Porting a hypervisor to the ARMv8-R Architecture

Porting a hypervisor to the ARMv8-R Architecture

Outline

Outline

- Introduction
- Use cases for a hypervisor in the automotive industry
- Porting of a hypervisor to the Cortex-R52
  - VM Instantiation
  - Memory Protection Unit Configuration
  - Interrupt Management
  - VM Context Switch
  - Peripheral Virtualization
- Conclusions

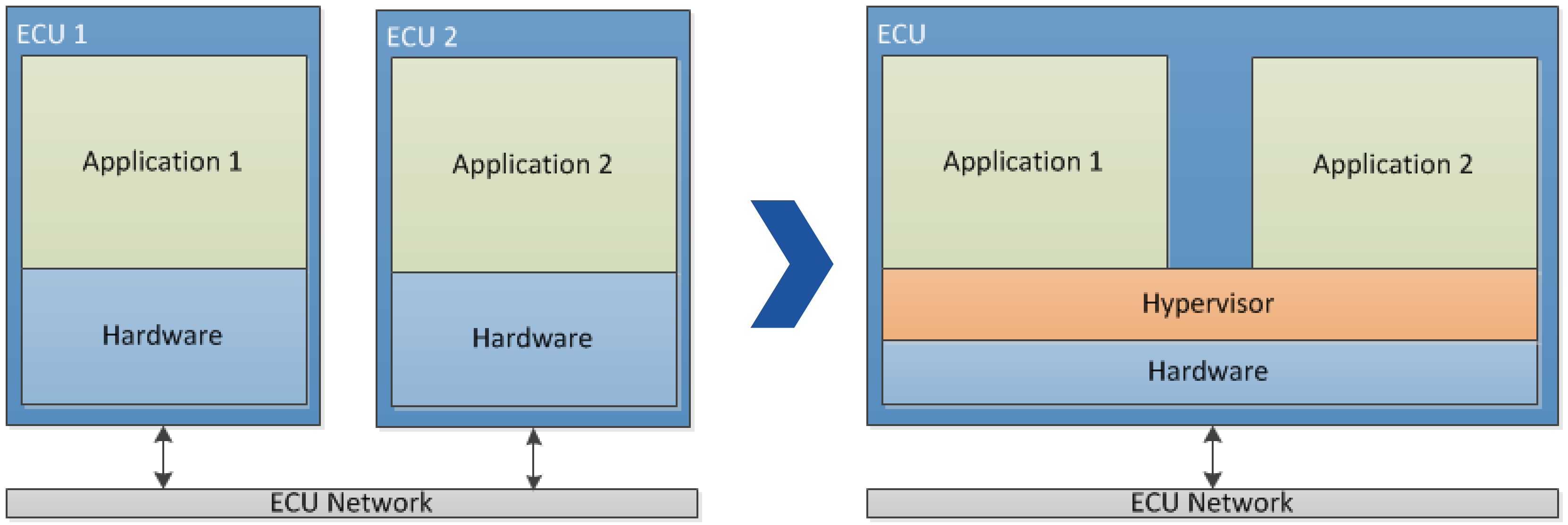
Introduction

– ARM is developing a new processor architecture that is designed to provide features which will help in the task of providing solutions to the automotive market demands

– This presentation focuses on the virtualization features for porting a hypervisor to the Cortex-R52

