Towards Real-Time Operating Systems for Heterogeneous Reconfigurable Platforms

Marco Pagani, Mauro Marinoni, **Alessandro Biondi**
Alessio Balsini and Giorgio Buttazzo

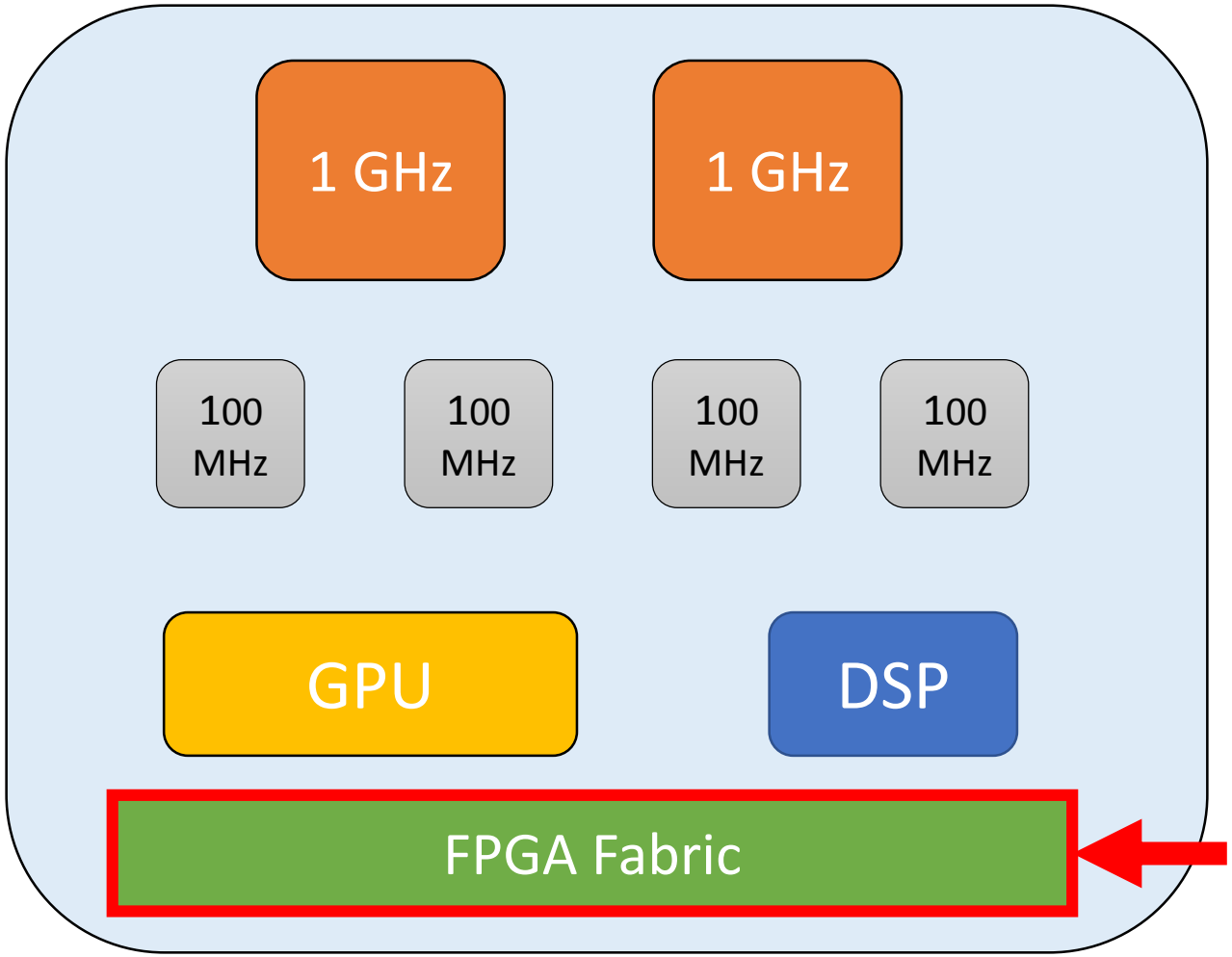
Scuola Superiore Sant'Anna, Pisa, Italy

Sant'Anna
Scuola Universitaria Superiore Pisa

Real-Time Systems Laboratory

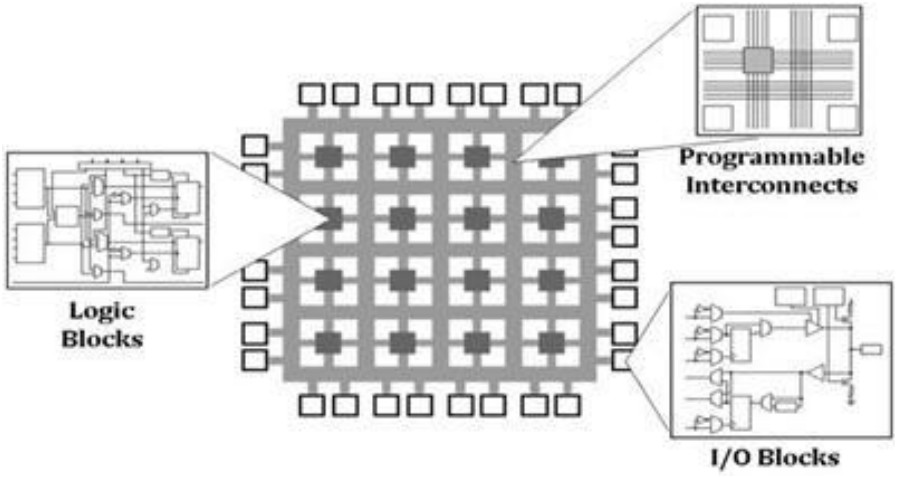
HETEROGENEOUS PLATFORMS

- Emerging in the **embedded** domain



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.



from ni.com

WHY FPGAs?

