INSTITUTE
OF COMMUNICATION,
INFORMATION
AND PERCEPTION
TECHNOLOGIES

Scuola Superiore
Sant'Anna

Retis
Real-Time Systems Laboratory

FRED: A Framework for Supporting Real-Time Applications on Dynamic Reconfigurable FPGAs

Marco Pagani, Alessandro Biondi, Mauro Marinoni, and Giorgio Buttazzo

ReTiS Lab, TeCIP Institute
Scuola superiore Sant'Anna - Pisa

Agenda

1 **Dynamically Reconfigurable FPGAs**

Modern **heterogeneous** platforms open a **new scheduling dimension**

2 **The FRED Framework**

Predictable FPGA virtualization by means of dynamic partial reconfiguration for **real-time applications**

3 **Prototype implementation with *Zynq***

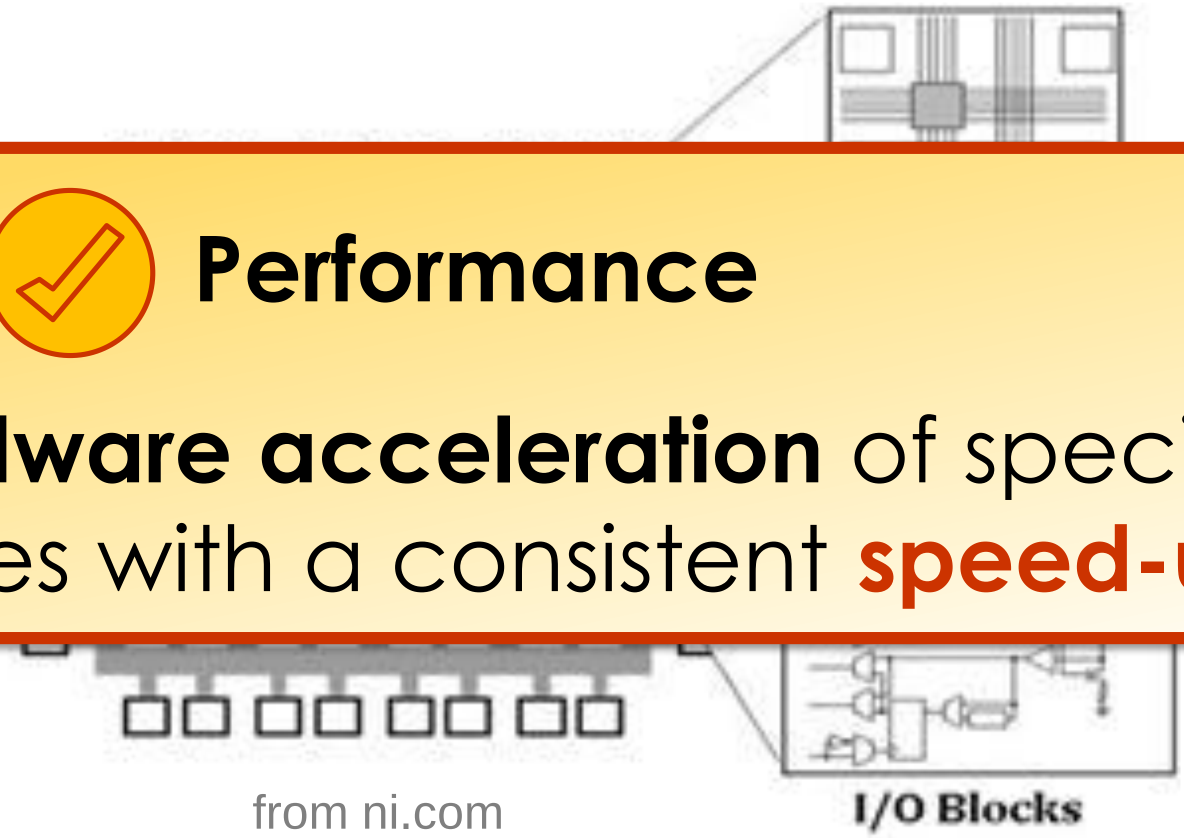
Preliminary **overhead** and **performance** evaluation show encouraging results

4 **Supporting FRED in Linux on *Zynq***

Enabling **predictable** FPGA virtualization for **Linux**

What is a FPGA?

- A field-programmable gate array (**FPGA**) is an **integrated circuit** designed to be configured (by a designer) **after** manufacturing
- FPGAs contain an array of **programmable logic blocks**, and a hierarchy of reconfigurable interconnects that allow to *"wire together"* the blocks.



Performance

Ad-hoc **hardware acceleration** of specific functionalities with a consistent **speed-up**

from ni.com

Dynamic Partial Reconfiguration

- Modern FPGA offers dynamic partial reconfiguration (**DPR**) capabilities.
- DPR allows **reconfiguring** a portion of the FPGA at **runtime**, while the rest of the device continues to operate.
- **DPR** opens a **new dimension** in the resource management problems for such platforms.
- Likewise multitasking, **DPR** allows **virtualizing** the FPGA area by "*interleaving*" (at *runtime*) the configuration of multiple functionalities

Analogy with multitasking

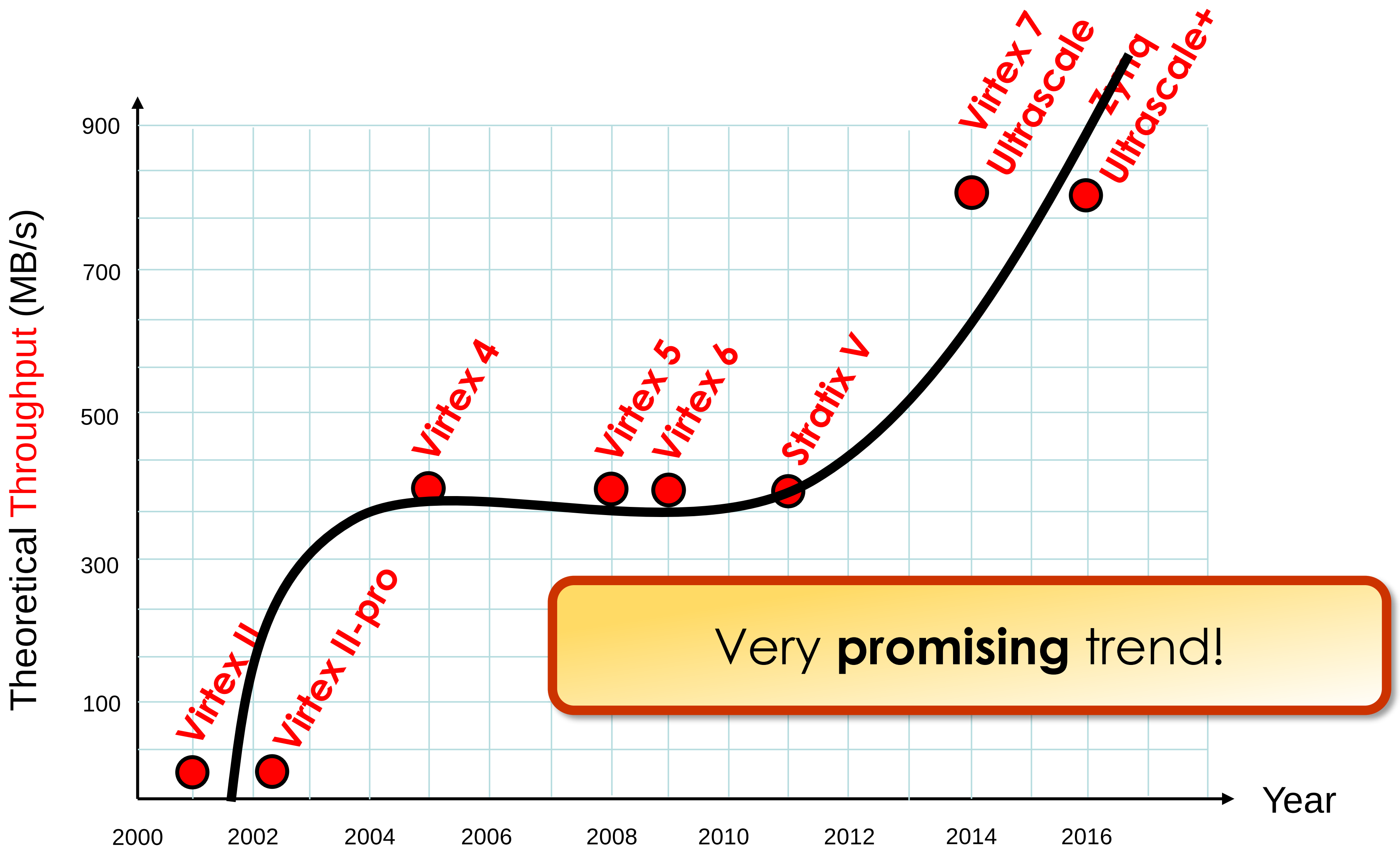
CPU	FPGA
Context switch CPU registers	**DPR** FPGA config. memory
Tasks SW	**Hardware accelerators** Programmable logic

Analogy with virtual memory

$$\begin{tikzcd}\textbf{Memory} \arrow[d, dashed, leftrightarrow] \\ \textbf{FPGA Area}\end{tikzcd}$$

The Payback

- **DPR** does **not** come for **free**!
  - **Reconfiguration times** are **~3** orders of magnitude higher than context switch times in today's processors.
  - Determines further complications in the resource management problems.



INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

Scuola Superiore Sant'Anna

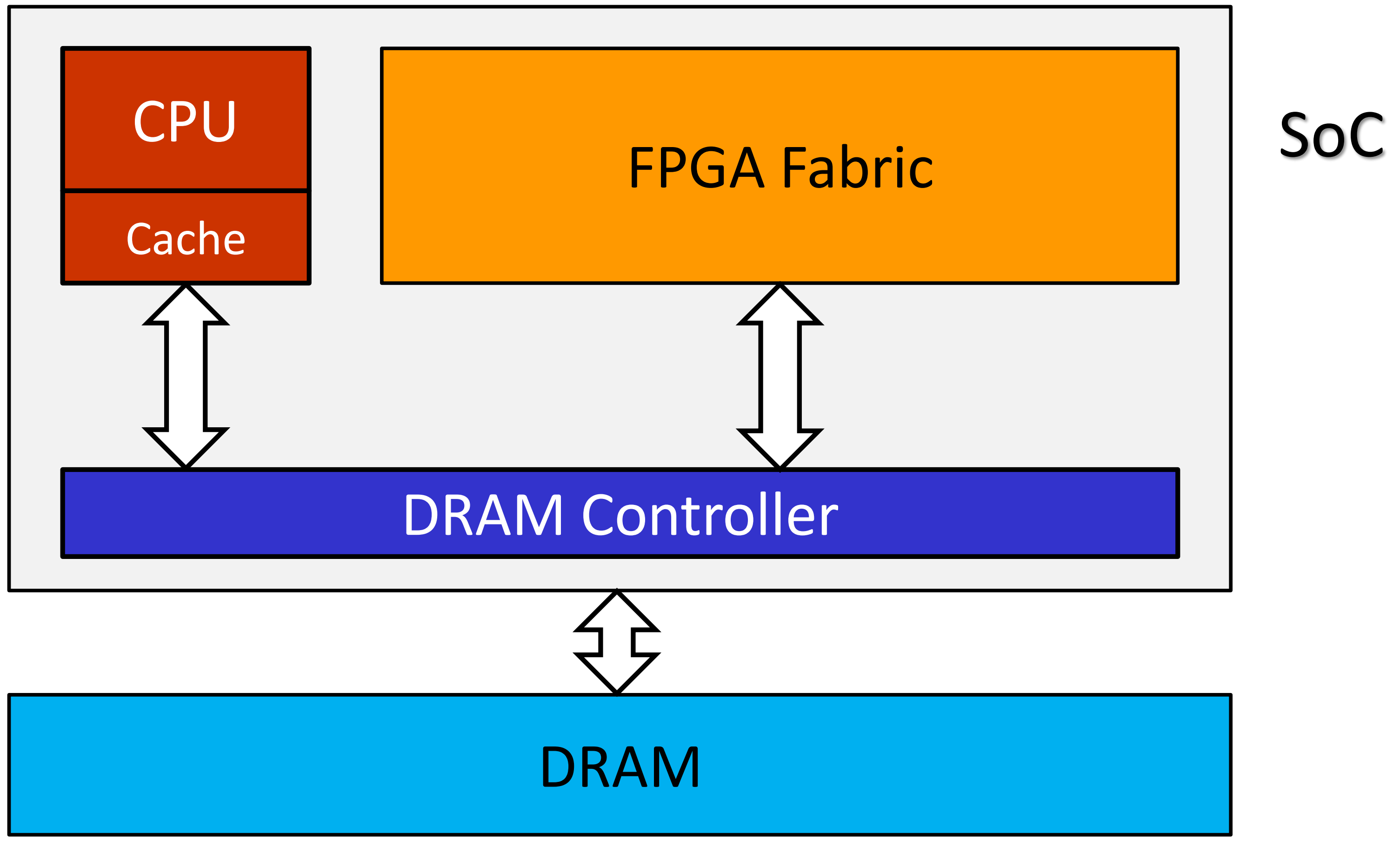
Retis Real-Time Systems Laboratory

The FRED Framework

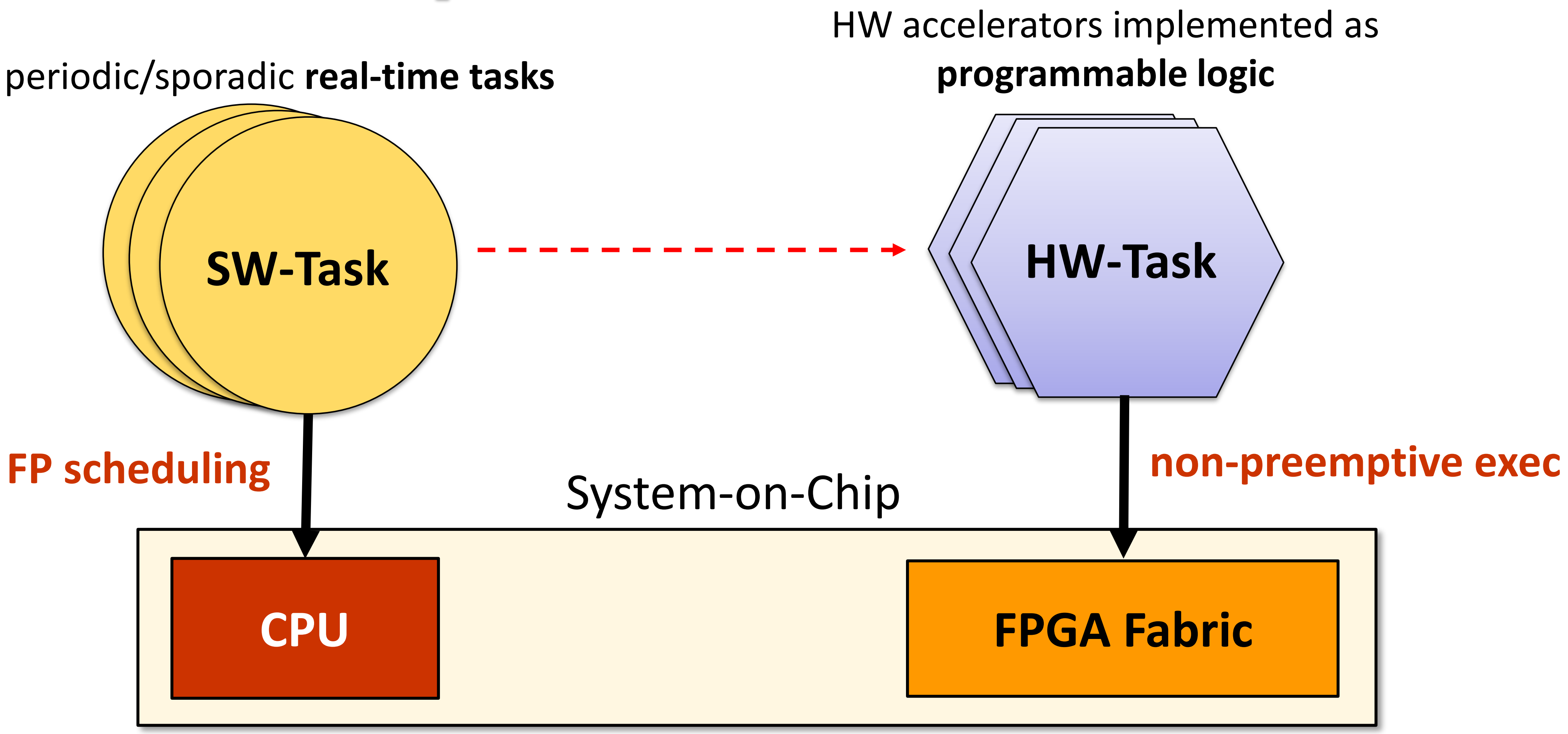
Exploiting dynamic reconfiguration of FPGAs to support real-time applications

System Architecture

- System-on-chip (**SoC**) that includes:
  - One **processor**;
  - One DPR-enabled **FPGA** fabric;
- DRAM **shared memory**.



Computational Activities


periodic/sporadic **real-time tasks**
HW accelerators implemented as **programmable logic**
SW-Task
HW-Task
FP scheduling
non-preemptive exec
System-on-Chip
CPU
FPGA Fabric

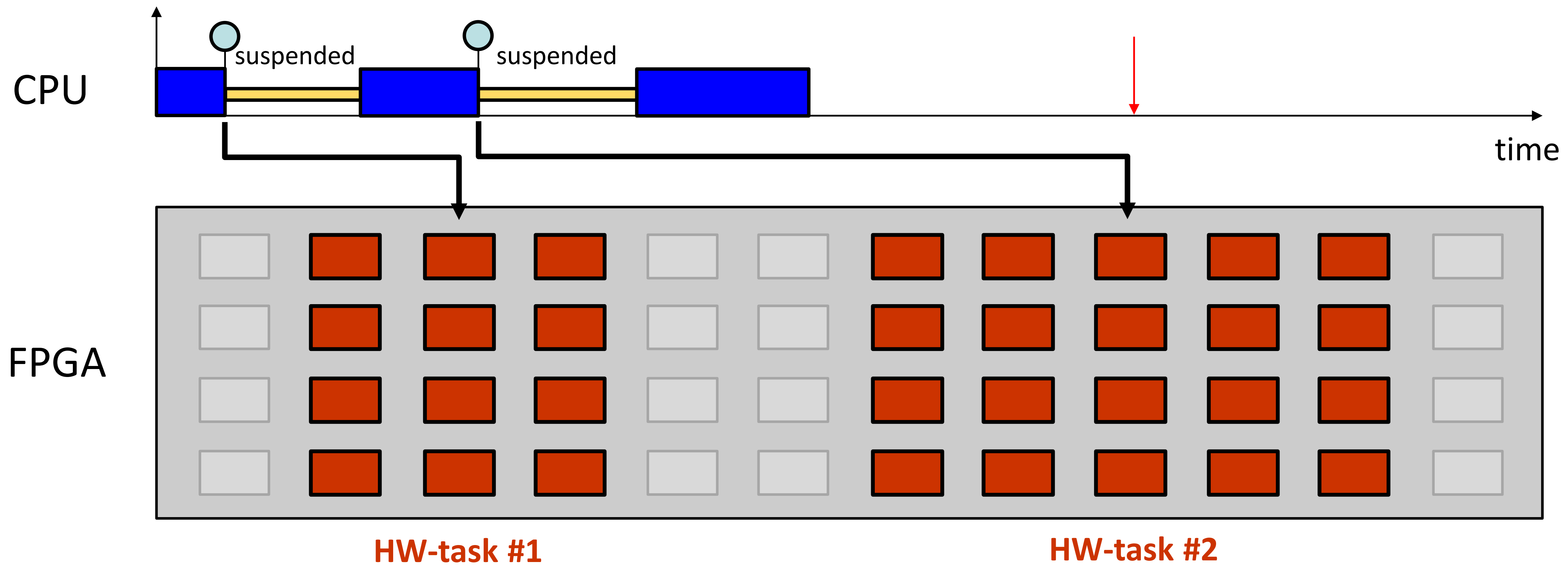
SW-Task

```
TASK(myTask)
{
        <…>
        <prepare input data>
        EXECUTE_HW_TASK(myHWtask);
        <retrieve output data>
        <…>
}
```

Suspend the execution until the **completion** of the HW-task

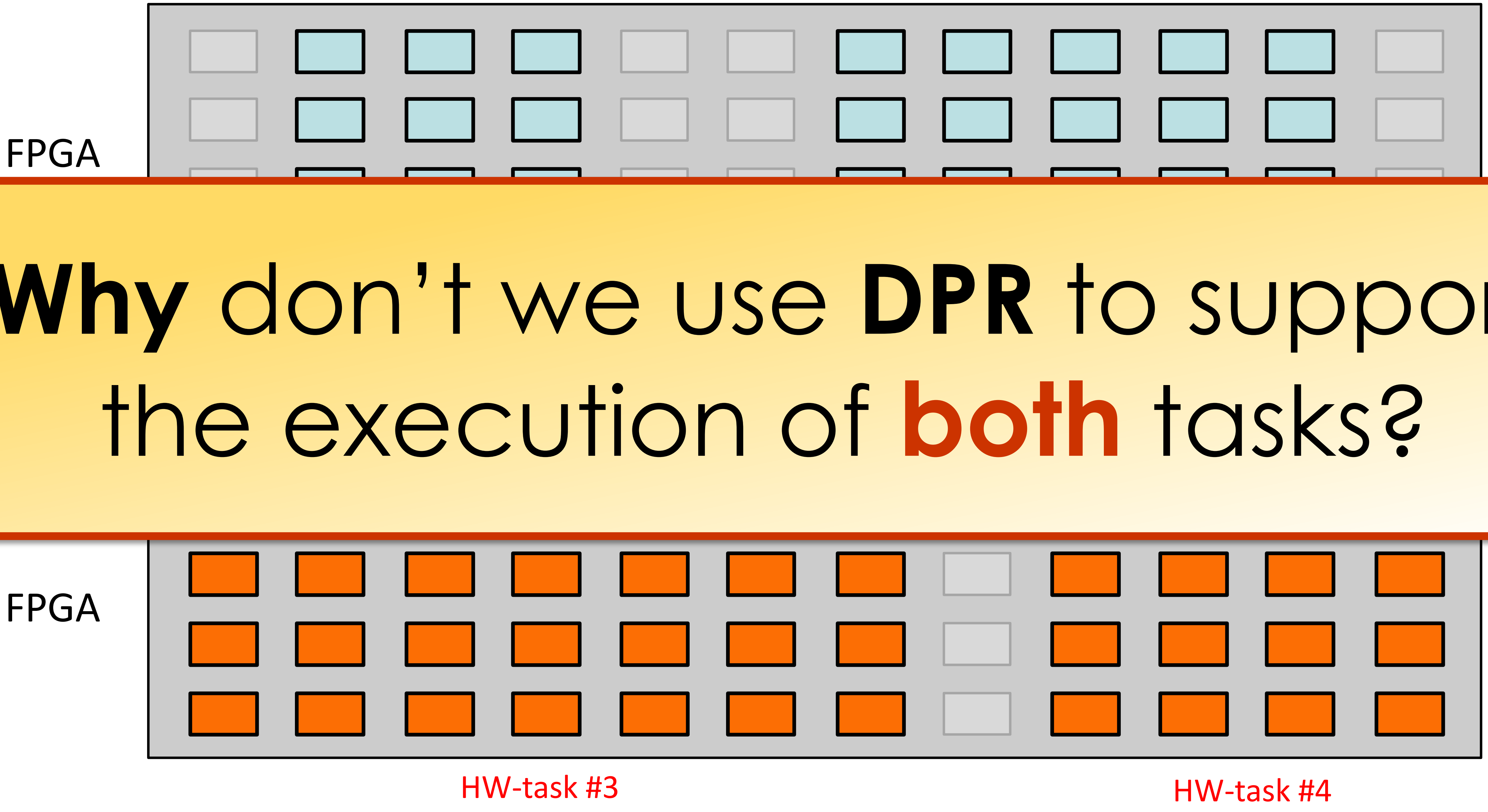
SW- and HW-Tasks

- A **SW-task** using **two HW-tasks**
- The SW-task has **3 execution regions** and self-suspends when HW-tasks execute



SW- and HW-Tasks

- Suppose we also want to execute ***another SW-task***, using two heavy HW-tasks that occupy **almost all** the FPGA area



Why don't we use **DPR** to support the execution of **both** tasks?

Reconfiguration Interface

- **DPR**-enabled FPGAs dispose of a **FPGA reconfiguration interface** (**FRI**) (e.g., PCAP, ICAP on Xilinx platforms).

- In most **real-world** platforms, the **FRI**

✓ can reconfigure an area **without affecting** HW-tasks that are executing in other areas;

✓ is an **external device** to the processor (e.g., like a DMA);

X can program **at most one** slot *at a time*.

Reconfiguration can be **preemptive** or **non-preemptive**

Single resource → **Contention**

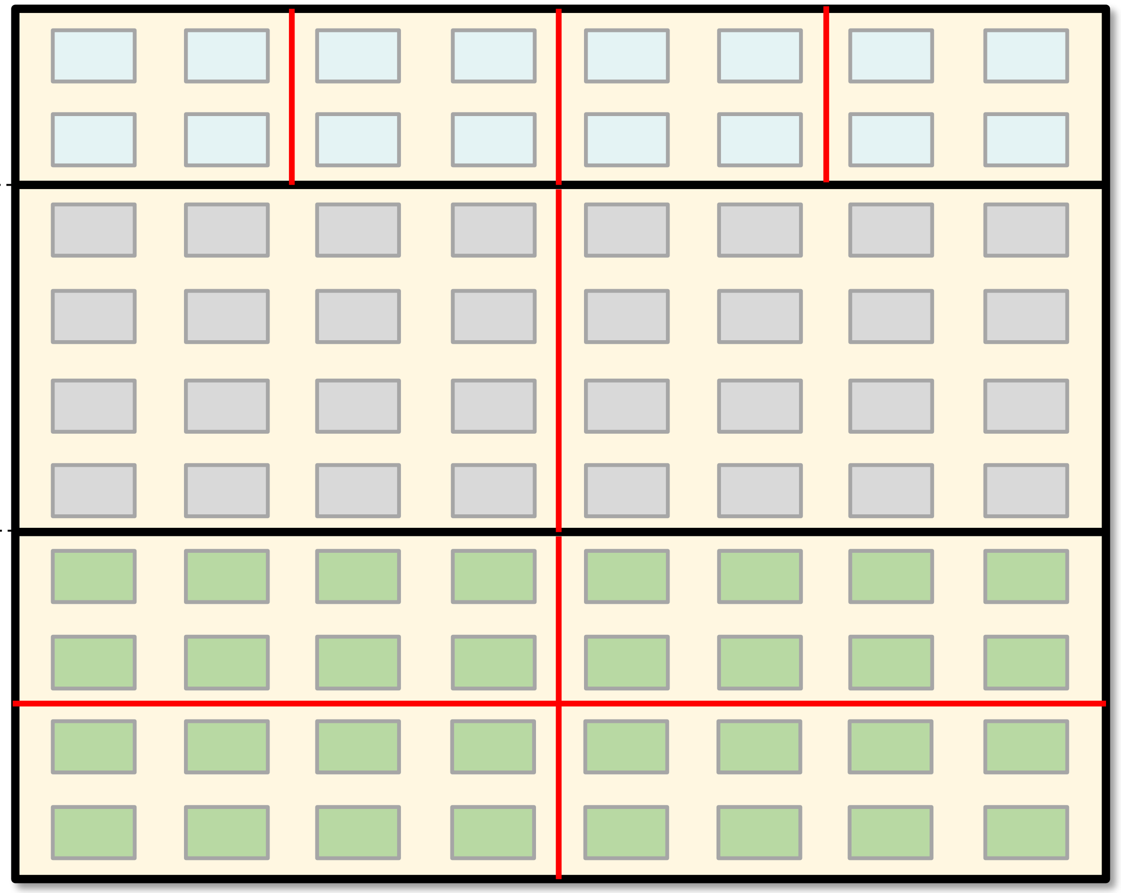
Slotted Approach

- FPGA area partitioned into **partitions**, each of them in-turn partitioned into **slots**
- HW-Tasks are programmed onto slots of a fixed partition (affinity)
- Partitioning can be done **off-line** as a function of the taskset

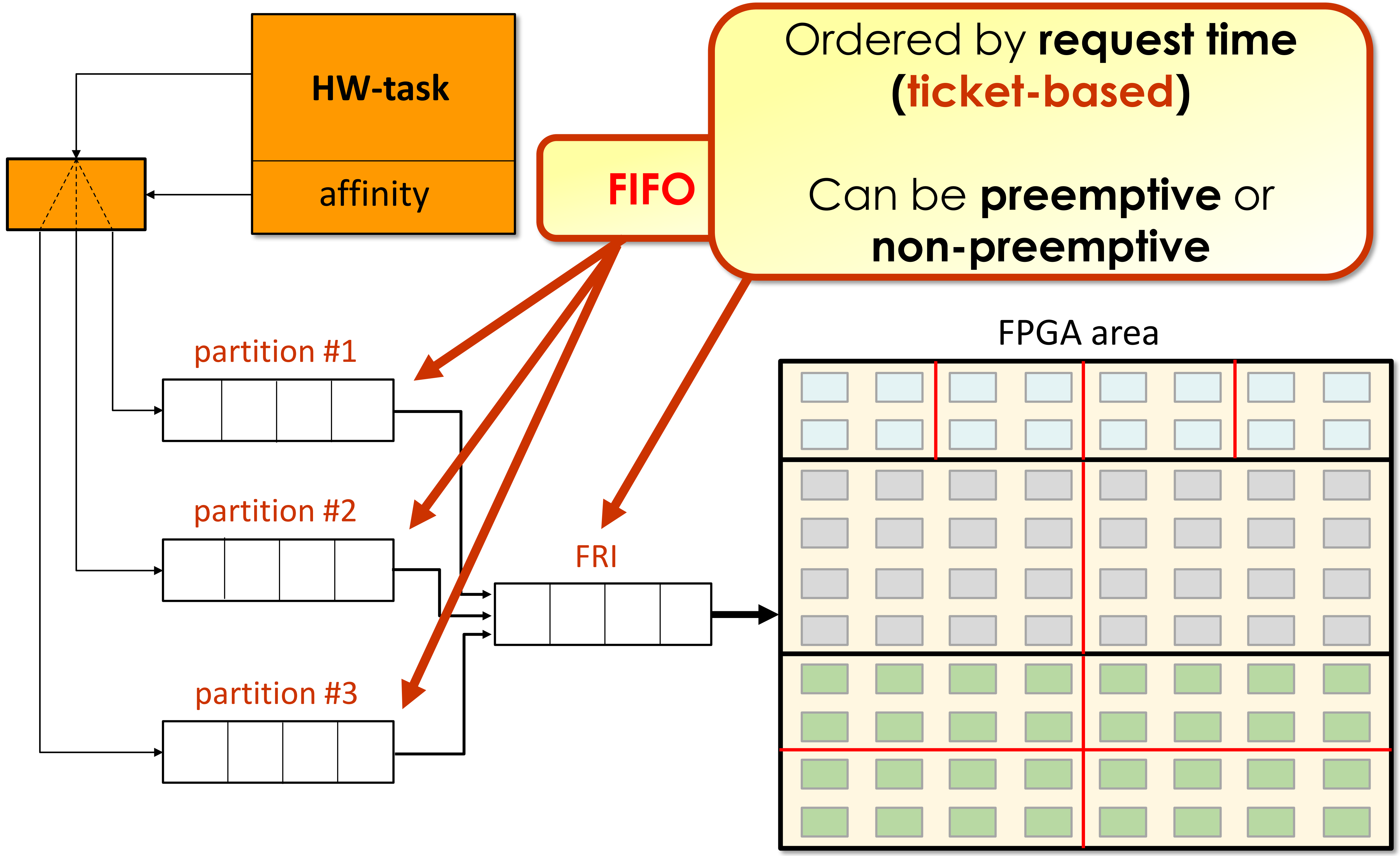
Partition #1
4 slots of 4 logic-blocks

Partition #2
2 slots of 16 logic-blocks

Partition #3
4 slots of 8 logic-blocks



Scheduling Infrastructure

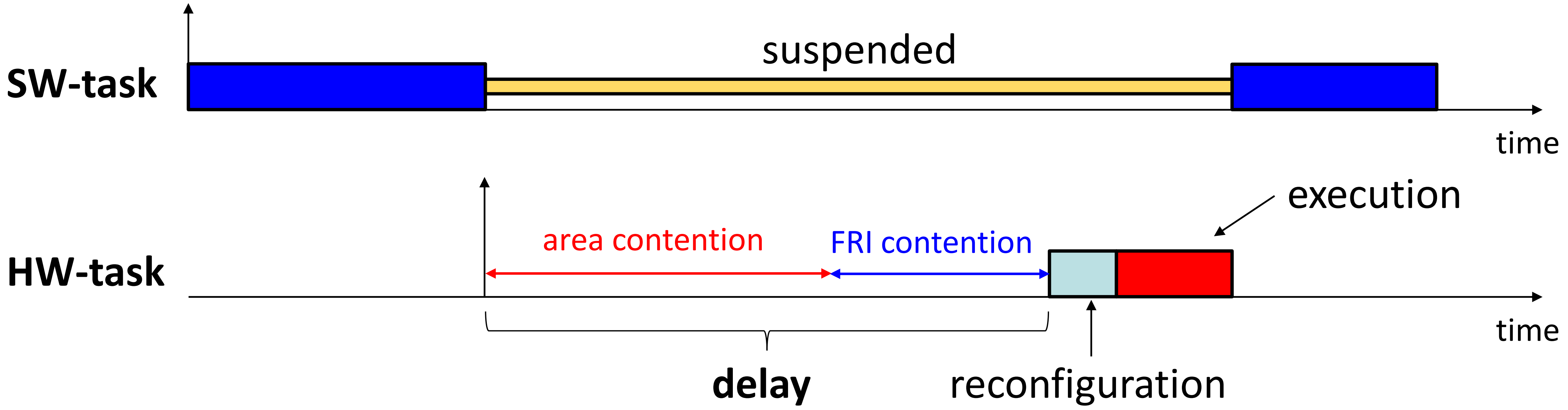


Ordered by **request time** (**ticket-based**)

Can be **preemptive** or **non-preemptive**

Response Time Analysis

- In Biondi et al. [RTSS'16] we derived *upper-bounds* on the **delay** incurred by **SW-tasks** when requesting the execution of **HW-tasks**
  - **delay** = **slot** contention + **FRI** contention
- Once computed the **delay bound**, we can ***transform*** each SW-task into a fixed-segment self-suspending task (SS-Task)
  - **Suspension** = delay bound **+** reconfiguration time **+** HW-task **WCET**
  - Can be analyzed using **Nelissen et. al**'s response-time analysis for SS-Tasks [ECRTS'15]



INSTITUTE
OF COMMUNICATION,
INFORMATION
AND PERCEPTION
TECHNOLOGIES

Scuola Superiore
Sant'Anna

Retis
Real-Time Systems Laboratory

Prototype implementation with Zynq

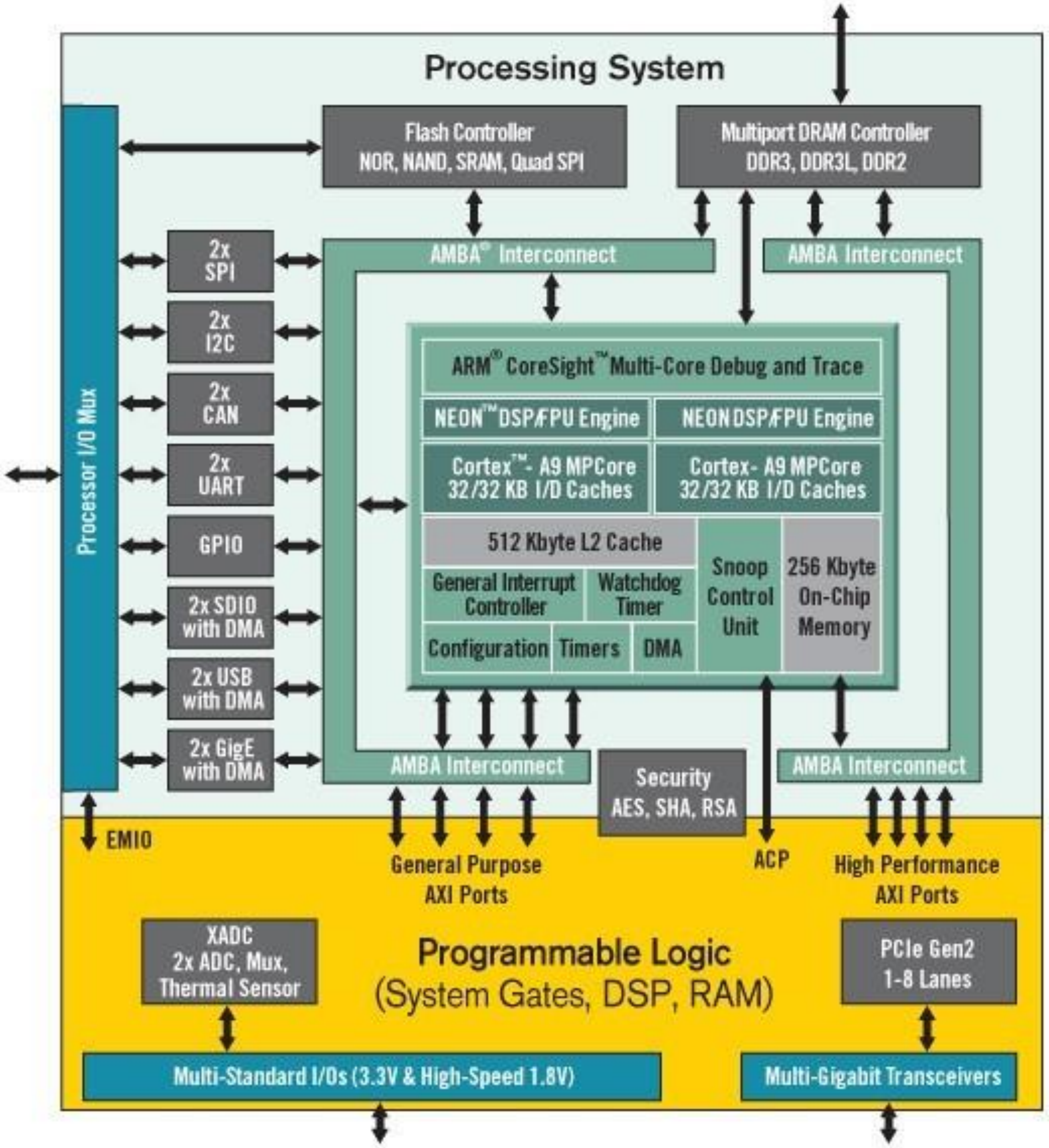
Preliminary overhead and performance evaluation

Reference Platform

Xilinx Zynq-7000 SoC

- 2x ARM Cortex A9
- Xilinx series-7 FPGA
- AMBA Interconnect

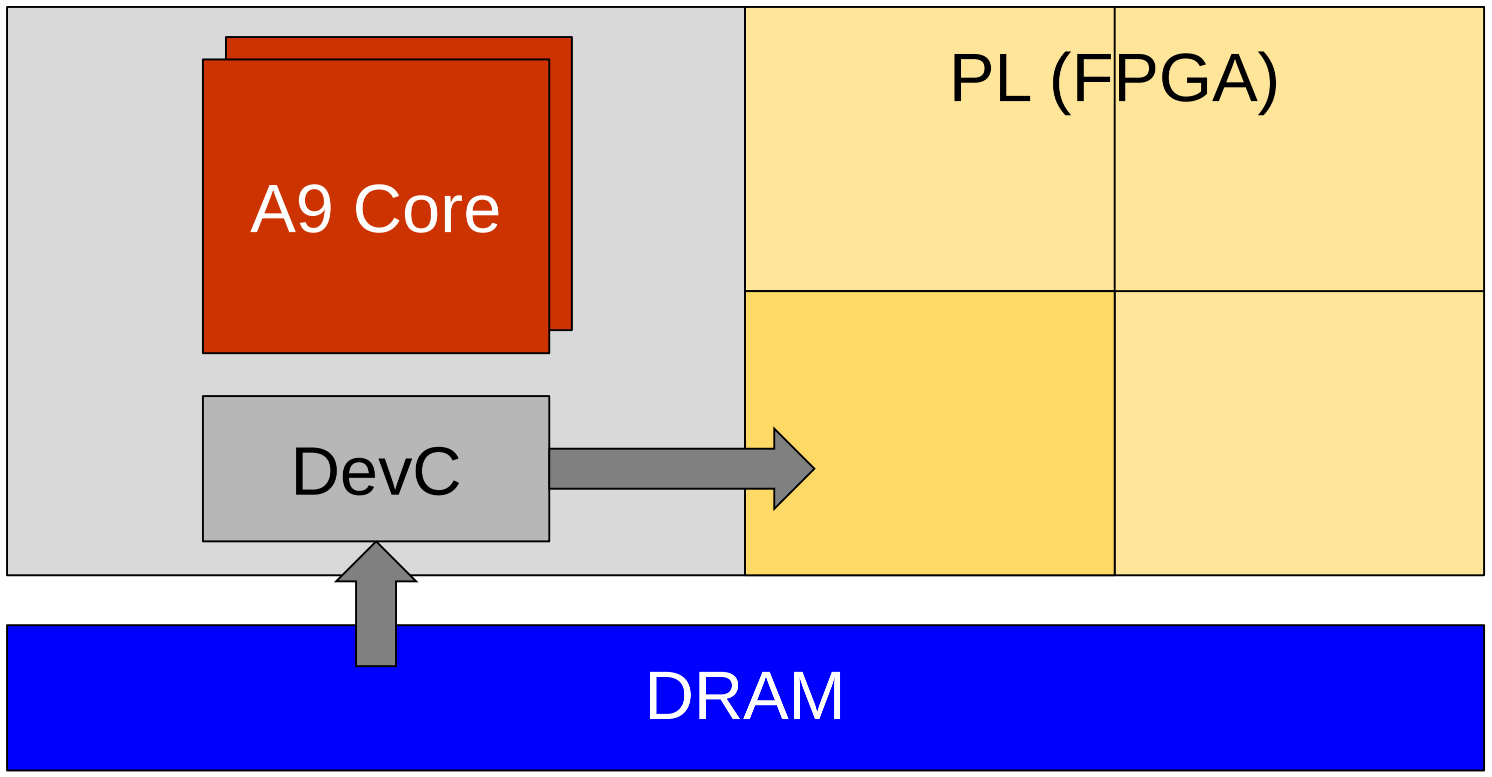
Prototype FRED implementation on top of FreeRTOS



Processing System
Processor I/O Mux
Flash Controller NOR, NAND, SRAM, Quad SPI
Multiport DRAM Controller DDR3, DDR3L, DDR2
2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO with DMA
2x USB with DMA
2x GigE with DMA
AMBA® Interconnect
AMBA Interconnect
ARM® CoreSight™ Multi-Core Debug and Trace
NEON™ DSP/FPU Engine
NEON DSP/FPU Engine
Cortex™- A9 MPCore 32/32 KB I/D Caches
Cortex- A9 MPCore 32/32 KB I/D Caches
512 Kbyte L2 Cache
General Interrupt Controller
Watchdog Timer
Configuration
Timers
DMA
Snoop Control Unit
256 Kbyte On-Chip Memory
AMBA Interconnect
Security AES, SHA, RSA
AMBA Interconnect
EMIO
General Purpose AXI Ports
ACP
High Performance AXI Ports
XADC 2x ADC, Mux, Thermal Sensor
Programmable Logic (System Gates, DSP, RAM)
PCIe Gen2 1-8 Lanes
Multi-Standard I/Os (3.3V & High-Speed 1.8V)
Multi-Gigabit Transceivers

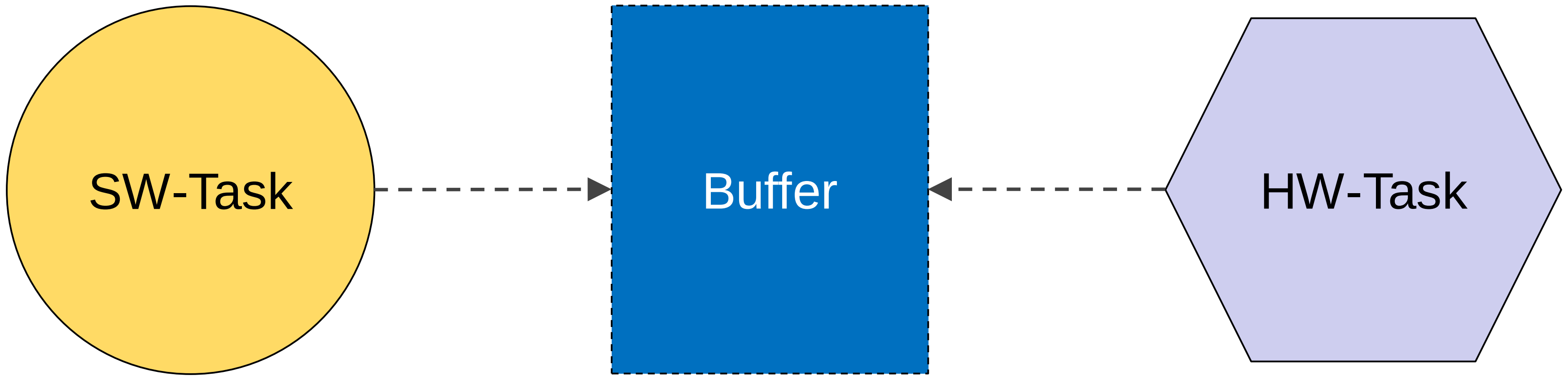
FRED on Zynq - FRI

- Built-in **device configuration subsystem** called *DevC*:
  - Internal interface to the **PCAP** port and a **DMA** engine.
    - *Can transfer a bitstream from the DRAM to the PL configuration memory.*
    - *No CPU cycles wasted during reconfiguration.*



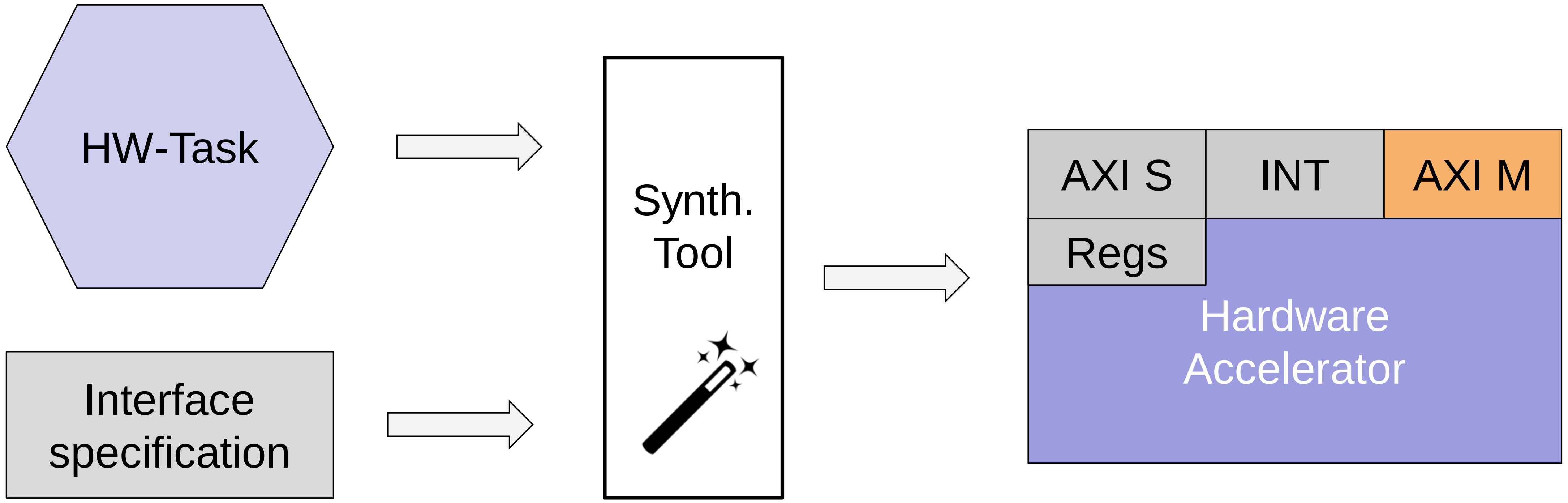
FRED on Zynq - Shared memory

- How to implement **FRED's shared memory** paradigm:
  - ✗ **PS** on chip memory (**OCM**)?
    - ***Too small*** *(256 KB) for many HW-Tasks.*
  - ✗ **PL buffers** using BRAMs?
    - *Small amount and **waste of resources**.*
  - ✓ Off-chip **DRAM**?
    - *Large amount and architecturally suitable:*
      - *Direct access from PL to DRAM controller through AXI HP ports.*

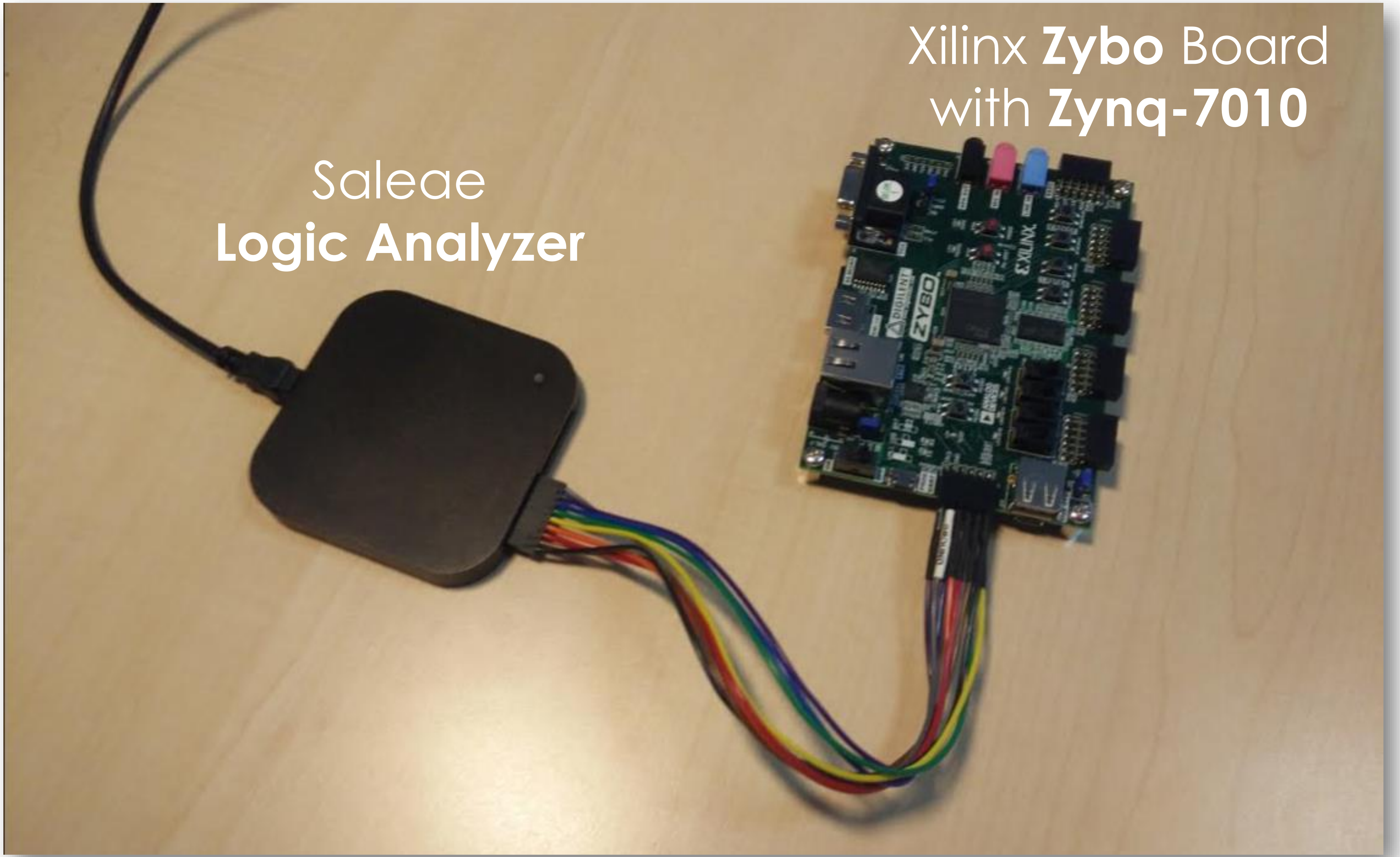


FRED on Zynq - Support design

- Each **slot** must be able to accommodate **any** kind of **HW-Task** belonging to its **partition**:
  - it is necessary to define a **common interface**:
    - **AXI MM Master** for accessing DRAM;
    - **AXI MM Slave** for control and up to 8 data registers;
      - data regs are HW-T dependant: pointers or params.
    - **Done signal** for interrupt signalling.



Experimental Setup



Case Study

- Four computational activities:
  - **Sobel** image filter @ 100ms
  - **Sharp** image filter @ 150ms
  - **Blur** image filter @ 170ms

  800x600 @ 24-bit

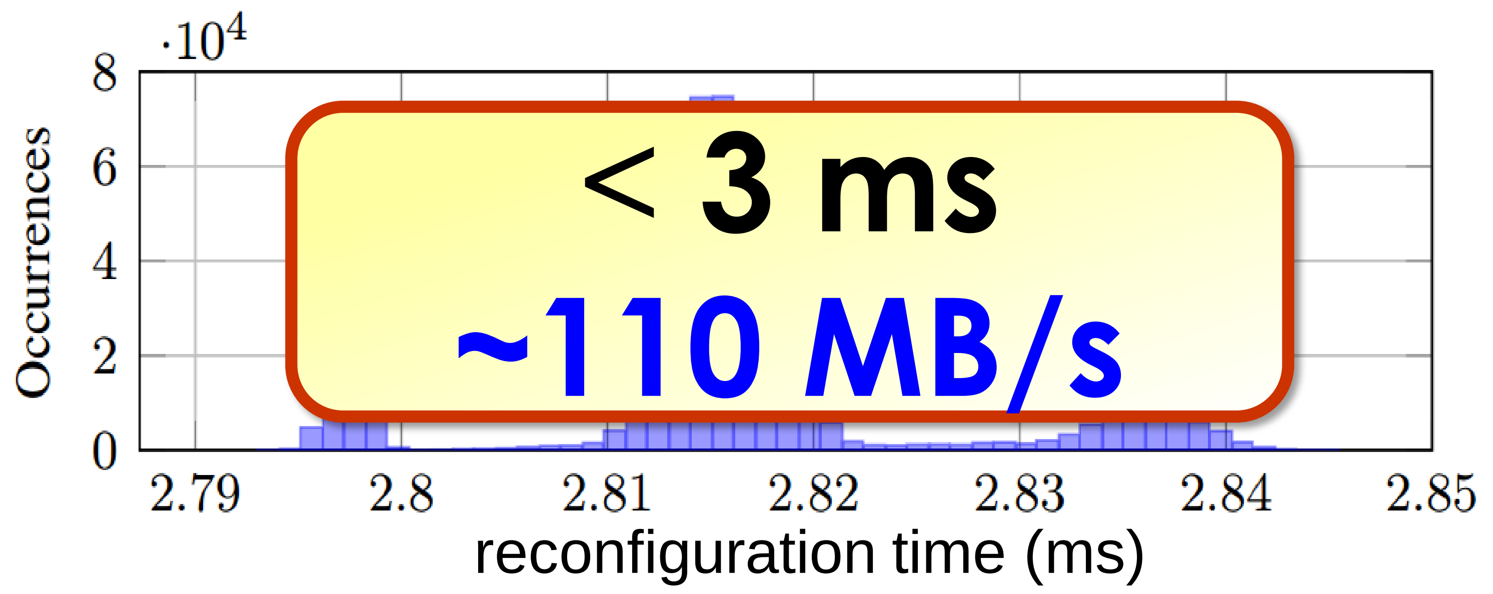
  - **Matrix** multiplier @ 2500ms

  512x512 elements

- Both HW-task and pure SW-task versions have been implemented
  - **Xilinx Vivado HLS** synthesis tool for HW-tasks
  - **C** language for SW-tasks

