Enabling Virtualisation on RISC-V Microcontrollers

Stefano Mercogliano
Daniele Ottaviano
prof. Alessandro Cilardo
Email:
stefano.mercogliano@unina.it,
daniele.ottaviano@unina.it,
acilardo@unina.it

May 31, 2023

DIETI. UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

Outline

DIETI UNINA

Virtualisation in Embedded Systems

- Nowadays industrial-related research is going towards the **consolidation of multiple real-time applications and OSes** on single boards.
  - *Automotive, Aerospace, Industrial, Defense, etc.*
- The goal is to optimize a set of parameters usually referred to as **SWaP-C**
  - **S**ize
  - **W**eight
  - **P**ower
  - **C**ost
- **Embedded Virtualisation** is the answer to the aforementioned problem. It is usually achieved by means of application processors and ad-hoc tailored hypervisors

Why virtualising microcontrollers?

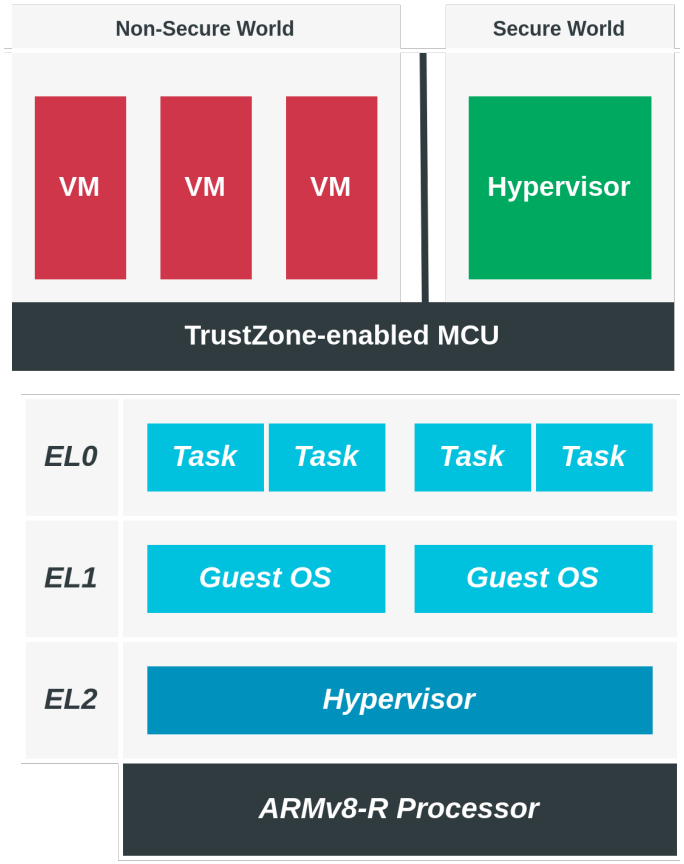
- Despite providing a powerful **virtualisation hardware support**, Application Processors imply non negligible downsides with regards to embedded virtualisation.
  - Increase in **Power Consumption**
  - Decrease in **Predictability**
  - Challenges in **Certification processes**
  - Inefficiency in **Resource Usage**
- Given the fact that microcontrollers are **cheap**, why caring about virtualization support when we can just buy more of them?
  - Advantages in **Scalability**
  - Ease for Software **Integration, Porting** and **Updating**
  - Benefits in terms of **Reliability & Availability**
  - Providing advantages in terms **Communication** for tightly coupled machines

