ALOE Session 1: Introduction to ALOE

Ismael Gomez, Vuk Marojevic, Antoni Gelonch
Universitat Politècnica de Catalunya

April 2011

1. Objective

This session introduces ALOE and guides the installation of the framework on a PC running Linux. It provides a summary of the ALOE concept and principal functionalities as the basis for the following sessions.

2. Overview

- ALOE concept and system approach
- ALOE utilities
- Download and install ALOE for Linux
- Execute an example waveform
- Experience some ALOE functionalities

3. Requirements

- PC running Linux, kernel 2.6.21 or above
- Basic Linux user skills

4. ALOE versions

We continuously evolve the ALOE framework and tools. The different ALOE versions are available at http://flexnets.upc.edu/downloads/source/. This session is compatible with ALOE version 1.3. Consult http://flexnets.upc.edu/trac/wiki/ALOEedu for updates.

5. Introduction to ALOE

5.1. The ALOE concept

The software-defined radio (SDR) concept envisages dynamic waveform reconfigurations. This, however, needs software and hardware support and computing resource management, in particular. The FlexNets initiative publishes research results on this topic and provides the abstraction layer and operating environment (ALOE), an open-source SDR framework with cognitive computing resource management capabilities. ALOE supports partial or total reconfigurations of the transceiver digital signal processing chain (waveform) while facilitating the deployment of waveforms on heterogeneous and distributed hardware resources. The main attributes and functionalities of ALOE are:

- **Flexibility** – be able to trade implementation efficiency against flexibility. An efficient implementation makes best use of the available computing resources (low resource overhead), whereas a flexible solution allows for dynamic reconfigurations at the cost of some resource overhead.
- **Execution control** – coordinate execution across the entire distributed computing system.
- **Abstractions** – hide platform details and heterogeneity from radio applications, enabling portability.
- **Data packet oriented messaging** – packet-oriented instead of processor or device-specific communication mechanisms.
- **Parameter control** – runtime signal (parameter or variable) management.
- **Resource monitoring** – computing system/environment awareness.
- **Computing resource management** – efficiently manage the distributed and limited computing resources.

5.2. ALOE layers

While defining a common framework for developing and deploying SDR applications it is important to eliminate any platform (hardware and supporting software) dependency. Radio applications are built through a set of precedence-constrained modules. (These modules may also be called “objects”.) Each module represents a more or less complex signal processing block that acquires information from the preceding modules in the processing chain and delivers the processed information to the following modules. ALOE assumes that module interfaces are unknown at design time. This enables dynamically composing and recomposing processing chains at execution time while integrating the modules that assemble the desired waveform.

Figure 1 illustrates the ALOE layers. The hardware layer typically consists of several processors or processing elements (PEs), which are physically interconnected. The ALOE Layer abstracts the hardware platform, providing a homogeneous execution environment, the ALOE platform, to applications. The abstract application layer models

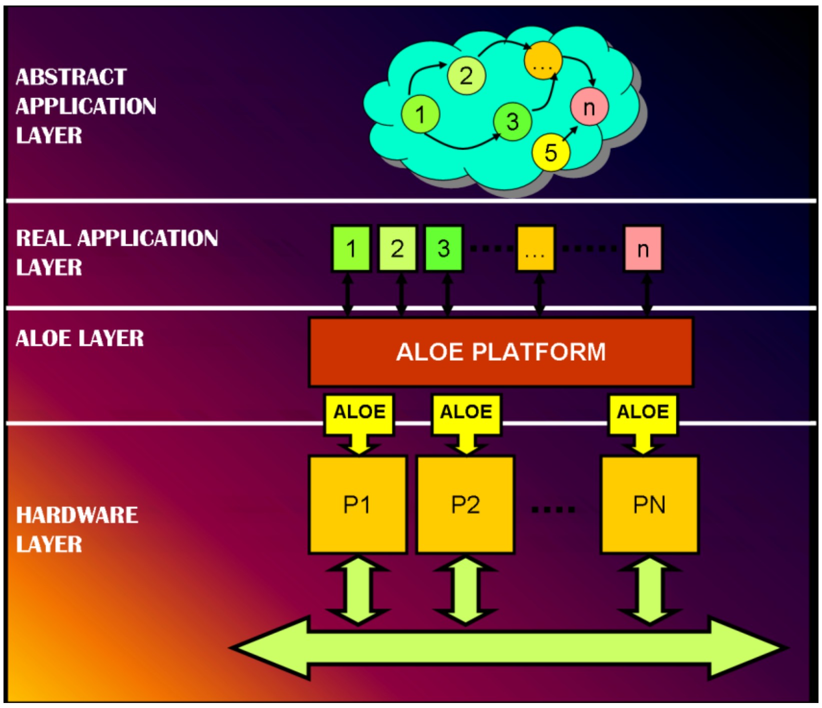

Figure 1 - ALOE layers.

a radio application or waveform by means of the task graph. It abstracts the waveform, providing information about the involved tasks (modules or objects), their precedence constraints and data flow requirements. The real application layer uses the services or functionalities provided by the ALOE layer for assembling the complete waveform and distributing its modules among the available computing resources.