Top 5 **benefits** (according to National Instruments)

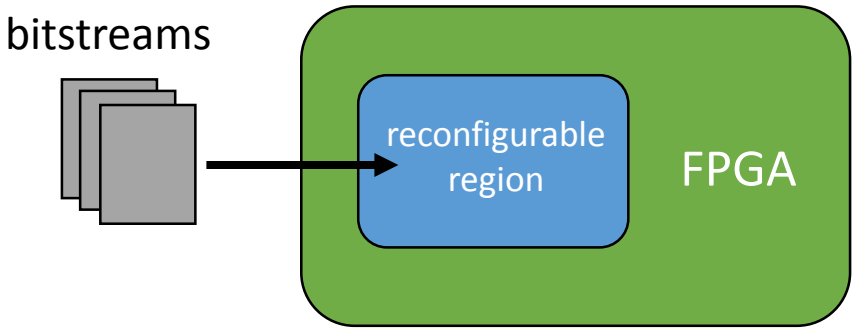
- **Cost**
- **Time-to-market**
- **Reliability**
- **Long-term maintenance**

Performance

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

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.



DYNAMIC PARTIAL RECONFIGURATION

- **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	DPR
CPU registers	FPGA config memory
Tasks	Hardware accelerators
SW	programmable logic

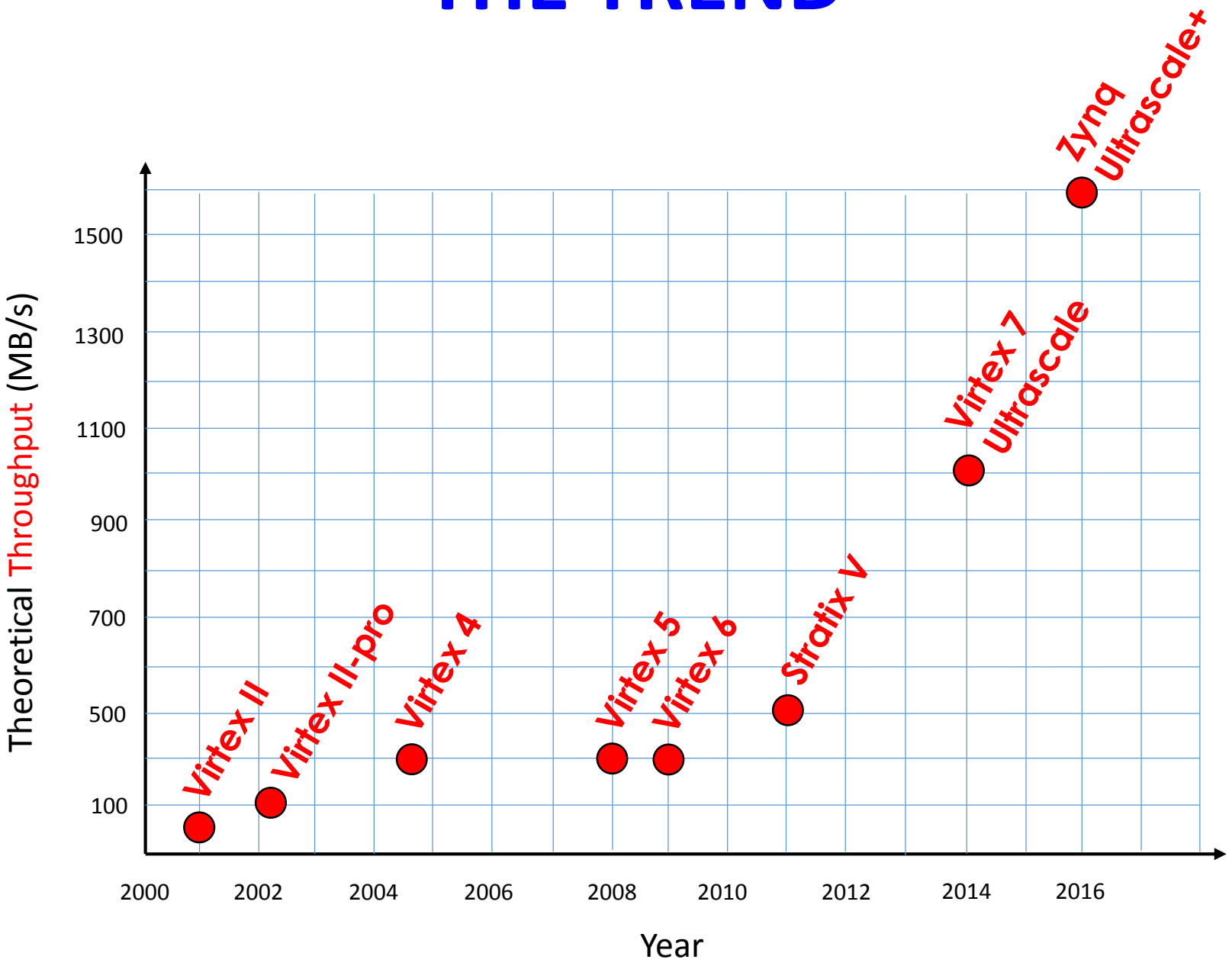
THE PAYBACK

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

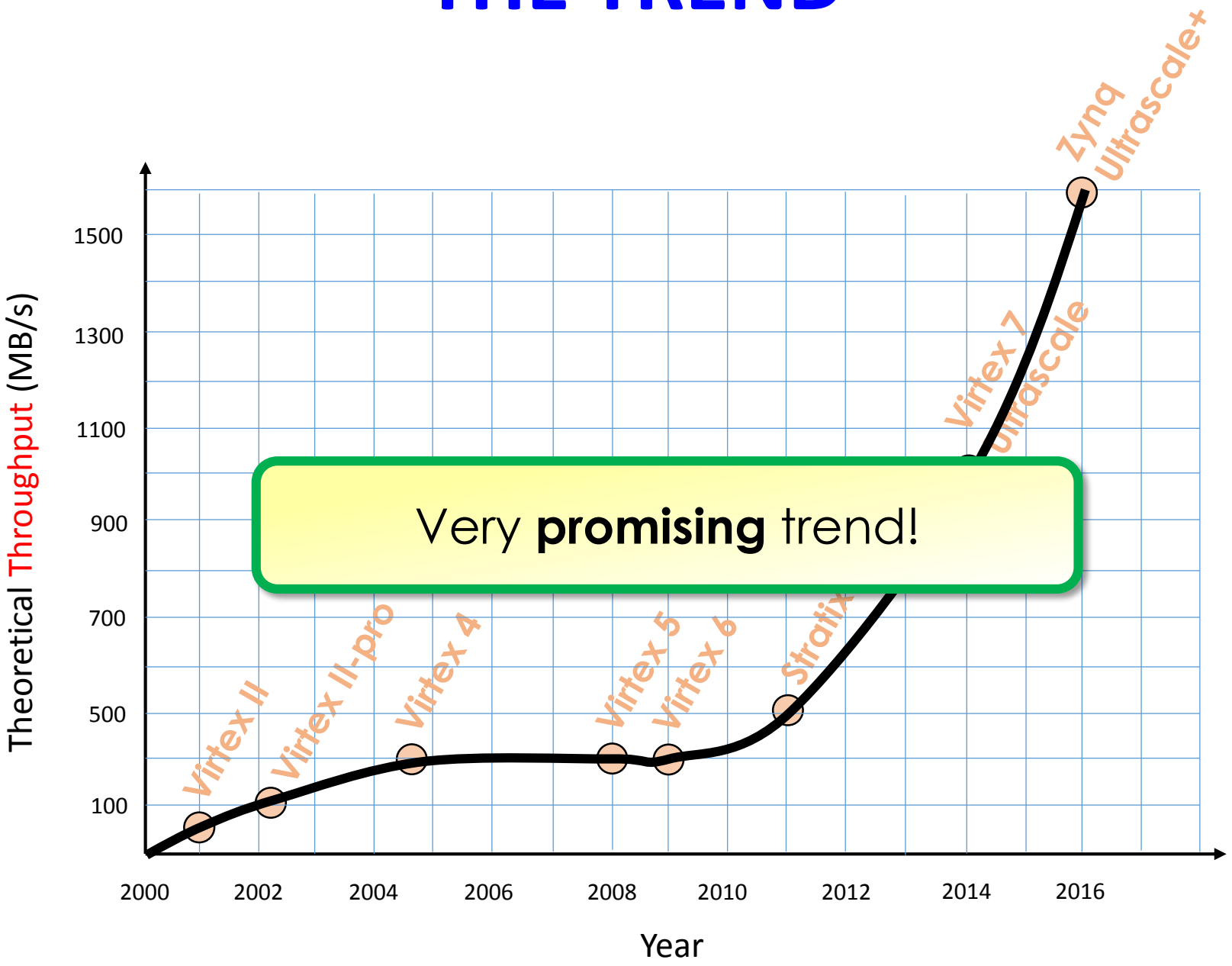
THE TREND


Theoretical Throughput (MB/s)
1500
1300
1100
900
700
500
100
2000
2002
2004
2006
2008
2010
2012
2014
2016
Year
What happened to reconfiguration times in the last 16 years?

THE TREND


Theoretical Throughput (MB/s)
1500
1300
1100
900
700
500
100
2000
2002
2004
2006
2008
2010
2012
2014
2016
Year
Virtex II
Virtex II-pro
Virtex 4
Virtex 5
Virtex 6
Stratix V
Virtex 7
Ultrascale
Zynq
Ultrascale+

