FPGA-Based Multi-Threading for On-board Processing

Pasquale Lombardi - Syderal

Andrea Guerrieri - EPFL

Bilel Belhadj - Syderal

Content

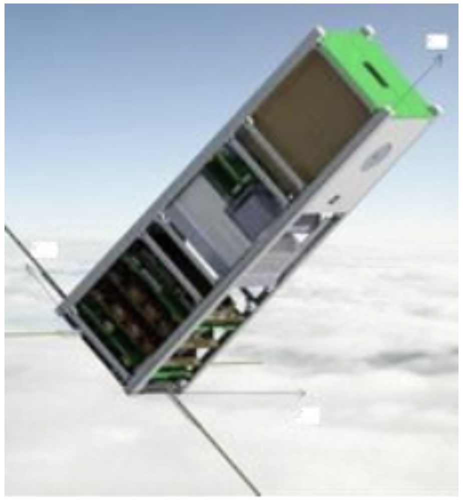
- Spacecraft on-board computing trend and needs
- FPGA-Based Multi-Threading (FBMT)
  - ❑ Introduction
  - ❑ Implementation
  - ❑ Proof of concept test case

SYDERAL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE lap

Spacecraft On-board Computing Trend

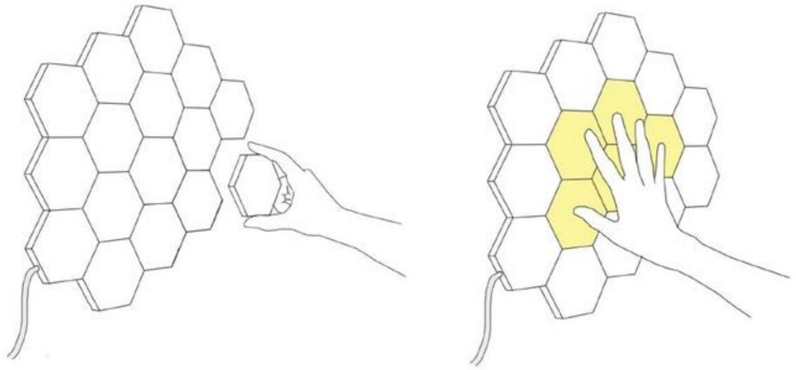
- Big data challenge
  - Increasing data volume
  - Requiring data reduction before downloading
  - On-board selection of useful data

Credit : Attunity.com


Credit : ESA

- Miniaturization allows for smaller satellites
  - Similar observational capabilities
  - Lower energy/power capacity
  - Lower download bandwidth

Spacecraft On-board Key Computing Needs



Credit : Helios Touch

- Scalable and energy efficient

Credit : mytypewriter.com

- Fast adoption of new technologies

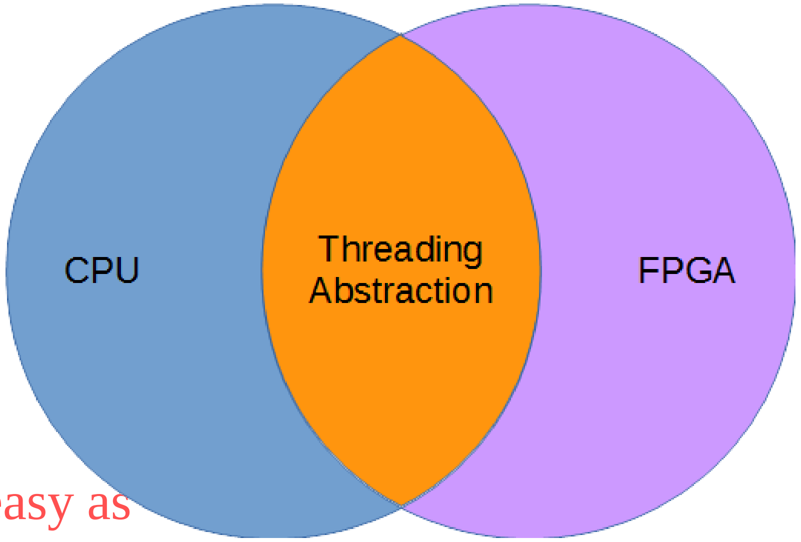
FBMT basic elements and their relevance

- FPGA (Field Programmable Gate Array) is an attractive technology for space missions
  - Unbeatable flexibility
  - Best performance-to-power ratio
- Multi-threading is effectively used in software to
  - Exploit multi-core as well as single-core platforms
  - Improve responsiveness
  - Improve real-time performance
- Heterogeneous computing is a new trend for achieving
  - High performance and
  - Energy efficiency

Credit : ESA

FBMT conceptual basis

- Do heterogeneous computing on a CPU + FPGA platform
  - Increase performance and power efficiency
- Use abstraction to handle threads on FPGA hardware
  - Make FPGA hardware threading as easy as software threading
- Use dynamic partial reconfiguration to allocate FPGA resources to threads
  - Maximize hardware utilization by reusing FPGA resources for different functions

