MULTI-LEVEL CONTROL OF RECONFIGURABLE ARCHITECTURES FOR AUTONOMOUS VEHICLES

Jean-Philippe Diguet
CNRS, Lab-STICC
Lorient / Brest, France

Eric Rutten
INRIA, Rhône Alpes
Grenoble, France

HPeC – Project

Self-Adaptive SoC for High-Performance Embedded Computing

- UAV case study -

- **French ANR Funding 2016/20**
- **Partners:**
  - **CNRS** / Lab-STICC (Lorient/Brest): *J-Ph. Diguet (CNRS), C. Dezan (UBO), C. Hireche (PhD)* [Archi., Tools, Self-Adaptation]
  - **INRIA** Rhône-Alpes (Grenoble): E. Rutten (INRIA), S. Mak-Karé Gueye (postdoc) [Automata Synthesis]
  - **Gipsa-Lab** (Grenoble): S. Mocanu (INP) [Control and Stochastic Methods]
  - **UCA** Univ. /Institut Pascal (Clermont-Ferrand): F. Berry (UCA), M.Pelcat (UCA), E.M Abdali (PhD), [Image, Architecture, FPGA]
  - **INPiXal**, SME (Rennes): L. Fangain, S. Buriau [Smart Cam]

- **HPeC "Friends":**
  - **QUT** (Bribsane, Australia), Prof. Luis Mejias
  - **USP** (Sao Carlos, Brazil), Prof. Kalinka Branco,

Outline

1. Motivations

(Reconfigurable Computing + Autonomous Vehicles + Robust Control)

2. HPeC: Global Multi-Layer Approach

3. Formal Reconfiguration Control

4. First Results from the UAV Case study

1-Motivations (1)

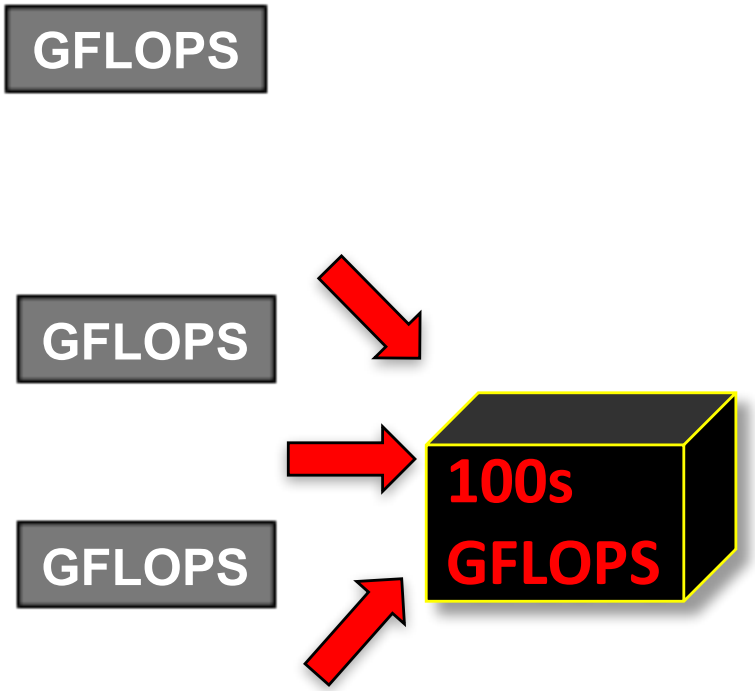
- **Autonomy means Embedded HPC**
  1. **To Observe the Environment** , e.g. Image Processing, Radar
  2. **To Make decision**, e.g. mission management
  3. **Latency = strong constraint => Local not Remote,** e.g. collision avoidance

1-Motivations

- **Use case of UAV** with multiple sensors (#cameras, radar, IR, US)
- 4 Classes of Tasks:
  - Flight Control
    - UKF / EKF
  - Navigation / Guidance
    - Egomotion
    - SLAM
    - Path-Planning

    …
  - Safety
    - Obstacle avoidance
    - On flight emergency landing areas detection
    - Health Management / Fault detection

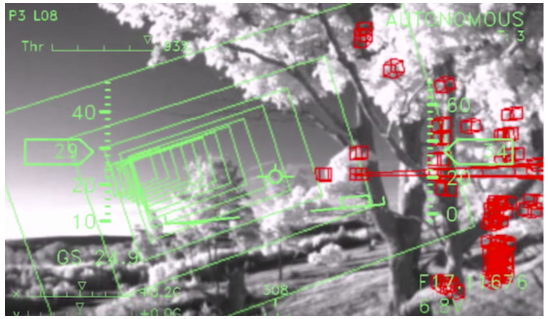
    …
  - Mission / Application
    - Mission Planning (multi-objective optimization)
    - Target detection and Tracking
    - Discovery and classification (CNN)
    - Encryption
    - …



1-Motivations

- **Examples**
  - Vision
    - MIT (15) : Obstacle Avoidance@50Km/h,

      2 cameras, 2 smartphone Quadcores.



    - QUT/ARCAA (12): Sense and avoid, GPU

    - KTH (15) : Xilinx Zinq, 4 cameras, FPGA: Disparity Estimation, Semi Global Matching, for 3D mapping

  - Decision
    - NVIDIA : Octo-cores ARM 64b + GPU Copro. Maxwell, API for Neural Network Programming, Objective: AI for UAV & Robots
    - CEVA : CEVA-XM FPGA Accelerator for *Convolutional Neuron Networks* applied to Embedded Computer Vision (released 2017)

1-Motivations

- **Efficient Reconfigurable Computing :**
  1. **HPC and Energy Efficiency for the right classes of applications**
     - Fixed-Point, Fine grain parallelism, Low-level Image processing, MapReduce
     - Graph traversal, FSM, Combinational Logic, CNN, …

     => Processing in Autonomous Vehicles

2D Conv - [FPGA'12]
FPGA vs GPU - **Speed x 10**
- **Power / 2.7**

CNN - [FPL16-Li] : 22 GOP/s/W Virtex7 - 30W vs GPU TitanX: - **Energy Efficiency x 2**
- **Power/8**

  2. **DPR = Massive Reuse to match with Cost and SWaP Constraints** (size weight power)
     - **Automotive**, Avionics, Space, **Drones**
     - Smart sensors, e-Health, IoT
     - Data centers, …
  3. **DPR = Reuse to Match with variable context and application evolution**
     - **Missions Phases of autonomous Vehicles**

[FPL2016-Li] H. LI et al., *A High Performance FPGA-based Accelerator for Large-Scale Convolutional Neural Networks*, FPL 2016
[FPGA-12-] J.Fowers et al., *A Performance and Energy Comparison of FPGAs, GPUs, and Multicores for Sliding-window Applications*, FPGA 2012

1-Motivations

- **Guarantee the configuration control**

1. **Autonomous Vehicles means critical Functions**
   - E.G. Safety Tasks
2. **Main Approach: Manual Encoding + Tests**
   - **Time Consuming and Error Prone**
   - **Do not guarantee not to get stuck in unwanted states**
3. **Use of a Formal Method to get correct-by-construction Configuration controller**
   - + DSL to capture the configuration behavior

2-HPeC / 3 Challenges

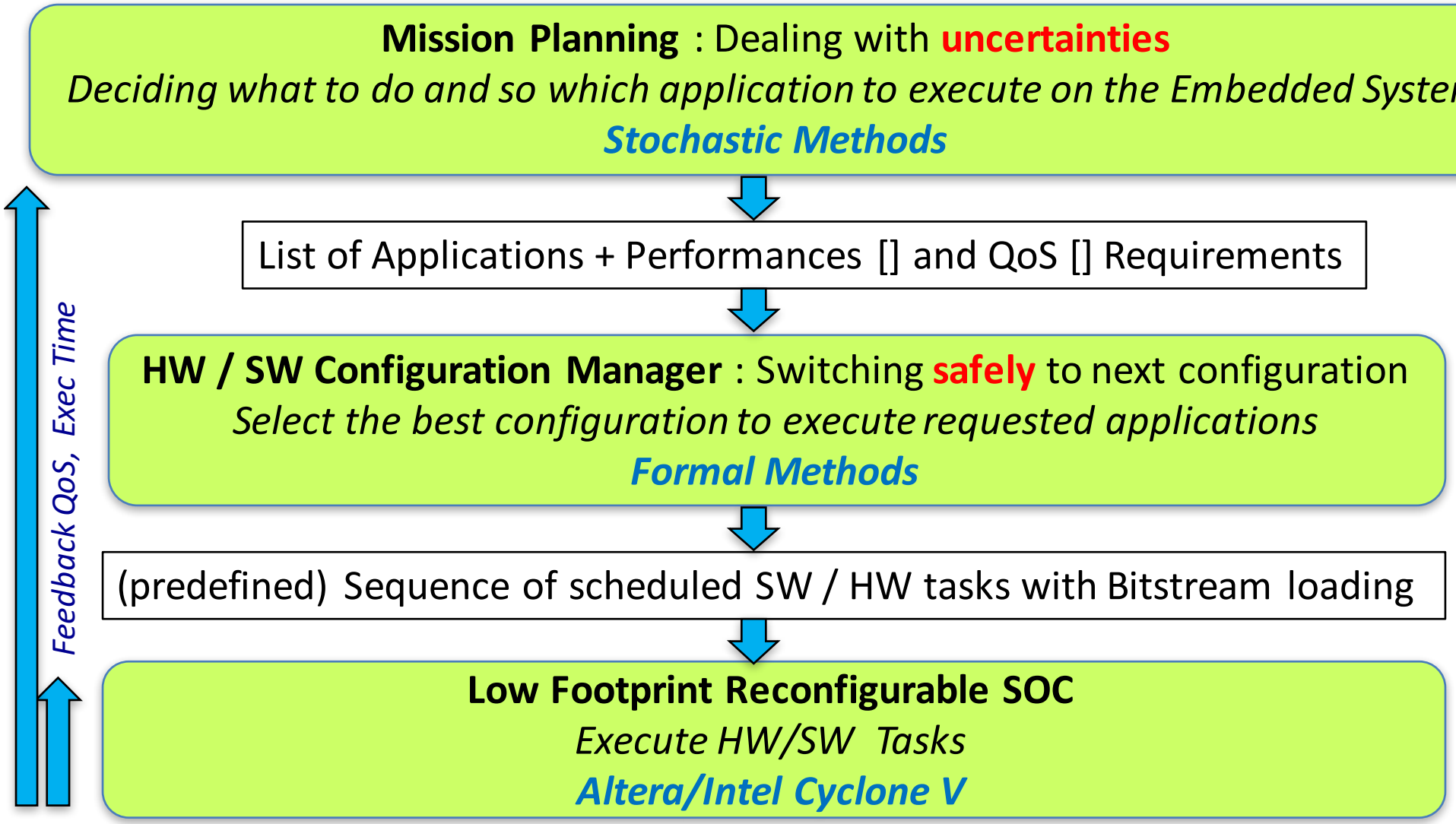
1) **HPC under Power, Area and Reliability Strong constraints**
   - GOPs / Watt / mm^2