THE TREND

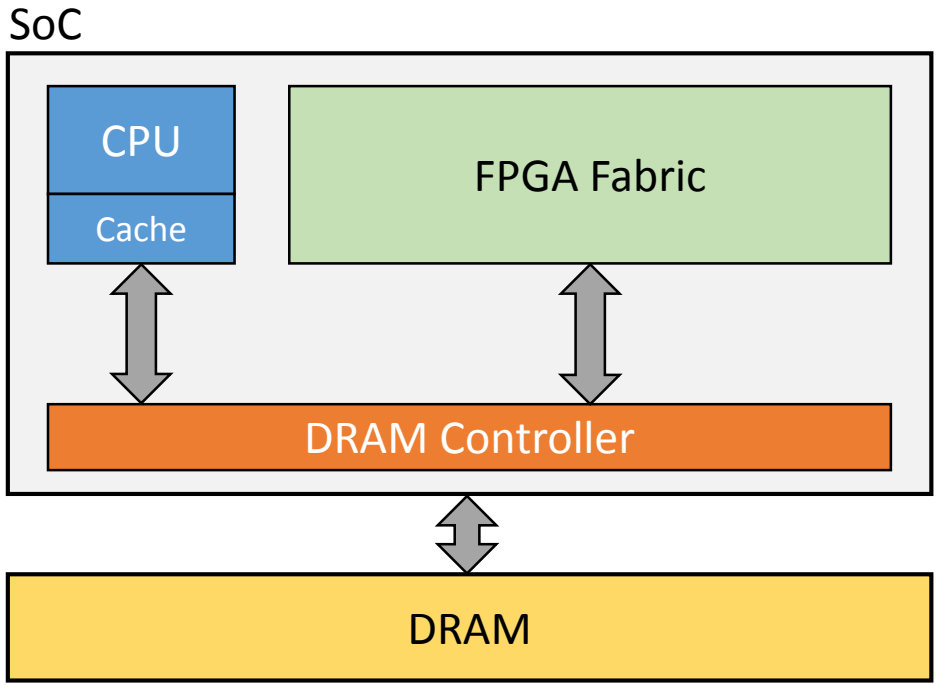

Theoretical Throughput (MB/s)
1500
1300
1100
900
700
500
100
2000
2002
2004
2006
2008
2010
2012
2014
2016
Year
Virtex II
Virtex II-pro
Virtex 4
Virtex 5
Virtex 6
Stratix
Virtex 7
Ultrascale
Zynq
Ultrascale+
Very **promising** trend!

EXPLOITING DPR FOR REAL-TIME APPLICATIONS

A proposal for a **system architecture**

SYSTEM ARCHITECTURE

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



COMPUTATIONAL ACTIVITIES

- Software Tasks (**SW-Tasks**)
  - Periodic (or sporadic) **real-time** tasks running on the **processor**;
- Hardware Tasks (**HW-Tasks**)
  - Functionalities implemented in **programmable logic** and executed on the **FPGA**.

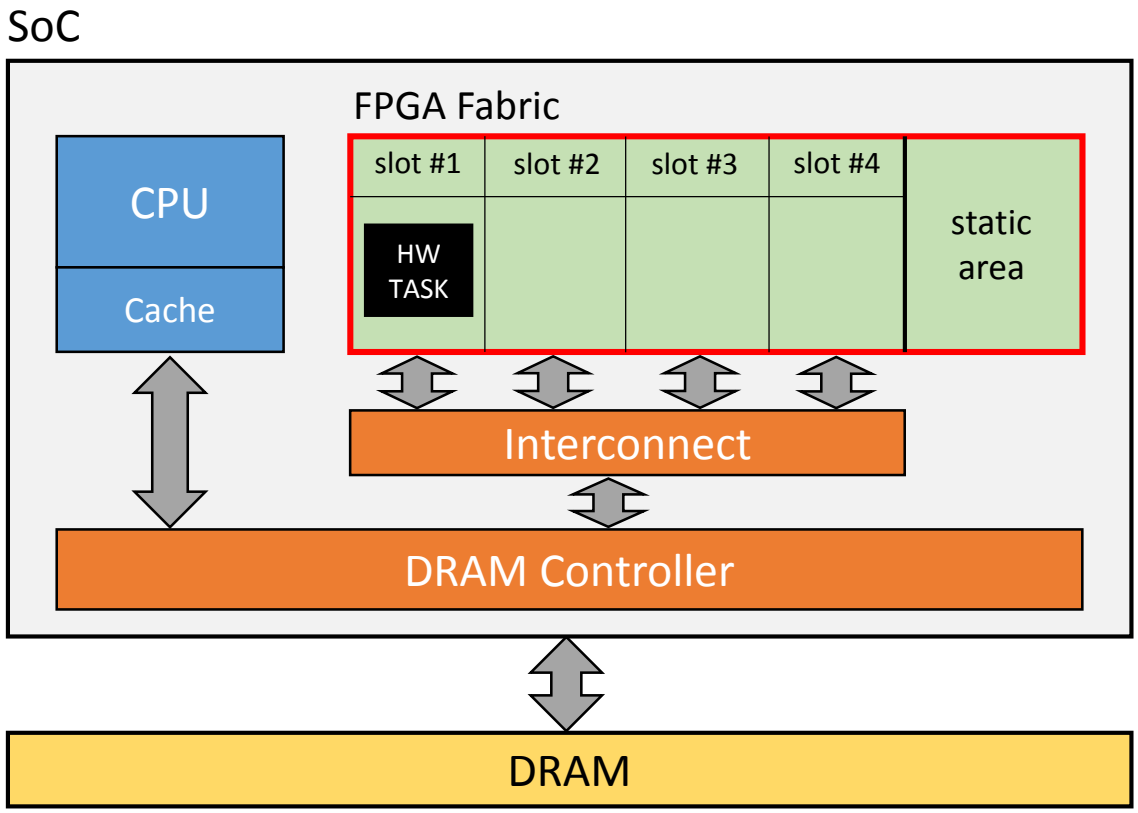
SW-Task

```
while(true)
{
        <…>
        <prepare input data>
        EXECUTE_HW_TASK(my_hw_task);
        <retrieve output data>
        <…>
        suspend_until(next_activation);
}
```

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

SLOTTED APPROACH

- FPGA area is statically partitioned in slots.
- HW-Tasks are programmed onto slots.

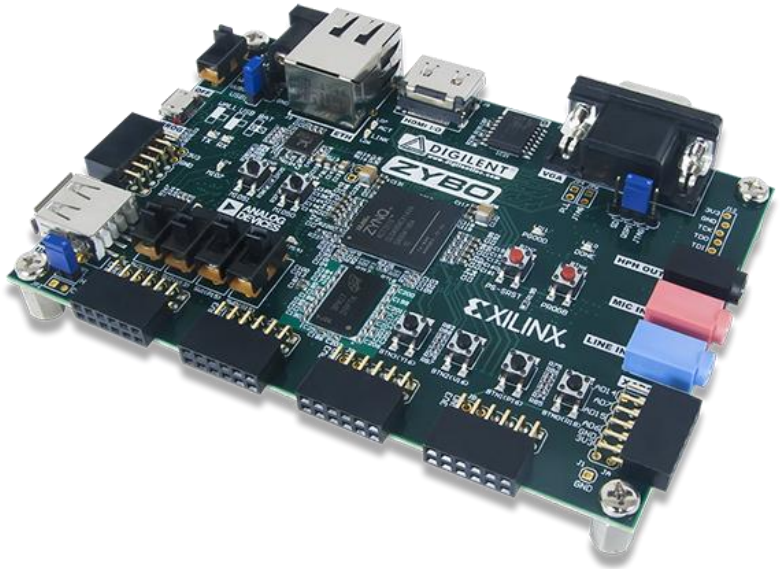


RECONFIGURATION INTERFACE

- **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 a slot **without affecting** HW-tasks that execute on the *other* slots;
  - is an **external device** to the processor (e.g., like a DMA);
  - can program **at most one** slot *at a time*.

