CPS Summer School 2018
Alghero, Italy
17-21 September, 2018

CERBERO

Self-Adaptivity in heterogeneous CPS platforms

Francesca Palumbo, Università degli Studi di Sassari

Eduardo de la Torre, Universidad Politécnica de Madrid

Horizon 2020
European Union funding
for Research & Innovation

Outline

- Adaptive systems: Concepts & Definition
  - Triggers and types of Adaptation
  - Levels of autonomy. How to build it
  - The Adaptation Loop
  - An example: Evolvable HW
- Adaptive CPS: The CERBERO approach
  - Big Picture. The CERBERO Adaptation Loops at CPS and CPSoS levels
- Deep Dive into CERBERO HW Adaptation
  - ARTICo3
  - MDC-compliant CG adaptation
  - Mixed-Grain Adaptivity
- CERBERO Beyond SoA & Take Out
  - Key Advancements and Integration

Outline

- Adaptive systems: Concepts & Definition
    - Triggers and types of Adaptation
    - Levels of autonomy. How to build it
    - The Adaptation Loop
    - An example: Evolvable HW
- Adaptive CPS: The CERBERO approach
    - Big Picture. The CERBERO Adaptation Loops at CPS and CPSoS levels
- Deep Dive into CERBERO HW Adaptation
    - ARTICo3
    - MDC-compliant CG adaptation
    - Mixed-Grain Adaptivity
- CERBERO Beyond SoA & Take Out
    - Key Advancements and Integration

Concepts

Self-adaptation: ***runtime*** action ***changing structure, functionality and/or parameters of a system***, according to environment, user or self-sensing info.

[F.D. Macías-Escrivá, et al. "*Self-adaptive systems: A survey of current approaches, research challenges and applications*" In Expert Systems with Applications, 2013]

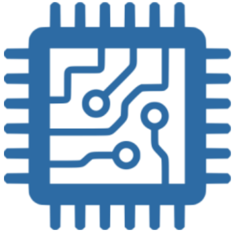
CERBERO

System self-adaptation: combination ***awareness*** and ***reconfiguration***. Reconfiguration decided ***inside the system*** itself by a ***self-adaptation manager***, which has some degrees of freedom when deciding which modifications to apply.

Triggers for Adaptation

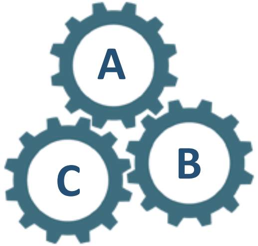
ENVIRONMENTAL AWARENESS: Influence of the environment on the system, i.e. daylight vs. nocturnal, radiation level changes, etc.
Sensors are needed to interact with the environment and capture conditions variations.

USER/EXTERNALLY-COMMANDED: System-User interaction, i.e. user preferences, commands from SoS managers (the boss), etc.
Proper human-machine interfaces are needed to enable interaction and capture commands.



SELF-AWARENESS: The internal status of the system varies while operating and may lead to reconfiguration needs, i.e. chip temperature variation, low battery.
Status monitors are needed to capture the status of the system.

Types of Adaptation



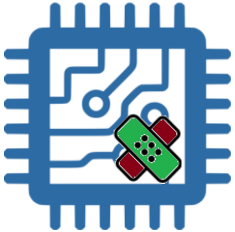
FUNCTIONALITY-ORIENTED:

To adapt functionality because the CPS mission changes, or the data being processed changes and adaptation is required.

It may be parametric (a constant changes) or fully functional (algorithm changes)

EXTRA-FUNCTIONAL REQUIREMENTS-ORIENTED:

Functionality is fixed, but system requires adaptation to accommodate to changing requirements, i.e. execution time or energy consumption.



REPAIR-ORIENTED:

For safety and reliability purposes, adaptation may be used in case of faults. Adaptation may add self-healing or self-repair features. e.g.: HW task migration for permanent faults, or scrubbing (continuous fault verification) and repair.

Autonomous System Adaptation

Autonomic Computing

[Horn 2001; Kephart et al. 2003; March 2004]

Address complexity and adaptation needs of future applications

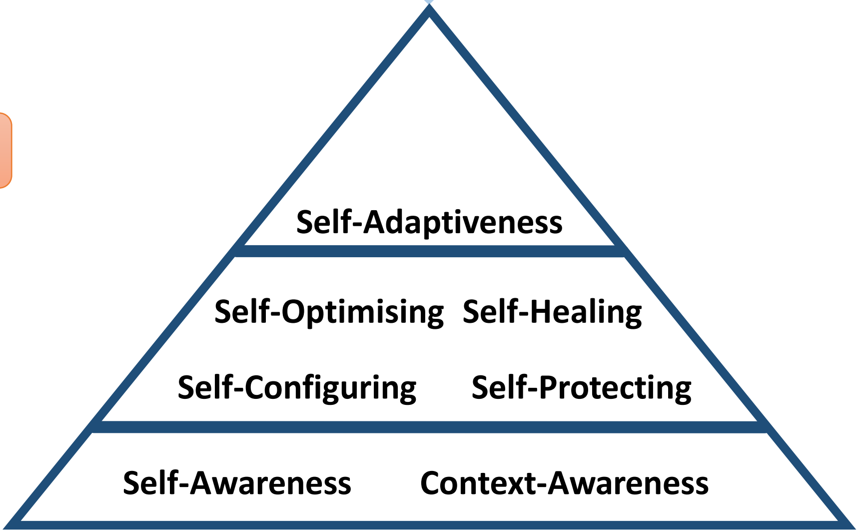
Accomplish high level goals

- **No external control**
- **No human intervention**
- **Runtime decision taking**
- **Optimising behaviour**

Self-adaptive systems

Self-* properties

[Salehie et al. 2009]

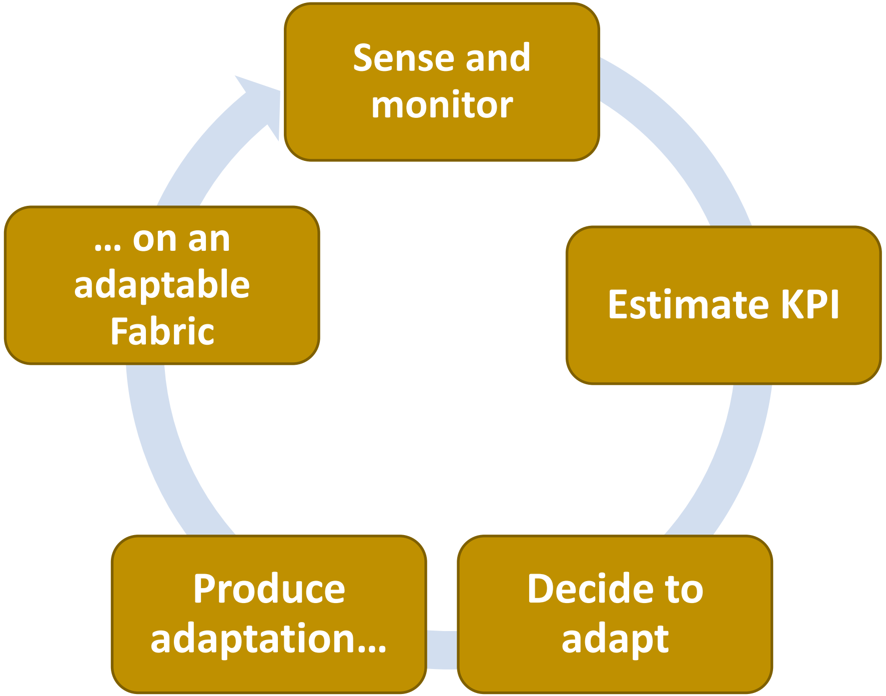


Towards more robust and autonomous systems

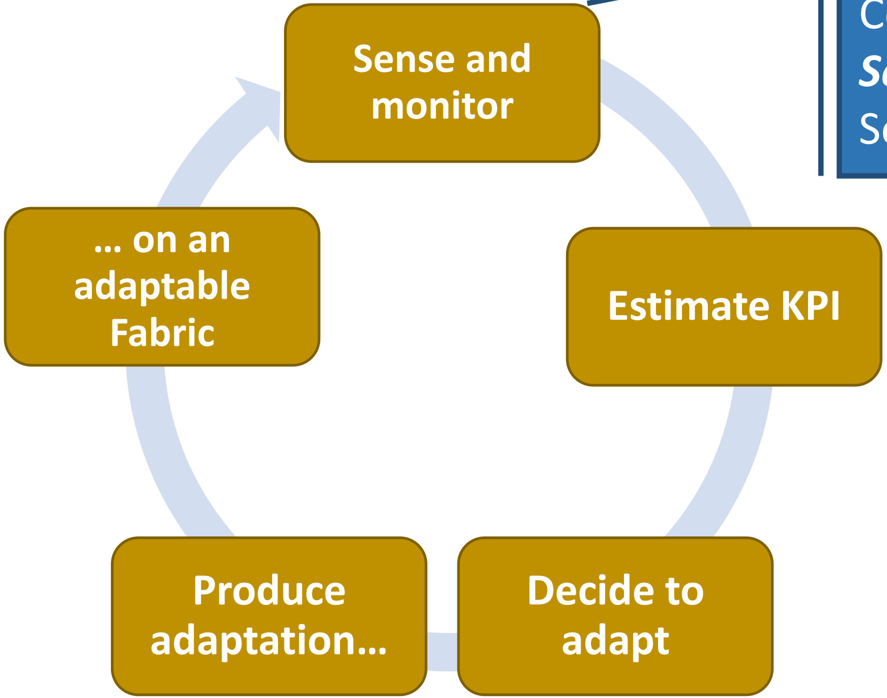
Levels of autonomy		With reconf. Devices, e.g. FPGAs
8	System learns	Evolvable HW
7	System grows	Scalable HW systems
6	System acts	Adaptability, self-healing
5	System searches	Evolutionary algorithms
4	System observes	Power-aware systems
3	Designer provides behaviour	Offline genetic algorithms
2	Designer provides capability	Self-configuration
1	Designer provides configuration	Remote configuration
0	Designer creates system	Off-line design

Source: ‘HW autonomy and space systems’, Steiner & Athanas
IEEE Trans on Automation Control, 2009

Adaptation Loop

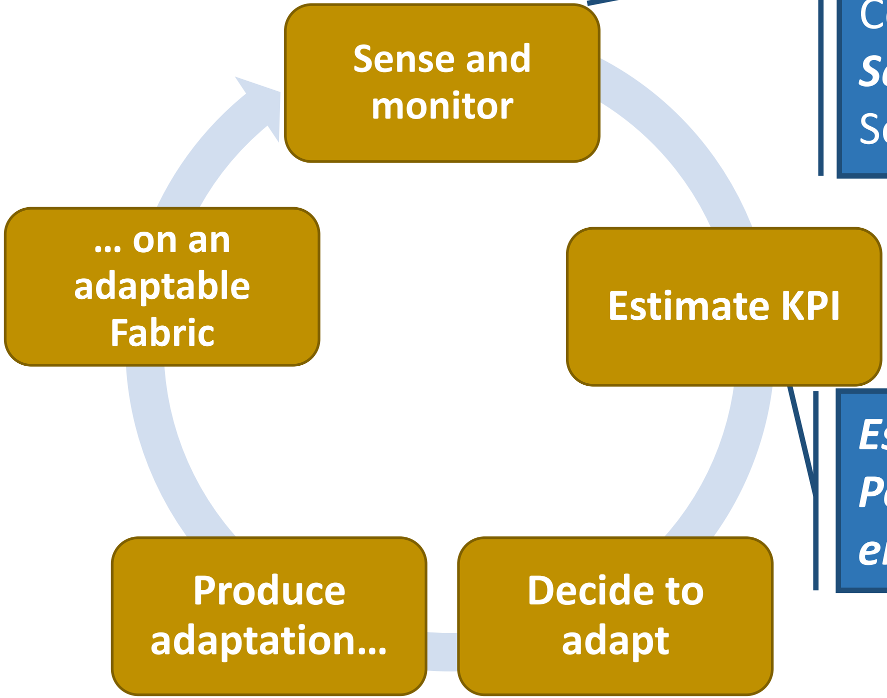


Adaptation Loop



Sense the world (PHY):
Context awareness
Sense the system (CYB):
Self-awareness

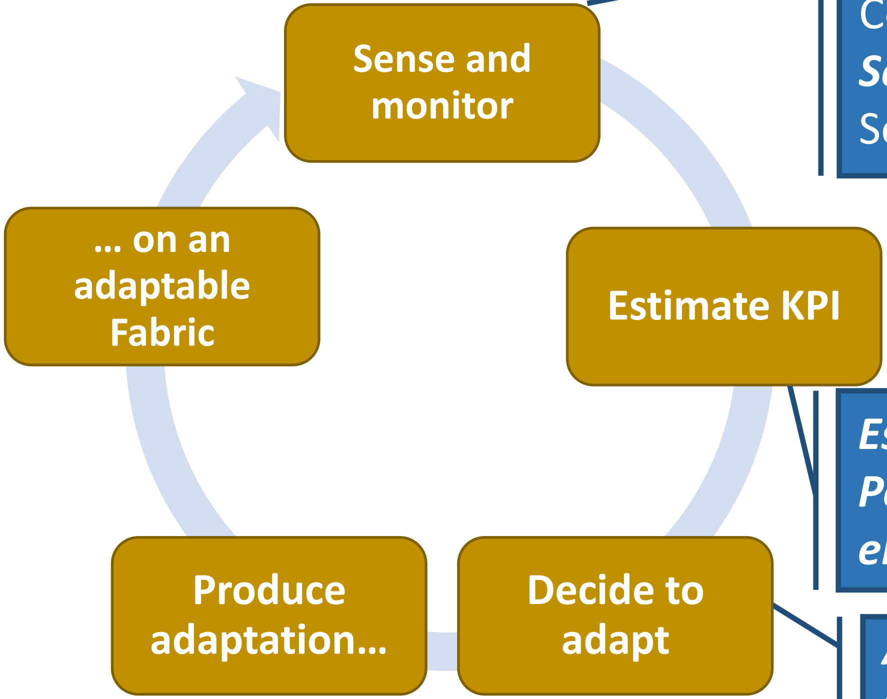
Adaptation Loop



Sense the world (PHY): Context awareness
Sense the system (CYB): Self-awareness

Estimate Performance/quality/ energy: runtime models

Adaptation Loop



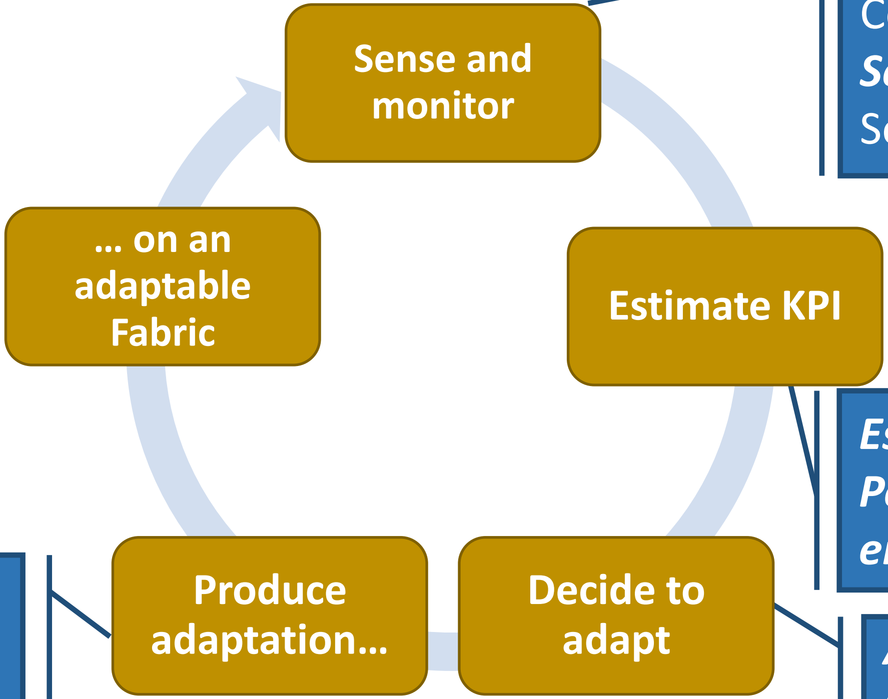
Sense the world (PHY): Context awareness
Sense the system (CYB): Self-awareness

Estimate Performance/quality/ energy: runtime models

Analyse & Decide: Change task – Optimize - Repair

Adaptation Loop

Sense the world (PHY):
Context awareness
Sense the system (CYB):
Self-awareness



Estimate Performance/quality/ energy: runtime models

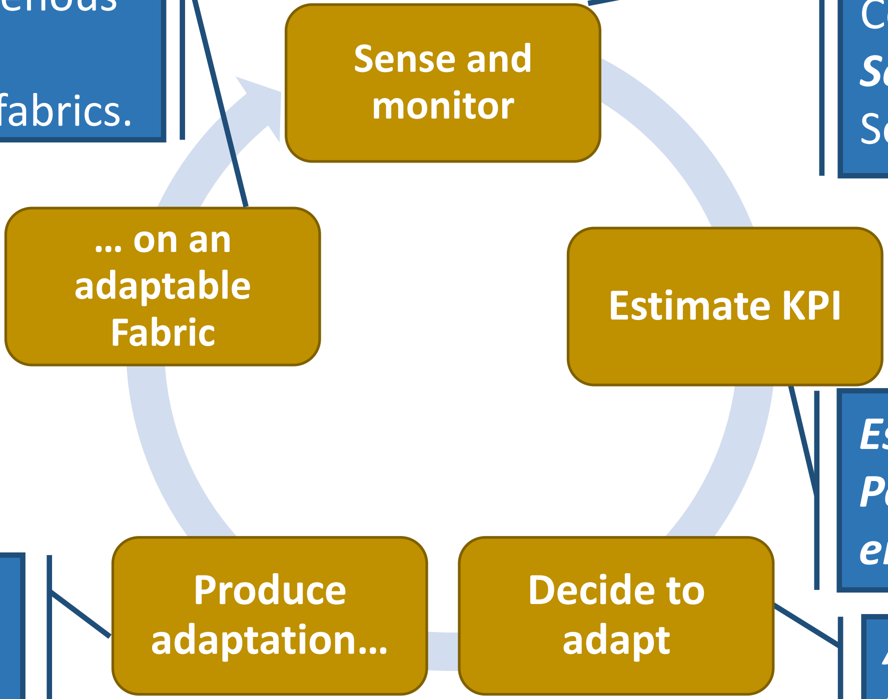
Command Adaptation:
Put in place the actions for the required adaptation

Analyse & Decide:
Change task – Optimize - Repair

Adaptation Loop

Adapt:
Reconfigure the heterogenous (HW-SW) computing infrastructure. Multiple fabrics.

Sense the world (PHY):
Context awareness
Sense the system (CYB):
Self-awareness

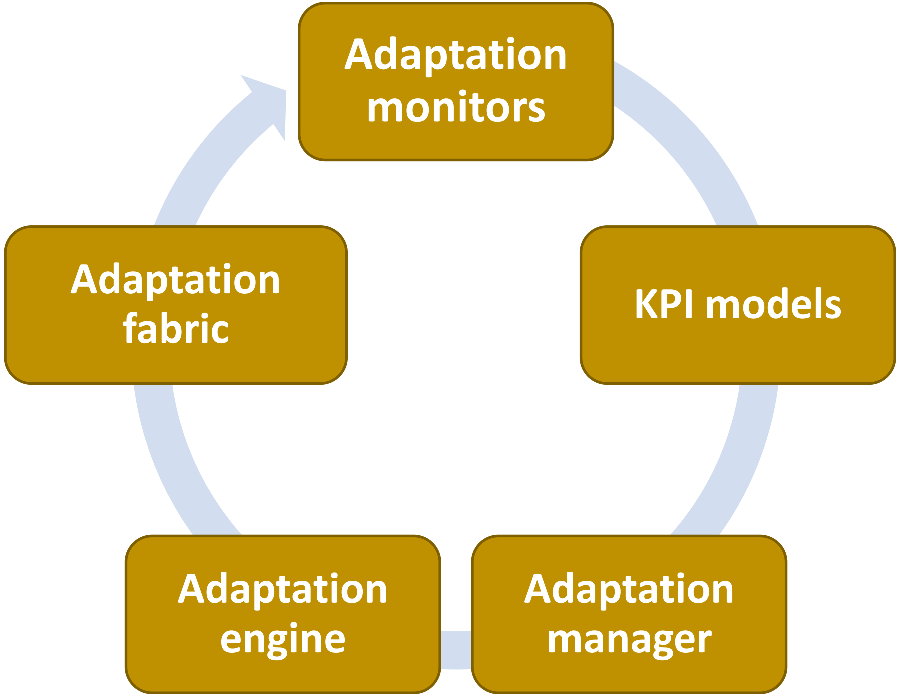


Estimate Performance/quality/energy: runtime models

Analyse & Decide:
Change task – Optimize - Repair

Command Adaptation:
Put in place the actions for the required adaptation

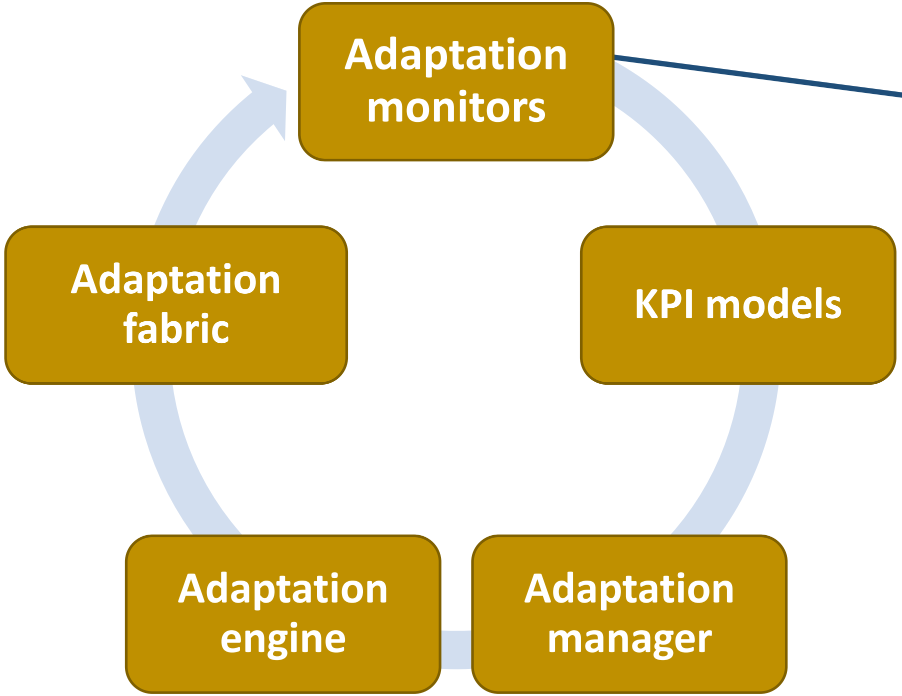
Adaptation Loop: Generalities



Who makes what?

- These components are somewhat present in all type of adaptable systems
- In the CPS domain, they all are contained in the CYBER part.
  - They need to coexist with the mission tasks.
- Adaptation may also require:
  - Predictability (how much does it take)
  - Safety (it cannot die while reconfiguring)
  - Security (secured sensing, secured bitstreams)
  - Real-time constraints (adapting too late can be critical)

Adaptation Loop: Monitors



Context awareness (PHY):

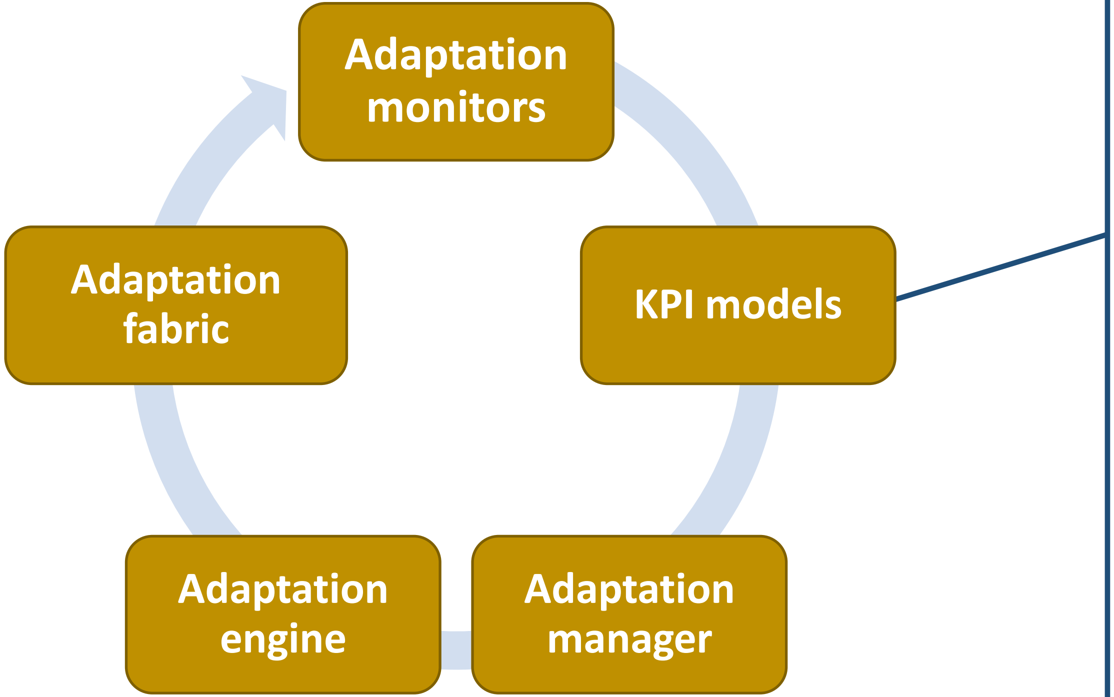
- Sensor fusion for multiple and, possibly, heterogeneous sensors
- Can be at CPS or CPSoS levels

Self-awareness (CYB):

- Performance sensors
- Energy sensors
- Fault detectors

Heterogeneous fabrics require a variety of CYB sensors → homogeneization is required.

Adaptation Loop: KPI Models



Models ***estimate factors*** that might trigger adaptation, i.e.

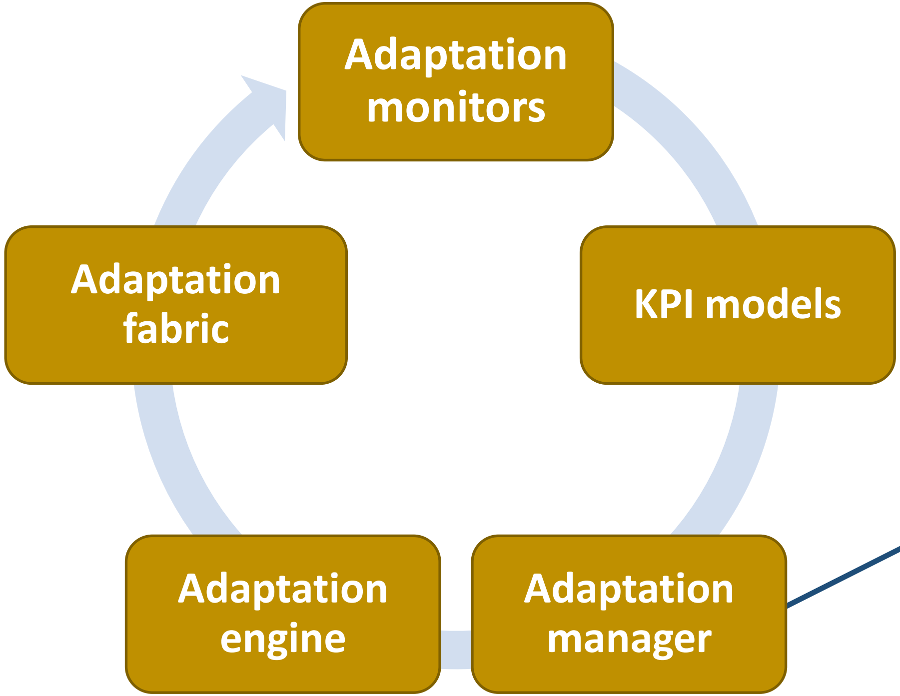
- motor consuming too much power (PHY);
- task going too slow (CYB);
- battery low (CYB).

Lightweight enough to run on the CYB part.

Features:

- CYB models are architecture specific,
- PHY models are application dependent.