2) **Reliable Dynamic Hardware Reconfiguration** (High-rate adaptation)
   - Guarantee Hardware / Software reconfiguration works

3) **Online adaptation for Mission Planning** (Low-rate adaptation)

   according to random events:
   - Sensor, UAV, Board failures: Online Diagnosis (Dynamic Bayesian Networks)
   - Application results (QoS metrics)
   - Mission Objective/Cost tradeoff (Markov Decision Process)

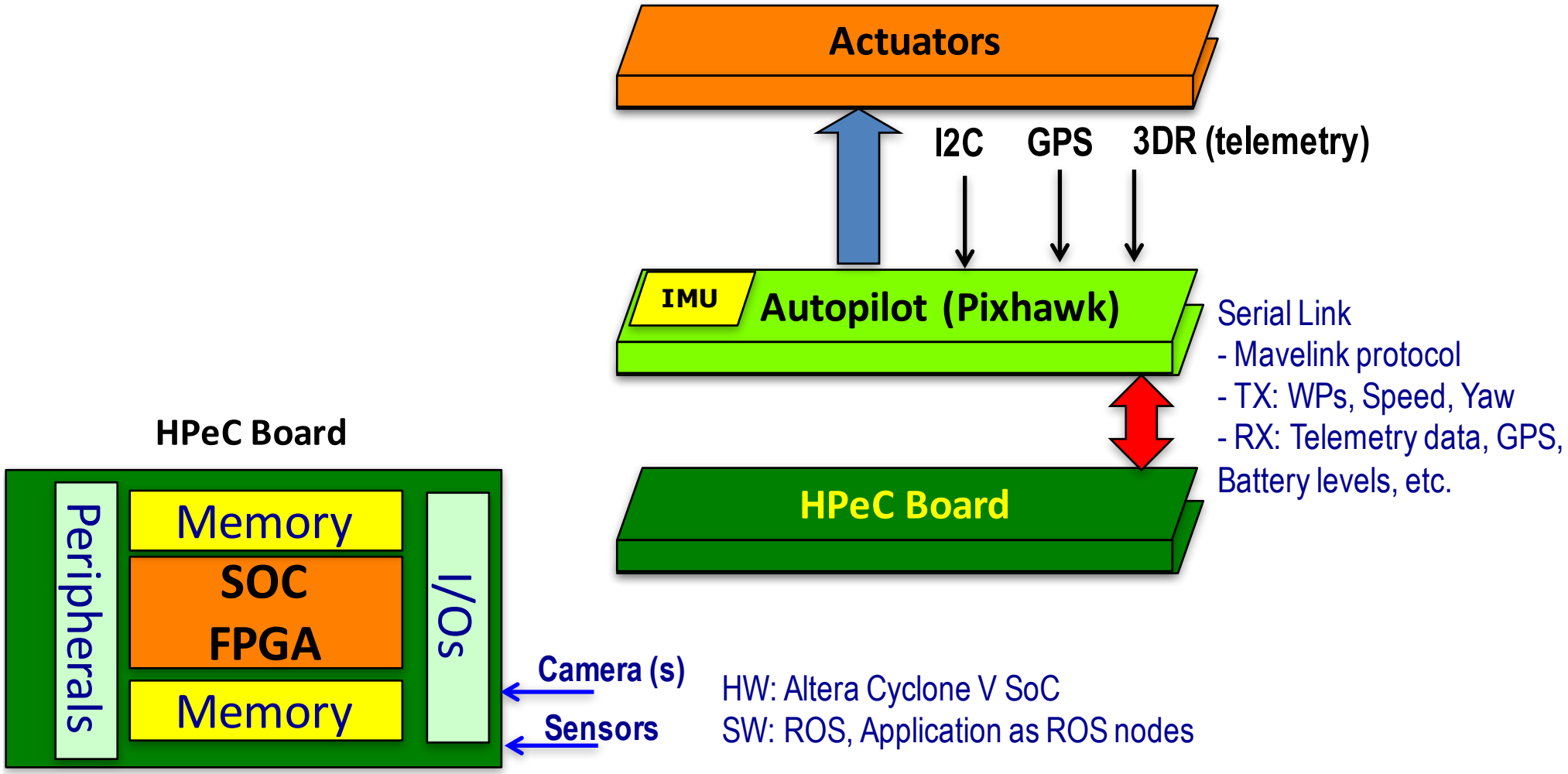
2-HPeC / Strategy



2-HPeC / Strategy

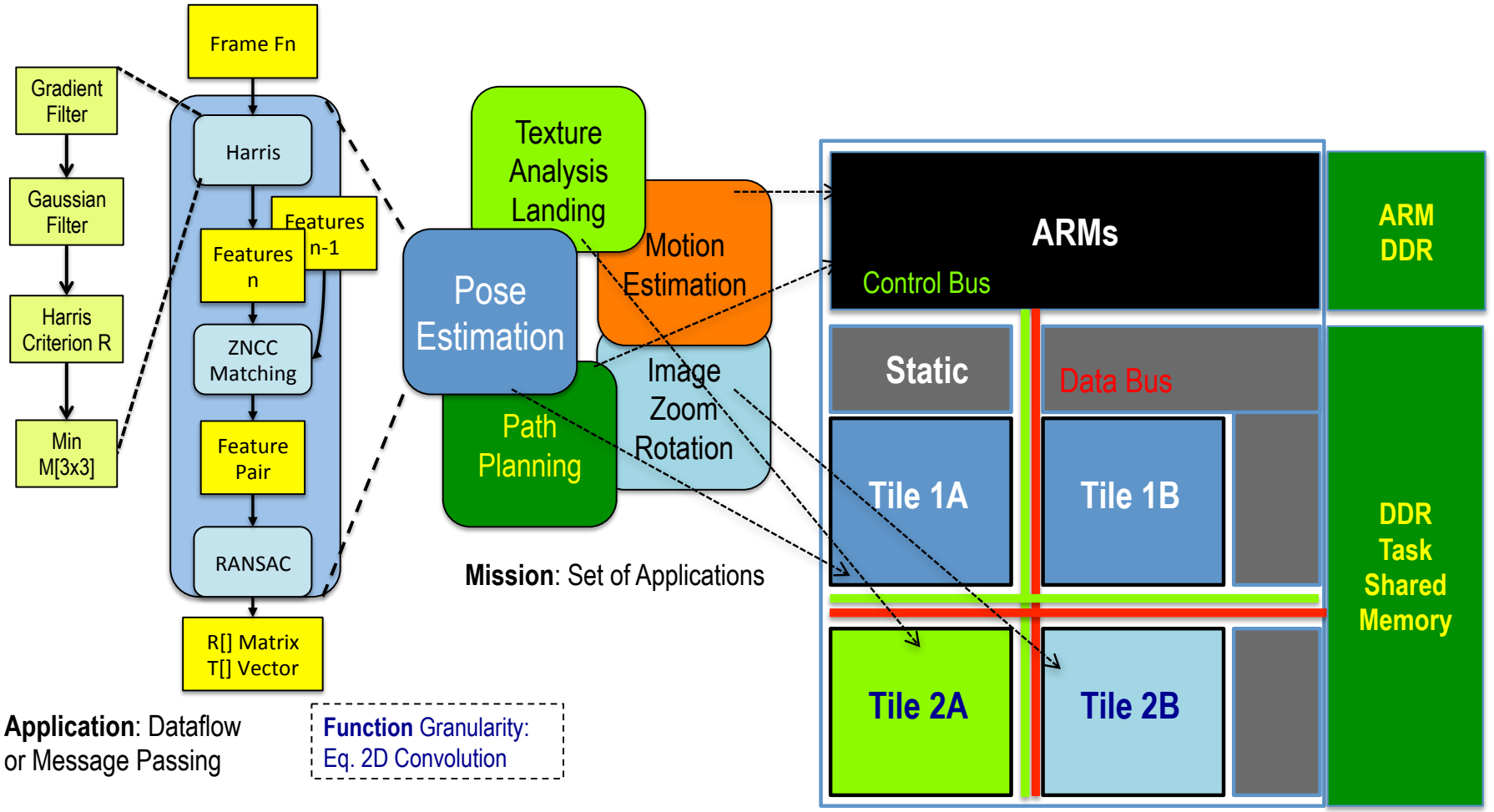
① Separate HPEC (Mission) / Autopilot (Navigation) Boards

– If HPeC fails : UAV failsafe processor in charge of Force landing



2-HPeC / Strategy

② Architecture model of the Mission board is based on Tiles implementing IP from Libraries or High-Level Synthesis

