RTOS extensions for dynamic hardware / software monitoring and configuration management

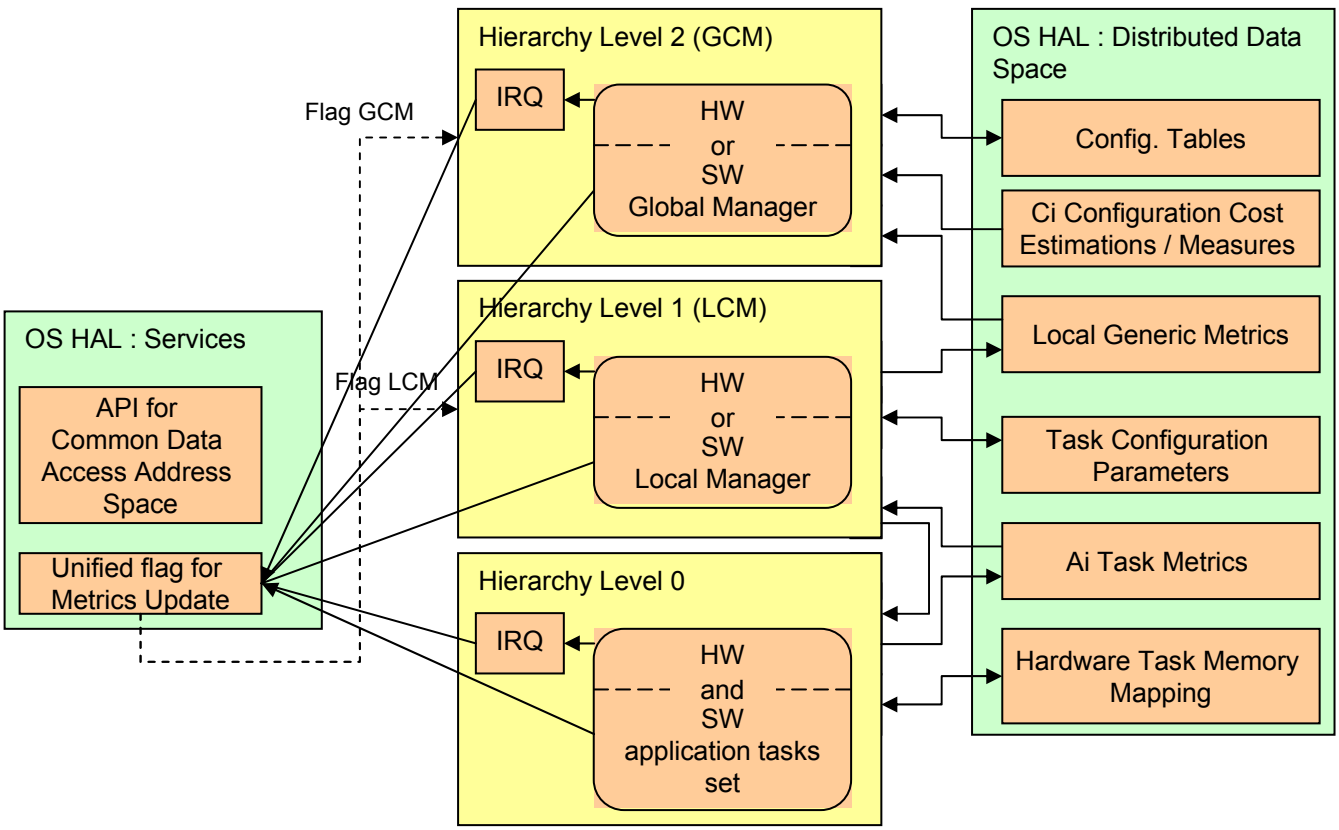
Yvan Eustache, Jean-Philippe Diguet, Milad Elkhodary
LESTER (UBS University / CNRS)
Rue de Saint-Maudé, 56325 Lorient Cedex - France
E-mail : yvan.eustache@univ-ubs.fr, jean-philippe.diguet@univ-ubs.fr

Raar Project supported by French FNS ACI JC9169

Context : How to manage the behavior of an embedded system in a fluctuating environment ?

