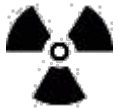
AADL in the INGEQUIP project
Achievements, Perspectives

The INGEQUIP team

Outline



1 – From INGEQUIP to CAPHCA: The characters and the story

2 – INGEQUIP: Some REX on AADL

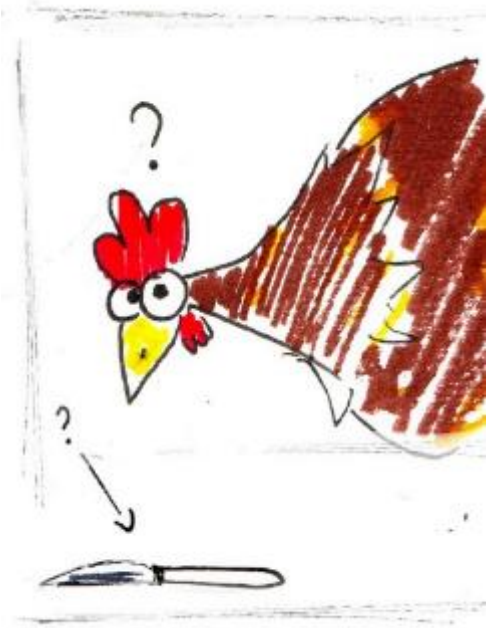
3 –CAPHCA: What's next?

From INGEQUIP to CAPHCA

The characters

1. Engineers with some kind of experience on topics related to safety critical real-time embedded systems (software and hardware) in the nuclear, **aeronautical** and **automotive** domains
2. Seconded by our companies (Thales, Continental, …) to work at the IRT Saint-Exupery lab.

1. *Newbies* on AADL



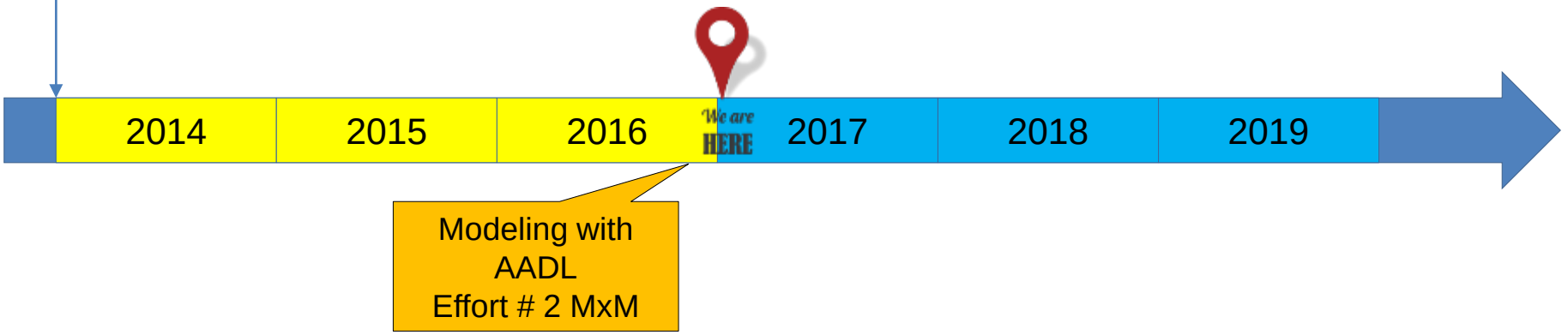
2. Great help from Pierre Dissaux and Jérôme Hugues (but all errors are ours)

From INGEQUIP to CAPHCA

The story : From "INGEQUIP" to "CAPHCA"

2014-2016 INGEQUIP, *Equipment Engineering*

Focus on development activitities "at large" (and formal methods)

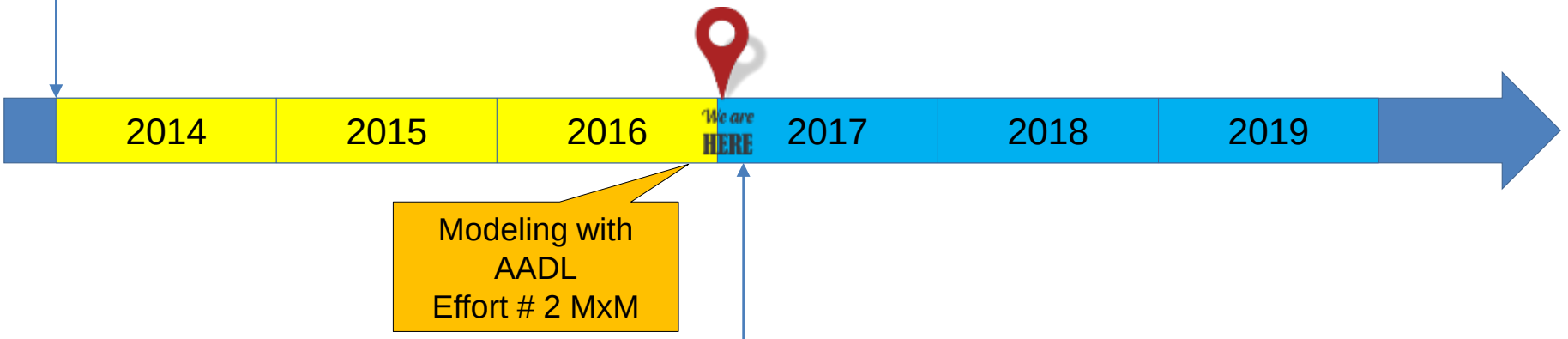


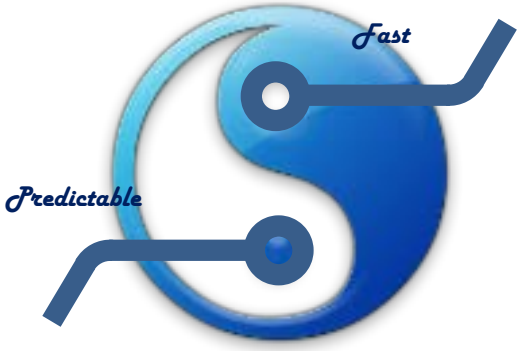
From INGEQUIP to CAPHCA

The story : From "INGEQUIP" to "CAPHCA"

2014-2016 INGEQUIP, *Equipment Engineering*

Focus on development activitities "at large" (and formal methods)





2017-2019: CAPHCA, *Critical Applications on Predictable High-performance Computing Architectures*

Focus on performance estimation and optimization on multicore platforms (incl. FPGA coproc.)

From INGEQUIP to CAPHCA

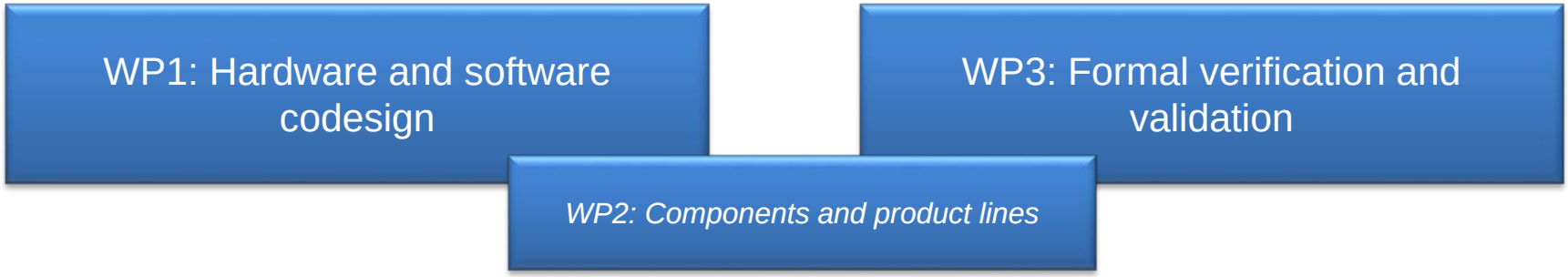
INGEQUIP

Support and automate **design space exploration**

Evaluate and extends **co-design** frameworks

Combine methods & tools to develop equipment components

Contribute to the integration of **formal development methods**



From INGEQUIP to CAPHCA

INGEQUIP

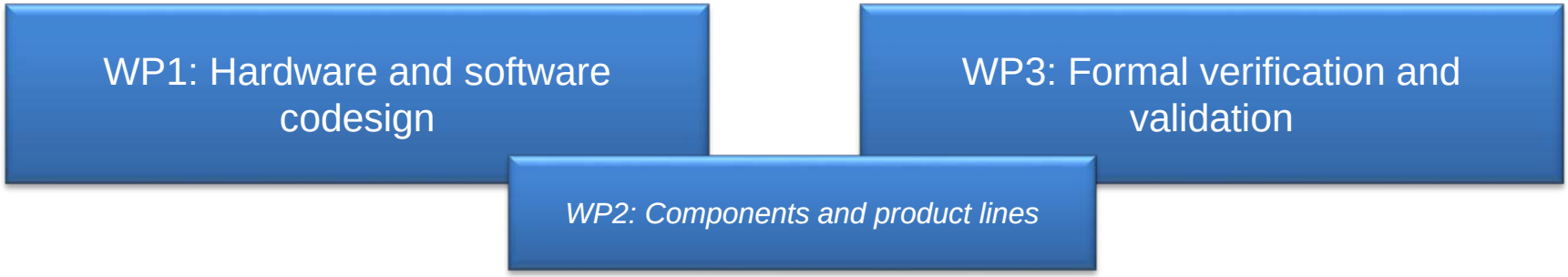
Support and automate **design space exploration**

Evaluate and extends **co-design** frameworks

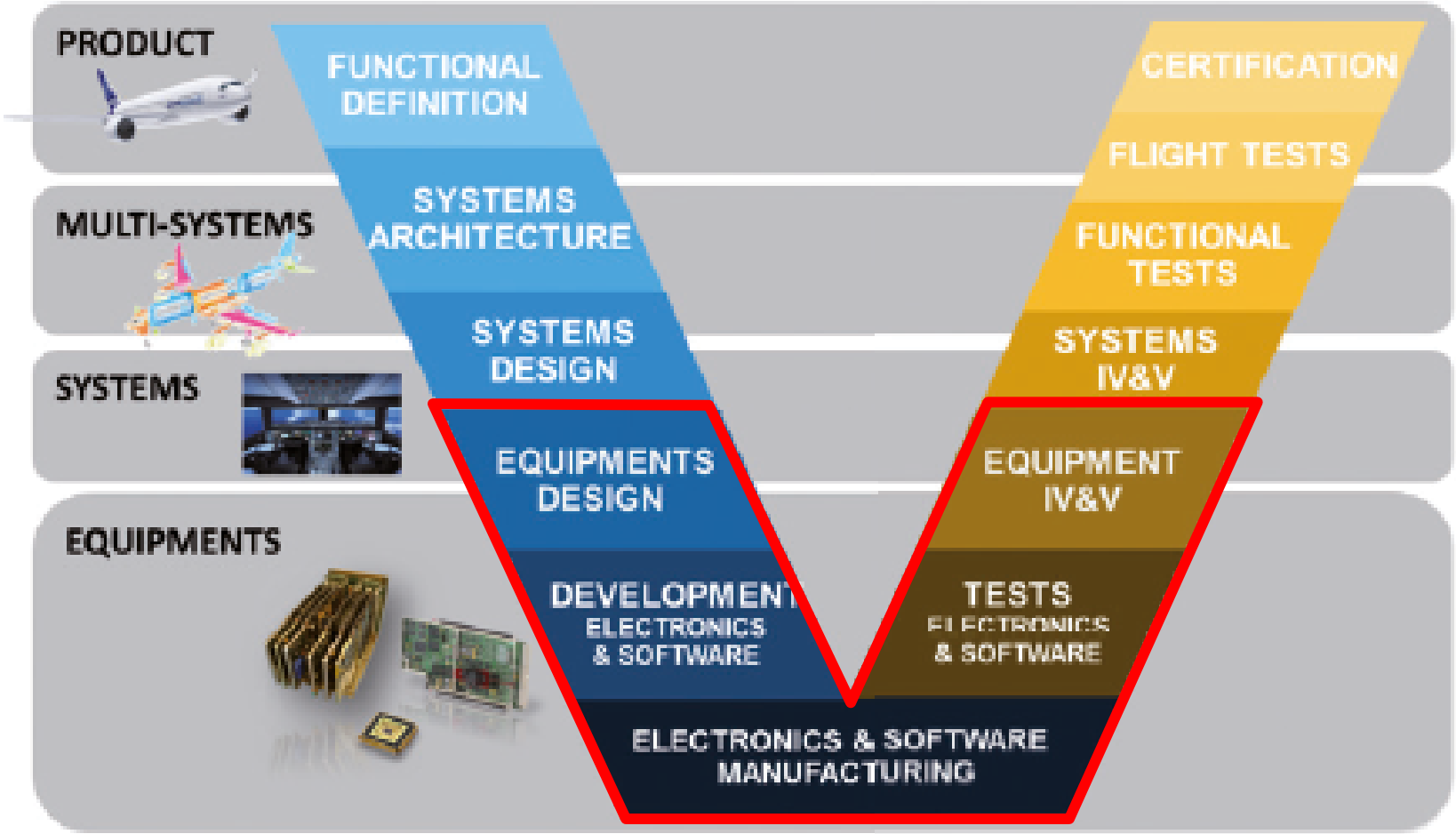
Combine methods & tools to develop equipment components

Contribute to the integration of **formal development methods**

Activities on FIACRE

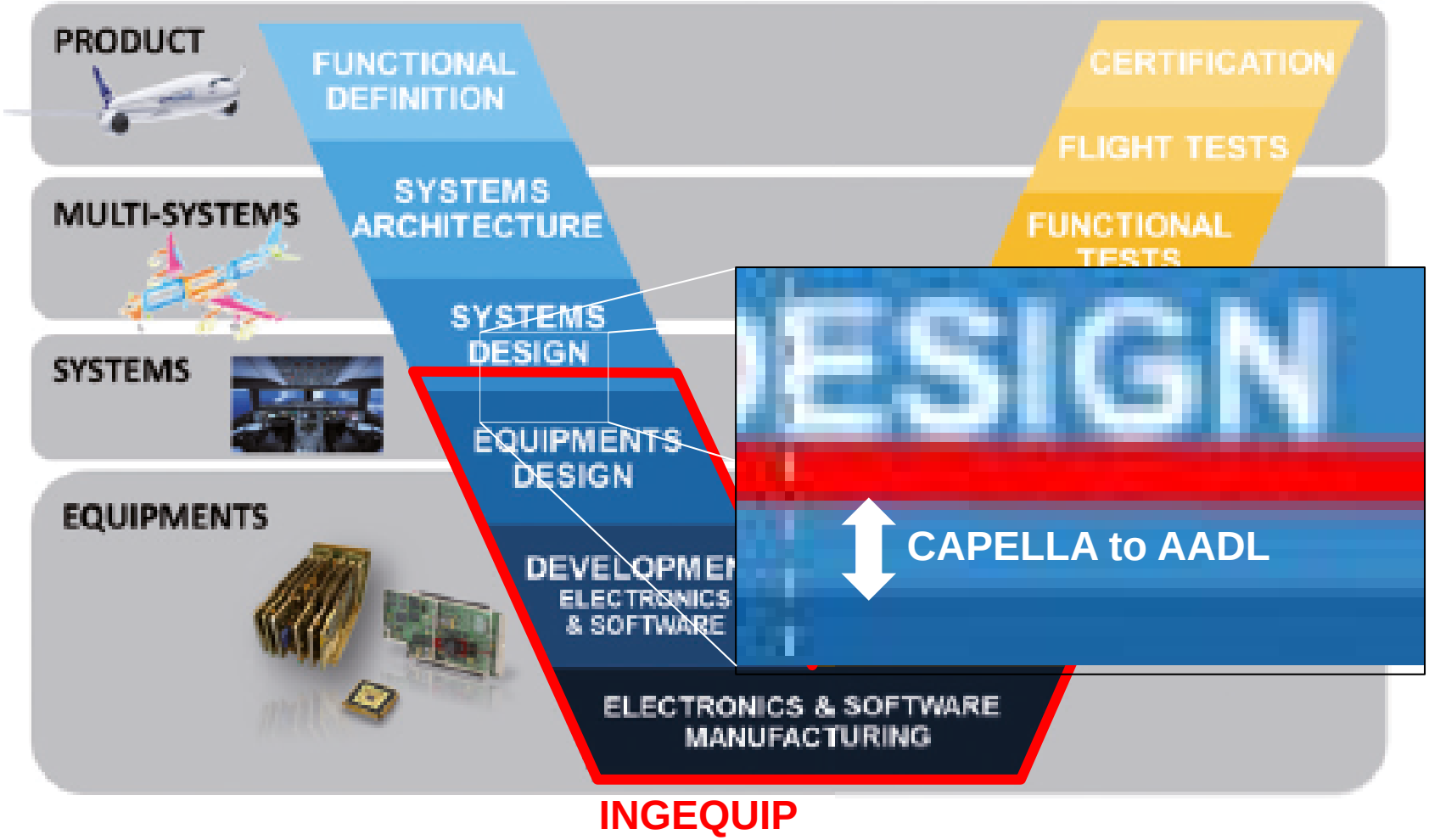


INGEQUIP
Positioning



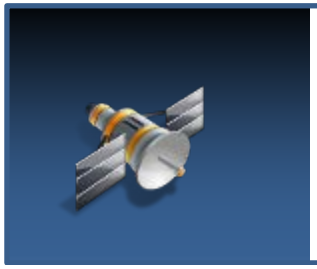
INGEQUIP

INGEQUIP
Positioning



INGEQUIP

Partners



AIRBUS DEFENCE & SPACE

AIRBUS SAGEM THALES AVIONICS

ACTIA® Continental The Future in Motion

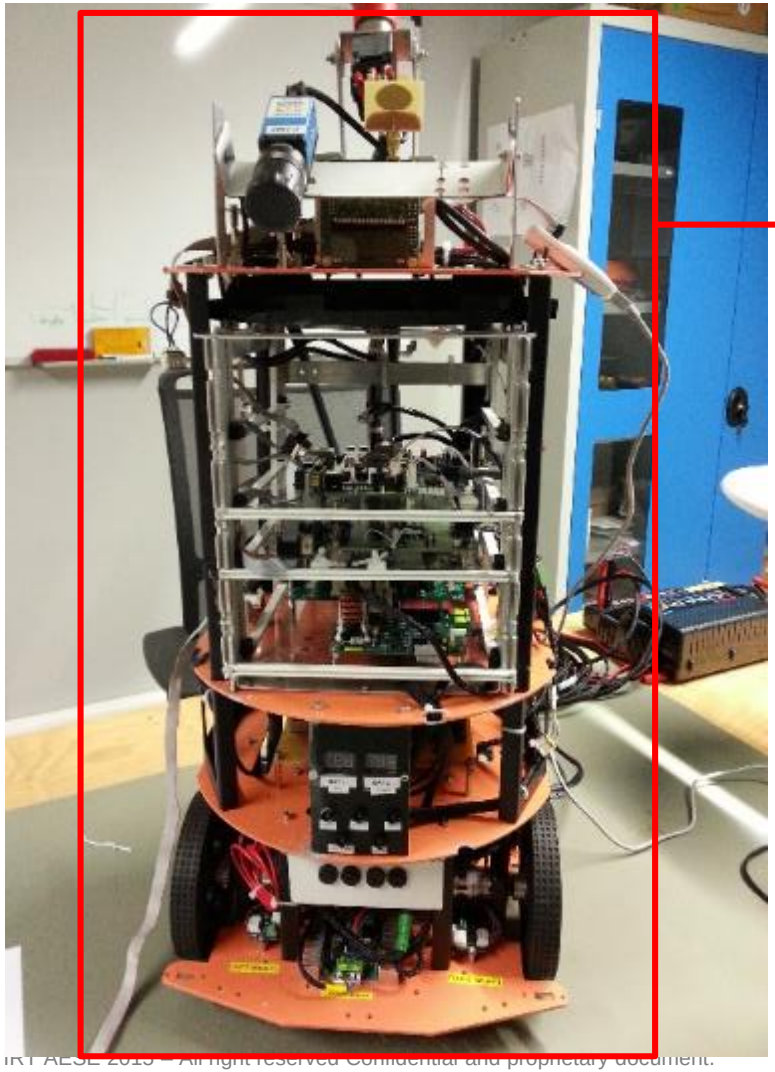
ONERA IRIT isae LAAS-CNRS

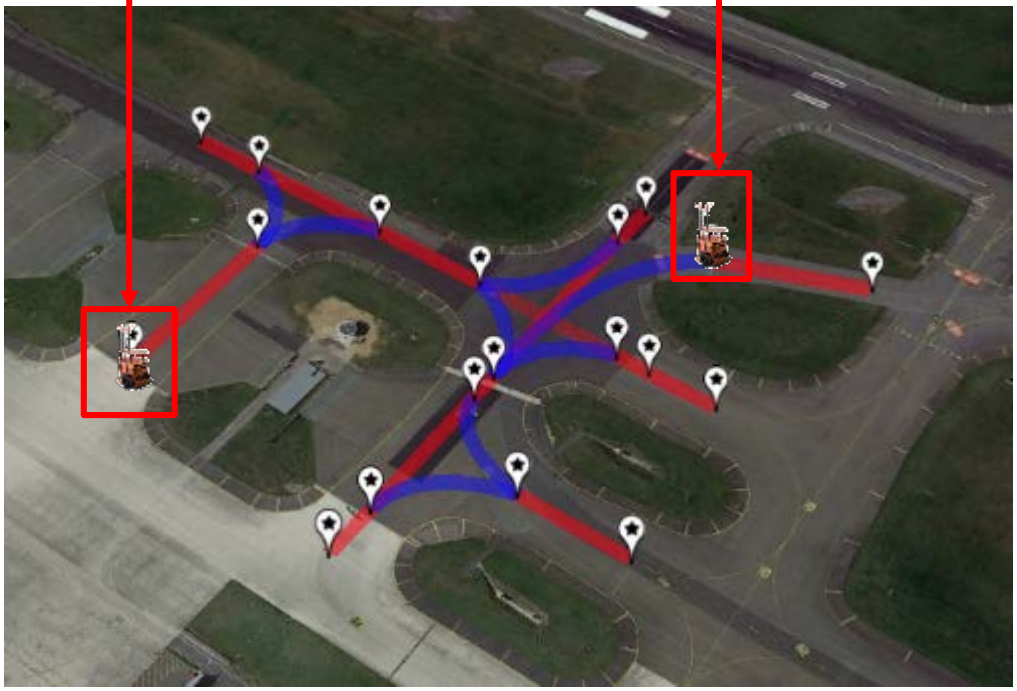
systerel Space Codesign astc design to order GreenSocs Ellidiss Technologies

Technical overview
The TwIRTee demonstrator

Our demonstrator

TwIRTee





Our demonstrator

TwIRTee

Ethernet CAM

WiFi

ethernet

EMBEDDED SYSTEM

TREK1000 (ground)

Our demonstrator
TwIRTee

Ethernet CAM

WiFi

ethernet

MISSION

CAN bus

POWER

TREK1000
(ground)

Our demonstrator
TwIRTee

Ethernet CAM

WiFi

ethernet

MISSION

CHANNEL LEFT

RIGHT

CAN bus

POWER

TREK1000
(ground)

Our demonstrator
TwIRTee

Ethernet CAM

WiFi

ethernet

MISSION

CHANNEL LEFT

MON

COM

CAN bus

POWER

TREK1000 (ground)

Our demonstrator
TwIRTee

Ethernet CAM

WiFi

ethernet

MISSION

MON: ZYNQ

COM: PPC

MON

COM

CAN bus

POWER

TREK1000
(ground)

Our demonstrator

TwIRTee

Ethernet CAM

WiFi

ethernet

MISSION

MON: ZYNQ

COM: PPC

MON

COM

CAN bus

POWER

left and right wheel

Encoder

Motor

H-bridge

discrete

PPC

TREK1000 (on-board)

I2C

Compas

Buck analog part

analog

discrete

TREK1000 (ground)

Our demonstrator
TwIRTee

Ethernet CAM

WiFi

ethernet

MISSION

ZYNQ
RMS
RAM
ARP
IBPM
MM
RCS
RM
CS
OS : LINUX

PPC
LOC
GM
MiM
PM
CS
OS: Trampoline

MON

COM

CAN bus

POWER

left and right wheel

Encoder

Motor

H-bridge

discrete

PPC
Power control
Motor control
MW: CAN driver
OS: Trampoline

TREK1000 (on-board)

I2C

Compas

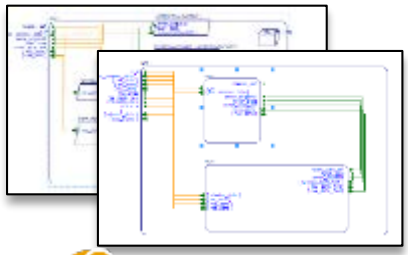
analog

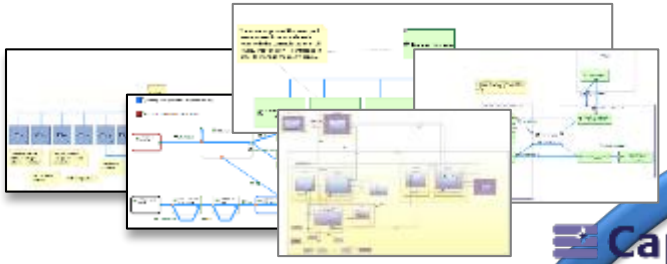
Buck analog part

discrete

TREK1000 (ground)

Models





Platform models

Simulators

AADL

Ellidiss Technologies

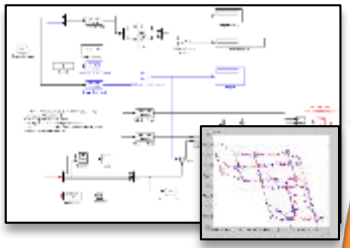
python

Capella

accellera SYSTEMS INITIATIVE™

IP-XACT

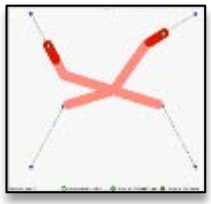
MATLAB



SystemC

C/C++

Lustre



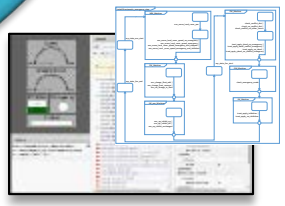
OMNeT++
Discrete Event Simulator

Fiacre

Clock synchronization models

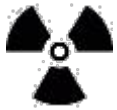
systerel

HLL/LLL



Software models

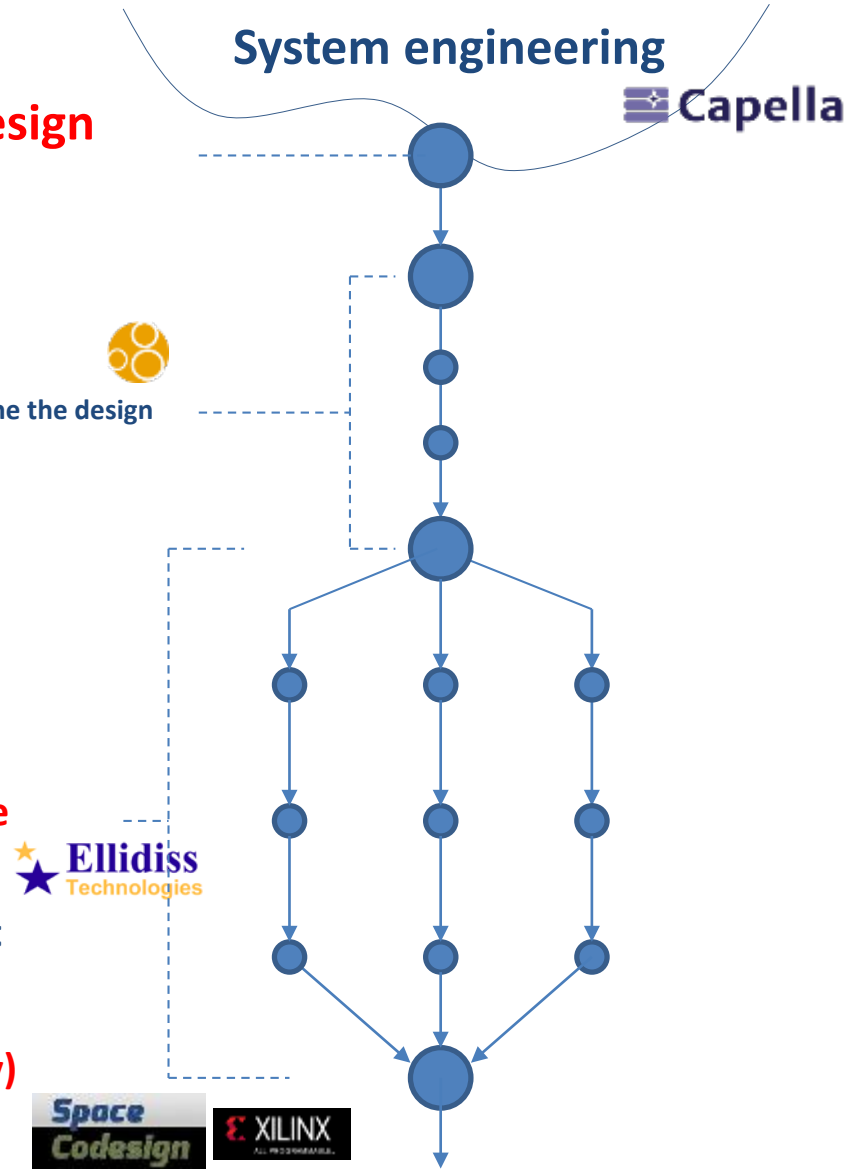
Back to the subject:
"High level" REX



REX on AADL

Step 2: Let's move on… AADL for what purpose?