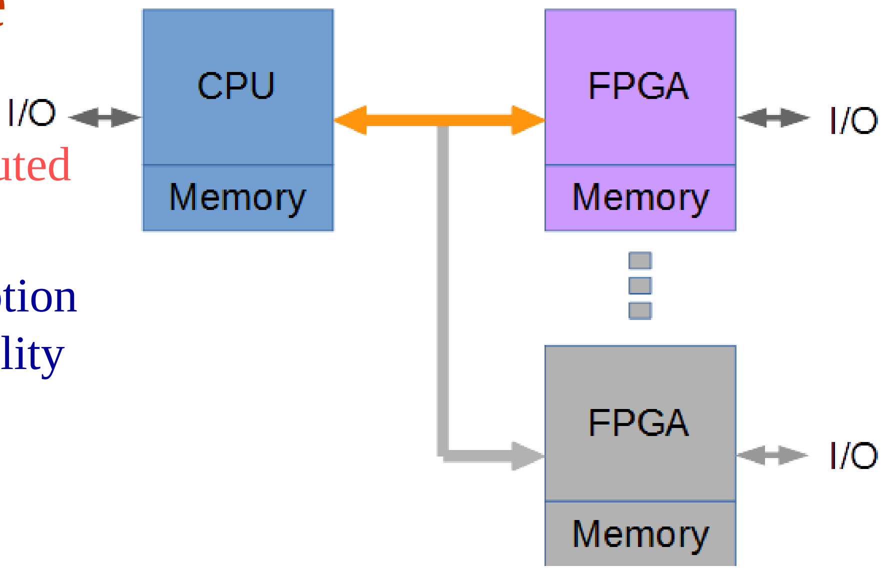


SYDERAL

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

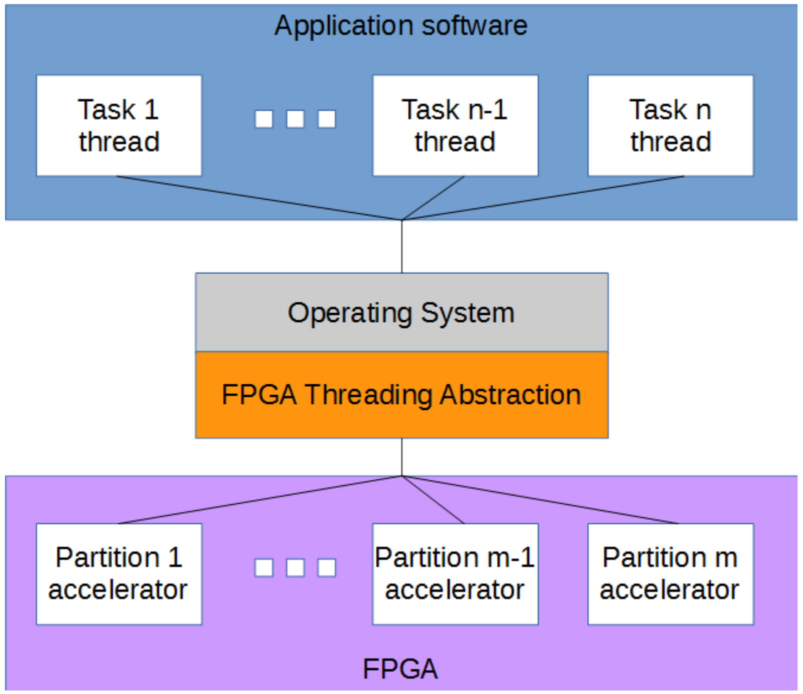
FBMT benefits and architecture

- Does not require a high performance CPU
  - Computational intensive tasks are executed by the FPGA
- Allows the design to be portable against adoption of different technologies and components quality
  - e.g. due to technology obsolescence or mission quality requirements
- Allows for scalability of system performance
  - More than one FPGA can be operated by the same CPU; CPU can be multicore
- Allows for graceful degradation of the system
  - Either CPU or FPGA can be reconfigured to replace any failed functionality in the system even at degraded performance



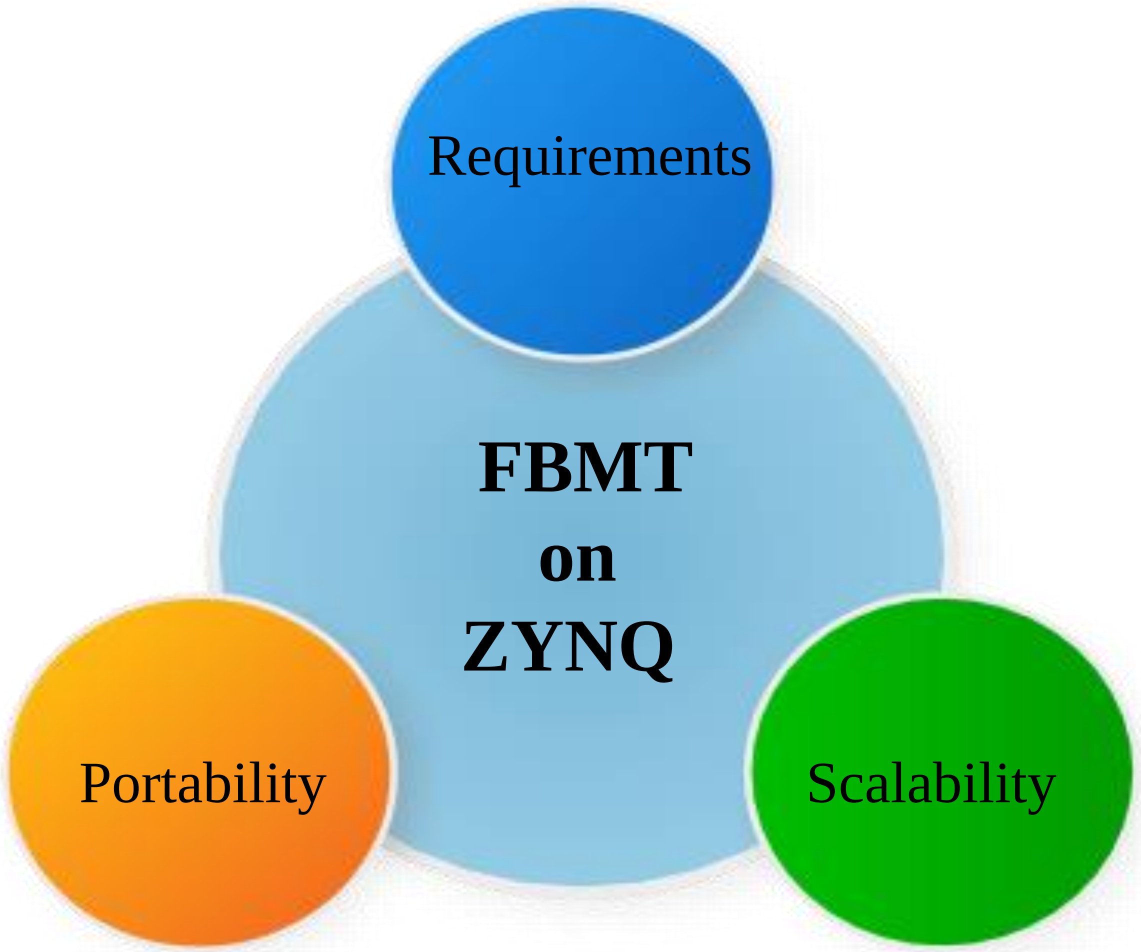
FBMT programming

- Multi-threading vs. sequential programming
  - Improved responsiveness
  - Optimized exploitation of parallel resources
  - Improved performance
- Allows for independent tasks and parallel execution
  - Threaded tasks can execute independently of each other and possibly in parallel
- Threads require operating system support
  - Most operating systems support threads
- FPGA threading abstraction
  - It provides support for FPGA hardware threads as for software threads