• Signal & image processing applications in **embedded systems** involve complex power-efficient resources, but the dynamic behavior of mobile applications (data & user dependent) doesn't fit with a general purpose architecture.
• **Dynamic reconfigurable platform** composed of processor (software for flexibility) and reconfigurable logic (hardware for performance) are promising candidates for adapting resources to application changeable requirements.
• This work focuses on a flexible & unified implementation of self-adaptive systems on reconfigurable architectures. It is based on a couple of local & global reconfiguration managers. We describe how managers are implemented in the context of an usual RTOS and the new services we add for HW and SW monitoring, reconfiguration decision and reconfiguration control which also includes hardware and software interface modeling.

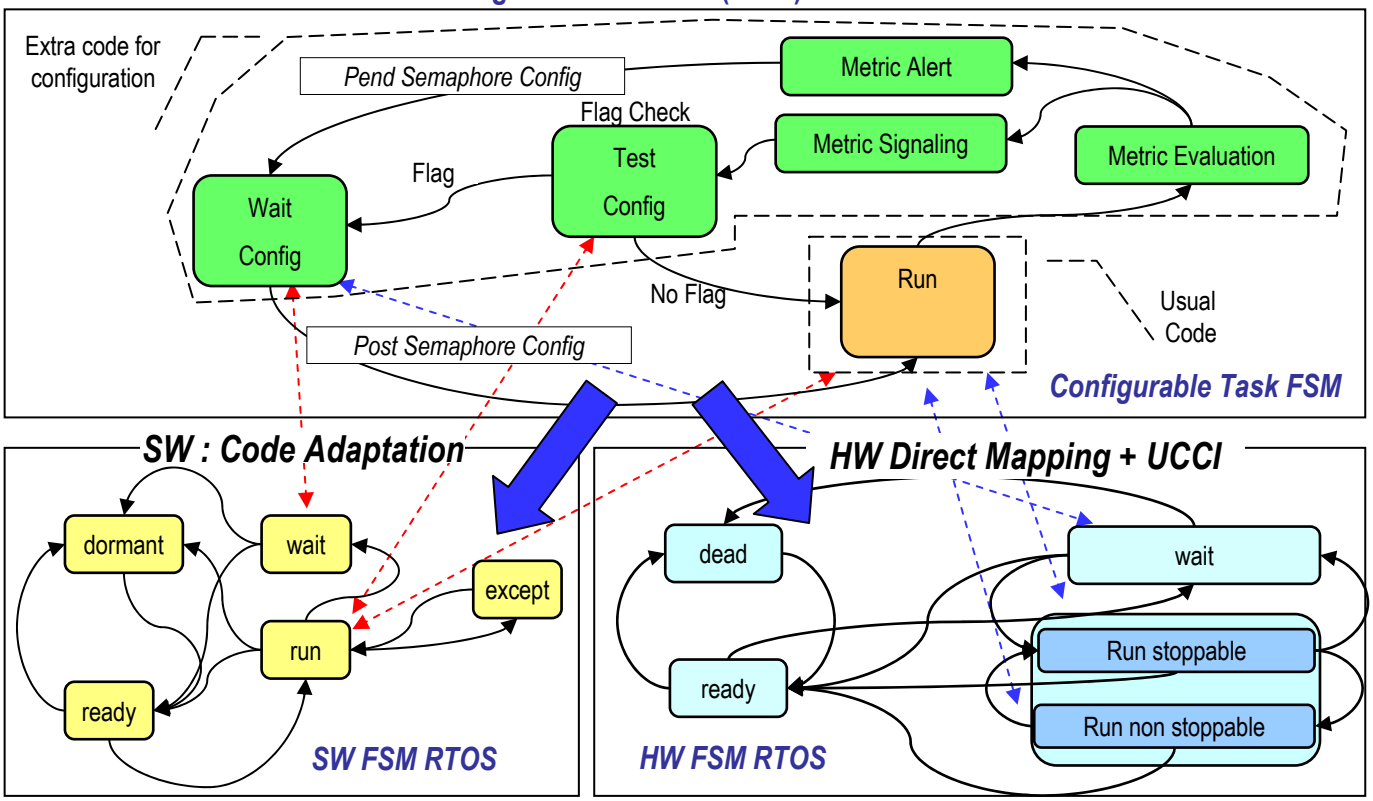
Communication between tasks and configuration managers