REX on AADL

Step 1: Convince people…

Sorry but… some FAQs

- What is the status of the community?

- Who are the industrial users? Is it deployed on a large scale? Why not?

- What are the long term perpectives, roadmap?

- Which tool to do what?
- What is the maturity of the tools?
- Do the tools scale up?

> *"Come, let Us go down and there confuse their language, that they may not understand one another's speech."*
> *So the Lord scattered them abroad from there over the face of all the earth, and they ceased building the city."*

- How does it compare to other architectural languages (MARTE, EAST-ADL, <insert your own personal DSL>…)?

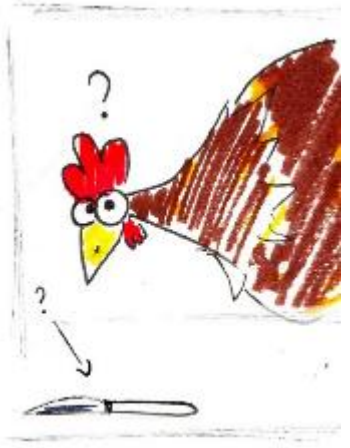
- One (language) to rule them all ? Is it really the appropriate approach?

REX on AADL

AADL to communicate

What?

- AADL for *human to human* communication (including self-comm.)



Achievements

- Most models elaborated by newbies with no **big** difficulty
  (at least for the software architecture)

Conclusions / questions

- Pretty friendly for SW engineers (textual concrete syntax, cut-paste,…)
  - (Usual) lack of automatic graphic representation, viewpoints,…
- Not so friendly for non-SW engineers (where’s my "box"?, my "block"?)
  - *Likely to change with the new generation of system designers (familiar with SysML, CAPELLA) and hardware designer (familiar with SystemC, …)*

REX on AADL

AADL to generate code

What?

- Generation of application code skeleton

Achievements

- Not really experimented…
- No support for part of our platform (Trampoline and OSEK)
- No generation of OIL configuration files

Conclusions / Questions

- Maturity / stability of the code generation tool?
- Capability to qualify the code generation tool? or alternative strategy?
- Traceability between the AADL and the generated code?
- If platform X is supported, under what hypotheses are the properties verified on my AADL **model** preserved on my actual **system**?

REX on AADL

AADL to perform (formal) timing analysis

What?

- Formal verification of temporal / timed properties
- Our use case: a clock synchronization protocol

Achievements

- Not really experimented, but…
- Significant effort on the FIACRE side (intermediate language that used to be generated from AADL + BA) in order to prepare the next step towards AADL+BA

Conclusions / Questions

- What about the modeling of multiple time sources on a distributed system (*)? (*a la MARTE clock description language)?*

(*) Including low level hardware systems with multiple clocks

REX on AADL

AADL for platform generation

What?

- Generation of configuration files for virtual platforms

Achievements

- First prototype of AADL to VP for Space Codesign using OCARINA

From AADL to SpaceCo

Automatic transformation using Ocarina

```
-- ----------------
-- A Zynq processing unit with a 2-core Xillinx
of RAM
-- ----------------
system implementation IBPM_Pf.zynq_2
    subcomponents
        the_cpu: system multicore.zynq_2;
        the_mem: memory hw_components::DDR {
        SpaceStudio::Subtype_component => gaisl
        the_mem_bus: bus { SpaceStudio::Subtype
        AMBA_AXIBus_LT; };
    [...]
end IBPM_Pf.zynq_2;
```

```
  -- Hough: hand-coded C implementation
  subprogram implementation Hough.hc
      properties
          Source_Text => ("hough.cpp");
          Source_Name => "hough";
  end Hough.hc;

  --  A generic Hough thread
  thread HoughThread
      features
          the_contour: in data port Domain::Image;
          the_segment_set: out data port Domain::Se
  end HoughThread;

  -- A generic implementation of a Hough thread
  thread implementation HoughThread.gen
      calls
          c1: {s1: subprogram Hough;};
  end HoughThread.gen;
```

```
PROJECT_NAME = 'main.zynq_2'
BUS_NAMES = [('busamba','AMBA_AXIBus_LT')]
MEMORY_NAMES = [('DDR','gaislerddr','busamba')]
# processor (processor_name, processor_type, simulation_model, binding_bus,
scheduling_protocol)
PROCESSOR_NAMES = [('cpu1','armCortexA9','None','busamba','round_robin'), ('cpu2',
```

```
PROCESSOR_NAMES = [('cpu1','armCortexA9','None',
'armCortexA9','None','busamba','round_robin')]
```

```
,'cpu2',18,'none',0), ('system_SegmentFiltering_P1','cpu2',18,'none',0), (
'system_PositionMonitoring_P1','cpu2',18,'none',0)]
DEVICE_NAMES = []
```

```
def create_designs(project, name, module_names, device_names, processor_names, bus_names, me
    Design_name = name + '.design'
```

```
def create_designs(project, name, module_name
    Design_name = name + '.design'
```

```
        bus[bus_name[0]] = design.createComponentInstance('Bus',bus_name[1])
    Timer = design.createComponentInstance('Timer', 'XilinxTimer')
    Timer.connectTo(bus[bus_names[0][0]])
    for memory_name in memory_names :
        memory = design.createComponentInstance('Memory', memory_name[1])
        memory.connectTo(bus[memory_name[2]])
    for processor_name in processor_names :
        if processor_name[2] is None:
            processor[processor_name[0]] = design.createProcessorInstance(bus[processor_name
            processor_name[1])
        else :
            processor[processor_name[0]] = design.createProcessorInstance(bus[processor_name
            processor_name[1], processor_name[2])
```

```
while(1)
{
    //Execution between 10 and 40 ms

    Hough();

    #if defined(SPACE_SIMULATION_RELEASE) || defined(SPACE_SIMULATION_MONITORING) ||
    defined(SPACE_SIMULATION_DEBUG)
        DeviceRead(XILINXTIMER1_ID, XILINX_TIMER_OFFSET_TCR0, &timerValue);
        if (timerValue - initValue < 0 * 1000)
            deadline::getInstance().addDc(system_SegmentExtraction_P1);
        else
            deadline::getInstance().addDl(system_SegmentExtraction_P1);
        DeviceRead(XILINXTIMER1_ID, XILINX_TIMER_OFFSET_TCR0, &initValue);
    #endif
}
```

SystemC

REX on AADL

AADL for platform generation

What?

- Generation of configuration files for virtual platforms

Achievements

- First prototype of AADL to VP for Space Codesign using OCARINA (work of ISAE+SpaceCodesign)

Conclusions / Questions

- Wouldn’t be wiser to target IP-XACT (AADL=>IP-XACT, IP-XACT=>virtual platform)?
- Is the choice of AADL really pertinent ? (see question about hardware modeling)
  - One-language-to-rule-them-all syndrom…
- Anyway: an important step towards the automation of Design Space Exploration

REX on AADL

AADL for Design Space Exploration

What?

- Specify and automate Design Space Exploration (DSE)

Achievements

- Ad-hoc modeling of multiple design alternatives of an image processing chain (see next slides) with harcores, softcores, FPGA-implemented coprocessors
- Evaluations were performed using SpaceStudio
- No automatic optimization was performed.

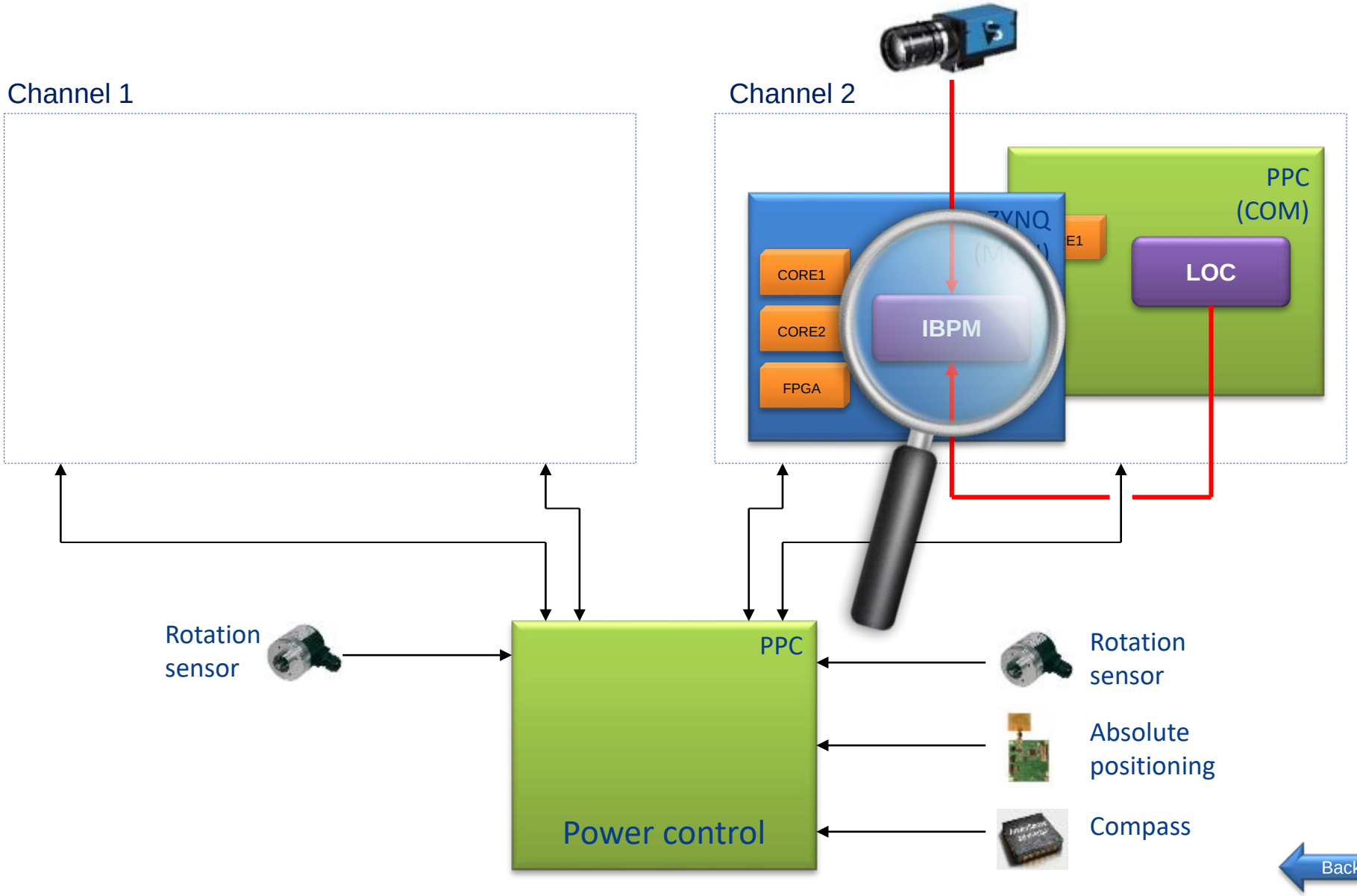
Conclusions / questions

- (Separate functional modeling from logical and physical modeling)
- How to express the "degrees of freedom" of the DSE problem
- How to express the constraints relating the various DSE parameters?
- How to generate candidate designs, evaluate them, optimize them

See the ex.

Design space exploration

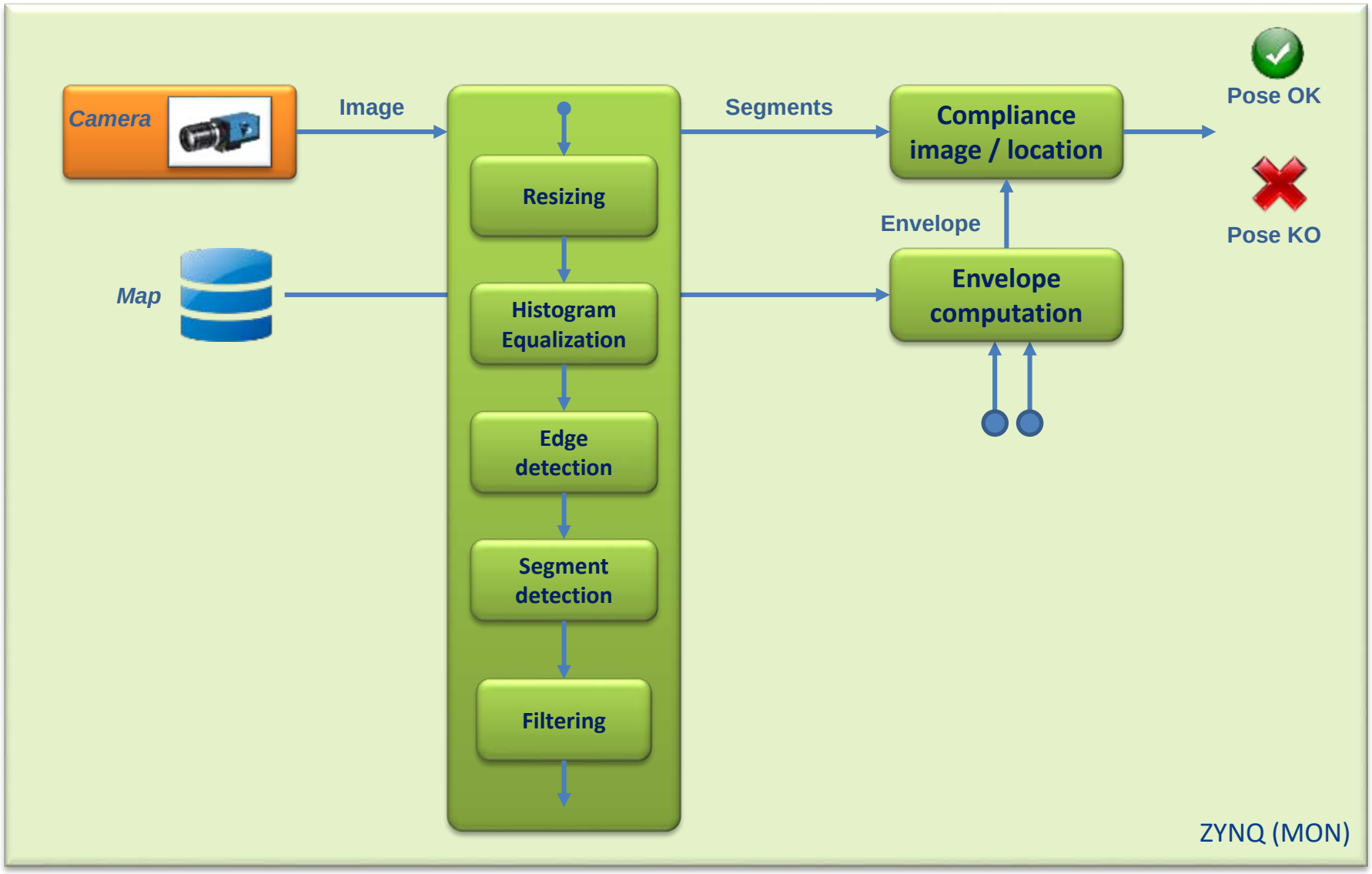
Experiment with SpaceStudio



Back

Design space exploration

Experiment with SpaceStudio



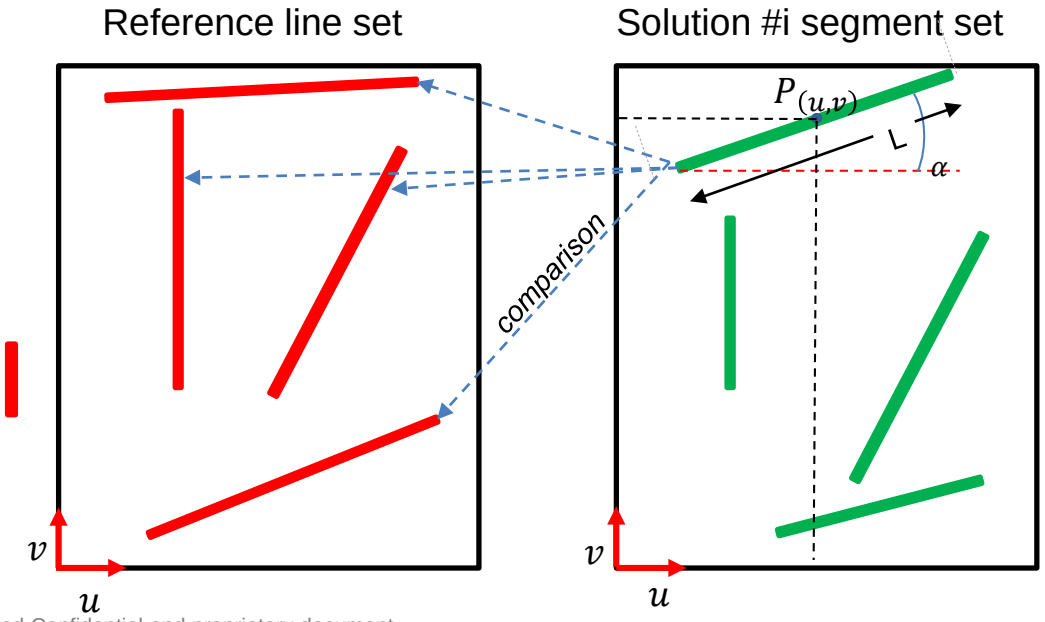
Back

Design exploration

Definition of QoS metrics

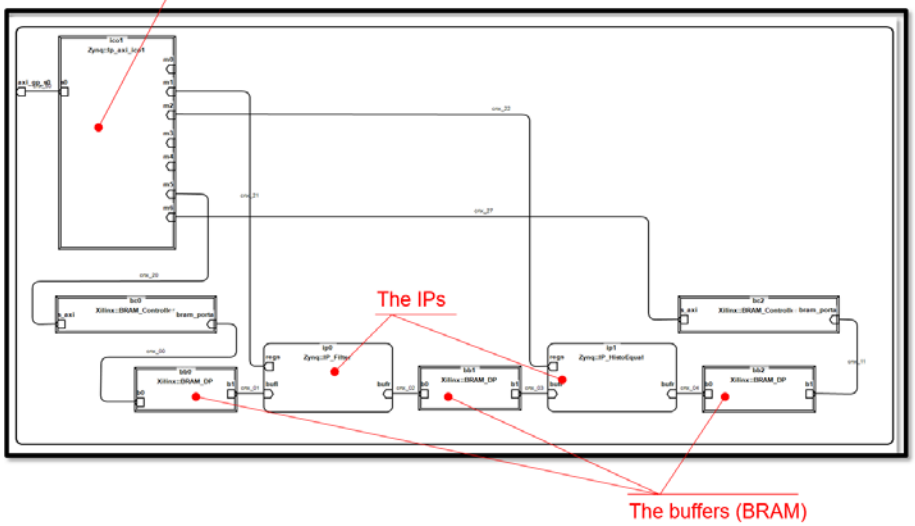
All solutions are compared to a reference (*ideal*) solution

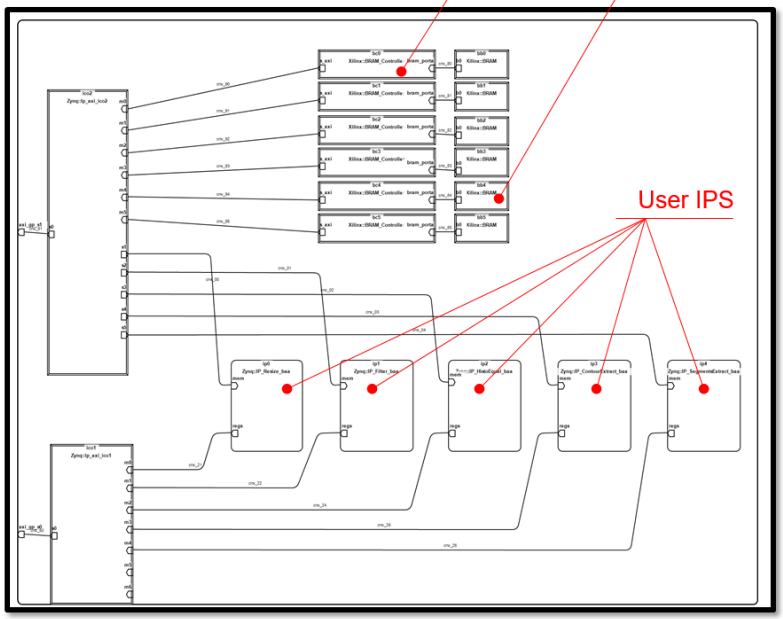
- The segment set corresponding to the part of the tracks “seen” by the camera
- We know the position of the rover at a given date
- We know the map
- We know the parameters of the camera
- We compute the set of segments which are visible by the camera in the image space

