane.
- *tools*: goodies that help in the development and debugging of platforms

3 Running Sulphane

3.1 Requirements

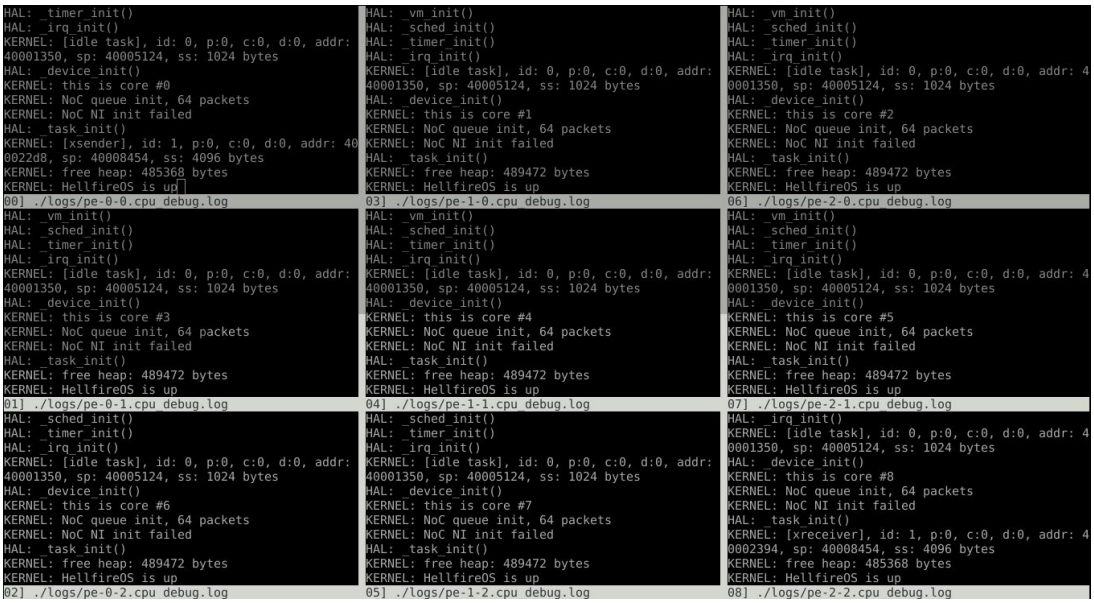
- Compile URSA and platforms using GCC version 6.3.0 (Debian 6.3.0-18+deb9u1). No guarantees for other versions.
- Compile RiscV-compatible applications with riscv32 toolchain.
    - Gaph users: `$module load riscv32-elf`

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

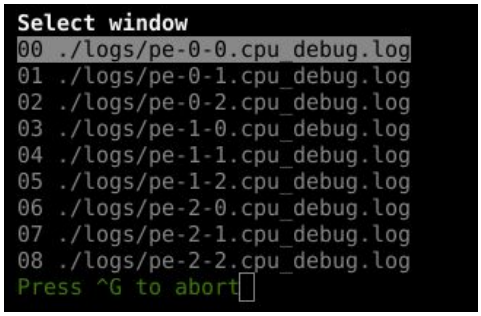
3.2 How to Run

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.