Unique resource → **Contention**

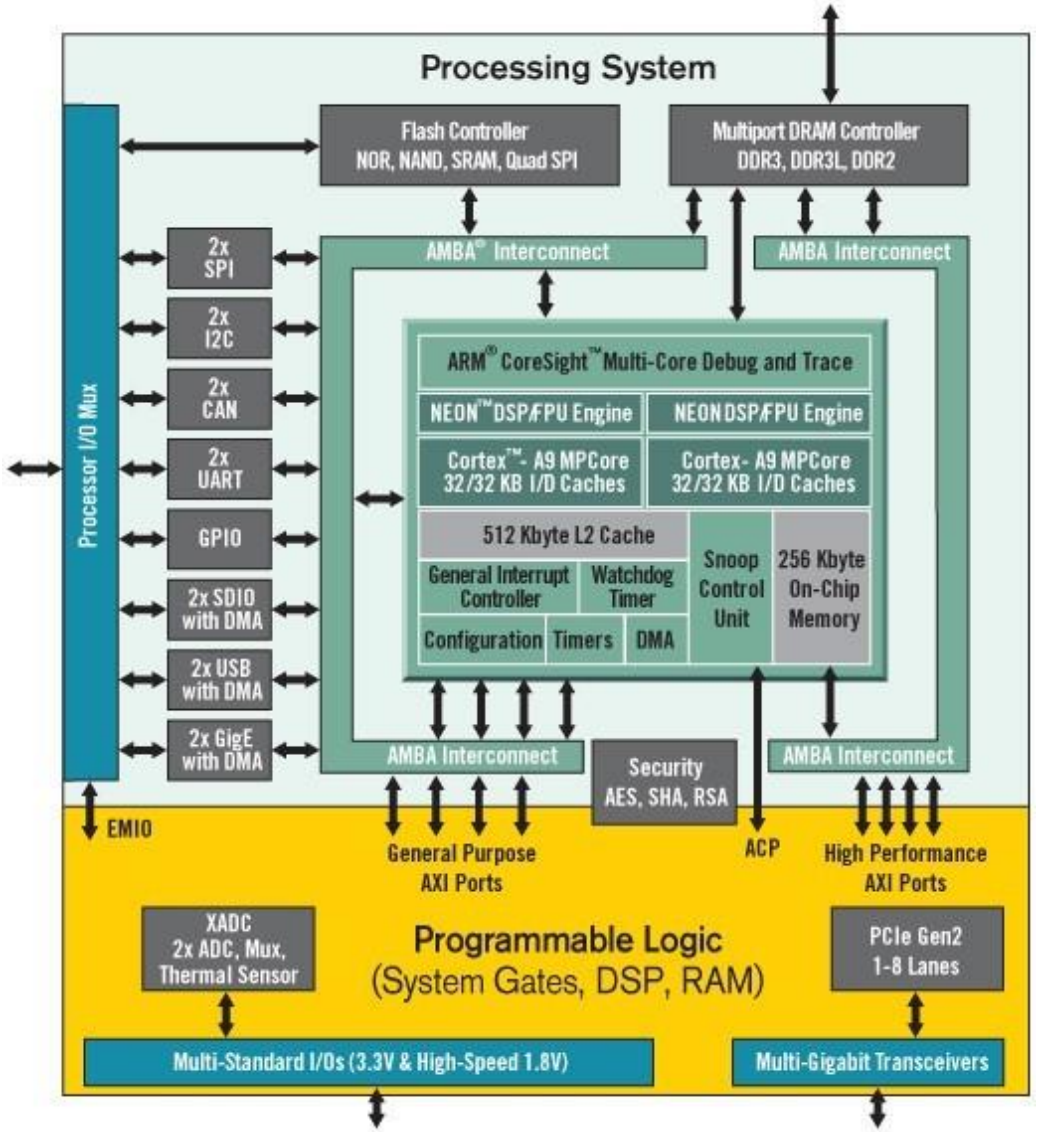
A SYSTEM PROTOTYPE



REFERENCE PLATFORM

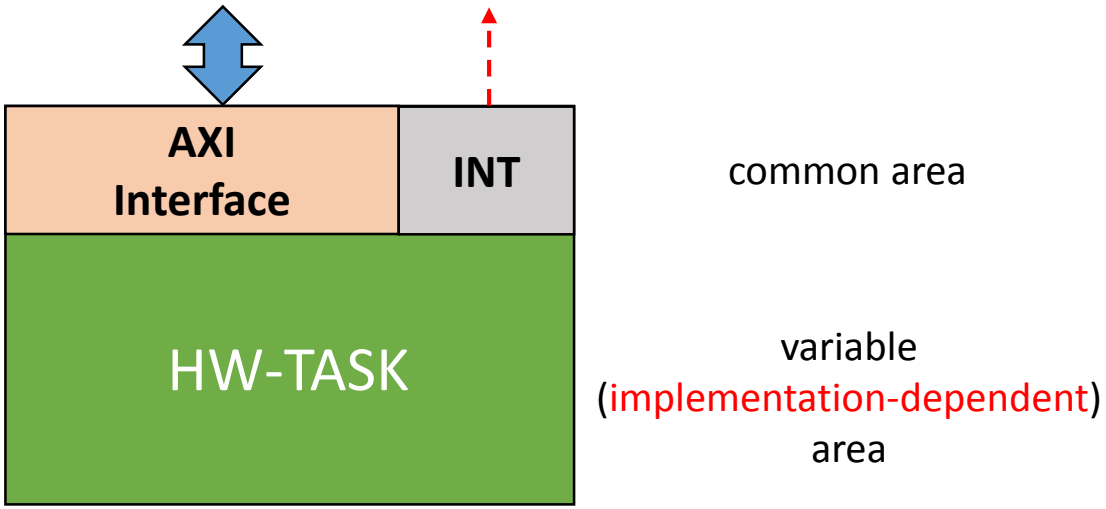
Xilinx Zynq-7000 SoC

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



HW-TASK INTERFACE

- Each **bitstream** has been equipped with a common interface:
  - **AXI Interface** – to access memory (via AMBA Interconnect)
  - **Interrupt signal** – to notify HW-task completion



SOFTWARE SUPPORT

HW-task management implemented as a user-level library in **FreeRTOS.**

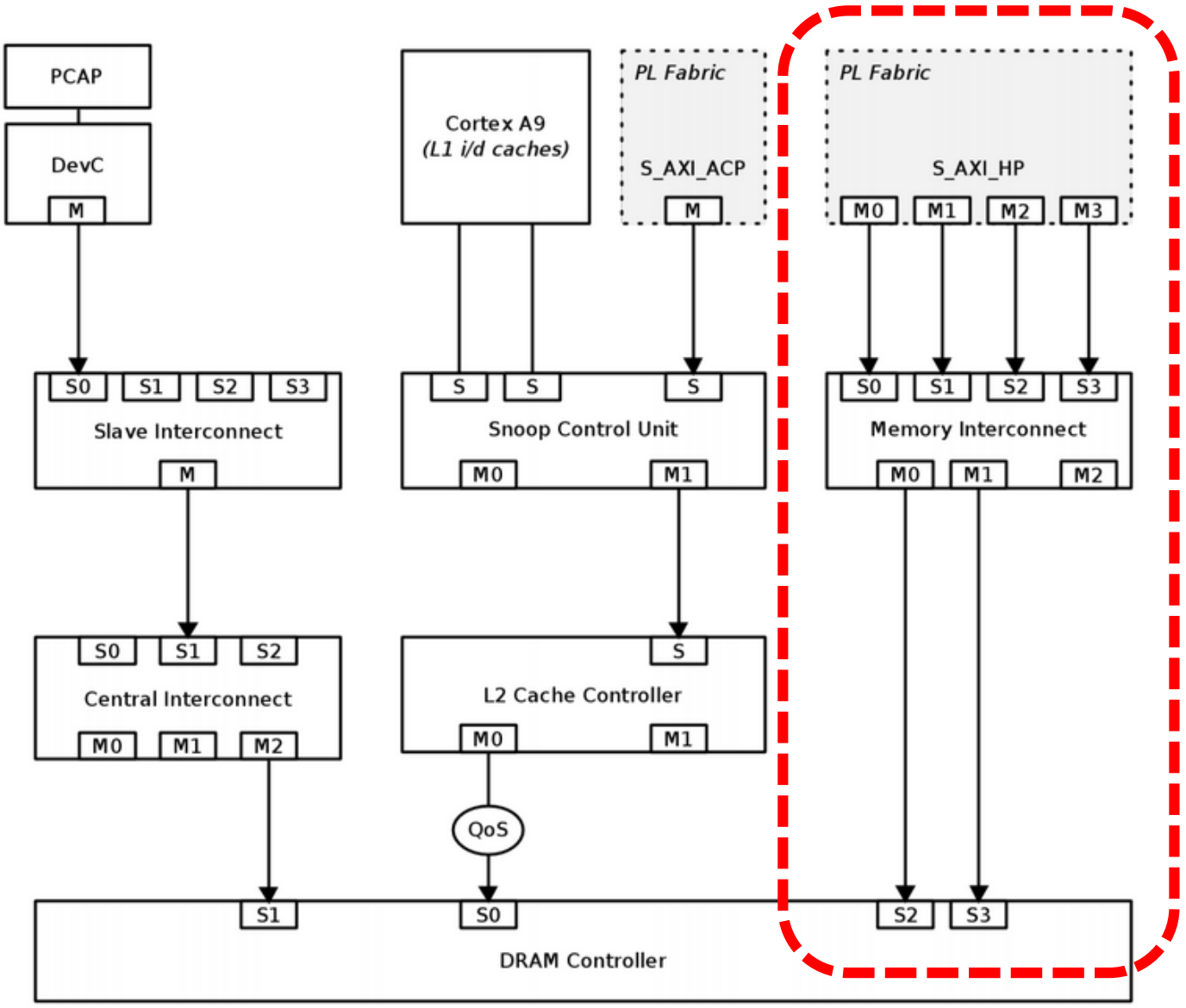
FIFO semaphores have been used to regulate the contention of **slots** and **FRI**