5.3. ALOE architecture

The following figure shows the relation between the different ALOE components and libraries. The application software (here represented by a single module) uses the ALOE services to interact with its environment. These services are accessible as function calls; the ALOE software library contains their implementation. The basic operations provided by the software library may require profound platform or hardware management. The ALOE hardware library makes these issues transparent to the software library. It takes advantage of the available hardware services and operating system tools, if present.

The ALOE software components (ALOE Software Daemons) are accessed though the ALOE software library and perform several tasks for successfully running a waveform on distributed computing resources. The implementation of these components is platform independent and, hence, directly portable to other platforms (if the hardware library is available on these). A short description of the software daemons and their functionalities follows, where MAN is an acronym for manager.

- CMD MAN: Provides a central access to ALOE for higher level control applications, such as GUIs and software development tools.
- HW MAN: Automates the computing resource management for a dynamic allocation and reallocation of computing resources.

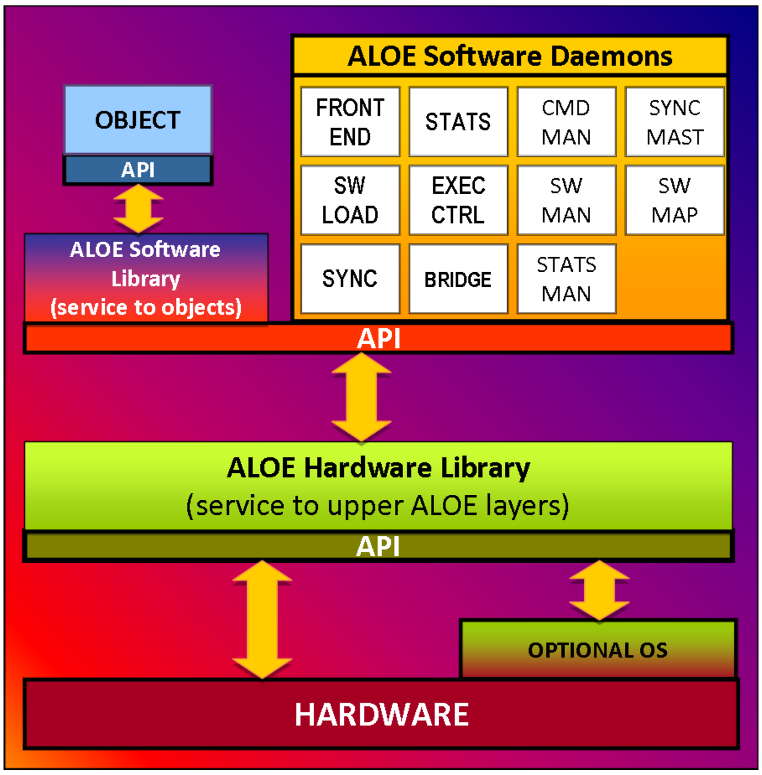

Figure 2 - ALOE architecture.

- SW MAN: Administrates the waveform and module repositories.
- STATS MAN: Provides initialization parameters and monitors the evolution of variables.
- BRIDGE: Acts as a link for data transfers between connected PEs.
- SYNC MAST: Provides the time reference for all PEs.
- FRONT-END: Routes the ALOE control packets among the daemons and gathers the hardware status information.
- SW LOAD: Assigns interfaces and other local resources to modules and their data flows.
- EXEC CTRL: Ensures that every software module is correctly running under the given quality of service (QoS) constraints (real-time computing resource requirements).
- STATS: Captures and modifies module variables and parameters.
- SYNC: Synchronizes the local time with the remote time reference.

5.4. Computing Resource Management

SDR presents a hard real-time computing challenge with varying (computing) system conditions. The computing resource management framework thus needs to track the states of the computing resources for being able to take advantage of the reconfiguration capabilities of mobile terminals and network elements.