Adaptation Loop: Manager



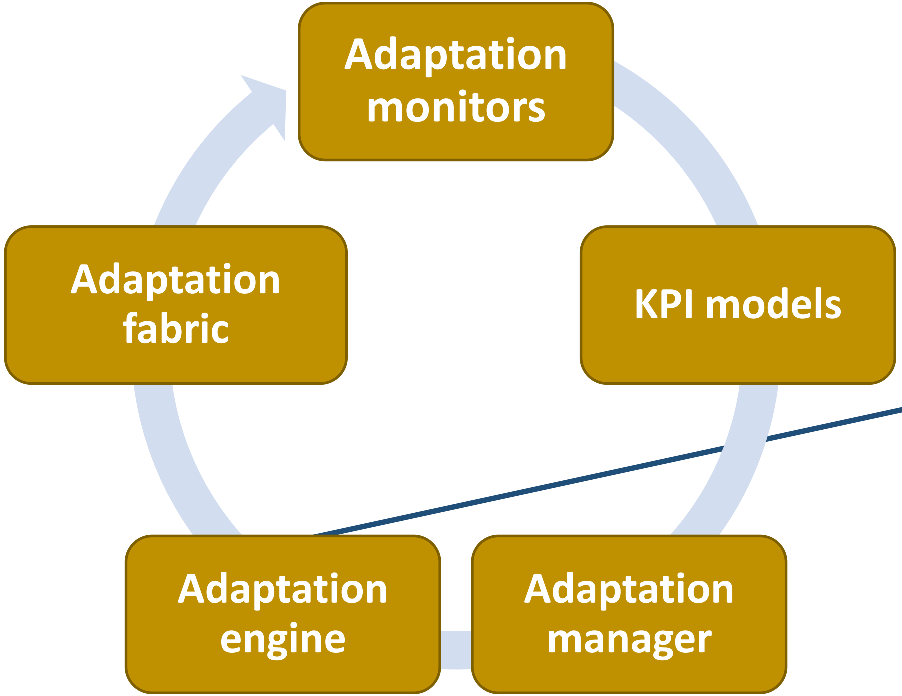
According to ***predefined criteria***, the manager must ***evaluate*** the situation and try to ***optimise*** misbehaving parameters, i.e. Is there a better energy efficient solution?

Optimization problem, solved by different means:

- Non-Linear programming
- Polyhedral approaches
- Genetic algorithms
- Deep learning
- ….

It must be dynamic, with sufficient Dynamic response

Adaptation Loop: Adaptation Engine



Provide means to perform fabric adaptation.

- Fabric-dependent
- Don’t take decisions, they do what the manager states

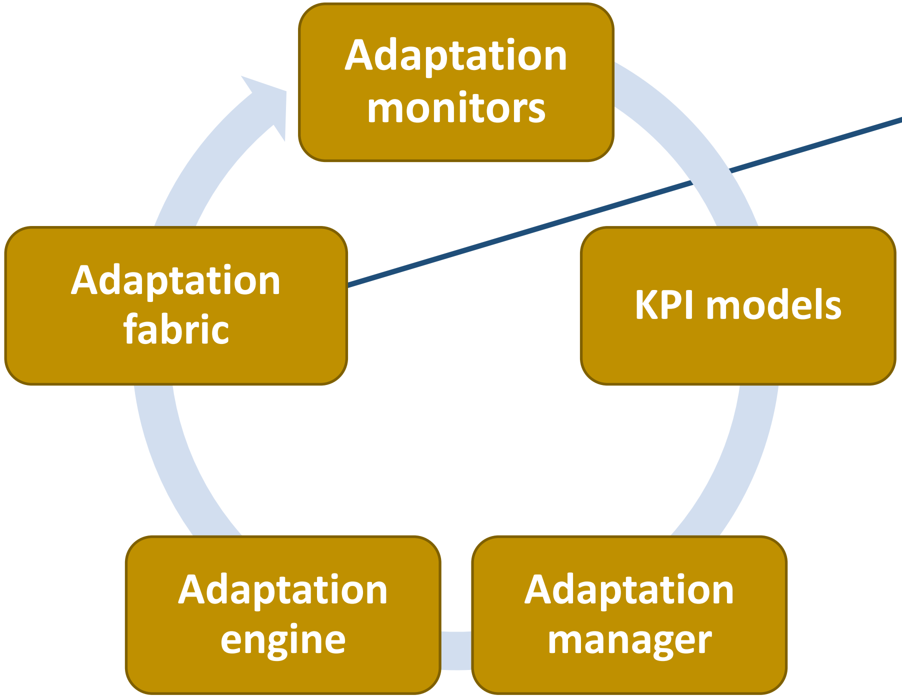
SW:

- Dynamic task assignment;
- Symmetric Shared-memory; Task to core assignment in NoC.

HW:

- Virtual Reconfiguration (VRC);
- Dynamic Partial Reconfiguration (DPR).

Adaptation Loop: Adaptation Fabric



The addressed components must contain sufficient flexibility to allow adaptation.

HW adaptation ***granularity***:

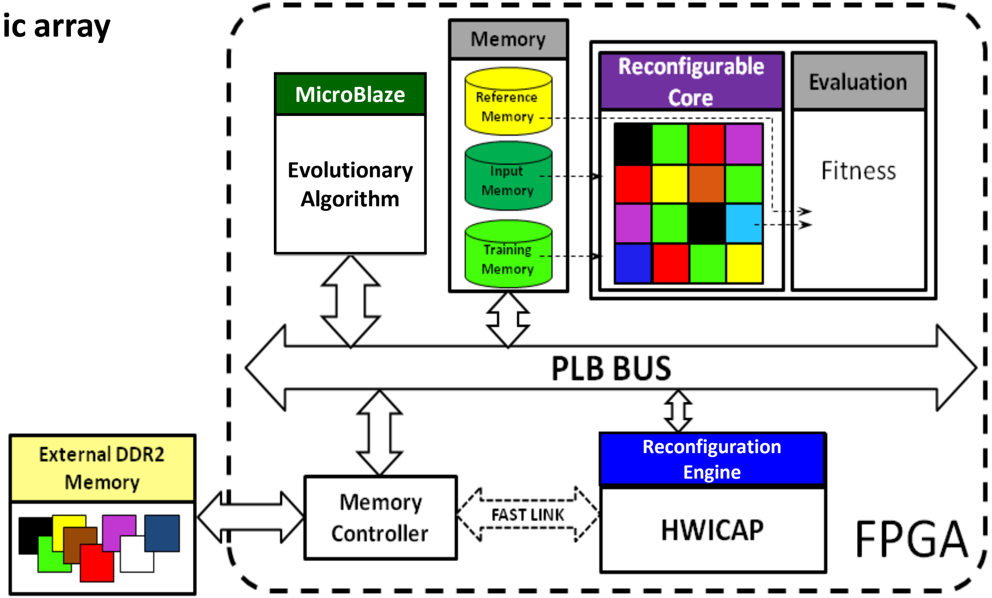
- Tiny elements → fine-grain
- Functions → coarse-grain
- Mixed-grain approaches

HW fabric ***types***:

- DPR on Large regions → slots, Reconfigurable Regions (RRs)
- VRC on large functions → CGRA
- DPR or VRC on small areas of large sections → HW overlays

Example: Evolvable HW

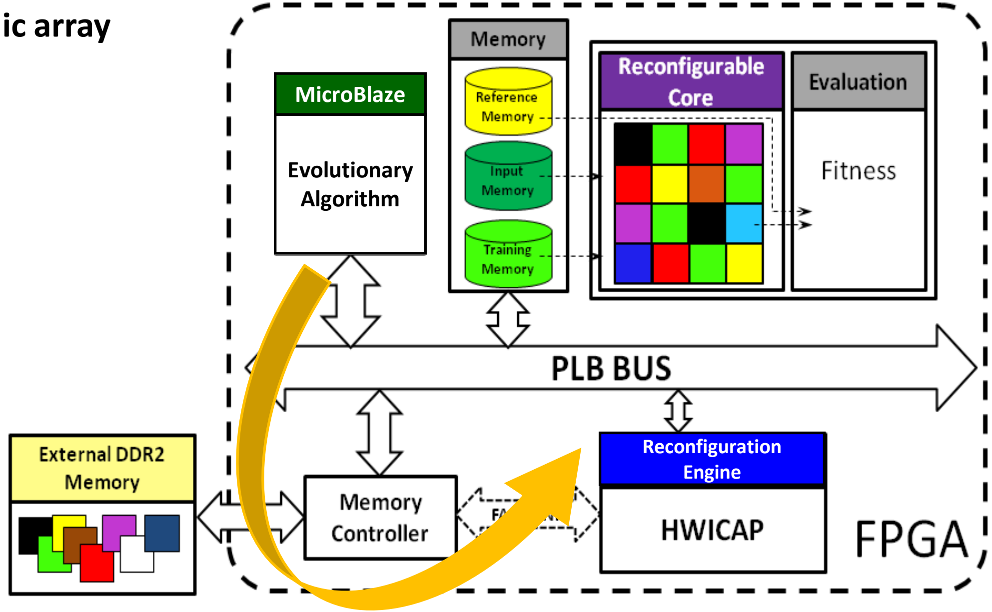
An Evolvable HW System based on a single processing systolic array



A. Otero, R. Salvador, J. Mora, E. de la Torre, T. Riesgo, L. Sekanina; "A fast Reconfigurable 2D HW core architecture on FPGAs for evolvable Self-Adaptive Systems", AHS 2011

Example: Evolvable HW

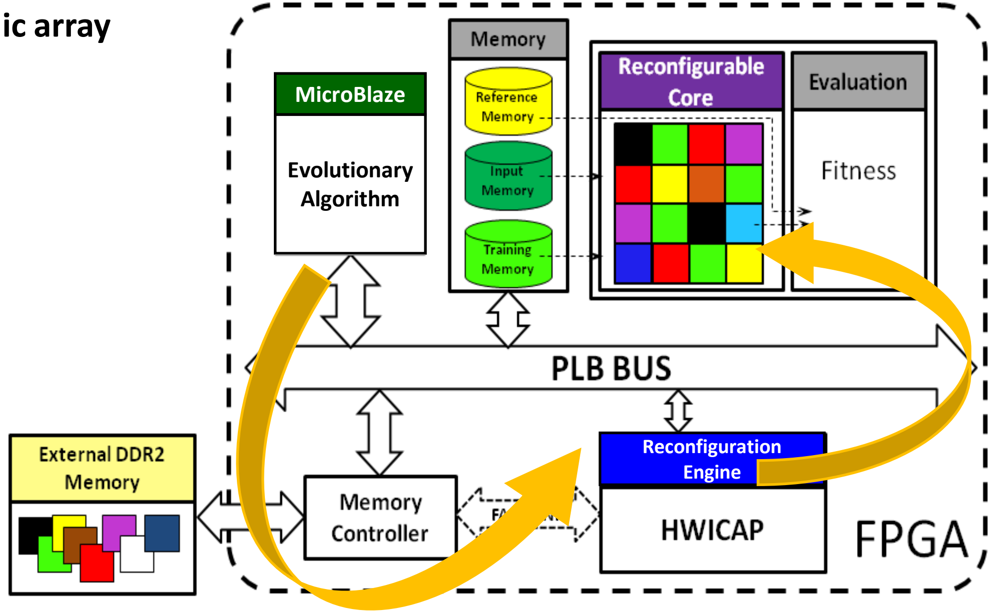
An Evolvable HW System based on a single processing systolic array



A. Otero, R. Salvador, J. Mora, E. de la Torre, T. Riesgo, L. Sekanina; "A fast Reconfigurable 2D HW core architecture on FPGAs for evolvable Self-Adaptive Systems", AHS 2011

Example: Evolvable HW

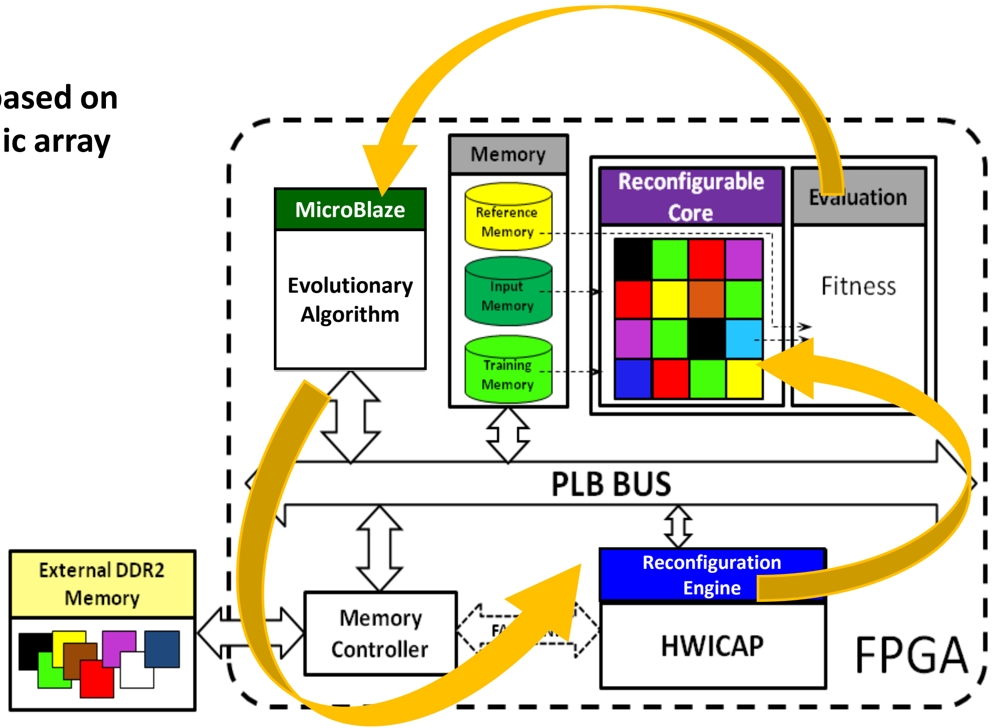
An Evolvable HW System based on a single processing systolic array



A. Otero, R. Salvador, J. Mora, E. de la Torre, T. Riesgo, L. Sekanina; "A fast Reconfigurable 2D HW core architecture on FPGAs for evolvable Self-Adaptive Systems", AHS 2011

Example: Evolvable HW

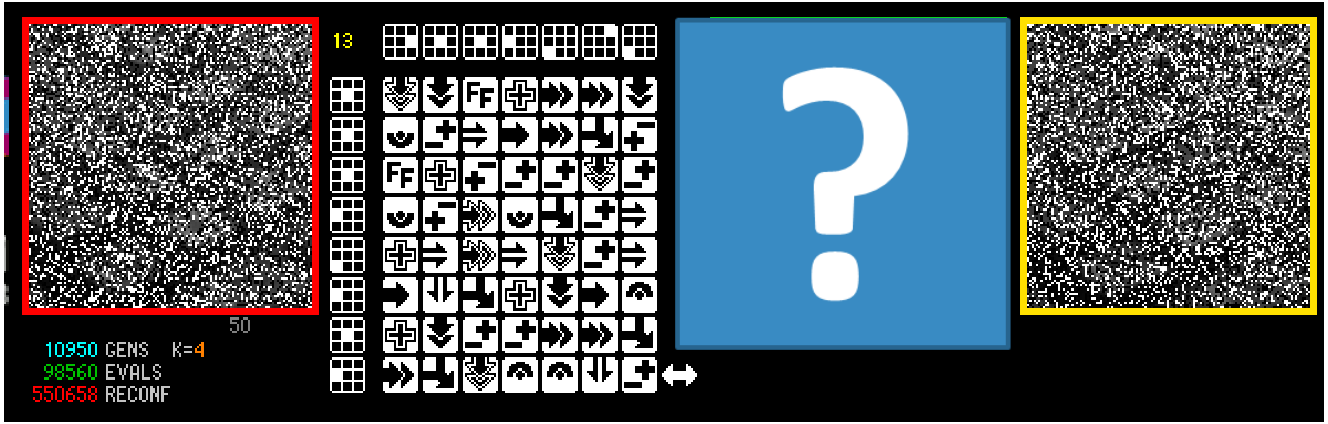
An Evolvable HW System based on a single processing systolic array



A. Otero, R. Salvador, J. Mora, E. de la Torre, T. Riesgo, L. Sekanina; "A fast Reconfigurable 2D HW core architecture on FPGAs for evolvable Self-Adaptive Systems", AHS 2011

System is adaptable and generalizable

Example: Evolvable HW system



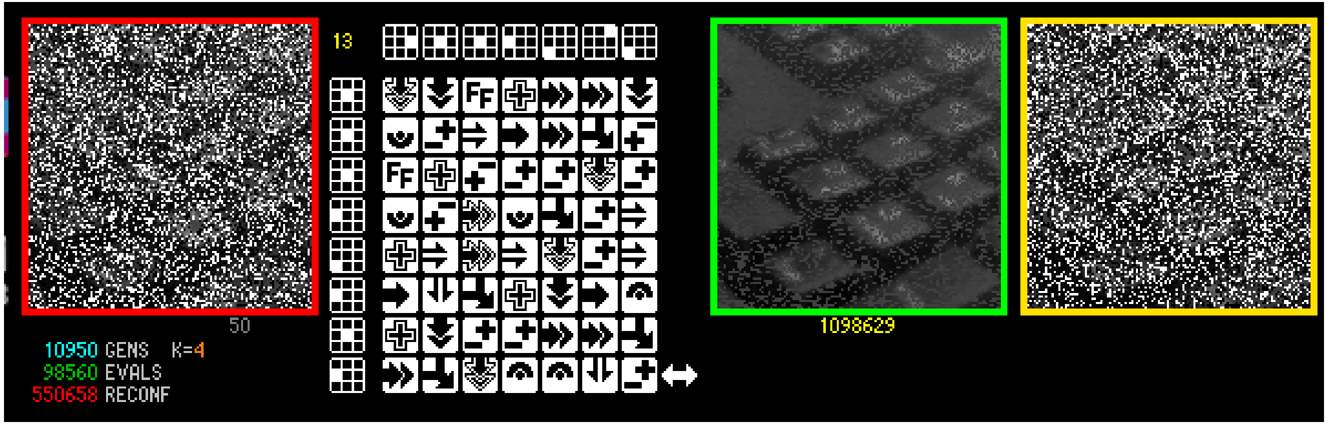
Adaptation Loop:

- **Adaptation Fabric**: Systolic array overlay
- **Monitor**: Fitness compute unit
- **KPI**: Sum of absolute differences (to minimise)
- **Adaptation Manager**: Genetic algorithm
- **Adaptation engine**: DPR on FPGA frames

Results

- Fast evolution: > 140.000 evals/sec, total: 1 sec
- Array works at 400 Mpixels/sec
- Small: 2 CLBs per PE
- Generalizable (noise filtering, edge detection, image enhancement, etc.)
- Scalable (grows or shrinks)
- Self-healing

Example: Evolvable HW system



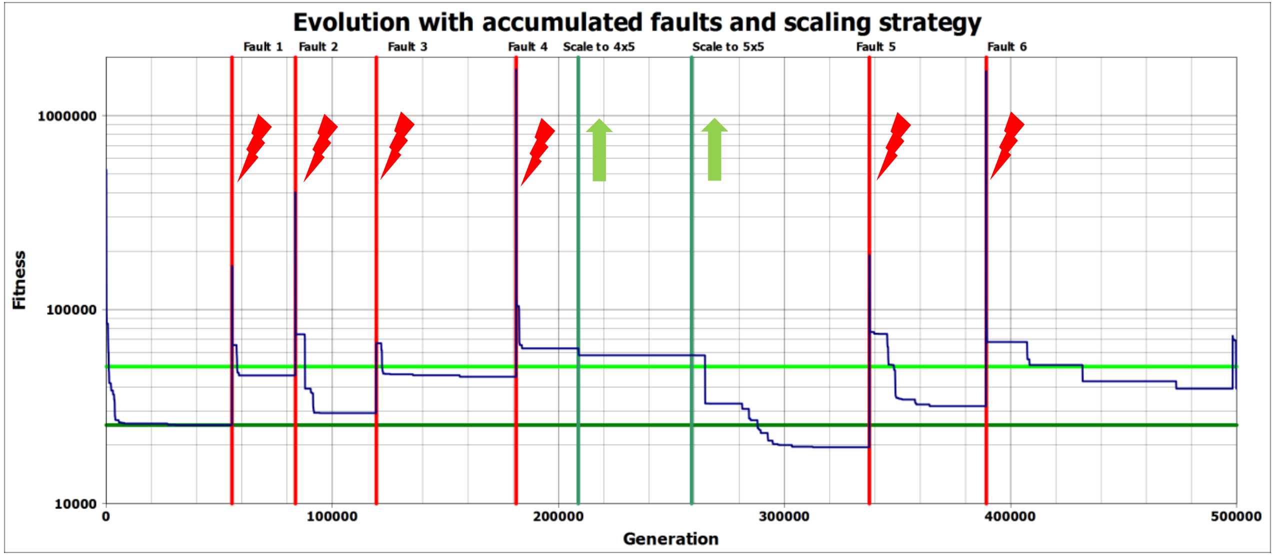
Adaptation Loop:

- **Adaptation Fabric**: Systolic array overlay
- **Monitor**: Fitness compute unit
- **KPI**: Sum of absolute differences (to minimise)
- **Adaptation Manager**: Genetic algorithm
- **Adaptation engine**: DPR on FPGA frames

Results

- Fast evolution: > 140.000 evals/sec, total: 1 sec
- Array works at 400 Mpixels/sec
- Small: 2 CLBs per PE
- Generalizable (noise filtering, edge detection, image enhancement, etc.)
- Scalable (grows or shrinks)
- Self-healing

Scalability and evolution for increased fault tolerance



Example of evolution with accumulated faults (threshold at 2x initial fitness)

A 4x4 recovers from 2 faults in average

A 7x7 recovers from 12 faults in average

Lifetime of the system extended 6 times

Outline

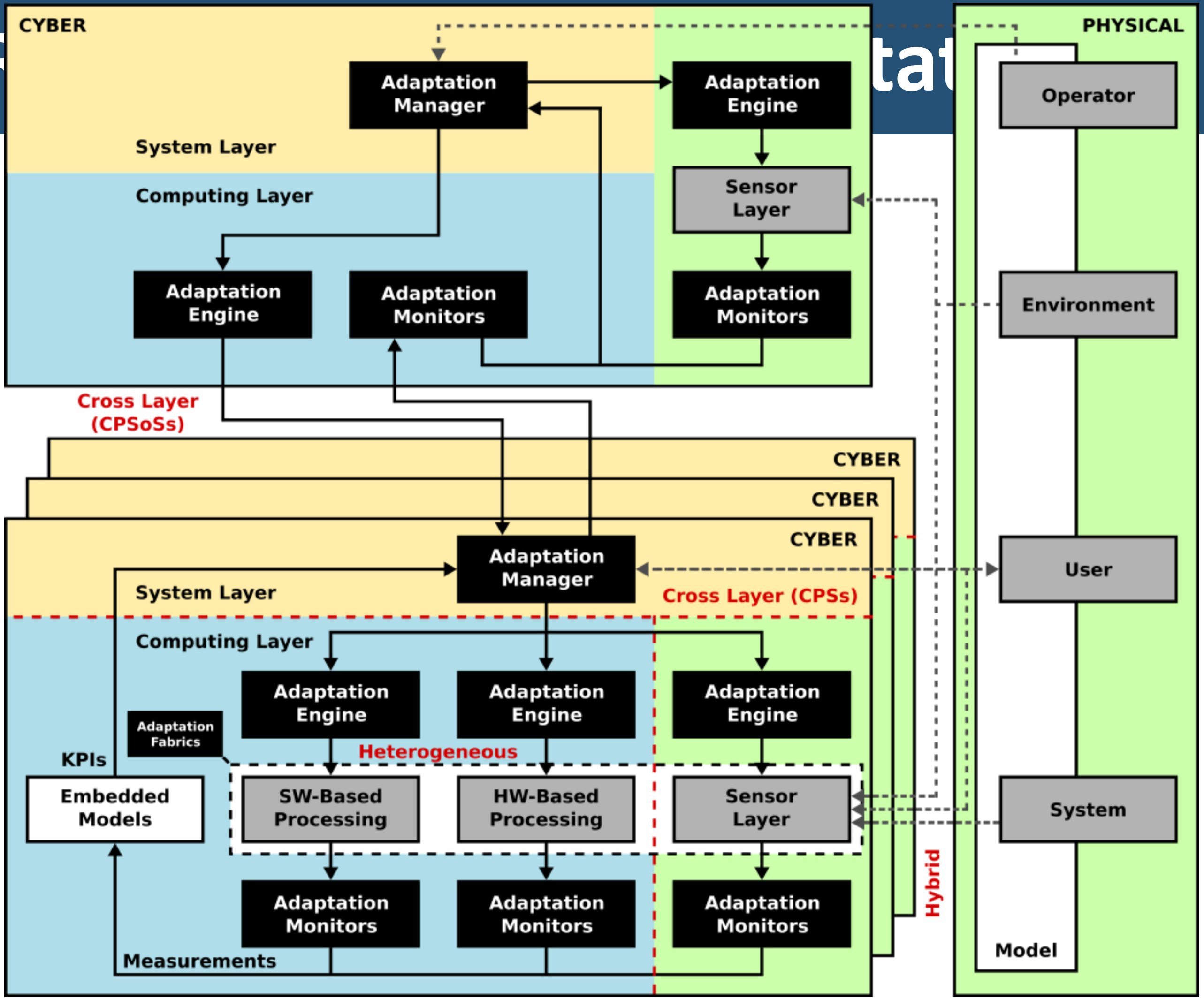
- Adaptive systems: Concepts & Definition
  - Triggers and types of Adaptation
  - Levels of autonomy. How to build it
  - The Adaptation Loop
  - An example: Evolvable HW
- Adaptive CPS: The CERBERO approach
  - **Big Picture. The CERBERO Adaptation Loops at CPS and CPSoS levels**
- Deep Dive into CERBERO HW Adaptation
  - ARTICo3
  - MDC-compliant CG adaptation
  - Mixed-Grain Adaptivity
- CERBERO Beyond SoA & Take Out
  - Key Advancements and Integration

CPS Self-Adaptation in the CERBERO Project

CERBERO

CERBERO CPS & SoS Self-Adaptation

CERBER tat


CYBER
Adaptation Manager
Adaptation Engine
System Layer
Computing Layer
Sensor Layer
Adaptation Engine
Adaptation Monitors
Adaptation Monitors
PHYSICAL
Operator
Environment
User
System
Model
Cross Layer (CPSoSs)
CYBER
CYBER
CYBER
Adaptation Manager
System Layer
Cross Layer (CPSs)
Computing Layer
Adaptation Engine
Adaptation Engine
Adaptation Engine
Adaptation Fabrics
Heterogeneous
KPIs
Embedded Models
SW-Based Processing
HW-Based Processing
Sensor Layer
Adaptation Monitors
Adaptation Monitors
Adaptation Monitors
Measurements
Hybrid

CERBERO CPS Self-Adaptation

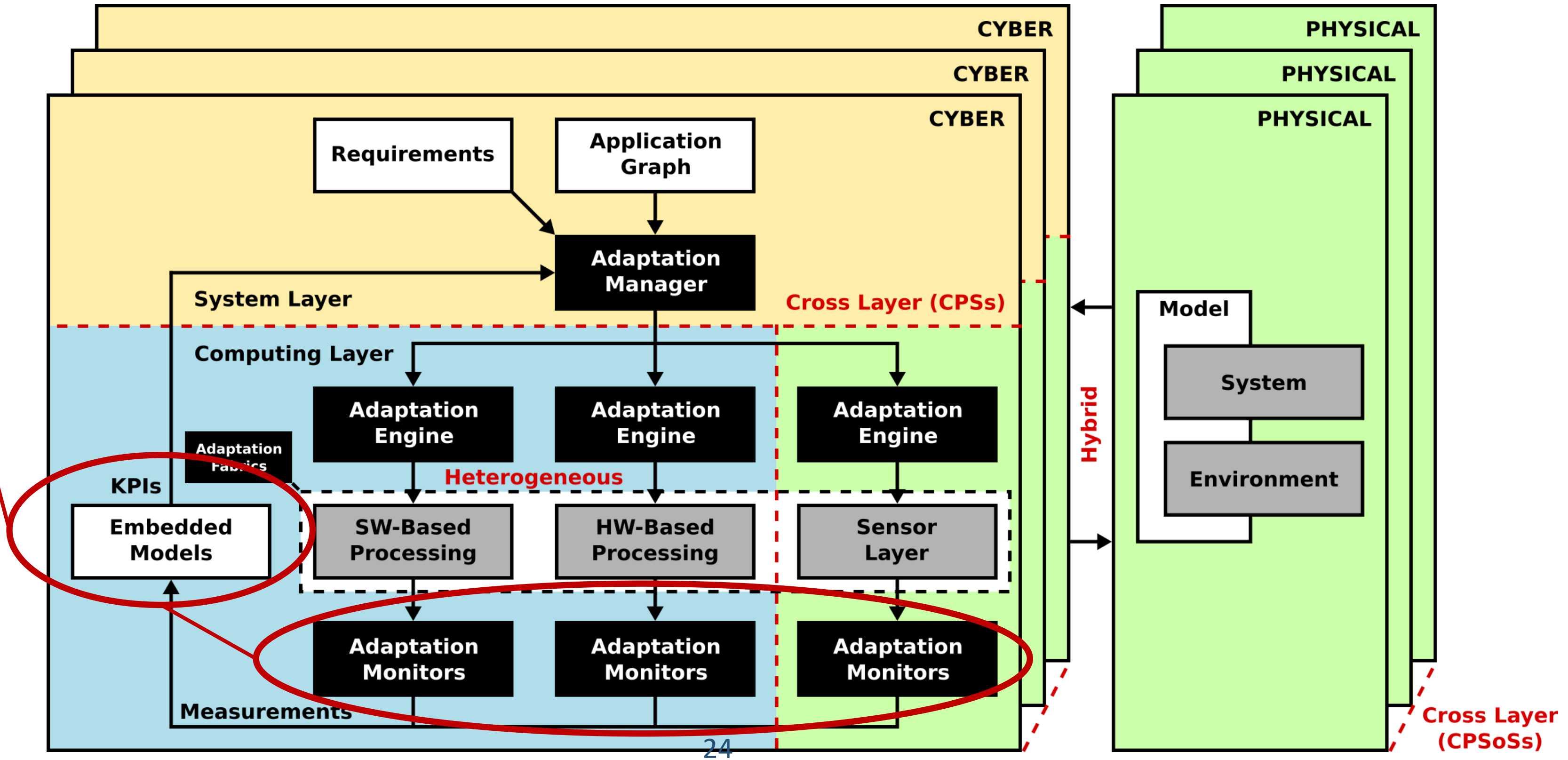
- ***Cross-layer Approach*** → ***CPS*** & ***CPSoS*** level
- All elements in the ***adaptation loop*** are included:
  - ***Monitoring***
    - Context-awareness → Multiple sensors + Sensor Fusion
    - Self-awareness → HW and SW tasks common monitoring infrastructure→ PAPI
  - ***KPI extraction*** → PHY and CYBER runtime models
  - ***Adaptation management*** → Dynamic task management → SPIDER
  - ***Adaptation fabrics***
    - HW adaptation → mixed-grain, multiple solutions
    - ARTICo3, MDC, Just-In-Time composition, and mixed approaches
    - SW adaptation → Task migration between cores