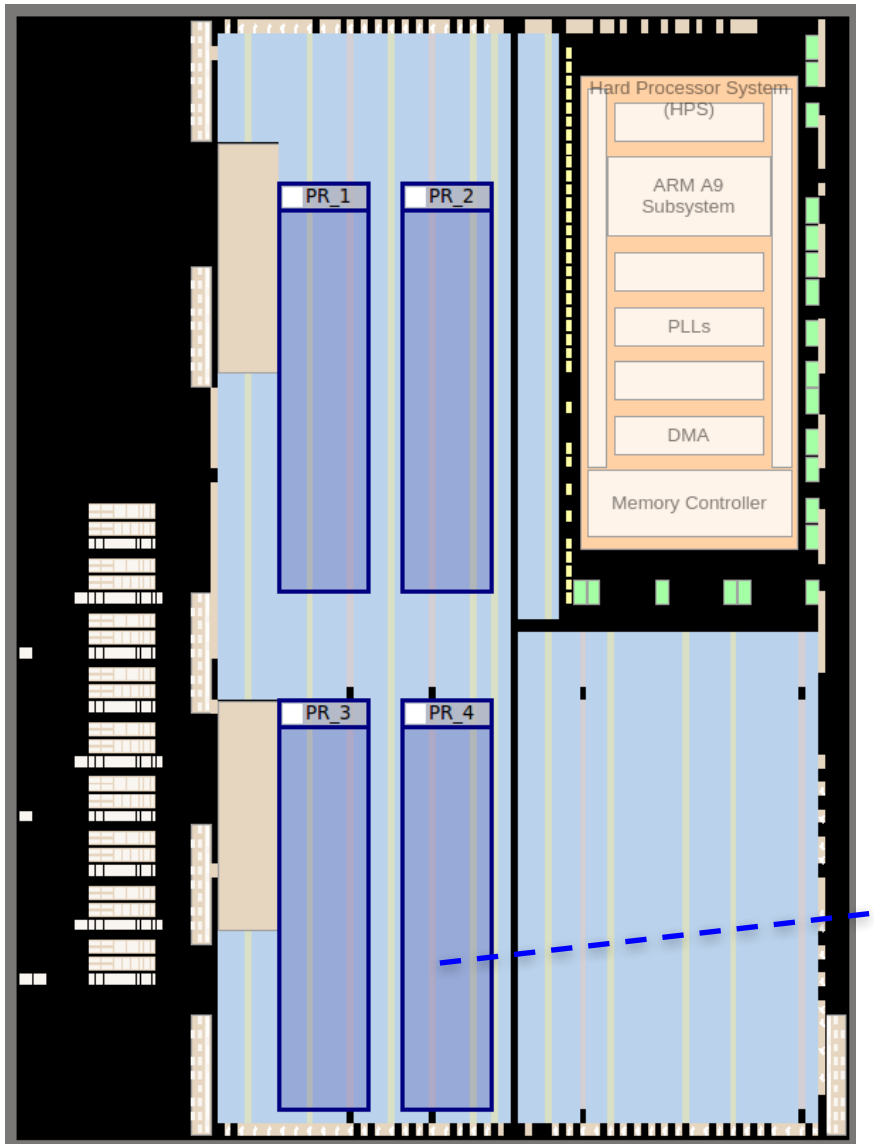


[RecoSOC17-Abdali] E-M.Abdali et al., *Exploring the Performance of Partially Reconfigurable Point-to-Point Interconnects, 12th RECOSOC 2017*

[ICDSC17-Abdali] E-M.Abdali et al., *Hardware Acceleration of the Tracking Learning Detection (TLD) Algorithm on FPGA*, 11th Int. Conf. On Distributed Cameras (ICDSC), 2017

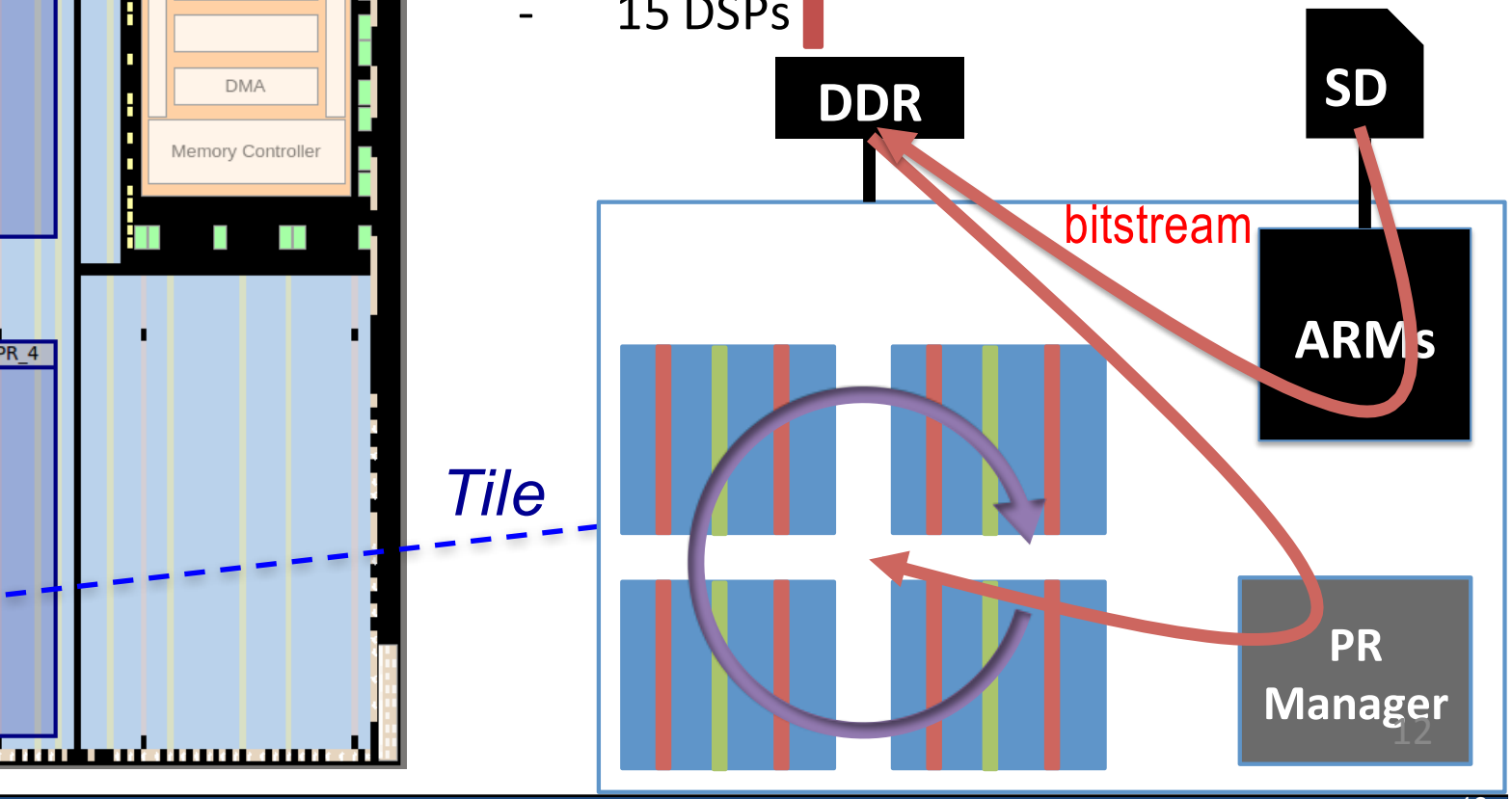
2-HPeC / Strategy

- DPR Tested on Cyclone V – Low Cost Device



4 PR tiles, each comprises approximately:

- 330 (11x30) LABs = 3.3K ALMs
- 30 RAM Blocks
- 15 DSPs

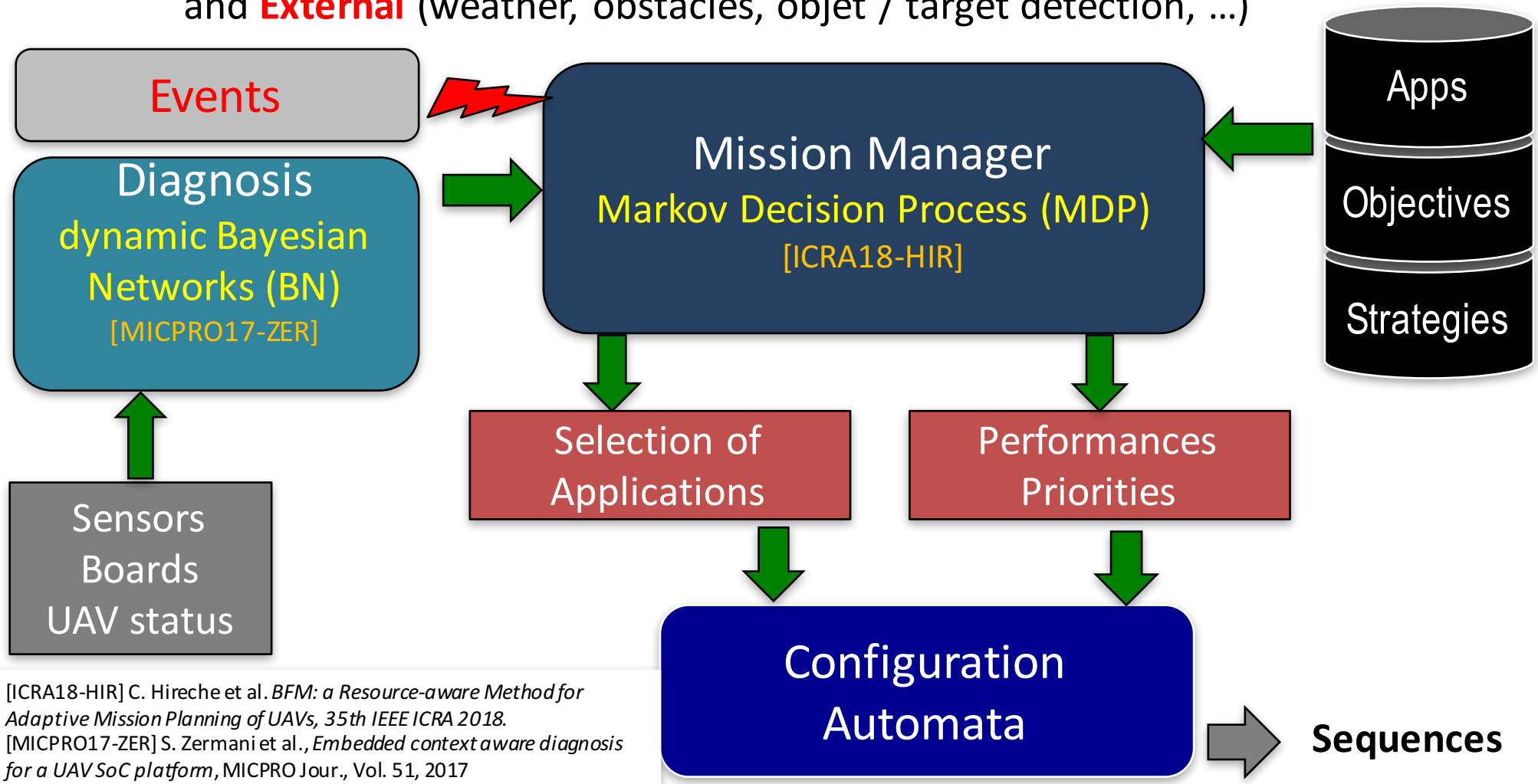


2-HPeC / Strategy

③ Configuration decision based on 2 levels

– **Level 2: Stochastic methods for online decision making** according to random events **Internal** (failures risks, failures …)

and **External** (weather, obstacles, objet / target detection, …)