MPU-based solutions

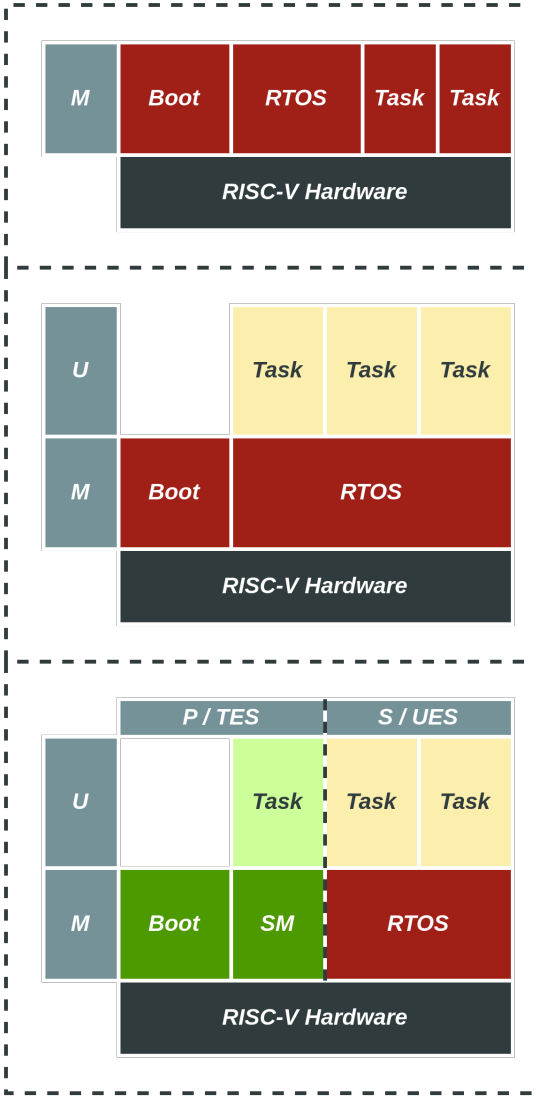
- ARM-based systems can rely on the **Memory Protection Unit** to provide full-virtualisation on microcontrollers. However many drawbacks are expected.
  1. Tradeoff between **performance and isolation**
  2. Inefficient **Trap Emulation**
  3. **No-overlapping** addresses
  4. **High overhead** context switch
  5. High I/O **Latency**

Can extra hardware do the trick?

- It is possible to adopt **Arm TrustZone-M** to improve MPU-based virtualisation
  - The Trusted World can provide better isolation
- Arm was the first to properly introduce a **hardware virtualisation extension for phyisical memory cores** in armv8-R specification
  - A new privilege level and a second MPU can solve most of the problems



What about RISC-V?



- RISC-V Microcontrollers can be protected by means of both firmware and hardware
  - Machine Mode can program the ePMP / PMP
  - M/U based microcontrollers cannot protect VMs
- Some proposals have been recently explored
  - They mostly focus on Security, on a "TrustZone-like" approach
  - Virtual Machines requires more than just isolation

A Summary

	ARM MPU	ARM TZ	ARM VM	RVM PMP	RVM VM
Performance & Isolation	X	V	V	X	?
Efficient Trap Handling	X	X	V	X	?
Overlapping Address	X	X	X	X	?
Low Overhead Context Switch	X	V	V	X	?
Low I/O Latency	X	X	V	X	?

Use Cases

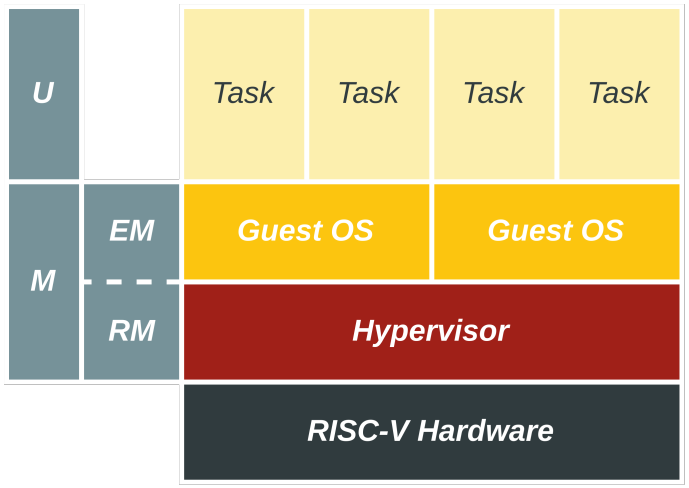
- We aim to design a hardware mechanism to efficiently support microcontrollers virtualisation on the RISC-V Archiectures
  - Both host and guests are supposed to be physical memory systems (M, M/U)
- Virtual Machines are supposed to host specific class of applications and workloads, most of them in the domain of safety and real-time (e.g. RTOSes, Baremetal)
  - This implies a focus on predictability, resource usage and memory management