CERBERO CPS Self-Adaptation

- ***Cross-layer Approach*** → ***CPS*** & ***CPSoS*** level
- All elements in the ***adaptation loop*** are included:
  - ***Monitoring***
    - Context-awareness → Multiple sensors + Sensor Fusion
    - Self-awareness → HW and SW tasks common monitoring infrastructure→ PAPI
  - ***KPI extraction*** → PHY and CYBER runtime models
  - ***Adaptation management*** → Dynamic task management → SPIDER
  - ***Adaptation fabrics***
    - HW adaptation → mixed-grain, multiple solutions
    - ARTICo3, MDC, Just-In-Time composition, and mixed approaches
    - SW adaptation → Task migration between cores

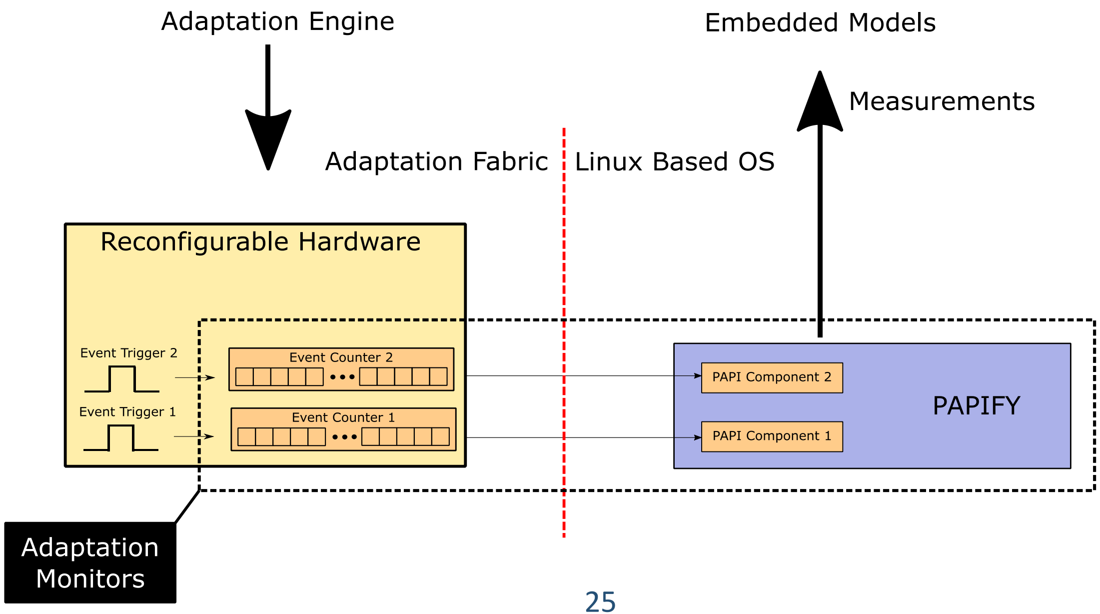
Monitors and KPIs in CERBERO

ADAPTATION MONITORS: hardware/software trackers for the status of the fabric



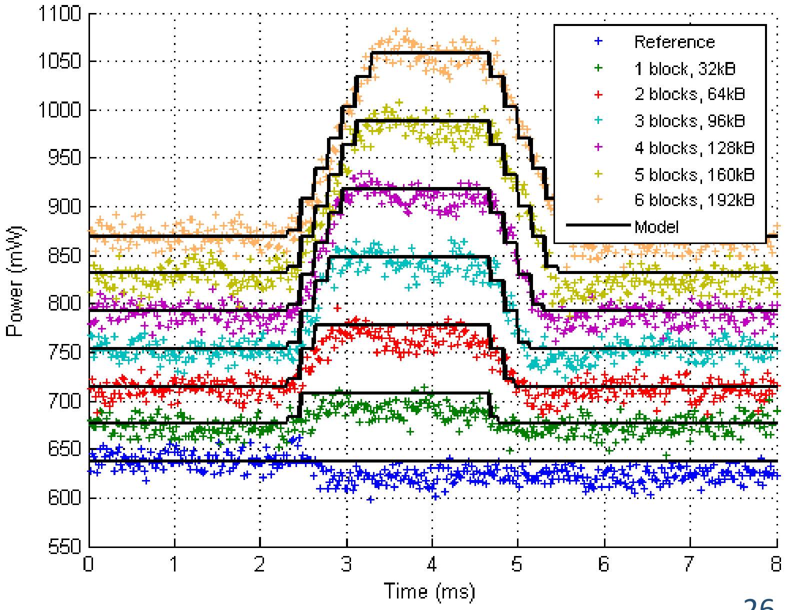
Unified access to Monitors: PAPI

PAPI (Performance API) : Standard SW approach for performance
Extension to HW
Extension to energy and fault monitors



Execution model for ARTICo^3

- Model estimations vs. real measurements
  - Kernel: AES-256 CTR
  - Platform #1: HiReCookie Node (table)
  - Platform #2: KC705 Board (table and figure)



$$P_{core}(t) = P_{base}(t) + P_{dma}(t) + \sum_{i=1}^{n_k} n_{exi}(t) \cdot P_{exi}(t)$$

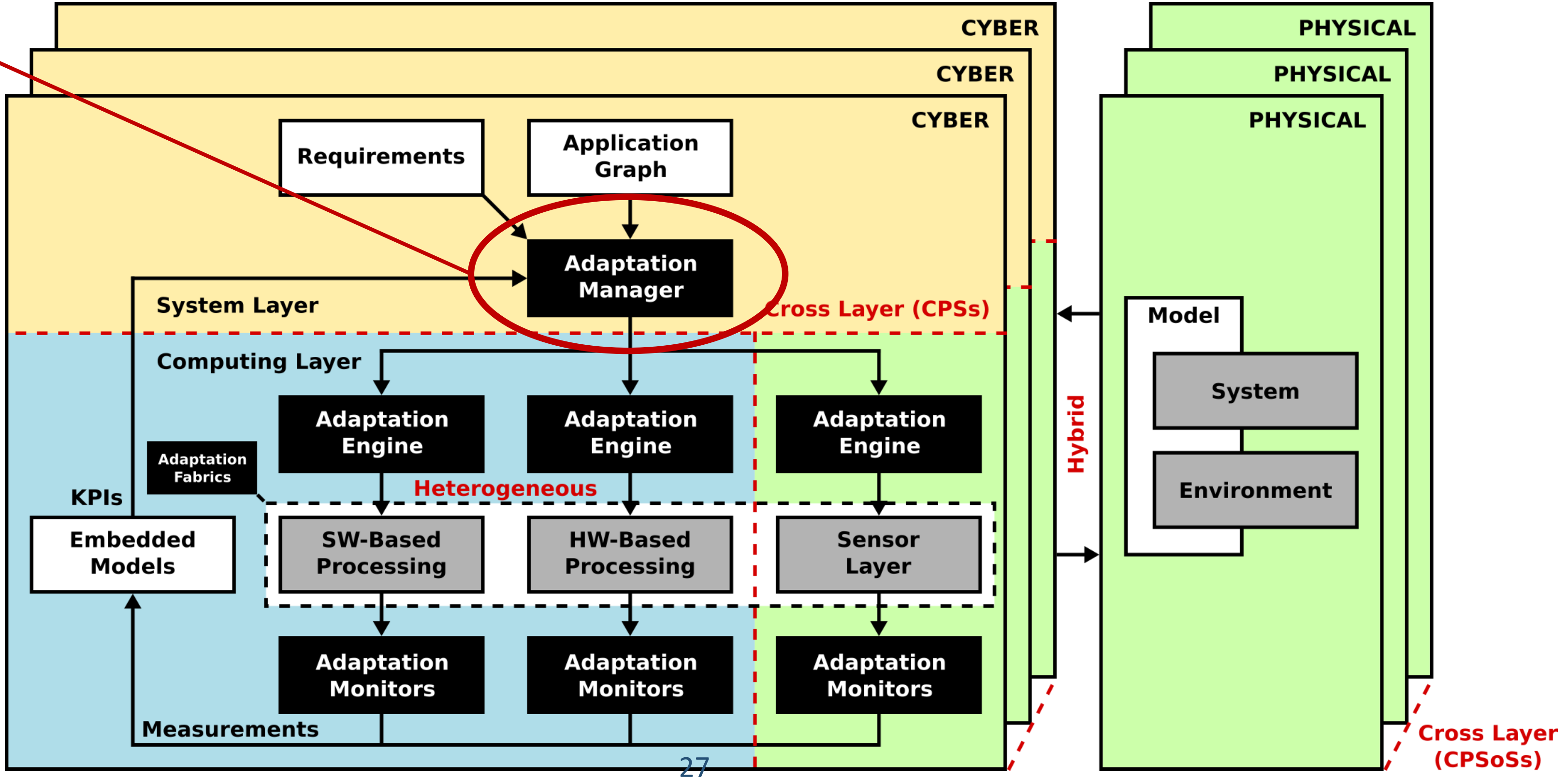
$$P_{mem}(t) = P_{mem,s} + P_{mem,d}(t)$$

Parameter	Value (mW)	
	KC705	HiReCookie
P_{dma}	6.93	5
P_i	38.66	44.55
P_{exi}	31.57	22.21
$P_{mem,s}$	792	91.6
$P_{mem,d}$ (read)	768	133.4
$P_{mem,d}$ (write)	1368	101.25

CERBERO

Self-Adaptation Management: CERBERO style

ADAPTATION MANAGER: high-level entity with run-time decision-making capabilities



CERBERO

Runtime Management Systems

- **WHO**:
  - OpenMP

    [OpenMP 4.5 Specification, Nov. 2015, www.openmp.org/wp-content/uploads/openmp-4.5.pdf]
  - LLVM Runtime

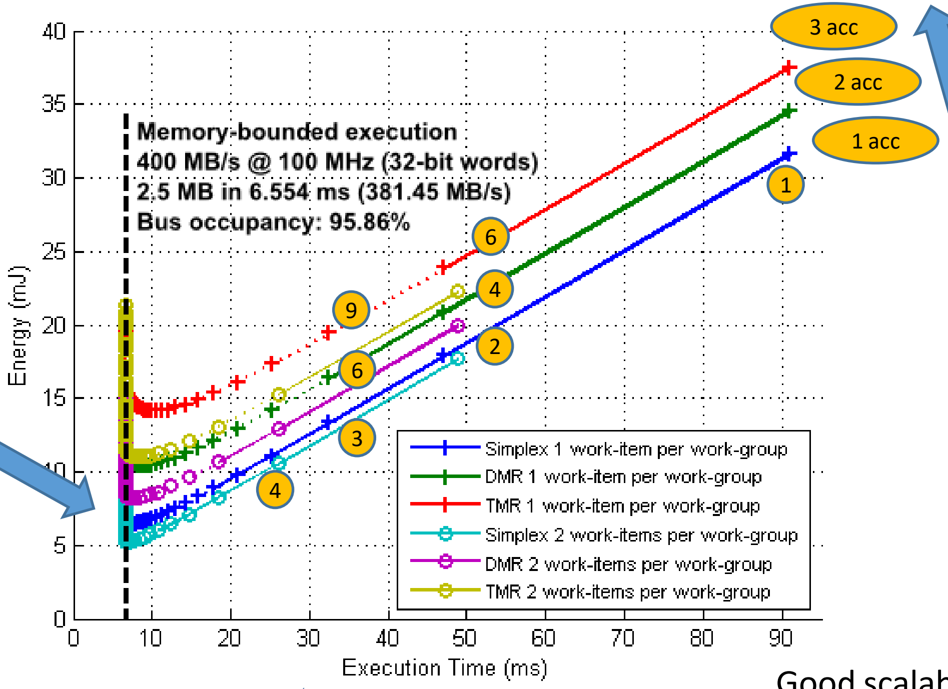
    [compiler-rt.llvm.org]
  - OpenCL

    [OpenCL Specification Version 2.2 , online: www.khronos.org/registry/OpenCL/specs/opencl-2.2.pdf]
- **WHAT**:
  - Deploy applications on the fly on the available computational, communication and storage resources, by using greedy strategies.
- **WHY**:
  - Functional needs
- **HOW**:
  - Functional Information by means of imperative MoCs.

Energy vs. Execution Time vs. Fault Tolerance

Run-time configurable trade-off between energy, computing performance and fault tolerance

Max. Acceleration AND energy efficiency when FPGA is fully busy

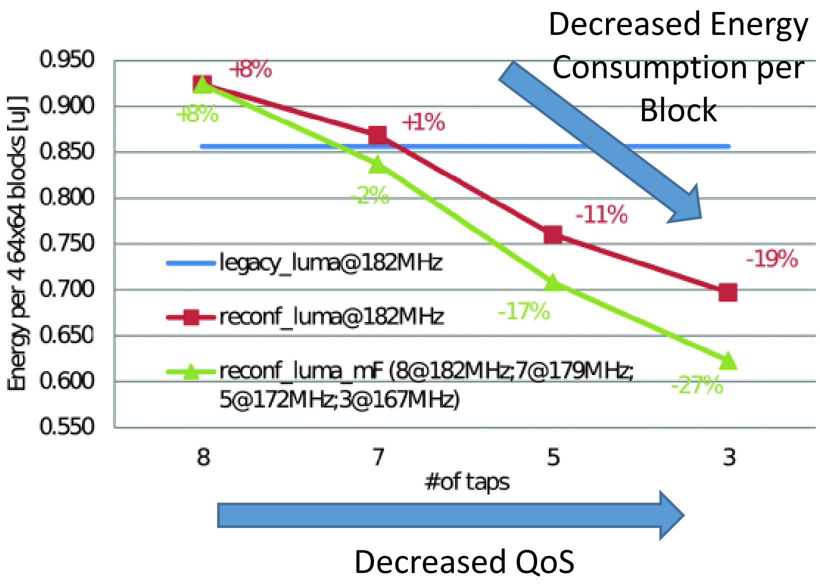


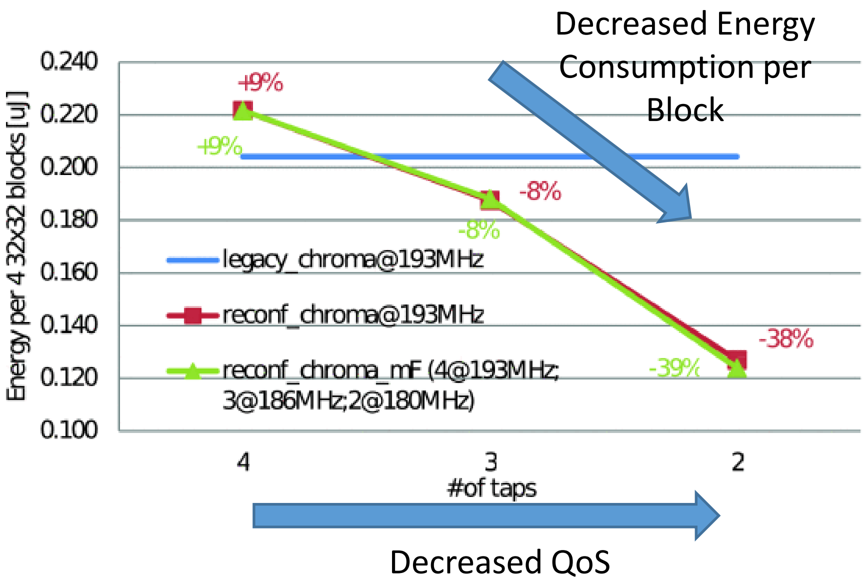
Increased fault tolerance (simple, double, triple redundancy)

Good scalability when increasing the number of HW accelerators

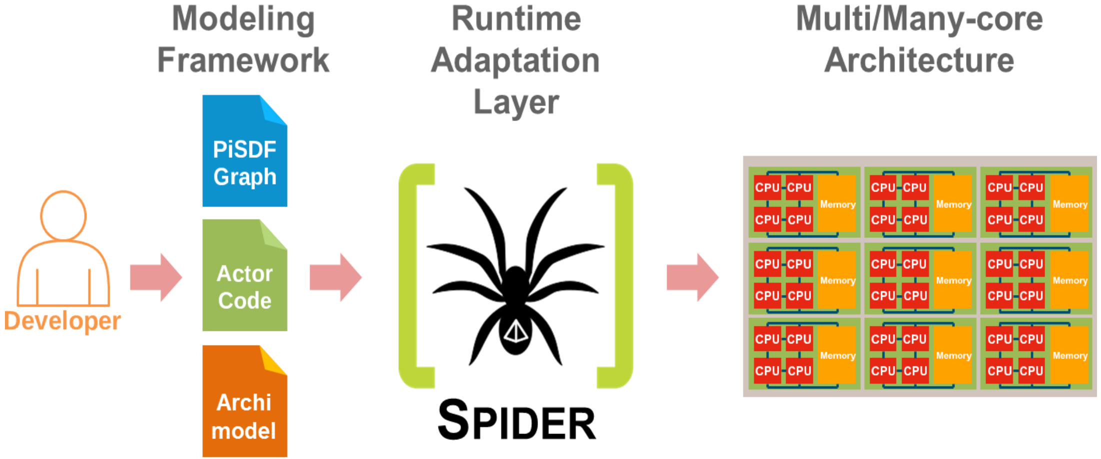
Energy vs. Quality of Service

Run-time configurable trade-off between energy and computation precision





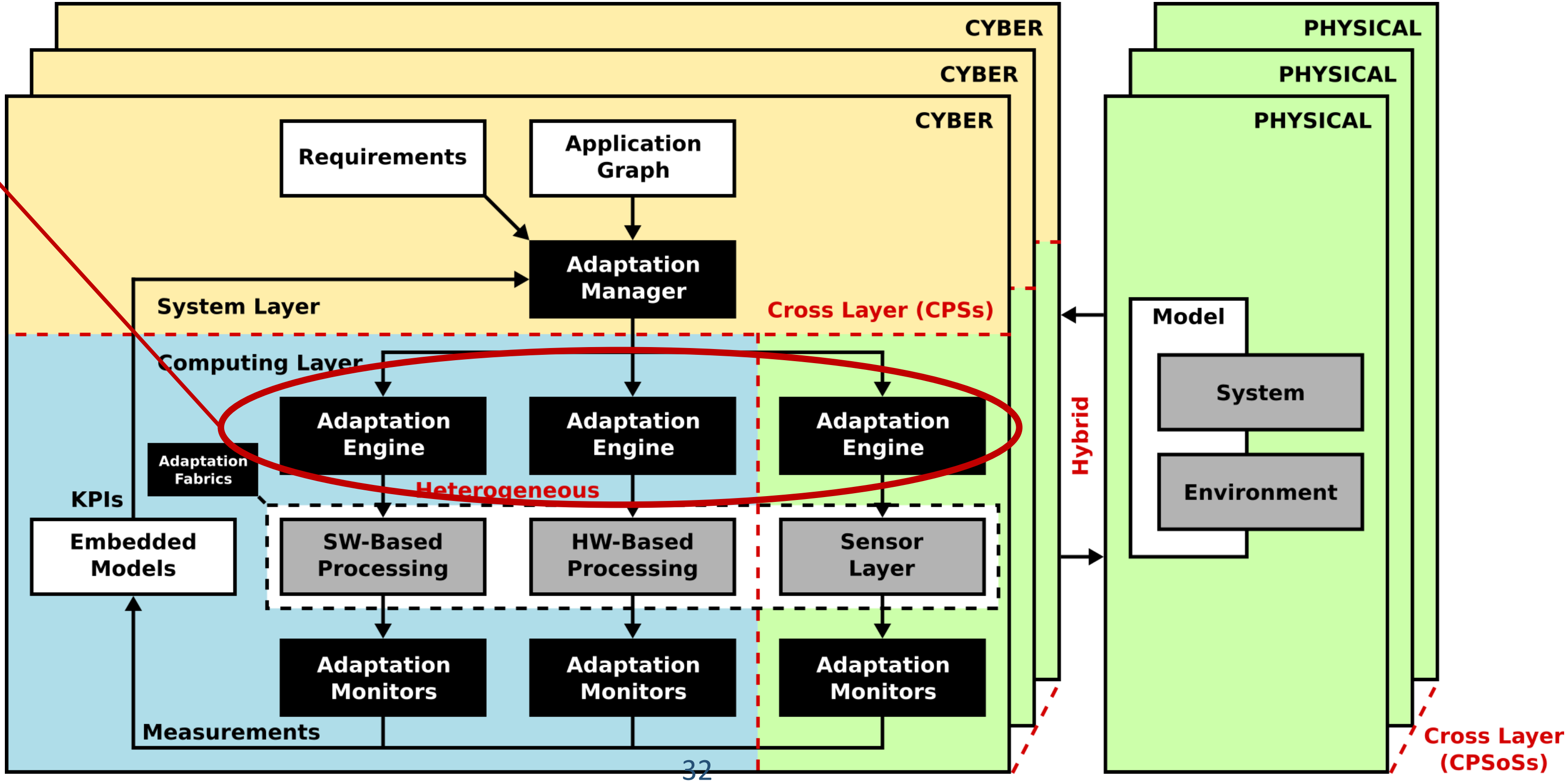
SPIDER



- **WHO**:
  - SPIDER
- **WHAT**:
  - Deploy applications in **HW and SW fabrics** on the fly on the available resources.
- **WHY**:
  - Functional **& Non-Functional** Needs
- **HOW**:
  - Parameterized Dataflow MoCs
  - KPI Runtime Models
  - Model of the Architecture

Self-Adaptation Management: CERBERO style

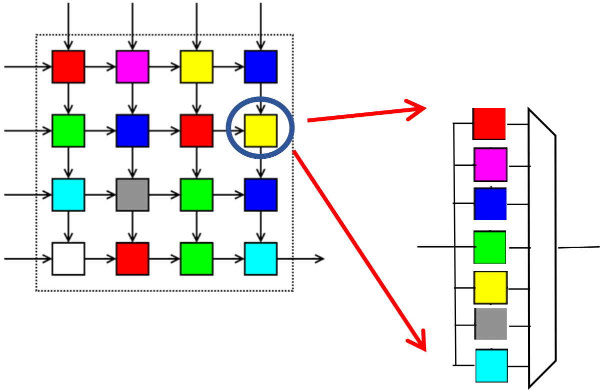
ADAPTATION ENGINES: hardware/software changing the fabric configuration



VRC & DPR

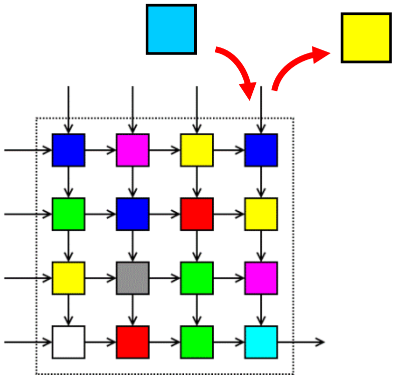
VRC → Virtual Reconfigurable Circuits

- High reconfiguration speed
- Lower operation speed (mux and size)
- Higher Area Overhead
- Technology independent (ASIC or FPGA)



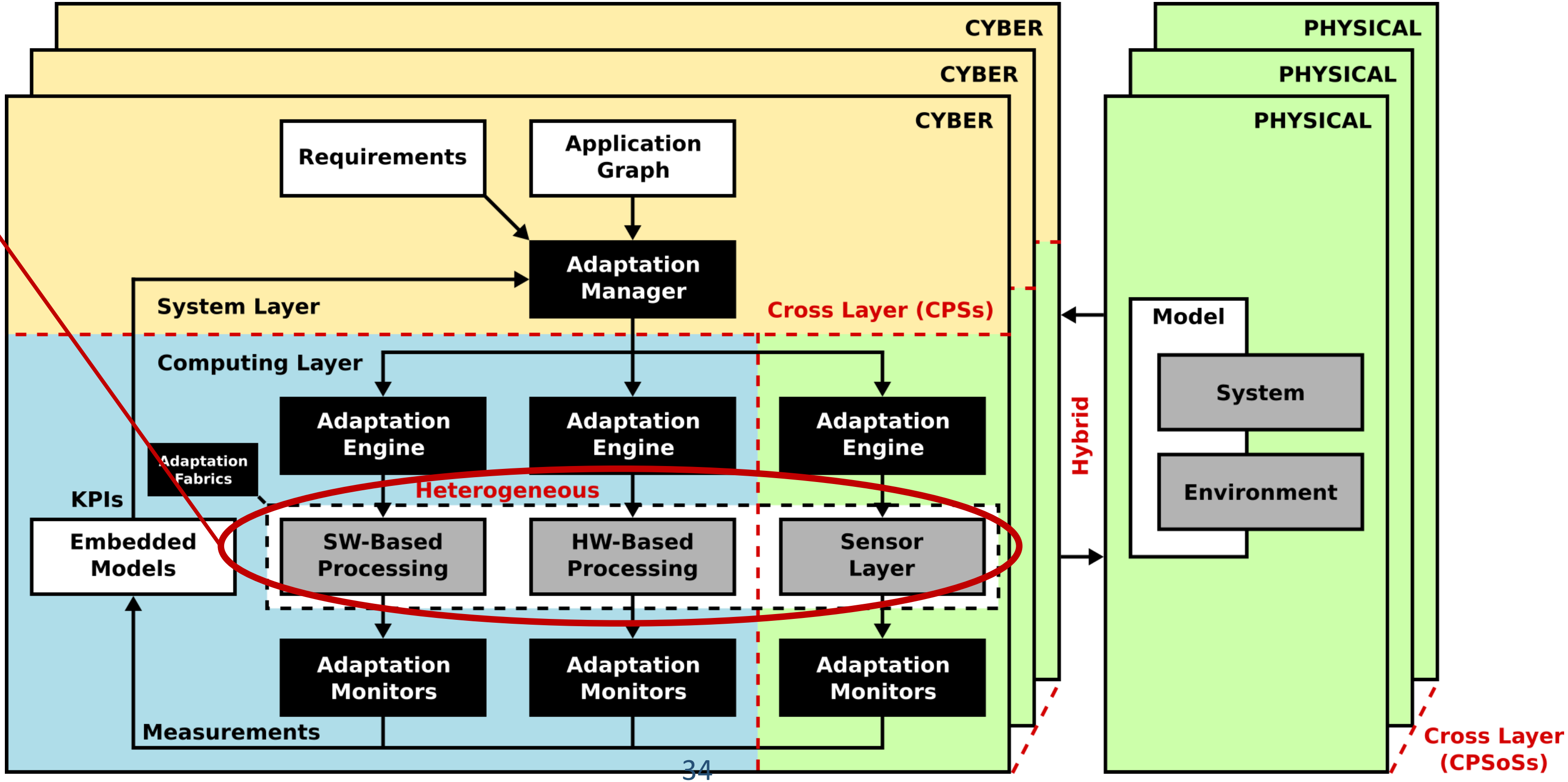
DPR → Dynamic and Partial Reconfiguration