[ICRA18-HIR] C. Hireche et al. *BFM: a Resource-aware Method for Adaptive Mission Planning of UAVs, 35th IEEE ICRA 2018.*

[MICPRO17-ZER] S. Zermani et al., *Embedded context aware diagnosis for a UAV SoC platform*, MICPRO Jour., Vol. 51, 2017

2-HPeC / Strategy

③ Configuration decision based on 2 levels

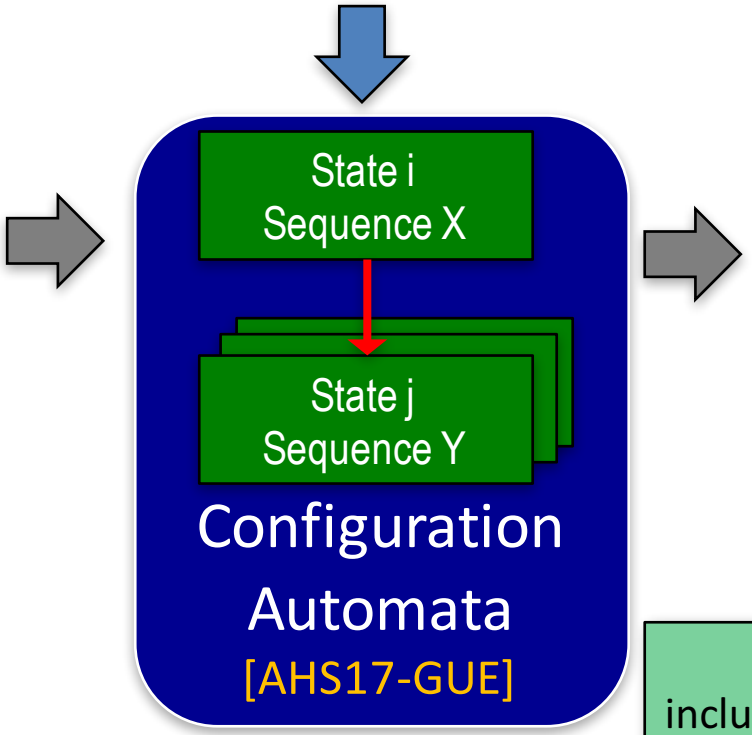
– **Level 1: Reliable** Control of Scheduling Sequences including HW reconfigurations with a controller generated with formal discrete controller synthesis techniques

Synthesis made **offline** for different cost functions
(QoS, Reliability, Power Consumption, Performance)

Uncontrollable Variables

From Mission Level:
- Task Priorities
- Performances
- …

From Architecture
- End of Frame
- End of Function
- Execution Times
- …



Selection of a Sequence per Application

A State Configuration = A Sequence
including:
- **Scheduling** of **Hardware** Functions
- **Scheduling** of **Bitstream loading**
- Software Task execution

[AHS17-GUE] S. Mak-Karé Gueye et al. *Autonomic Management of Missions and Reconfigurations in FPGA-based Embedded System*, 11th NASA/ESA Conf. On Adaptive HW and Systems (AHS), 2017.

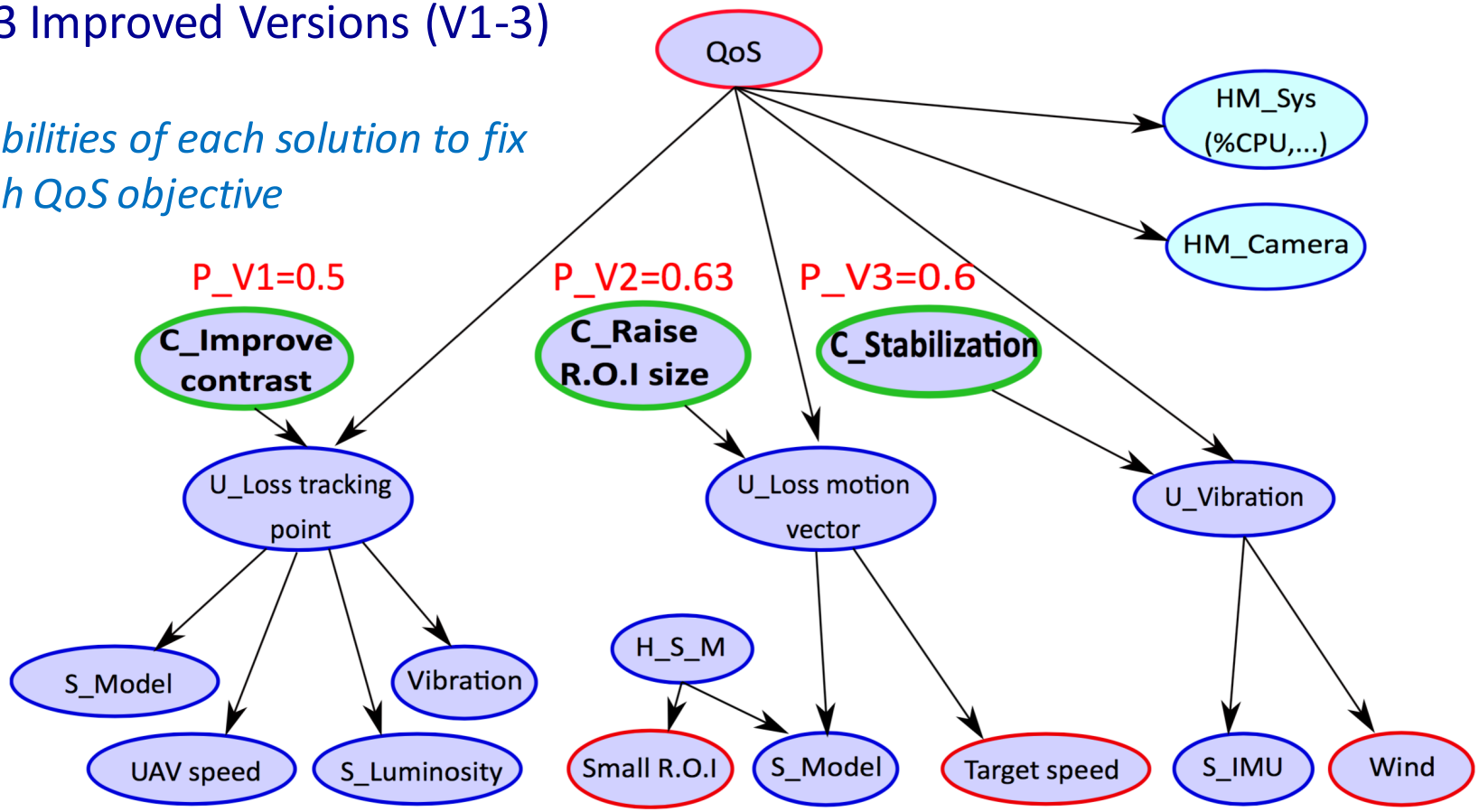
2-HPeC / Strategy

- **Level 2: Stochastic methods**
- **BN** for Diagnosis, Nodes = random variable, Computes Probabilities to feed MDPs

e.g. Tracking Application :
Nominal (V0) + 3 Improved Versions (V1-3)

2) Fix the QoS expected value (e.g. High)

3) Result = Probabilities of each solution to fix the issue and reah QoS objective