Requirements

1. **Control**
   - *VMs shall handle a set of unique held resources and shared ones.*
2. **Equivalence**
   - *VMs code must not be modified*
3. **Performance**
   - *The architecture must provide native-comparable execution time.*
4. **Isolation**
   - *The architecture must support isolated environments called Virtual Machines (VMs).*
5. **Predictability**
   - *The design shall minimize non-determinism.*
6. **Memory Transparency**
   - *VMs shall run with their native memory layout*

Extending the privilege model

- Machine level is split in the **Emulated Machine mode (EM)** and the **Root Machine mode (RM)**
  - VMs run in EM-mode
  - The hypervisor lies in RM-mode
- Both EM and RM must have a partial/full set of control and status registers (CSRs)
  - RM-mode can access EM-mode CSRs.
  - EM-mode sees its CSRs as original M-mode CSRs
  - RM-mode adds new CSRs (e.g. rmemid).



Memory protection & relocation

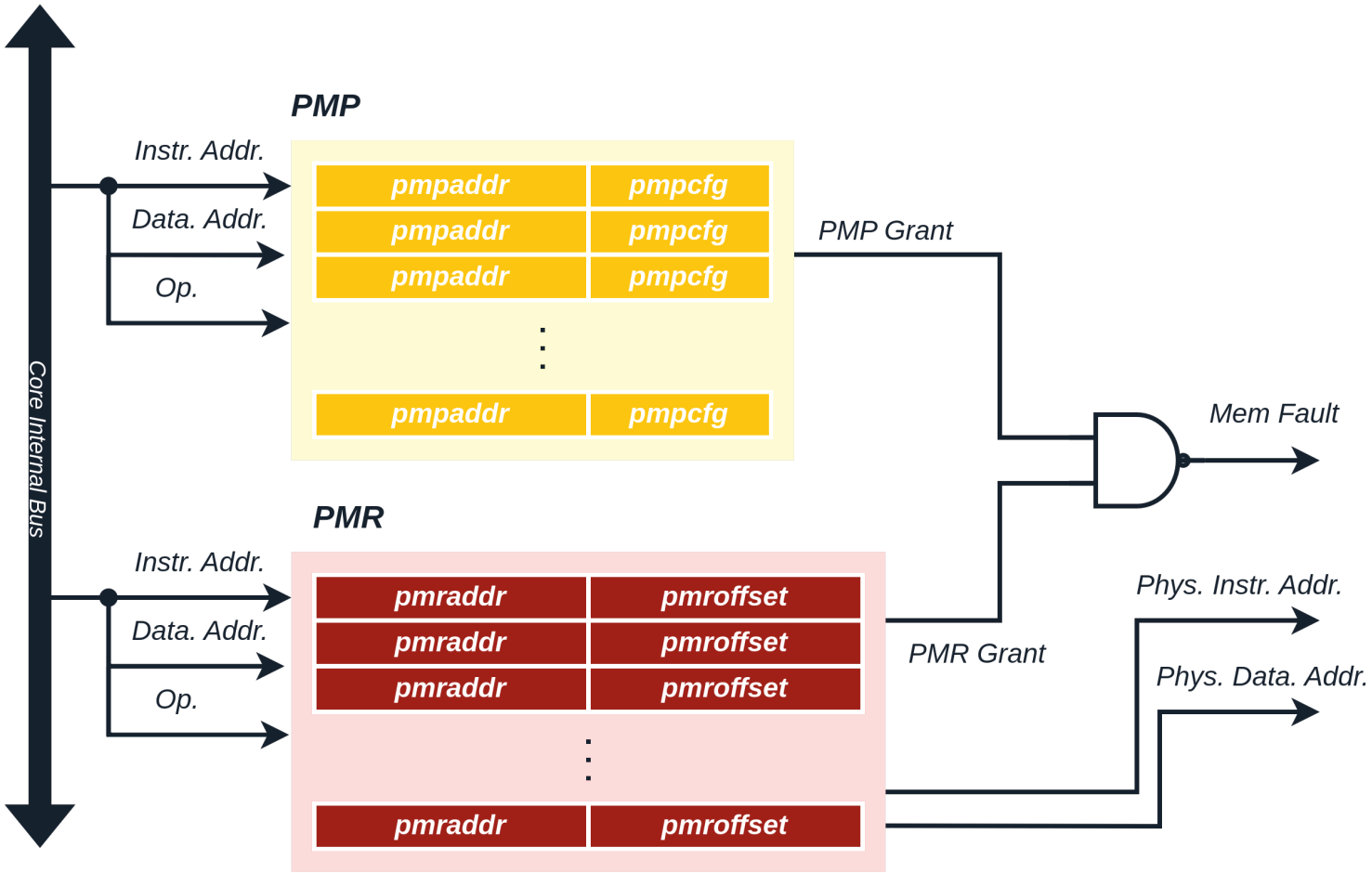
- VMs are compiled to run on a specific memory layout
- Relocating guest segments on the host memory provides both flexibility and VMs isolation
  - Relocation can be performed statically during linkage and/or modifying code
  - Can we provide a dynamic mechanism lighter than virtual memory?