- Lower reconfiguration speeds
- Better operation speed (no mux/less logic)
- Better Resource Utilization (no dark logic)
- Higher Flexibility and Scalability
- Technology dependent (FPGA)



Self-Adaptation Management: CERBERO style

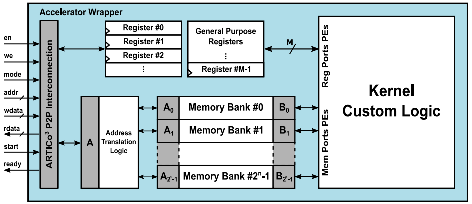
HETEROGENOUS ADAPTATION FABRIC: computing and sensing resources

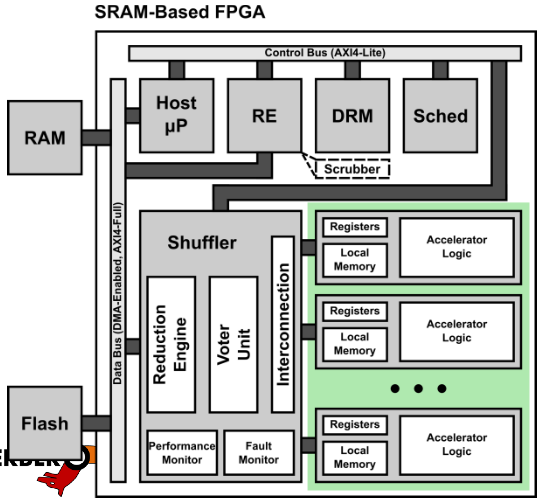


Adaptation fabrics addressed in CERBERO

Adaptation fabrics addressed in CERBERO

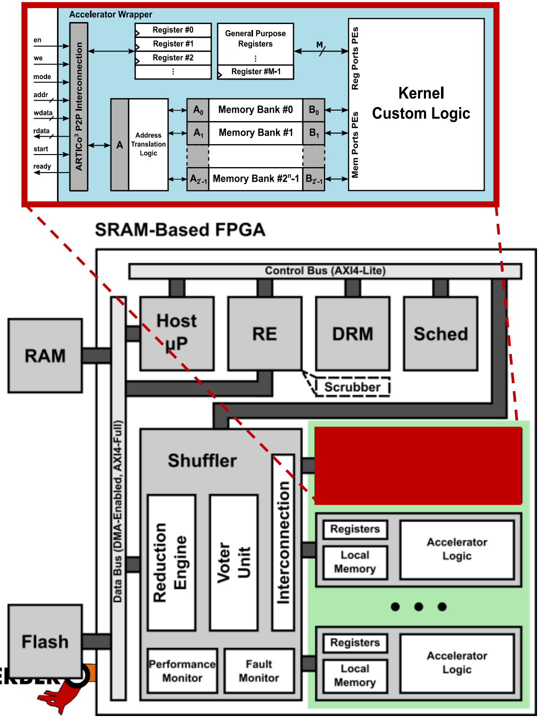
ARTICo3





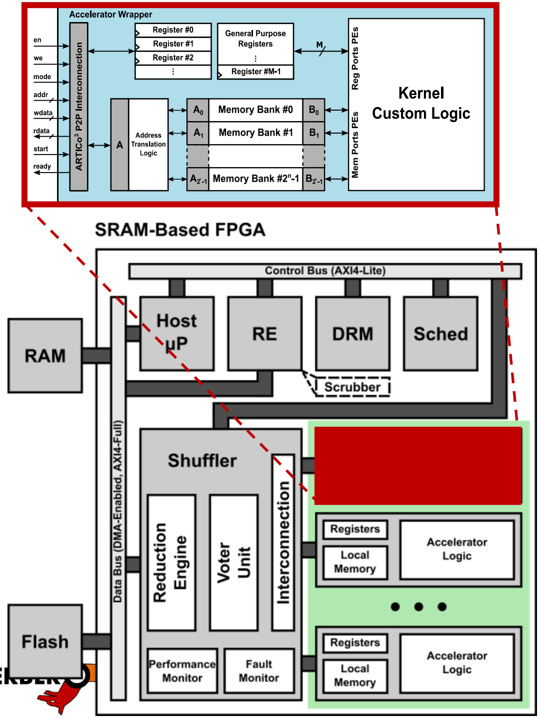
Adaptation fabrics addressed in CERBERO

ARTICo3

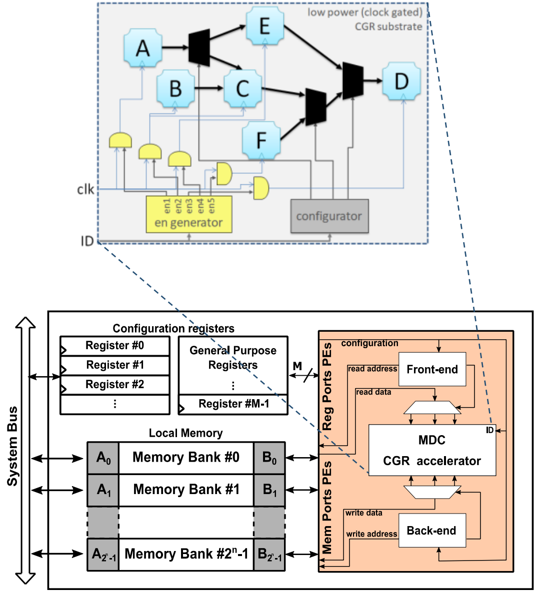


Adaptation fabrics addressed in CERBERO

ARTICo3

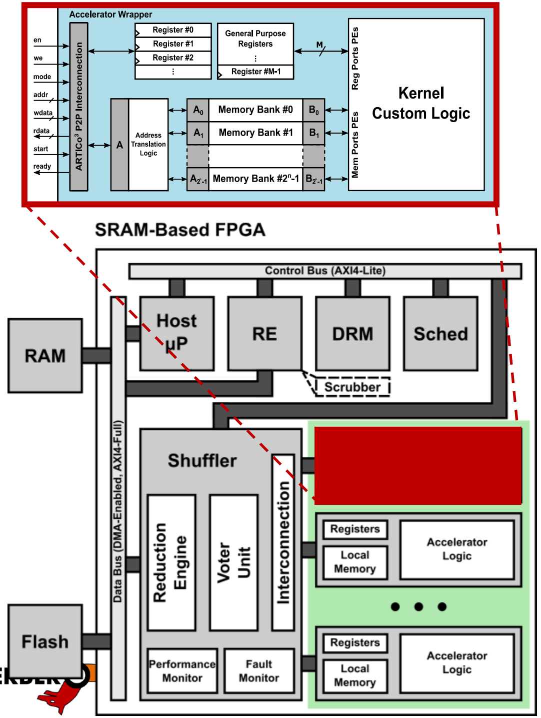


Coarse-Grain MDC Logic

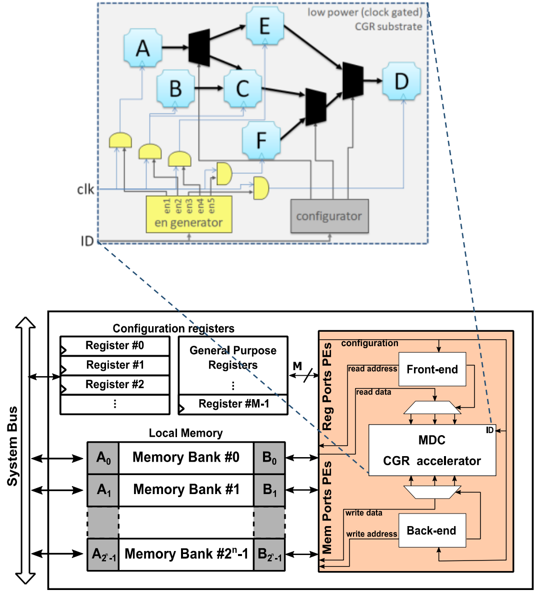


Adaptation fabrics addressed in CERBERO

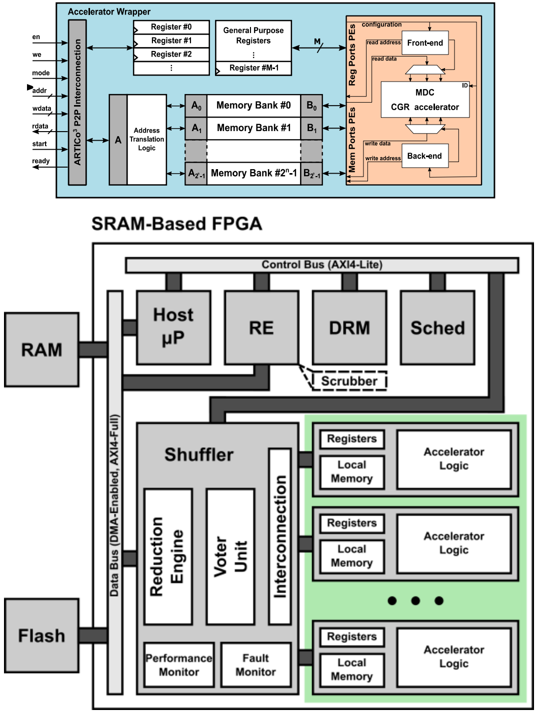
ARTICo3



Coarse-Grain MDC Logic

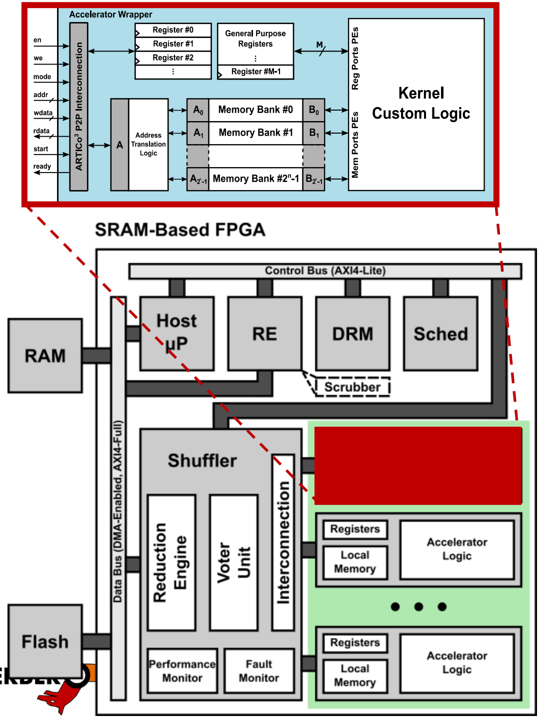


Mixed-Grain A3+MDC

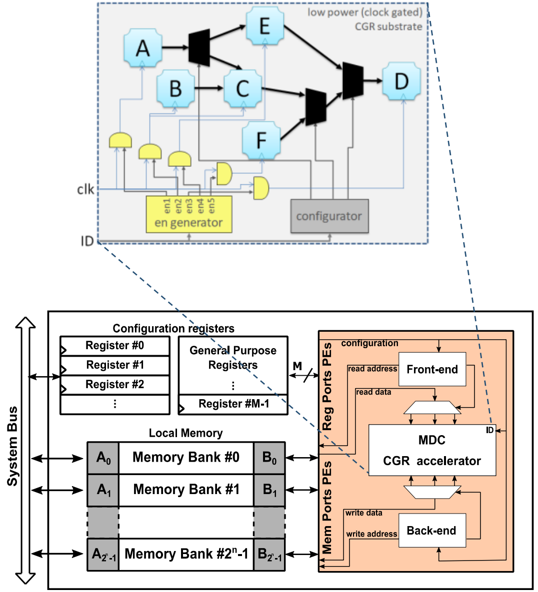


Adaptation fabrics addressed in CERBERO

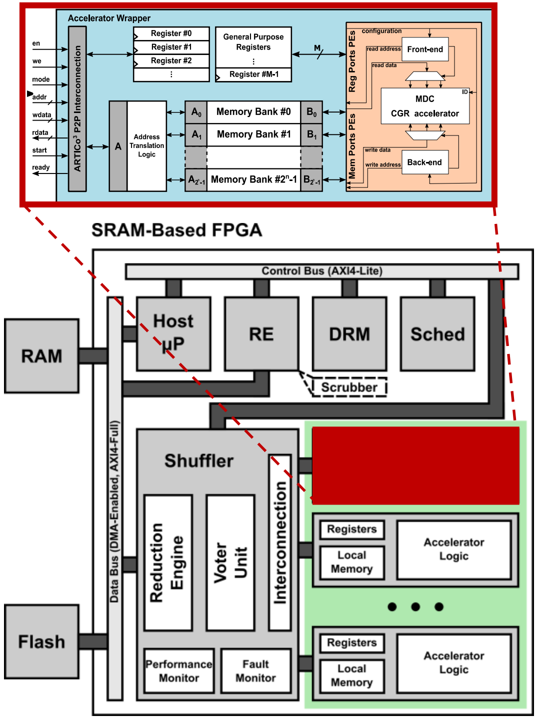
ARTICo3



Coarse-Grain MDC Logic



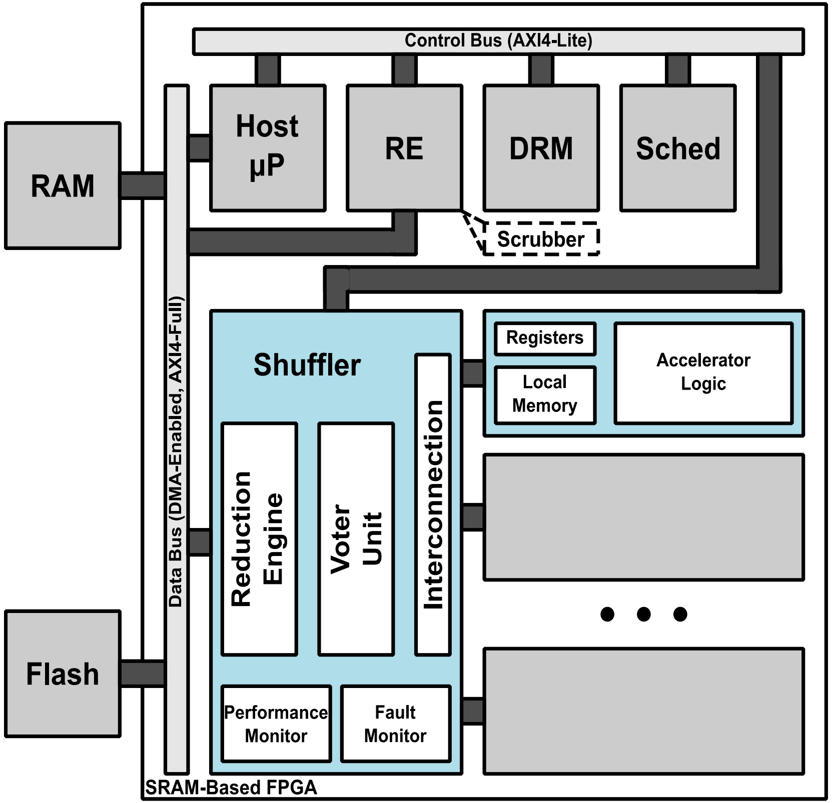
Mixed-Grain A3+MDC



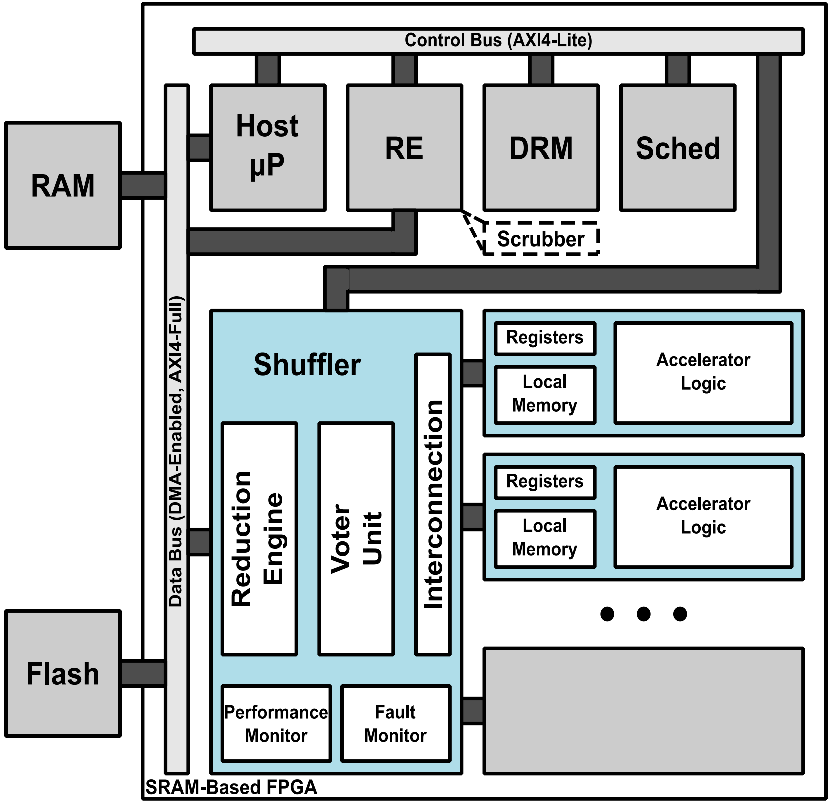
Outline

- Adaptive systems: Concepts & Definition
  - Triggers and types of Adaptation
  - Levels of autonomy. How to build it
  - The Adaptation Loop
  - An example: Evolvable HW
- Adaptive CPS: The CERBERO approach
  - Big Picture. The CERBERO Adaptation Loops at CPS and CPSoS levels
- **Deep Dive into CERBERO HW Adaptation**
  - **ARTICo3**
  - **MDC-compliant CG adaptation**
  - **Mixed-Grain Adaptivity**
- CERBERO Beyond SoA & Take Out
  - Key Advancements and Integration