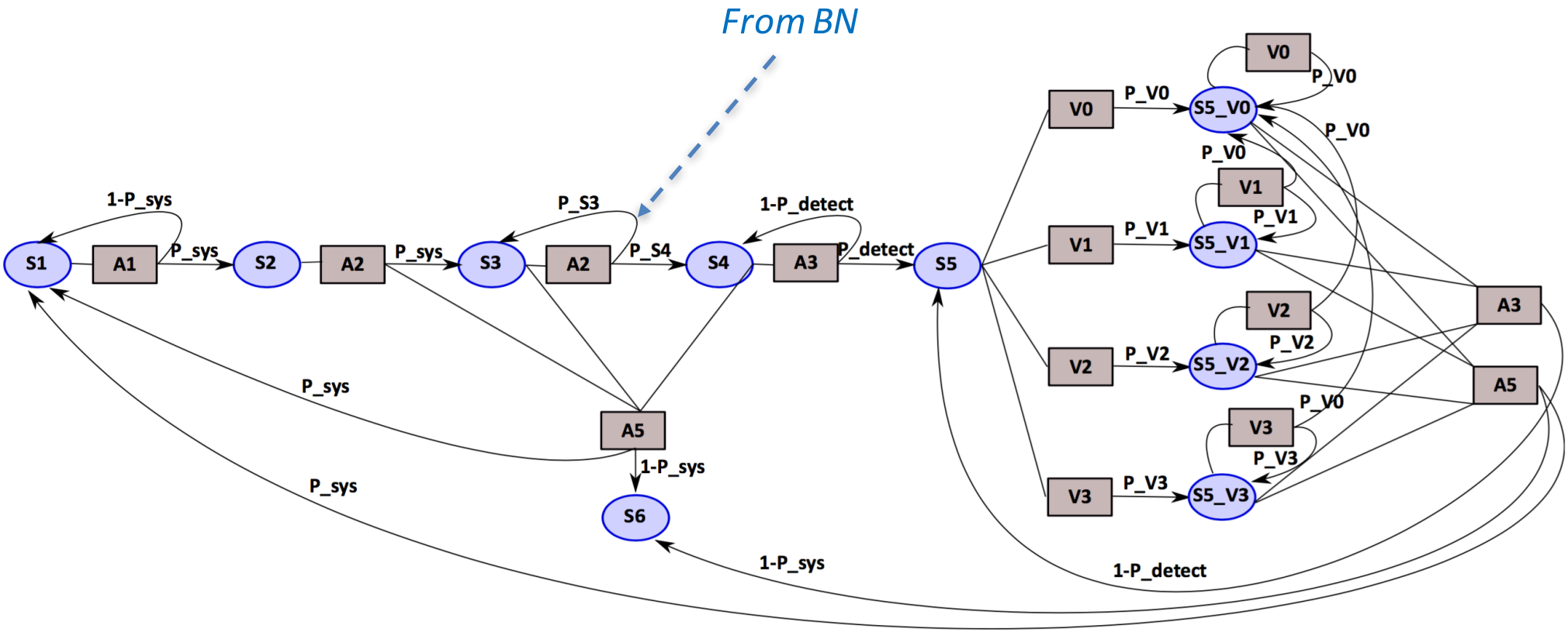


1) Observations based on sensors, models or application metrics

2-HPeC / Strategy

- **Level 2: Stochastic methods**
- **MDP for Mission Planning**

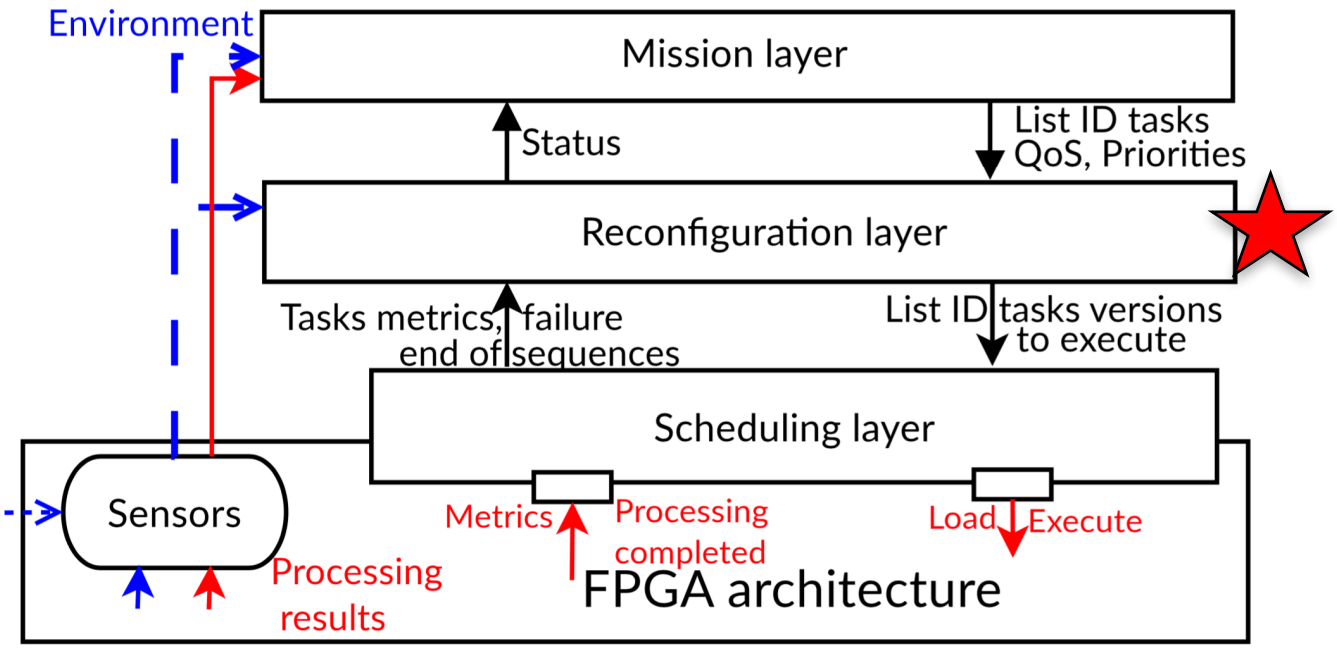
e.g. Mission “Cruise, Hover and Track”:



Resolution of MDP, e.g. Finite Horizon Bellman Ford

2-HPeC / Strategy

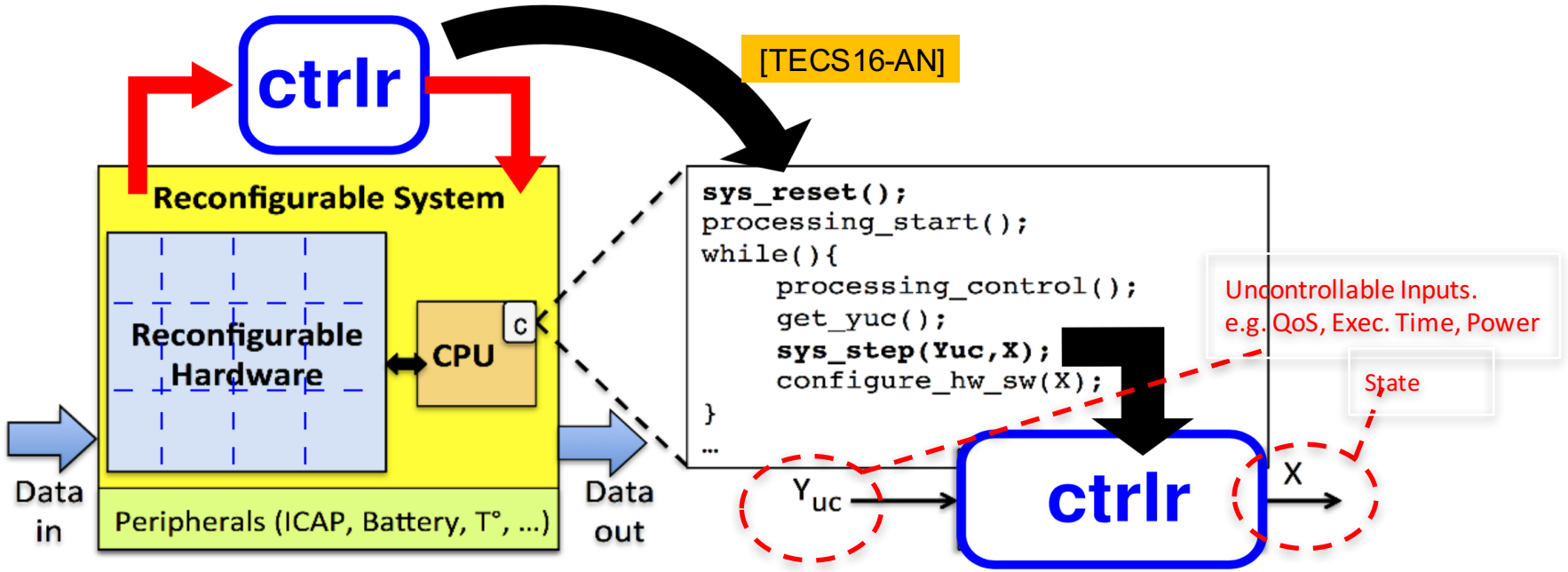
- **Summary**



1. **Mission layer**
   - Evaluates the environment & system health (BN)
   - Dynamically adapts the mission (MDP) => Determine subset of Active Tasks
2. **Configuration Layer**
   - Manages the concurrent execution of tasks
   - Dynamically adapts the active task versions => Determine Task Versions
3. **Scheduling layer**
   - Perform hardware reconfiguration => Execute Tasks + bitstreams loading

3- Formal Reconfiguration Control

- **Approach** : Tool-supported formal method to automatically design a correct-by-construction control of the reconfiguration
  - Reactive programming that provides formal semantics (BZR)
  - Discrete Controller Synthesis from declarative objectives (Heptagon)
  - Set of FPGA configurations ⇔ **States of an Automata**
  - Adaptation ⇔ **Controlled Behavior of the Automata**

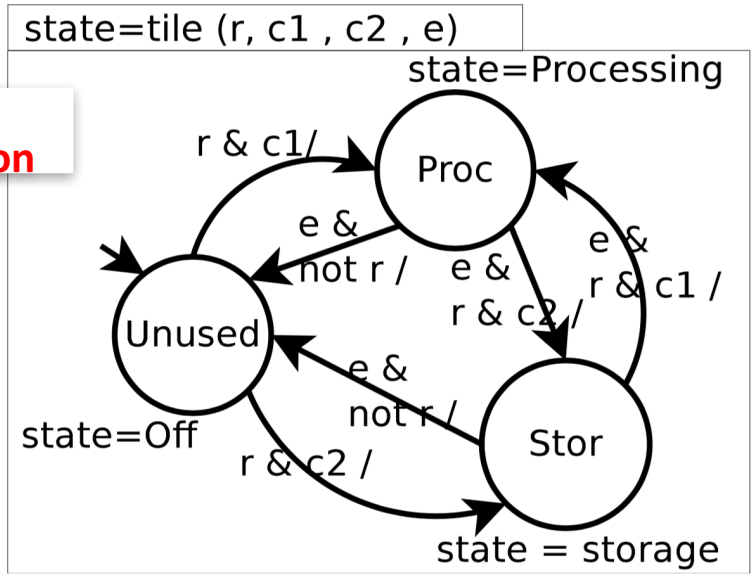


[TECS16-AN] X. An et al., *Model-based design of correct controllers for dynamically reconfigurable architectures*, ACM TECS, Jul. 2016