  One level of Segmentation can be adopted instead of paging because of the nature of the workloads
  - Memory segments are variable in size and small in number
  - VMs segments are allocated once at deploy time and they won't be moved around / swapped
  - Guest memory is sparse

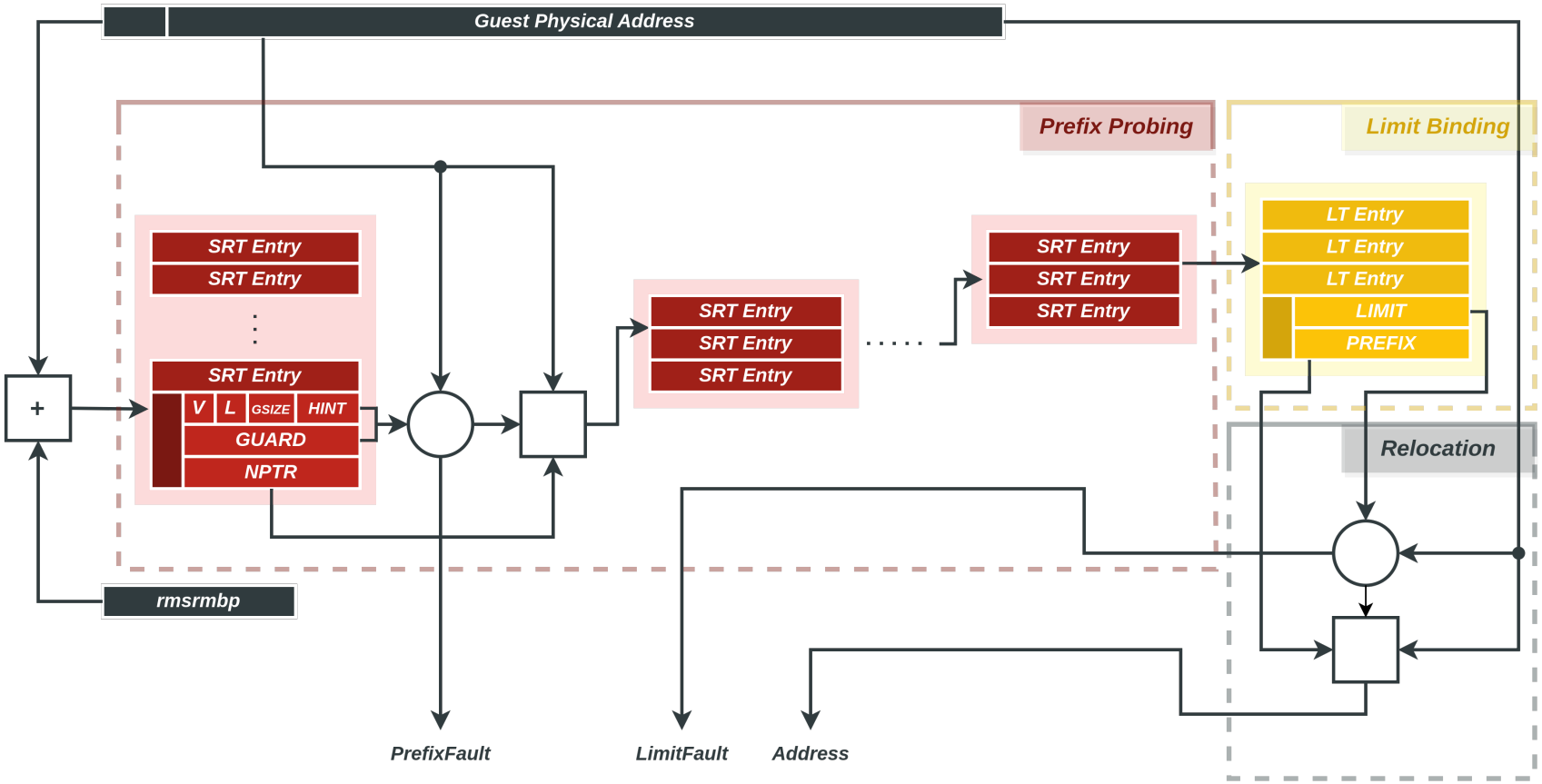
Physical Memory Relocation (PMR)

- The **Physical Memory Relocation** is an hardware unit responsible for providing guest protection and relocation on the host memory
  - It can be implemented as a simpler 2^{nd} stage PMP or as a Compressed Guarded Trie
  - Its impact on performance must be as minimal as possible
  - It can be turned off if a static approach is preferred

Table-based PMR



GRT-based PMR



Fine-grained resource partitioning

- Embedded virtualisation proved to be quite effective when using a partitioning-based software approach
  - Interference is minimized
  - At the cost of resource usage efficiency
- A **resource partitioning mechanism** can be used to provide a fine-grain partitioning of host resources (e.g. Regfile, PMP, PMR, etc.)
  - It minimizes interferences providing a better resource usage
  - As long as guests can run with a subset of host physical resources

Comparisons

	ARM MPU	ARM TZ	ARM VM	RVM PMP	RVM VM
Performance & Isolation	X	V	V	X	V
Efficient Trap Handling	X	X	V	X	V
Overlapping Address	X	X	X	X	V
Low Overhead Context Switch	X	V	V	X	V
Low I/O Latency	X	X	V	X	V

Still much to do

- Fast context switch
  - The partitioning mechanism can reduce the context size
  - However, can a new instruction help?
- IO and Interrupts are to be further explored
  - Interrupt controllers could be modified (CLIC, PLIC)
  - Shared memories could be used to abstract communication channels
  - What happens with the IOPMP? Should we have an IOPMR?
- Caching mechanisms are still a todo
  - Cache coloring techniques should be taken into account
- A first PoC
  - The Ibex core from LowRISC is our first candidate

Enabling Virtualisation on RISC-V Microcontrollers

Stefano Mercogliano
Daniele Ottaviano
prof. Alessandro Cilardo
Email:
stefano.mercogliano@unina.it,
daniele.ottaviano@unina.it,
acilardo@unina.it

May 31, 2023

DIETI. UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE