Partitioned Embedded Architecture based on Hypervisor: The XtratuM approach

M. Masmano, I. Ripoll, A. Crespo, S. Peiro
Instituto de Automática e Informática Industrial (AI2)
Universidad Politécnica de Valencia

XtratuM
www.xtratum.org
Outline

● Real Time Embedded Partitioned Systems
● XtratuM Hypervisor Overview
● XtratuM Trustability enforcement
● XtratuM FSM model
● Conclusions
RT Embedded focus on build highly secure systems:

- The separation kernel approach (SK) [Rushby, 1981].
- Recently a variant of SK has gained acceptance in avionics: partitioning kernel SKPP [NSA, 2007].
- SKPP basic requirements: CC [ISO/IEC, 2006]
  - Separate and isolate multiple partitions (TSP).
  - Protect all resources (including CPU, memory, devices) from unauthorized access.
  - Mediate information flows between partitions.
  - Audit services.
Two technologies achieve partitioned systems in desktop/server and recently embedded computing:

- Micro-kernels (μkernels)
  - Architectural solution implement large & complex OS.
  - Basic (μkernel services) and System (servers) Layer.
  - Current μkernels implementations: L4, Mach.

- Hypervisors: Platform virtualisation
  - Native hypervisors more suitable for RT than hosted:
    - Virtual Machine Monitor close to native hardware
  - In embedded systems with RT constrains
    - partial virtualisation seems the most appropriate

Relevant Aspects: OS, Devices: Dedicate/IO Server
### Summary of main differences: Hypervisors vs. μkernels:

<table>
<thead>
<tr>
<th></th>
<th>Hypervisors</th>
<th>Micro-kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Thread management</strong></td>
<td>Only knows about partitions, threads not known/handled</td>
<td>Thread services provided, involve microkernel invocation</td>
</tr>
<tr>
<td><strong>Scheduling</strong></td>
<td>Schedule partitions. Threads not visible.</td>
<td>Schedule threads of partitions.</td>
</tr>
<tr>
<td><strong>Memory management</strong></td>
<td>Sees physical memory area regions associated to each partition.</td>
<td>Manage memory area regions for the OSs. Memory context switch of the OS threads</td>
</tr>
<tr>
<td><strong>Communications</strong></td>
<td>partitions channels or shared memory</td>
<td>Messages as basic IPC between threads.</td>
</tr>
<tr>
<td><strong>Device</strong></td>
<td>Dedicated devices to permit direct access. Guest OSs implement the device driver.</td>
<td>Use device abstractions shared by guest OSs</td>
</tr>
<tr>
<td><strong>Interrupt management</strong></td>
<td>Capture all interrupts and provide a virtual interrupt model to partitions</td>
<td>Capture all interrupts and provide a virtual interrupt model to partitions</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Very small: minimal services closely mapping hardware (5KLOCs)</td>
<td>Larger than hypervisors: more complex to (&gt;12KLOCs)</td>
</tr>
<tr>
<td><strong>Guest OS adaptation</strong></td>
<td>Guest OSs have to be para-virtualised to operate as partitions (update HAL).</td>
<td>Manage the main abstractions of the guest OSs both at the μkernel level: Replication</td>
</tr>
</tbody>
</table>

### Main difference: Level of knowledge of each technology.
XtratuM Hypervisor Overview

- XtratuM is a bare-metal hypervisor designed to meet safety critical real-time requirements:
  - **Strong temporal isolation.**
  - **Strong spatial isolation.**
- Uses para-virtualisation techniques
  - Virtual Machine Monitor close to native hardware
  - Partition code (bare or OS) must be modified to cooperate with the hypervisor.
- Open source.
XtratuM architecture

- Supervisor Partitions
- User Partitions

Supervisor Mode

- Memory Manager
- Scheduler
- IP Communications
- Health Monitor
- Clock & Timers Mgmt
- Interrupt Mgmt

User Mode

- RTOS
- Para-virtualised services
- Partition Control table

- RTS
- Para-virtualised services
- Partition Control table

Hypercall Interface

Peripherals

Non shared

Shared
Trustability enforcement: Approach

• **Idea:** Extend the hardware trustability to the hypervisor:

• **Assume**
  - Hardware (LEON2) provides trusted mechanisms:
    - Internal processor registers (PRregs) work properly when used in the correct way.

• **Prove**
  - XtratuM hypervisor correctness properties.