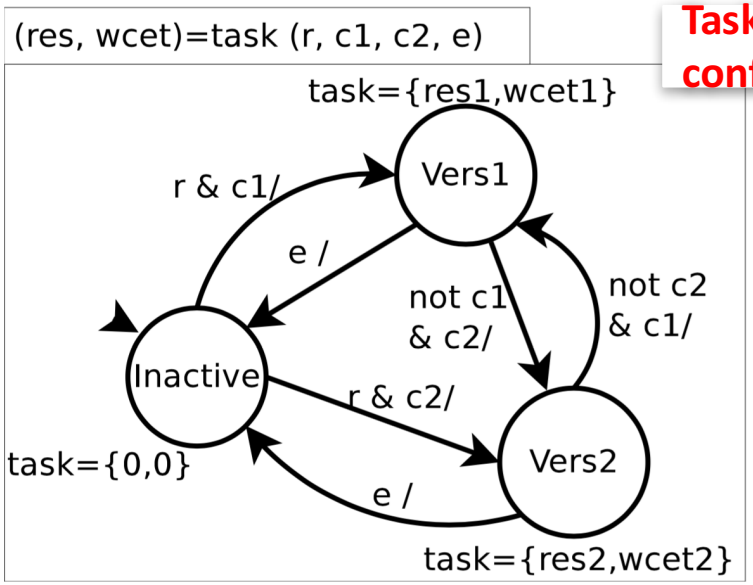
3- Formal Reconfiguration Control

- **Tools**
  - Controlling choices in **Combinatorial spaces**
    - Focus on **Discrete Event Systems**
  - **Discrete Control Synthesis** (DCS)
    - Maximally Permissive : all solutions are explored
    - Construct logic enforcing the objectives (e.g. Energy)
  - **Deterministic behavior**
    - **Guarantee not to be stuck** in an unexpected state

Tile configuration

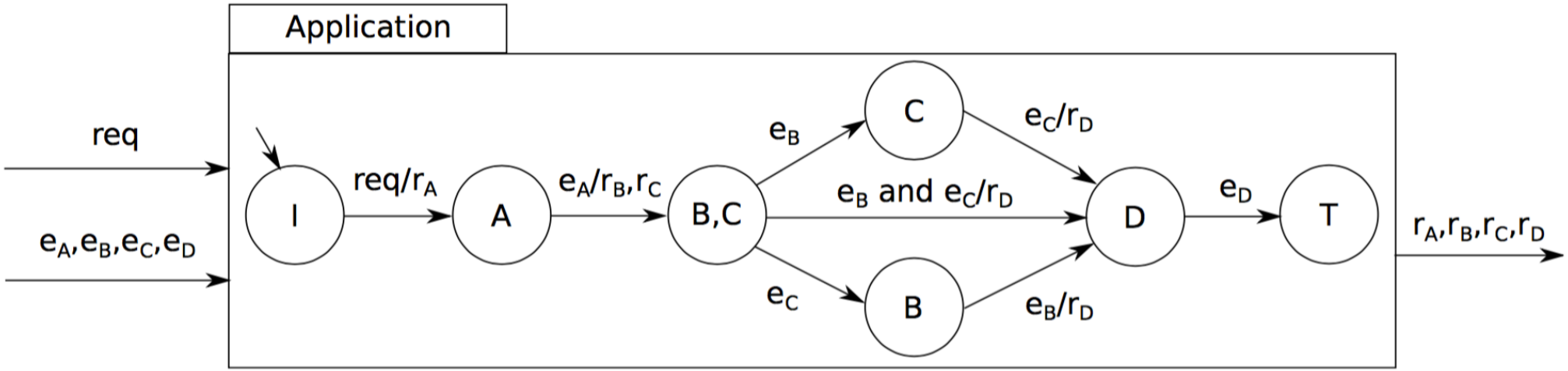


Task configuration



3- Formal Reconfiguration Control

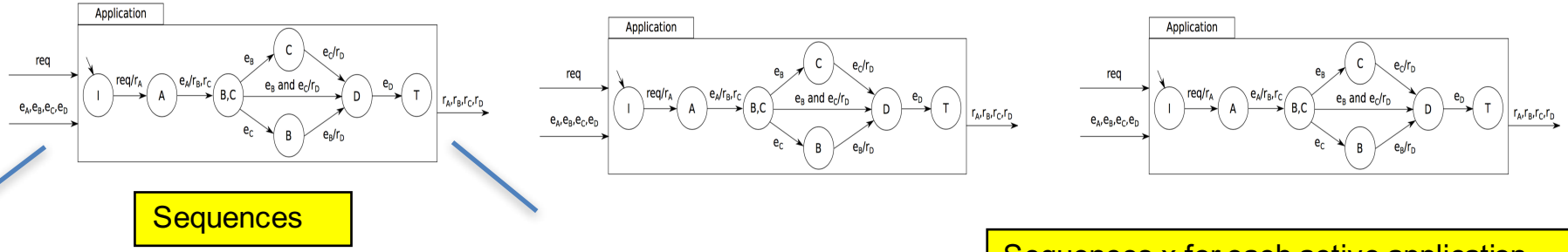
- **Application model (initial model - 2015):**
  - e.g. of Direct Acyclic Graph (**DAG**)



- **Task Versions** *Automata*
  - SW / HW, Size (tiles), Exec time, Power, QoS
- **Architecture** *Automata*
  - Sleep Mode, DVFS, Bitwidth, etc.
- **Policies** *Synthesis Objectives*
  - Power Peak, QoS, Resource use

3- Formal Reconfiguration Control

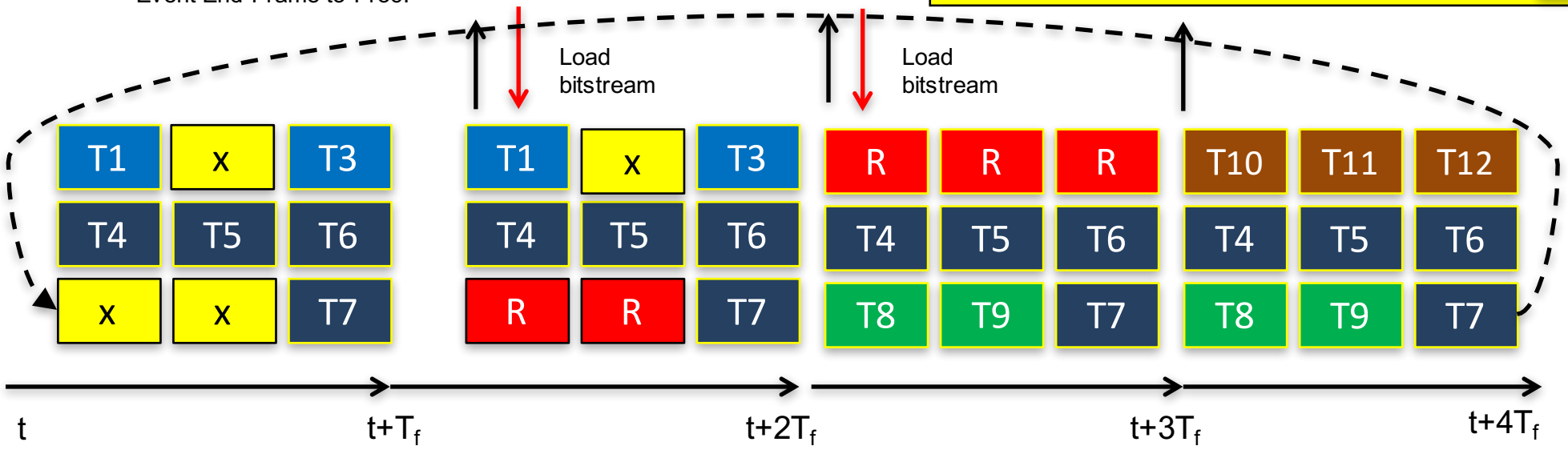
- **Application model - HPeC version [AHS17]:**
  - Coarse grain to match with complexity of multiple application
  - Predefined sequence for each application



Sequences x for each active application
- Appli 1 (T1 >T3): Period = 2Tf
- Appli 2 (T4 > T5 > (T6 // T7)): Period = 4Tf
- Appli 3 (T8 > T9): Period = 2Tf
- Appli 4 (T10 >T11>T12): Period = 1Tf

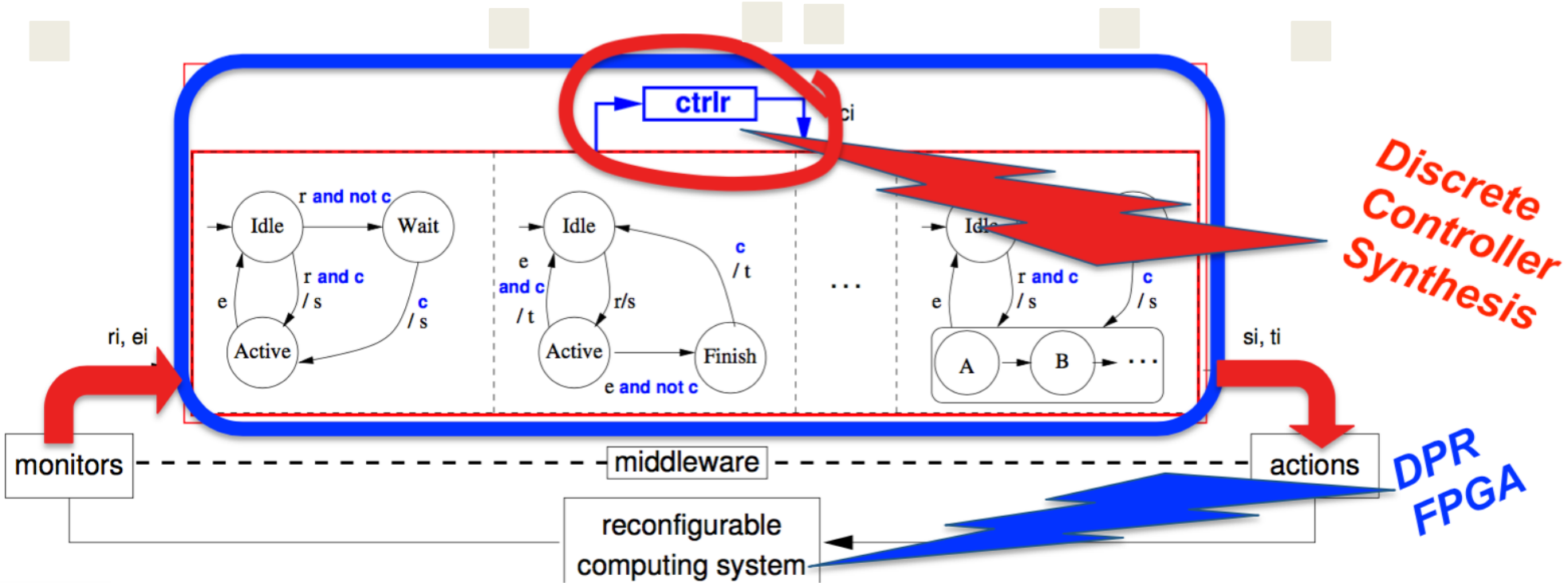
ARM1 (SW / ROS) | ARM2 Scheduler

Event End Frame to Proc.



