

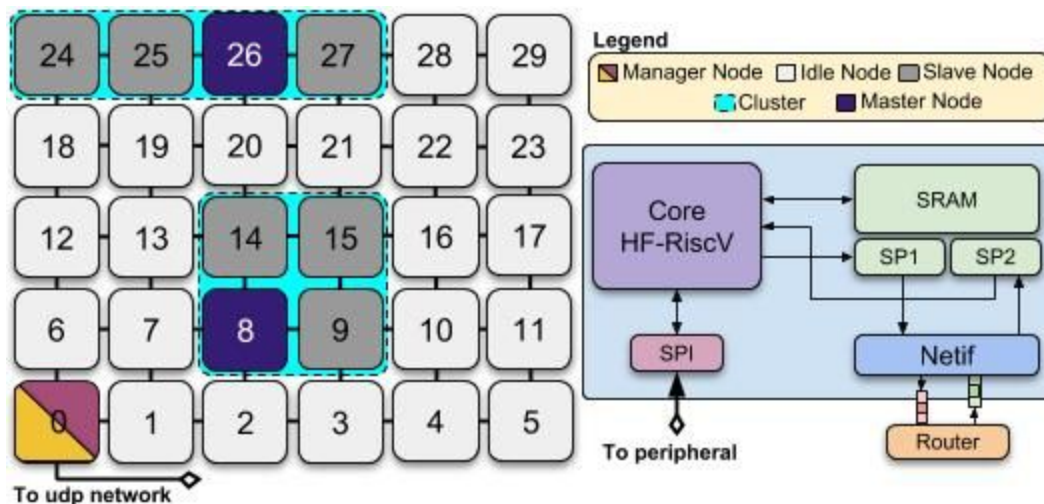
URSA - The Lazy Manual

Anderson Domingues, 11-Mar-19

1 Before Start

URSA is an API for building functional, cycle-accurate hardware emulators. Emulators are also called *functional simulators* since they mimic the behaviour of components they emulate. 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.

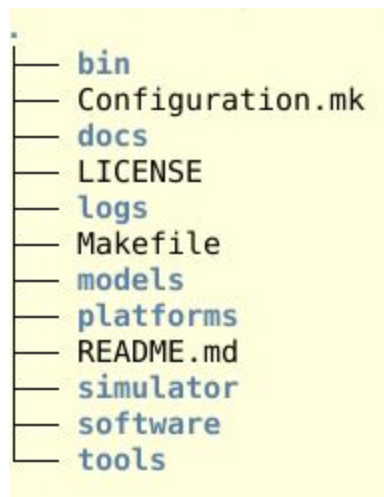
The ORCA platform is a configurable MxN MPSoC platform with focus on self-adaptation. It uses HF-RiscV processor in their processing elements, which are interconnected by a network-on-chip (NoC). The NoC is based on Hermes NoC. The image below shows an overview of the hardware for the ORCA platform. 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.

Simulation Log			
UDP bridge is up tail: ./logs/pe-0-0.cpu_debug.log: file truncated UDP bridge is up tail: ./logs/pe-0-0.cpu_debug.log: file truncated UDP bridge is up tail: ./logs/pe-0-0.cpu_debug.log: file truncated UDP bridge is up tail: ./logs/pe-0-0.cpu_debug.log: file truncated UDP bridge is up tail: ./logs/pe-0-0.cpu_debug.log: file truncated 00] ./logs/pe-0-0.cpu_debug.log	HAL: device init() KERNEL: this is core #1 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 04] ./logs/pe-1-0.cpu_debug.log	2 pass) cpu 15, port 1000, channel 15, size 500, crc 50829e5c [free queue: 64] (CRC32 pass) cpu 5, port 1000, channel 5, size 500, crc 02a7347b [free queue: 59] (CRC32 pass) cpu 15, port 1000, channel 15, size 500, crc fb99eec5 [free queue: 64] (CRC32 pass) 08] ./logs/pe-2-0.cpu_debug.log	HAL: device init() KERNEL: this is core #3 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 12] ./logs/pe-3-0.cpu_debug.log
HAL: device init() KERNEL: this is core #4 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 01] ./logs/pe-0-1.cpu_debug.log	40008434 8e 31 14 7e 9f ab 92 4a 57 fe 91 0a 4a db ef 7c .1....JW...J. 40008444 cb a1 db 42 a3 f2 a7 42 58 93 73 9c 60 4e 45 56 ...B...BX.s.'N EV 40008454 ea f0 b4 30 fd 32 47 49 a3 06 07 08 69 2f 10 69 ...0.2GI....i. .i hf send(): channel=5 05] ./logs/pe-1-1.cpu_debug.log	HAL: device init() KERNEL: this is core #6 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 09] ./logs/pe-2-1.cpu_debug.log	HAL: device init() KERNEL: this is core #7 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 13] ./logs/pe-3-1.cpu_debug.log
HAL: device init() KERNEL: this is core #8 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 02] ./logs/pe-0-2.cpu_debug.log	HAL: device init() KERNEL: this is core #9 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 06] ./logs/pe-1-2.cpu_debug.log	HAL: device init() KERNEL: this is core #10 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 10] ./logs/pe-2-2.cpu_debug.log	HAL: device init() KERNEL: this is core #11 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 14] ./logs/pe-3-2.cpu_debug.log
HAL: device init() KERNEL: this is core #12 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 03] ./logs/pe-0-3.cpu_debug.log	HAL: device init() KERNEL: this is core #13 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 07] ./logs/pe-1-3.cpu_debug.log	HAL: device init() KERNEL: this is core #14 KERNEL: NoC queue init, 64 packets KERNEL: NoC driver registered HAL: task init() KERNEL: [receiver], id: 1, p:0, c:0, d:0, addr: 4000251c, sp: 400086f4, ss: 4096 bytes KERNEL: free heap: 485368 bytes KERNEL: HellfireOS is up 11] ./logs/pe-2-3.cpu_debug.log	40008424 20 00 3e 00 0f 00 e8 03 88 13 f4 01 05 00 0f 00 .>..... 40008434 7b 92 90 99 03 b6 de c4 9d 55 28 f5 af c5 fa 8b {.....U{..... 40008444 bd a5 13 67 fc 07 27 ca db 67 67 8d 66 54 96 8c ...g...'.gg.ft.. 40008454 d9 6a ba c5 cc 1b e2 00 3b 66 08 83 66 ed 6a a5 .j.....;f..f.j. hf send(): channel=15 15] ./logs/pe-3-3.cpu_debug.log

Press B to bring up the window selection menu.

```
Select window
00 ./logs/pe-0-0.cpu_debug.log
01 ./logs/pe-0-1.cpu_debug.log
02 ./logs/pe-0-2.cpu_debug.log
03 ./logs/pe-0-3.cpu_debug.log
04 ./logs/pe-1-0.cpu_debug.log
05 ./logs/pe-1-1.cpu_debug.log
06 ./logs/pe-1-2.cpu_debug.log
07 ./logs/pe-1-3.cpu_debug.log
08 ./logs/pe-2-0.cpu_debug.log
09 ./logs/pe-2-1.cpu_debug.log
10 ./logs/pe-2-2.cpu_debug.log
11 ./logs/pe-2-3.cpu_debug.log
12 ./logs/pe-3-0.cpu_debug.log
13 ./logs/pe-3-1.cpu_debug.log
14 ./logs/pe-3-2.cpu_debug.log
15 ./logs/pe-3-3.cpu_debug.log
Press ^G to abort
```

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