• **Approach**
  - Ensure hardware is programmed correctly to prove that the hypervisor fulfills correctness properties.
Trustability enforcement: Properties

- XtratuM is build upon trustable design criteria:
  - Strong Spatial isolation: partition memory protection.
  - Strong Temporal isolation: fixed cyclic scheduler.
  - Fine-grained resource allocation using XML.
  - Robust communication mechanisms: ARINC Channels.
  - Interrupt Model: secure partition interrupt model.
  - Fault Management Model: detection and handling.
- Small and Non-preemptable hypervisor:
  - Reduce design complexity, validation and formal verification (certification).
Trustability enforcement: TSP

- XtratuM provides TSP (Time and Space Partitioning)
- TSP System specification

- Uses XML format to describe the System:
  - Hypervisor
  - Cyclic Plan
  - Partitions
  - HM
  - Devices
  - Communication
Trustability enforcement: XML

- System specification using XML
- Specification translated to static data structures: XM_CF
- Hypervisor uses XM_CF to prove correctness properties

- Fully static (unmodified) system specification ensures all information flows specified are secured [SKPP/CC]
Trustability enforcement: HM

- Fault Management Model: Health Monitor
  Detect (Events) and react (Actions) to anomalous states

- HM Events (HW):
  - Exception, Interrupt, Trap

- HM Events (SW):
  - Partition/Hypervisor self-checks

- HM Actions:
  - Ignore
  - Propagate
  - Shutdown/Reset
  - Suspend/Halt

![Diagram of fault, error, and failure management]

Detect (Events) and react (Actions) to anomalous states.
XtratuM FSM model

- Hypervisor model: \( \Sigma = (H, HM, \Delta, \Omega, \Pi) \)

- Hypervisor: \( H = (\Lambda, \rho) \)
- Health Monitor: \( HM = \{ hm_i = (hm_{i\text{event}} \to hm_{i\text{action}}) \} \)
- Devices: \( \Delta = \{ \delta_i = (\delta_{i\text{start}}, \delta_{i\text{end}}) \} \)
- Channels: \( \Omega = \{ \omega_i = (\omega_{i\text{type}}, \omega_{i\text{inport}}, \omega_{i\text{outport}}, \omega_{i\text{maxmsg}}, \omega_{i\text{maxsize}}) \} \)
- Partitions: \( \Pi = \{ \pi_i = (\lambda, HM_{act}) \} \)

- XtratuM formal model is a direct mapping (equivalence) of the system specification (XML)
XtratuM FSM model: diagram

- **XtratuM FSM Model**: $XM_{FSM} = (\Omega, \alpha, S, S_0, \Theta, F)$
- FSM is constructed from a given System configuration:
  - States: $\{S_i\}$ set of slots in the cyclic plan.
  - Initial State: $S_0$ boot and partitions loaded.
  - Final State: Halted
  - State Transitions: $S_j = \Theta(S_i, \alpha)$
  - Input Alphabet: $\alpha$
    - nextSlot: scheduler
    - Trap (hypercall)
    - Halt (exception)
    - Interrupt
Verify hypervisor properties of System Specification

At each transition event: \( S_i \rightarrow S_j \), verify:

- Post-conditions on \( S_i \)
- Pre-conditions on \( S_j \)

**Conditions:**

verify \( PRregs \) contents

\[ f^\Omega(S_i) \rightarrow PRregs \]

- \( PRregs \): on LEON2 registers: IOP, IV, WPR, PMS, TIMER

\[ f^\Omega \] : correct values on state \( S_i \) derived from specification \( \Omega \)
Xtratum FSM Model: verification

- **General Conditions:**
  - partitions are executed in user mode
  - hypervisor are executed in privileged mode

- **Spatial Isolation Conditions:** \( \text{event} : S_i \rightarrow S_j \)
  - Hypervisor entry: \( PRregs = f ^{\Omega} (S_i) \)
  - Hypervisor exit: \( PRregs = f ^{\Omega} (S_j) \)

- **Temporal Isolation Conditions:**
  - partition can't interfere other partition execution
    \( (\text{TIMER}_{\text{period}}) = f ^{\Omega}_{\text{clk}} (S_i) \)

- **Abstract Machine Tests (AMT):**
  - Hardware checks on idle periods [CC,2007]
Conclusions

- TSP based architecture.
- Extension of Trusted hardware environment.
  - Extended Interrupt Model.
  - Fault management model: Health Monitoring.
- Fully configurable.
- Predictable and deterministic.
- Small (<60KB) and efficient.
- Closely follow ARINC-653 at partition level.
- Formal Model uses FSM.
- Certifiable.
THANK YOU