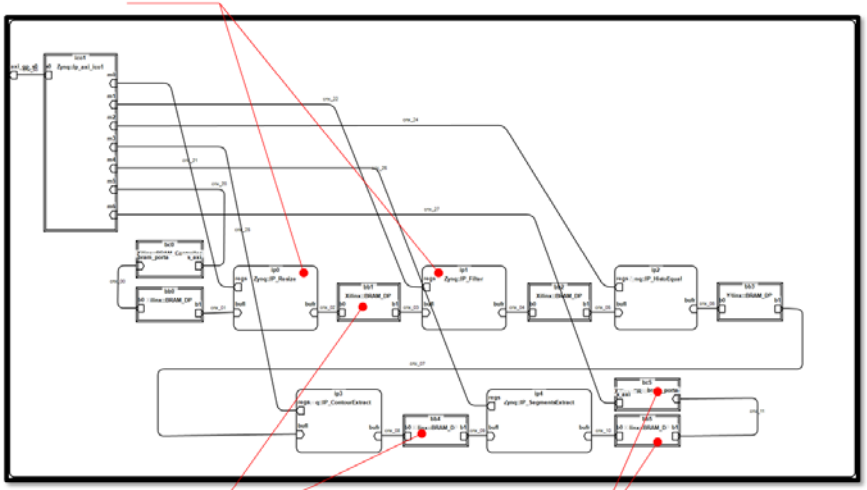


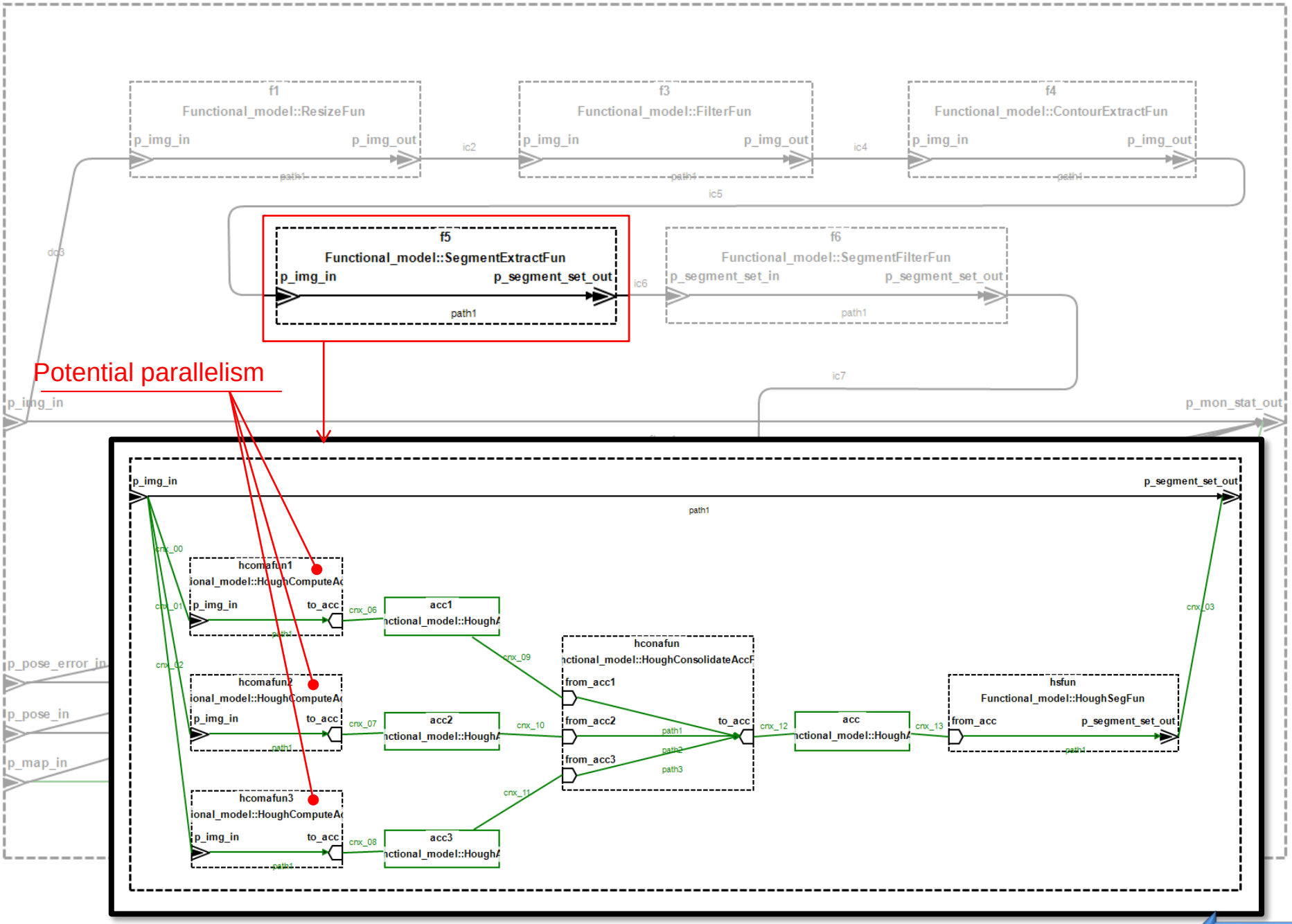
Back

Design alternatives









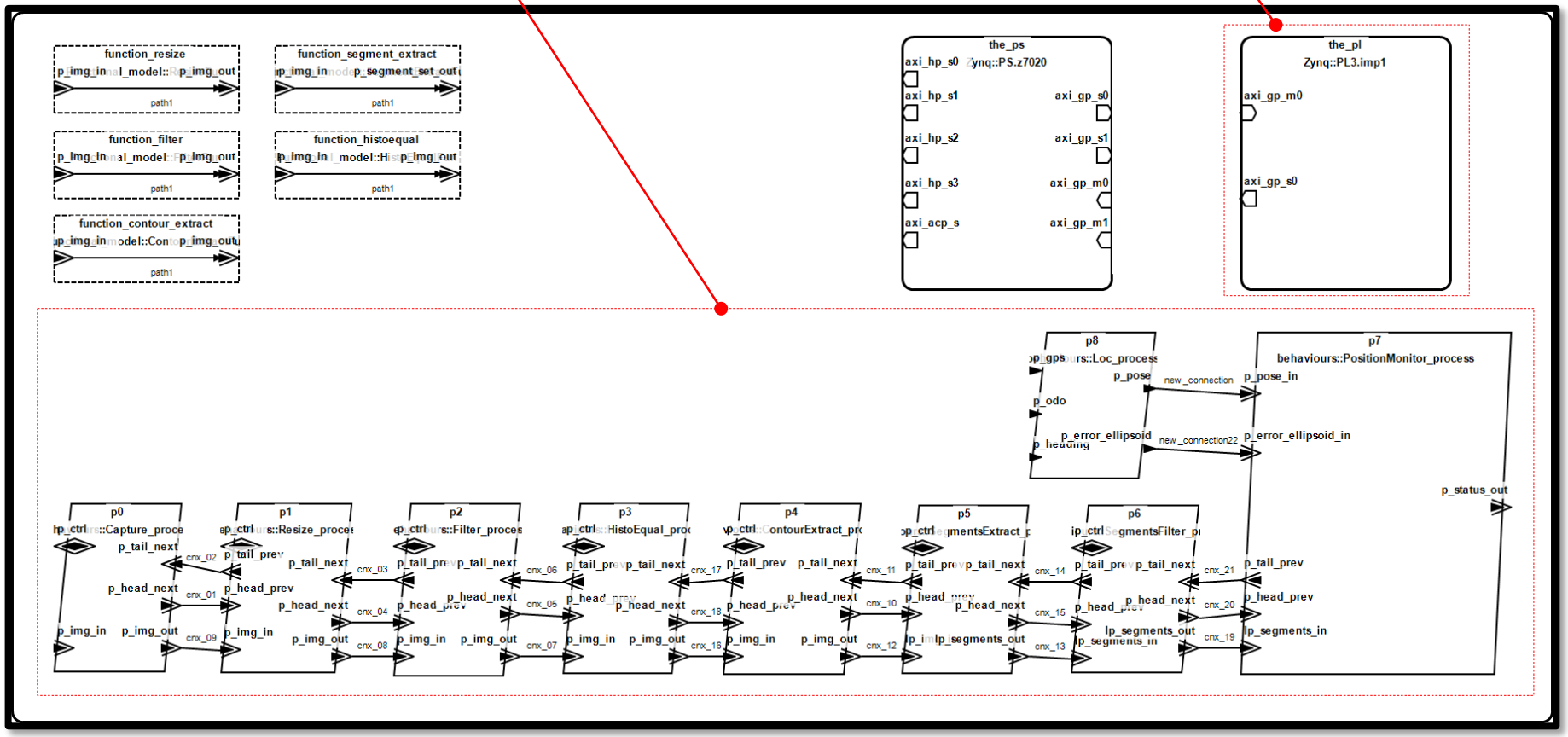
Back

AADL for Design Space exploration

Implementation solutions

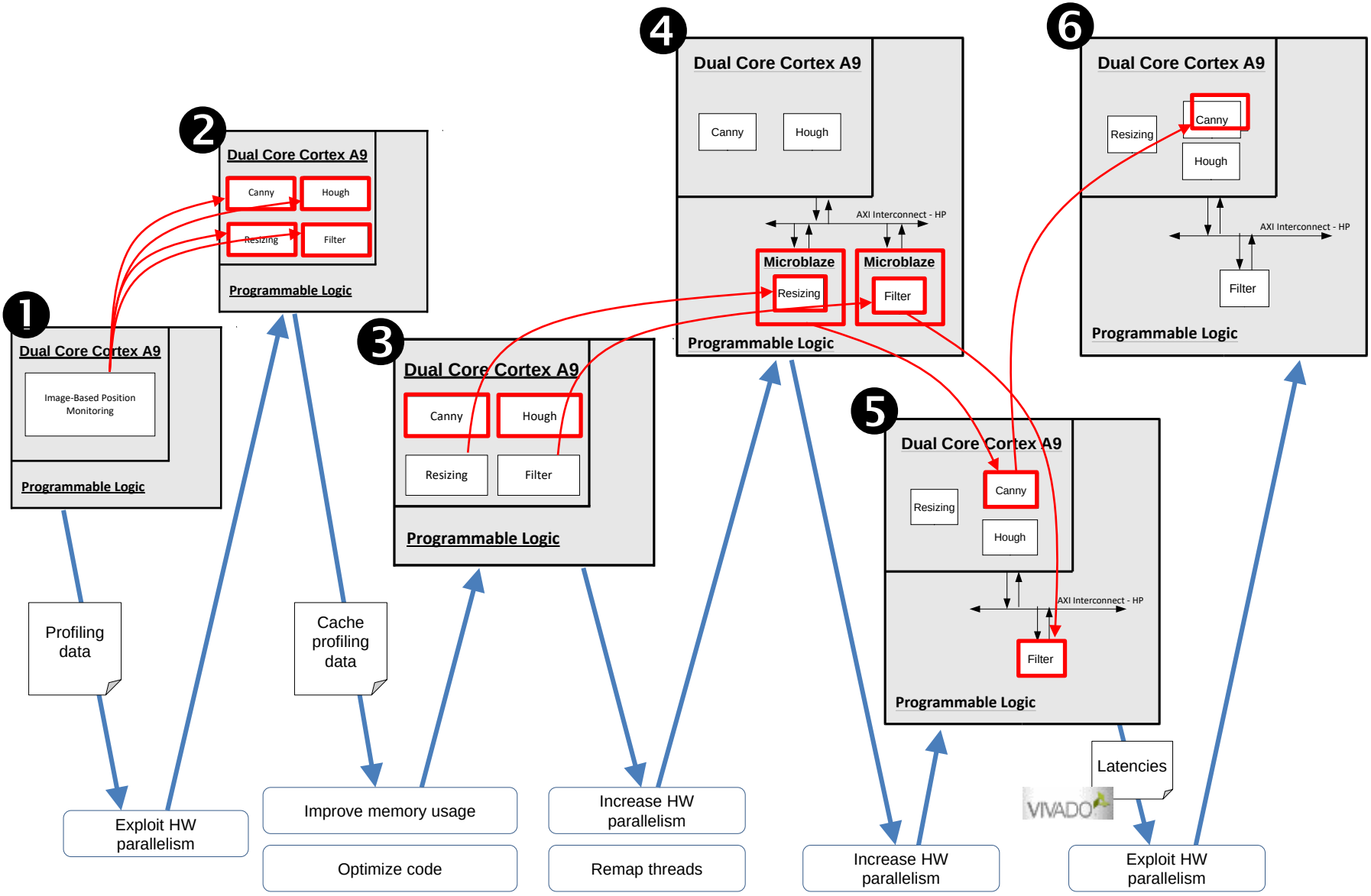
The Programmable Logic (FPGA)

The behaviors



Back

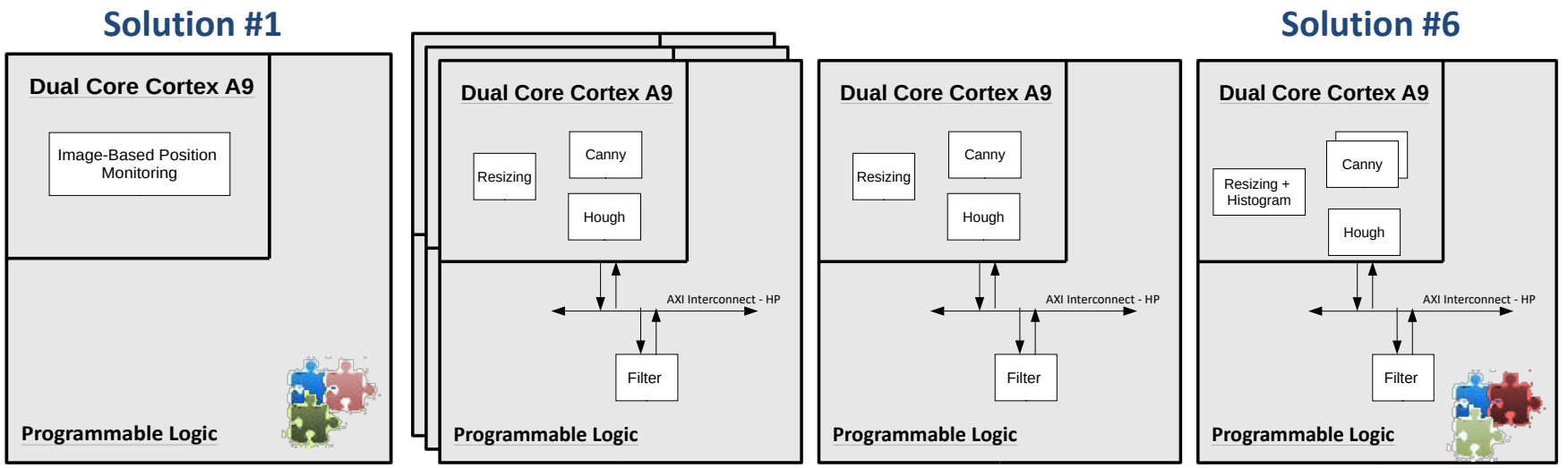
Evaluation process



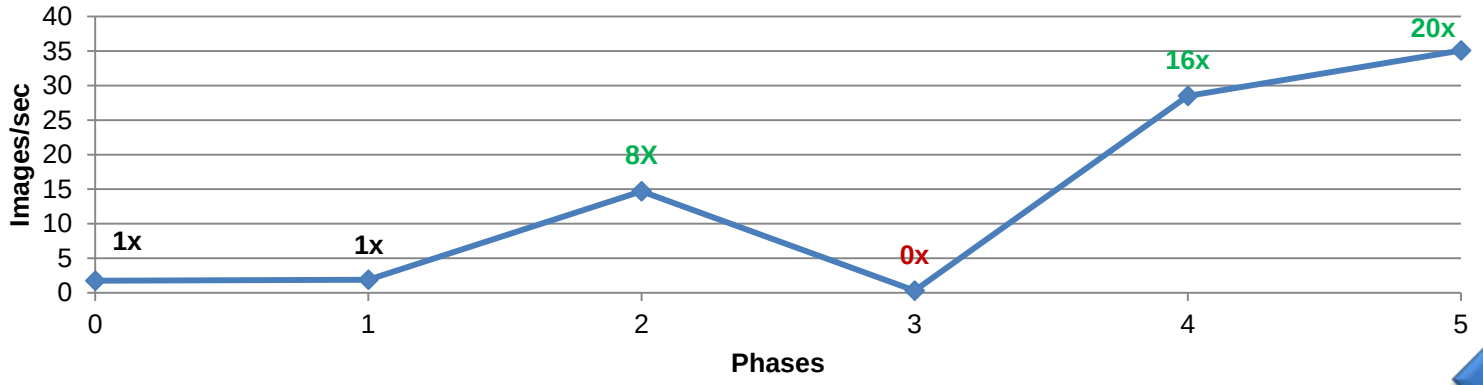
Design space exploration

Experiment with SpaceStudio

Solutions

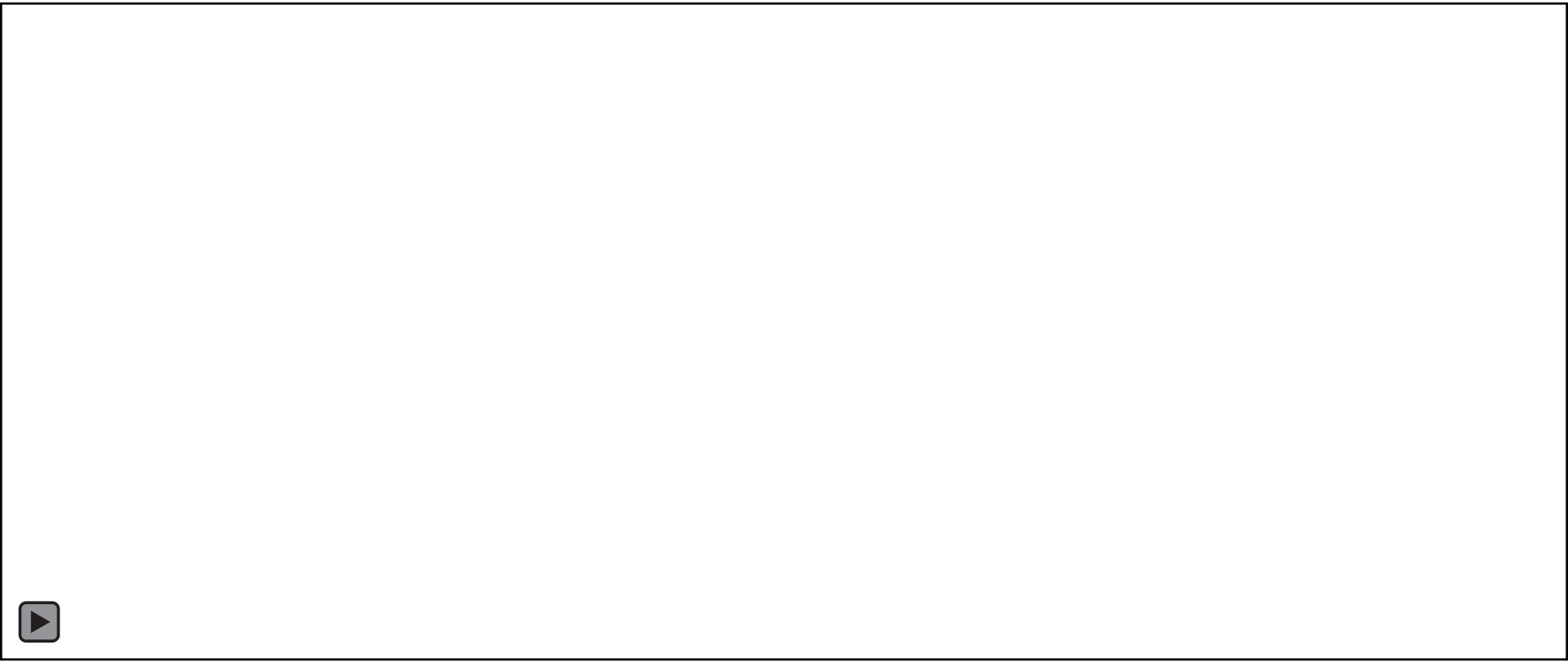


Speed-up of Image Processing Chain algorithm over the design space exploration



Go back

Time for a break…



…the ARP (non proved implementation) in action

REX on AADL

AADL for empirical validation

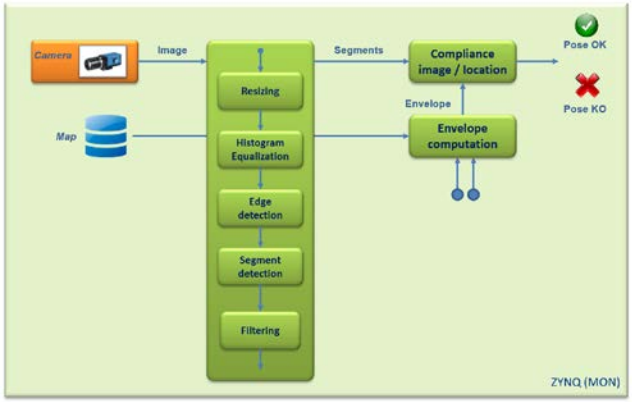
What?

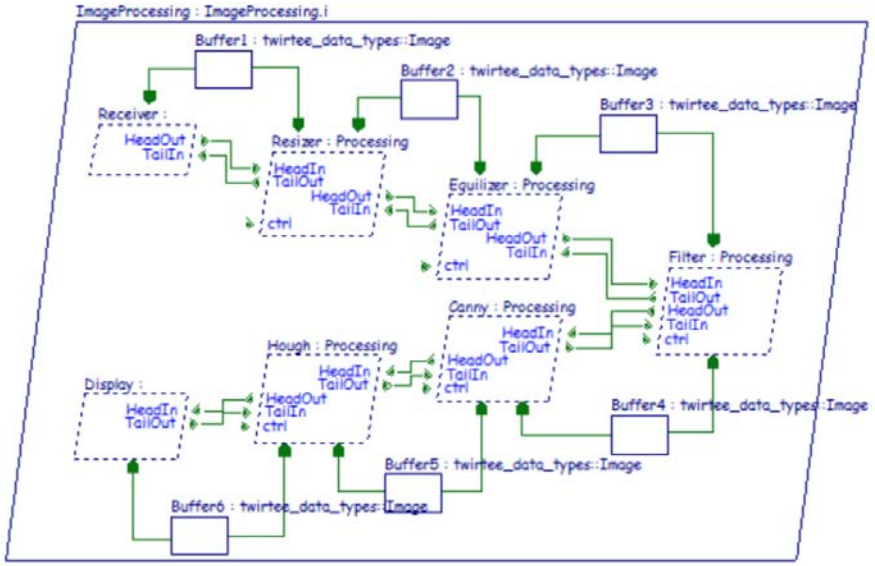
- High-level validation of architectural choice regarding temporal properties (incl. Scheduling with HW and SW events)

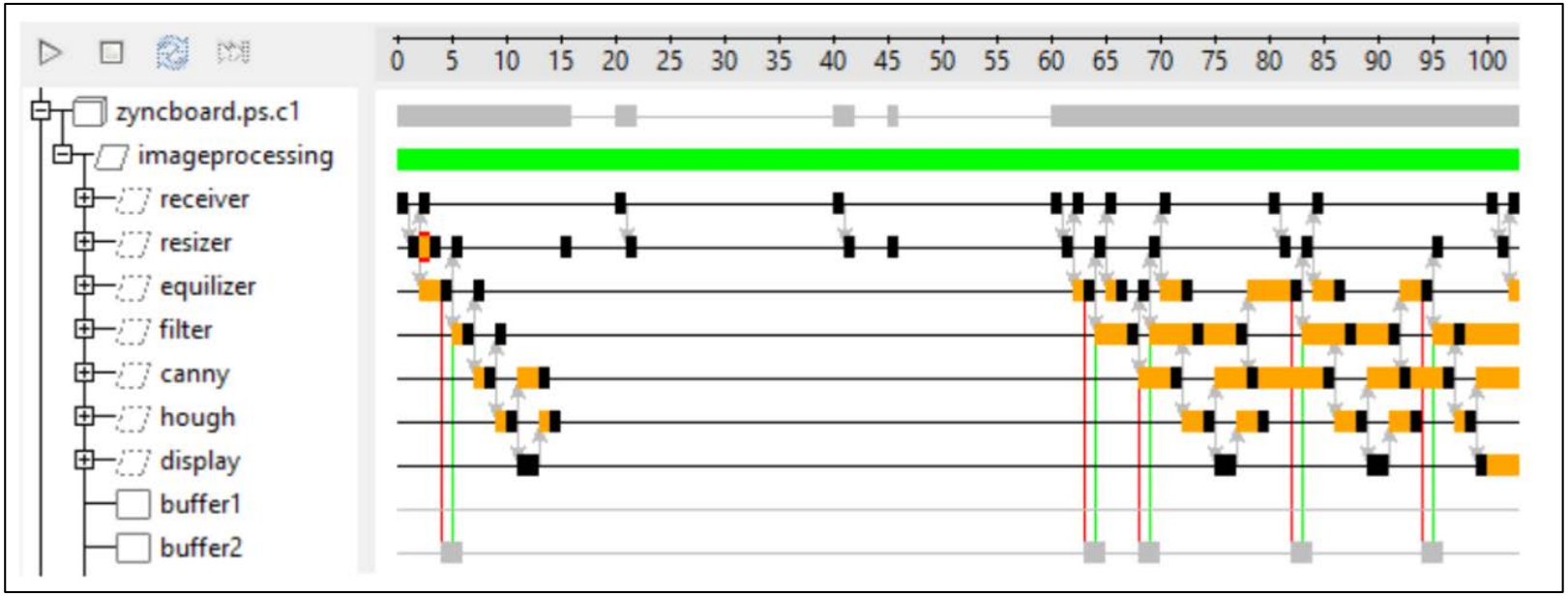
Achievements

- (Very) simple model of the *Image Processing Chain* pipeline (see next slides)
- Simple model of the buck controler (see next slides)

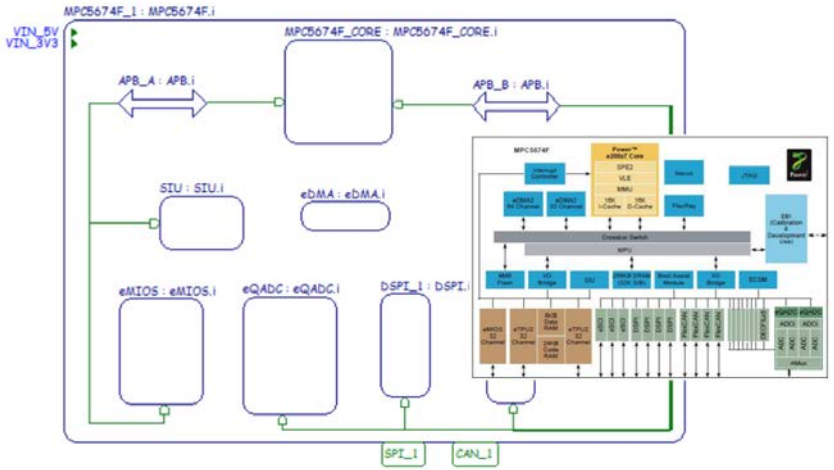
The IBPM

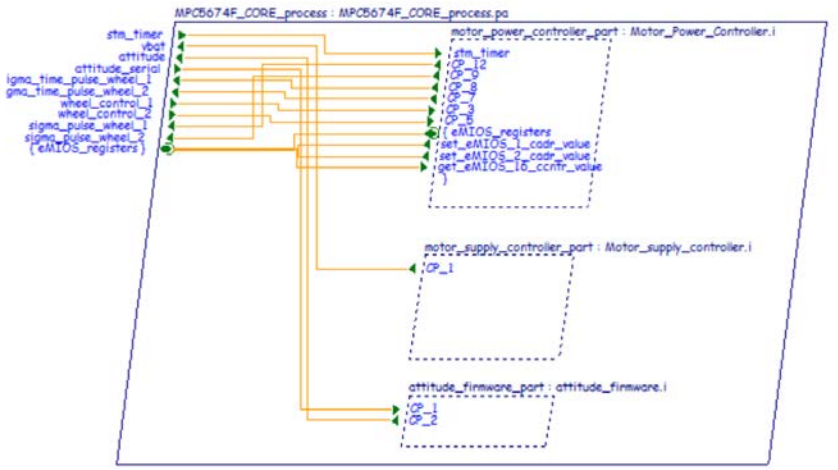


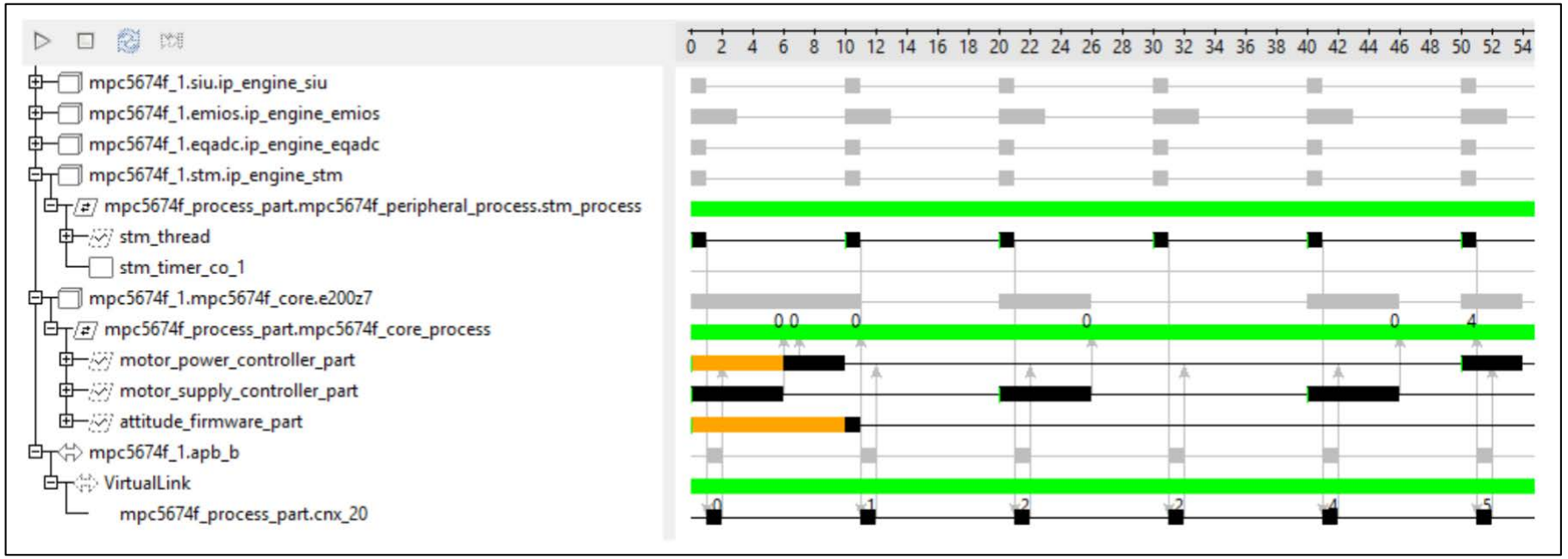




The Buck controler







REX on AADL

AADL for empirical validation

What?