Use cases for a hypervisor in the automotive industry

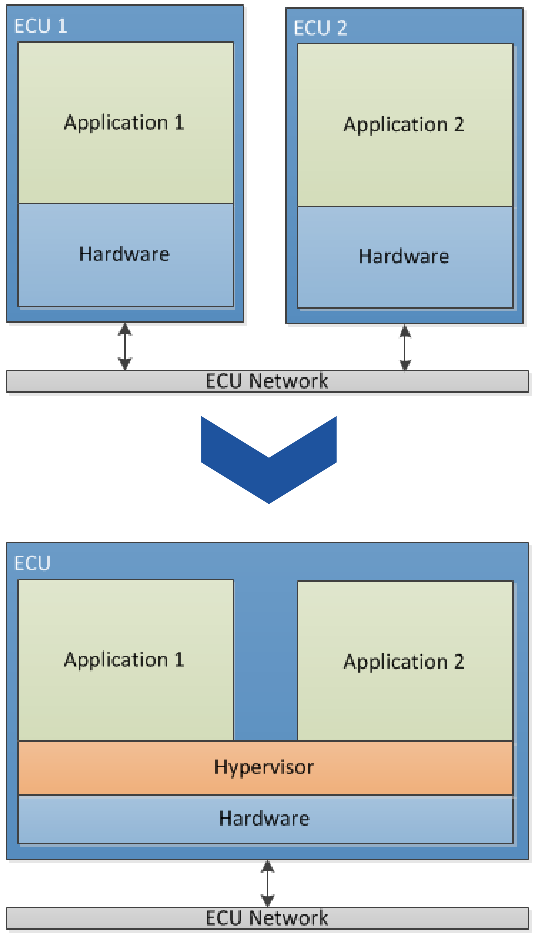
- ECU Consolidation/Integration
  - The hypervisor technology provides an alternative approach for consolidating multiple applications on a single ECU platform whilst maintaining safety



Use cases for a hypervisor in the automotive industry

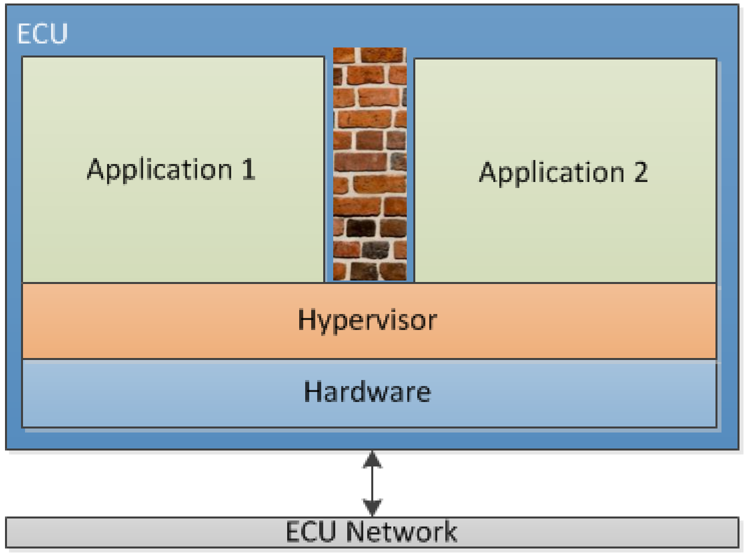
- ECU Consolidation/Integration
  - High flexibility
    - Legacy ECU software can be reused
    - Dynamic partial software update should allow partitions to be independently updated
  - Strong isolation
  - Benefits in cost, weight and network communication by combining ECUs

MCAL and OS impact should be considered



Use cases for a hypervisor in the automotive industry

- Safety/Security
  - A hypervisor offers strong isolation
    - Memory Protection
    - Time Protection
    - Hardware Virtualization
  - Strict partitioning between functionality with differing safety and security requirements necessary



Porting a hypervisor to the ARMv8-R Architecture
Porting a hypervisor

Porting a hypervisor to the ARMv8-R Architecture

- VM Instantiation
- Memory Protection Unit Configuration
- Interrupt Management
- VM Context Switch
- Peripheral Virtualization

VM Instantiation

- This is the process of configuring hardware and software to set up an isolated application container (virtual machine)

- It can be broken into the following steps:
  - Setting up VM Internal state
  - Memory Protection Unit Configuration
  - Interrupt Controller Configuration
  - Configuring trust mode and entering the VM

VM Instantiation – Setting up VM Internal state

- The hypervisor sets up a private data structure
  - ROM data: VM configuration data
  - RAM data: VM state data, including the context memory
    - CPU registers
    - MPU regions and attributes
    - Any virtual hardware context data
    - Interrupt context data

- In the Cortex-R52 a two stage MPU is available
  - One stage is used by the hypervisor to set up regions which are not accessible to any VM
    - Typically, private ROM/RAM and I/O address space
  - The other stage is used by the guest OS running in the VM