IMPLEMENTATION

Architecture



Architectural Overview

Key Ideas

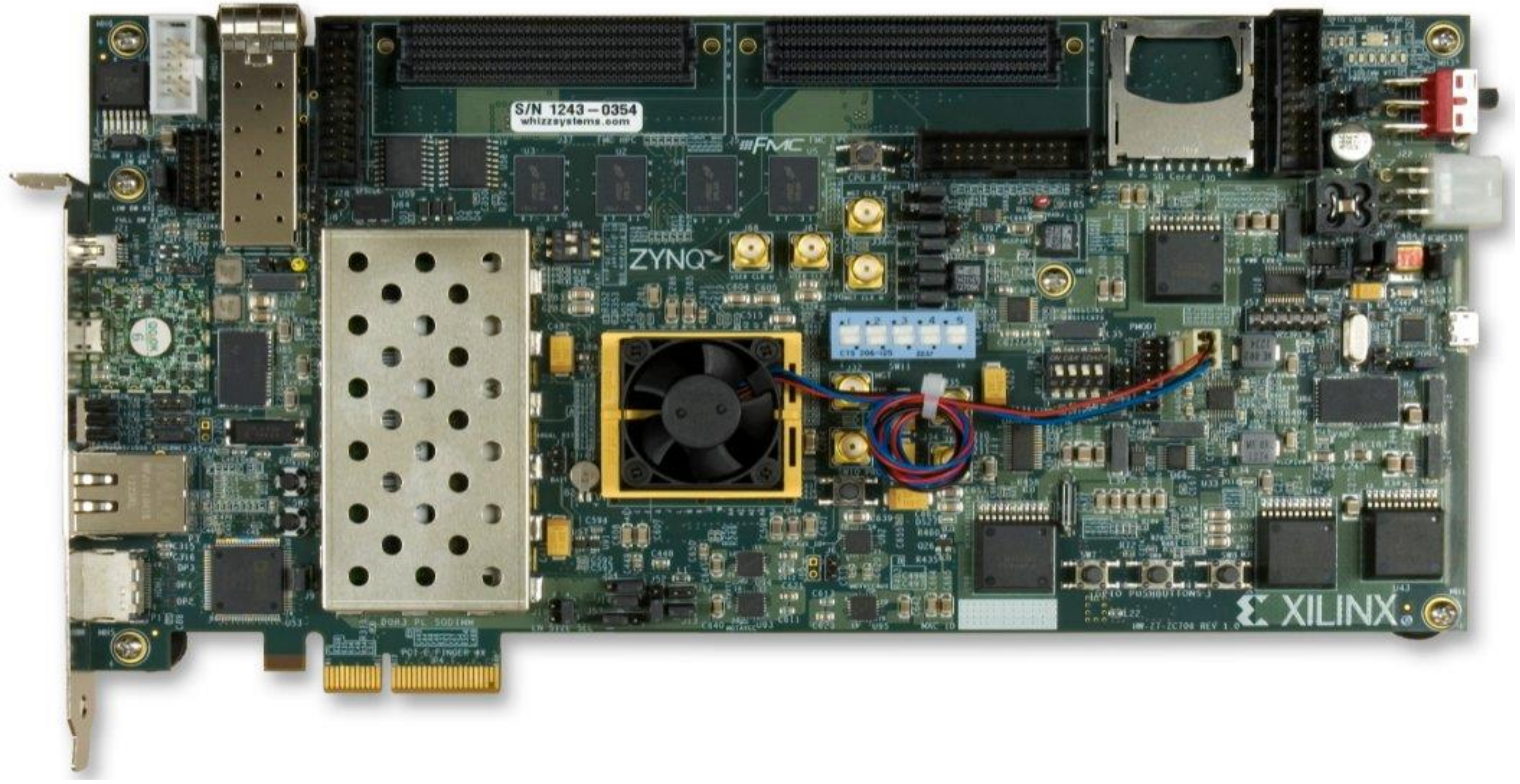
- Manage the FPGA resources by software: Execution Controller (EC) (Flexibility)
- Logical separation between the Host Machine (HM) and EC (Portability and Scalability)
- HM and EC both part of the Processing System (PS) (Hard IPs)
- Modular and scalable FPGA design (Reusability and Power Scalability)