- High-level validation of architectural choice regarding temporal properties (incl. Scheduling with HW and SW events)

Achievements

- (Very) simple model of the *Image Processing Chain* pipeline (see next slides)
- Simple model of the buck controler (see next slides)

Conclusions / Questions

- Simulation is a key feature for many people
- "Interesting" results (even though those results are questionable)
- A complete (with respect to BA) and reliable simulator is missing. Simulator X is promising but need to be improved. And is the design paradigm really appropriate? robust?
- Model application stimulation scenarii (Not investigated yet… possible link with Accellera's Portable Stimulus?)
- How to connect an AADL simulation with other simulators?

REX on AADL

AADL for (low-level) SW and HW /SW modeling

What

- Use AADL to account for HW/SW effects
- Estimate bus loads, CPU loads

Achievements

- First models of software down to the hardware such as IP configuration registers (e.g. Autosar like)

Conclusions / questions

- Requires modeling of HW/SW interactions: modeling of inter-core interferences due to the sharing of resources (incl. interconnect, caches effect,…)
- Requires modeling of low-level hardware devices (interconnect, behavior…)
- Is it pertinent?
- Abstraction is not always easy… (if one wants to maintain precision and accuracy)

REX on AADL

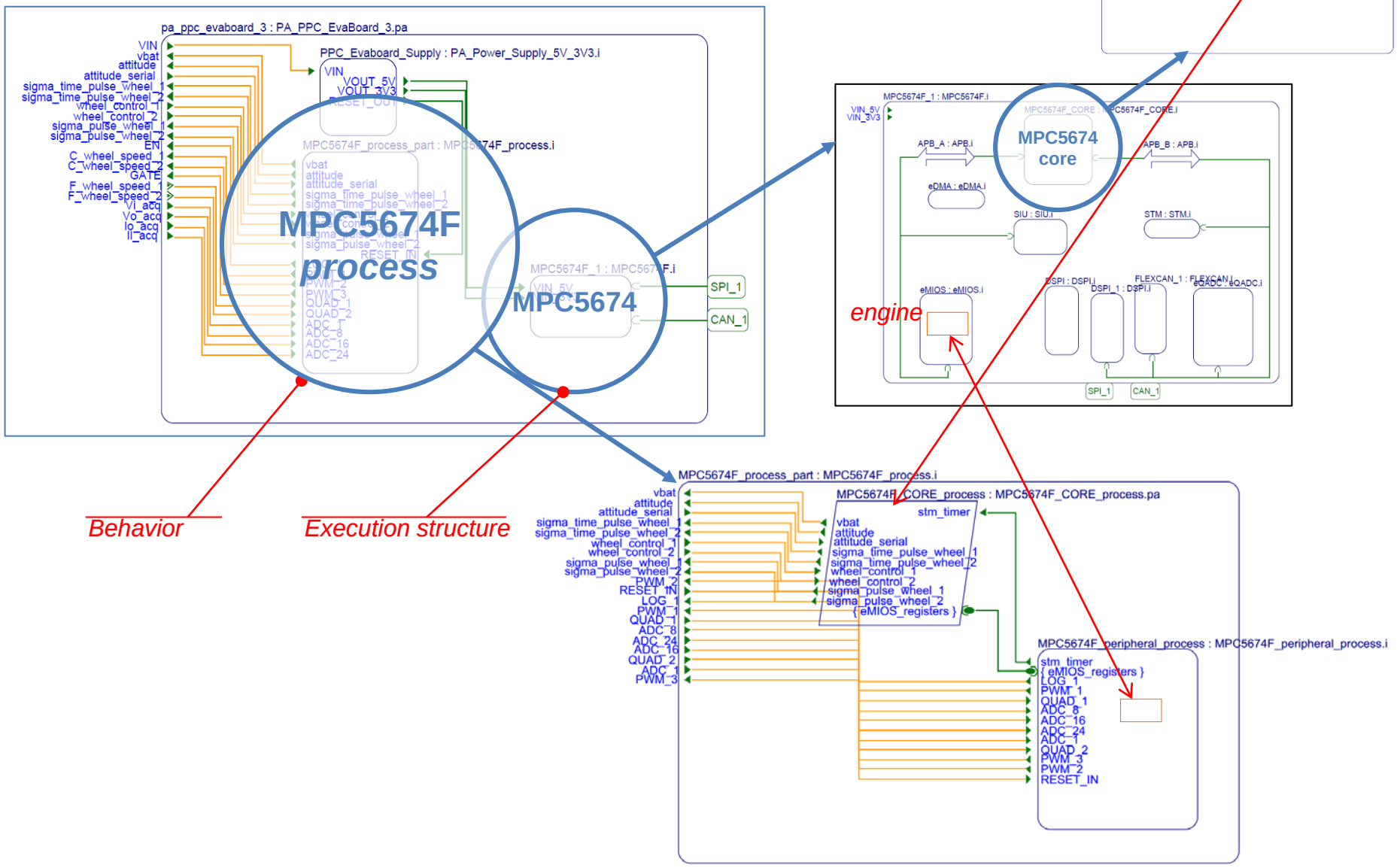
AADL for hardware modeling

What

- Use AADL to model the hardware
  - HW elements relevant to the configuration of virtual platforms (*already mentioned*)
  - HW elements that impact significantly SW performances (interco, cache coherency, DMA, caches effect, synchronous and asynchrounous hardware trigger…)
  - HW elements relevant to software design (e.g., IPs control regs)

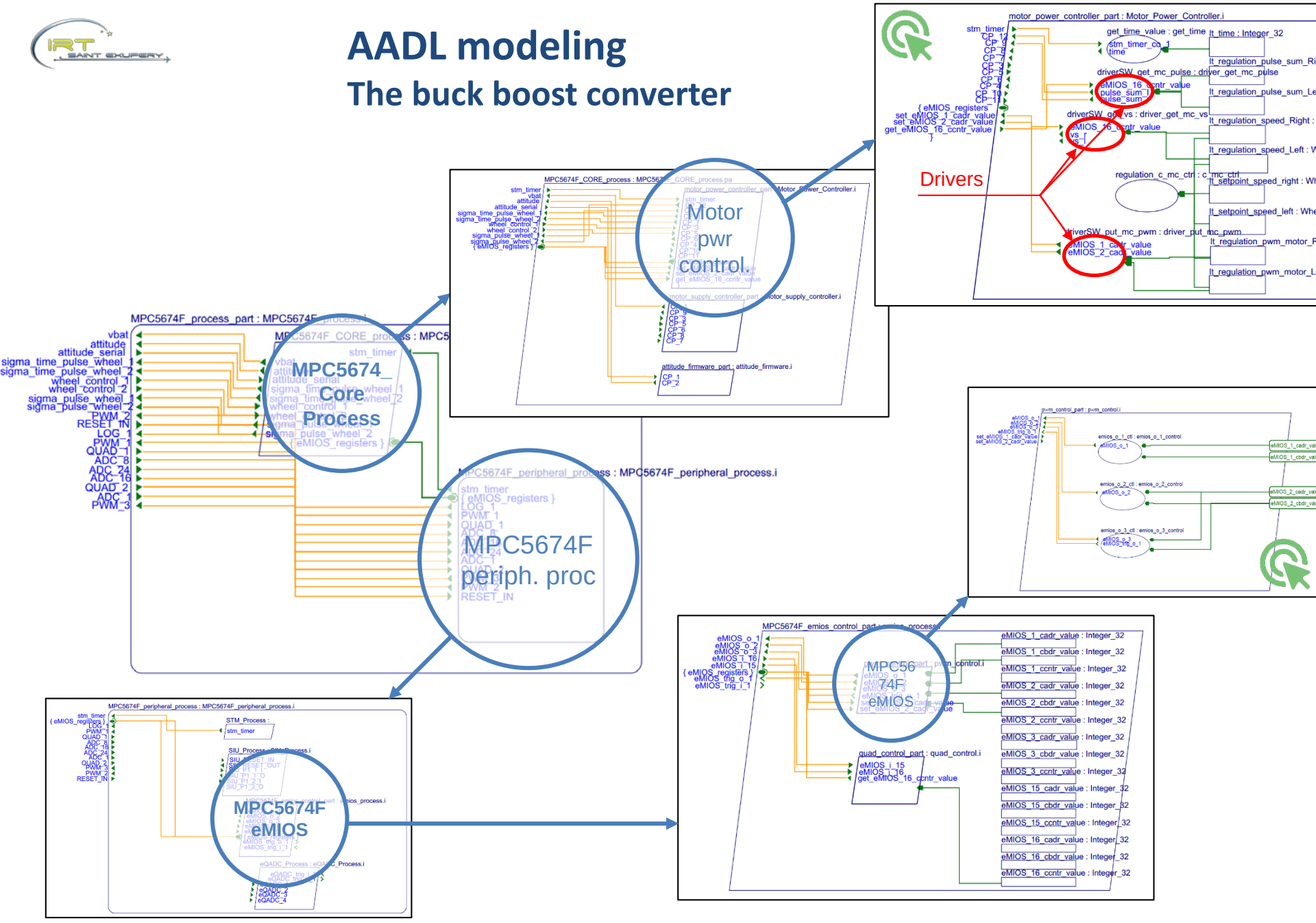
AADL modeling

The buck boost converter



AADL modeling

The buck boost converter



REX on AADL

AADL for hardware modeling

What

- Use AADL to model the hardware
  - HW elements relevant to the configuration of virtual platforms (*already mentioned*)
  - HW elements that impact significantly SW performances (interco, cache coherency, DMA, caches effect, synchronous and asynchrounous hardware trigger…)
  - HW elements relevant to software design (e.g., IPs control regs)

Conclusions / Questions

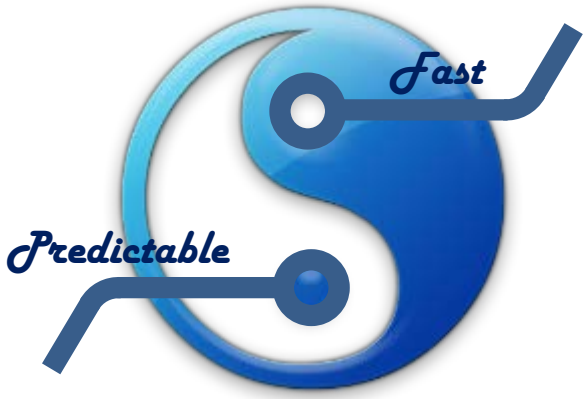
- Models distinguish the static architecture (block and interface) from the behavior (abstraction of IP behavior for the scope of the application)
- Execution semantics of hardware (parallel execution and synchronous behavior, how does it fits with the current standard)
- Multiple clocks (with drift, offsets,…)

REX on AADL

AADL for hardware modeling (continued)

- Modeling of low level hardware signals is heavy / non intuitive (for a HW engineer): modeling of the logical connection then modeling of the hardware connection using buses (a simple wire would be better)
- Is an hardware block a *system* or a *device* (we used a « system » due to tooling constraints). How shall a hardware-implemented behavior be represented (currently: a thread »)?
- How to ensure consistency between the device and its « implementation » as a system…
- Is using feature groups to model Hardware Register declaration (and access accross software architecture) the appropriate choice?
- Other preoccupation not investigated in the project
  - Modeling of Bus bridge modeling with parallel access (component properties not investigated yet)
  - Modeling of effects of prefetch from flash on bus loading
  - Modeling of Effects of DMA traffic
  - Modeling of interrupt and reset behavior (is it possible via event port?)

And next…



CAPHCA

Critical Applications
on
Predictable High-Performance Computing Architectures

INGEQUIP

Partners

5 industrial partners

2 industrial partners

4 labs

5 technology providers in EDA

AADL in CAPHCA

“To provide methods and tools to develop safety critical applications using modern, high-performance, multi-core SoCs.”

Yes. We are not the first.

Targets:

- Aero: T1040 / LS1088
- Auto: AURIX 2G / 1G
- PRET architecture (based on Risc-V)

Multicore schemes: AMP or BMP schemes

AADL in CAPHCA

Focus

Determinism

- Where are the sources of SW/SW, SW/HW non determinism?
- How to model them?
- How to mitigate them?

Performance

- How to express and exploit potential parallelism
- How to find the best configuration (task mapping)
- How to obtain performance measures as early as possible during the development

AADL in CAPHCA

Potential role for AADL

- Model parallel architectures (are AMP and BMP supported by AADL?)
- Model HW and SW/HW interactions
  - Low-level SW/HW effects and SW/SW interference channels (via shared res.)
  - Interference analysis
  - Virtual platform configuration
  - (Support DSE.)
- Model instrumentation (what shall be observed?…)
- Model stimuli
- Verify behavioral properties related to concurrency (deadlocks, race conditions,…)
- Verify schedulability
- Other aspects: evaluation of PRET architectures, evaluation of Krono-Safe technology

(SW and) HW design using AADL

The buck converter

Our demonstrator

The buck-boost converter

Ethernet CAM

WiFi

ethernet

MISSION

ZYNQ
RMS
RAM
ARP
IBPM
MM
RCS
RM
CS
OS : LINUX

PPC
LOC
GM
MiM
PM
CS
OS: Trampoline

MON

COM

CAN bus

POWER

left and right wheel

Encoder

Motor

H-bridge

discrete

PPC
Power control
Motor control
MW: CAN driver
OS: Trampoline

TREK1000 (on-board)

I2C

Compas

analog

Buck analog part

discrete

TREK1000 (ground)

Our demonstrator

The buck-boost converter

Ethernet CAM

WiFi

ethernet

MISSION

ZYNQ

RMS

RAM

ARP

IBPM

MM

RCS

RM

CS

OS : LINUX

PPC

LOC

GM

MiM

PM

CS

OS: Trampoline

MON

COM

CAN bus

POWER

left and right wheel

Encoder

Motor

H-bridge

discrete

PPC

Power control

Motor control

MW: CAN driver

OS: Trampoline

TREK1000 (on-board)

Compas

I2C

Buck analog part

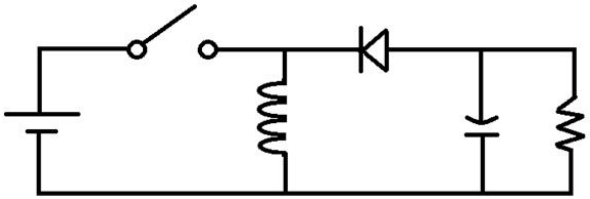
analog

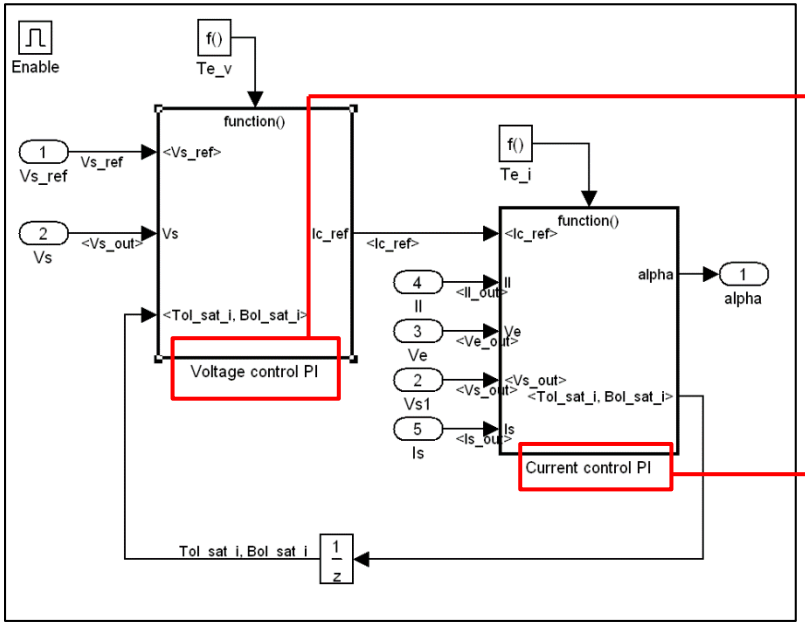
discrete

TREK1000 (ground)

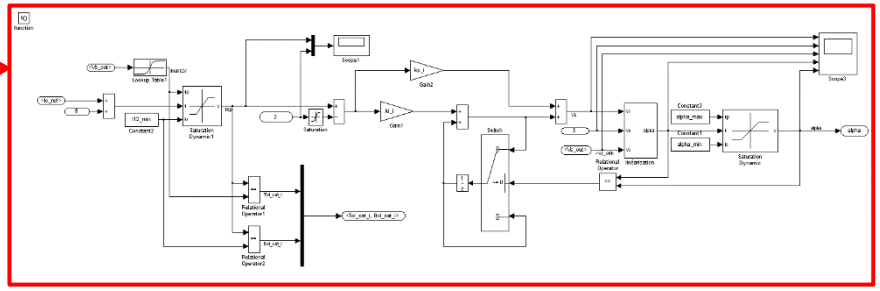
AADL modeling

The buck boost converter

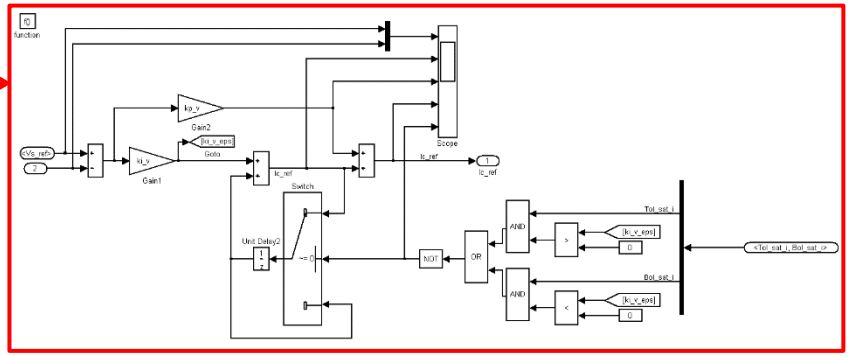




PI voltage control

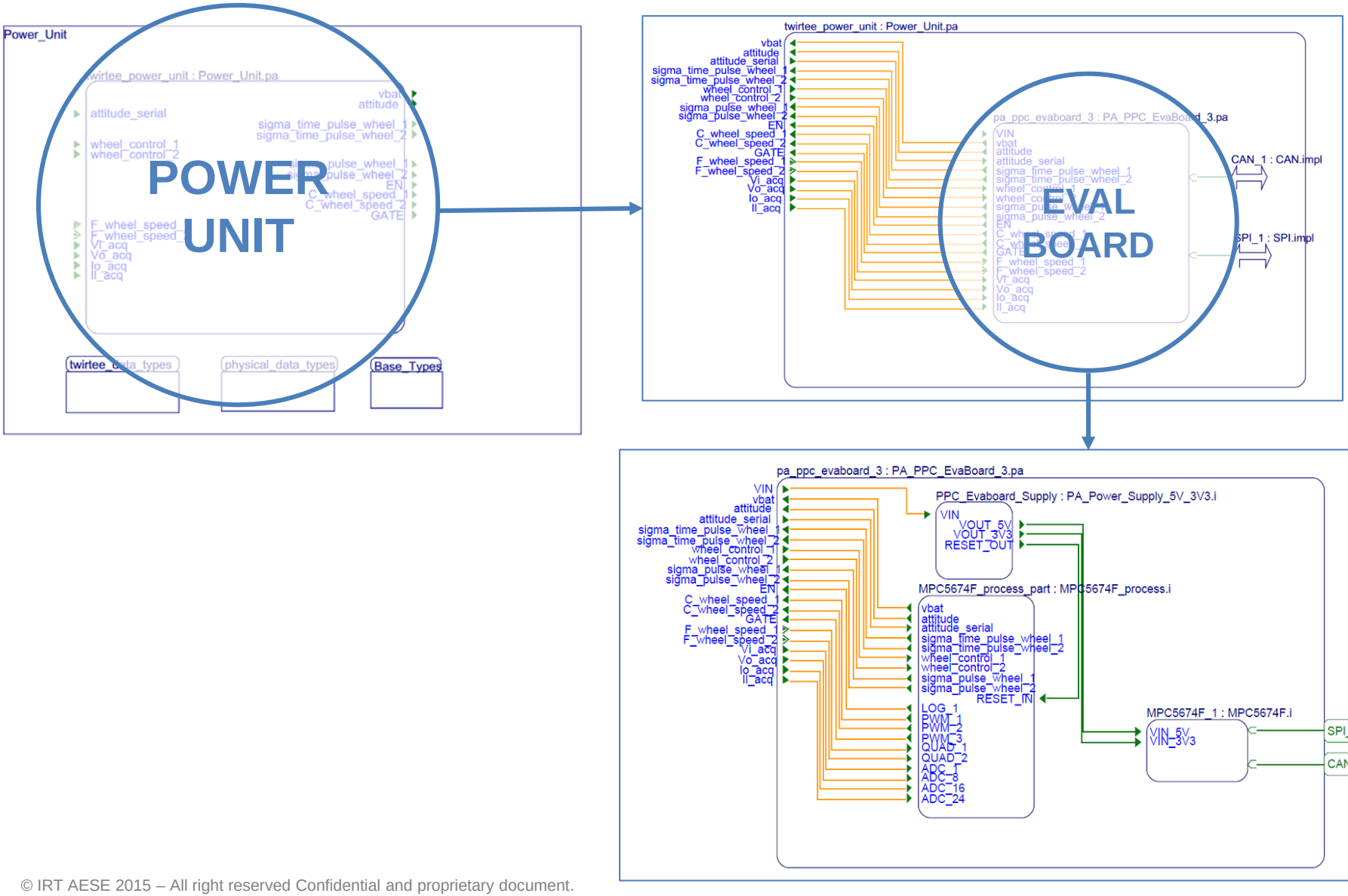


PI current control



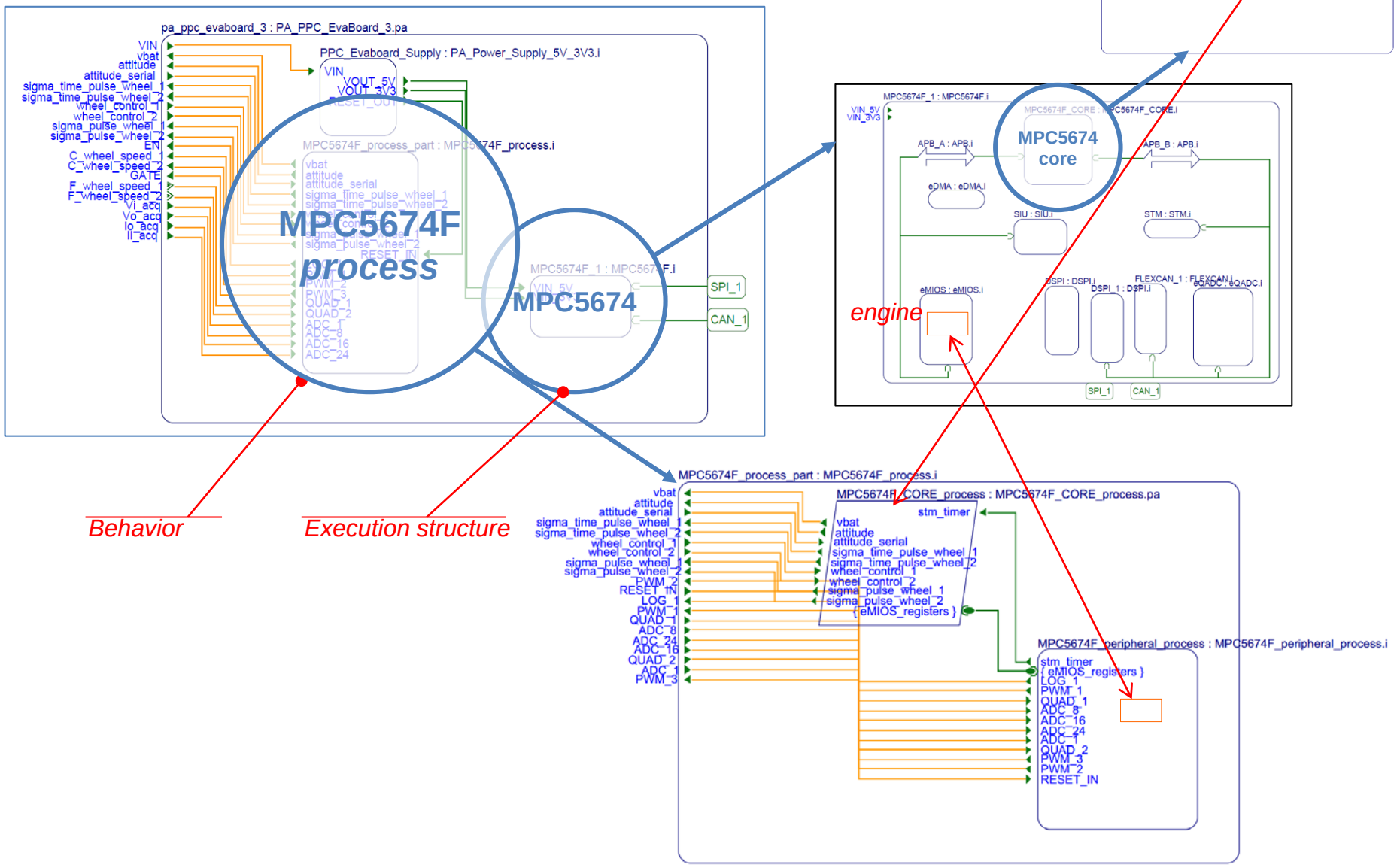
AADL modeling

The buck boost converter



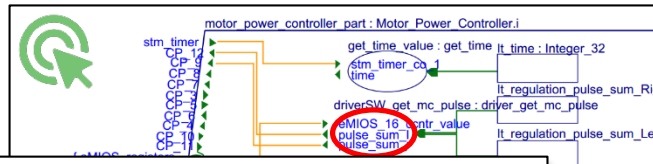
AADL modeling

The buck boost converter

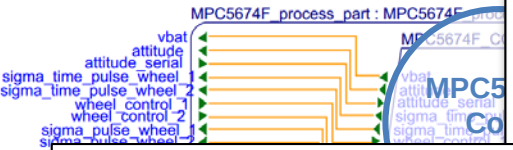


AADL modeling

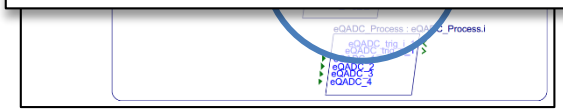
The buck boost converter

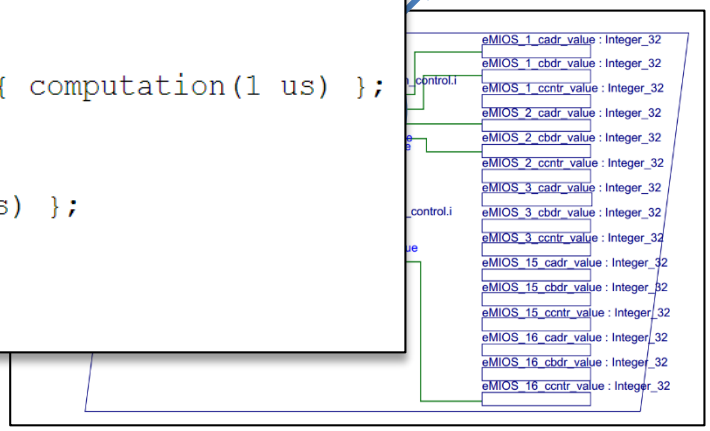



```
ANNEX Behavior_Specification {**
  STATES
    s : INITIAL COMPLETE FINAL STATE;
  TRANSITIONS
    t : s -[ ON DISPATCH ]-> s {
      lt_setpoint_speed_right := CP_3;
      lt_setpoint_speed_left := CP_5;
      get_time_value!(x,y,lt_time);
      CP_8 := lt_time;
      CP_9 := lt_time;
      driverSW_get_vs!(u,v,w,lt_regulation_speed_Right,lt_regulation_speed_Left);
      regulation_c_mc_pwm!(
        lt_regulation_speed_Right,lt_regulation_speed_Left,
                                                  motor_Left);
```