A two step configuration strategy :

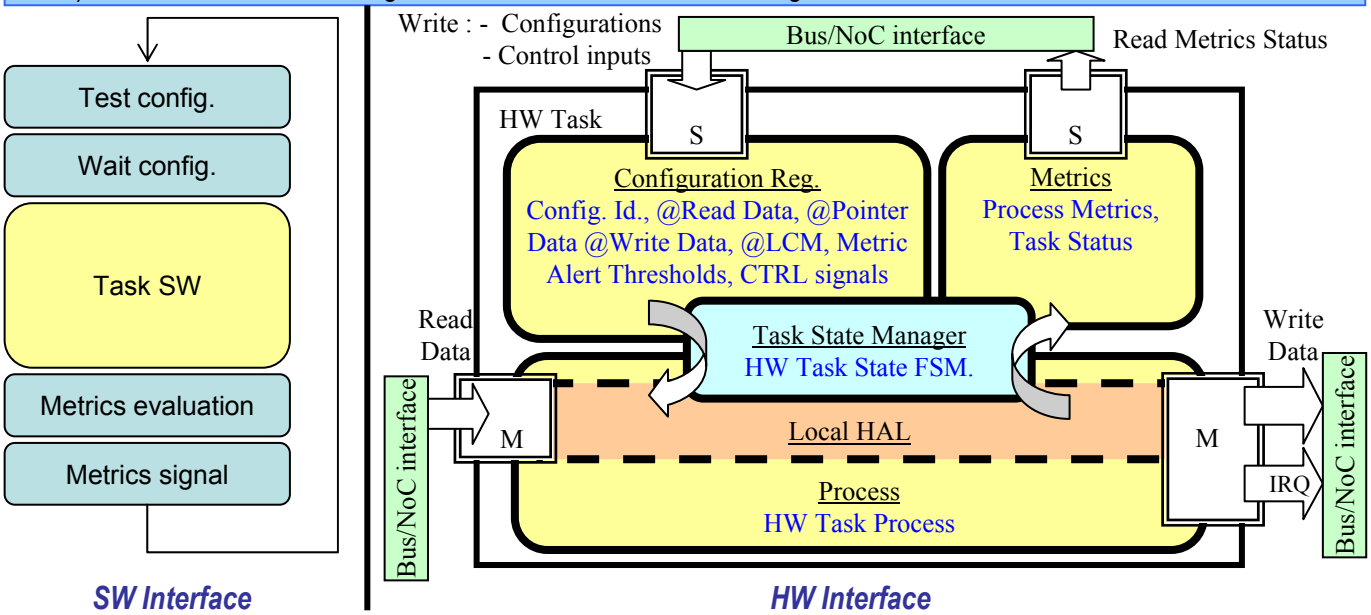
- **The Local Configuration Manager (LCM) for each application**
  - ➔ supervises all tasks,
  - ➔ reads Metrics from each task and computes them,
  - ➔ provides the desired algorithm configuration to the upper configuration level.
- **The Global Configuration Manager (GCM)**
  - ➔ has a global view of the system and is application independent,
  - ➔ receives data from sensors (Gas Gauge, OS Timers, Application QoS),
  - ➔ and is in charge of global system parameters and HW/SW implementation.

The Unified Communication and Configuration Interface (UCCI).



The abstracted FSM modelizes the task configuration independently from the implementation.