Memory Protection Configuration

- Hypervisor sets up VM’s to access to
  - RAM
  - ROM
  - I/O devices

- In the case of AUTOSAR guest software, additional VM internal memory protection regions may be required
  - Needed to support AUTOSAR partitions

- In the Cortex-R52 a two stage MPU is available
  - The hypervisor uses the stage-2 MPU for configuring VM address boundaries
  - VMs use the stage-1 MPU for configuring internal partitions

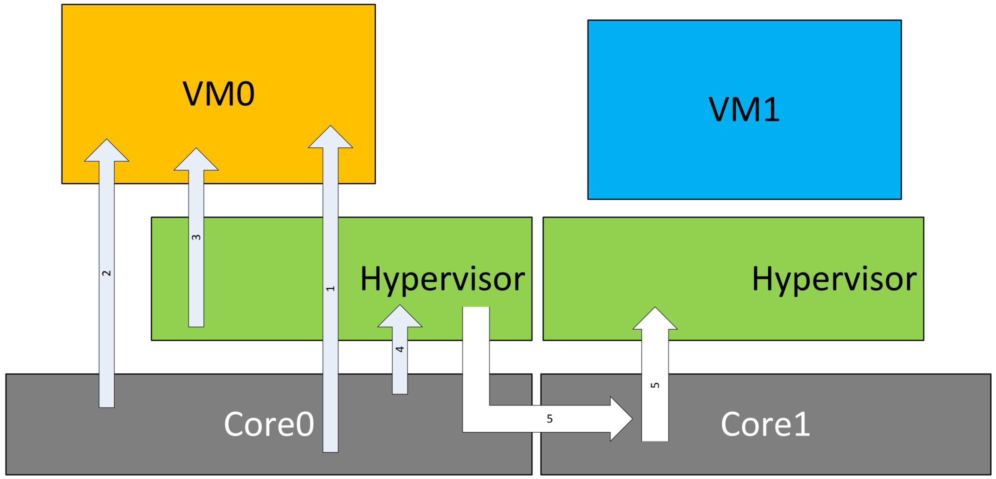
Porting a hypervisor to the ARMv8-R Architecture

Porting a hypervisor

Interrupt Controller Configuration

– The hypervisor sets up the interrupt controller:

1. For sending interrupts to VMs via the hypervisor
2. For sending interrupts directly to VMs
3. For sending software generated interrupts to VMs
4. For sending interrupts to the hypervisor
5. For sending interrupts between parts of the hypervisor on different cores



– Not all of these configurations may be possible simultaneously

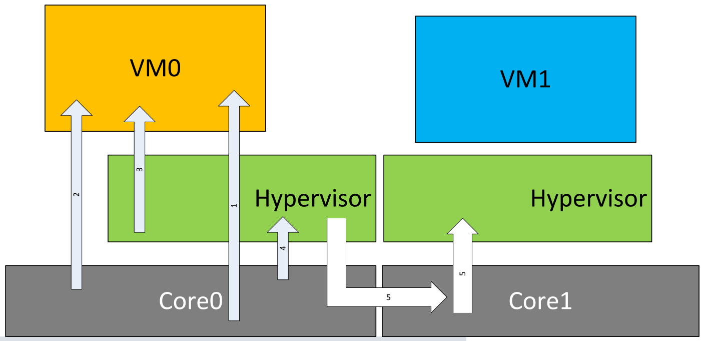
Porting a hypervisor to the ARMv8-R Architecture

Porting a hypervisor

Interrupt Controller Configuration

- This configuration is typically time critical due to the short interrupt latency required in real-time automotive systems

- In the Cortex-R52, the GIC provides
  - A mechanism to route physical interrupts to VMs with no hypervisor intervention → no additional latency for the guest software, provides real time capability
  - A mechanism to route virtual interrupts to VMs where the hypervisor can forward, cancel or queue interrupts (i.e. in the case of a dormant VM)
  - Any VM can access its own GIC interface independently
  - A privilege execution level for VMs protects from system vs guest GIC registers access