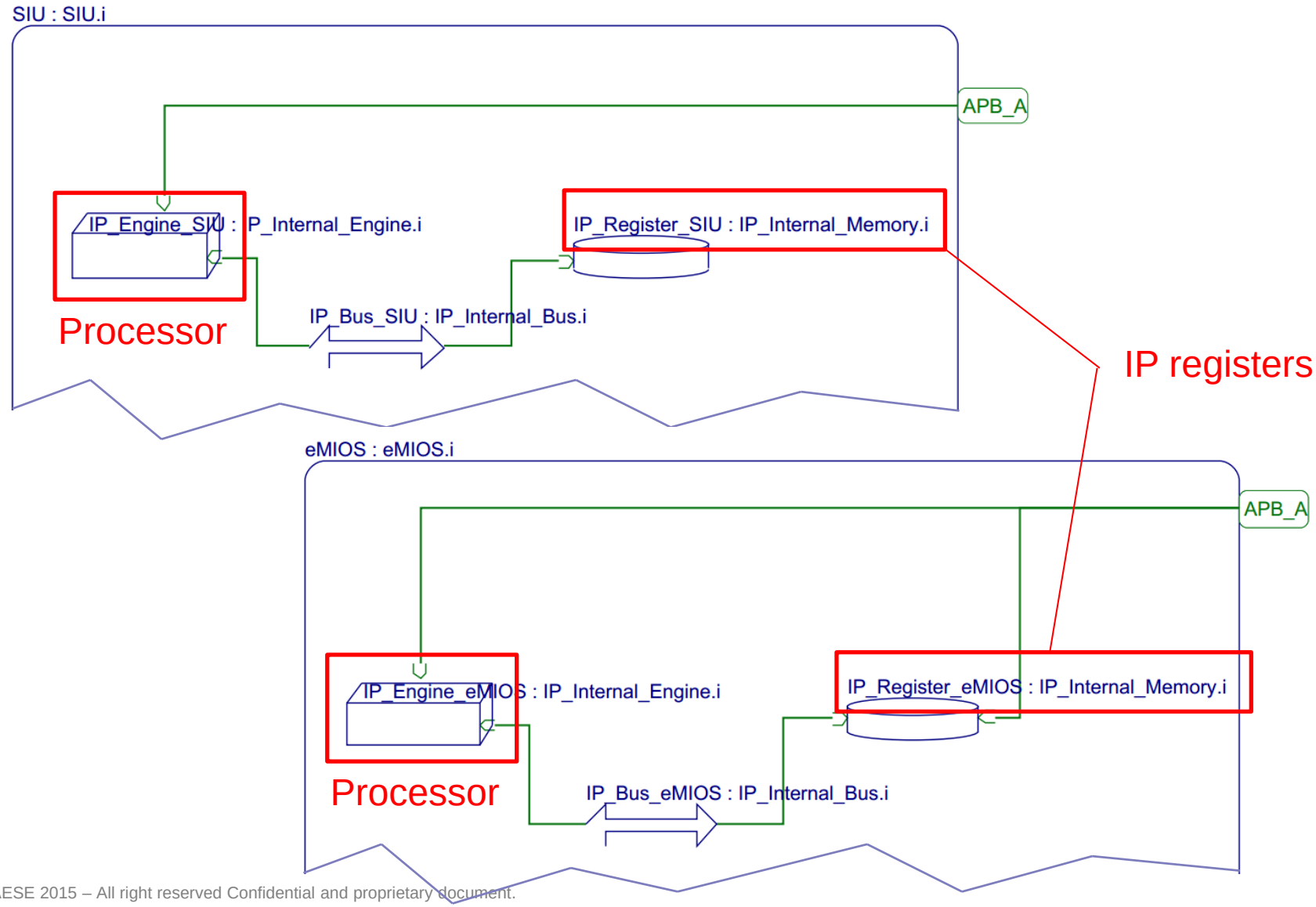

```
ANNEX Behavior_Specification {**
  VARIABLES
    d : Integer_32;
  STATES
    s : INITIAL FINAL STATE;
  TRANSITIONS
    t : s -[]-> s {
      eMIOS_1_cadr_value := set_eMIOS_1_cadr_value;
      eMIOS_o_1 := 0;
      for ( i : Integer in 0..eMIOS_1_cadr_value ) { computation(1 us) };
      eMIOS_o_1 := 0;
      d := eMIOS_1_cbdr_value - eMIOS_1_cadr_value;
      for ( i : Integer in 0..d ) { computation(1 us) };
      eMIOS_o_1 := 1 };
**};
END emios_o_1_control;
```





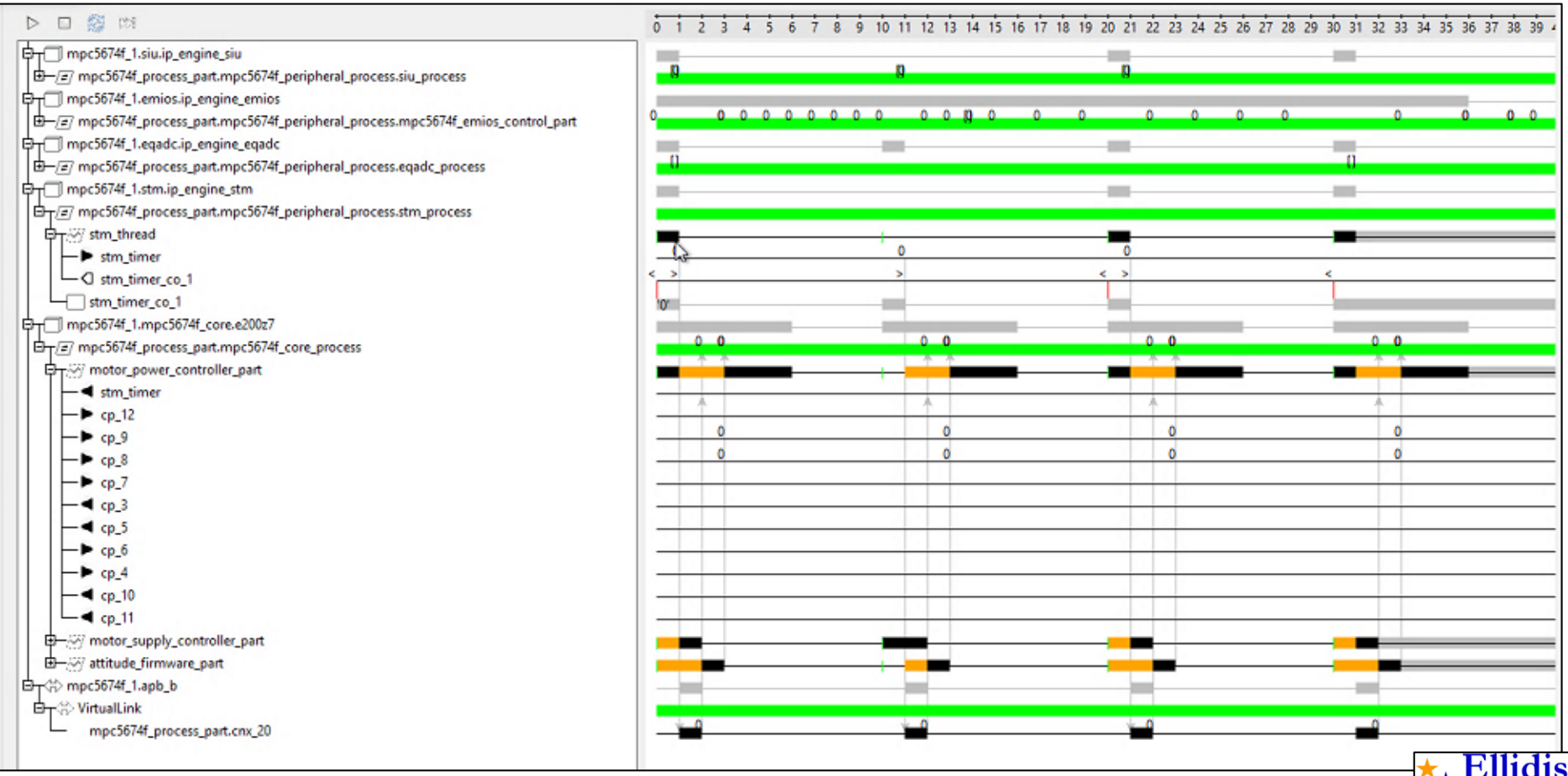
AADL modeling

The buck boost converter



AADL modeling

The buck boost converter – Simulation using Marzhin



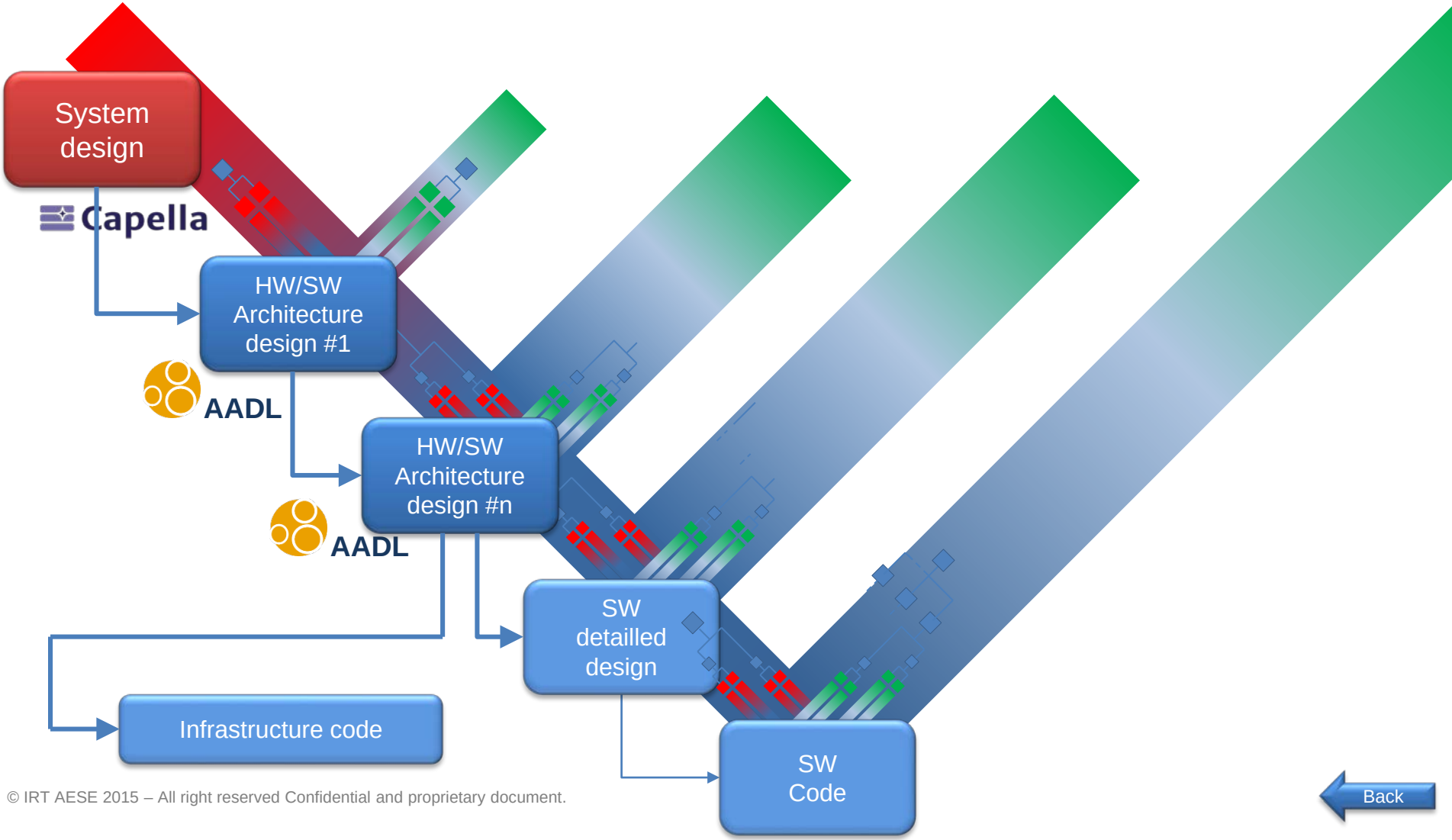
SW and HW design

Design Space Exploration

Back

From models to virtual platforms

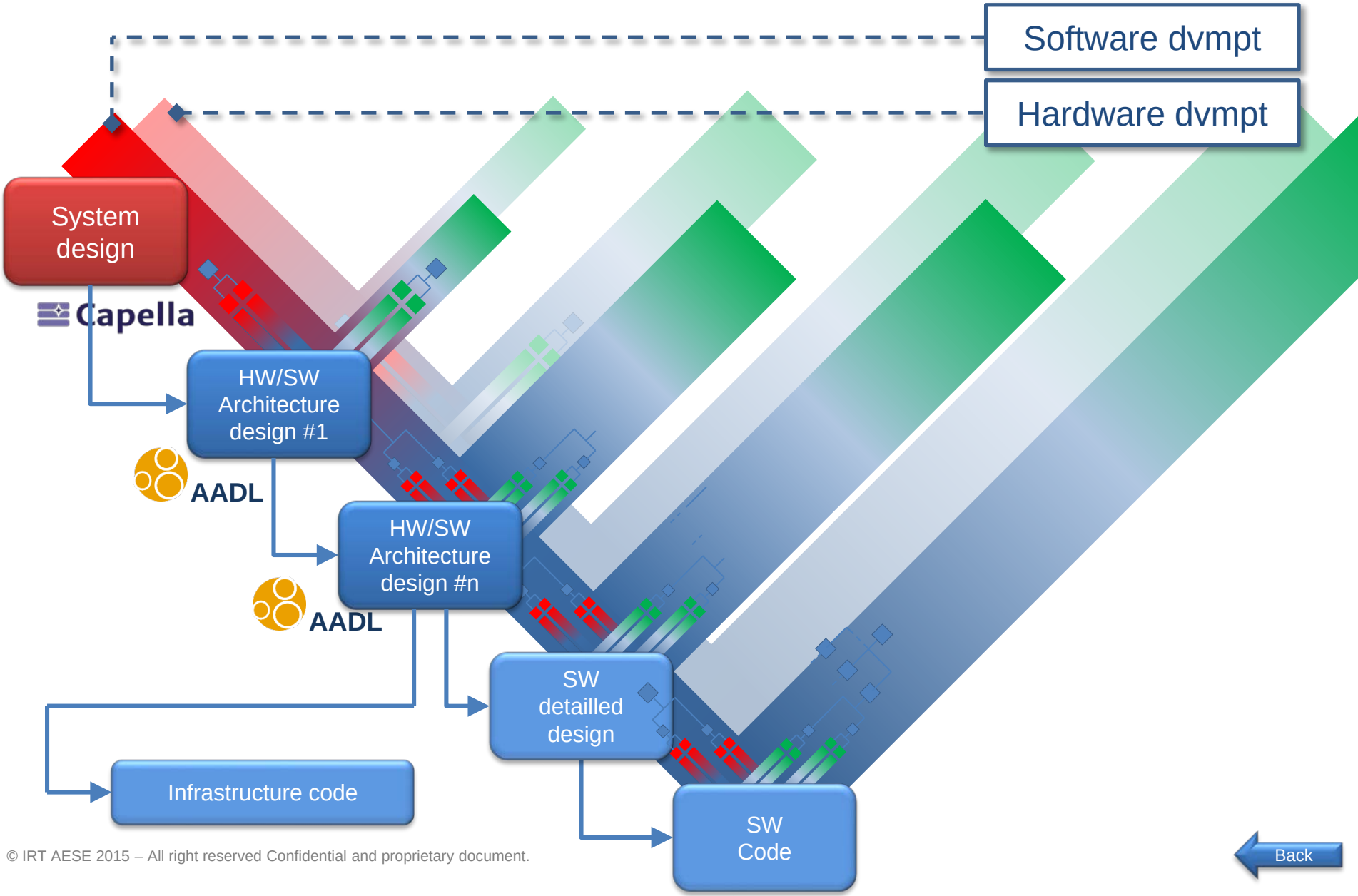
Overview



Back

From models to virtual platforms

Overview



Back

From models to virtual platforms

Overview

- Abstract (virtual) hardware model (simulation)
- Software (actual)

System design

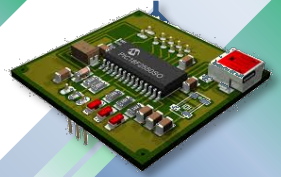
Capella

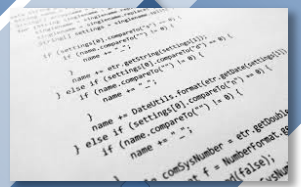
HW/SW Architecture design #1

AADL

HW/SW Architecture design #n

AADL





SW detailled design

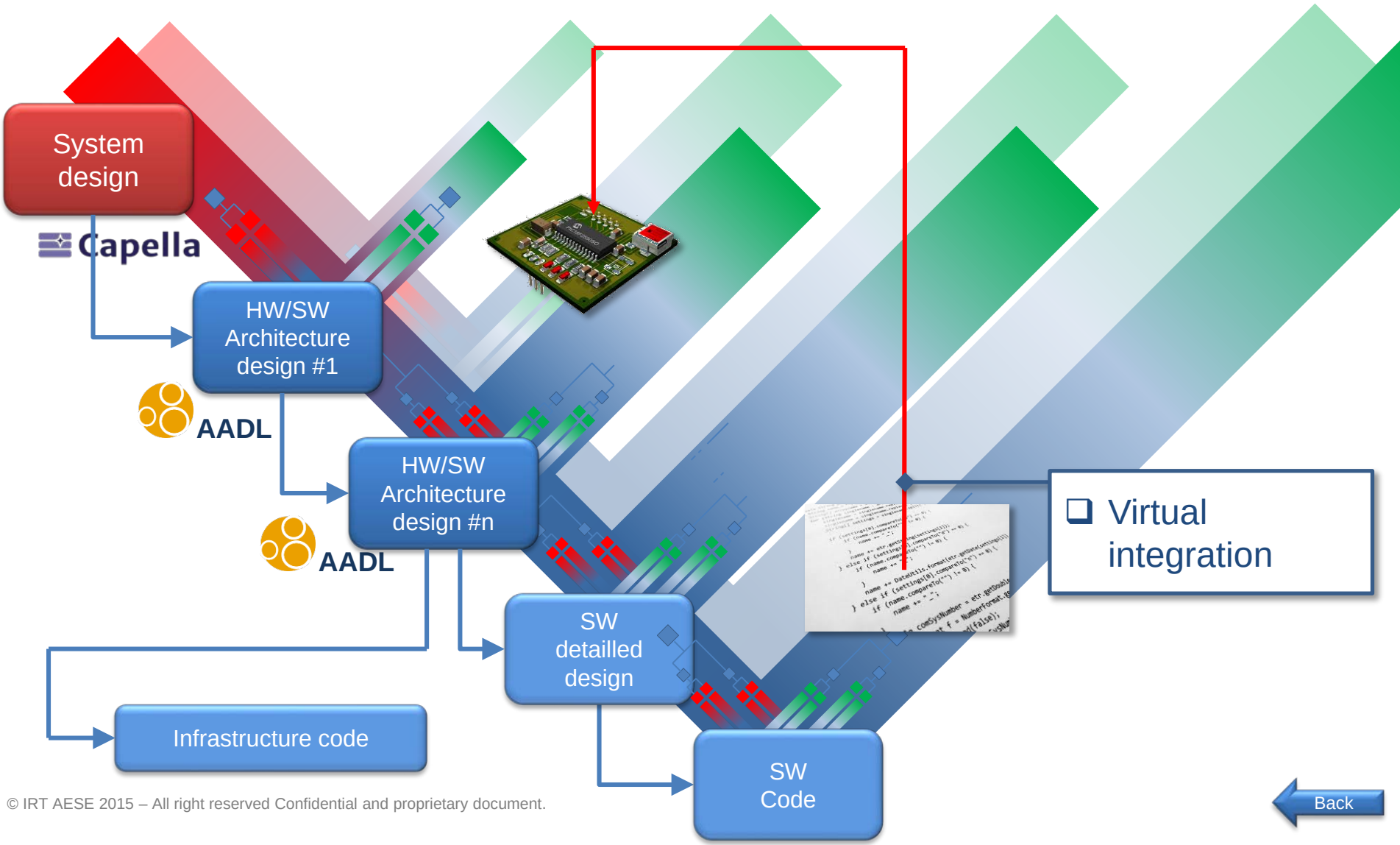
Infrastructure code

SW Code

Back

From models to virtual platforms

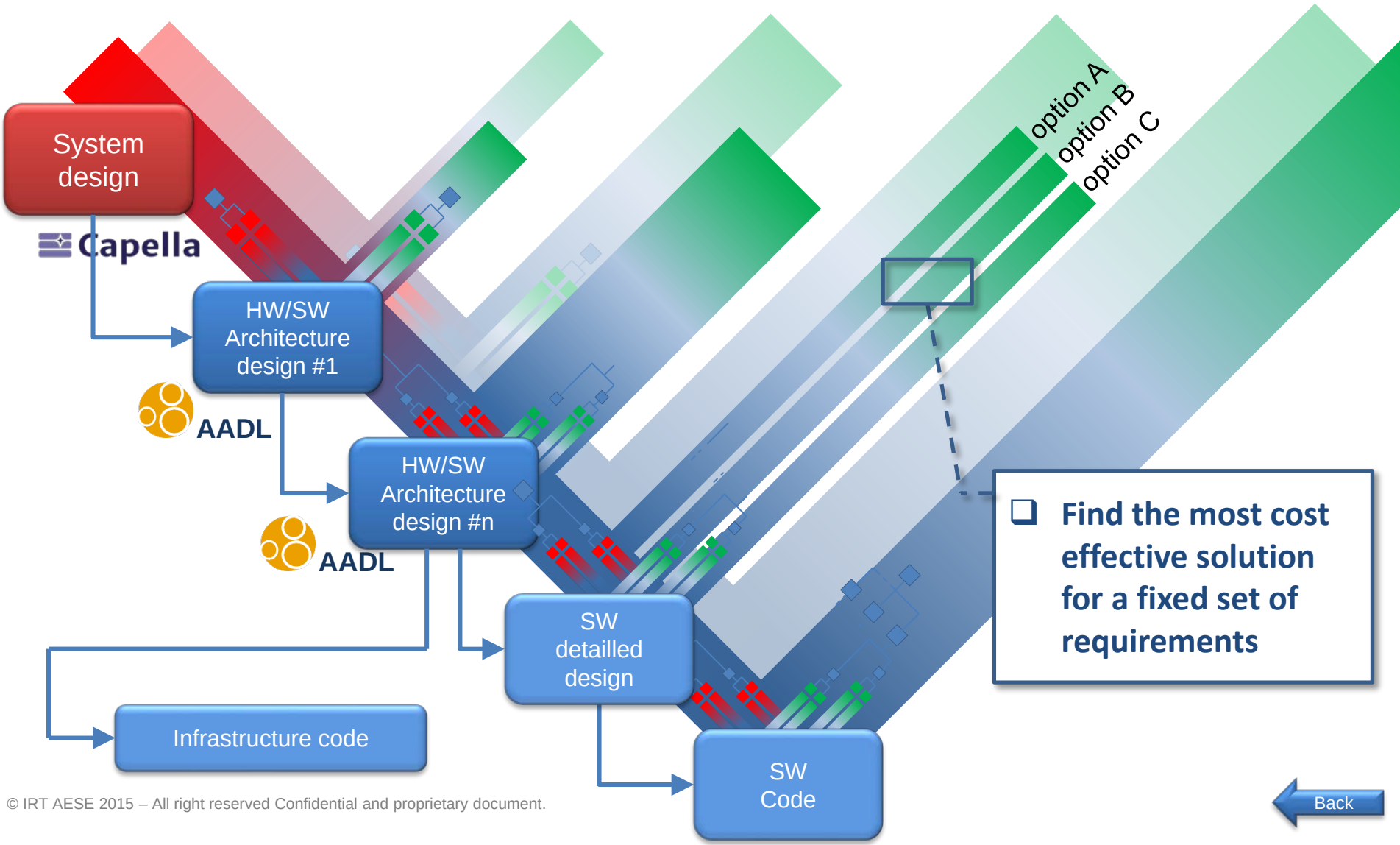
Overview



Back

Design space exploration

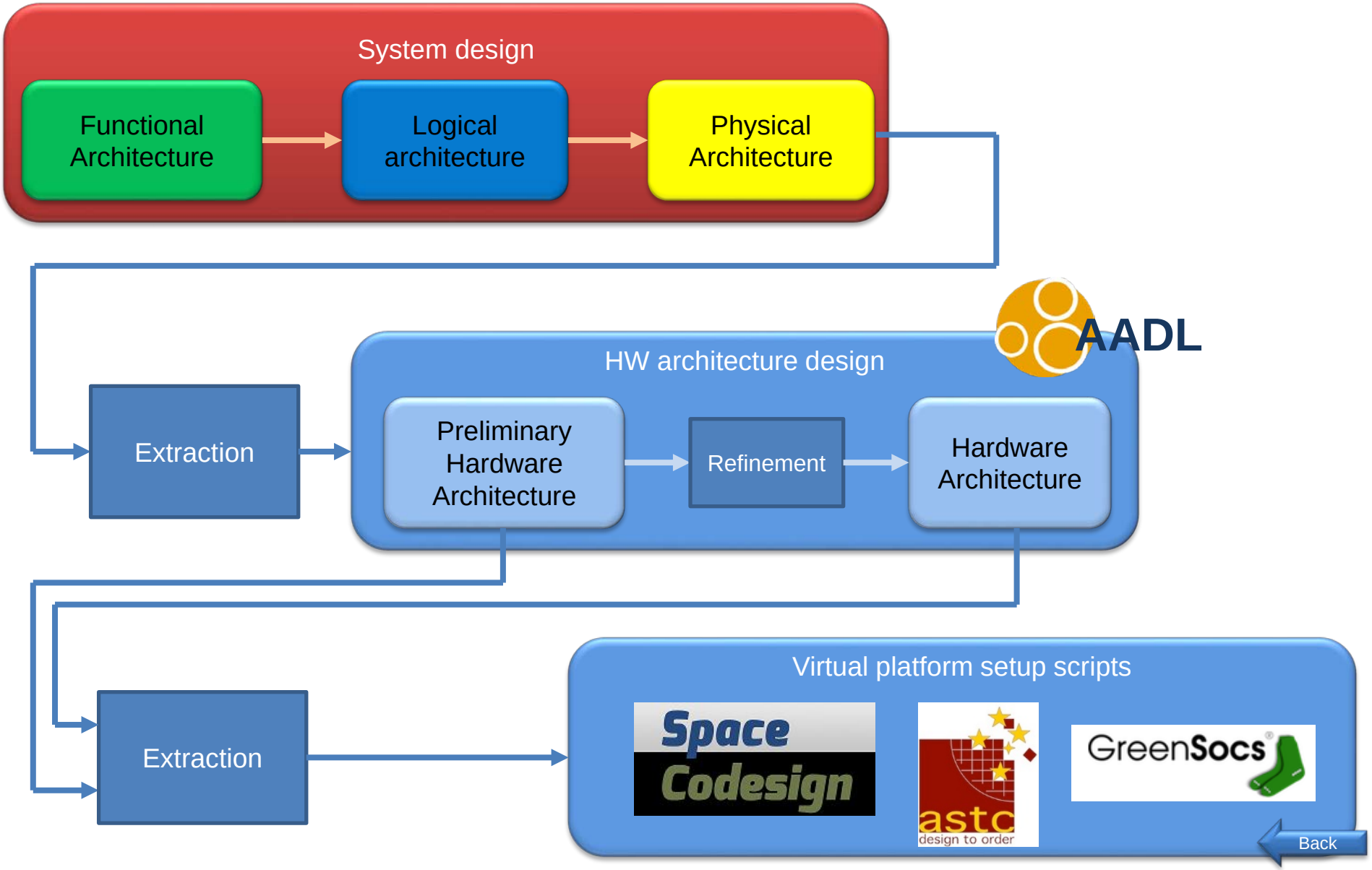
Experiment with SpaceStudio



Back

From models to virtual platforms

Overview



Our demonstrator

The buck-boost converter

Ethernet CAM

WiFi

ethernet

MISSION

ZYNQ
RMS
RAM
ARP
IBPM
MM
RCS
RM
CS
OS : LINUX

PPC
LOC
GM
MiM
PM
CS
OS: Trampoline

MON

COM

CAN bus

POWER

left and right wheel

Encoder

Motor

H-bridge

discrete

PPC
Power control
Motor control
MW: CAN driver
OS: Trampoline

TREK1000 (on-board)

I2C

Compas

analog

Buck analog part

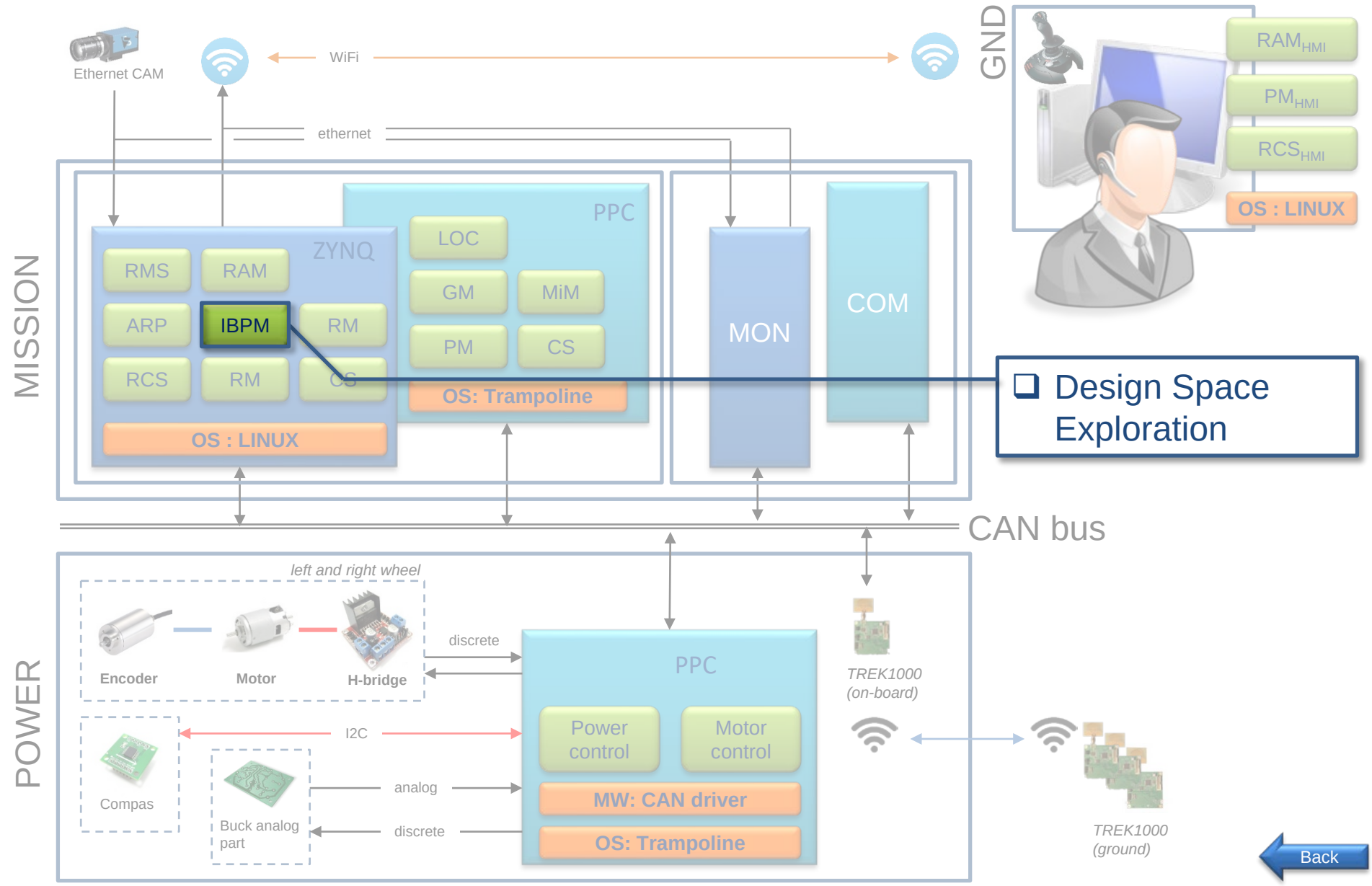
discrete

TREK1000 (ground)

Back

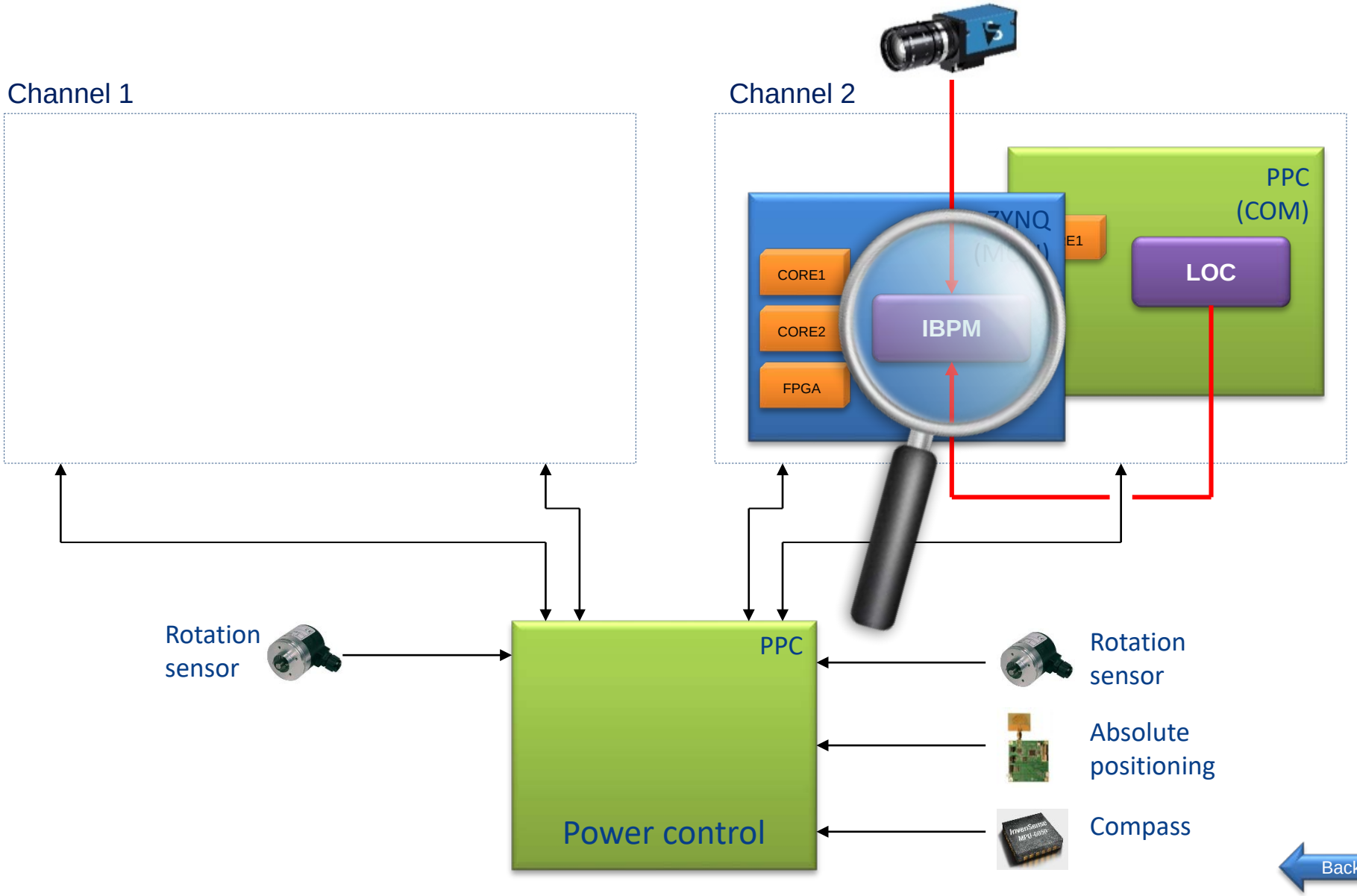
Experiment with SpaceStudio

The target



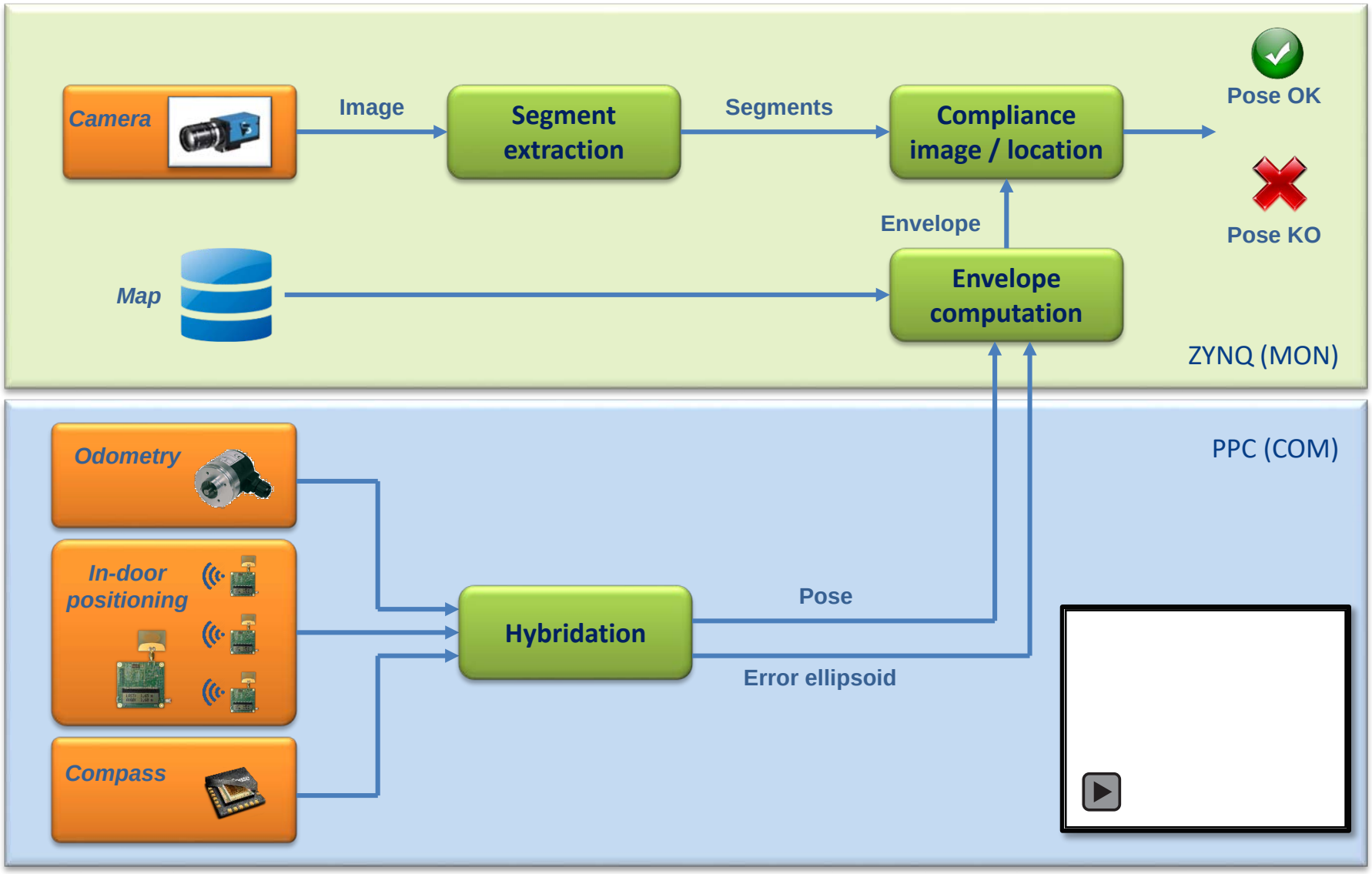
Design space exploration

Experiment with SpaceStudio



Design space exploration

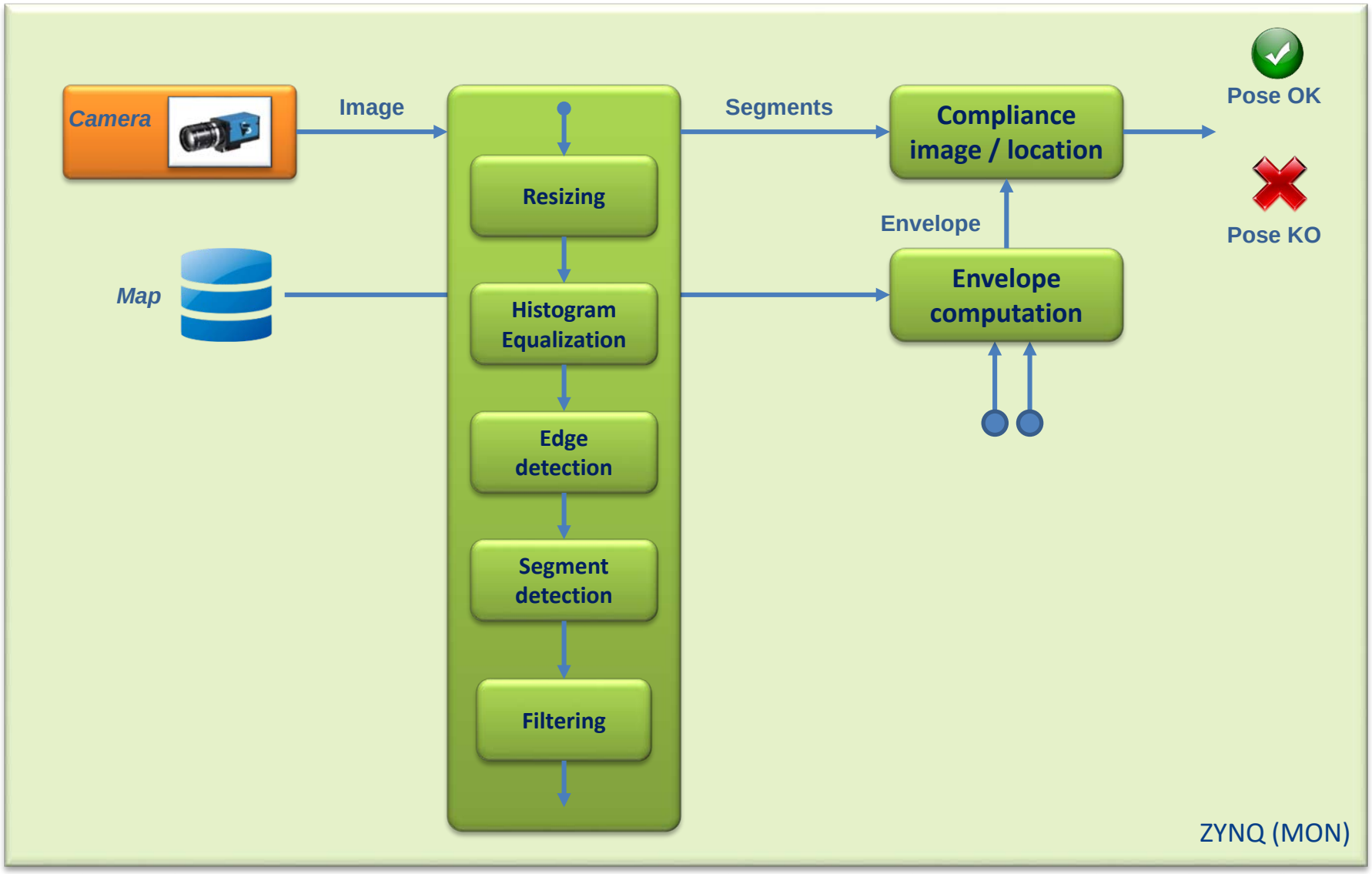
Experiment with SpaceStudio



Back

Design space exploration

Experiment with SpaceStudio



Back

Design space exploration

Experiment with SpaceStudio

Requirements

- of 5 images / second
- Less than 33% of the hard core processing resources
- Less than 50% of the DDR memory
- Less than 80% of the FPGA resource
- Less than 60% of PS/PL bus bandwidth

Architectural DoF

Type of core (hard, soft)

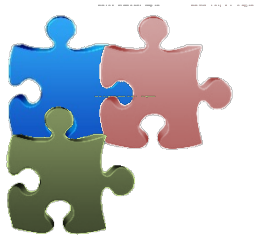
Implementation {HW, SW}

Number of threads

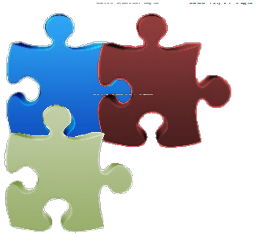
etc.

Solutions

Solution #1



Solution #2



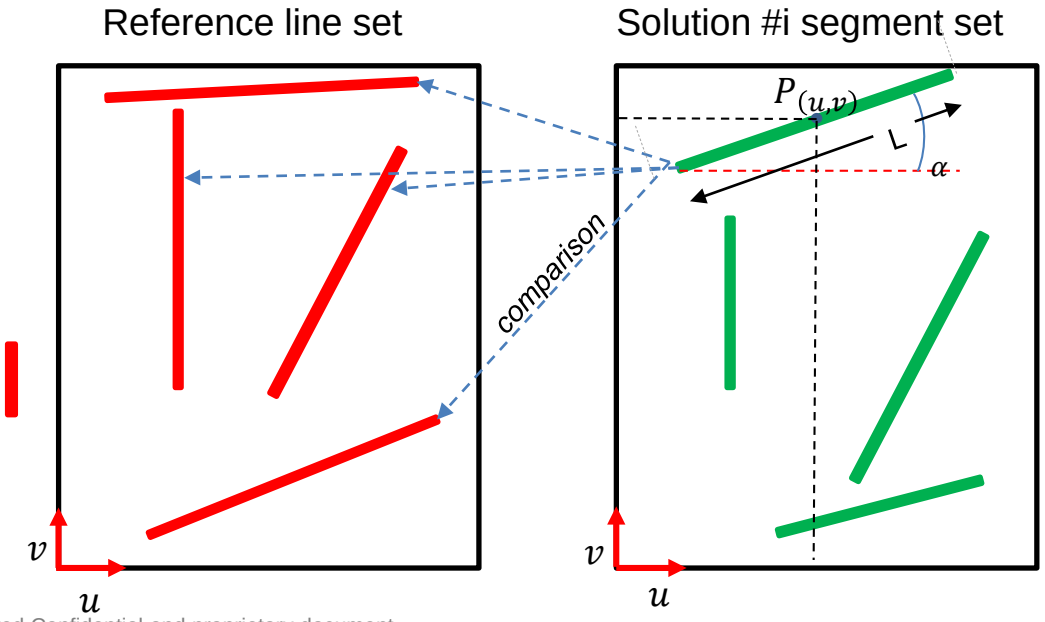
Back

Design exploration

Definition of QoS metrics

All solutions are compared to a reference (*ideal*) solution

- The segment set corresponding to the part of the tracks “seen” by the camera
- We know the position of the rover at a given date
- We know the map
- We know the parameters of the camera
- We compute the set of segments which are visible by the camera in the image space



Back

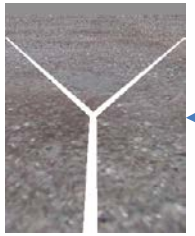
Definition of QoS metrics

Impact of processing time

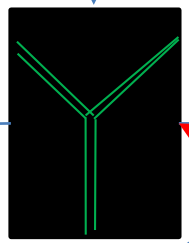
Image sent at date T_{start}

Image processed at date T_{end}

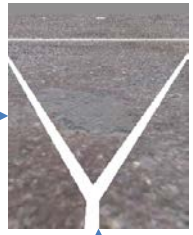
Image the robot see at T_{end}



Is the result of

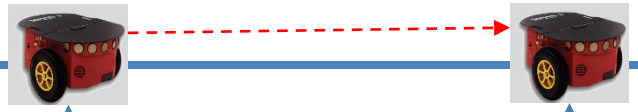


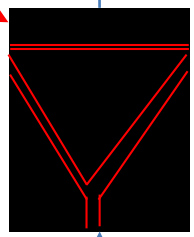
Is not the result of



To be compared

Is calculated from





Position at T_{start}

Position at T_{end}

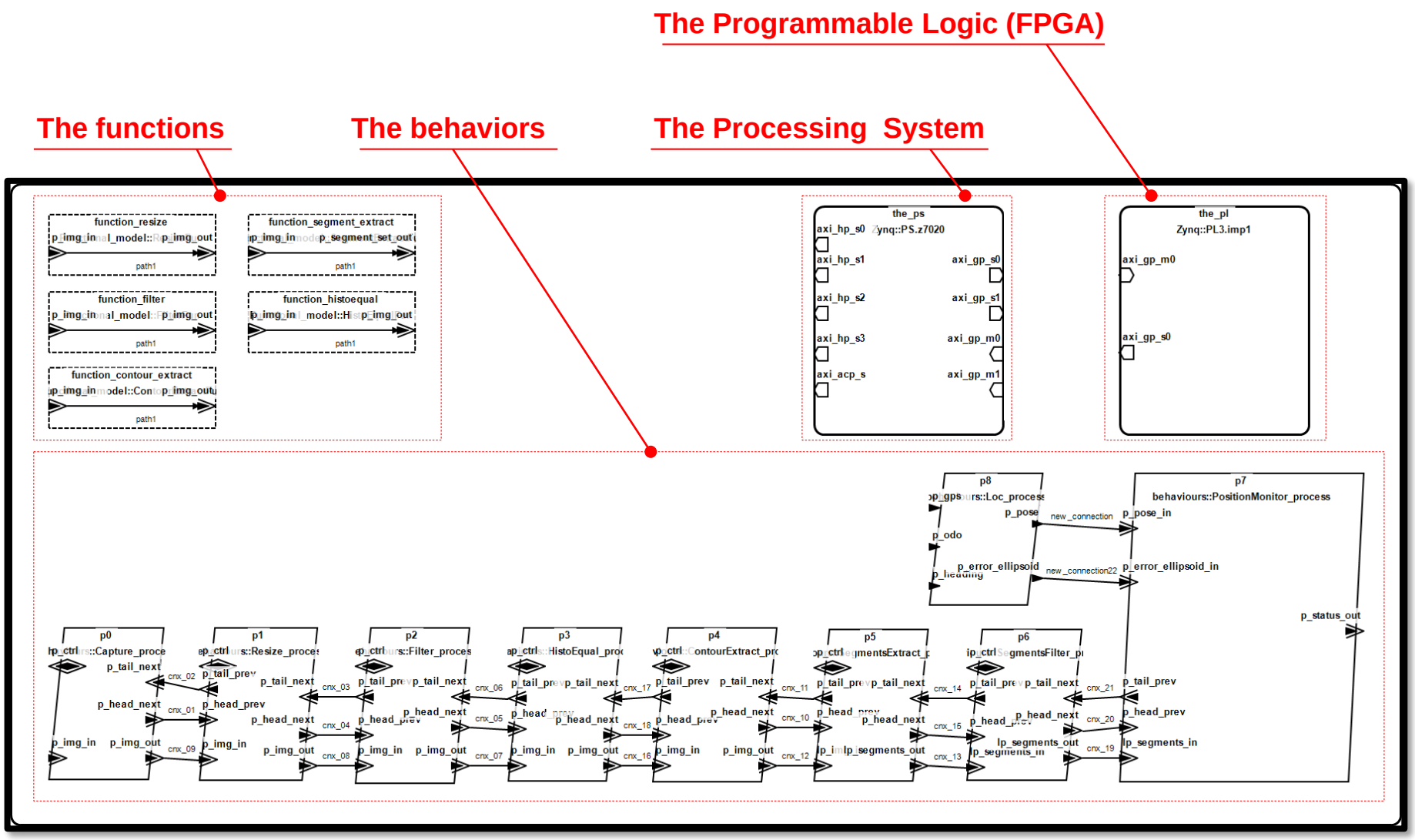
Reference: what we want to get

$$P_{[T_{start}]} = \begin{pmatrix} X_{[T_{start}]} \\ Y_{[T_{start}]} \\ Z_{[T_{start}]} \\ \theta_{[T_{start}]} \end{pmatrix} \qquad P_{[T_{end}]} = \begin{pmatrix} X_{[T_{end}]} \\ Y_{[T_{end}]} \\ Z_{[T_{end}]} \\ \theta_{[T_{end}]} \end{pmatrix}$$