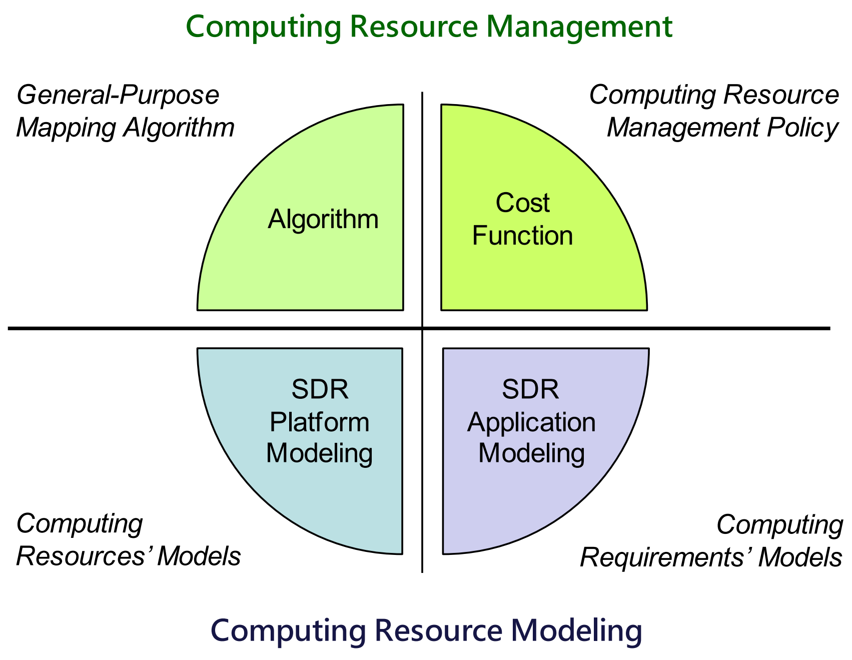
Computing Resource Management
General-Purpose Mapping Algorithm
Computing Resource Management Policy
Algorithm
Cost Function
SDR Platform Modeling
SDR Application Modeling
Computing Resources' Models
Computing Requirements' Models
Computing Resource Modeling

Figure 3 - Computing resource management framework.

Our computing resource management framework is modular. It features the computing system modeling on the one hand and management mechanisms on the other (Figure 3). The SDR computing system modeling captures the SDR platforms' computing resources and the SDR applications' computing requirements. We, therefore, suggest equivalent metrics for modeling computing resources and requirements: million operations per second (MOPS) and mega-bits per second (Mbps), particularly, model the processing and interprocessor data flow capacities and requirements.

The ALOE computing resource management is based on two simple time management principles: time slots and pipelining (Figure 4). This facilitates the synchronized execution of modules on distributed computing resources, while taking advantage of

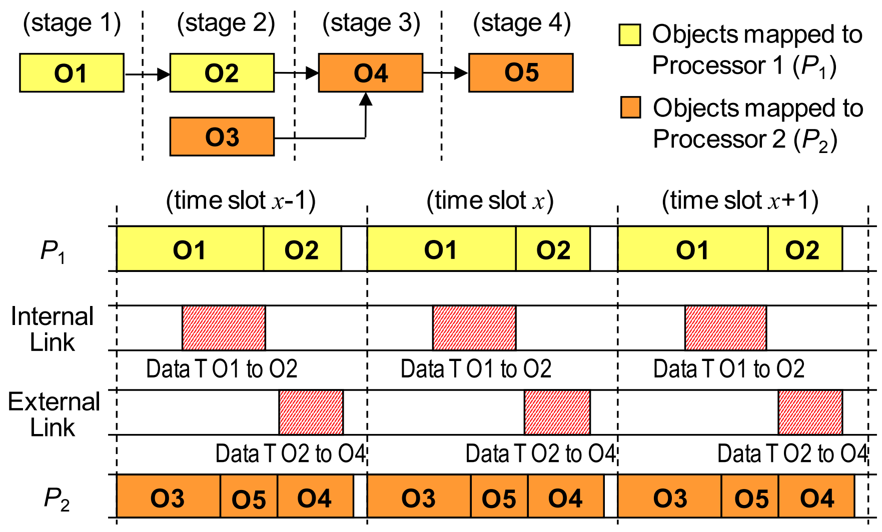

Figure 4 - Time slots and pipelining.

the continuous data flow that characterizes wireless communications. Based on these principles, ALOE applies general-purpose mapping algorithms and problem-specific cost functions. The cost function implements the computing resource management objective or policy while guiding the allocation of computing resources to computing requirements in a controlled manner.

6. Procedure

The core of the ALOE project is the ALOE framework, which incorporates the corresponding interfaces (APIs) for accessing the ALOE tools. We proceed with the installation of ALOE before experiencing some of its tools.