Porting a hypervisor to the ARMv8-R Architecture

Porting a hypervisor

Configuring trust mode and entering the VM

- A VM has an associated level of trust

- The Cortex-R52 provides a 3-level privilege execution system:
  - EL0: user level, untrusted
  - EL1: OS level, include modes such as System and Supervisor
  - EL2: Hypervisor level, trusted

  - The hypervisor can also configure the Cortex-R52 for restricting EL1 access (i.e. GIC interface configuration)
  - The guest OS can configure its own MPU and start trusted or untrusted AUTOSAR partitions (i.e. trusted or untrusted AUTOSAR OS-Applications)

ETAS

VM Context Switch

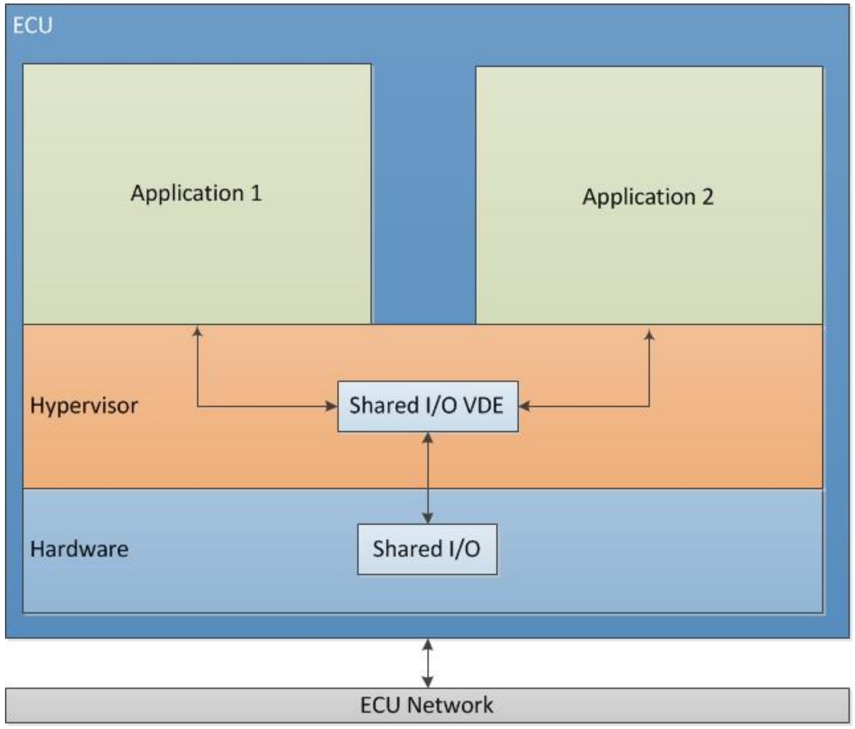
- This is the task of saving the current VM context onto the memory and restoring another one so that VMs can be switched at run time

- The Cortex-R52 provides a generic approach so the user can decide on the implementation strategy
- The hypervisor should make a copy of
  - CPU registers (status and GPRs)
  - GIC virtual interface registers
  - MPU configurations

- In case of sufficient number of MPU regions, an optimized approach could be to enable and disable only the required regions

Peripheral Virtualization

- A hypervisor can also provide a virtual device interface
- This is typically used for shared and emulated devices



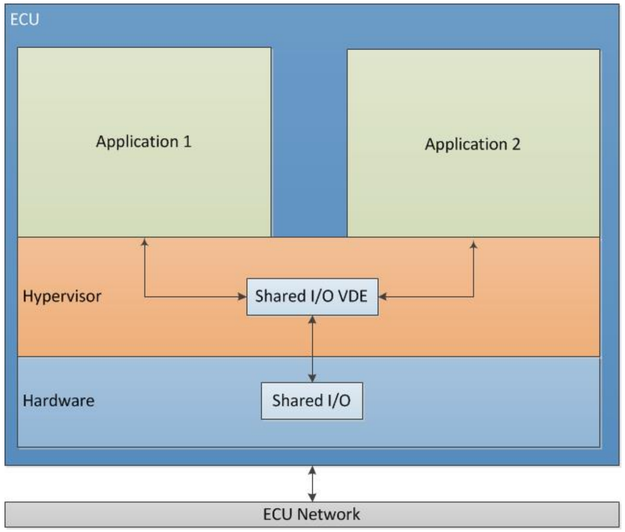
Porting a hypervisor to the ARMv8-R Architecture