Advantages

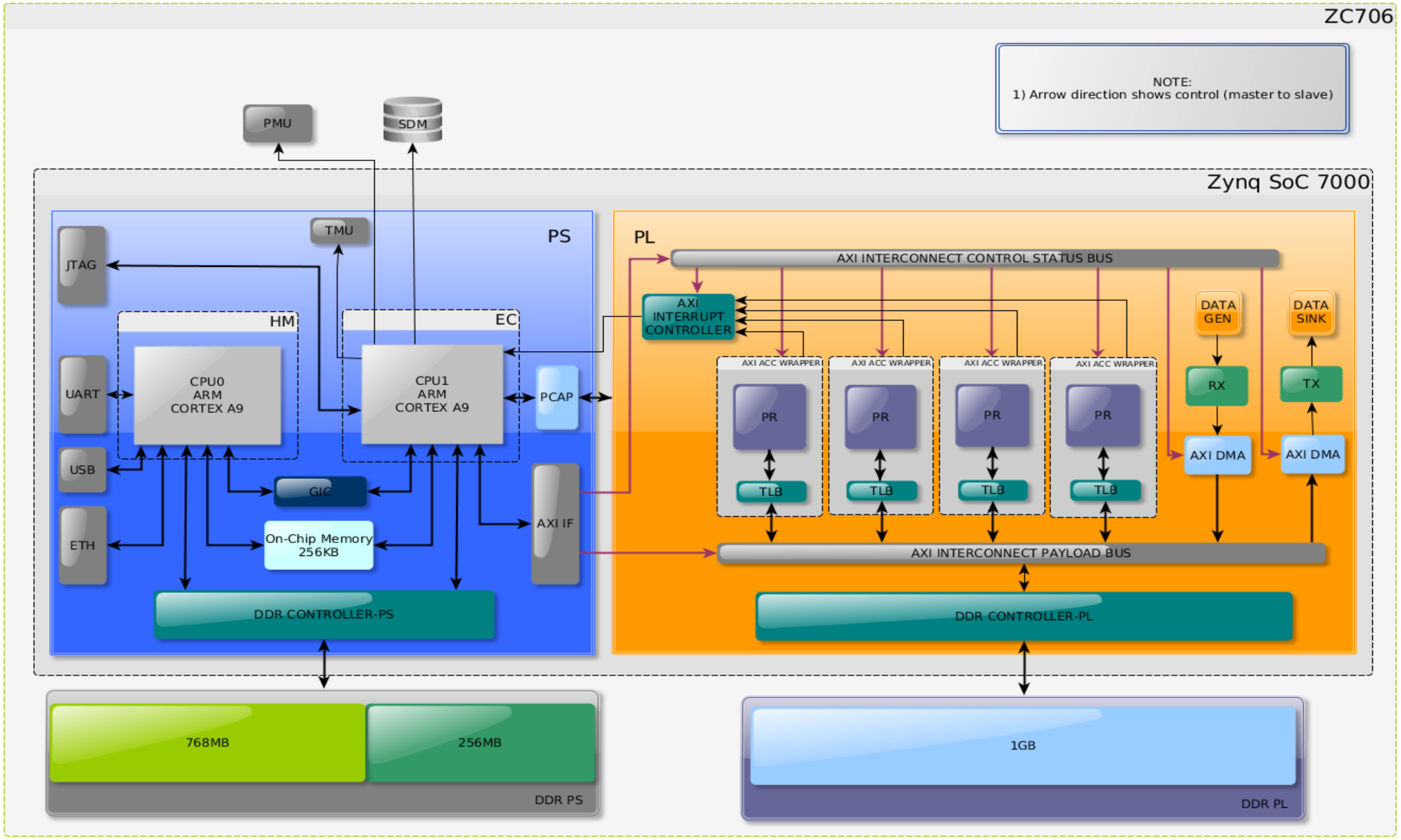
- The HM and EC continue running even during the full FPGA reconfiguration
- No halt or reboot needed to change the size and the number of the PRRs
- Could perform SEE mitigation techniques such as partial and full FPGA scrubbing

Demonstrator

Xilinx ZC706 Development Board



Internal Block Diagram



Execution Controller

Management software running on ARM CortexA9 (CPU1) of Zynq SoC.

Main functions

Executes a sequence of predefined operations for FPGA management:

- Exchange information with HM (asynchronous commands and status)
- Load partial/full bitstreams and configure the PL
- Implements non-preemptive scheduling algorithms
- Monitor power consumption/temperature of the SoC

Advantages

- Written in ANSI C for maximum portability
- High-Performance and Real-time Execution
- Real Concurrency of operation in respect with respect to HM

Host Machine

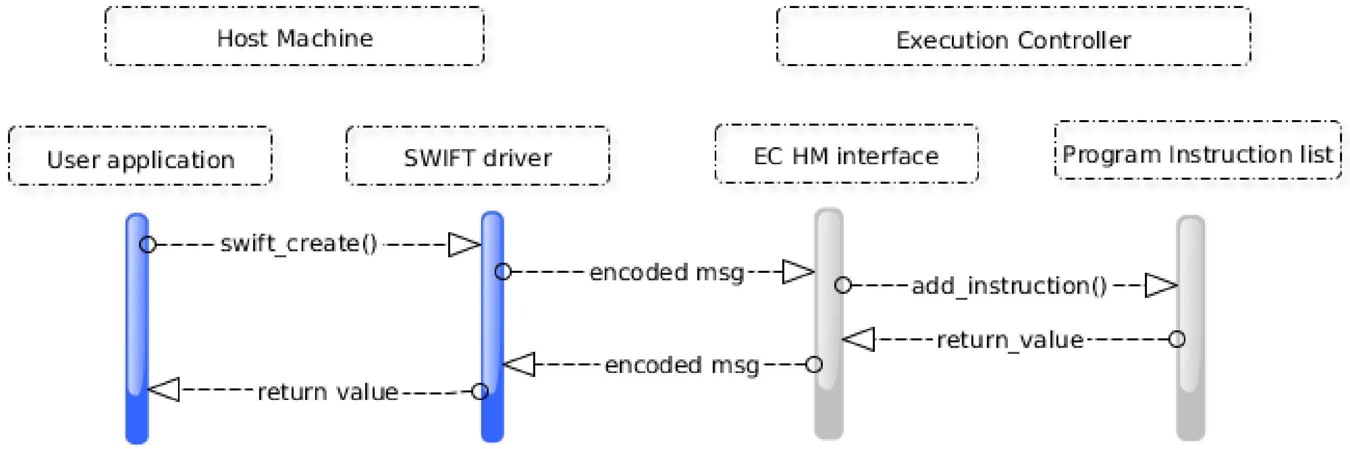
Embedded Linux on ARM CortexA9 (CPU0) of the Zynq SoC

- Application code C/C++
- Manage Hardware Threads using APIs

APIs

Set of predefined functions to control and use the FPGA resources:

- Simple for a software developer familiar with Pthreads
- Hides the complexity and details of FPGA



List of APIs

API	Description
fbmt_initialize ( )	Set the maximum number of concurrent HW Threads
fbmt_create ( )	Create one or more HW Threads
fbmt_join ( )	Wait until the end of one or more HW Threads
fbmt_cancel( )	Terminate the execution of one or more HW Threads
fbmt_set_sched ( )	Select the scheduling algorithm
fbmt_malloc ( )	Allocate virtual memory space for HW Thread
fbmt_free ( )	Free virtual memory space for HW Thread
fbmt_mutex ( )	Mutual Exclusion for HW Thread

