Microkernel Dedicated for Dynamic Partial Reconfiguration on ARM-FPGA Platform

XIA Tian, Jean-Christophe Prévotet and Fabienne Nouvel

INSA, IETR, UMR 6164, F-35708 RENNES

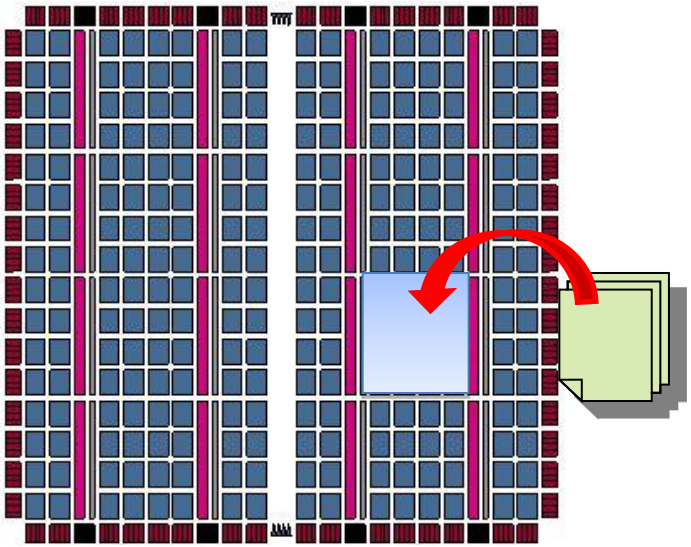
EWiLi'14, Lisboa, Portugal
2014-11-13

Outline

- Motivation and Object
- Overview of Proposed Platform
- Microkernel Architecture
- Case study and Analysis

Motivation and Object

DPR: Dynamic Partial Reconfiguration



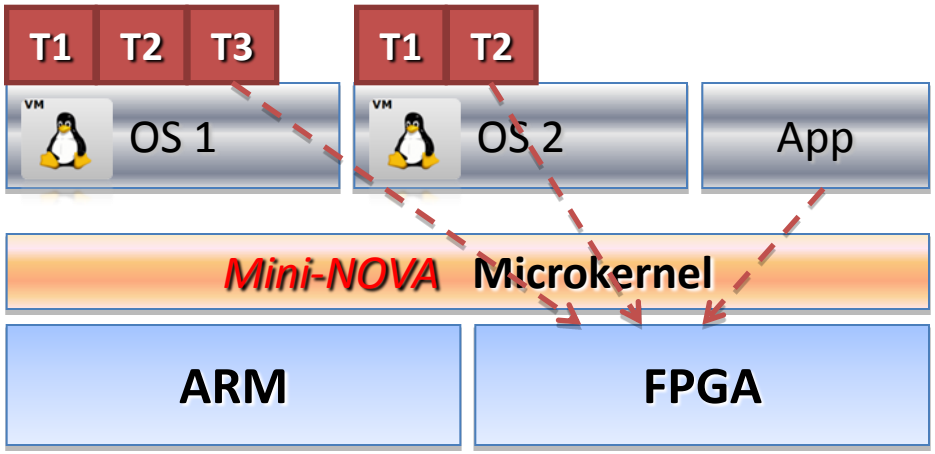
- ❑ Reduced hardware resource utilization
- ❑ Reduced reconfiguration latency
- ❑ Improved design efficiency

Supporting technologies

- Bare-metal application
- Existing OS extention (Linux): RAMPSoCVM
- Embedded OS: CAP-OS, ReconOS

Motivation and Object

Object: Microkernel-supported ARM-FPGA platform

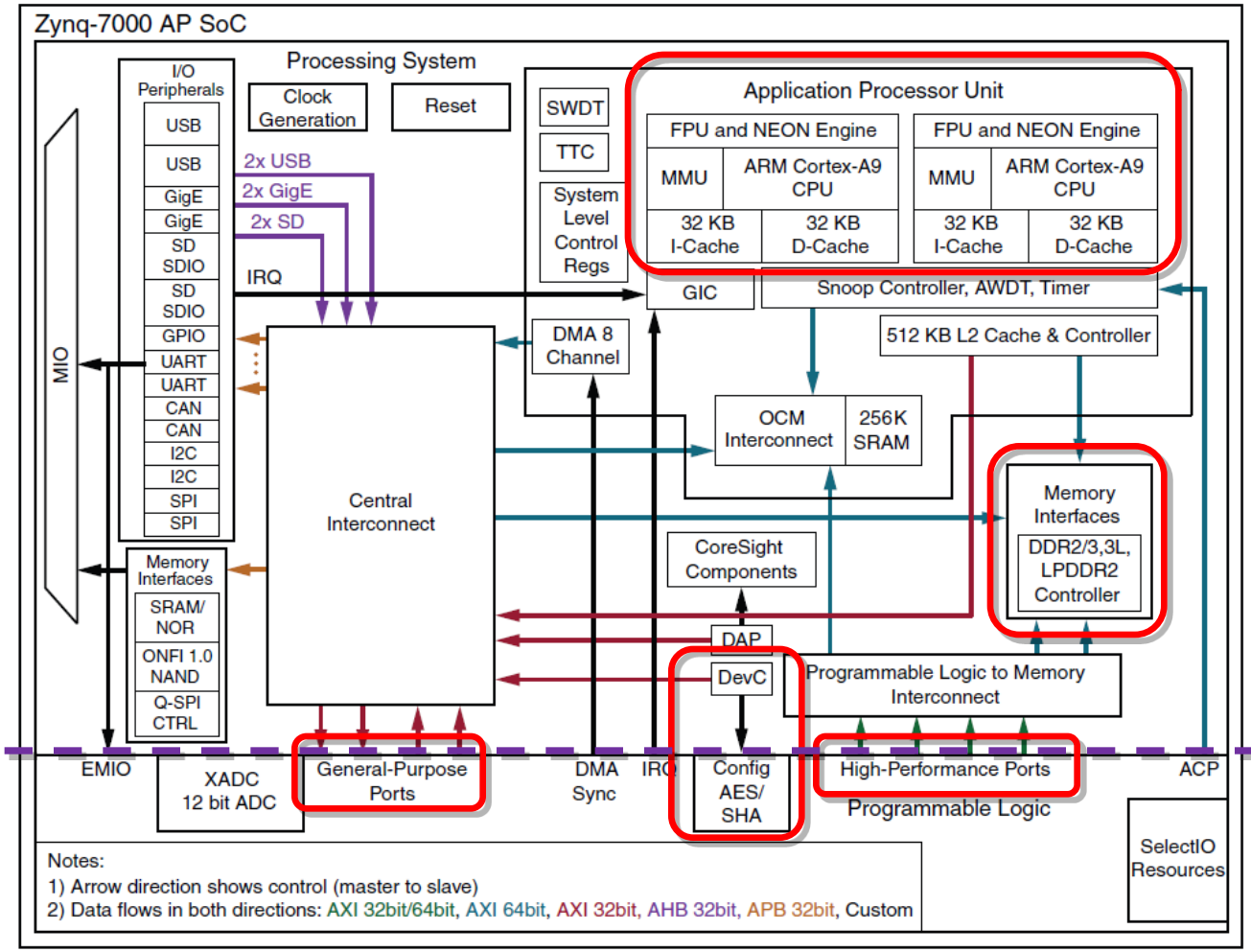


Advantages:

- Higher security level: completely isolated environment
- Smaller trust computing base (TCB) than traditonal OS
- Mixed criticals: hard/soft/non real-time applications

Proposed Platform

Zynq-7000 All programmable SoC



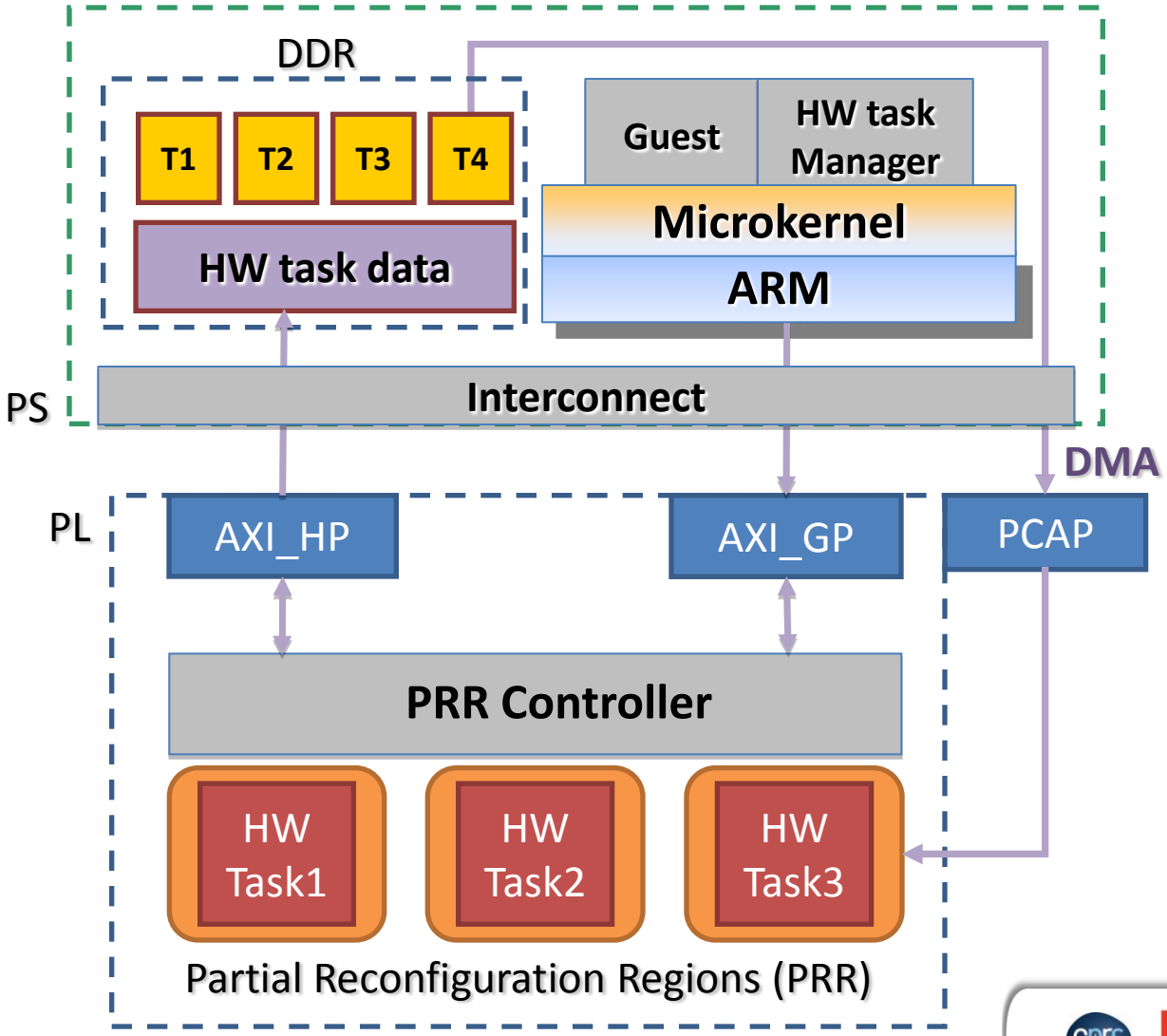
- Dual core ARM Cortex-A9 (Single processor currently)
- 512MB DDR
- AXI bus based PS/PL Interface
- Processor Config Access Port (PCAP)

PS (Processing System)

PL (Programmable Logic)

Proposed Platform

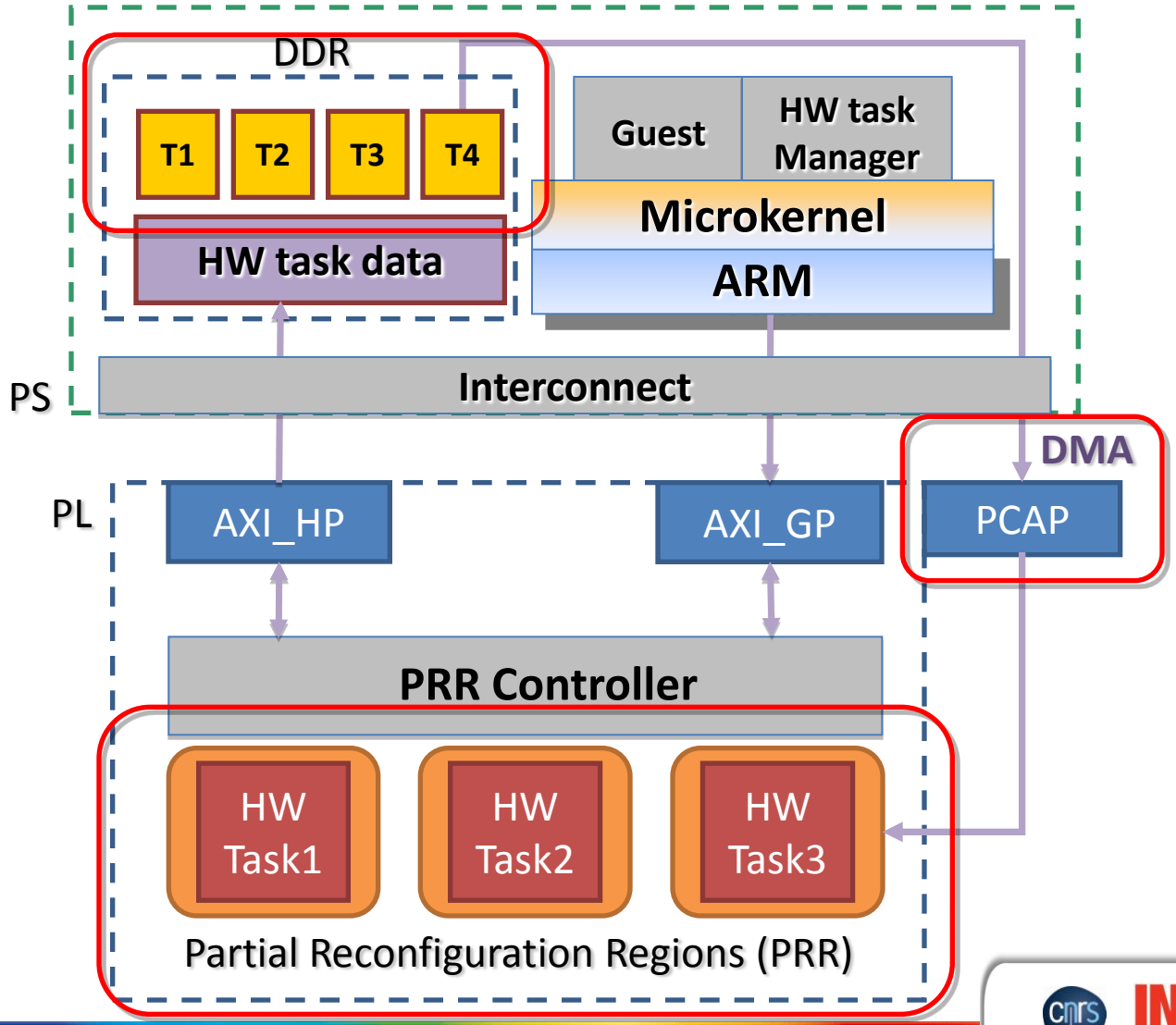
Proposed Architecture



- Hardware tasks
- Communication Interface
- PRR controller

Proposed Platform

Proposed Architecture

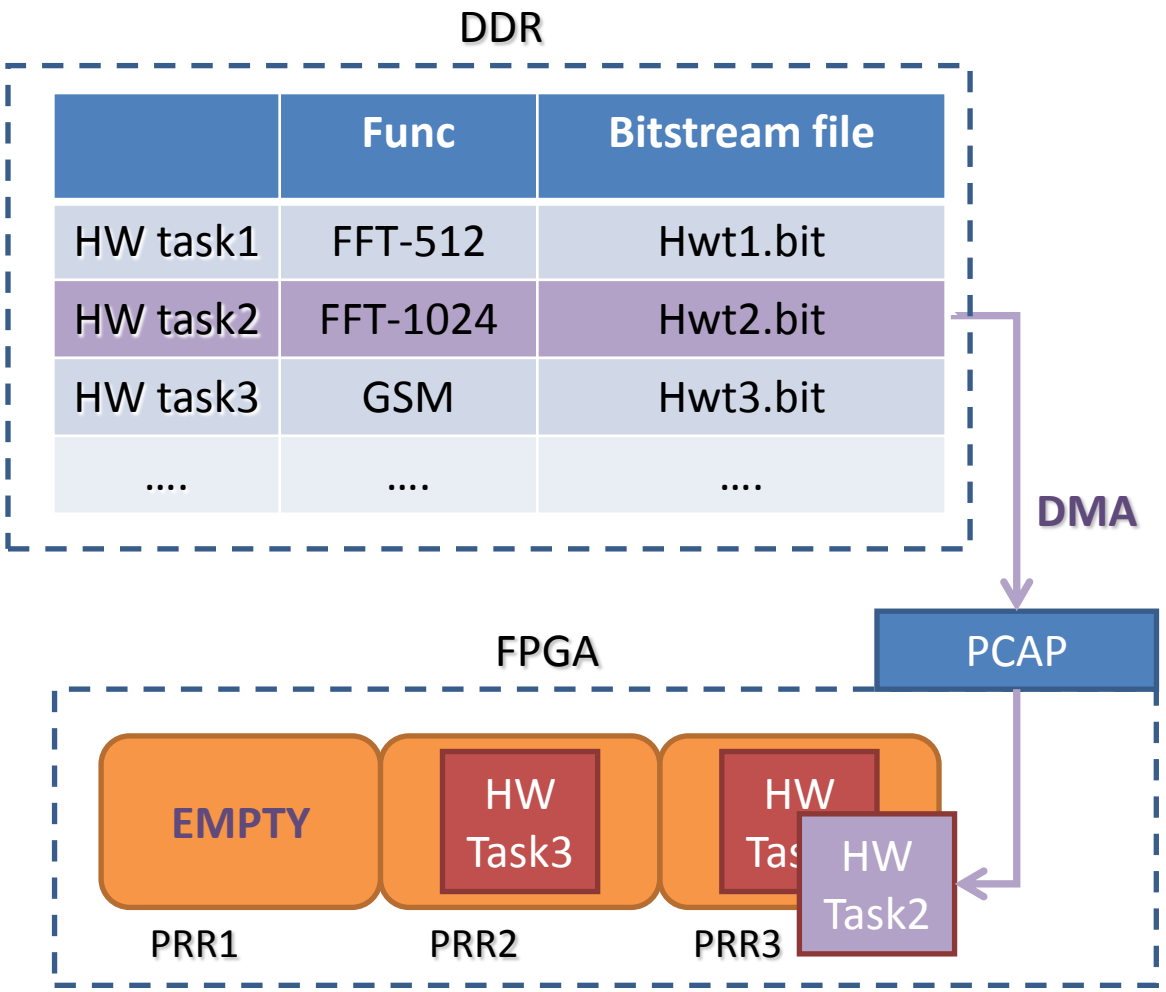


- Hardware tasks
- Communication Interface
- PRR controller

Proposed Platform

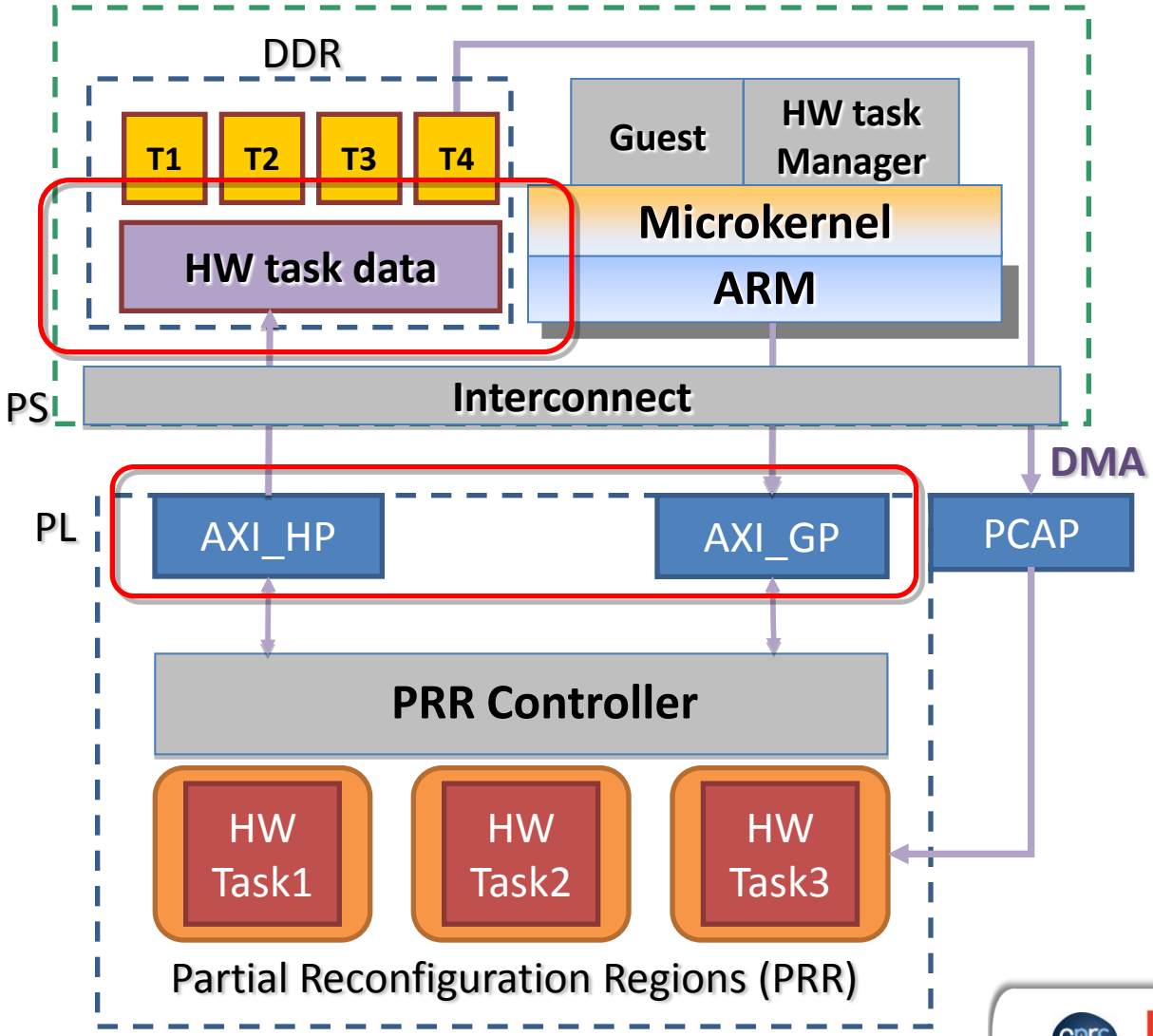
Hardware tasks

- Container: PRR (Partial Reconfig Region)
- Stored by DDR Memory
- Download by DMA transfer through PCAP
- Reconfig. overhead is linearly correlated to PRR size, which means predictable latency of hw task switch.



Proposed Platform

Proposed Architecture



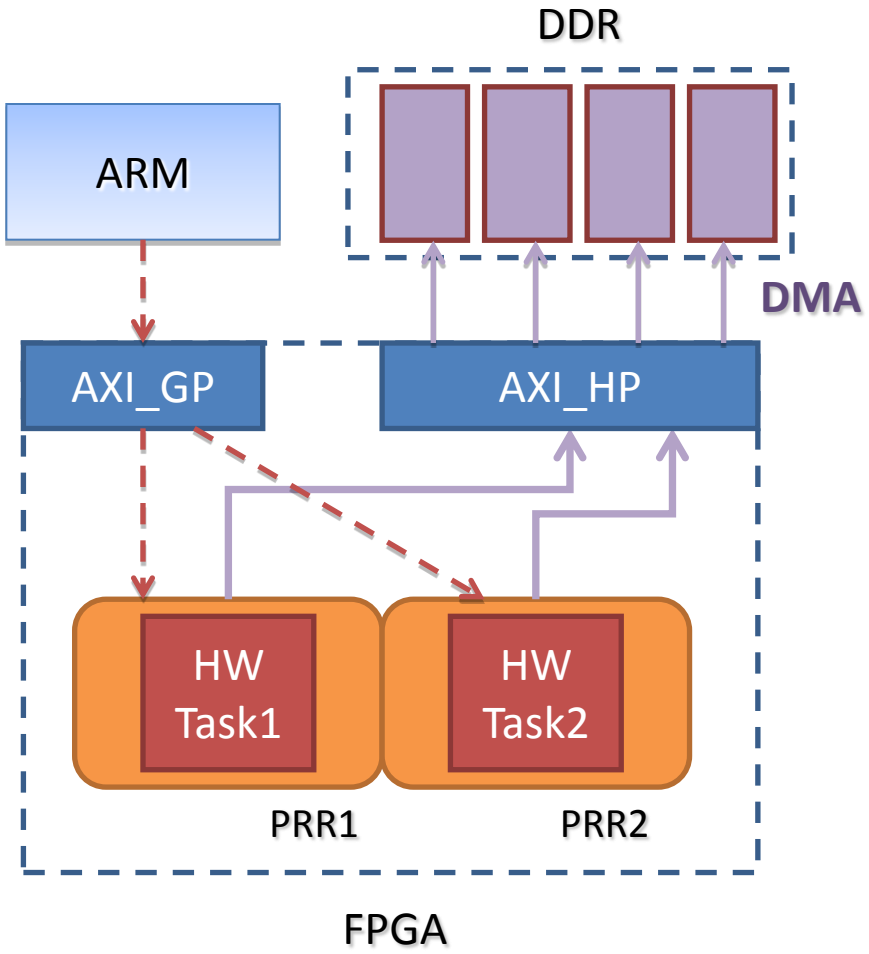
- Hardware tasks
- Communication Interface
- PRR controller

Proposed Platform

Communication Interface

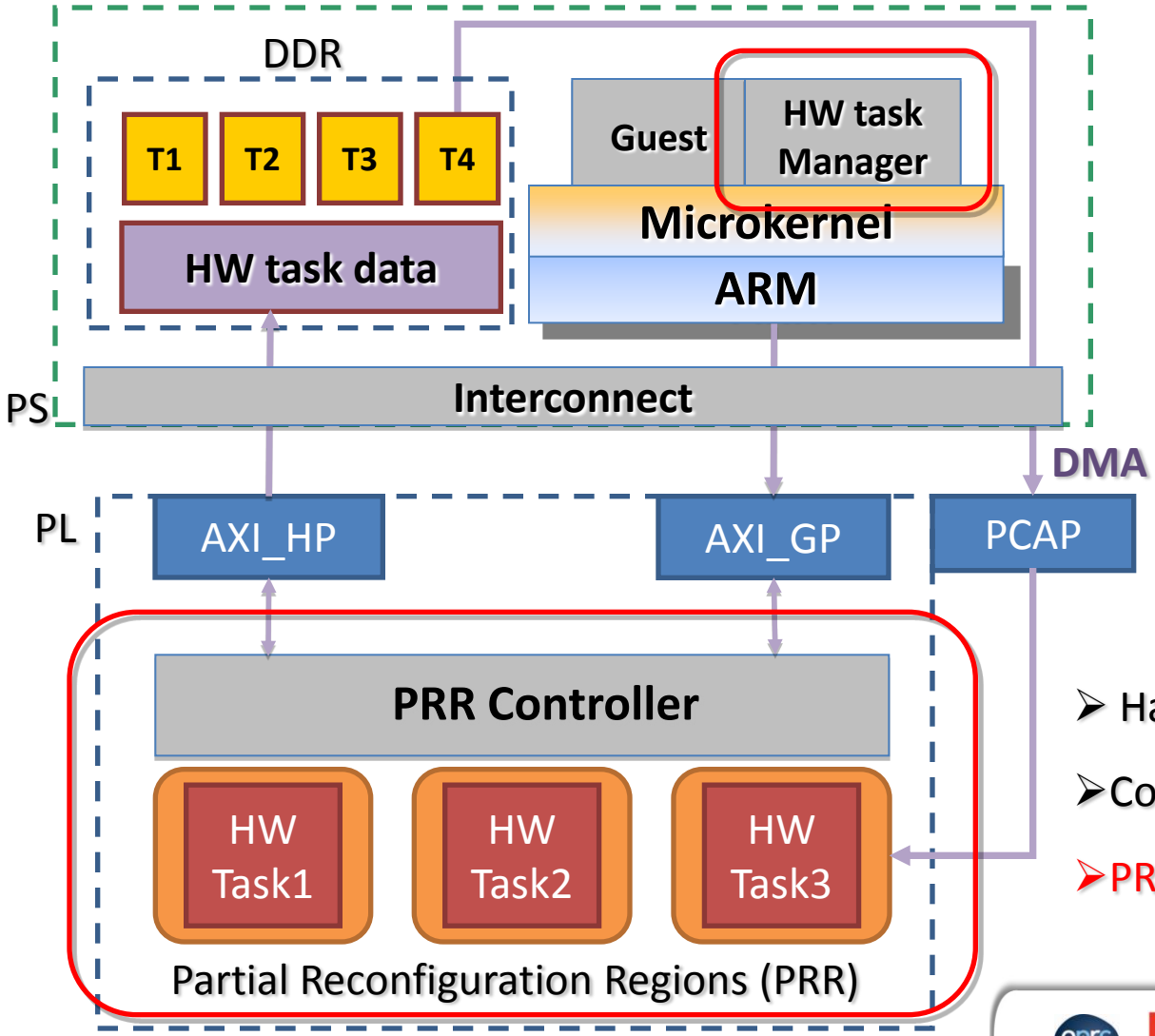
	AXI_GP	AXI_HP
Number	2	4
Mode	Master	Slave
Access	Unified Addr space	DMA
Throughput	600MB/s	1200MB/s

- AXI_GP: Control HW task behaviors
- AXI_HP: High speed data exchange



Proposed Platform

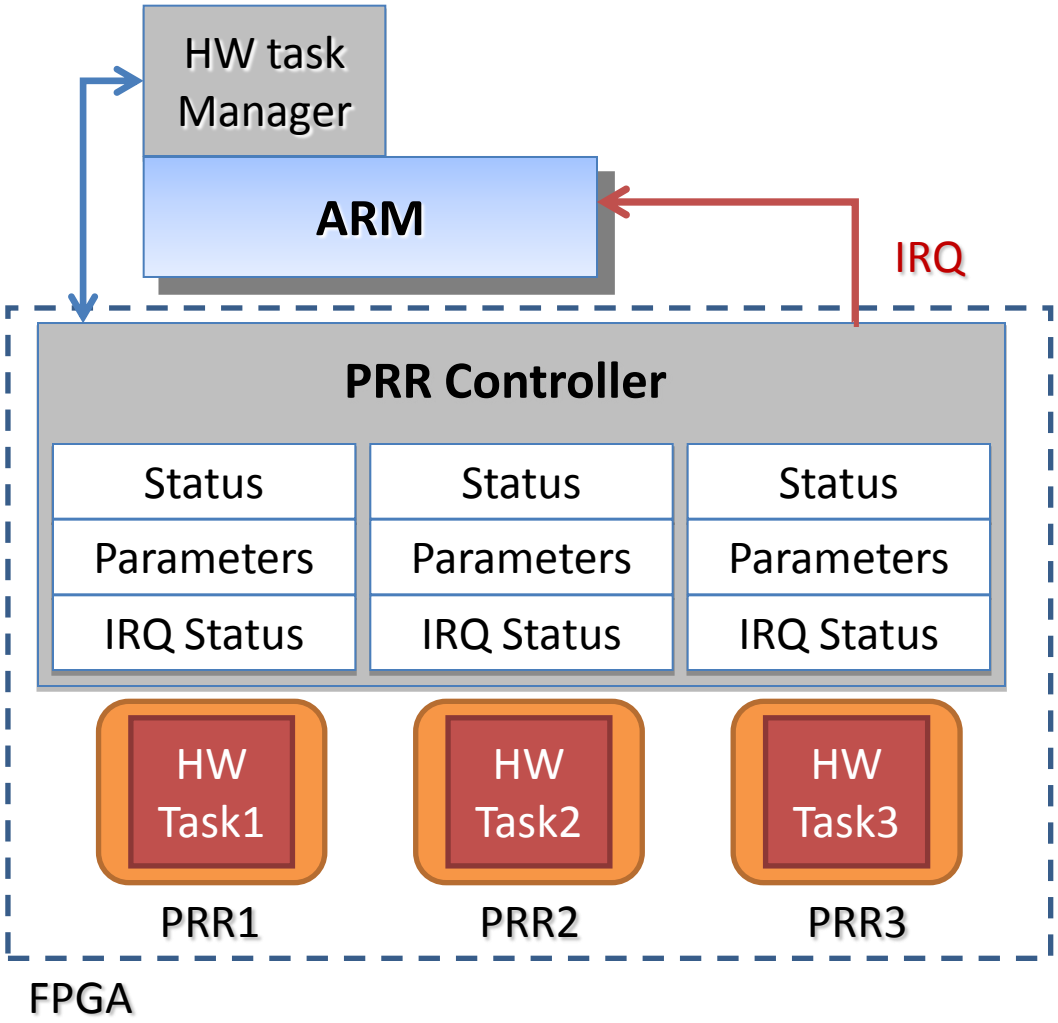
Proposed Architecture



- Hardware tasks
- Communication Interface
- PRR controller & synchronization

Proposed Platform

PRR Controller Structure

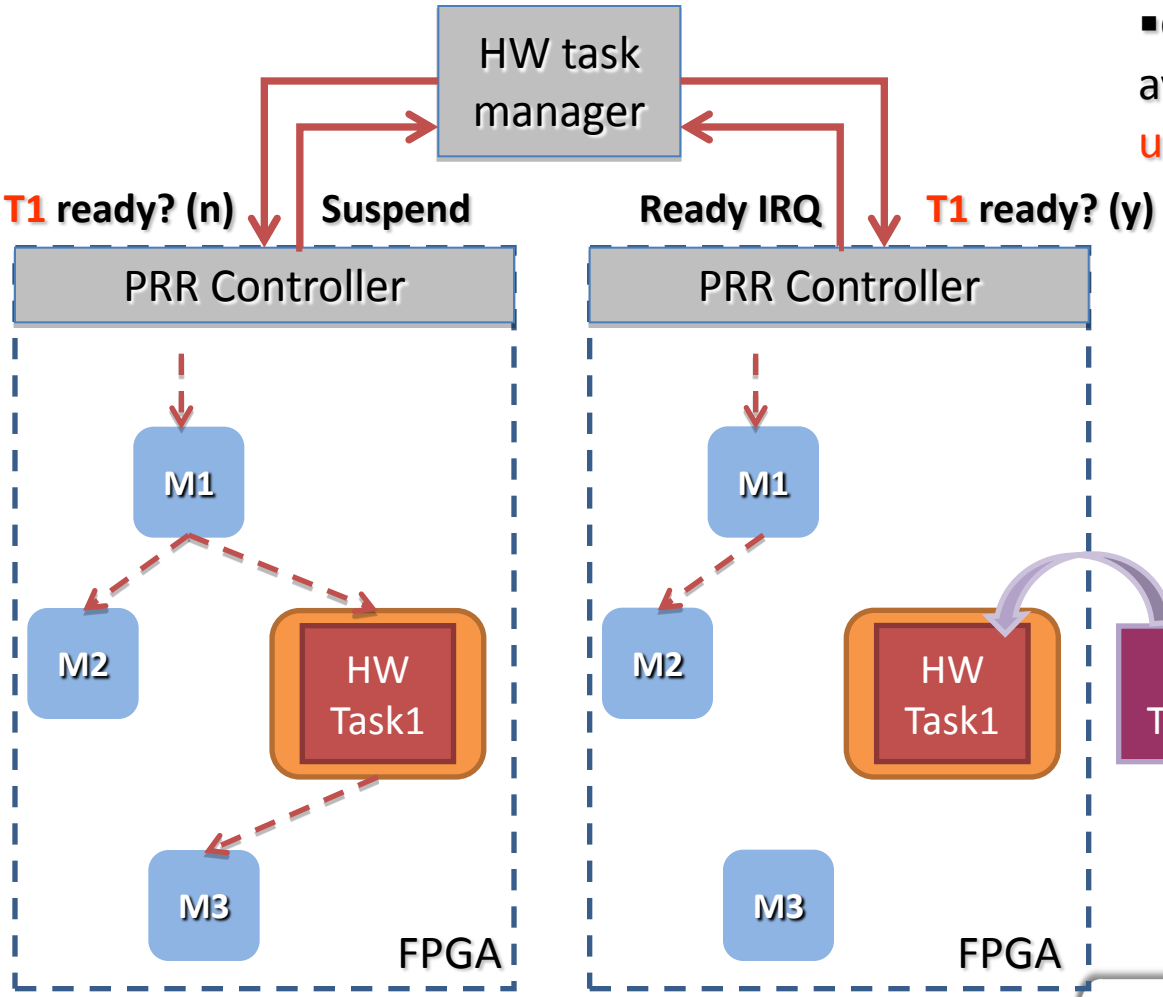


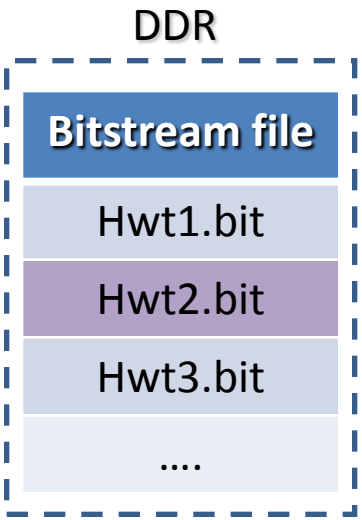
- Cooperate with HW task manager
- Configue HW task parameters:
  - DMA address, data size
  - Working mode
- Generate HW task Synchronization IRQ
- Monitor HW switch

Proposed Platform

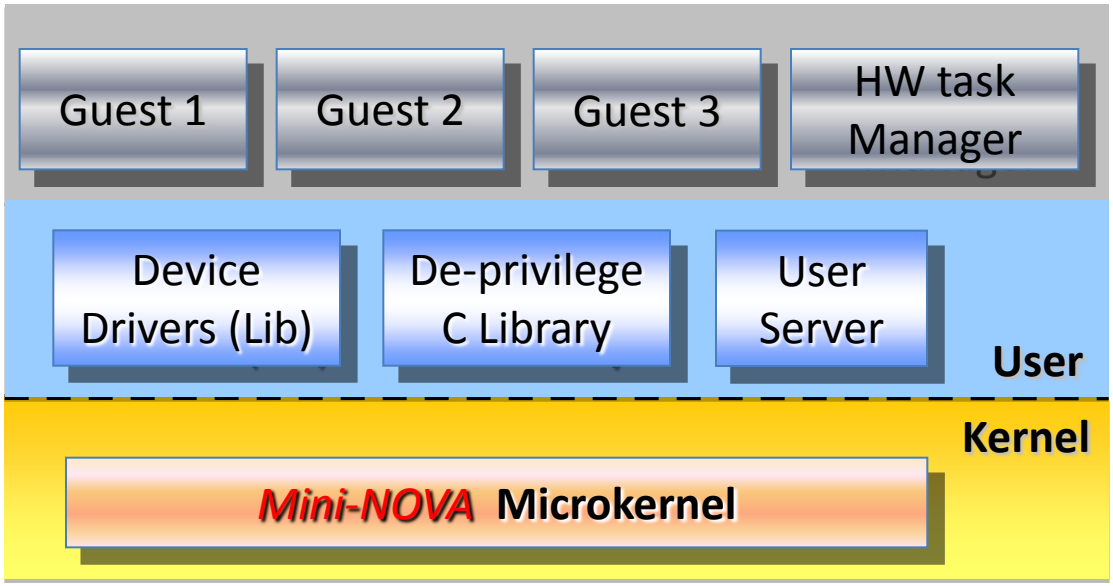
PRR Controller Monitor

- Cooperate with HW task service to Monitor HW switch.
- Guarantee the HW task security, avoiding invalid data output and undesired task state.





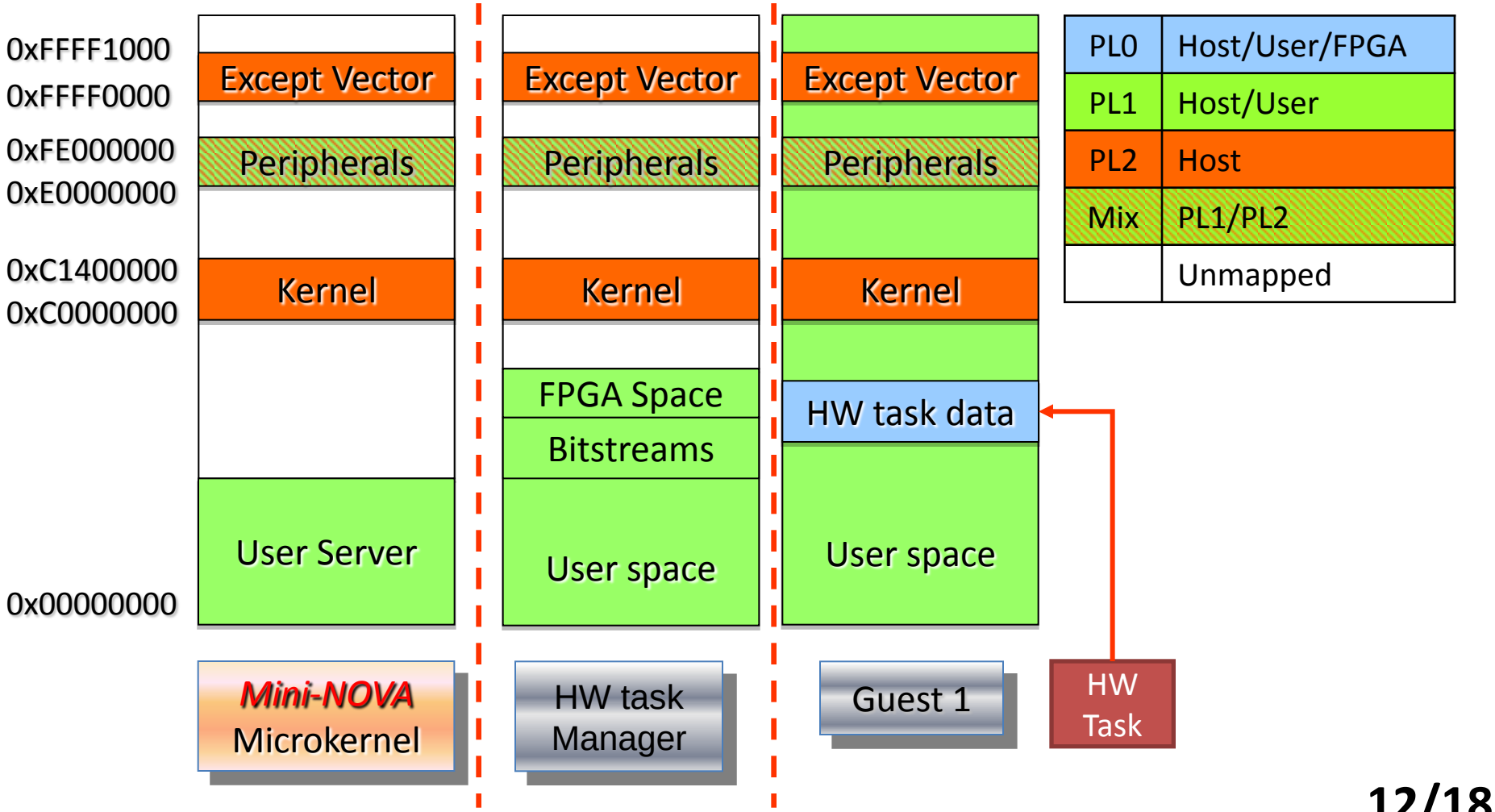
Microkernel Architecture



- Small TCB Size (Kernel: 8 KLOC; User environment: 3 KLOC)
- De-privileged C Library to handle privileged operations: Cache, page table
- Bootloader for guest OS/Applications (User Server)
- Separate virtual address spaces for kernel and guests
- Specific Priority-based round-robin scheduling

Microkernel Architecture

Isolated virtual address spaces

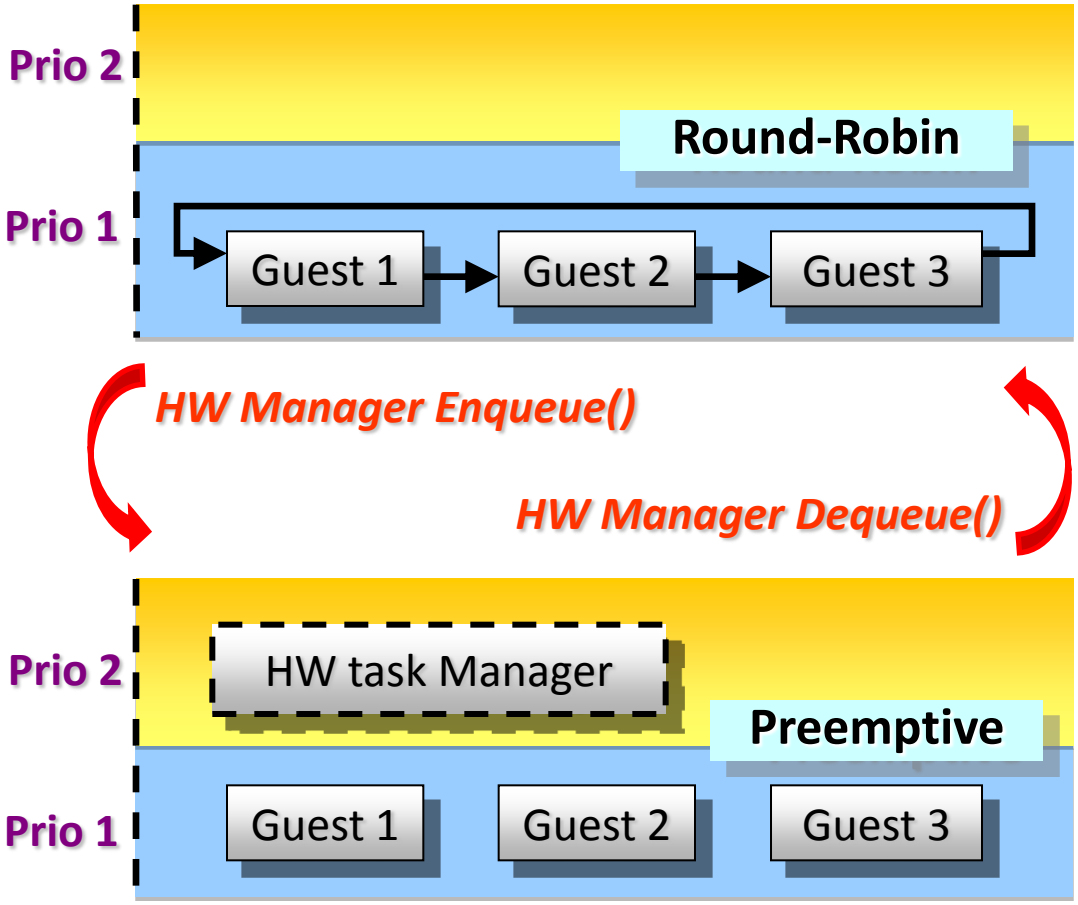


Mini-NOVA Privilege Level (PL)

PL0	Host/User/FPGA
PL1	Host/User
PL2	Host
Mix	PL1/PL2
	Unmapped

Microkernel Architecture

Priority-based scheduling



Scheduling Principle:

- HW task requires tighter time constrain (hard real-time)
- Quick response for the HW task management should be guaranteed

Functions:

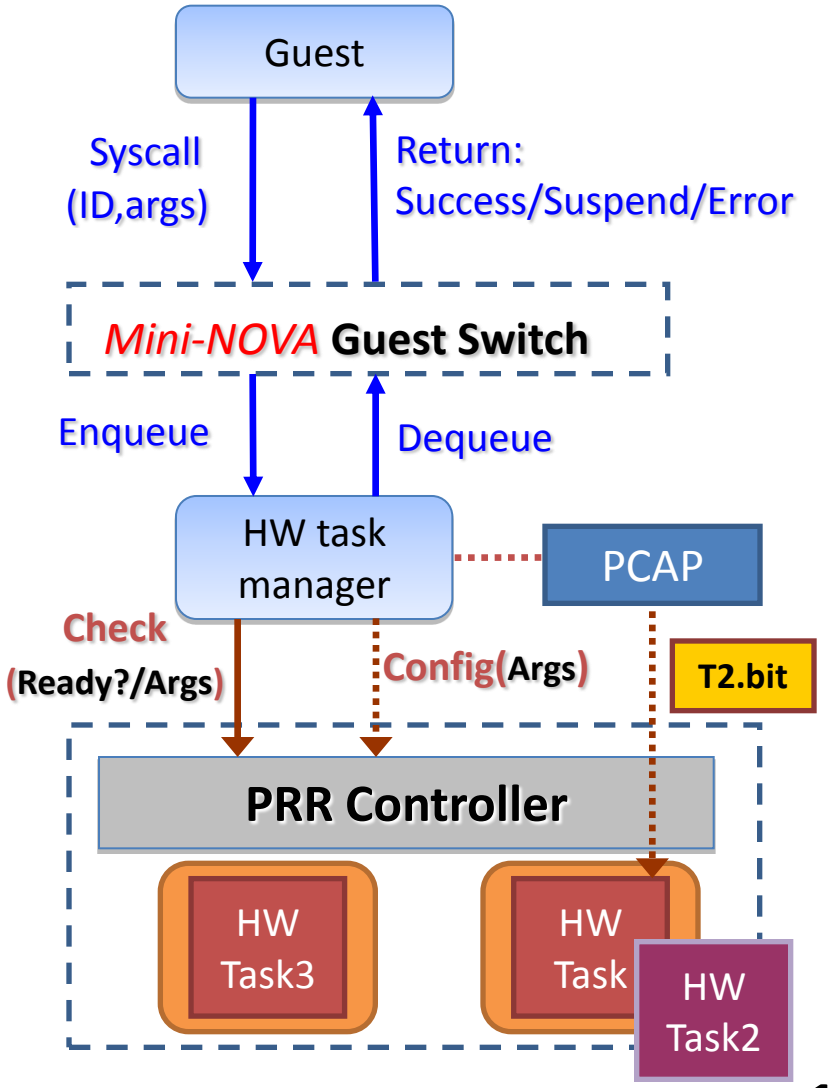
- *HW Manager Enqueue()*: Add the HW task manager into the run queue and preempt lower prio
- *HW Manager Dequeue()*: Remove the EC of the HW task manager from the run queue
- *reschedule()*: Update the schedule and dispatch the highest priority EC

Microkernel Architecture

HW Task Manager

- HW tasks' switch/config is isolated from other guests, should be done by the HW task manager.
- This mechanism is to ensure the security of the FPGA fabric.
- Process flow:
  1) Guest's System call (hw task id, args)
  2) Check HW task ready/args
  3) Switch/Configure hw task
  4) Return to guest

Syscall_HW_Manager
(HW task id, arg01, arg02, arg03)

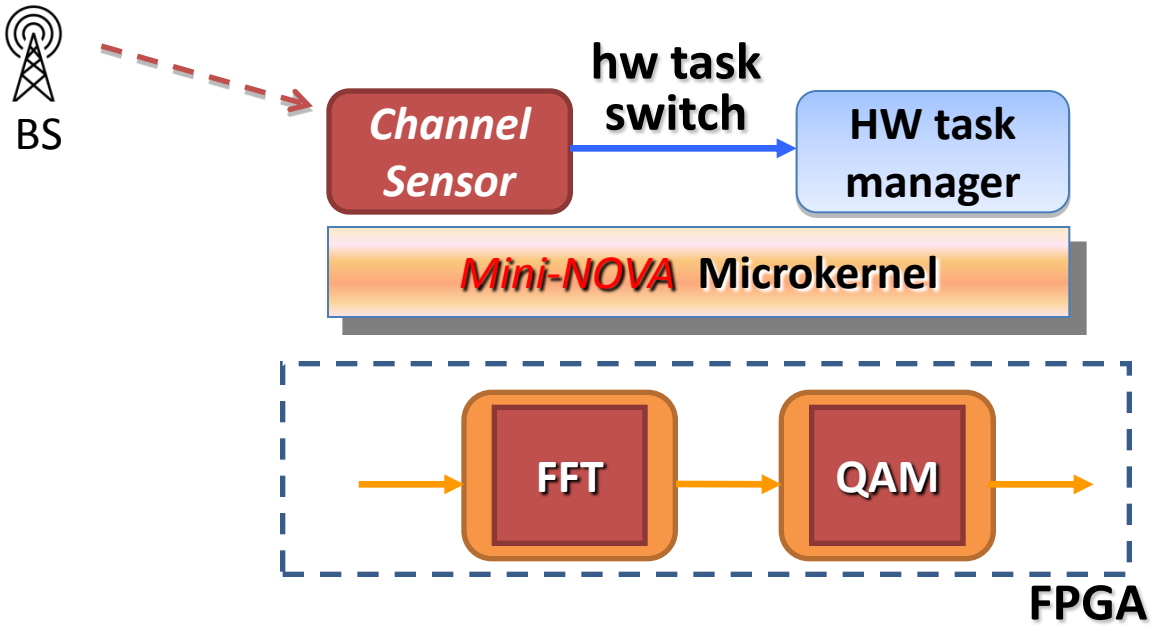


Case study and analysis

Case Description

Proposed scenario:

Computing system within a mobile wireless terminal, which is capable of dynamically change its configuration in order to obtain the best level of performances according to the channel conditions.

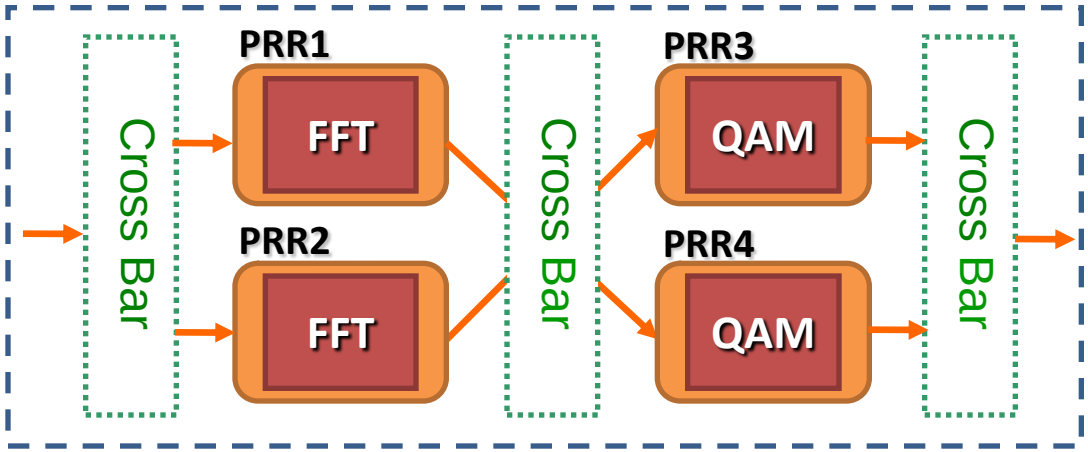


DDR

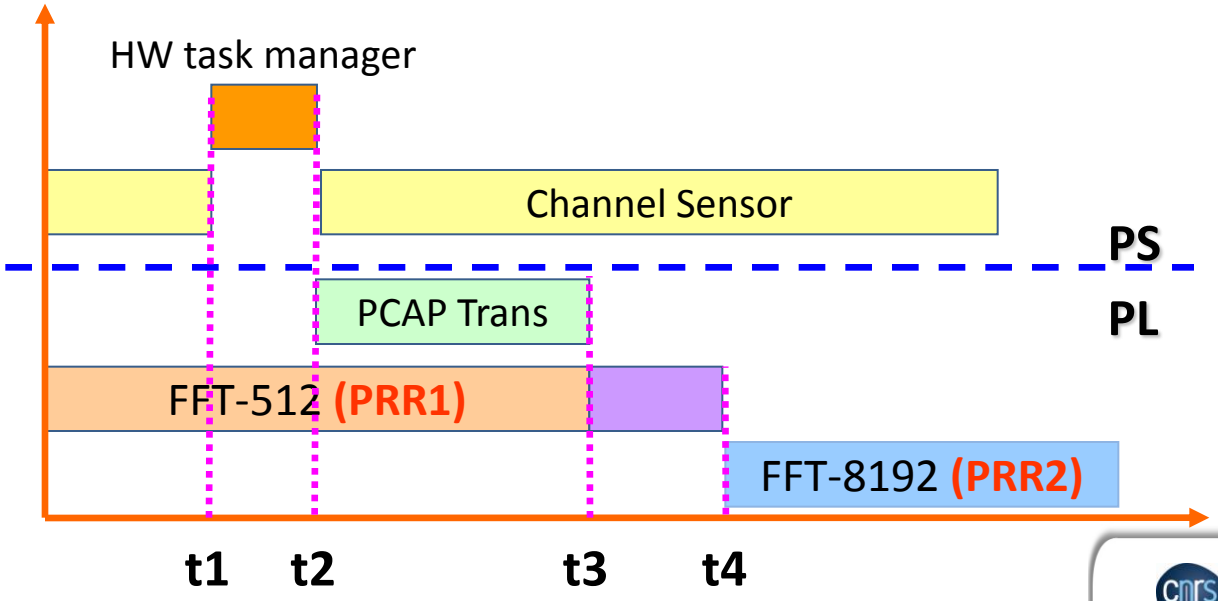
Bitstream file	
QAM4	FFT 256
QAM16	...
QAM64	FFT 8192

Case study and analysis

Implementation



- Modulation and IFFT execute in pipeline, the reconfiguration will casue a suspension.
- To minimize the significant time overhead we proposed a multiple-path structure.



t1: Syscall HW Manager;

t2: PCAP Start,
HW Manager dequeue;

t3: PCAP Done;

t4: Data frame over.

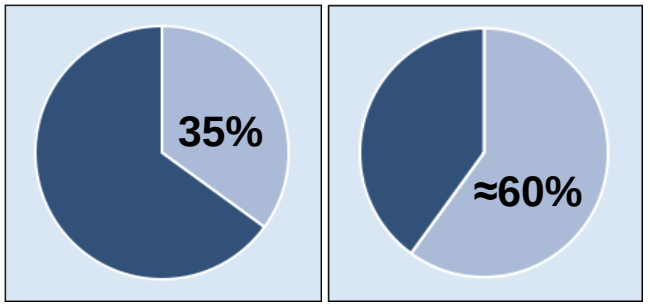
Pipeline suspension

Case study and analysis

Results & Analysis

Task name	Type	Execution Time(ms)	Reconfig.Time(ms)	Resource Usage
ChannelSensor	SW	3	no	no
HW Manager	SW	0,0096	no	no
Guest Switch	SW	0,00232	no	no
QAM (4/16/64)	HW	0,09-0.03(1 frame)	0.231	2%
IFFT (256-8192)	HW	0,006-0,168(1 frame)	2.72	13%

- Frame size: 18,800 bits
- FPGA Freq: 100MHz
- ARM Freq: 660MHz



FPGA Utilization rate with / w/o DPR

Reconfig Overhead:

Pipeline suspension: 0.168 ms (8096 points FFT).

Conclusions

- An ARM based microkernel is bulit on Zynq-7000 architecture.
- Propose hardware task manager and PRR/PRR Controller to support DPR.
- Apply seperate memory space and multiple access privileges to improve the system security, especially for FPGA access.
- Use priority-based round-robin scheduling to guarentee run-time hw task management.
- Perspectives:
  - Further virtualization with Linux and other RTOS;
  - Performance evaluation with standard benchmarks.

Thank you for your attention!