Orientation (rotation angle around Z axis)

Back

The design in AADL

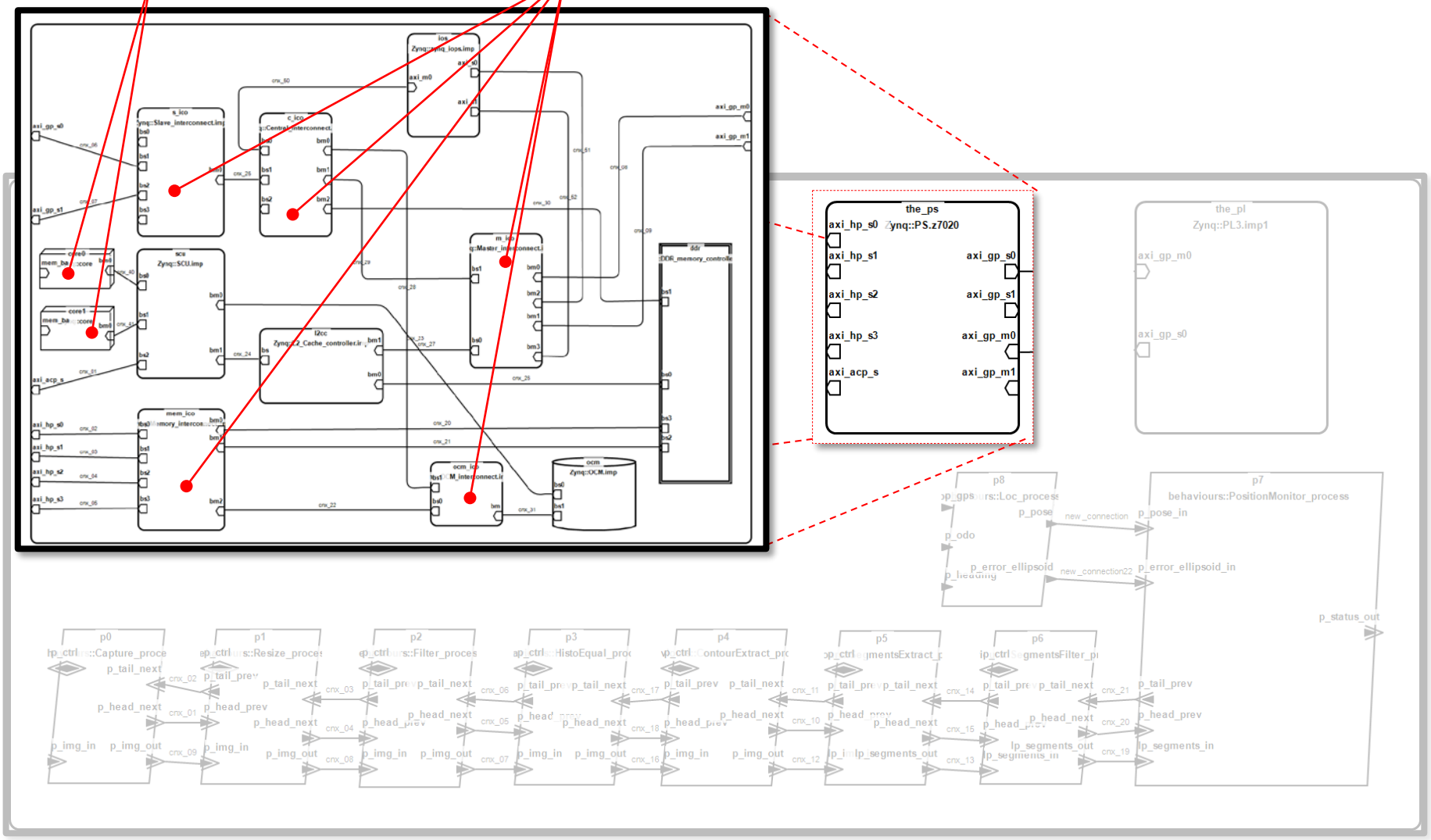


Back

The Zynq board

The cores

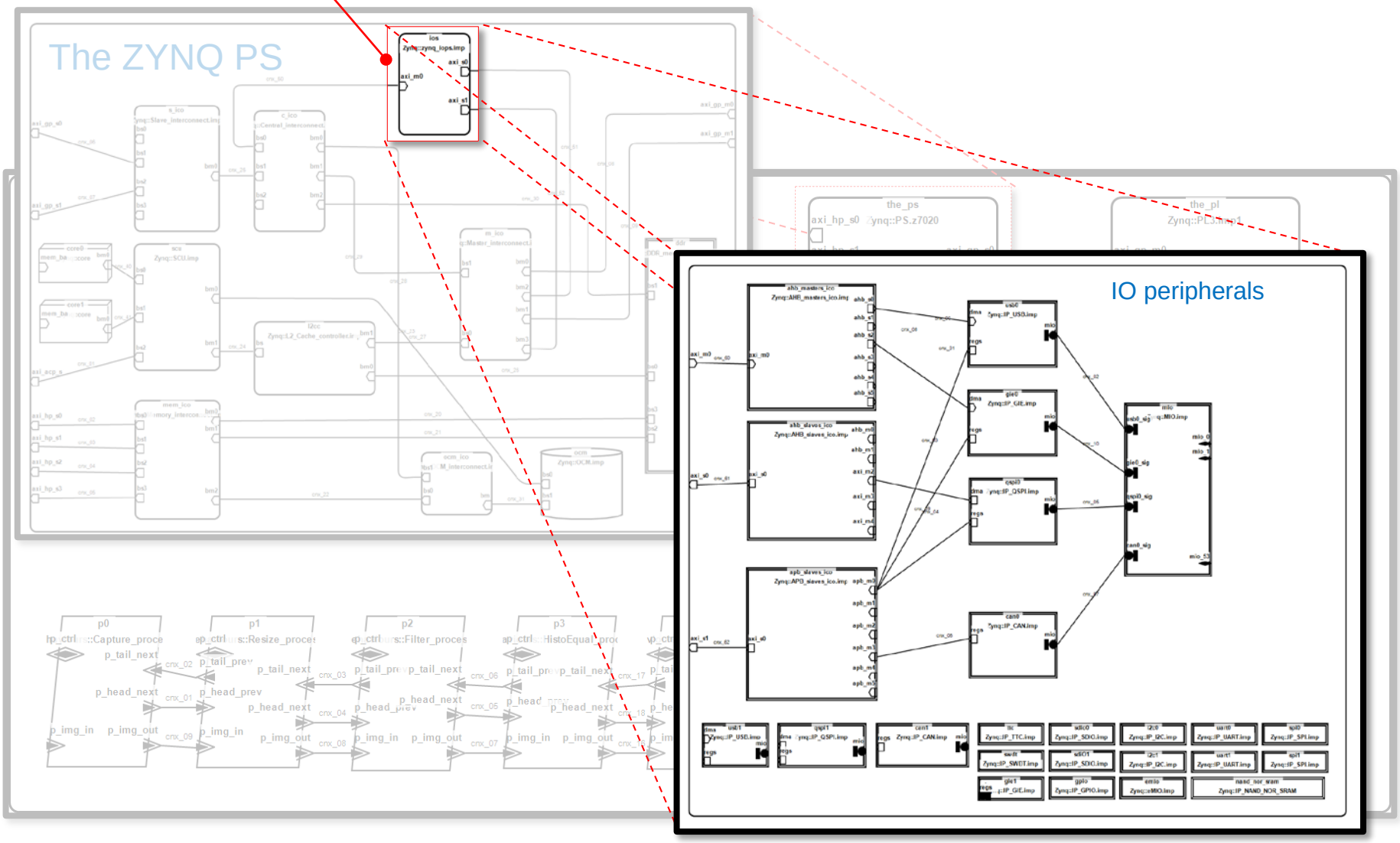
The interconnect



Back

The Zynq board

The IO peripherals

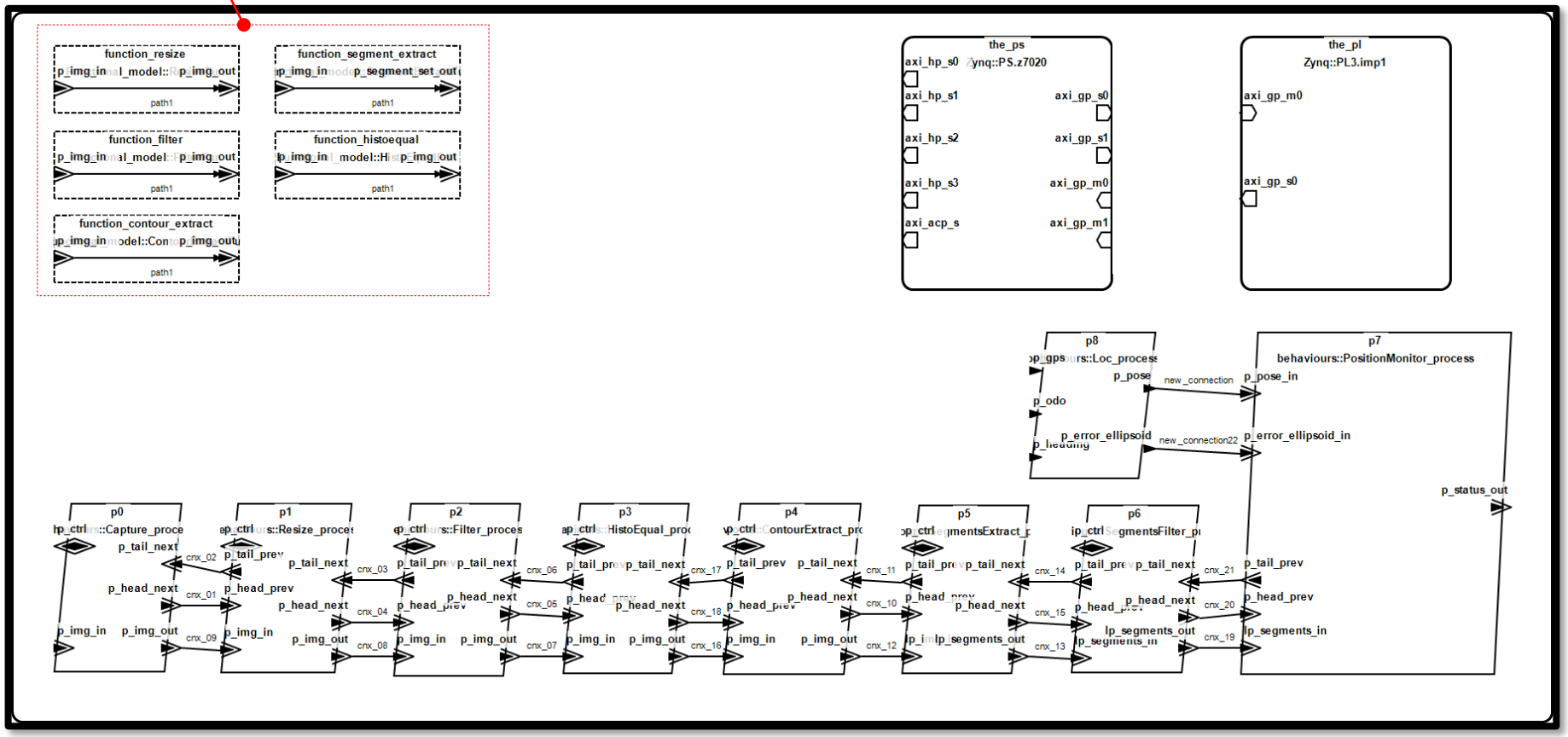


Back

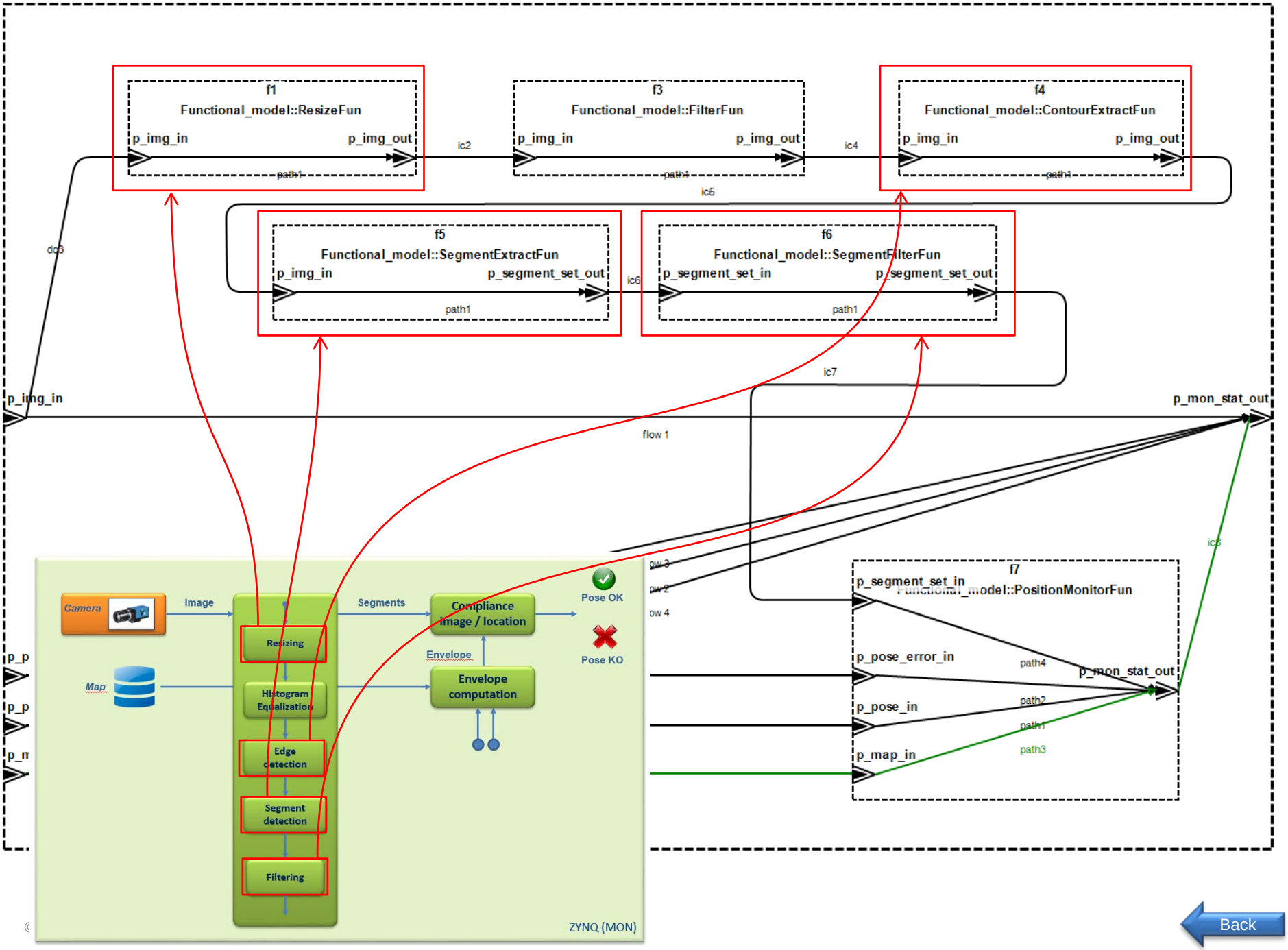
The design in AADL

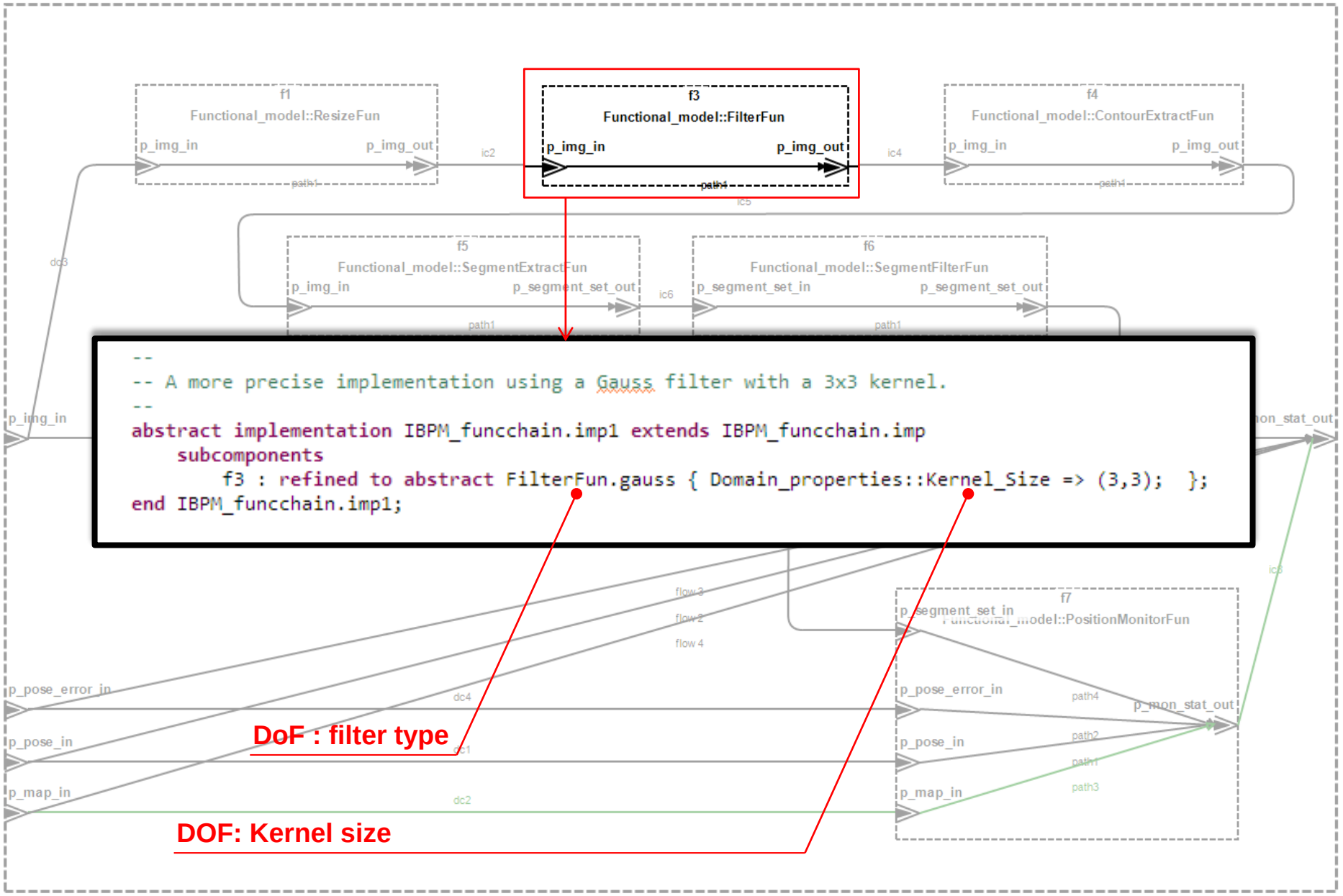
The functions

The functions

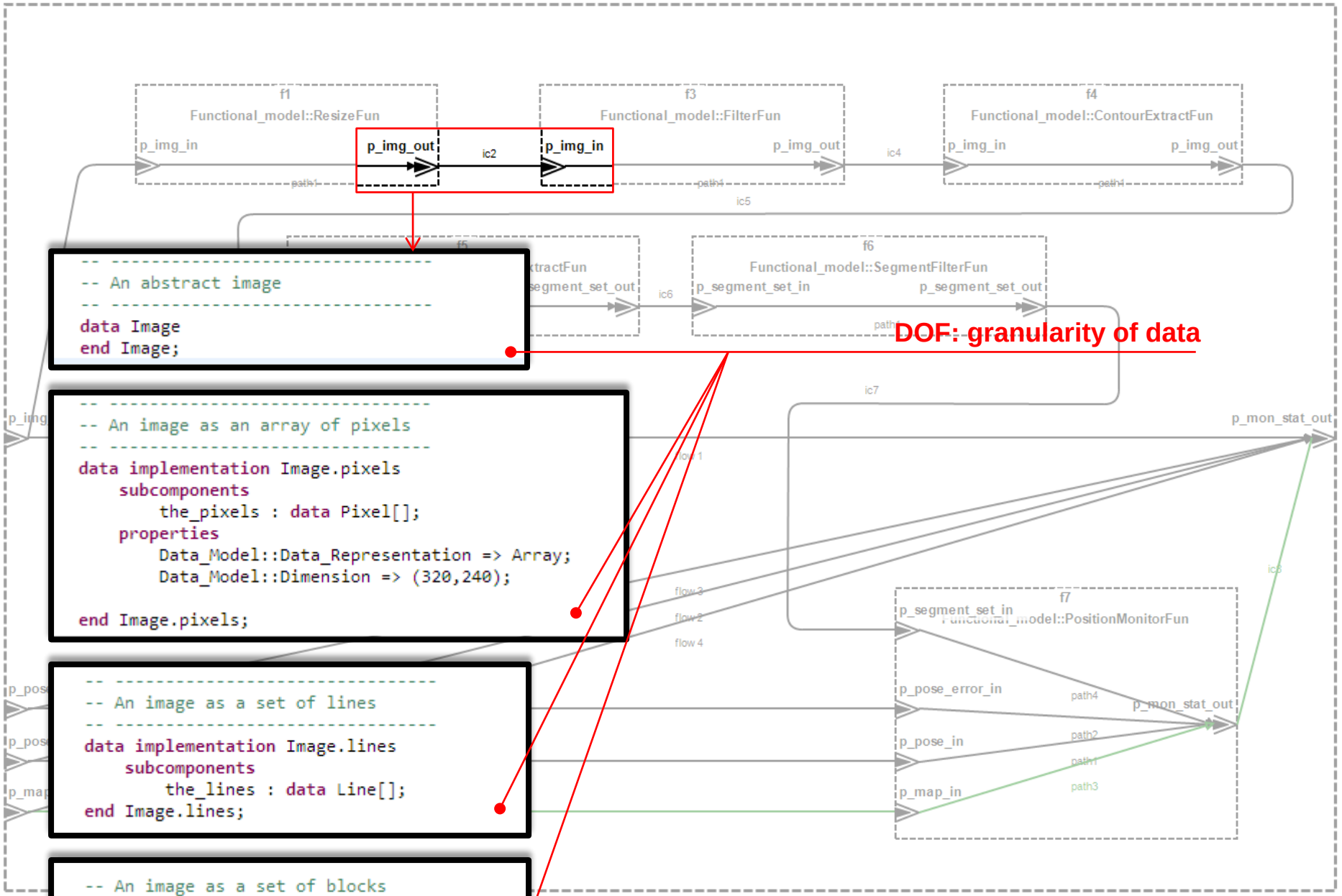


Back


f1
Functional_model::ResizeFun
p_img_in
p_img_out
path1
ic2
f3
Functional_model::FilterFun
p_img_in
p_img_out
path1
ic4
f4
Functional_model::ContourExtractFun
p_img_in
p_img_out
path1
ic5
f5
Functional_model::SegmentExtractFun
p_img_in
p_segment_set_out
path1
ic6
f6
Functional_model::SegmentFilterFun
p_segment_set_in
p_segment_set_out
path1
ic7
dc8
p_img_in
p_mon_stat_out
flow 1
Camera
Image
Map
Resizing
Histogram Equalization
Edge detection
Segment detection
Filtering
Segments
Compliance image / location
Envelope
Envelope computation
Pose OK
Pose KO
ZYNQ (MON)
ow 3
ow 2
ow 4
ic8
f7
p_segment_set_in
p_pose_error_in
p_pose_in
p_map_in
path4
path2
path1
path3
p_mon_stat_out
Back



Back



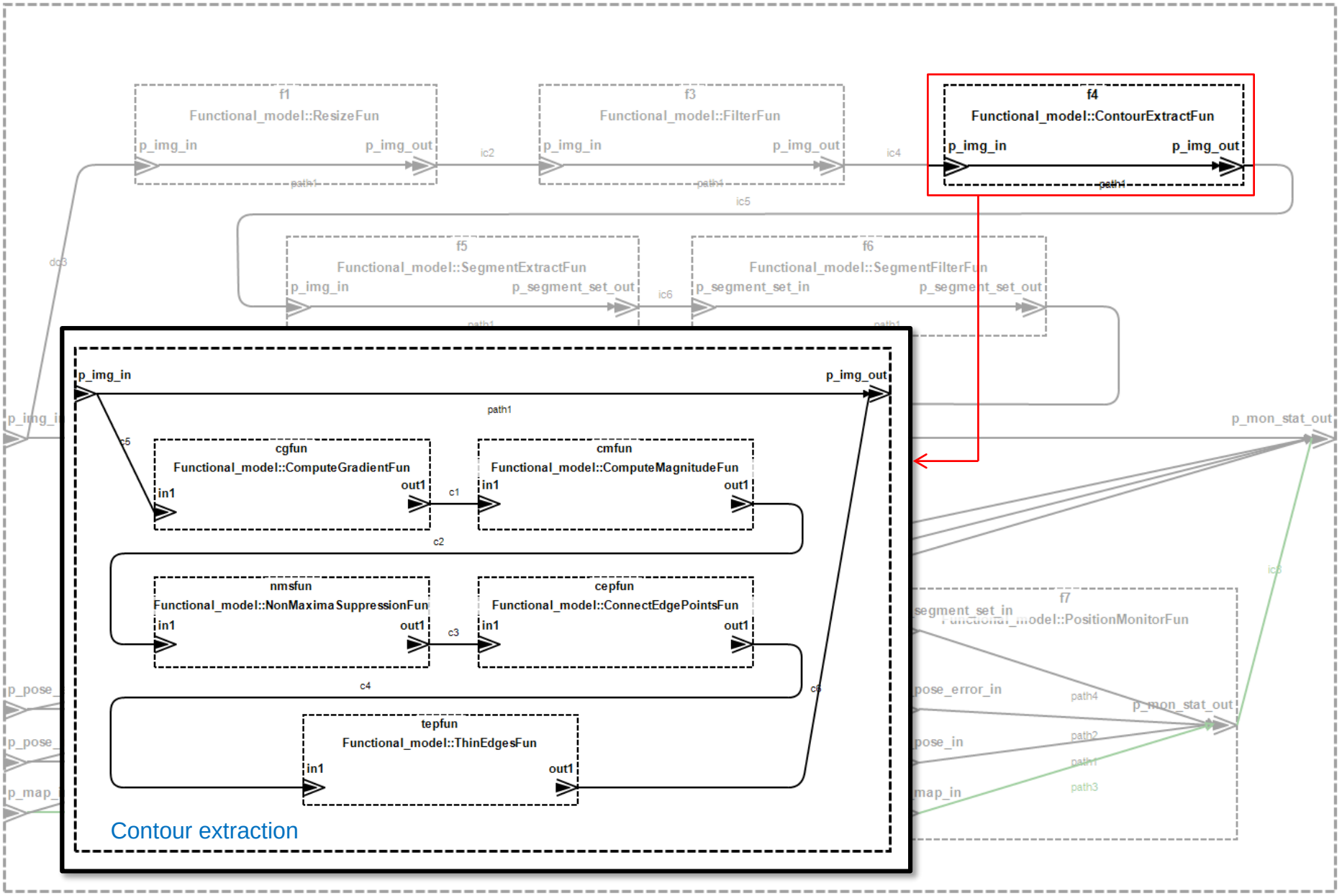

```
-- ---------------------------------
-- An abstract image
-- ---------------------------------
data Image
end Image;
```

```
-- ---------------------------------
-- An image as an array of pixels
-- ---------------------------------
data implementation Image.pixels
    subcomponents
        the_pixels : data Pixel[];
    properties
        Data_Model::Data_Representation => Array;
        Data_Model::Dimension => (320,240);

end Image.pixels;
```