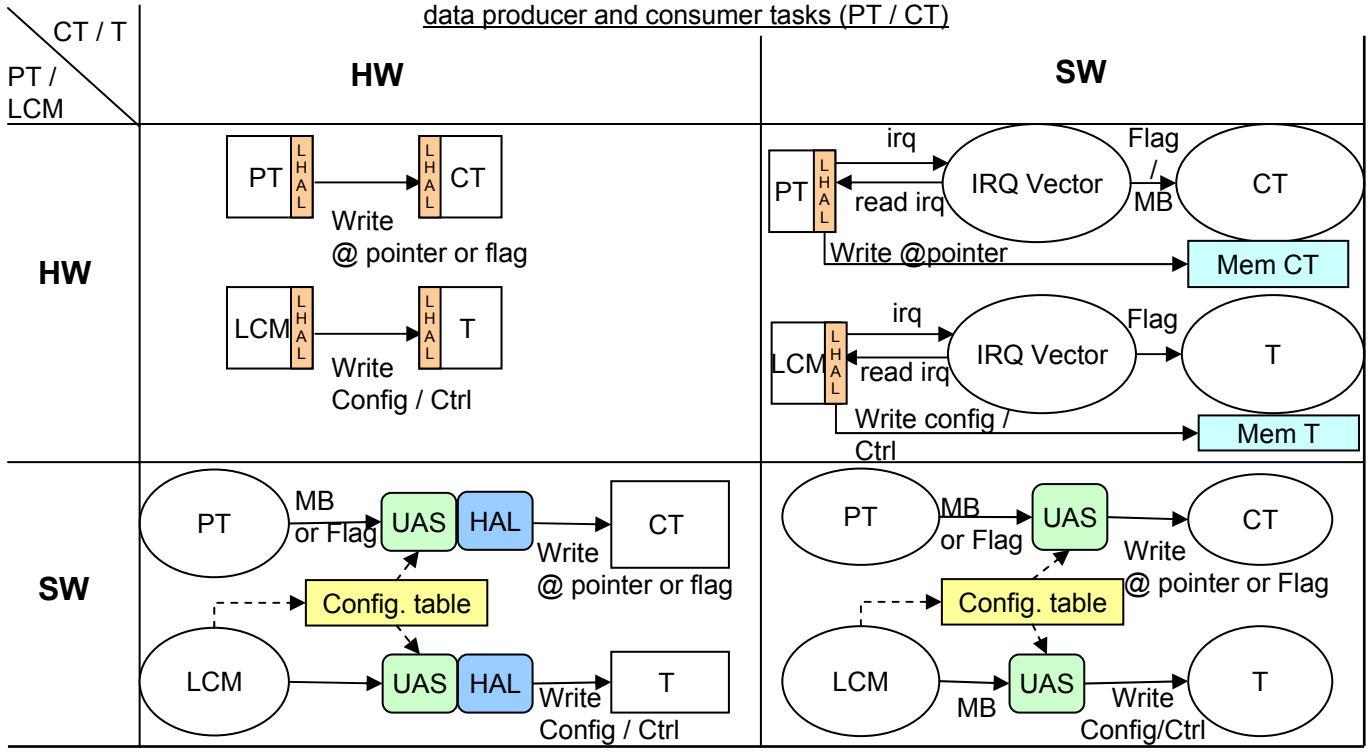
1.) **IF** a new configuration flag is raised ("Test Config." state)
2.) **THEN** the task moves to "Wait Config." state
and pends for a new configuration on a semaphore request.
3.) **WHEN** the semaphore is released, the task moves to the "Run" state.
4.) The task computes specific its own metrics
5.) And signals the Local Manager in order to read them
6.) Or modifies itself its configuration and alerts the Local Manager.