APIs Usage Example

Host Source Code

```
#include <iostream>
#include "fbmt_APIs.h"

void main ( int argc, char *argv[ ] )
{

   std::cout << «FBMT APPLICATION  TEST 7.1.7 \n\r";

   fbmt_initialize ( ONE_HW_THREAD_MAX );


   hw_thread[HW_THREAD_0].id = 1;
   hw_thread[HW_THREAD_0].type = CLOUD_DETECTION;


   fbmt_create ( 1, &hw_thread[HW_THREAD_0] );

   fbmt_join ( 1, &hw_thread[HW_THREAD_0] );

   return;
}
```

FBMT APIs

INIT FPGA

INIT SW STRUCTURE

INSTANTIATE

WAIT

SYDERAL

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

lap

Performance

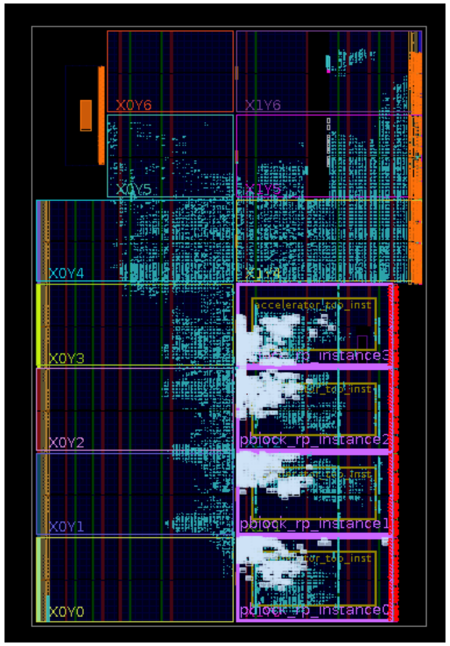
APIs Performance

Description	API	Latency [ms]
HOST-FPGA Command/Response	fbmt_create( )	1
HW Thread Instantiation	fbmt_create ( )	60*
FPGA Initialization	fbmt_initialize ( )	700*

*Full Bitstream/Partial Bitstream Configuration: latency dependent on the size of FPGA/Hardware Thread

FPGA Configuration Throughput:
210 Mib/s

SoC Layout



- C++(APIs)/ C(EC)
- VHDL (no Vendor IPs)
- 200MHz

Platform's Main Features

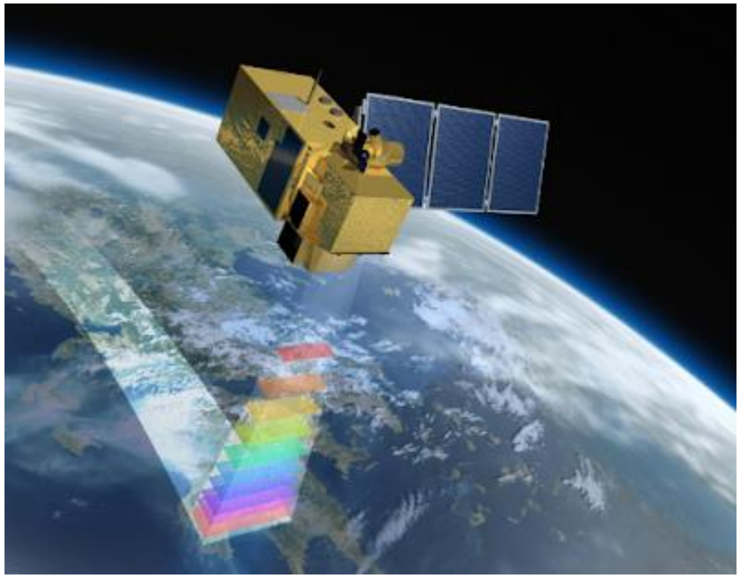
- Continues to operate during the full FPGA reconfiguration
- Has the capability to change the size and the number of concurrent Hardware Threads at run-time
- Manages the Hardware Threads using similar scheduling algorithms used for software threads;
- Provide virtual memory Support to Hardware Threads
- Monitors power consumption and temperature of the SoC and adapts the scheduling accordingly;
- Could autonomously perform SEE mitigation techniques such as full and partial FPGA scrubbing;
- Source code fully portable and scalable over different platforms

PROOF OF CONCEPT TEST CASE

Cloud Detection Application

- Motivation
  - On-board processing of multi-spectral images
  - 66% of cloudy pixels are considered useless data
  - Save storage resources and downlink capacities

- Sentinel-2 multi-spectral images
  - Copernicus Open Access Hub
  - Download S2 Level-1C products
  - Process Scene Classification Algorithm
  - Accelerate execution with FBMT Hardware Threads



- Data volume
  - Real-World, High volume data
  - High memory throughput
  - High processing power

Cloud Detection Algorithm

Input Data

- 9 spectral bands of S2 MSI ~ 250 MB
- Covers 100x100 km of earth surface
- 60 meters resolution per pixel

Output Data

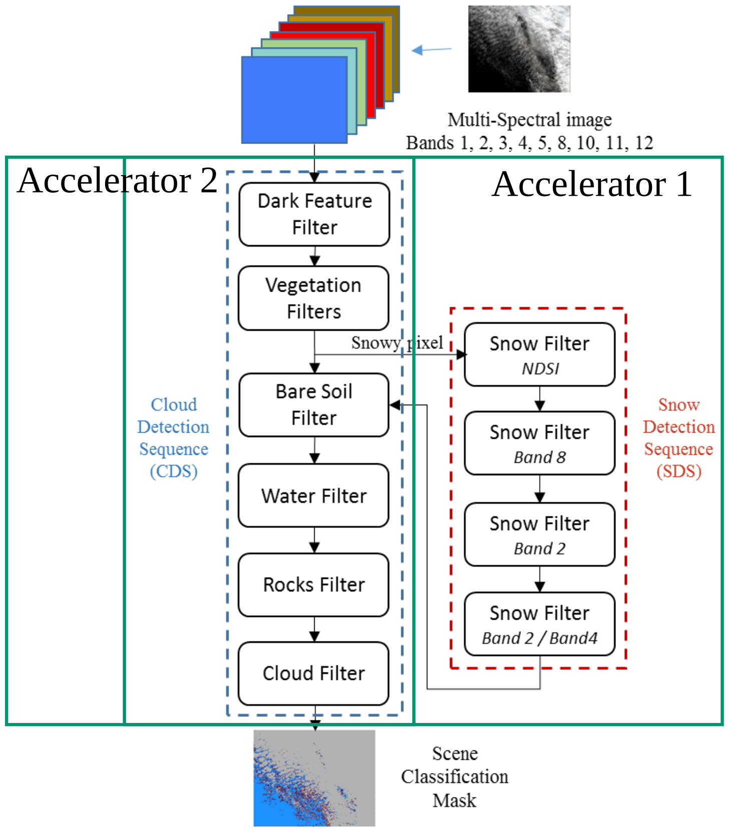
- Scene Classification Mask
- Cloud Mask
- Snow Mask