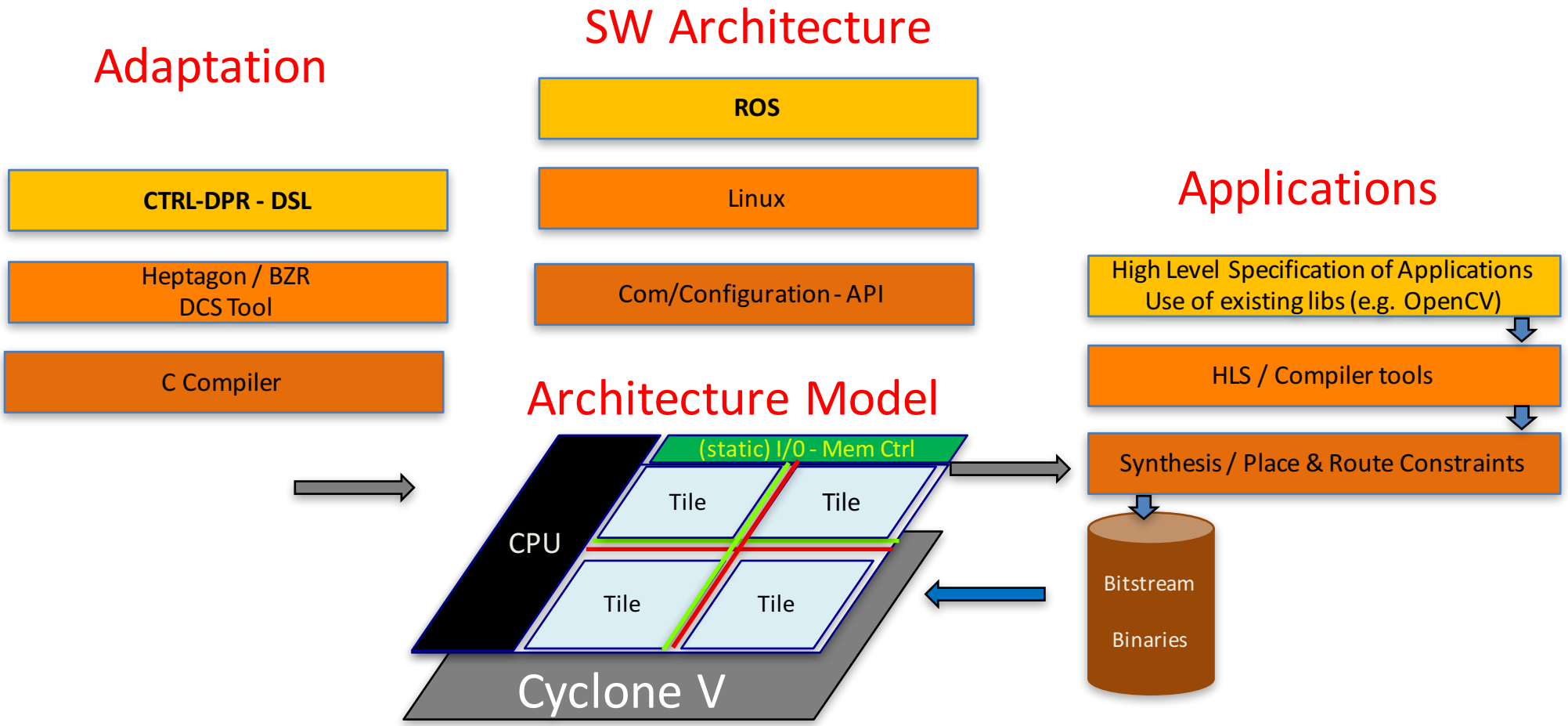
3- Formal Reconfiguration Control

- **Implementation of Model-based reconfiguration control**
  - **DCS code generator => C code**, PGM running on the CPU or small softcore
  - **Interfaces => API** for collecting events and Select / Activate Sequences



3- Formal Reconfiguration Control

- **How to specify the Adaptation behavior ?**
  - **Automata language are too specific**
  - **A Domain Specific Language (DSL) is necessary :** *(code name)* **CTRL-DPR**



3- Formal Reconfiguration Control

- **DSL DPR-CTRL**
  - **Generic element structure**

```
type name(id) :
  in : var [int | bool],… ;
  out : var [int | bool] (= default value), … ;
  …
  …
  policy :
      local strategies
end
```

  - **Resource type: CPU, Tile, Peripheral**
    - **Utilisation** (free or used, Exclusive (e.g., tile) or Shareable (e.g. CPU)
    - **Availability** (e.g. unavailable if failure)
    - **Configurations** (e.g., Dynamic Voltage Frequency Scaling in CPU)

```
resource tile :
  fallible;
  usable as Pe, Mem (shareable by 3);
end
```

```
resource arm :
  out : speed int, energy int;
  shareable by 5;
  configurations DVFS :
      config Low : speed = xx, energy = yy end
      config Medium : speed = xx, energy = yy end
      config High : speed = xx, energy = yy end
  end
end
```

3- Formal Reconfiguration Control

- **DSL DPR-CTRL**
  - **Task**
    - **Processing states :** (inactive, active, waiting)
    - **Versions**

```
task task_a :
  in : e_a bool, e_b bool;
  out : wcet int;
  version v_1 : uses arm, wcet = 500 end
  version v_2 : uses tile.Pe, wcet = 300 end
  policy :
      e_a then v_1;
      e_b then v_2;
end
```

  - **Application**
    - **Set of tasks**
    - **Execution mode**
      - **Sequence** ;
      - **Parallel** ||
      - **Data-flow** ▷
      - **Mix of them**

```
application app :
  execution : (task_a;task_b) end
end
```

3- Formal Reconfiguration Control

- **DSL DPR-CTRL**
  - **Policy**
    - **Objectives**
      1. **Maintaining the performance in defined intervals**
      2. **Ensuring coherent usage of resources,** e.g., mutual exclusion wrt to the tiles.
      3. **Ensuring coherent configuration of resources** to reduce energy while maximizing the performance.
    - **DCS problem** : **Synthesis must find a solution and objectives can be conflicting**
    - **Solution :** **- Assign Priority to Objectives, Highest to be solved first**
      - Objective Priority K = Condition for Objective Priority K-1
    - **Adaptable:** **- Priority = Variables, can be modified at runtime**
    - **Keywords :** **- not, or, and, then, prior, pre, ◁, <, >**

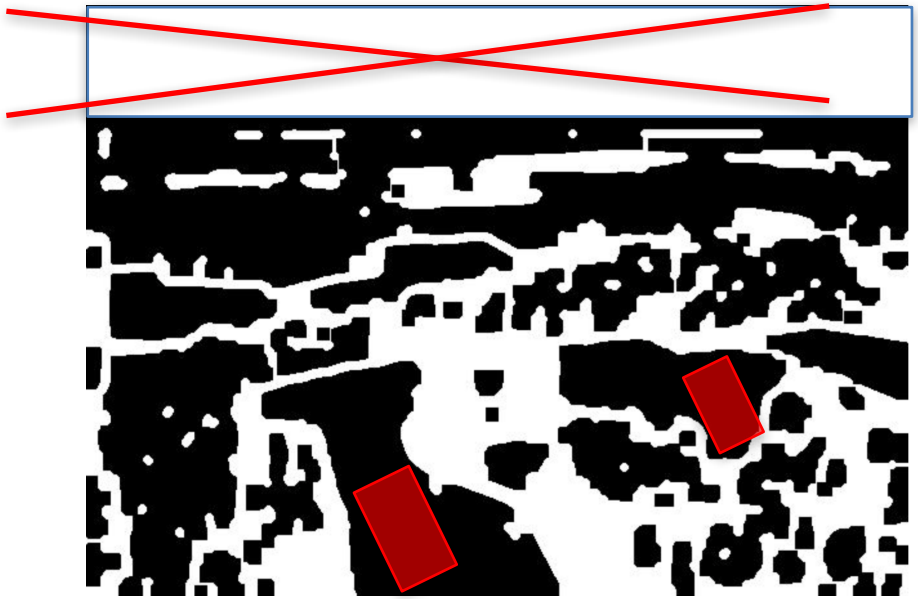
Policy:
C1: (min_T < time_t) **and** (time_t < max_T) **and** (time_t ~= wcet);
C2: (pow_t < max_P) **and** (pow_t ~= pmin);

4- UAV case study

- **Search-Landing Application**
  - One of the HPeC scenario application from the **security task set**
  - **Image Processing Tasks :**
    1. Format Conversion
    2. Median Filter: noise reduction
    3. Canny (Gaussian, Sobel): edge detection
    4. Morphological Operator (dilate/erode)
    5. Area candidate extraction
    6. Area Selection

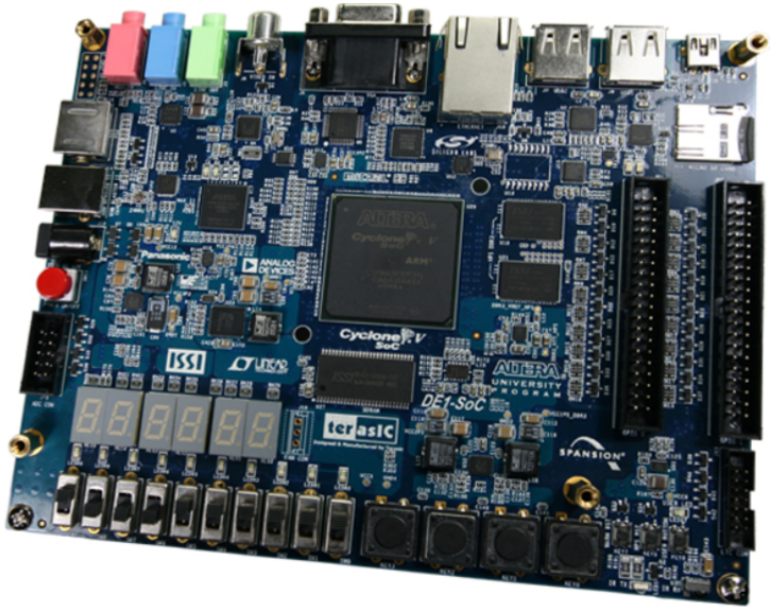
HW or SW: T1(F1,F2,F3,F4)