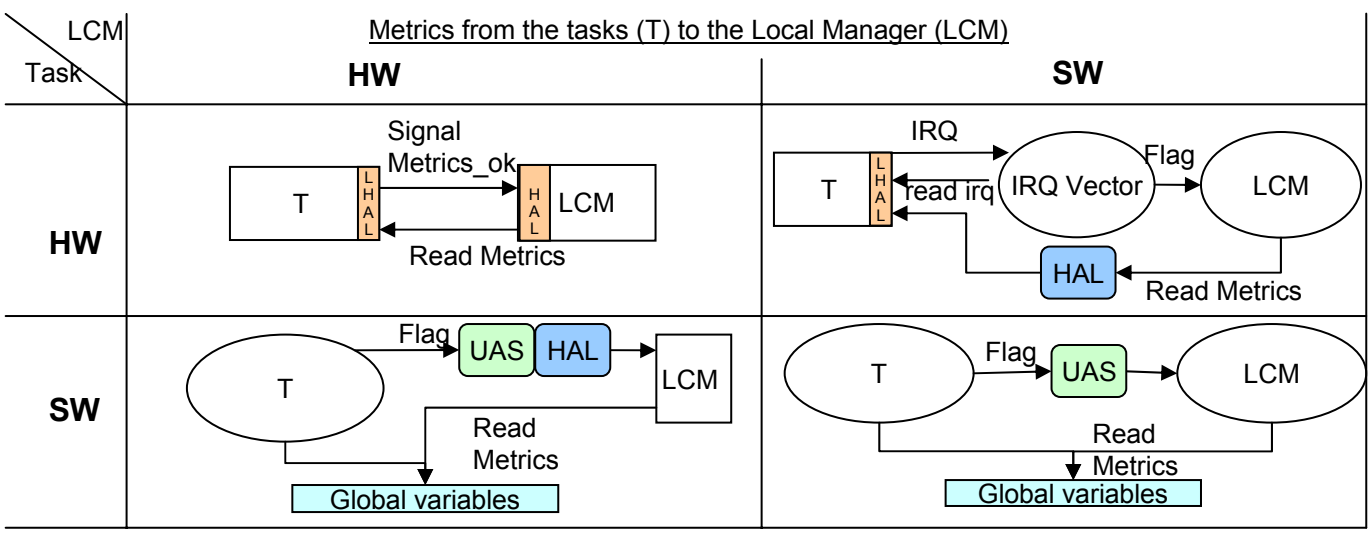
The SW Interface implements four new functions : a test of a new configuration, a "wait state" semaphore request, a metrics evaluation and a metrics signalization .
The HW Interface implements **configuration registers** like the address of LCM, preceding and following tasks in the process flow, control signals to start and stop the task, pointers to the valid produced data and the configuration. It implements also **status registers** for the metrics and task status. The **local HAL** manages the communication of data between tasks. **The Task State Manager** supervises the task state with a FSM.

Deterministic Non Blocking Operations of Communication and Synchronization cases.

@ pointer for MailBox (MB), flag event and configuration between the Local Manager (LCM) and tasks (T) or data producer and consumer tasks (PT / CT)