- Counting semaphore for the slots;
- Mutex to protect the FRI.

Cache coherency for input/output data of HW-tasks is **explicitly** handled by the library

- The AXI interfaces in the programmable logic are not subject to cache coherency

SIMPLIFIED ARCHITECTURE



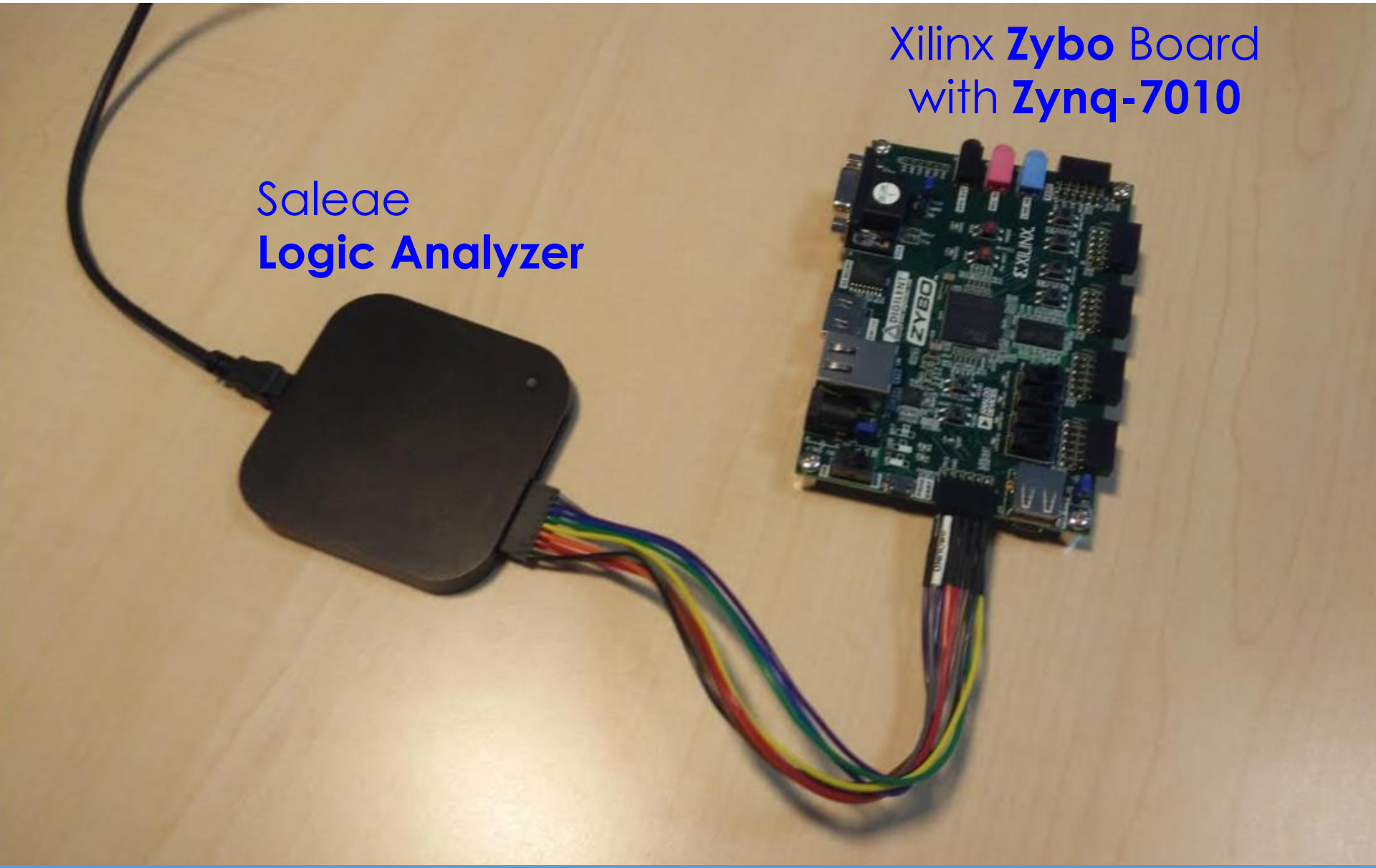
HW-TASK DESCRIPTOR

- **One bitstream per each slot**
  - *bitstream relocation is not supported by the Xilinx tools*
  - *Example of size: 4 slots → 4 x 338 KB*
  - *Bitstreams are preloaded in RAM at the system startup*
- **Two callbacks**
  - Start and completion of the HW-task
- **Input and output parameters**
  - Memory pointers (or data)

EXPERIMENTAL STUDY

Is the use of **DPR** viable for real-time applications?