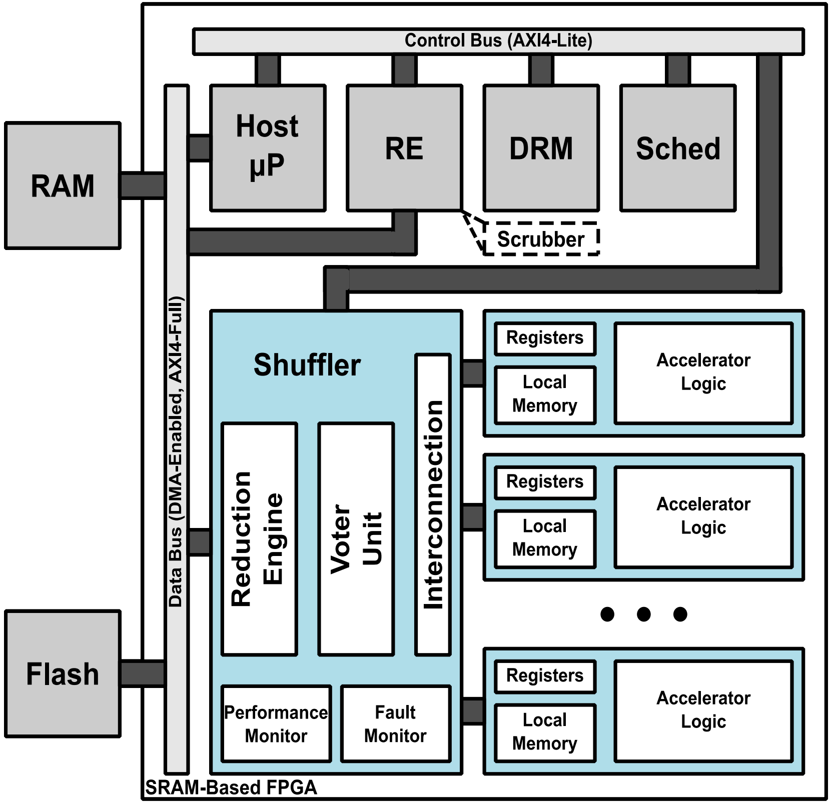
ARTICo^3 - Framework



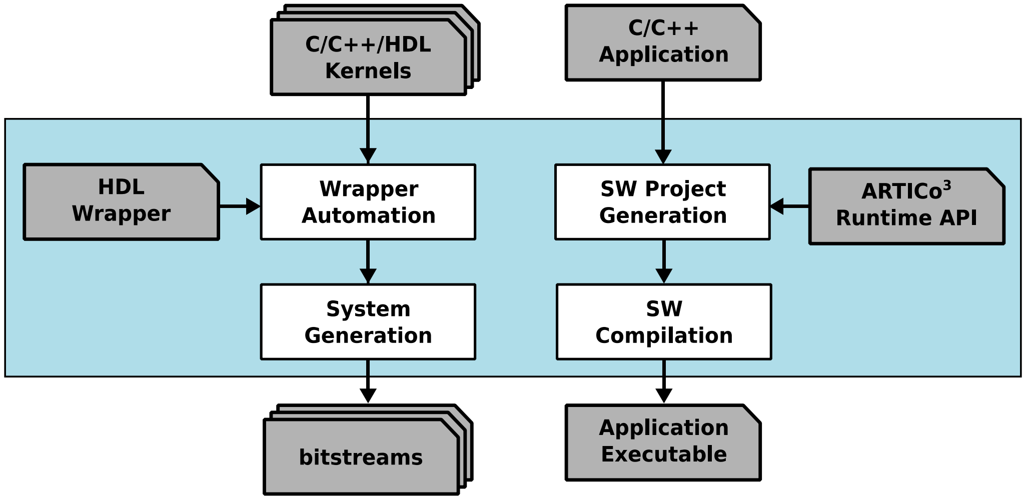
ARTICo³ - Framework

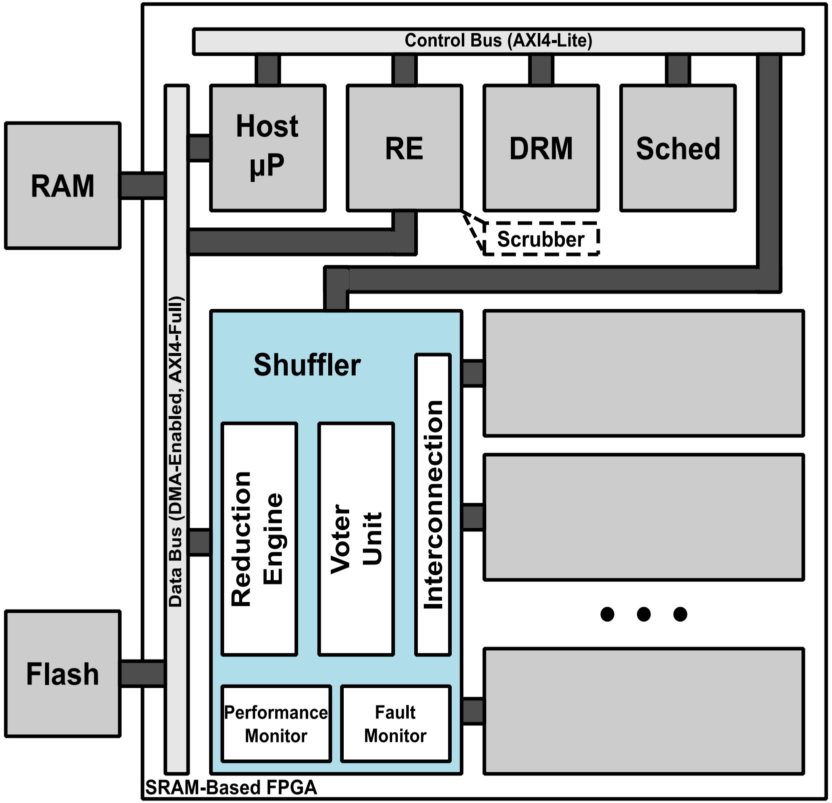


ARTICo^3 - Framework

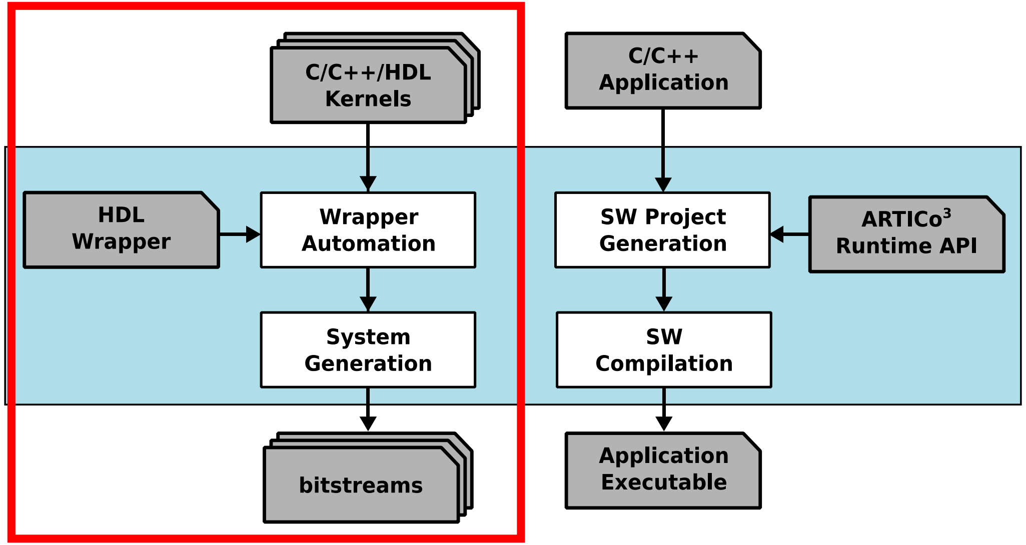


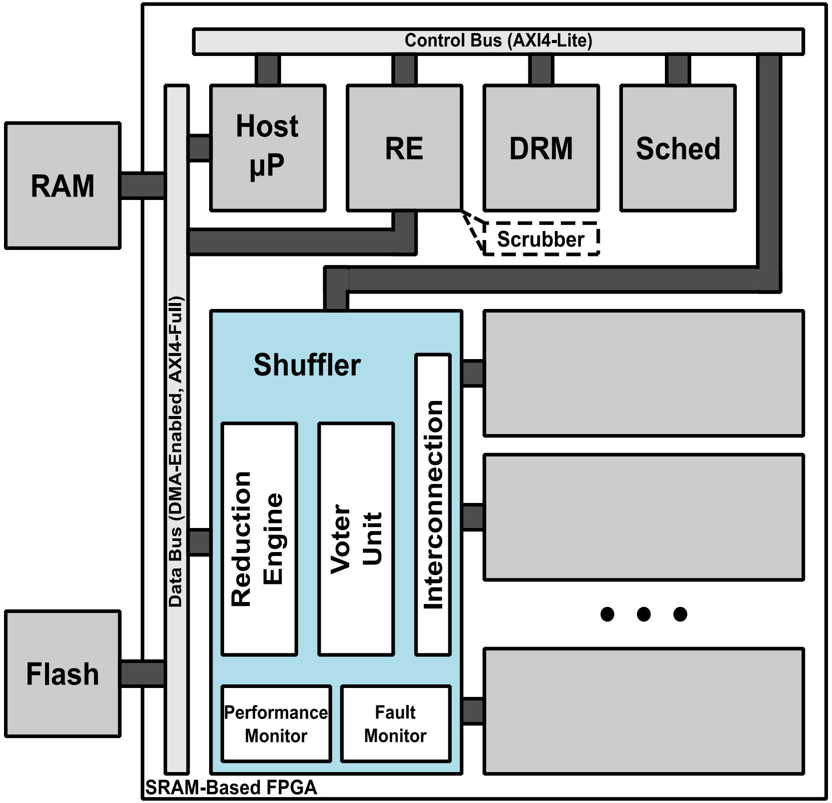
ARTICo³ - Framework



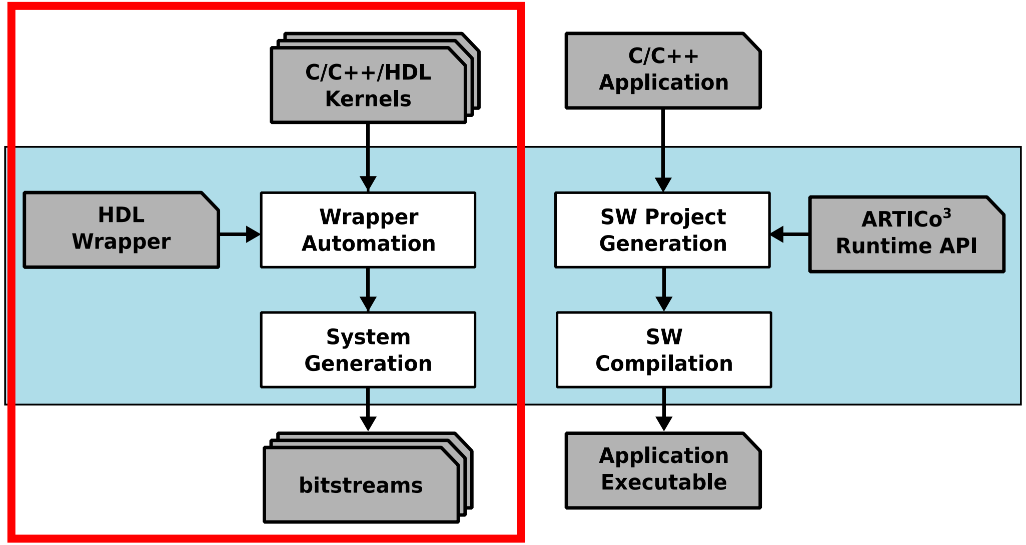


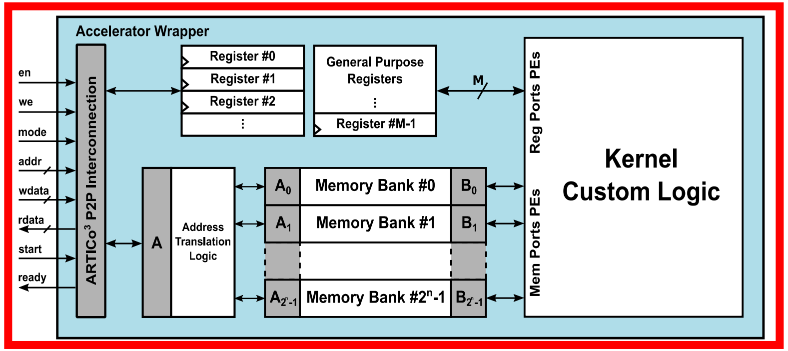
ARTICo³ - Framework

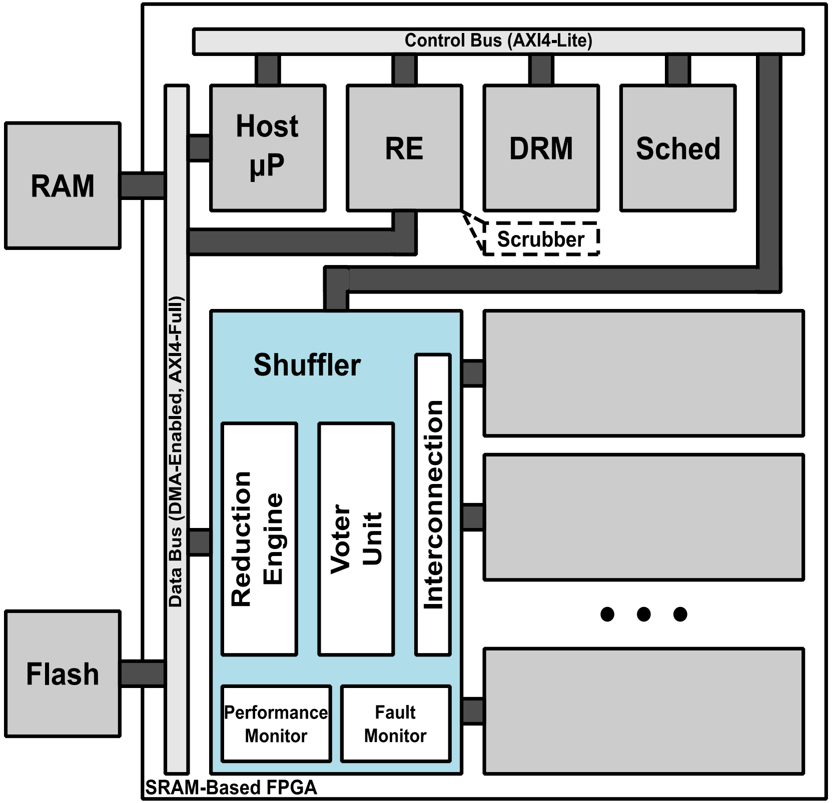




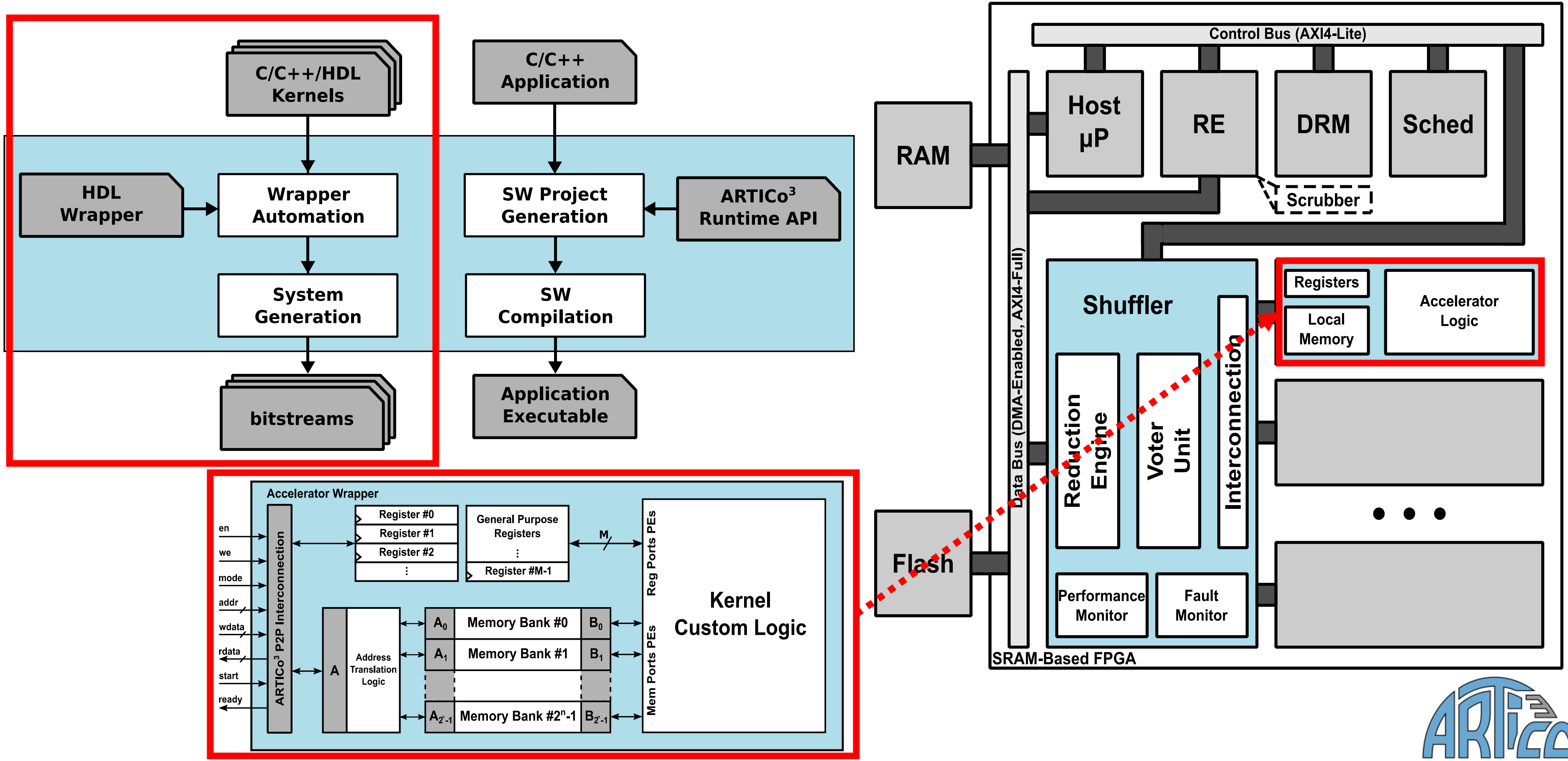
ARTICo³ - Framework



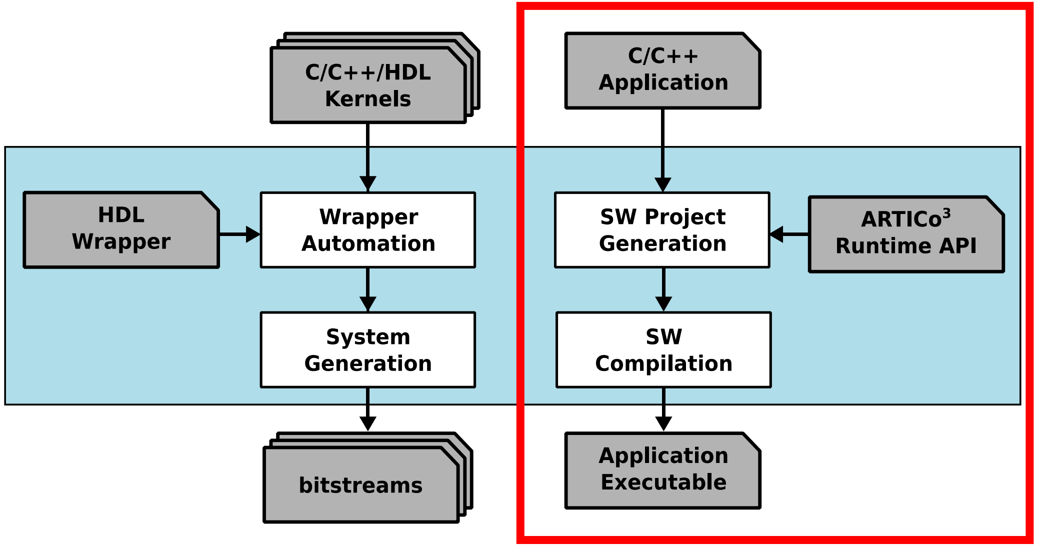


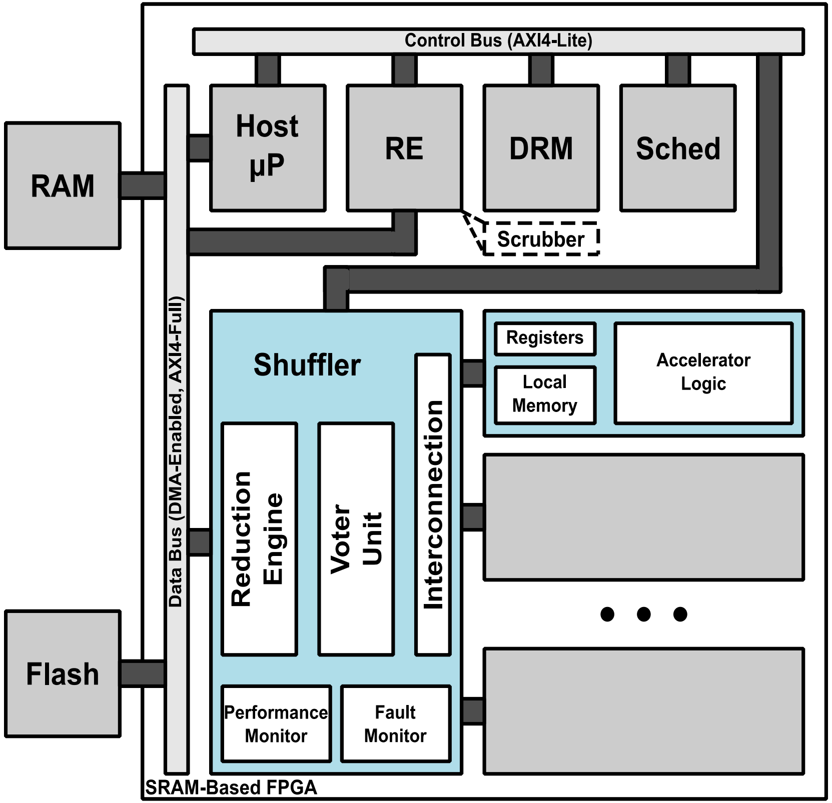


ARTICo³ - Framework

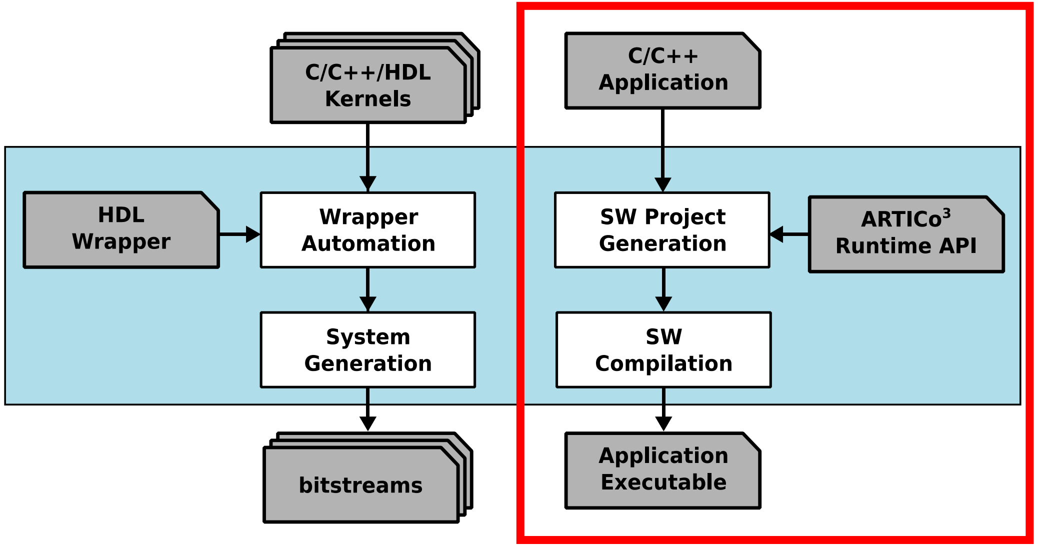


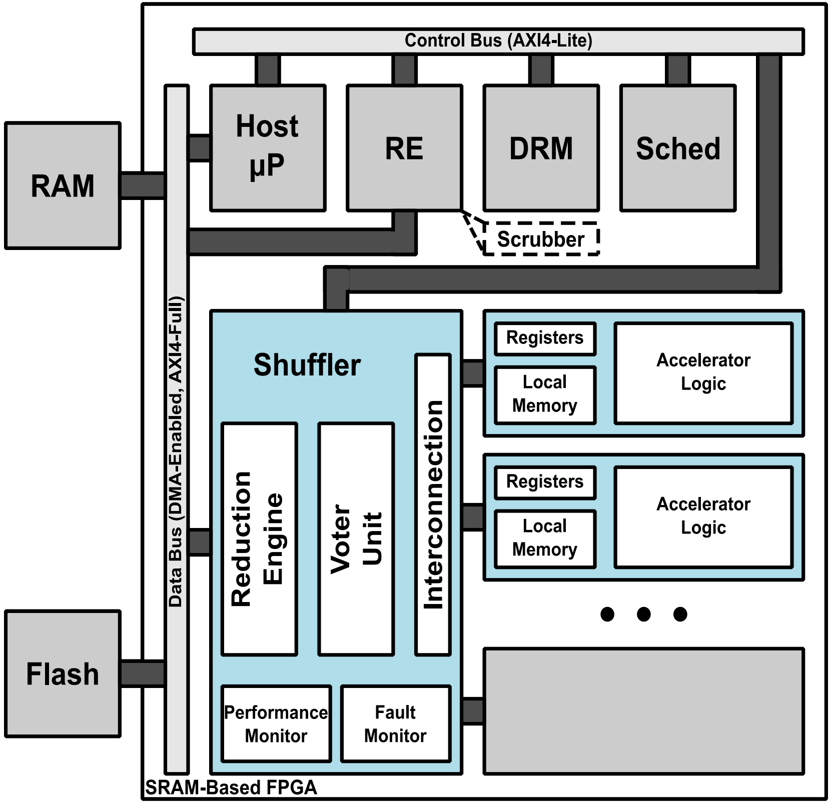
ARTICo^3 - Framework



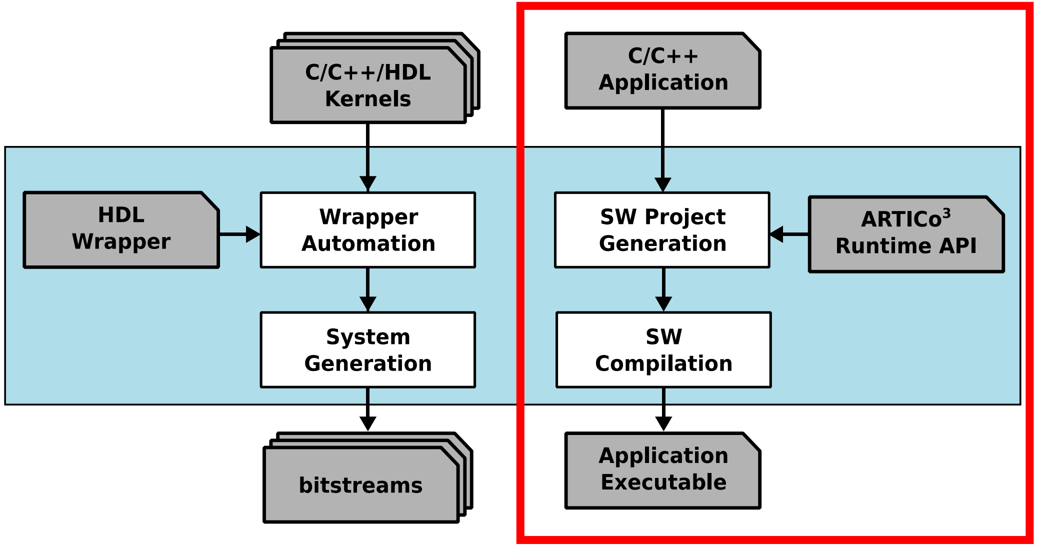


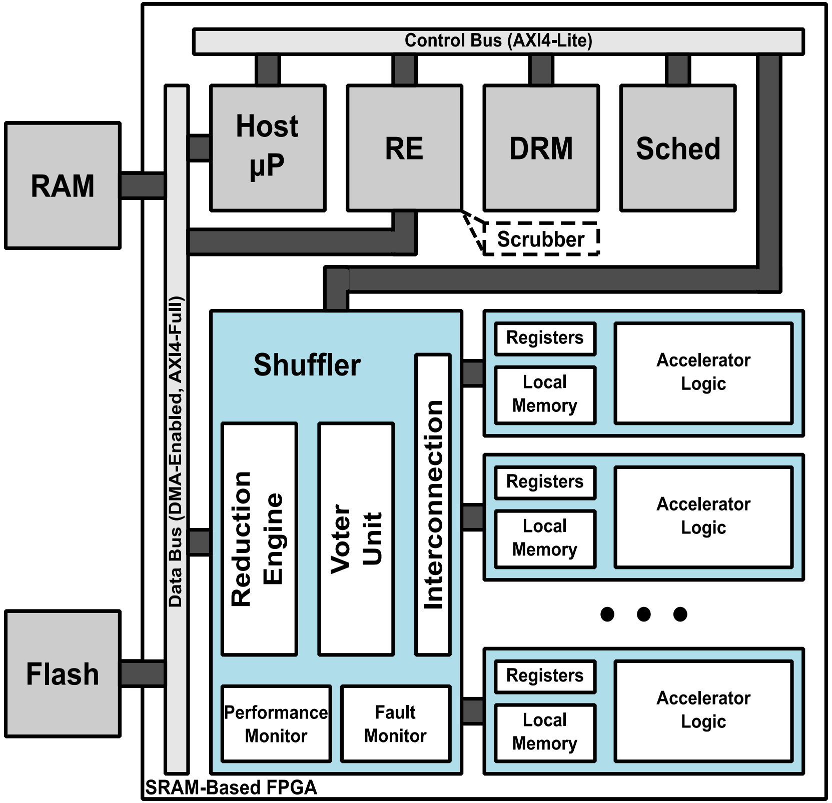
ARTICo³ - Framework





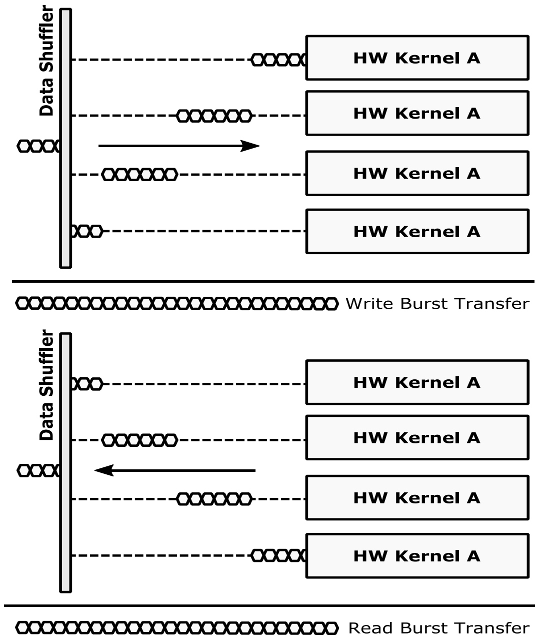
ARTICo^3 - Framework



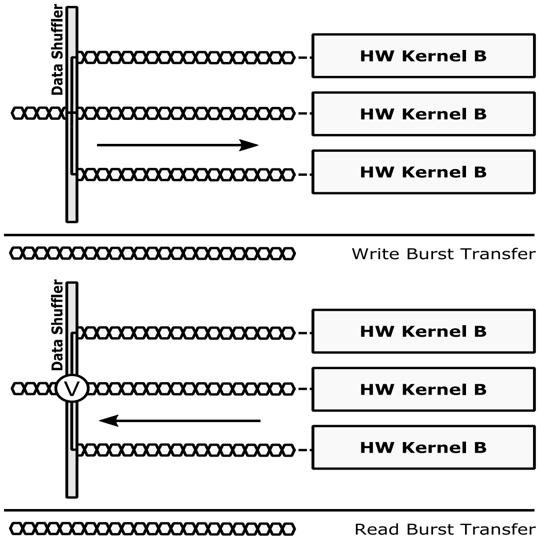


ARTICo^3 - Transaction Modes

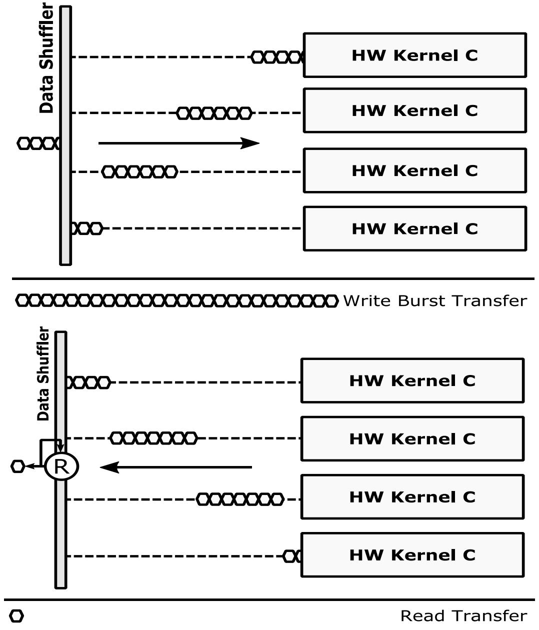
Parallel



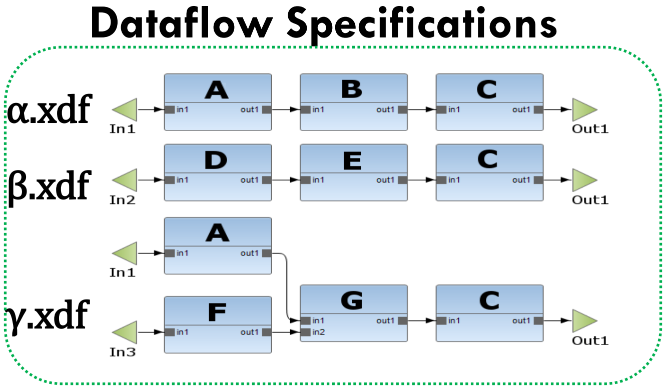
Redundant

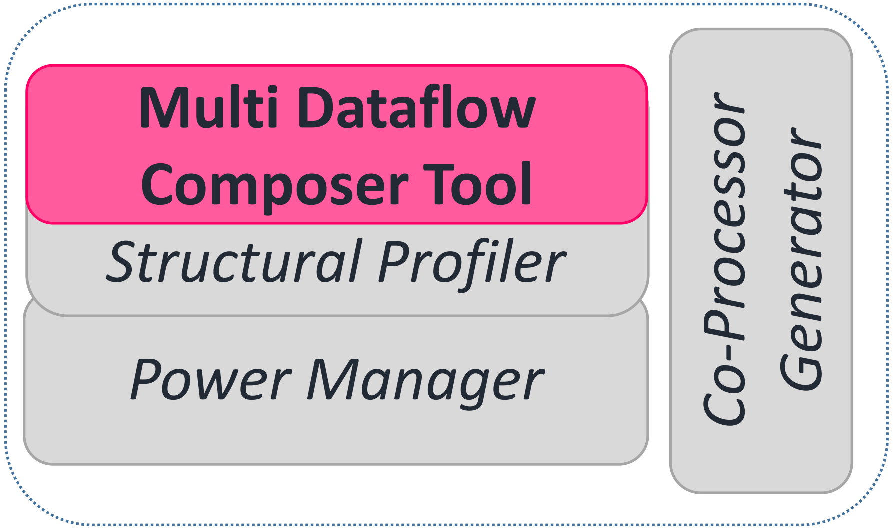


Reduction



MDC tool - Dataflow to HW Mapping

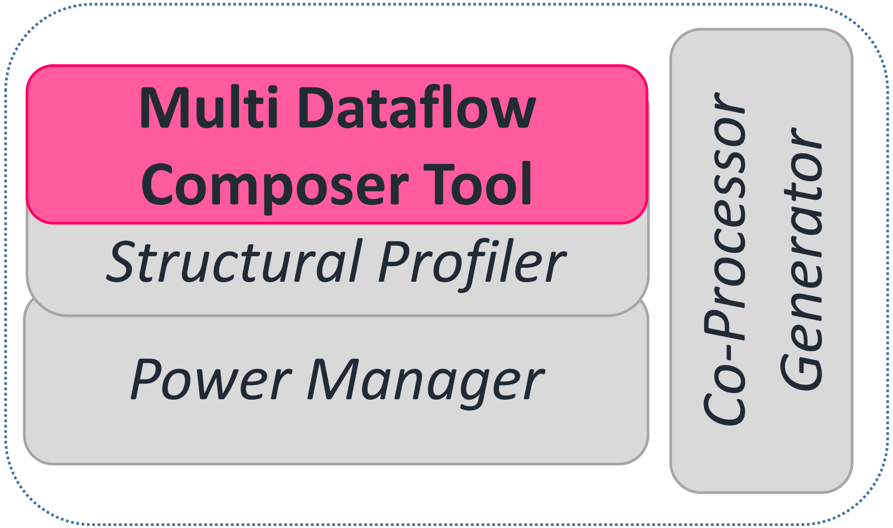




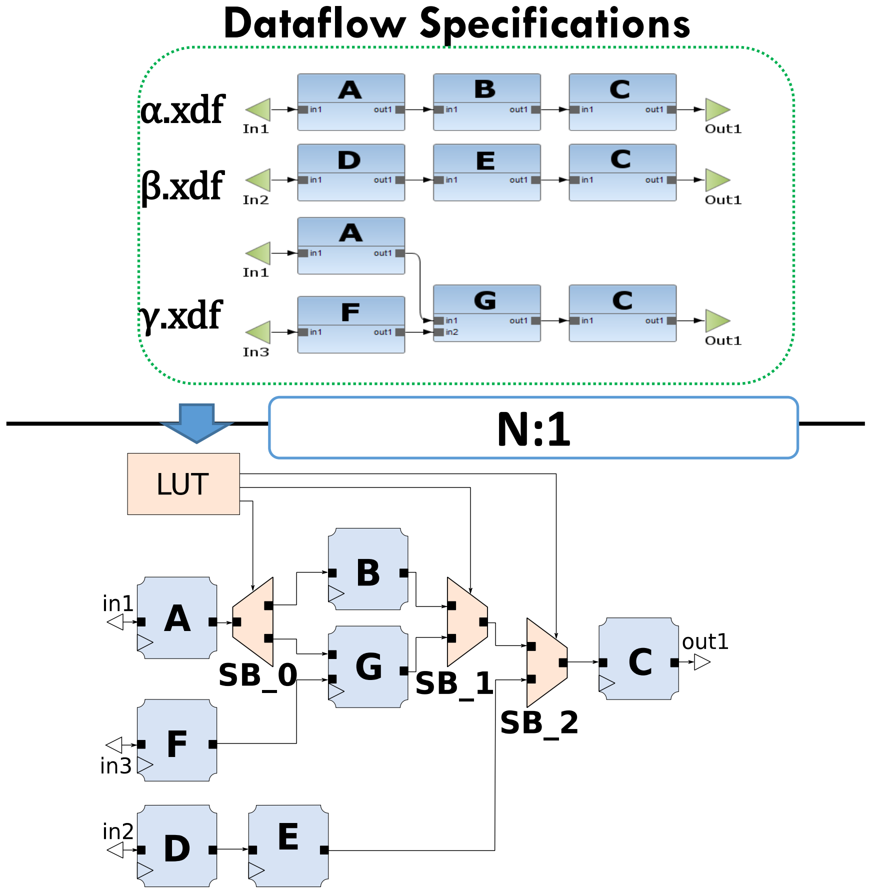
MDC design suite

http://sites.unica.it/rpct/

MDC tool - Dataflow to HW Mapping



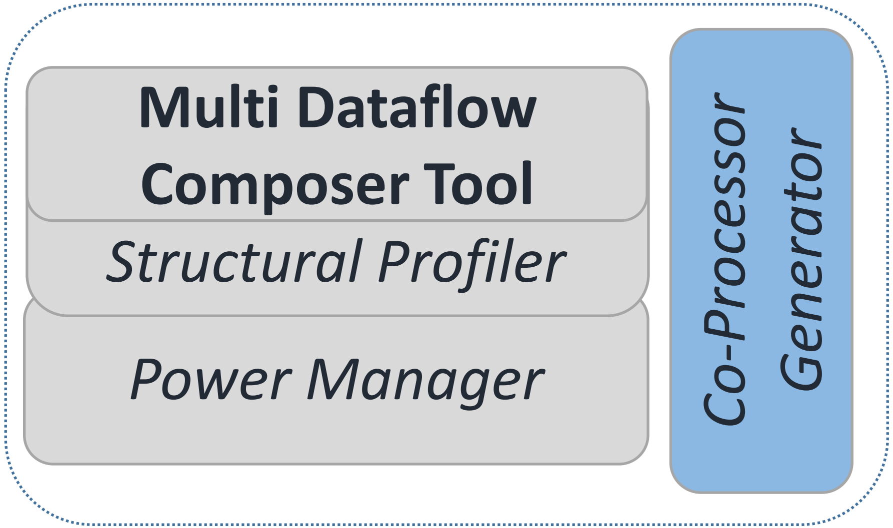
MDC design suite
http://sites.unica.it/rpct/