Reconfiguration Time and Speed-up

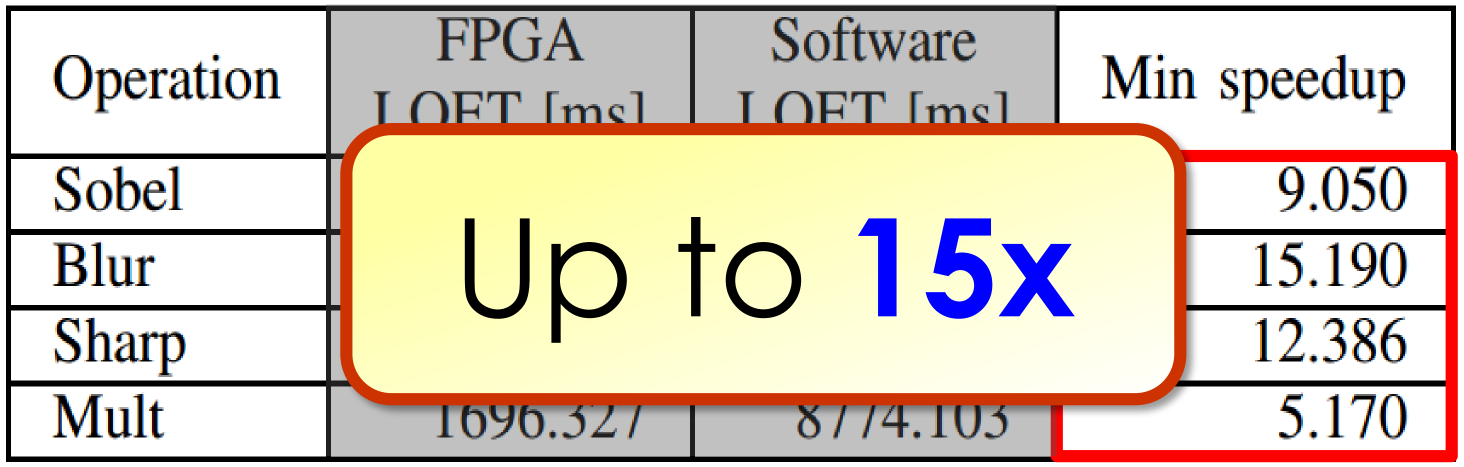
Time needed to reconfigure a region of ~4K logic cells, **25%** of the total area



< 3 ms

~110 MB/s

Speed-up analysis comparing SW-task and HW-task implementations

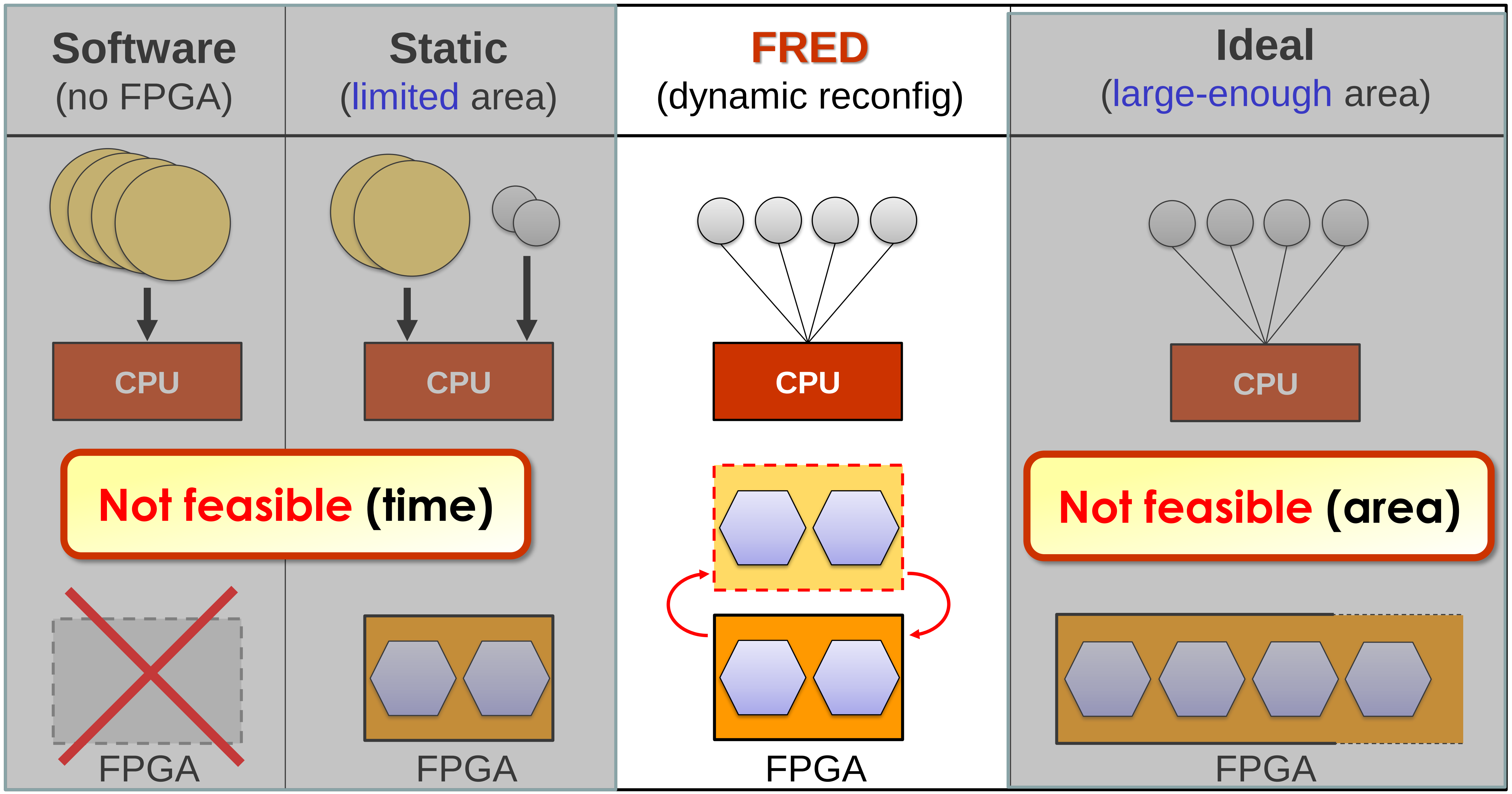


Operation	FPGA LOET [ms]	Software LOET [ms]	Min speedup
Sobel	[illegible]	[illegible]	9.050
Blur	[illegible]	[illegible]	15.190
Sharp	[illegible]	[illegible]	12.386
Mult	1696.327	8774.103	5.170

Up to **15x**

CPU: Cortex A9 @ 650Mhz **FPGA**: Artix-7 @ 100Mhz

Possible Approaches



Response Times

- The case study is **not feasible**
  - with a **pure SW** implementation (CPU *overloaded*);
  - with **any combination** of SW and statically configured HW tasks (only two of them can be programmed).

With **FRED** we never observed a deadline miss in a **8-hour run**

Task	Period [ms]	Longest Observed Response Time [ms]
Sobel	100	43.748
Blur	150	69.438
Sharp	170	74.855
Mult	2500	1723.200

INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

Scuola Superiore Sant'Anna

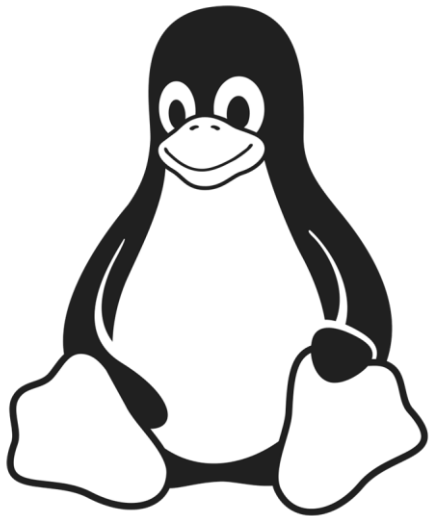
Retis Real-Time Systems Laboratory

Supporting FRED in Linux on Zynq

Enabling predictable FPGA virtualization for Linux

FRED on Linux - How to…

- Implement **FRED's** shared **memory buffers**?
  - **Linux** uses **virtual memory**!
    - *Each **SW-Task** (process) has its own **virtual address** space;*
    - ***HW-Tasks**, like other HW devices, use **physical addresses**;*
    - *How to handle cache coherence?*
- Implement the **FRED's scheduling policy**?
  - *Receive and handle acceleration requests.*
- Access and control **hardware resources**:
  - ***HW-Accelerators** modules;*
  - ***DevC**, **Decouplers**.*



FredLinux - Software design keypoints

- FredLinux had been implemented, as much as possible, in **user-space** to improve **maintainability** and **safety**:
  - *User space process to handle and schedule acceleration requests;*
  - *Minimal kernel support.*
- **Zero-copy** design for **shared buffers** to **avoid** unnecessary **copy** operations overhead and related **memory traffic**;
  - *Linux DMA layer provides functions for allocating and mapping **large coherent** memory buffers (using CMA).*
- **Modular design** to allow reusability and future extensions:
  - *Core mechanisms are independent from the platform and hardware specific support.*

FredLinux - Software architecture overview

- The **central component** of **FREDLinux** is a user space process named **FRED server** (fredd):
  - **Receives** and **handles** acceleration request from **SW-Tasks**.
  - Relies upon **two custom kernel modules**, and the **UIO framework**, for **low-level** operations.