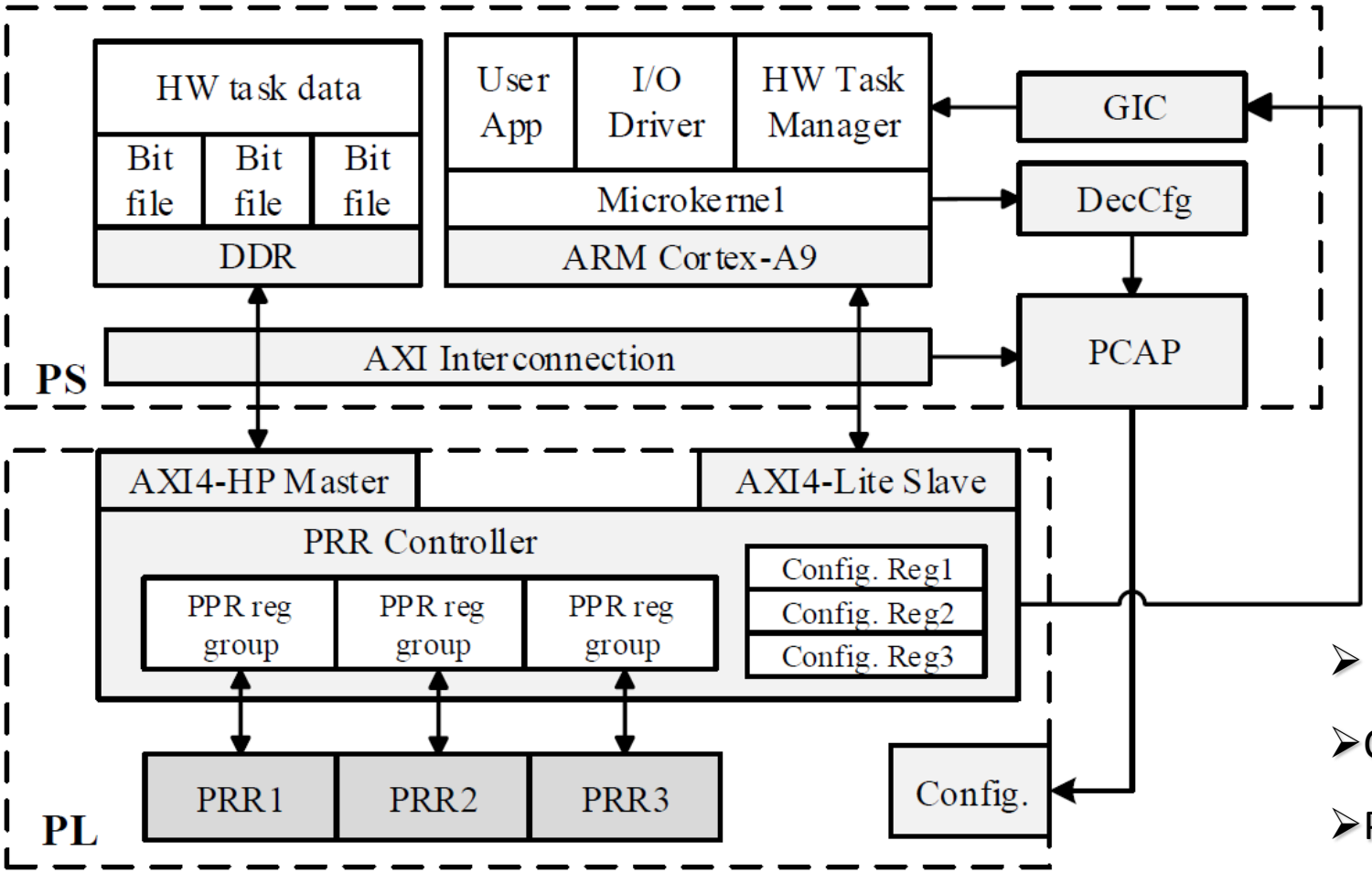
XIA Tian

INSA, IETR, France

EWiLi'14, Lisboa, Portugal
2014-11-13

Proposed Platform

Proposed Architecture

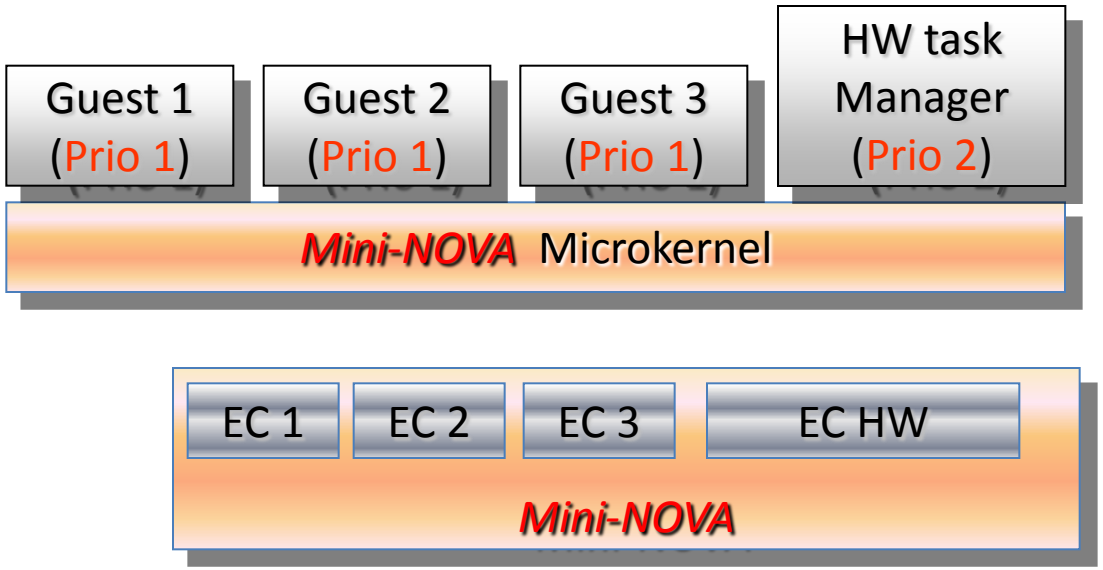


- Hardware tasks
- Communication
- PRR controller

❖PRR: Partial Reconfiguration Region

Microkernel Architecture

Priority-based scheduling



Execution Context (EC):

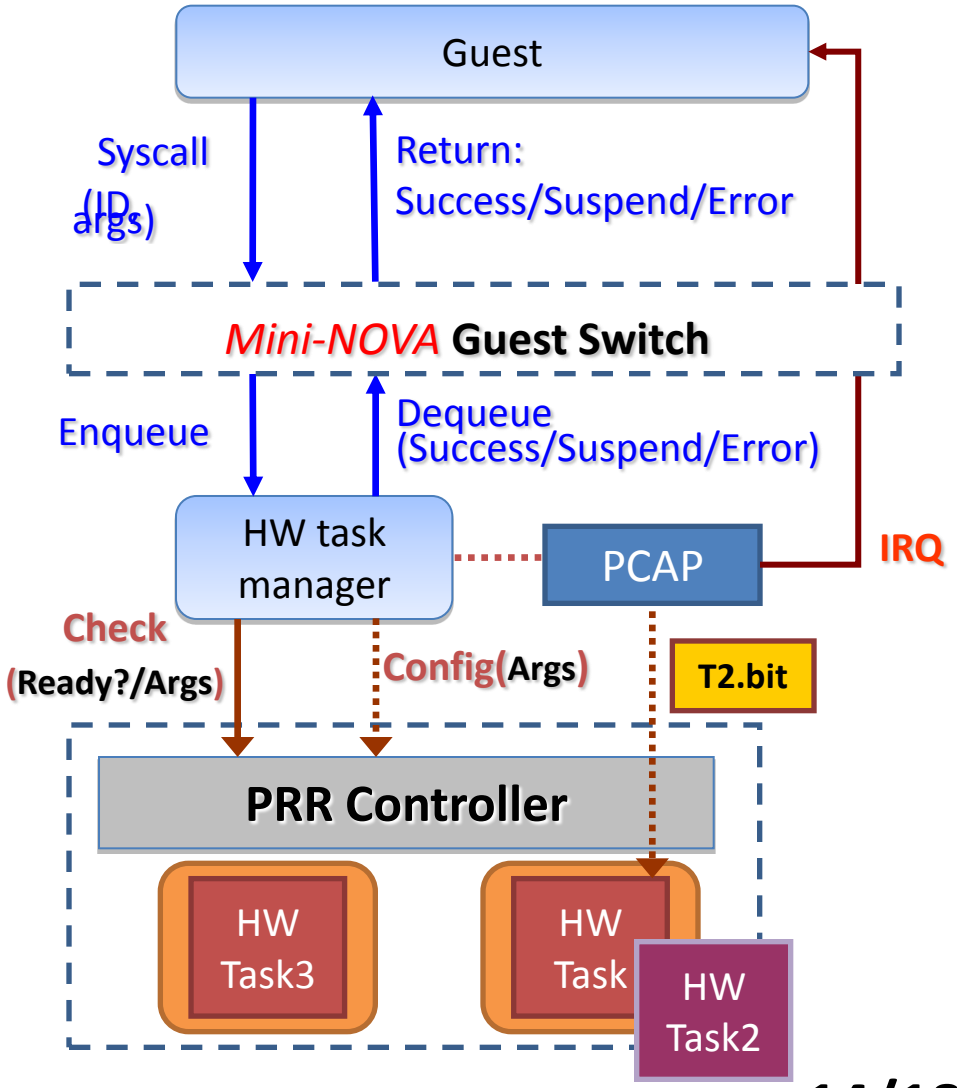
- CPU/FPU register state
- Coprocessor state
- Page Table Address (TTBR)
- Scheduling Priority

Microkernel Architecture

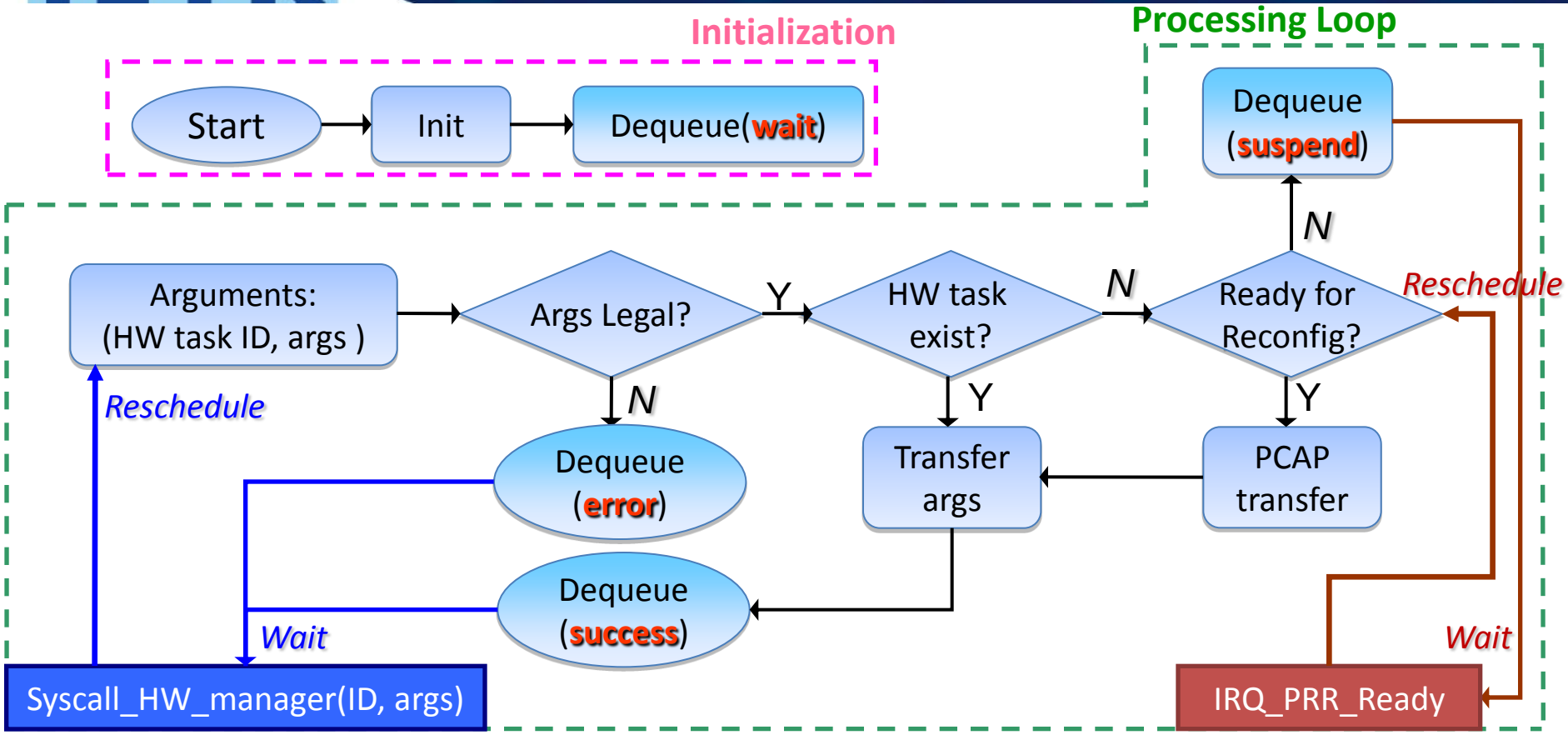
HW Task Manager

- HW tasks' switch/config is isolated from other guests, should be done by the HW task manager.
- This mechanism is to ensure the security of the FPGA fabric.
- Process flow:
  1) Guest's System call (hw task id, args)
  2) Check HW task ready/args
  3) Switch/Configure hw task
  4) Return to guest

Syscall_HW_Manager
(HW task id, IRQ_en, arg01, arg02, arg03)



Microkernel Architecture



Syscall_HW_Manager (HW task id, arg01, arg02, arg03)