Porting a hypervisor

Peripheral Virtualization

- The Cortex-R52 provides two options for implementing peripheral virtualisation
  - Using memory mapped registers; via the MPU it is possible to trigger events at EL2
  - Using the Hypervisor Service Call

- Also the MPU and GIC both provides relevant support for this purpose
  - The stage-2 MPU should allow only hypervisor access to the I/O
  - The GIC should route VDE interrupt to the hypervisor only. The hypervisor then can generate interrupts for the VMs as required



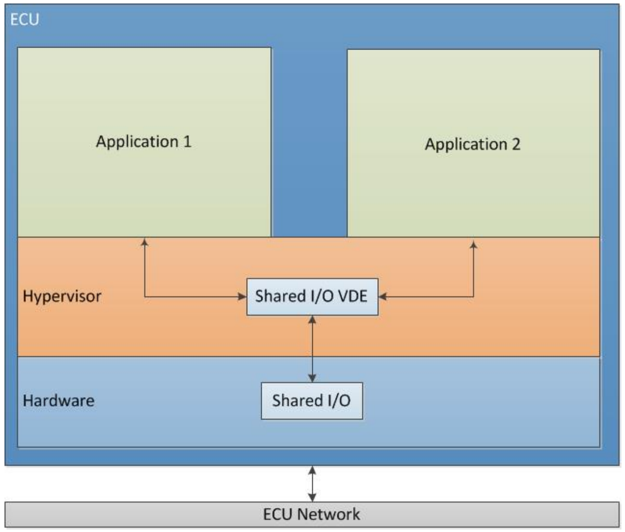
Porting a hypervisor to the ARMv8-R Architecture

Porting a hypervisor

Peripherals Virtualization

- In addition, the Cortex-R52 provides a virtual timer peripheral

- Allow the guest software to implement a private timer
- It could be used for implementing an internal time model (i.e. relative vs. absolute time model)

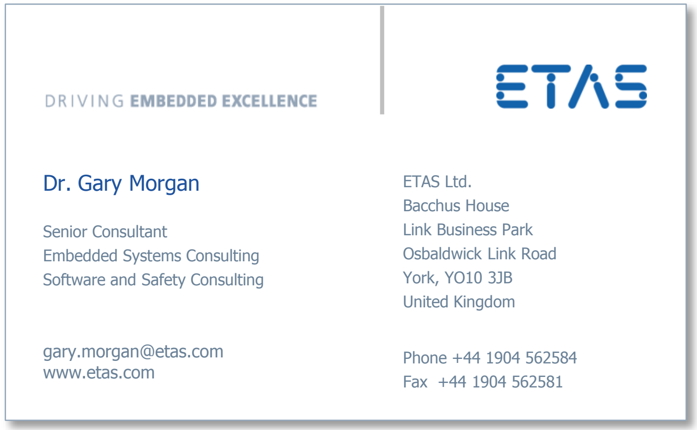


Conclusions

- The Cortex-R52 provides a comprehensive solution for supporting hypervisors with real time capabilities

- The Cortex-R52 s key features
  - Two stages MPU
    - Hypervisor MPU
    - Guest MPU
  - Three level execution privilege scheme
    - User level
    - Guest OS level
    - Hypervisor level
  - Interrupt controller virtual interface
    - Physical and virtual interrupt support

Porting a hypervisor to the ARM ARMv8-R Architecture
Contact details



DRIVING EMBEDDED EXCELLENCE

ETAS

Dr. Gary Morgan

Senior Consultant
Embedded Systems Consulting
Software and Safety Consulting

gary.morgan@etas.com
www.etas.com

ETAS Ltd.
Bacchus House
Link Business Park
Osbaldwick Link Road
York, YO10 3JB
United Kingdom

Phone +44 1904 562584
Fax +44 1904 562581

Thank you