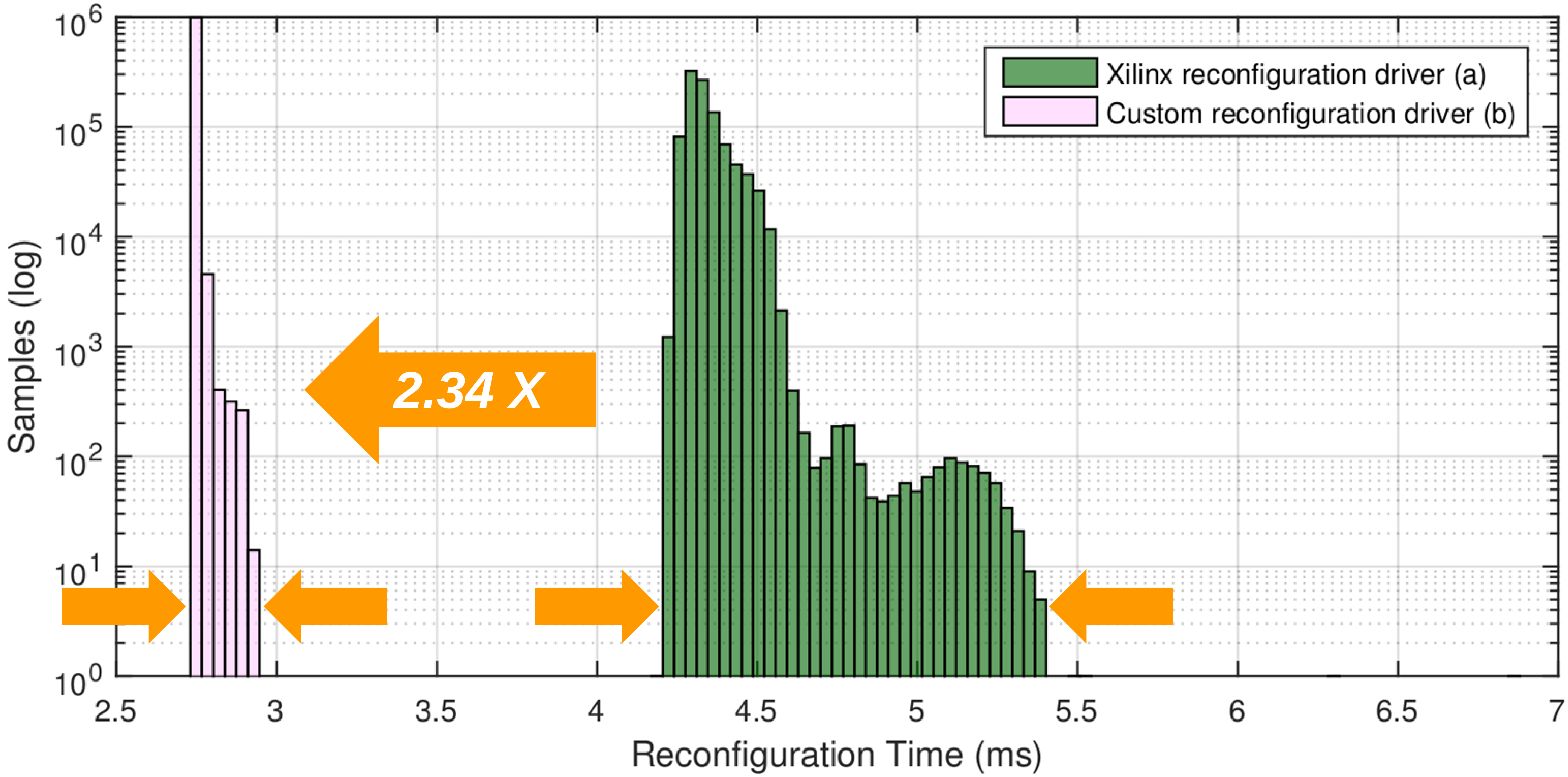
- Kernel modules functionalities:
  - ***Buffer allocator*** provides shared memory buffers;
  - ***DevC*** custom driver for reconfiguration;

- ***UIO framework:*** userspace drivers for **HW-Tasks** and decouplers.

FredLinux - Kernel space - Reconfiguration Driver

- Xilinx's **reconfiguration** **(DevC) driver** is designed to be safe and easy to use, **not for efficiency**:
  - For **each** reconfiguration:
    - **Allocates** a **new** contiguous **buffer**;
    - **Copies** the whole bitstream from **userspace to kernel**;
    - **Busy wait** until completion.
- **Unsuitable** for the **intensive use** of partial reconfiguration required by **FRED**!
- To overcome those issue the **DevC driver** has been **modified**:
  - **Exploit** the **allocator module** to preload all the **bitsreams** in a set of physically contiguous buffers;
- Now the reconfiguration can also be initiated by an ioctl() call passing, as argument, a reference to the buffer;
  - **Minimal overhead**!

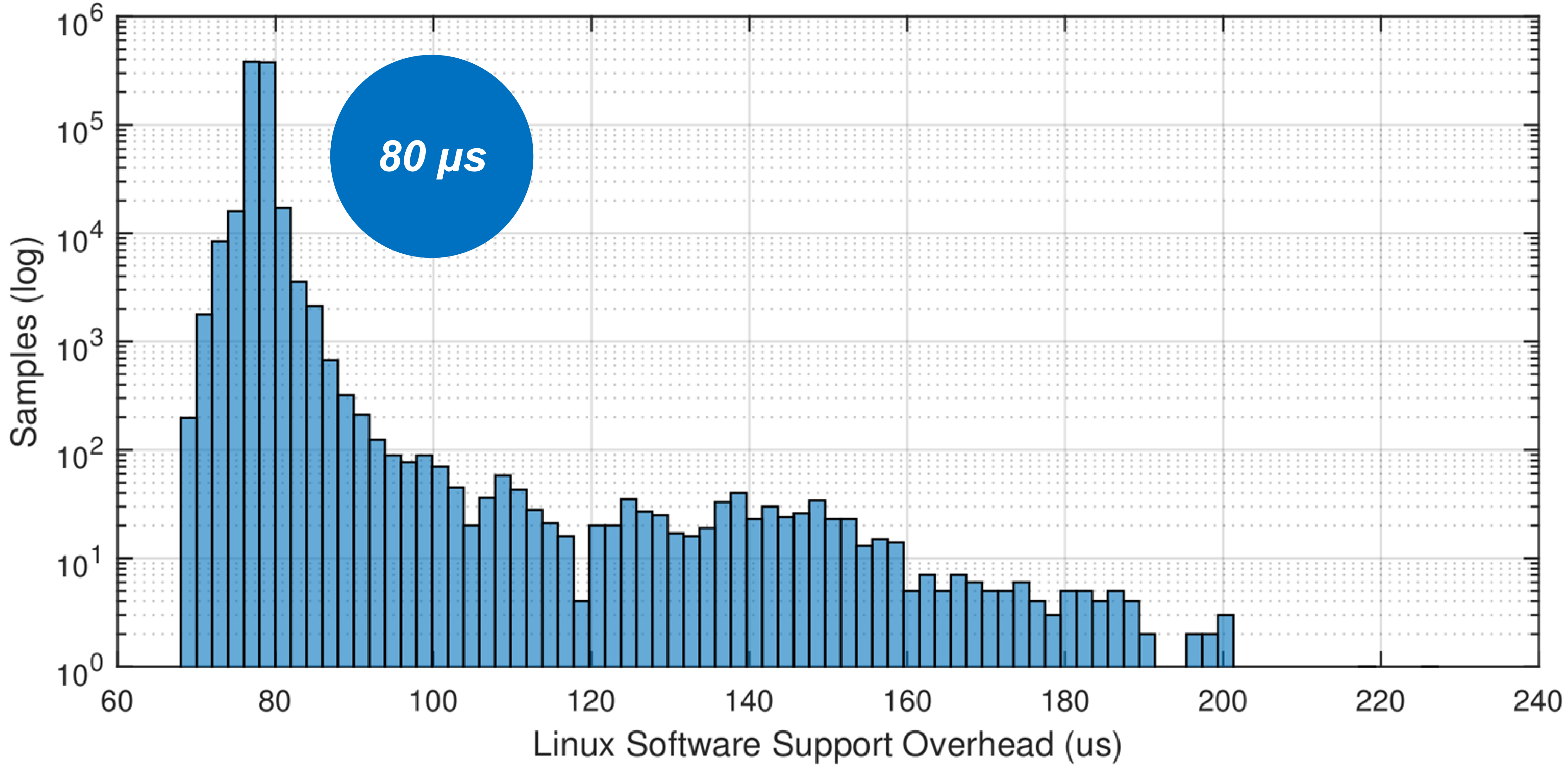
FredLinux - Reconfiguration driver performance



- For a 338 KB bitstream, worst case: 2.94 ms vs 6.87 ms
  - ***Speedup*** of **2.34 X** and ***reduced variance***.

FredLinux - Overhead evaluation

- For a single acceleration request:



Maximum measured **overhead** **less than** **228 μs** (Average **80 μs**)

Conclusions

- Presented a framework to support the development of **real-time applications** on top of **SoC** including both a **CPU** and a **DPR-enabled FPGA**
- Proposed **response-time analysis**
- Performed a validation with a prototype **implementation** in a RTOS
- Implemented in Linux with:
  - **Improvement** of **reconfiguration** times by a factor of **2.34 X** wrt stock driver
  - Maximum measured **overhead** introduced by software support **less than 228 μs**

- **Reconfiguration times** in today's platforms are **not prohibitive** (and are *likely to decrease* in future)
- **DPR** can **improve** the performance of real-time application upon *static FPGA management*

Future Work

- There are **a lot of possible future works** and open problems
  - Development and analysis of other scheduling algorithms for HW-tasks
  - Worst-case analysis of the interconnect
  - Investigation on partitioning approaches for the FPGA
  - Integration of the support for preemptive FRI
  - …

Thank you!

Mauro Marinoni - m.marinoni@santannapisa.it