Threshold based Algorithm

- Compare reflectance to thresholds
- Two main filter sequences
  - Cloud Detection Sequence (CDS)
  - Snow Detection Sequence (SDS)

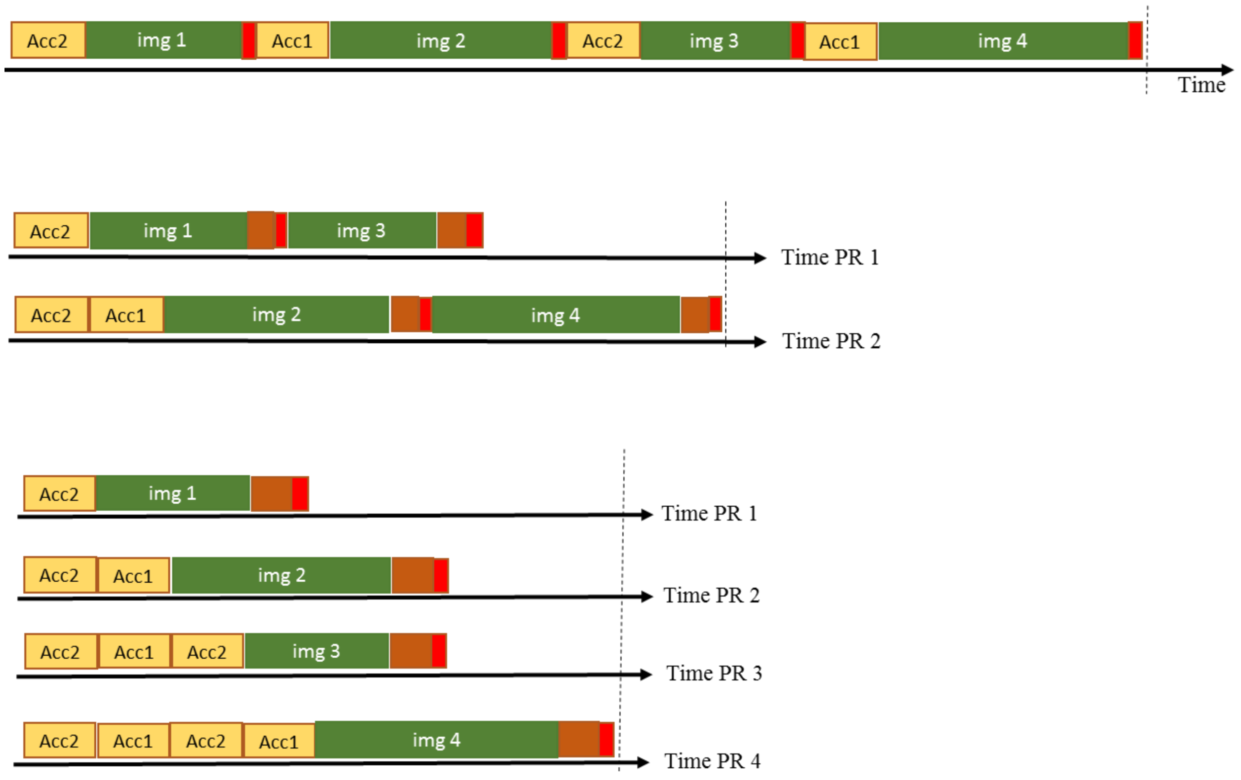
Two accelerator designs

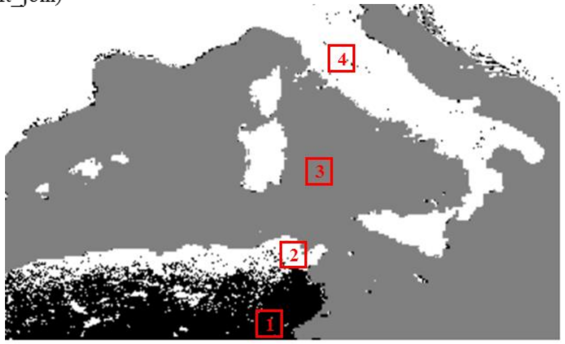
- Accelerator 1 includes CDS and SDS
- Accelerator 2 includes CDS only
- Dynamic reconfiguration of accelerator designs according to image content

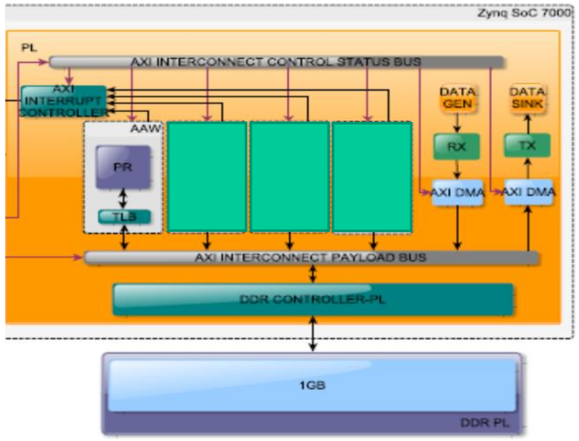


Processing Examples

🟨 Reconfiguration time (swift_create) 🟩 Processing time 🟫 Data routing time 🟥 Host-FPGA comm. (swift_join)