EXPERIMENTAL SETUP



CASE STUDY

- Four computational activities:
  - **Sobel** image filter;
  - **Sharp** image filter;

  } 800x600 @ 24-bit
  - **Blur** image filter; 640x480 @ 24-bit
  - **Matrix** multiplier. 64x64 elements
- Both HW-task and SW-task versions have been implemented
  - **Xilinx Vivado HLS** synthesis tool for HW-task
  - **C** language for SW-tasks

QUESTIONS

1. **Speed-up** evaluation

   How much is it possible to speed-up the execution with an implementation in **programmable logic**?

2. **Reconfiguration times** profiling

   What is the actual **throughput** for reconfiguring a portion of the FPGA?

3. **Response-time** evaluation

   Besides **reconfiguration times** and the additional **contention**, can DPR provide **benefits** for real-time applications?

SPEED-UP EVALUATION

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

Algorithm		Mult	Sobel	Sharp	Blur
Observed HW execution times	Average [ms]	0.785	12.710	24.631	24.628
	Longest [ms]	0.785	12.712	24.633	24.629
Observed SW execution times	Average [ms]	1.980	115.518	304.975	374.785
	Longest [ms]	2.017	115.521	304.994	374.811
Speedup	Average	2.523	9.089	12.381	15.217
	Minimum	2.515	9.087	12.380	15.216

Up to **15x**

RECONFIGURATION TIMES

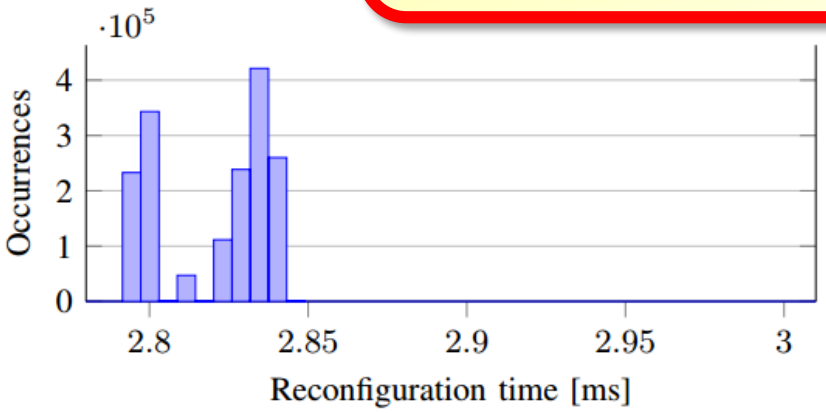
- Time needed to reconfigure a region of ~4K logic cells, **25%** of the total area
- **Memory-intensive** SW activity: data transfer between two 32MB buffers (> L2 cache size)

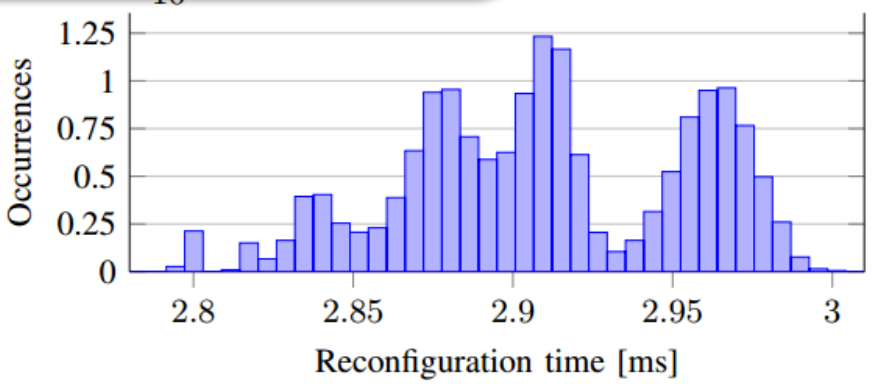
< 3 ms

~110 MB/s

Without me

nterference





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.

With **DPR** seems **feasible**!

SW-task		Mult	Sobel	Sharp	Blur
Period [ms]		30	50	80	100
Cache flush [ms]		0.030	1.123	1.754	1.754
Cache invalidate [ms]		0.017	1.240	1.939	1.939
Observed Response time	Average [ms]	3.829	17.603	31.416	35.624
	Longest [ms]	24.017	20.418	33.086	43.160

THE RESULTS ARE ENCOURAGING!

CONCLUSIONS

- **Experimental study** aimed at evaluating the use of **DPR** for implementing a timesharing mechanism to virtualize the FPGA area.
- We proposed a *possible* **system architecture** and developed a **prototype** as a user-level library on FreeRTOS.

- **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*

BOTTLENECKS

Two major **bottlenecks** have been identified

- The **FRI** is an **unique** resource
  - More FRIs would help in reducing the contention
- The **reconfiguration process** creates **additional contention** on the **bus** (*bitstreams are stored in the main memory*)
  - Dedicated memory for bitstreams would improve performance and predictability

CHALLENGES

a lot…

- Scheduling algorithms for HW-tasks
- Real-time Analysis
  - Task scheduling, Interconnect,…
- Investigation on partitioning approaches for the FPGA
- Improved inter-task communication mechanisms
- **Design and implementation of RTOS support**

Thank you!

Alessandro Biondi
alessandro.biondi@sssup.it