MDC Tool - Coprocessor Generator

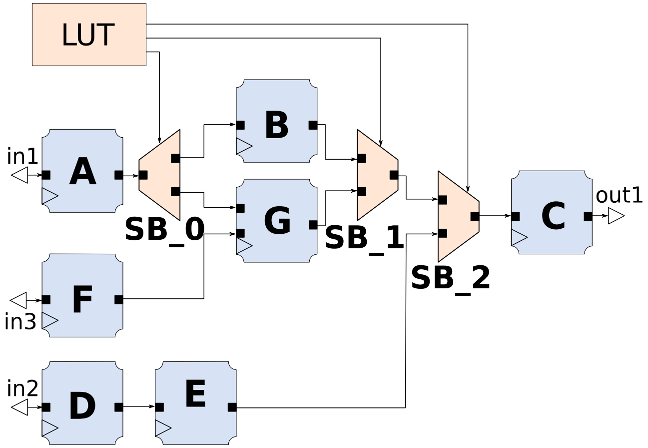
Co-Processor Generator:

generation of ready-to-use Xilinx IPs



MDC design suite

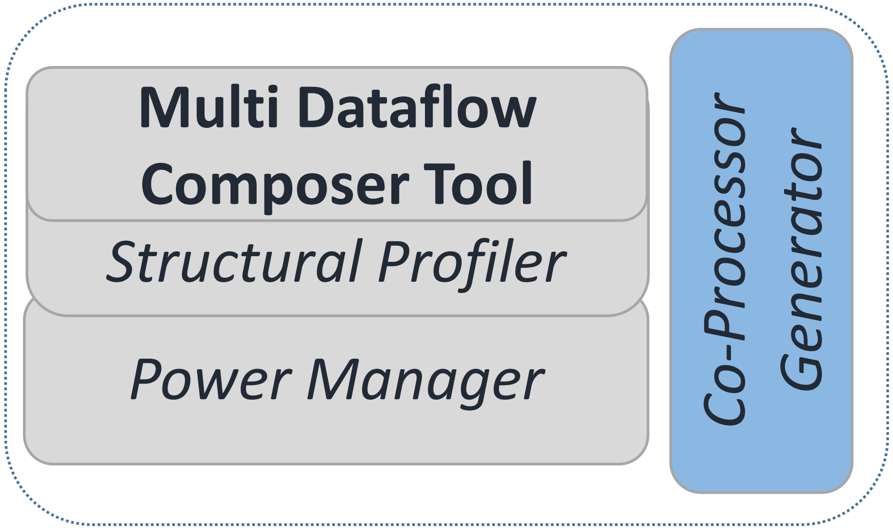
http://sites.unica.it/rpct/



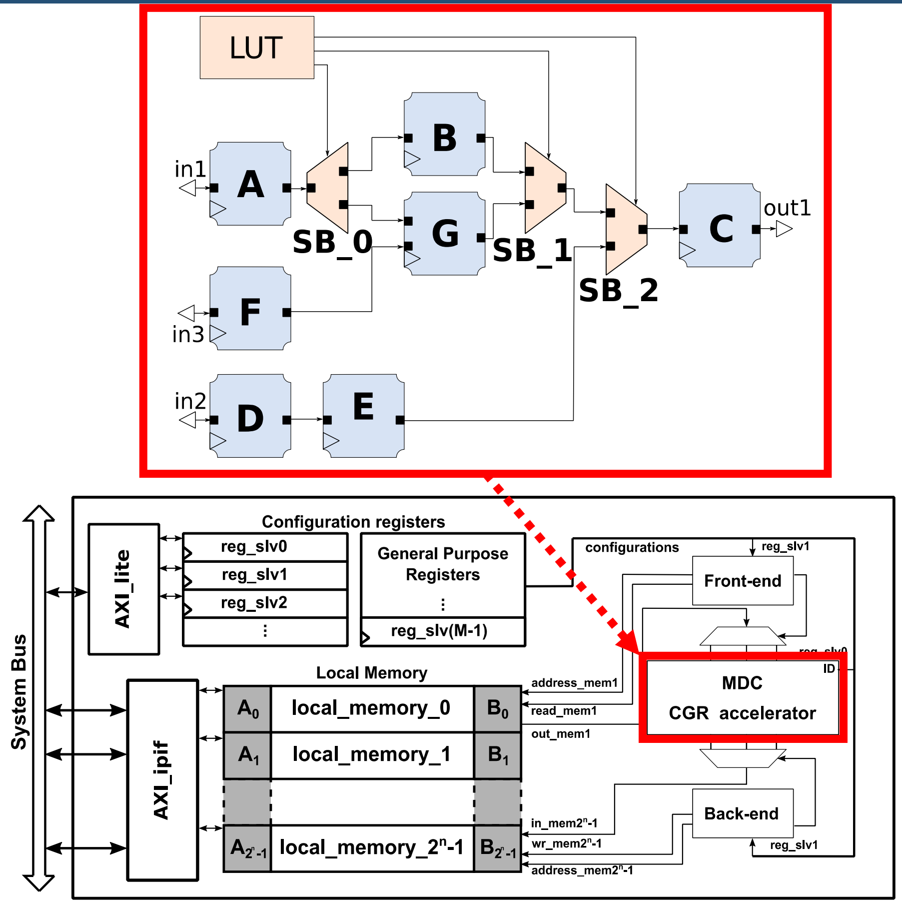
MDC Tool - Coprocessor Generator

Co-Processor Generator:

generation of ready-to-use Xilinx IPs



MDC design suite
http://sites.unica.it/rpct/



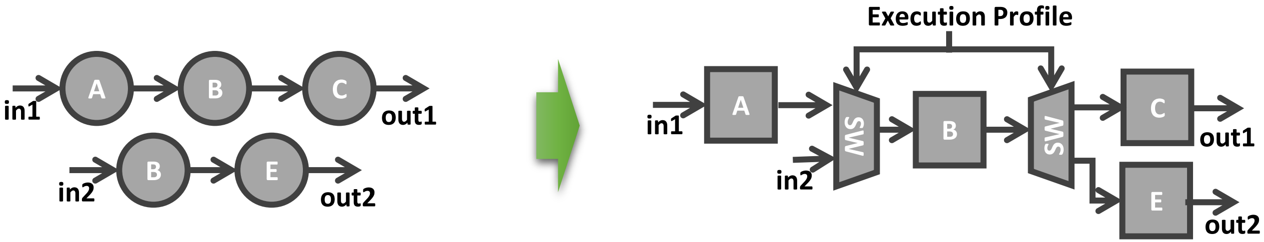
CG Reconfiguration: Adaptation Types in MDC

Functional

Non Functional

CG Reconfiguration: Adaptation Types in MDC

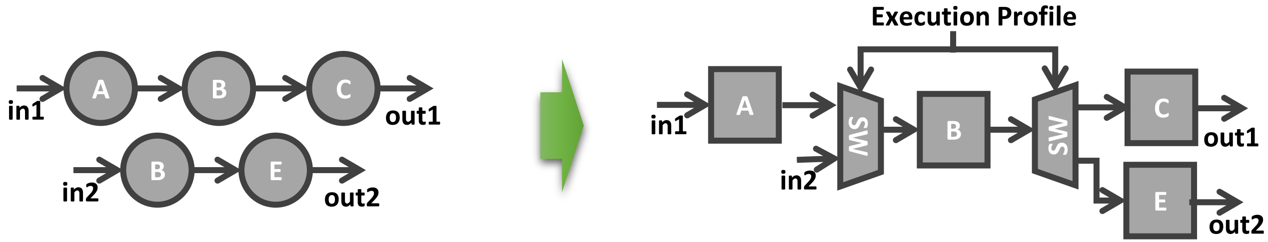
Functional



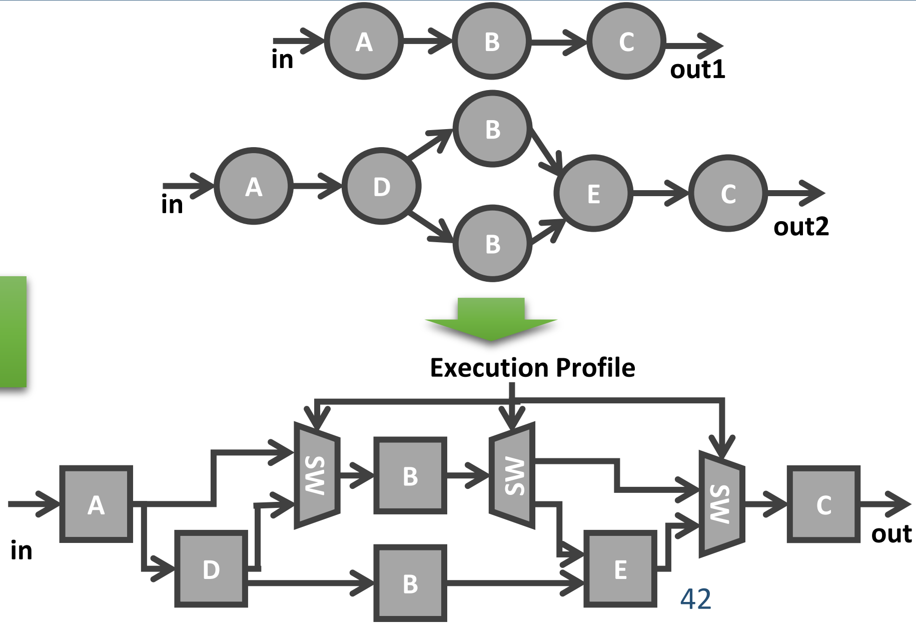
Non Functional

CG Reconfiguration: Adaptation Types in MDC

Functional

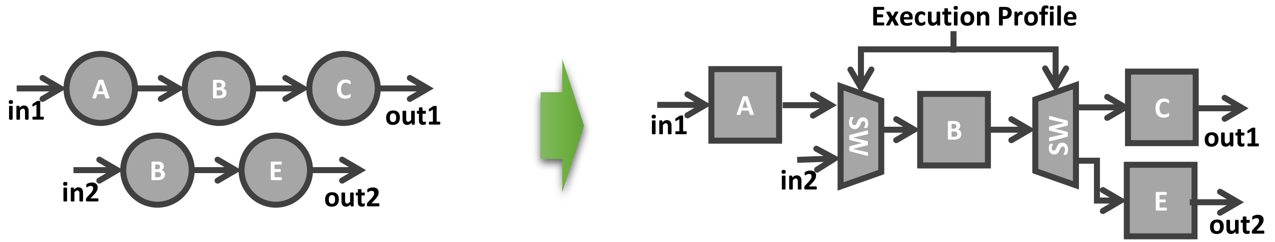


Non Functional

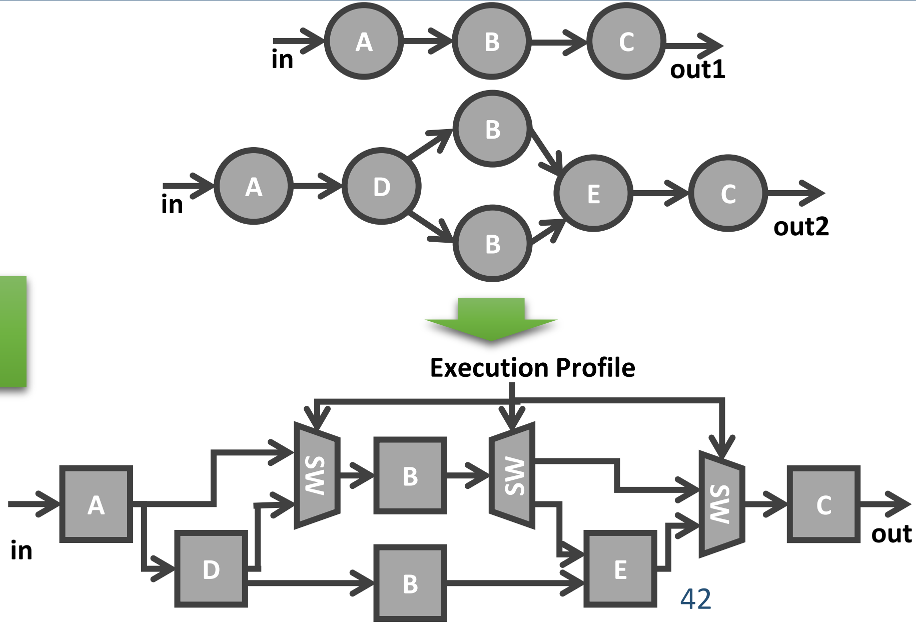


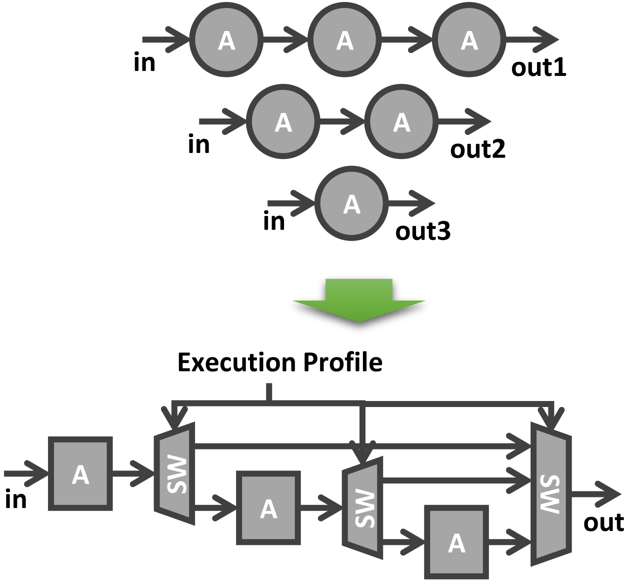
CG Reconfiguration: Adaptation Types in MDC

Functional

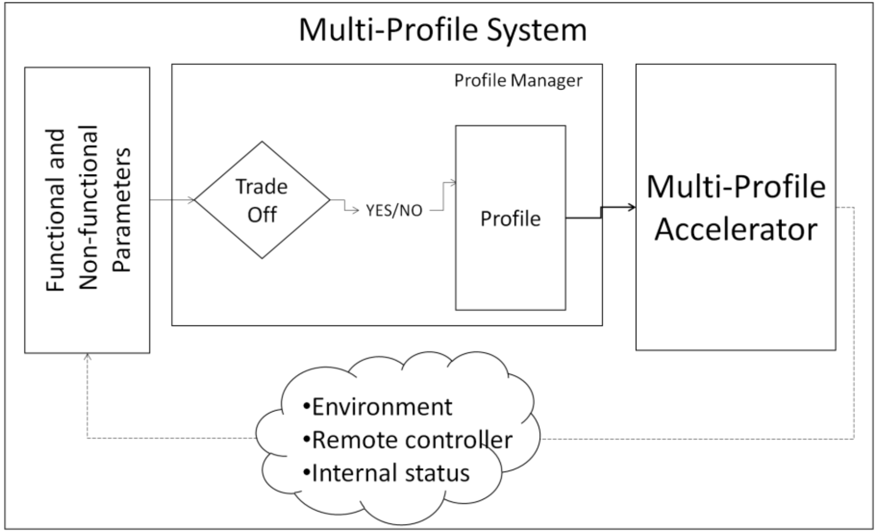


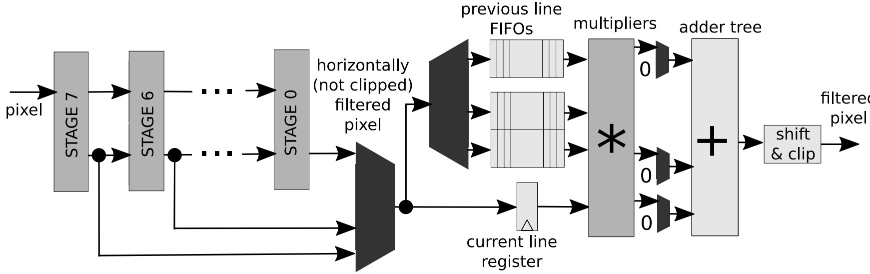
Non Functional





CG Reconfiguration: Runtime KPI Trade-Offs





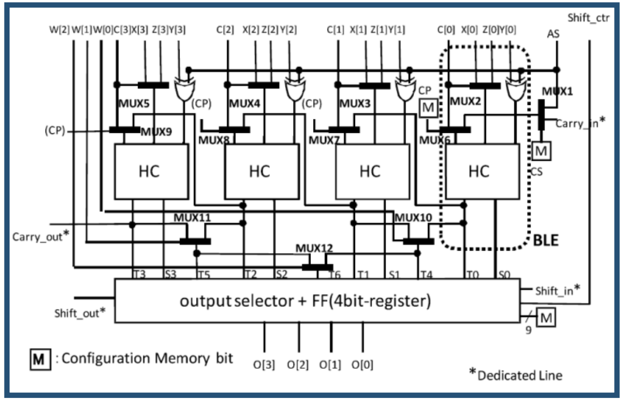
Example of multi-profile CGR system: HEVC interpolator

[ESL17] Carlo Sau, Francesca Palumbo, Maxime Pelcat, Julien Heulot, Erwan Nogues, Daniel Menard, Paolo Meloni, and Luigi Raffo. "***Challenging the Best HEVC Fractional Pixel FPGA Interpolators with Reconfigurable and Multi-frequency Approximate Computing***" in IEEE Embedded Systems Letters, vol. 9, no. 3, pp. 65-68, Sept. 2017.

Outline

- Adaptive systems: Concepts & Definition
  - Triggers and types of Adaptation
  - Levels of autonomy. How to build it
  - The Adaptation Loop
  - An example: Evolvable HW
- Adaptive CPS: The CERBERO approach
  - Big Picture. The CERBERO Adaptation Loops at CPS and CPSoS levels
- Deep Dive into CERBERO HW Adaptation
  - ARTICo3
  - MDC-compliant CG adaptation
  - Mixed-Grain Adaptivity
- **CERBERO Beyond SoA & Take-Out**
  - **Key Advancements and Integration**

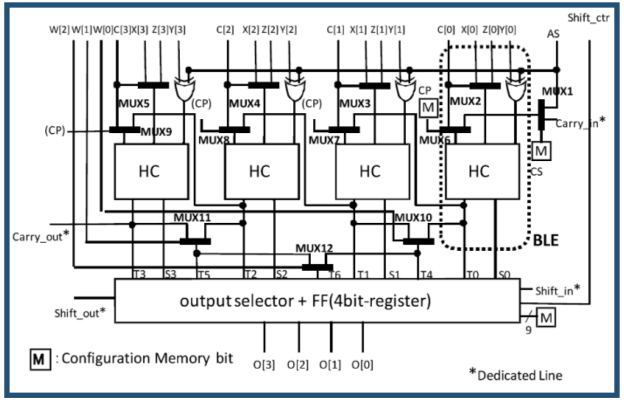
Mixed-Grain Adaptivity: SoA



[K. Inoue, et al. *"A Variable-Grain Logic Cell and Routing Architecture for a Reconfigurable IP Core"*. In ACM Transactions on Reconfigurable Technology and Systems, 2010]

The Variable Grain Logic Cell (VGLC) architecture is based on a ***4-bit*** adder including ***configuration bits***, and can perform operations such as arithmetic logic, random logic, and multiplexing in any application

Mixed-Grain Adaptivity: SoA

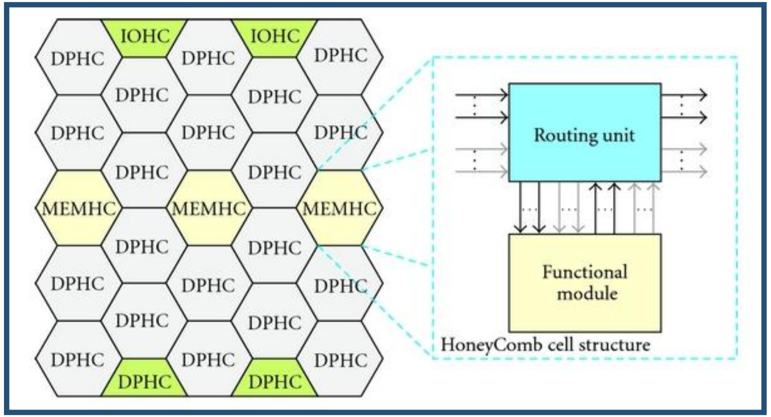


[K. Inoue, et al. *"A Variable-Grain Logic Cell and Routing Architecture for a Reconfigurable IP Core"*. In ACM Transactions on Reconfigurable Technology and Systems, 2010]

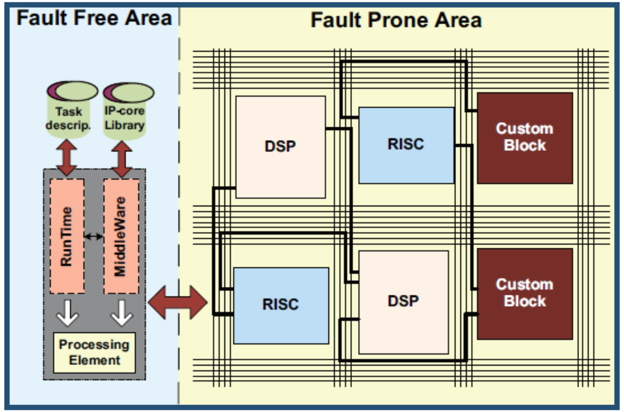
The Variable Grain Logic Cell (VGLC) architecture is based on a ***4-bit*** adder including ***configuration bits***, and can perform operations such as arithmetic logic, random logic, and multiplexing in any application

[A. Thomas, et al. *"HoneyComb: An Application-Driven Online Adaptive Reconfigurable Hardware Architecture"*. In International Journal of Reconfigurable Computing, 2012]

HoneyComb is an adaptable ***dynamically reconfigurable cell array***. Cells are composed of a routing unit and a functional module. Routing units, responsible of connecting neighbours, compose the reconfigurable communication network. Functional modules can be enabled, disabled, or modified ***using DPR***.



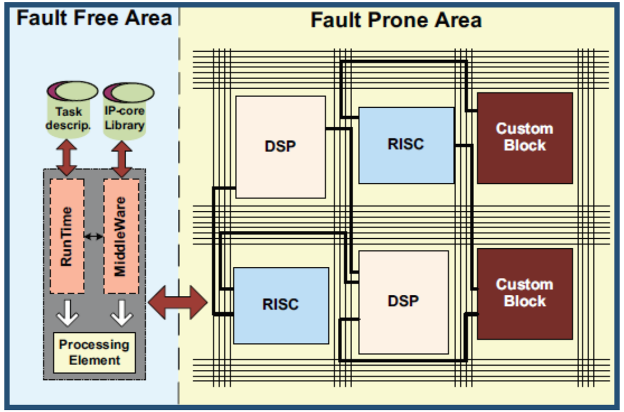
Mixed-Grain Adaptivity: SoA



[I. Sourdis, et al. "*Desyre: On-demand system reliability*". In Microprocessors and Microsystems, 2013].

The DeSyRe Soc contains different sub-components surrounded by reconfigurable interconnects. If a ***fault*** occurs, the sub-component can be ***replaced***: with re-routing, retargeting functionalities on an unused sub-component, or by a functionally equivalent instance implemented in FG reconfigurable hardware.

Mixed-Grain Adaptivity: SoA

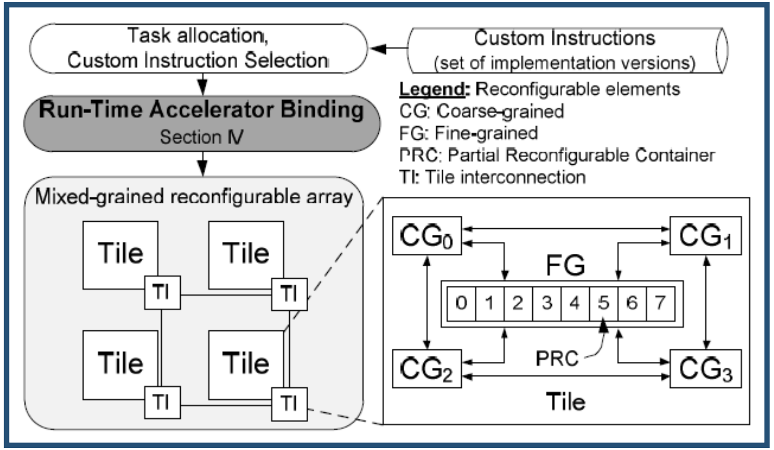


[I. Sourdis, et al. "*Desyre: On-demand system reliability*". In Microprocessors and Microsystems, 2013].

The DeSyRe Soc contains different sub-components surrounded by reconfigurable interconnects. If a ***fault*** occurs, the sub-component can be ***replaced***: with re-routing, retargeting functionalities on an unused sub-component, or by a functionally equivalent instance implemented in FG reconfigurable hardware.

[C. M. Diniz, et al. "*Run-Time Accelerator Binding for Tile-Based Mixed-Grained Reconfigurable Architectures*". Conference on Field Programmable Logic and Applications, 2014]

Mixed-grained reconfiguration is used within the tiles of tile-based processor. Each tile consists of ***multiple CG and FG reconfigurable elements***.



Self-Adaptation & Mixed-Grain: Beyond SoA

Self-Adaptation & Mixed-Grain: Beyond SoA

Partially reconfigurable CG arrays, with identical Processing Elements

CERBERO Mixed-Grain Support
ARTICo^3 + MDC

Partially reconfigurable slots of the FPGA (ARTICo^3 compliant) filled with heterogeneous application specific CG datapaths (MDC compliant).

CERBERO

Self-Adaptation & Mixed-Grain: Beyond SoA

Partially reconfigurable CG arrays, with identical Processing Elements

CERBERO Mixed-Grain Support ARTICo^3 + MDC

Partially reconfigurable slots of the FPGA (ARTICo^3 compliant) filled with heterogeneous application specific CG datapaths (MDC compliant).

Lack of self-adaptivity support for heterogeneous environment

CERBERO Self-Adaptation Manager

Build proper hardware abstractions fed with real time monitored and sensed data, to enable self-adaptive behaviours.

CERBERO

Self-Adaptation & Mixed-Grain: Beyond SoA

Partially reconfigurable CG arrays, with identical Processing Elements

Lack of self-adaptivity support for heterogeneous environment

Lack of frameworks to partition functionalities and design Processing Elements

CERBERO Mixed-Grain Support ARTICo^3 + MDC

Partially reconfigurable slots of the FPGA (ARTICo^3 compliant) filled with heterogeneous application specific CG datapaths (MDC compliant).

CERBERO Self-Adaptation Manager

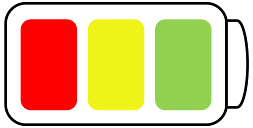
Build proper hardware abstractions fed with real time monitored and sensed data, to enable self-adaptive behaviours.