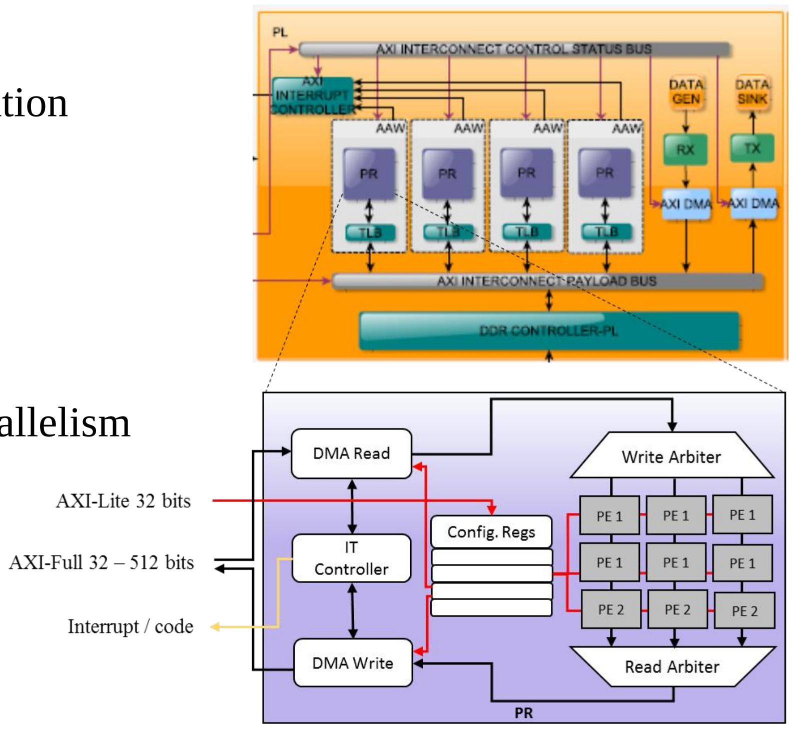






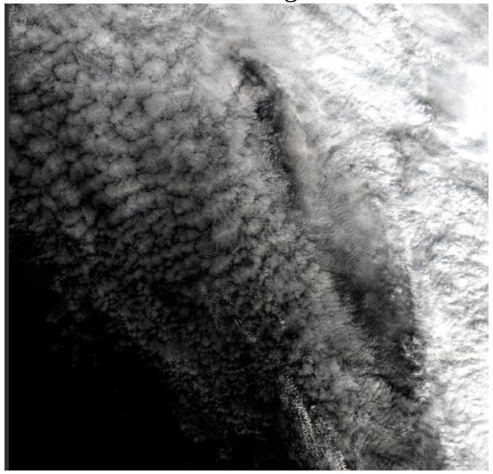
Inside PR regions: Accelerator Architecture

- Common Modules
  - DMA engines, Arbiters, AXI interfaces
  - Internal Buses, Configuration Registers
- Custom Processing Elements (PEs)
  - From simple addition to user-defined computation
  - Could communicate together
  - May be heterogeneous
- Data Triggered Configuration
  - Computation is a side effect of data transfer
  - Provides opportunities for data processing parallelism
- Configurable parameters
  - PE number, DMA size, Register map, …
  - AXI data bus width, AXI burst length