6.1. ALOE Installation

The latest ALOE releases can be downloaded from the FlexNets web site. This session is compatible with ALOE version 1.3. ($ALOE here stands for aloe-1.3.)

The installation of ALOE basically comprises downloading the source code and running standard configure and compiling scripts. Before compiling, make sure you have the termcap and readline libraries installed on your system. In a Debian-based Linux distribution, type the following command in a shell:

```
sudo apt-get install libreadline5-dev
```

If you are running Fedora, use yum to install the libraries:

```
sudo yum install readline-devel
```

Then download, untar and compile ALOE:

```
wget http://flexnets.upc.edu/downloads/source/$ALOE.tar.gz
tar xzvf $ALOE.tar.gz
cd $ALOE
./configure
make
sudo make install
```

Remember that you need to substitute $ALOE with an available ALOE distribution; aloe-1.3, for instance.

Before running ALOE, we have to tune some kernel parameters to enable larger messages buffers. You may edit the *sysctl.conf* file to make these changes permanent. Therefore, add the following lines to */etc/sysctl.conf*:

```
kernel.msgmnb=1048576
kernel.msgmax=10485760
kernel.msgmni=128
```

You need to log in as root for editing the file. Now run sysctl to make the changes effective:

```
sudo /sbin/sysctl -p
```

Now you can run a simple example waveform, which is also provided in the download package.

6.2. Running a waveform

Launch ALOE from the ALOE installation directory (*$ALOE/*—the directory where you extracted and installed the ALOE source package on your PC):

```
sudo runph
```

This will open the ALOE prompt where you can enter ALOE commands. Should you experience problems while trying to launch ALOE, the reason may be that a previous ALOE session is still running. If this is the case, use

```
sudo runph –f
```

to terminate the process and try launching ALOE again.

ALOE defines different execution states (Figure 5). The following list briefly defines these states, numbered from 1 to 5 (as displayed by the *execinfo* tool):

- LOADED (1): Initial status after the component has been loaded. The component waits until the status changes.
- INIT (2): During the initialization phase, the object retrieves parameters that configure its operation. Some tasks can be performed without real-time deadlines, e.g.: compute filter coefficients, look-up-table values, etc.
- RUN (3): The signal processing is performed in this state with real-time constraints that need to be met. The waveform module checks for new samples on its input interfaces (typically data vectors), processes these, and sent them to the output interfaces.
- PAUSE (4): The component pauses its execution; data is neither retrieved from nor sent to any interface.

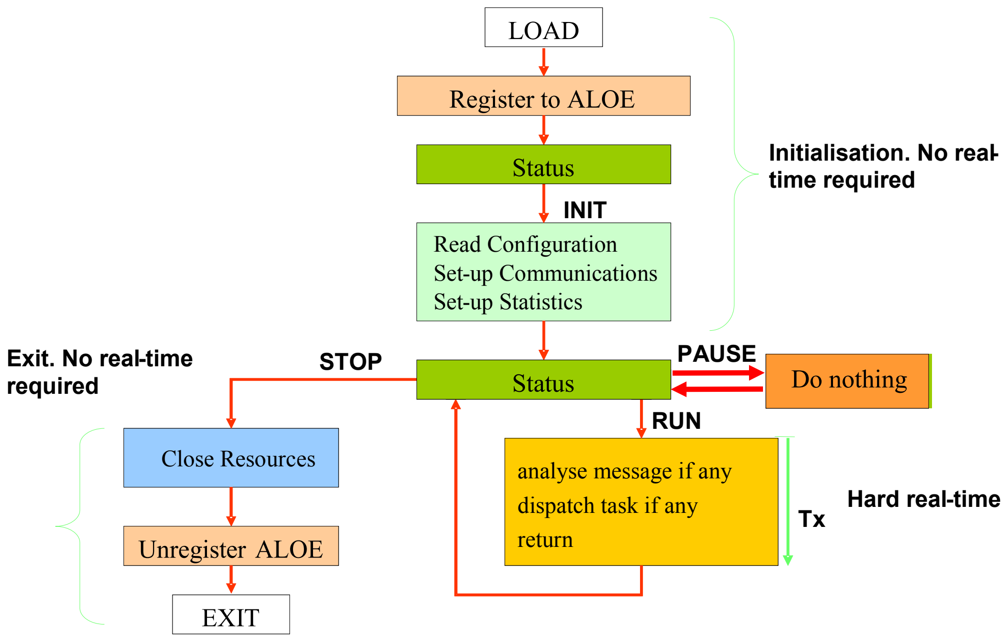

Figure 5 - Waveform execution flow chart.

- STEP (5): The component executes during one or a finite number of time slots. This is useful for debugging modules or entire waveforms.

To load the ***example*** waveform, type

```
runph$: phload example
```

Section 5.4 introduced the ALOE computing resource management approach, which maps waveform modules to processing elements (PE) as a function of the real-time computing constraints. If all computing requirements can be met with the given computing capacity, the waveform will run in real-time. Hence, real-time failures during execution are not supposed to occur frequently, but rather sporadically. A common reason for this is a runtime variation of the waveform’s data processing requirement. A deadline violation implies that the corresponding data frame is lost, increasing the bit error rate.

After loading the ***example*** waveform, the shell informs about the waveform’s computing requirements and the platform’s computing resources. Search for the following lines:

```
-- c vector --
47.00     358.00     572.00     60.00     47.00     47.00
```

The *c vector* contains the processing requirements in multiply-accumulate operations (MACs) per time slot. The processing capacities are modeled by the *C vector*. The processing capacity is 600 million MACs per second (MMAC/S) by default (see

/usr/local/etc/platform.conf, **macs** field). With a time slot of 10 ms this is equivalent to 6 million MACs per time slot (MMAC/TS). Since running the waveform on a single PE with a processing capacity of 6 MMAC/TS, the vector size is one:

```
-- C vector --
6000000.00
```

These numbers indicate that the waveform can be loaded and executed in real-time. The mapping information is contained in

```
-- P_m vector --
0     0     0     0     0     0
```

It indicates that all six modules of the ***example*** waveform have been mapped to a single PE, PE0.

You can now initialize and run the waveform:

```
runph$: phinit example
runph$: phrun example
```

Use the *phpause* tool to temporary stop (pause) the waveform execution:

```
runph$: phpause example
```

If continuous messages start appearing on your shell, you can still type `phpause example` and press enter to pause the waveform execution.

We use the *execinfo* tool to verify that each component is in the PAUSE state (status 4):

```
runph$: execinfo example
```

Observe that the execution time of each module is very low since not performing any useful signal processing tasks. Now step one time slot further:

```
runph$: phstep example
```

During this time slot each component will process one data block and deliver it to the next pipelining stage. Observe—using *execinfo*—that the time slot counter increased by one.

6.3. Basic resource management tools

The *execinfo* tool also informs about the execution time and the real-time performance (number of deadlines misses – RT FAULTS) for each component. Real-time violations depend on the throughput of the waveform and the performance of your system, in this case the general-purpose processor. The system's computing capacity thus determines the real-time processing capabilities. For the example waveform we should observe that all real-time deadlines have been met.

For illustrative purposes we now force real-time failures by increasing the waveform computing demands. We, therefore, configure the source module so that it generates more bits:

```
runph$: statset example source numbits 2048
```

The previous command increases the waveform's throughput requirements.

The shell outputs a message whenever a deadline violation occurs. Try running the waveform for one time slot and observe the total number of deadlines misses:

```
runph$: phstep example
runph$: execinfo example
```

If the number of deadline violations (RT FAULTS) has not increased, increase the variable *numbits* further and run the example waveform during several time slots:

```
runph$: statset example source numbits 1000000
runph$: phrun example
```

If continuous messages start appearing—indicating real-time violations—type `phpause example` and press enter to pause the waveform execution. If not, pause the waveform execution, further increase the *numbits* variable, and run the waveform again. Use the *execinfo* tool to observe the number of packet losses (RT FAULTS).

You can check all loaded waveforms with

```
runph$: applist
```

Unload the example waveform and try loading the ***example2*** waveform:

```
runph$: phstop example
runph$: phload example2
```

The computing resource requirements of the ***example2*** waveform cannot be met with the available computing resources. The resource manager (HWMAN) detects this and does not load the waveform. You should observe an output similar to Figure 6.

Figure 6 - Output after trying to load the example2 waveform.

```
-- c vector --
47.00 358000000.00      572000000.00      60000000.00 47000000.00 47.00
-- B matrix --
100000.0

-- C vector --
6000000.00
HWMAN: Map (allocate) software: Mapped with cost 1000000.00
HWMAN: Error mapping software: Platform does not have enough resources
for the application
      Application can't be loaded.

HWMAN: Error mapping software: Error in mapping.
```

Observe that the sum of processing requirements,

47 + 358 000 000 + 572 000 000 + 60 000 000 + 47 000 000 + 47 MACs/TS,

by far exceeds the total processing power of 6 MMAC/TS.

Use Ctrl-C to exit the ALOE prompt. This will unload all loaded applications, if any, unregister ALOE and terminate its execution. With this we finish ALOE session 1. Please send us your feedback (flexnets.pmt@upc.edu).