CERBERO Tool Set

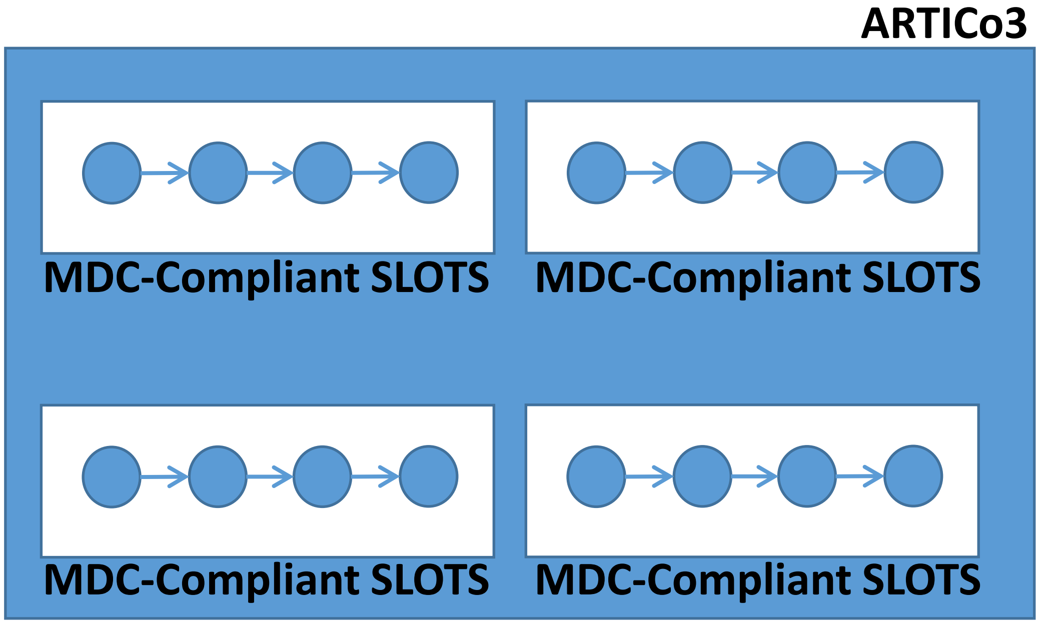
CERBERO Tool Set is specifically conceived to support designers in the different phases of deployment, from partitioning (DSE and automatic mapping/scheduling) to customization (HLS and deployment)

CERBERO

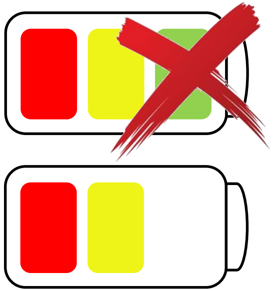
Mixed-Grain: The Best of Both



Max Troughput
Max QoS

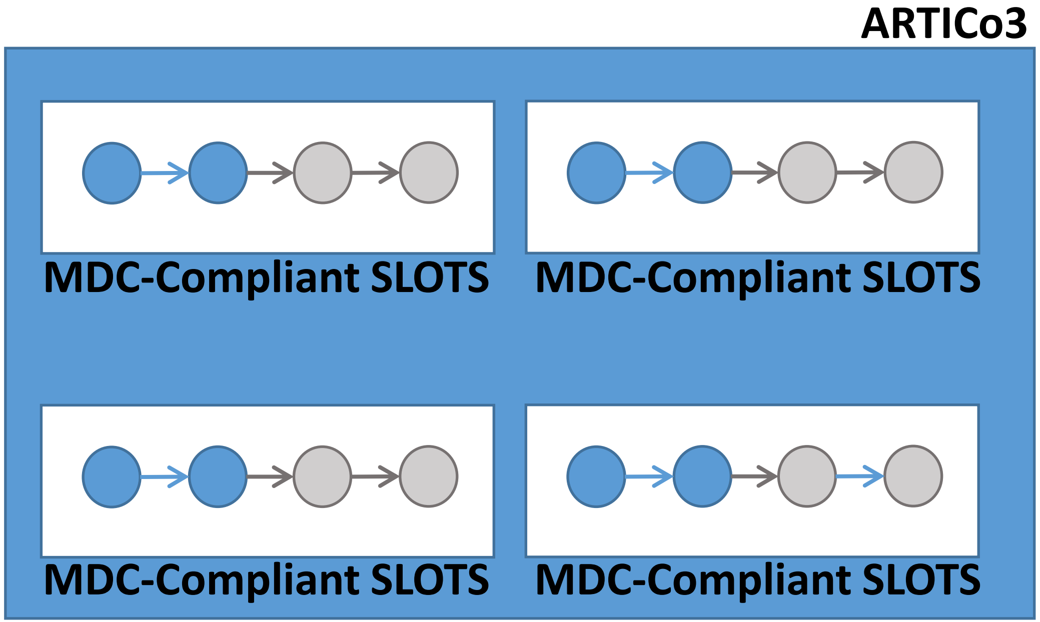


Mixed-Grain: The Best of Both

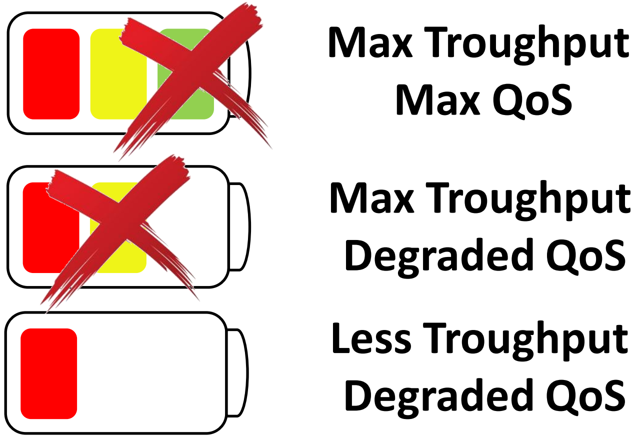


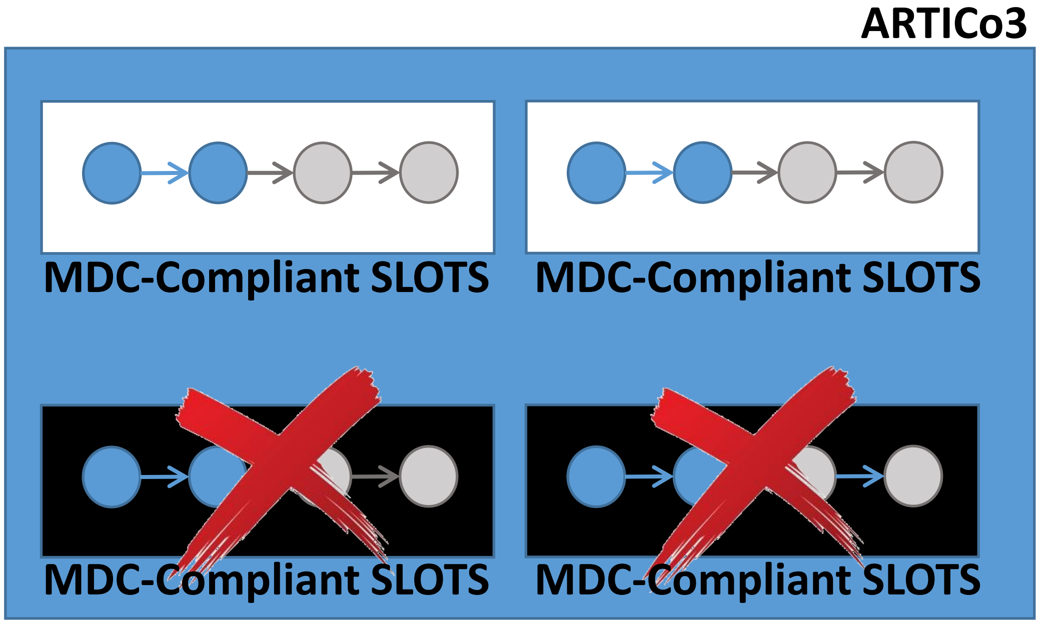
Max Troughput
Max QoS

Max Troughput
Degraded QoS

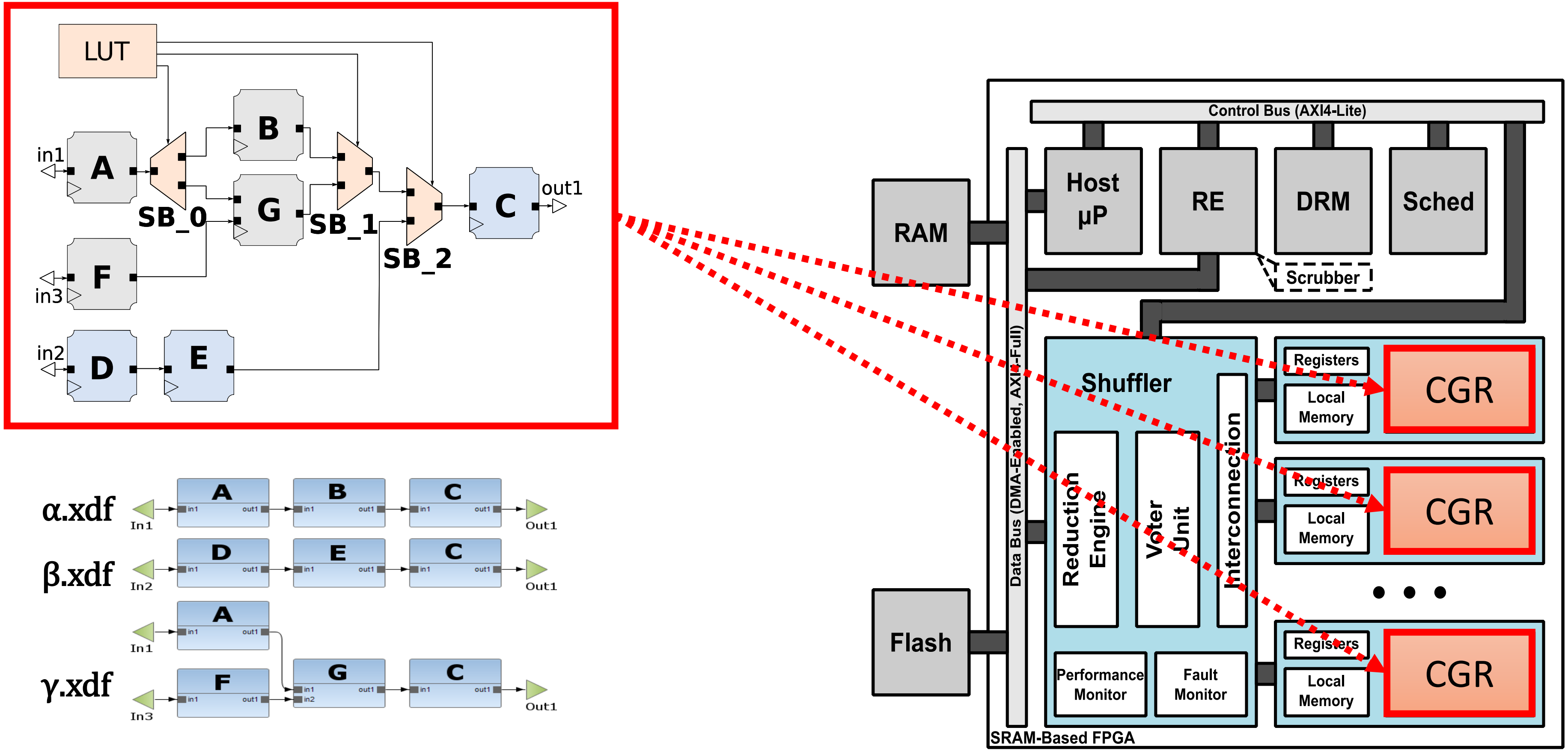


Mixed-Grain: The Best of Both



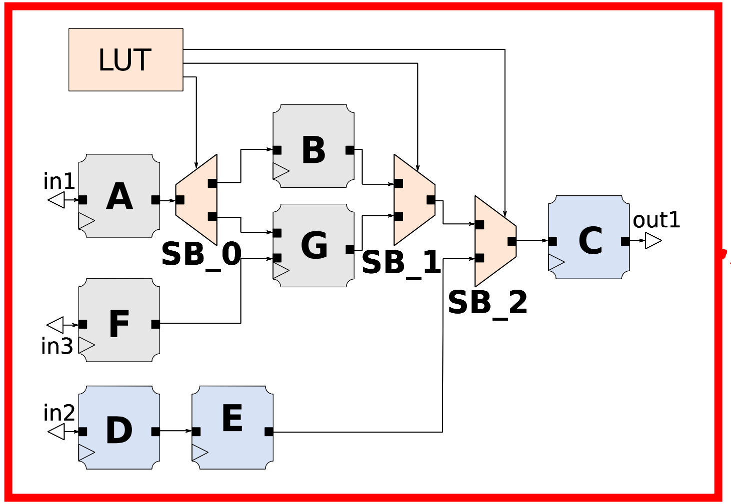


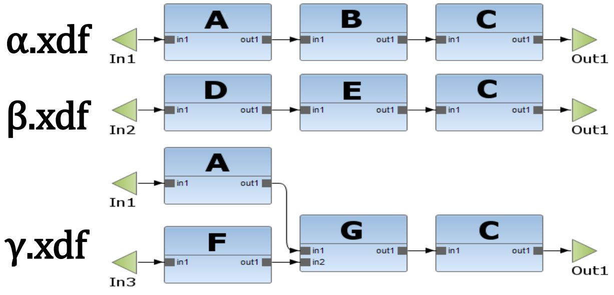
Multi-Grain Adaptivity

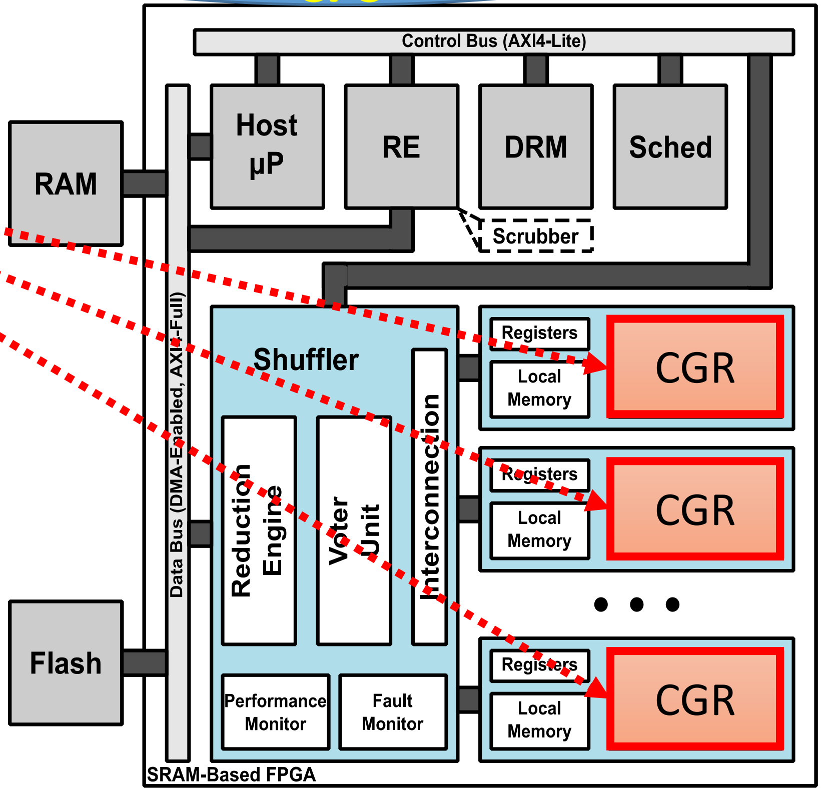


Multi-Grain Adaptivity

Tutorial: Multi-Grain Reconfiguration for Advanced Adaptivity in CPS

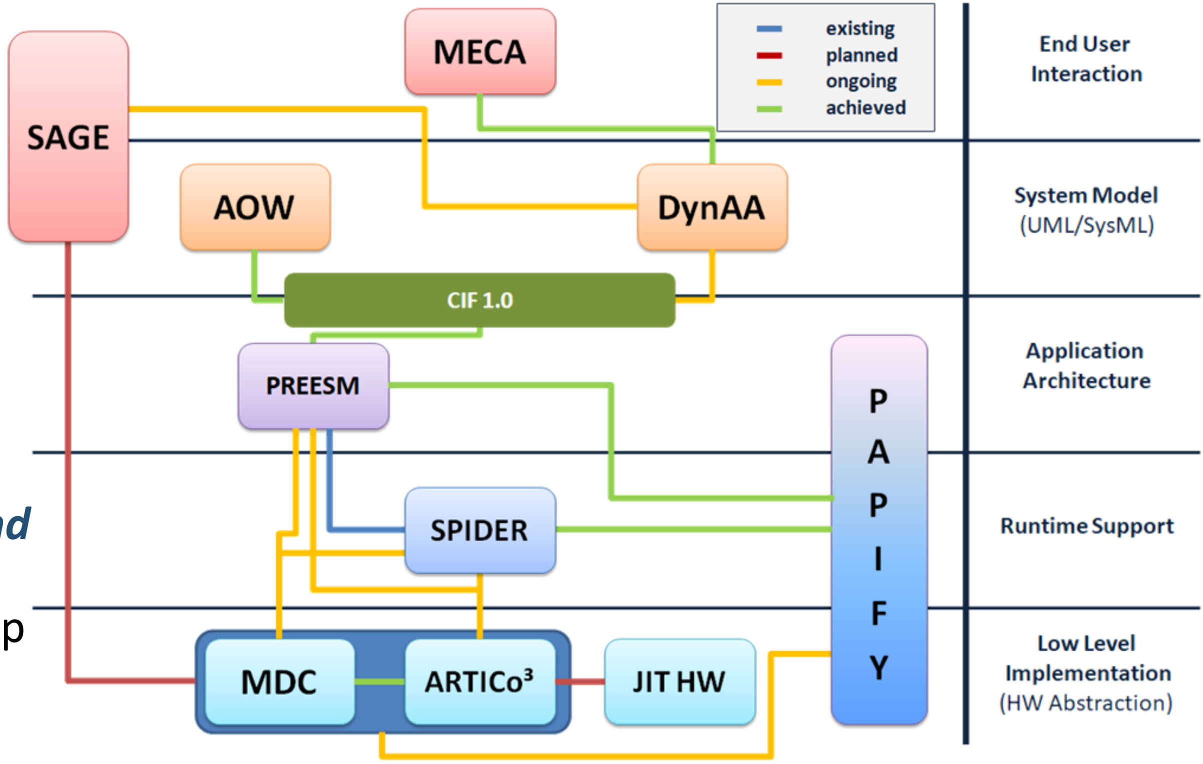






Adaptivity Support: CERBERO Tool Set

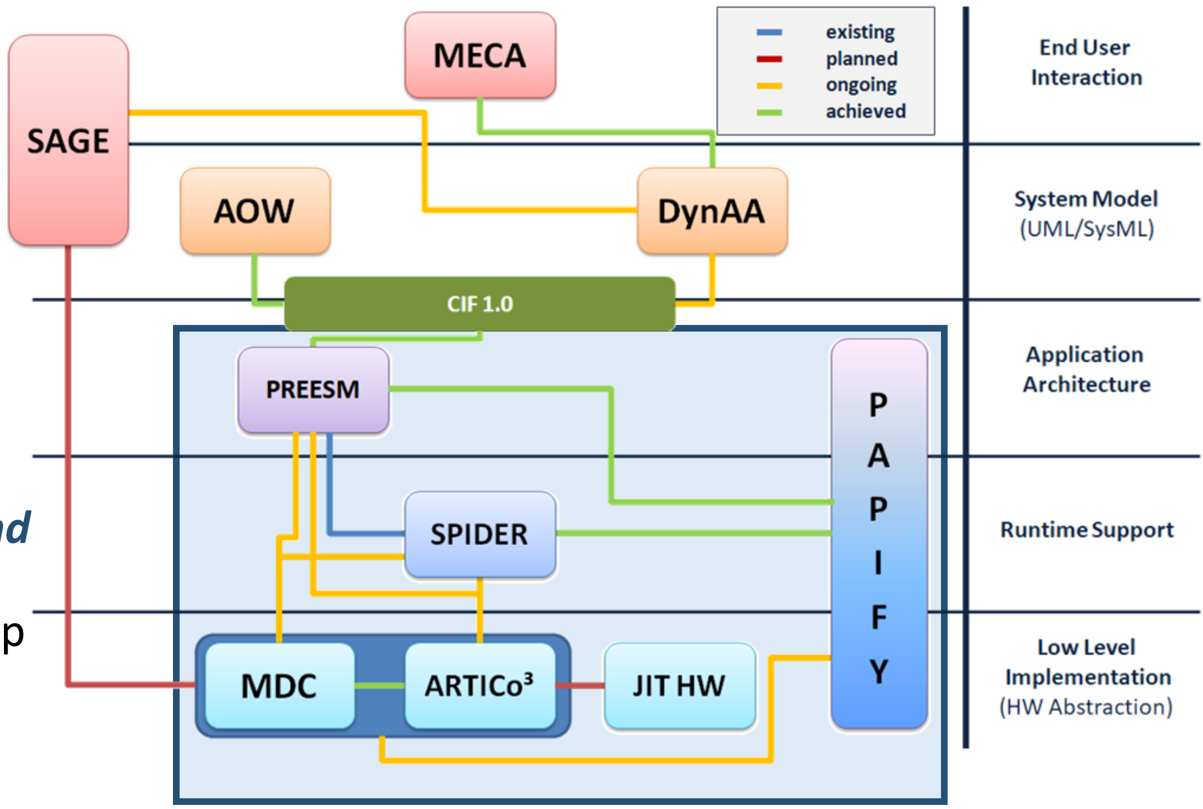
- CERBERO Tool Set
  - Design environment to ***model, explore, deploy and verify*** complex ***adaptive CPS***
  - Address the lack of integrated toolchains capable of:
    - Spanning across layers
    - ***Dealing with adaptivity and heterogeneity***
    - Providing system in the loop co-simulation



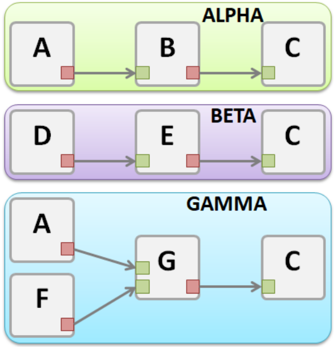
CERBERO

Adaptivity Support: CERBERO Tool Set

- CERBERO Tool Set
  - Design environment to ***model, explore, deploy and verify*** complex ***adaptive CPS***
  - Address the lack of integrated toolchains capable of:
    - Spanning across layers
    - ***Dealing with adaptivity and heterogeneity***
    - Providing system in the loop co-simulation



Design-Time Support 4 HW Run-Time Adaptivity



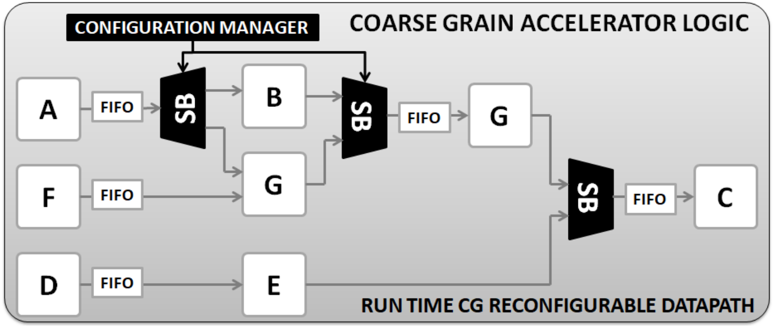
N

PREESM

+

MDC

1

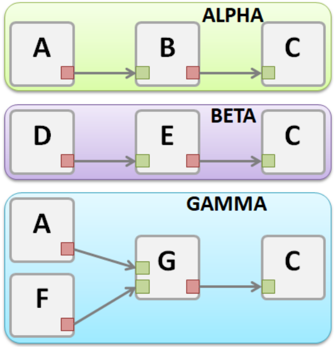


Partitioning

+

CGR Accelerator Synthesis

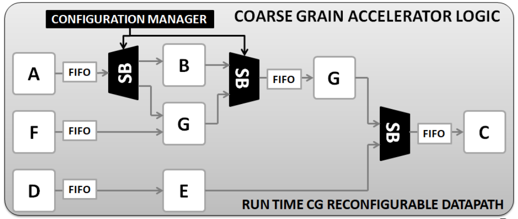
Design-Time Support 4 HW Run-Time Adaptivity



N

PREESM
+
MDC

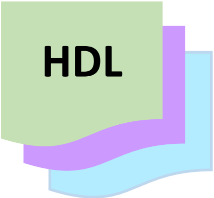
1



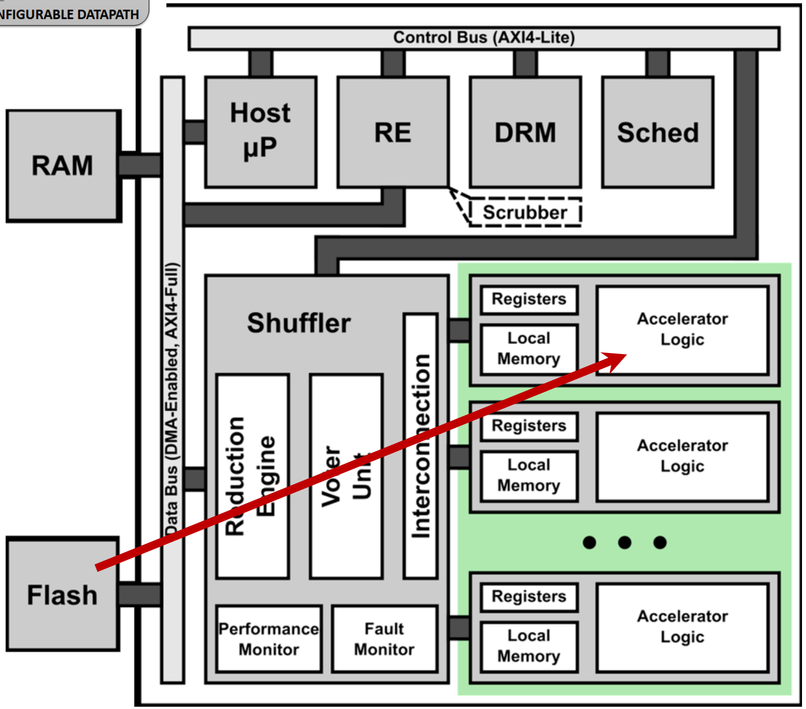
N

PREESM
+
SDSoC

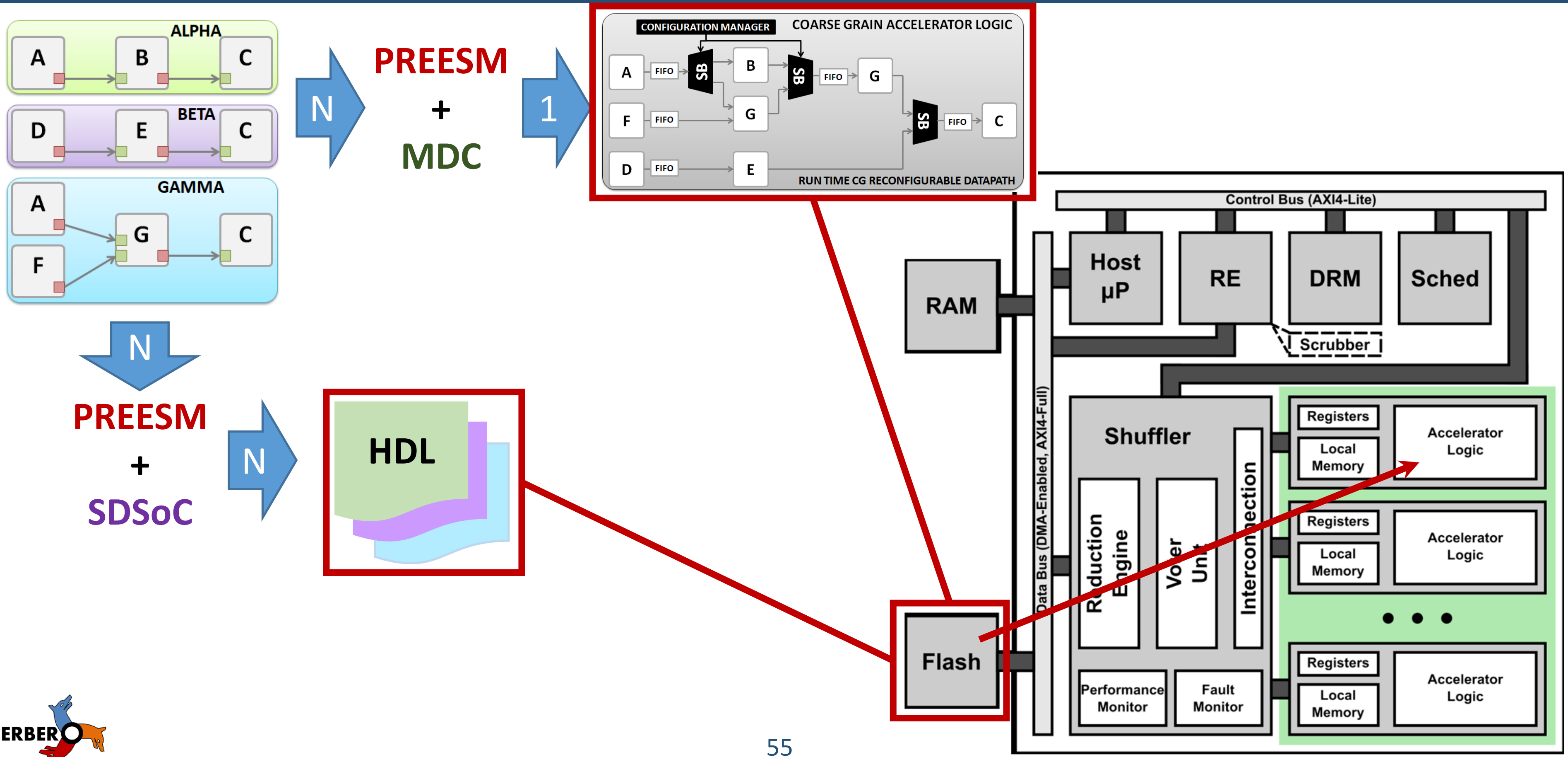
N



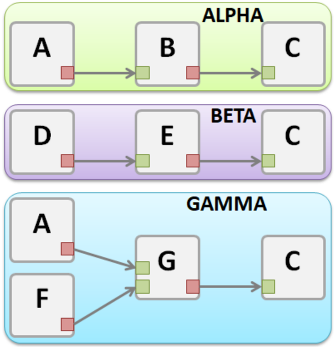
Partitioning
+
Slots Synthesis



Design-Time Support 4 HW Run-Time Adaptivity



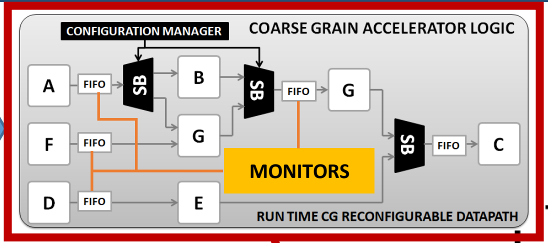
Design-Time Support 4 HW Run-Time Adaptivity