SW : T2 (F5,F6)



4- First UAV case study

- **Dev. Platform:** DE1-SOC Altera/Intel Cyclone
  VGA Images (on SD Card)
- **HPS (Dual-Core ARM):**
  - ROS (Robotic Operating System) Middleware on Linux
  - Each Application is a ROS Node
  - Manager and Schedulers are ROS Nodes
  - Sensor and Autopilot access
- **Hardware implementation:**
  - 1 / 4 of a “4 Tiles” architecture model
  - 55ms @25MHz
- **Full Software Implementation**
  - 1500ms (including SD card access and ROS overhead)



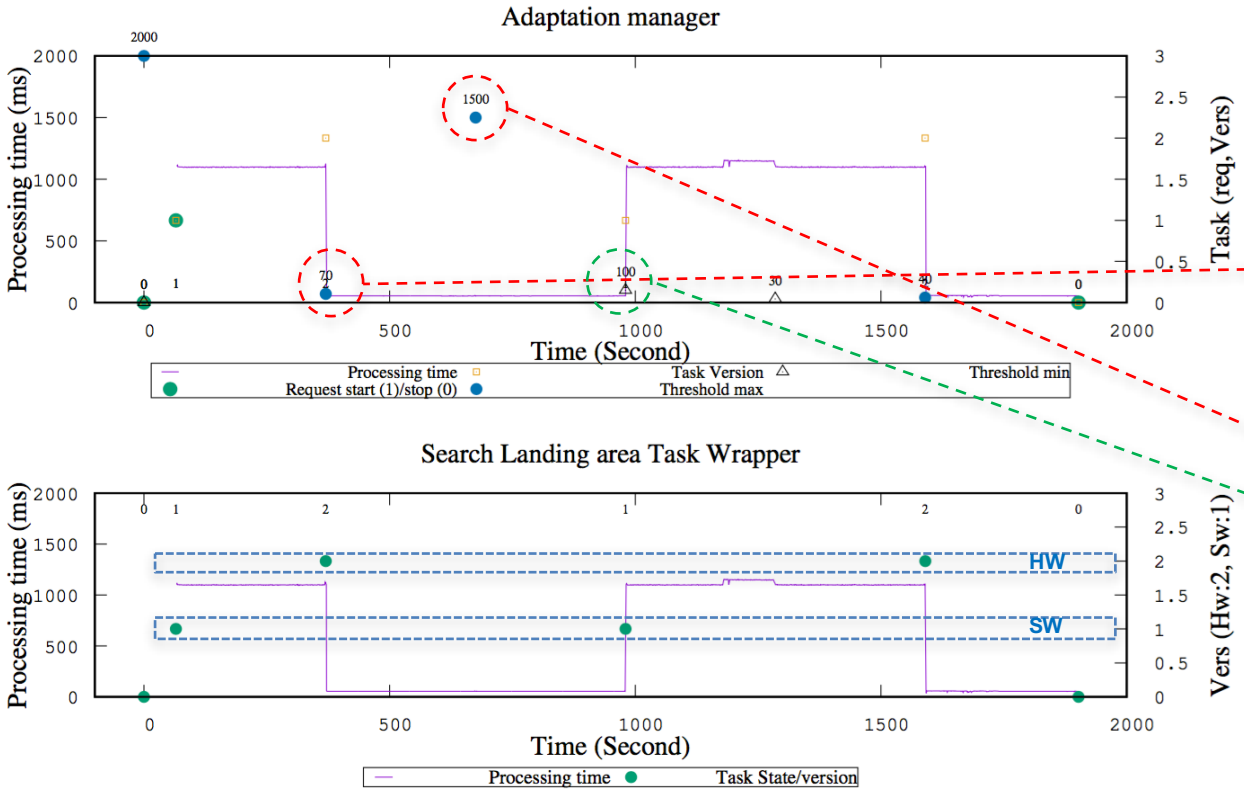
4- First UAV case study

- **Case 1 : 1 Controlled Task and Execution Time Variation**

```
resource tile :
  usable as Pe;
end

resource arm :
  shareable ;
end

resource camera :
  shareable ;
end

Task searchArea :
  in : min int, max int, time_t int;
  out : wcet int;
  version Software : uses arm, wcet = 1500 end
  version Hardware : uses tile, wcet = 55 end
  Policy :
      (min < time_t) and (time_t < max) and (time_t ≅ wcet);
end

main :
  Involve :
      1 tile, 1 arm, 1 camera, 1 searchArea;
end
```

4- First UAV case study

- **Case 1 : 1 Controlled Task and Max-Time Requirement changes**



1. The manager starts the software version of the task
2. max_thres = 70 :
   - `(time_t > max_thres)`
   - The manager switches to the hardware version
3. max_thres = 1500 :
   - `min_thres = 100`
   - `(time_t < min_thres)`
   - The manager switches to the software version

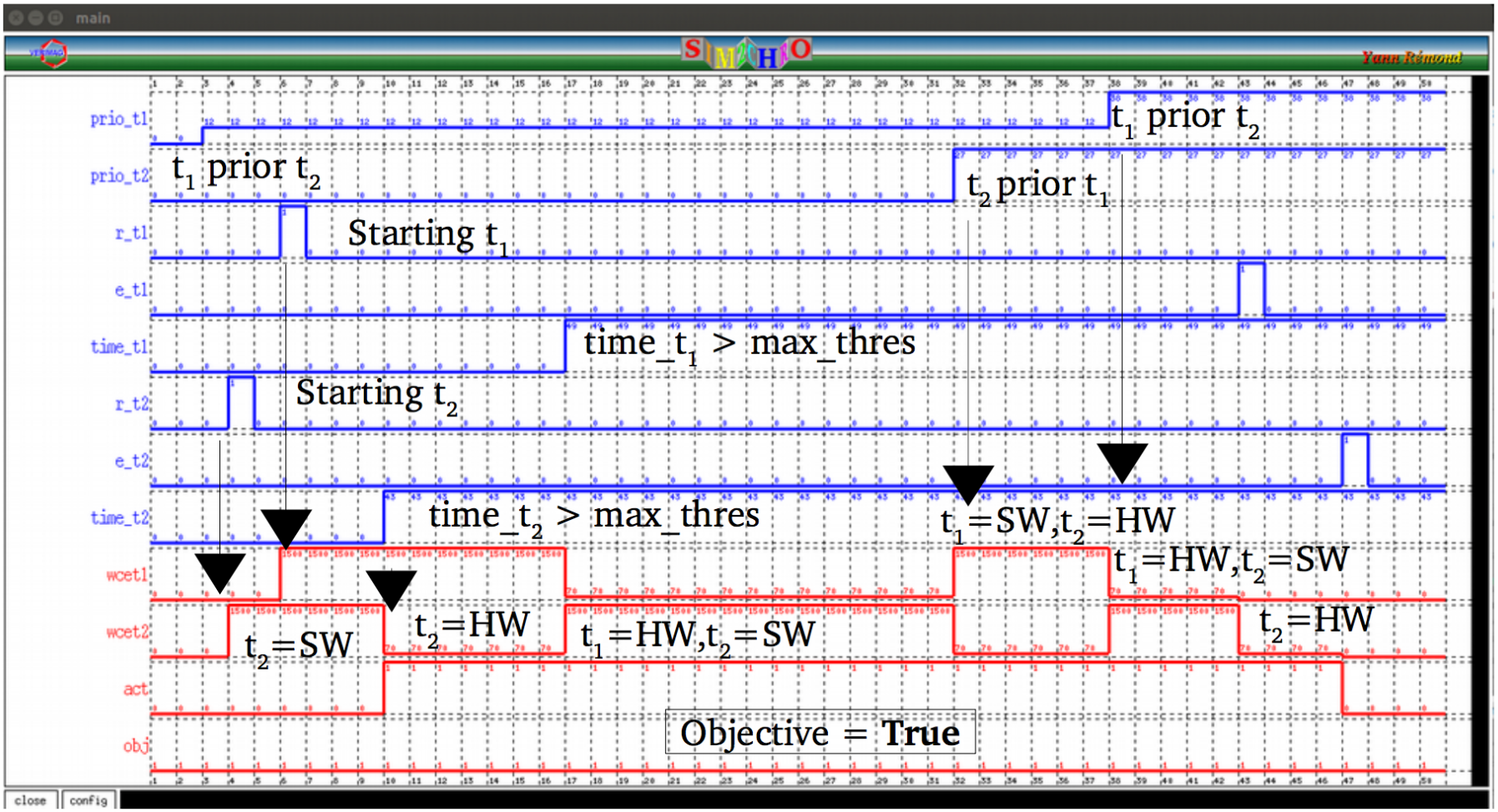
4- First UAV case study

- **Case 2 : 2 Controlled Task and Execution Time Variation**

```
...
main :
  in : prio_t1 int, prio_t2 int;
  Involve :
       1 tile, 1 arm, 1 camera, 1 task1, 1 task2;
  Policy :
       (prio_t2 ≤ prio_t1) then (task1 prior task2);
       (prio_t2 > prio_t1) then (task2 prior task1);
end
```

4- First UAV case study

- **Case 2 : 2 Controlled Task and Execution Time Variation**



Conclusion

- **DPR is a solution for HPC on low-cost Embedded system**
- **Multi-Layer Approach:**
  - **Unpredictability of mission due Environment**
    - **=> Stochastic Layer**
  - **Guarantee the behavior of the reconfigurable system**
    - **=> Deterministic Configuration Control with DCS**
  - **Simplify control of Bistream loading**
    - **=> Interfaces / API**
    - **=> Architecture model**
    - **=> Lib. of bitstreams / binaries**
  - **DSL for autonomic computing**
- **Real-life demonstrator with Autonomous Vehicles**
  - **Compliant with ROS standard**
- **On going work:** **make the whole scenario working in the real world with multiple applications**