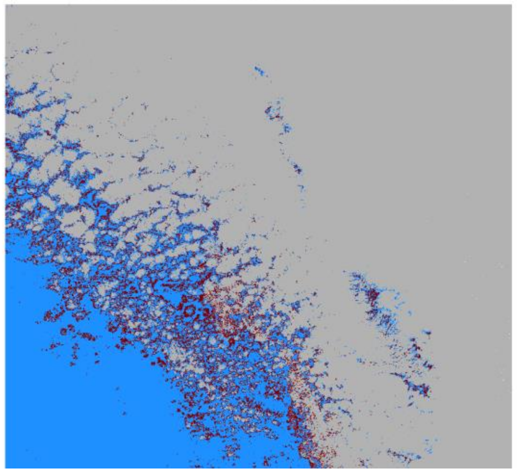


Cloud Detection Example

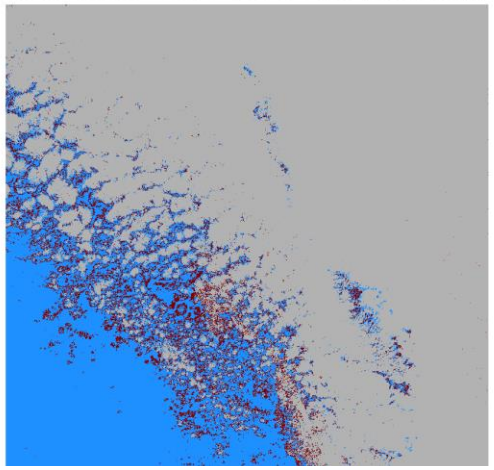
RGB image



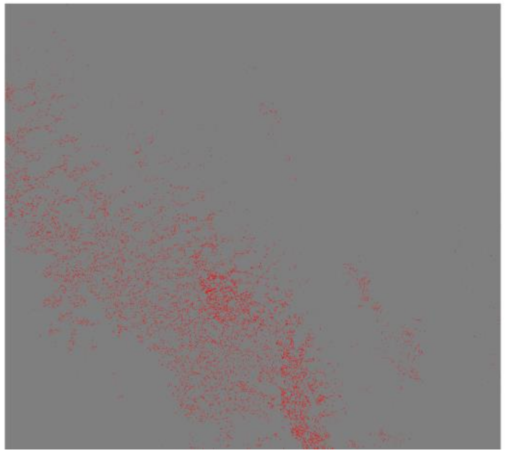
Class Mask -- Software processing

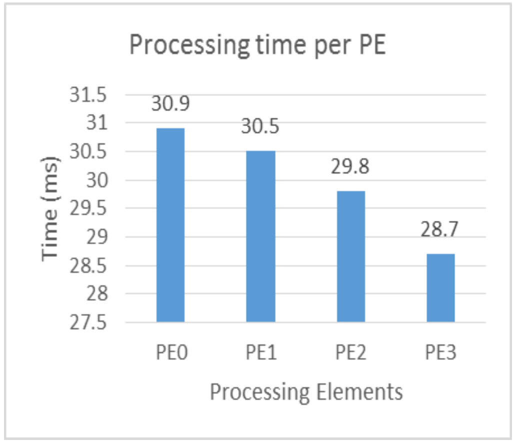


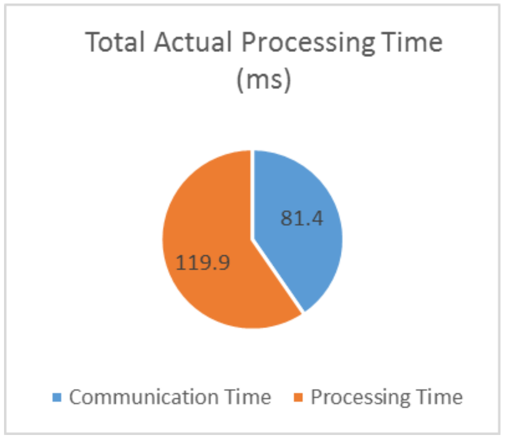
Class Mask -- Hardware threads processing



Class Mask -- Software-Hardware mismatch

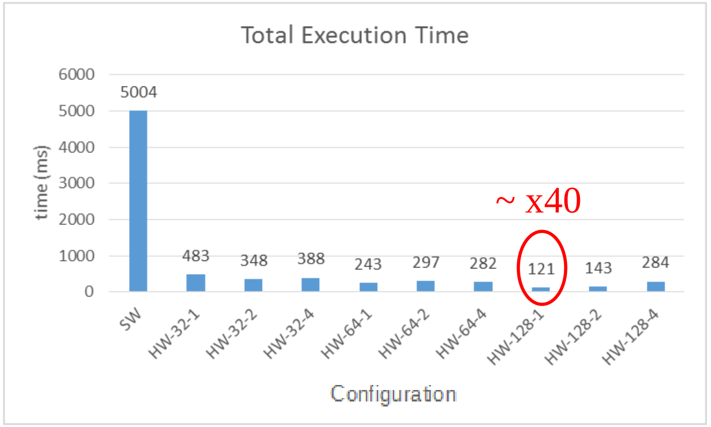


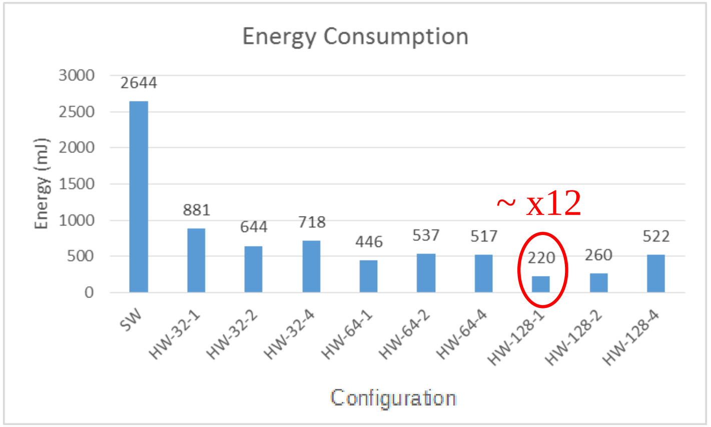




Performance

- Software version
  - ARM Cortex-A9 hard core, 1 GHz
  - Linux OS, 1 GB of RAM
- FBMT resources
  - FPGA utilization (4 PRRs, 32 bits)
    - Static Design ~ 5%
    - Total Design ~ 38%
  - Accelerator - DDR3 throughput
    - Read ~ 2 GB/s (max. 6 GB/s)
    - Write ~ 2.5 GB/s (max. 8 GB/s)





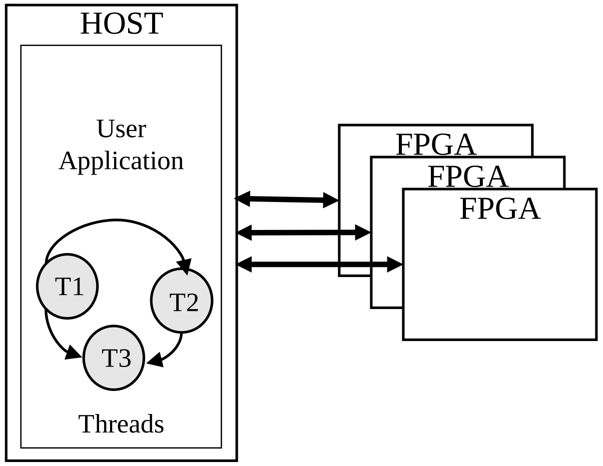
CONCLUSION AND PERSPECTIVE

Conclusion

- FPGA-based multi-threading system for on-board processing
  - Opens opportunities for flexible parallel computing
  - Best performance-to-power ratio
- FBMT demonstrator
  - Implemented on Xilinx Zynq SoC FPGA
  - Leverages dynamic partial reconfiguration technology
  - Full-abstraction of FPGA thread creation and synchronization
  - Portable and industry-friendly design
- Cloud detection test case
  - Proof of FPGA-based multi-threading concepts
  - Processing element based accelerator design
  - Performance
    - x40 faster
    - x12 more energy efficient

Perspectives

- Develop a library of accelerators
  - High-level synthesis
  - Automatic RHBD (Radiation Hardened By Design) design generation
- Scalability Road map
  - Multi-FPGA system
  - Compact backplane technology
  - High-speed links
- FBMT qualification
  - System-level radiation analysis
  - COTS component selection
  - Radiation tests



For further information or feedback

Syderal

Pasquale Lombardi
pasquale.lombardi@syderal.ch

Bilel Belhadj
bilel.belhadj@syderal.ch

EPFL LAP

Andrea Guerrieri
andrea.guerrieri@epfl.ch

Sahand Kashani-Akhavan
sahand.kashani-akhavan@epfl.ch

Paolo Ienne
paolo.ienne@epfl.ch