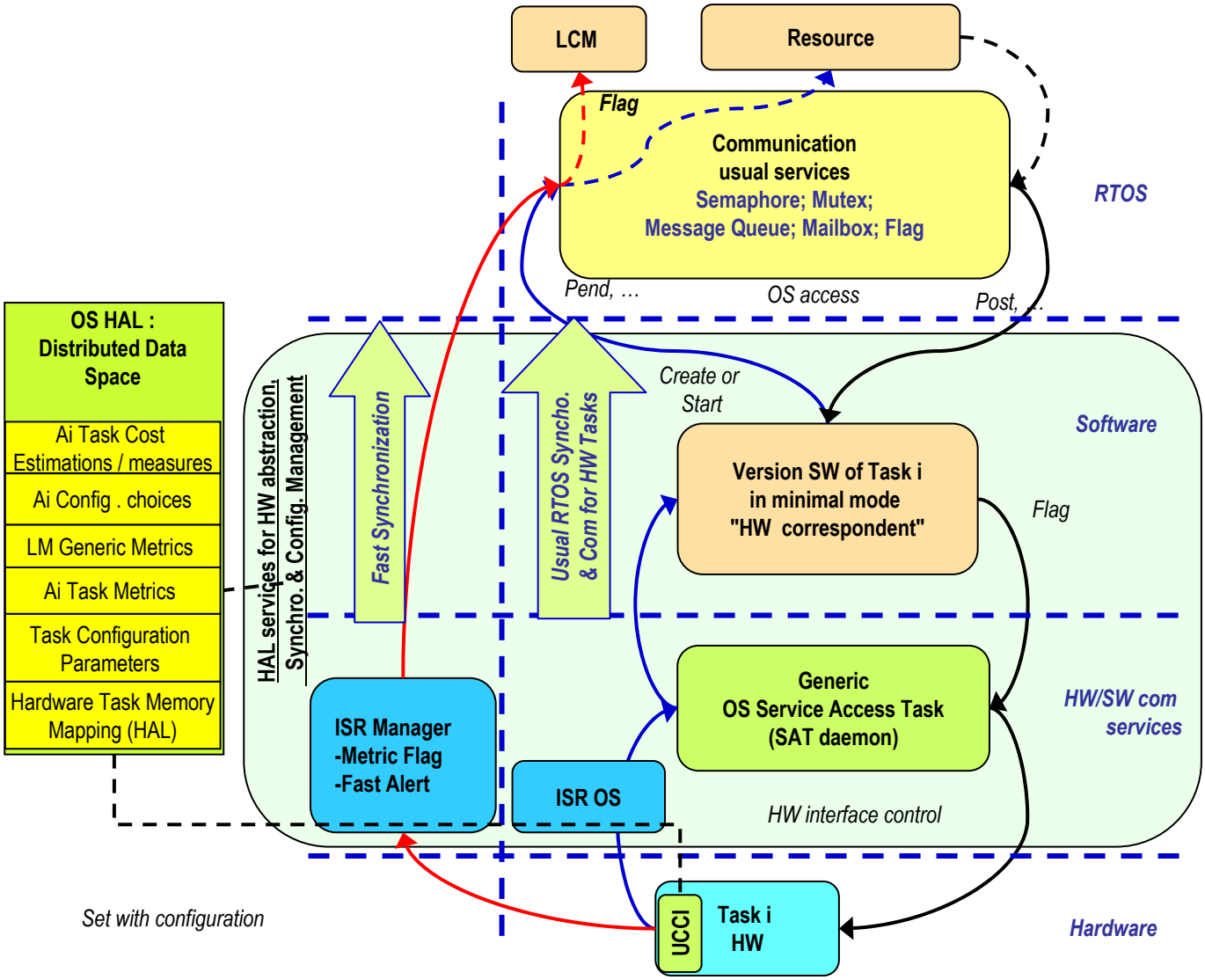


The Abstraction Layer routes communication, synchronization and configuration data between hardware and software tasks independently to the implementation. Each hardware task required a Local Hardware Abstraction Layer (LHAL) configured by the Local Manager whereas software tasks share the HAL service of the RTOS.
HW ➔ HW : direct from the producer to the consumer through the LHAL.
HW ➔SW : producer sends an IRQ transformed into a flag by the SW IRQ Vector and writes the message in the memory.
SW ➔ SW : direct from the producer to the consumer through the Unified Access Service.
SW ➔ HW : direct from the producer to the consumer through the Unified Access Service then RTOS HAL service.



HW ➔ HW : The task signals the LCM which reads the metrics.
HW ➔SW : producer sends an IRQ transformed into a flag by the SW IRQ Vector. The LCM then reads the metrics through the HAL.
SW ➔ SW : The task signals the LCM through the Unified Access Service and writes metrics into a global variable read by the LCM.
SW ➔ HW : The task signals the LCM through the Unified Access Service and HAL and writes metrics into a global variable read by the LCM.

HAL services for HW abstraction, synchronization and configuration management.



Necessary when a hardware task needs to pend on a semaphore, mutex or flag event for synchronization or communication, the Service Access Task (SAT) is a daemon that tests, periodically in a non blocking way, flags set by the hardware tasks through ISR. In A software version of the hardware task is started, uses an software "correspondent Hardware" event of RTOS and waits.
For a fast signalization (alert or metric flag) the SAT can be bypassed by a specific ISR sending directly a software flag to the LCM.

Conclusion

This work is a part of the RaaR project dedicated to auto-adaptive systems on embedded reconfigurable SOCs. In this paper, we have presented the HAL services and the interfaces we have designed for implementing hardware and software tasks. The first interest of our work is that both hardware and software task specifications can be implemented in this new adaptive architecture with minor modifications through a systematic encapsulation procedure. The second point is that all usual OS services are available transparently between hardware and software tasks. The last point is the integration of services required for the implementation of the configuration controller. As a proof of concept we currently implement a smart camera application for object tracking on a Stratix device.