N

PREESM
+
MDC
+
PAPIFY-DT

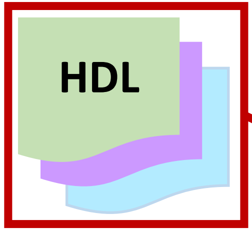
1

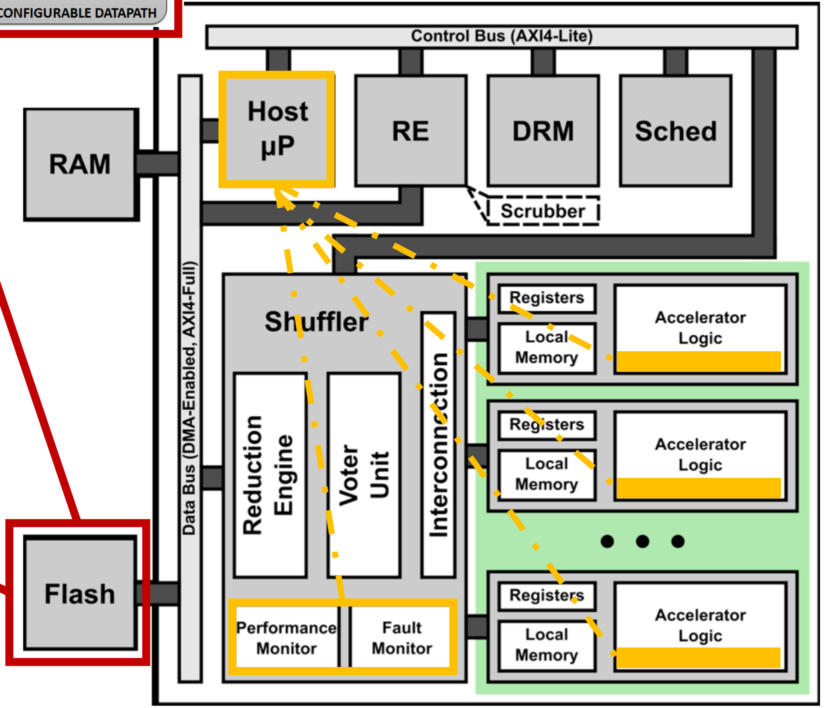


N

PREESM
+
SDSoC
+
PAPIFY-DT

N

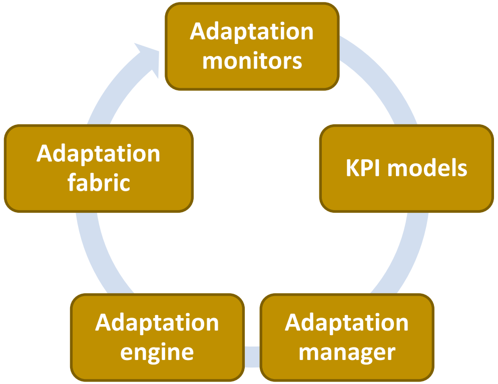




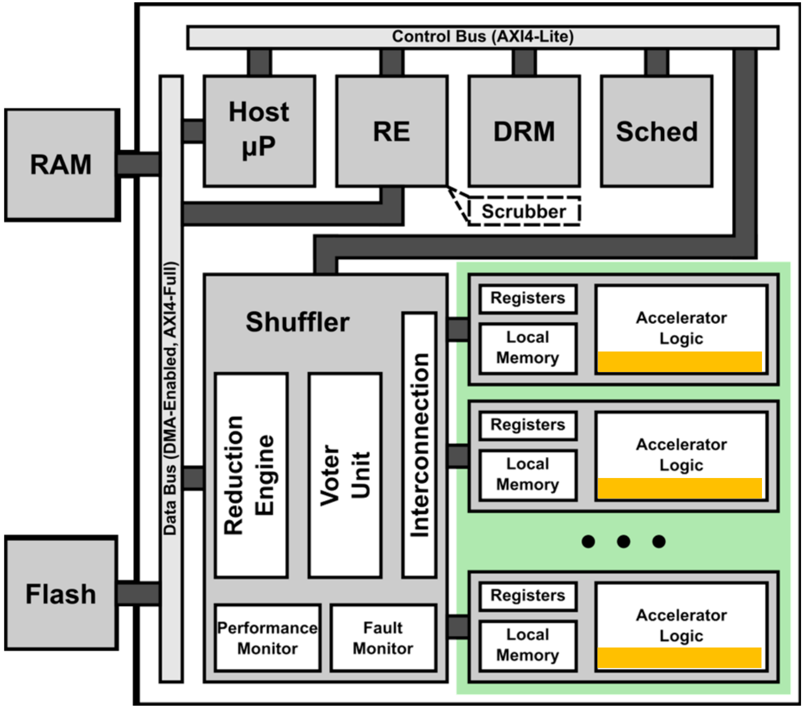
CERBERO

Runtime Performance Monitoring

HW Run-Time Adaptivity



ACT

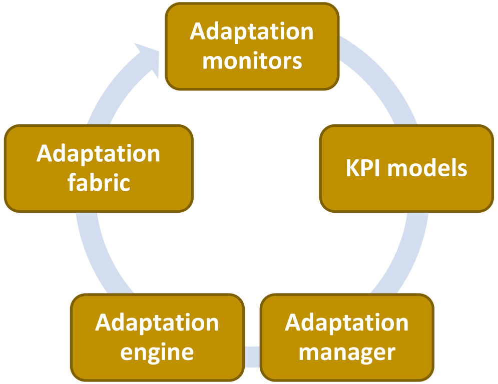


HW Run-Time Adaptivity

PAPIFY-RT

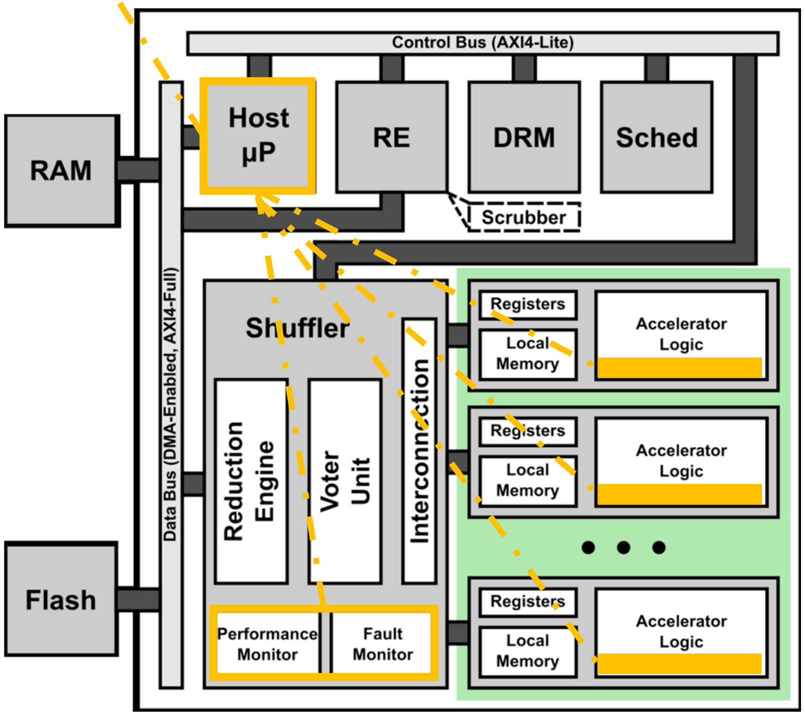
+

PAPIFY-VIEWER

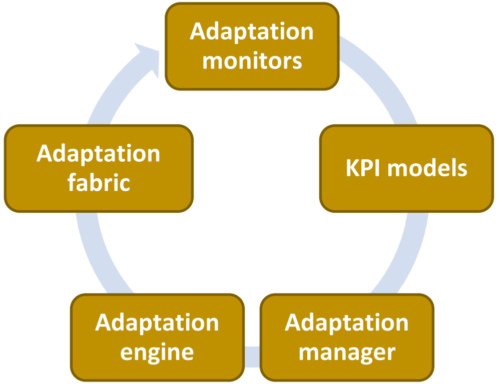


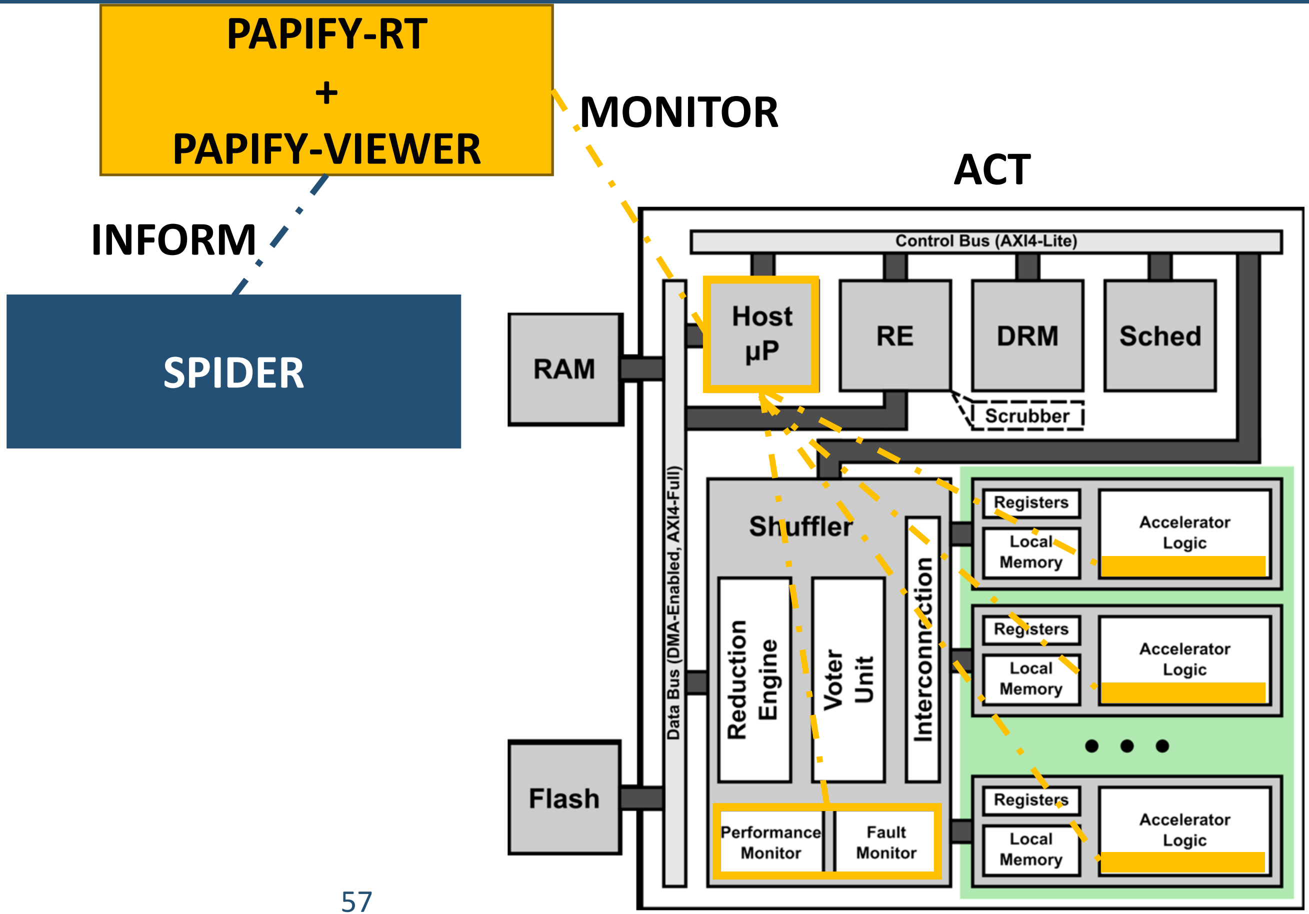
MONITOR

ACT

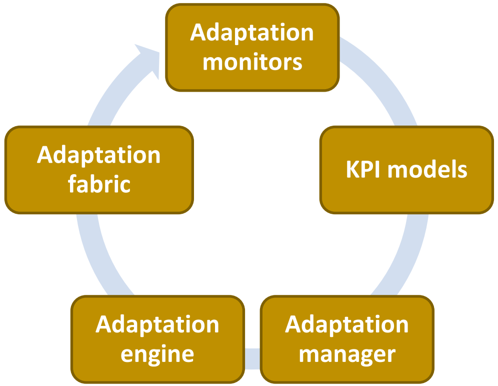


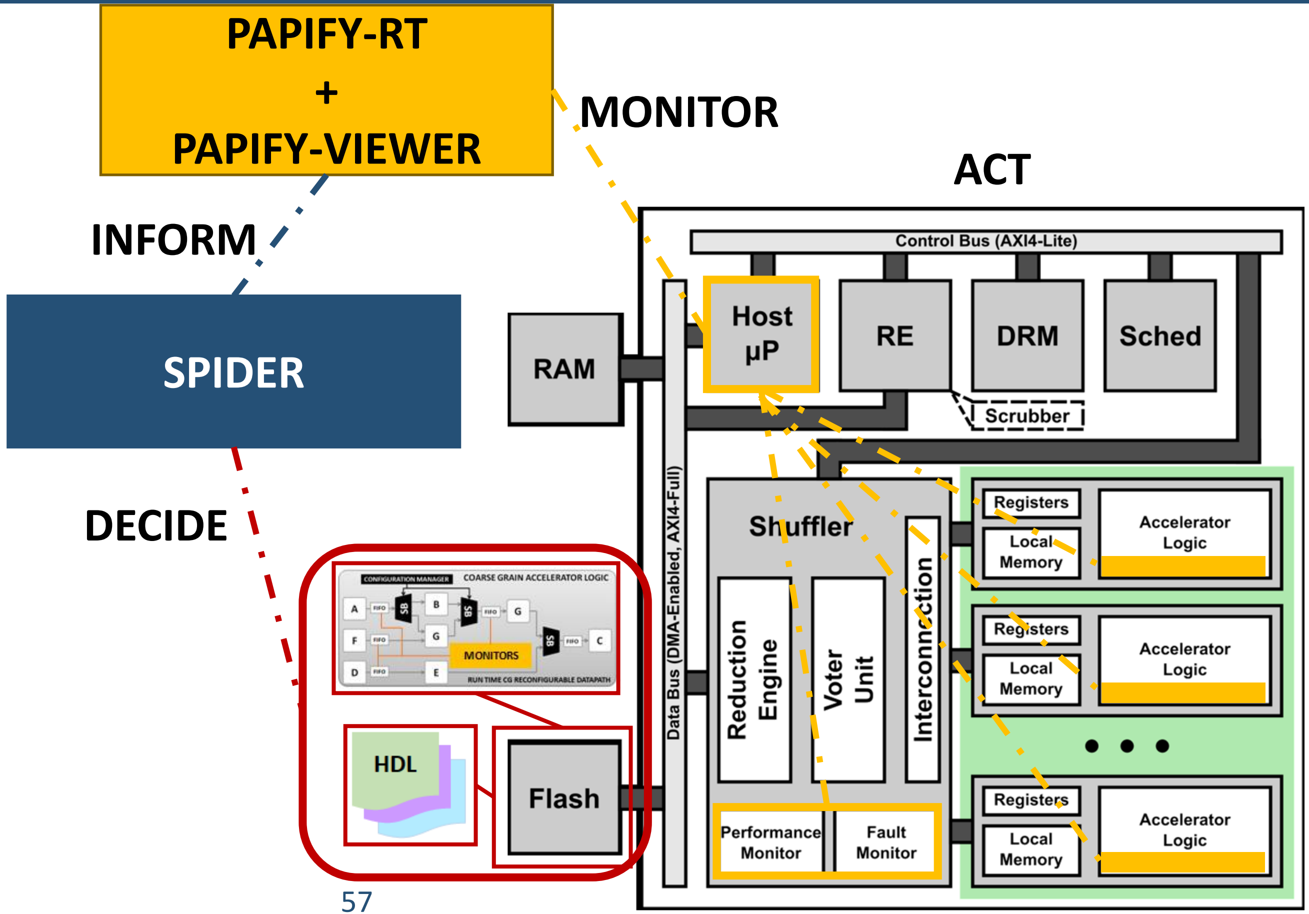
HW Run-Time Adaptivity





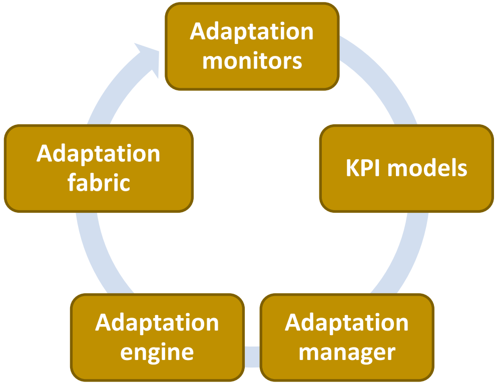
HW Run-Time Adaptivity

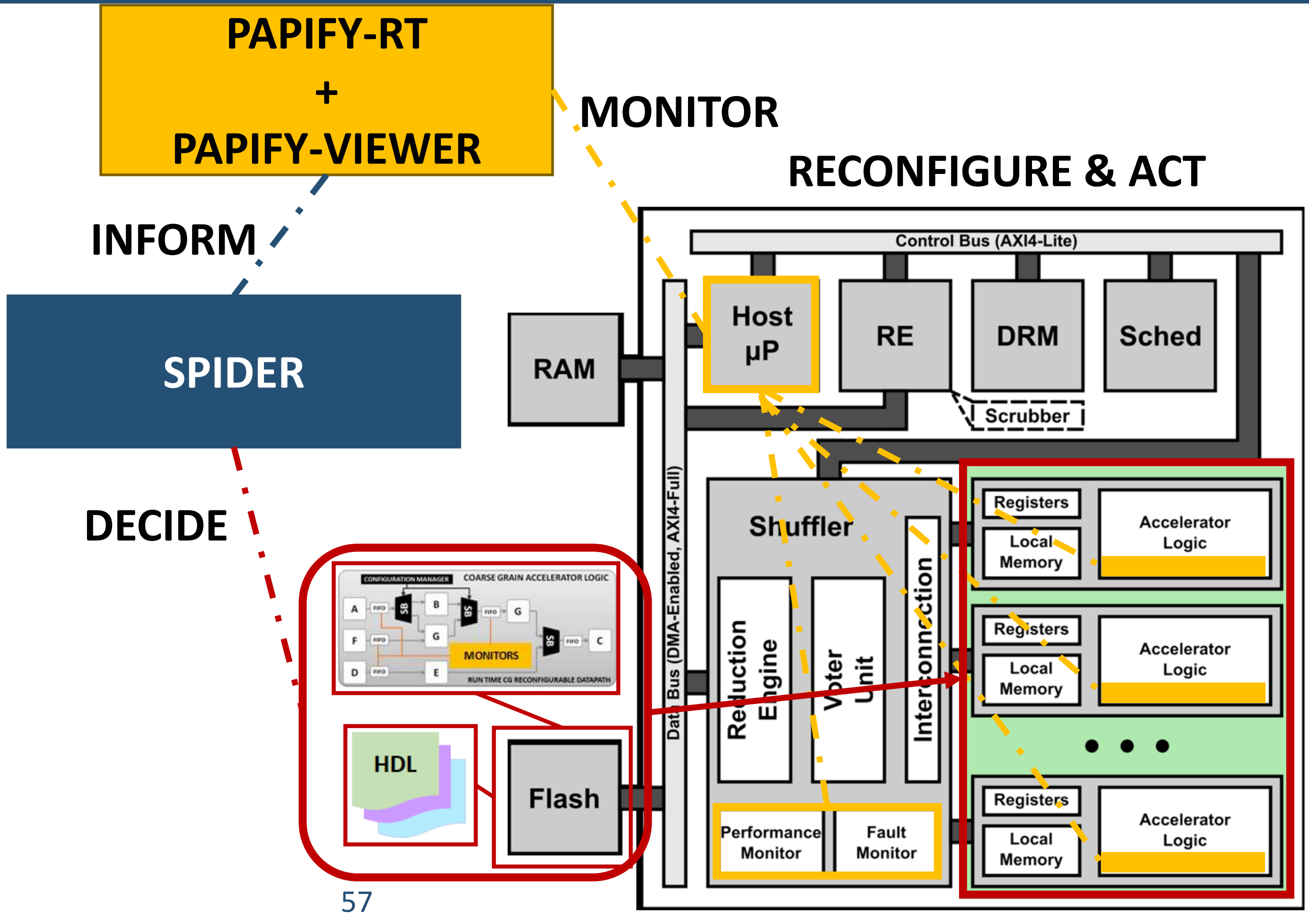




CERBERO

HW Run-Time Adaptivity





CERBERO

Self-Adaptivity Challenges & CERBERO solutions

Definition of Complex Adaptive CPS

Self-Adaptivity Challenges & CERBERO solutions

Definition of Complex Adaptive CPS

Classify Components/Layers

Identify the Common Ground

Self-Adaptivity Challenges & CERBERO solutions

Definition of Complex Adaptive CPS → Classify Components/Layers → Identify the Common Ground

Adaptation monitors

KPI models

Adaptation manager

Adaptation engine

Adaptation fabric

Generic Adaptation Loop

CERBERO

Self-Adaptivity Challenges & CERBERO solutions

Definition of the Adaptive Infrastructure

Self-Adaptivity Challenges & CERBERO solutions

Definition of the Adaptive Infrastructure

Implementation of an Heterogeneous Cross-Layer Infrastructure

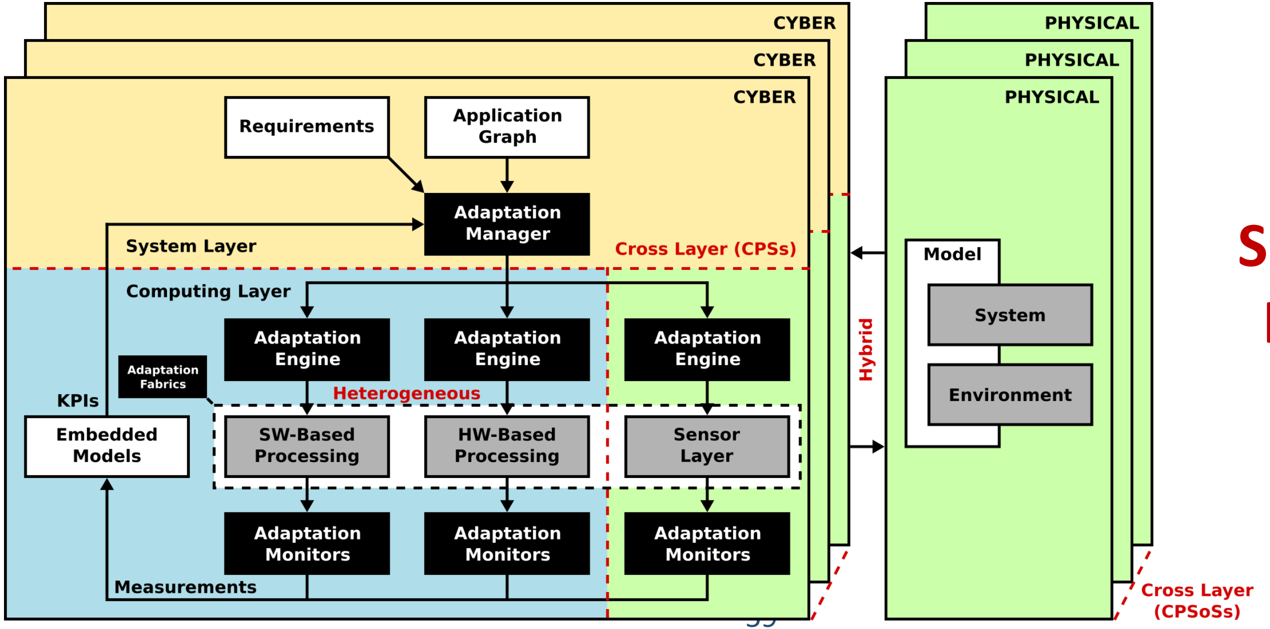
Support Self-Adaptiveness

Self-Adaptivity Challenges & CERBERO solutions

Definition of the Adaptive Infrastructure

Implementation of an Heterogeneous Cross-Layer Infrastructure

Support Self-Adaptiveness



CERBERO Self-Adaptation Infrastructure

Self-Adaptivity Challenges & CERBERO solutions

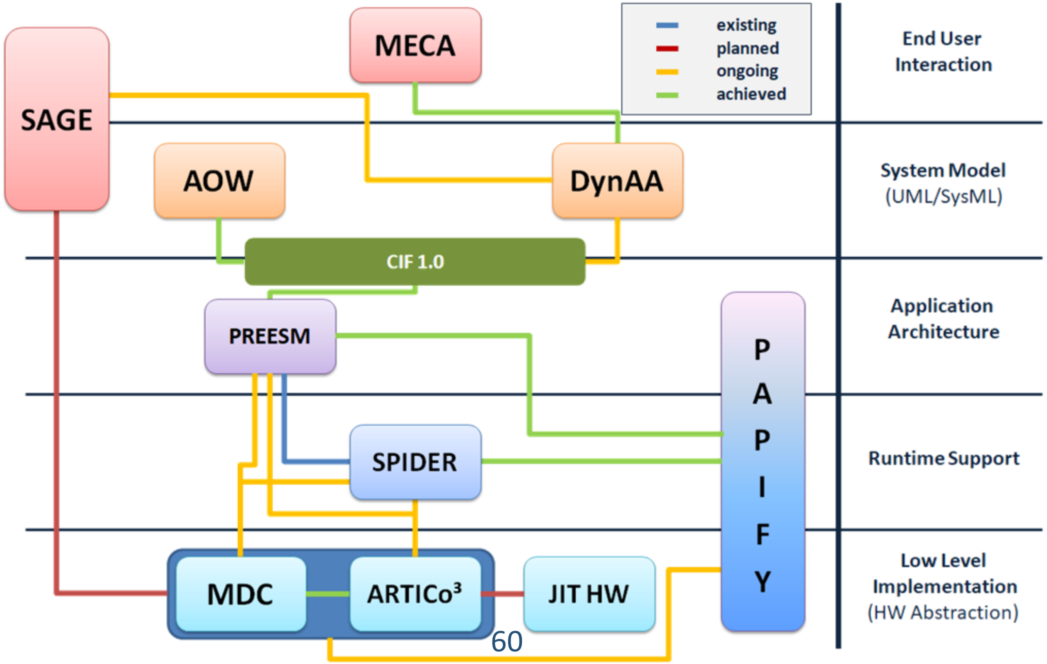
Management of Adaptive Infrastructure

Self-Adaptivity Challenges & CERBERO solutions

Management of Adaptive Infrastructure

Modelling Adaptive CPS & KPIs

Offer Design- & Run- Time Support



CERBERO Tool Set