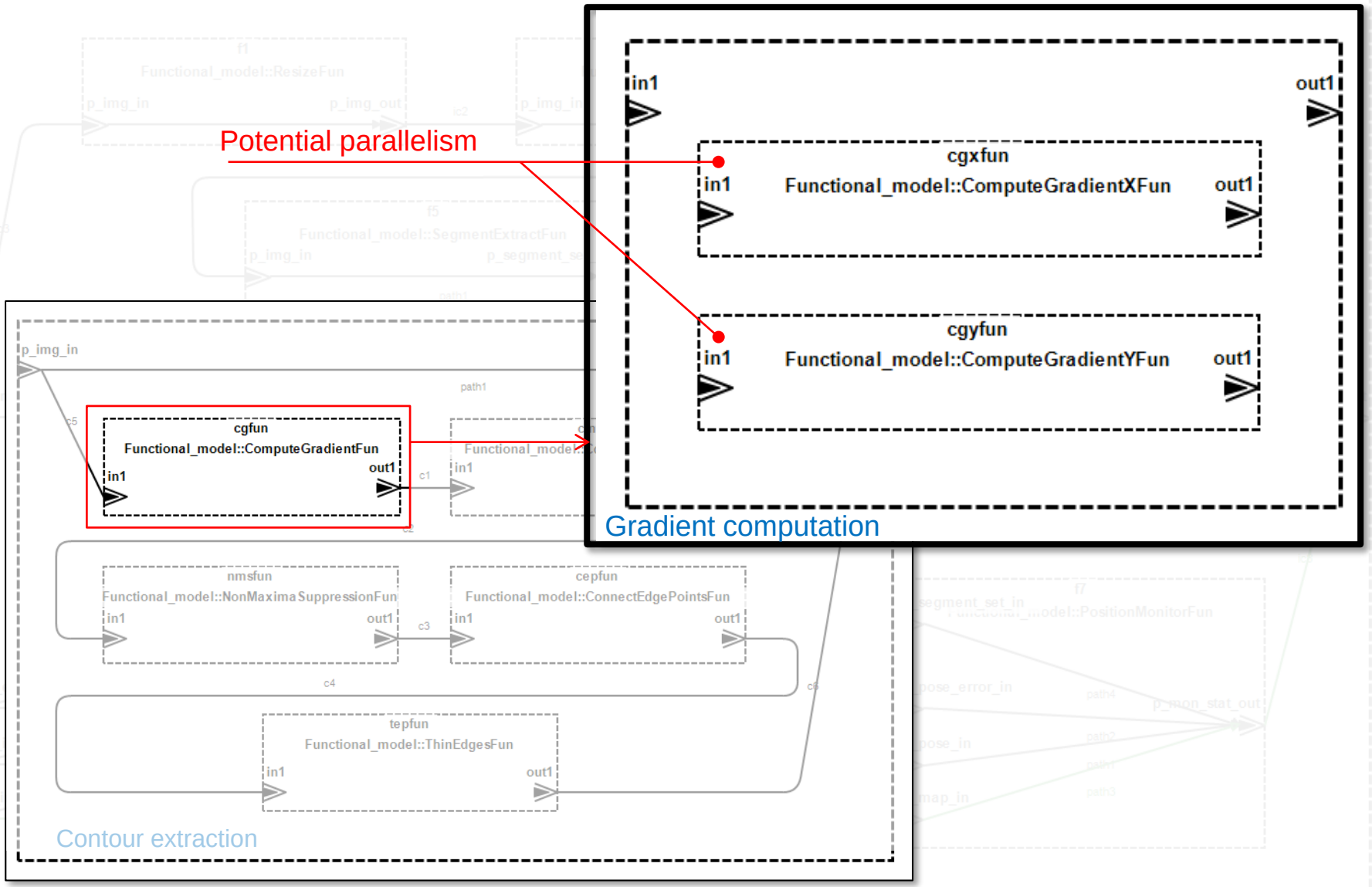
```
-- ---------------------------------
-- An image as a set of lines
-- ---------------------------------
data implementation Image.lines
    subcomponents
        the_lines : data Line[];
end Image.lines;
```

```
-- An image as a set of blocks
data implementation Image.blocks
    subcomponents
        the_blocks : data Block[];
end Image.blocks;
```


Back

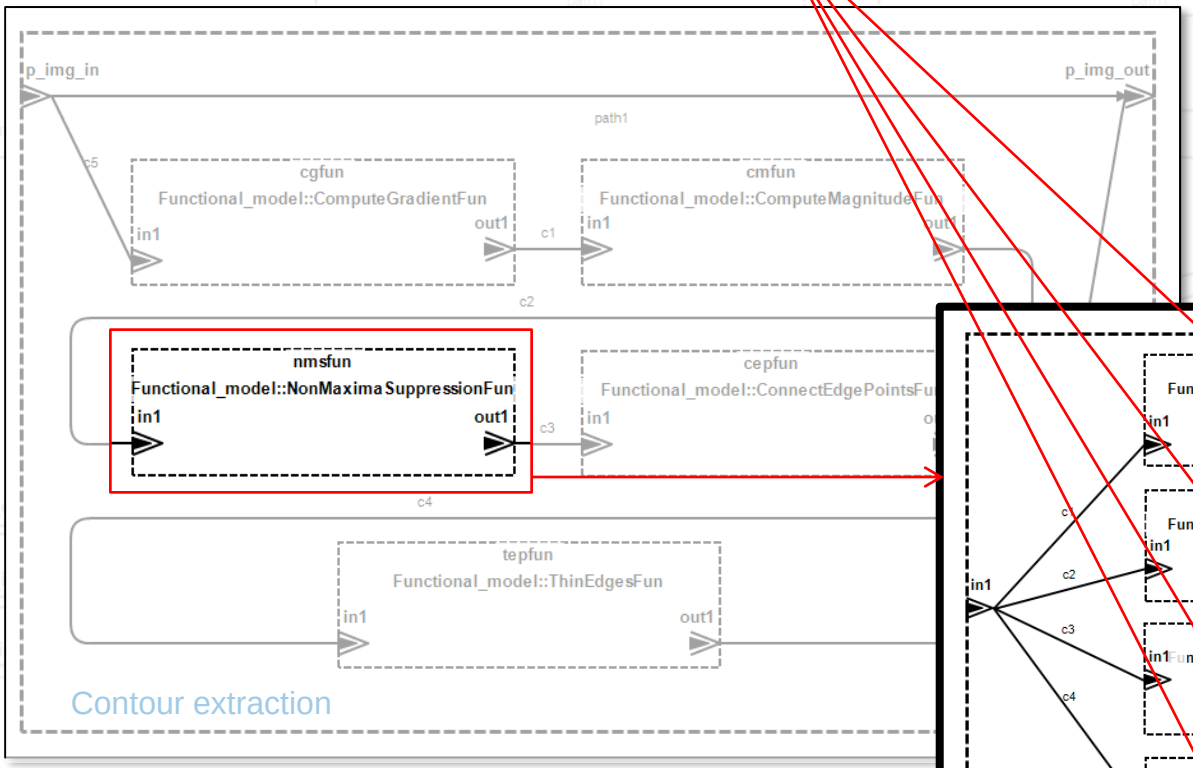


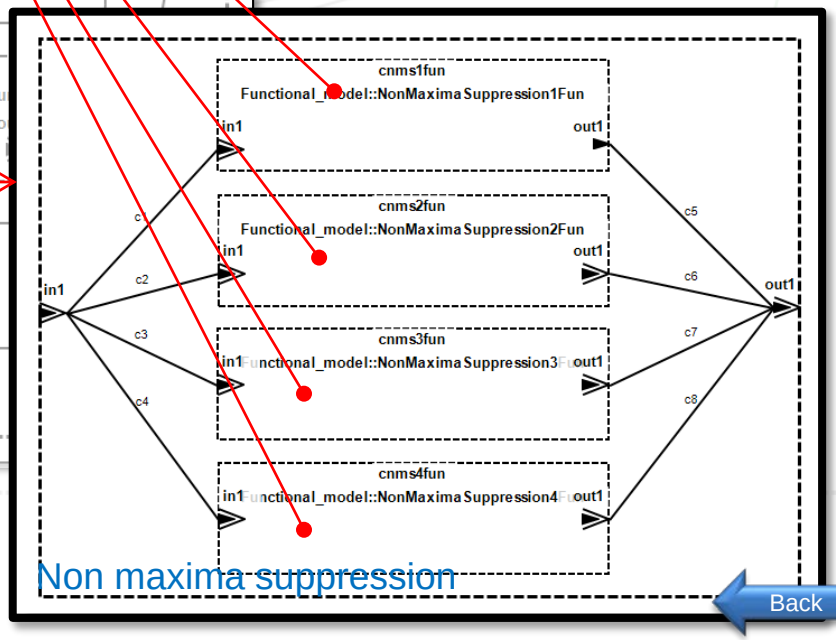
Back



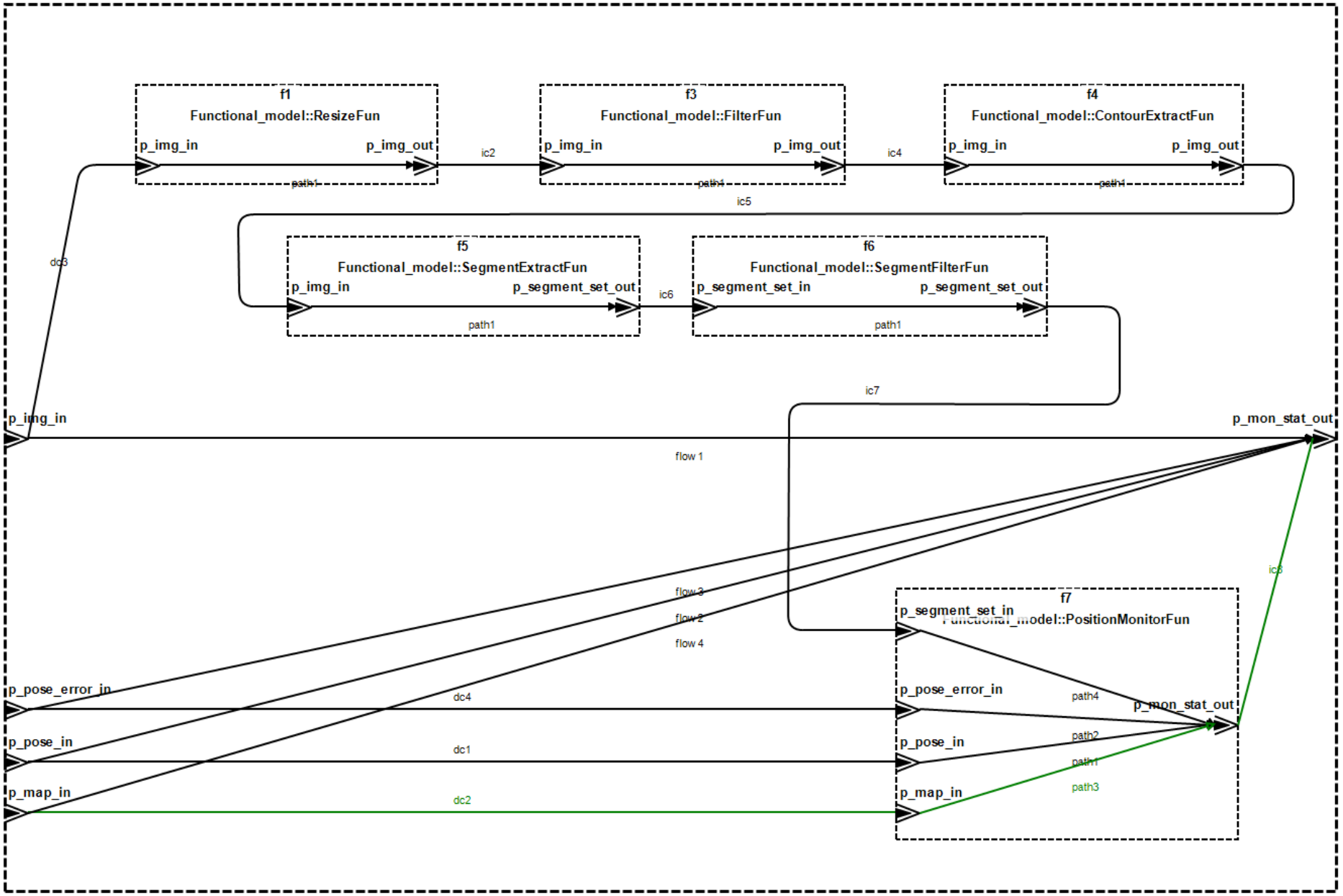
Back

Potential parallelism

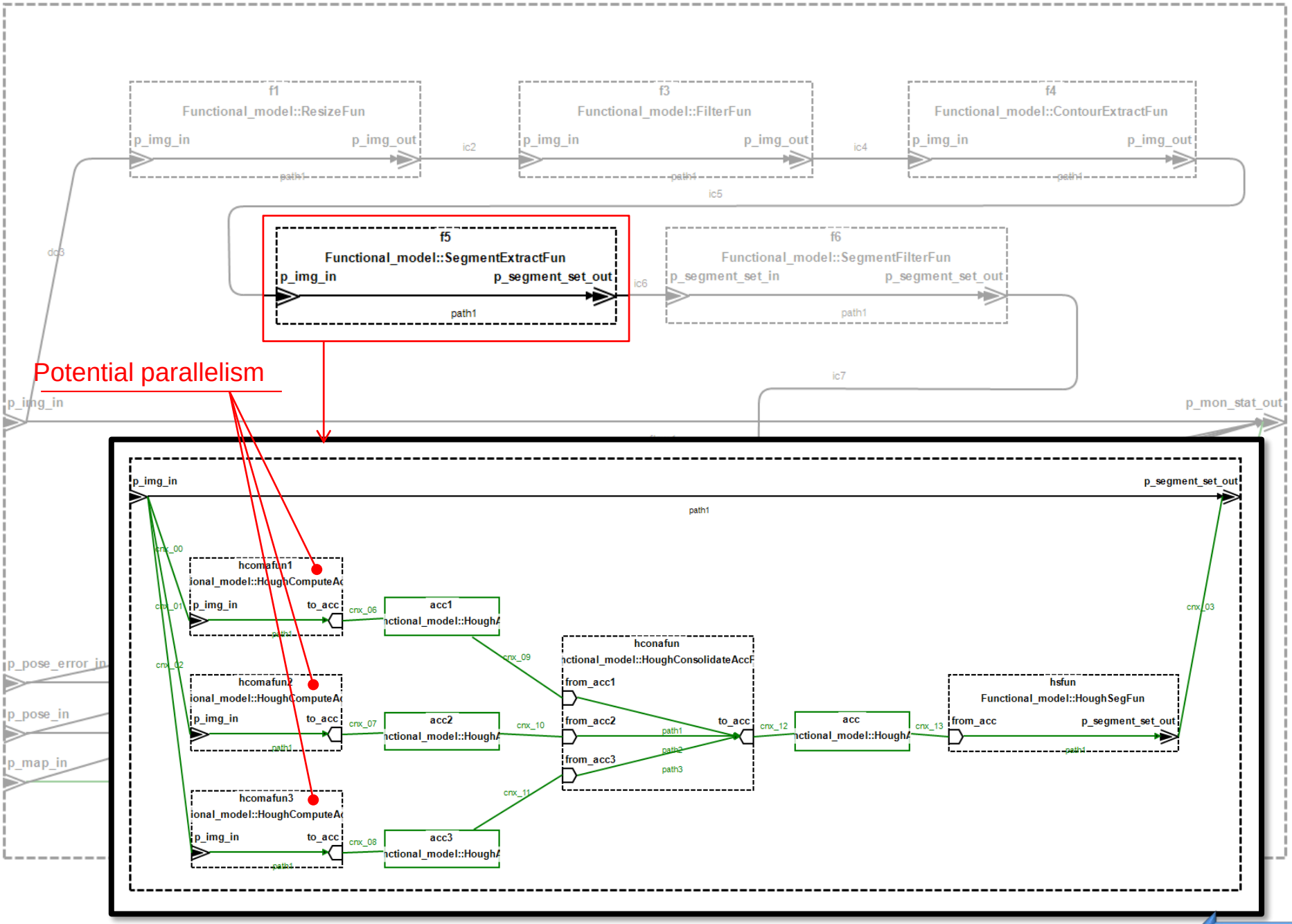




Back



Back



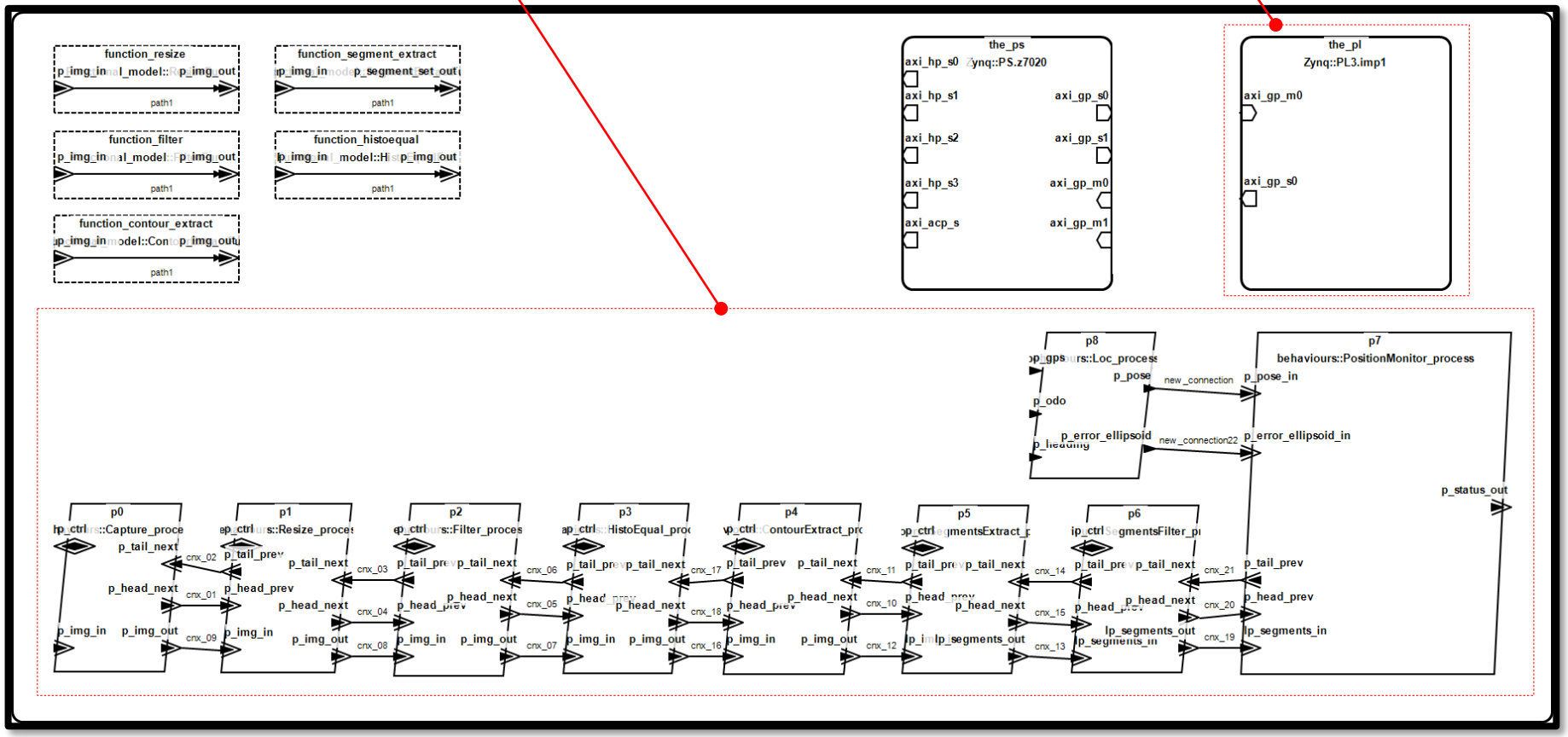
Back

AADL for Design Space exploration

Implementation solutions

The Programmable Logic (FPGA)

The behaviors



Back

AADL for Design Space exploration

Implementation solutions

```
1   thread implementation IBPM_th0.imp1
2       subcomponents
3           cap_img: data ImageQueue.array;
4           res_img: data ImageQueue.array;
5           fil_img: data ImageQueue.array;
6           con_img: data ImageQueue.array;
7           seg_seg: data SegmentSetQueue.array;
8       calls
9           seq1: {
10                  cap_sp: subprogram CaptureFromUdp.morse;
11                  rez_sp: subprogram Resize.openCV;
12                  fil_sp: subprogram Gauss.openCV;
13                  con_sp: subprogram Canny.OpenCv;
14                  seg_sp: subprogram Hough.OpenCv;
15                  sfi_sp: subprogram SegmentFiltering.hc;
16                  pmo_sp: subprogram PositionMonitoring.hc;
17                  };
18      connections
19      [...]
20  end IBPM_th0.imp1;
```

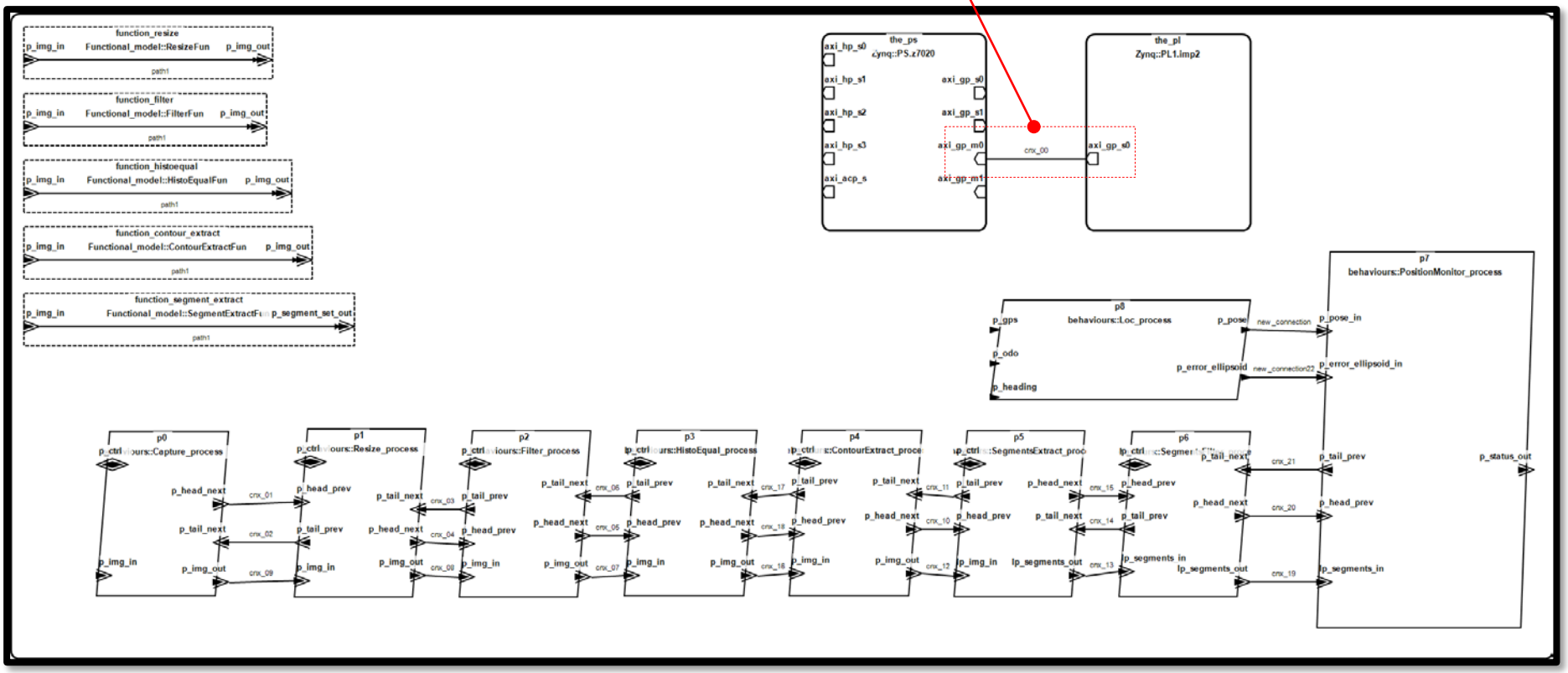
```
23  thread implementation IBPM_th0.imp2 extends IBPM_th0.imp1
24  calls
25          c1: {
26                  cap_sp: subprogram Capture.file;
27                  res_sp: subprogram Resize.hc;
28                  fil_sp: subprogram Gauss.hc;
29                  con_sp: subprogram Canny.hc;
30                  seg_sp: subprogram Hough.hc;
31                  sfi_sp: subprogram SegmentFiltering.hc;
32                  pmo_sp: subprogram PositionMonitoring.hc;
33                  };
34  end IBPM_th0.imp2;
```


Back

Design alternatives

Alternative #1: *2 IPs, buffers in BRAM*

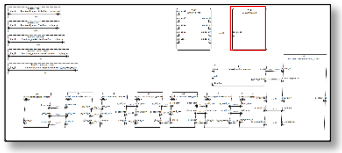
IPs are slaves



Back

Design alternatives

Alternative #1: *2 IPs, buffers in BRAM*

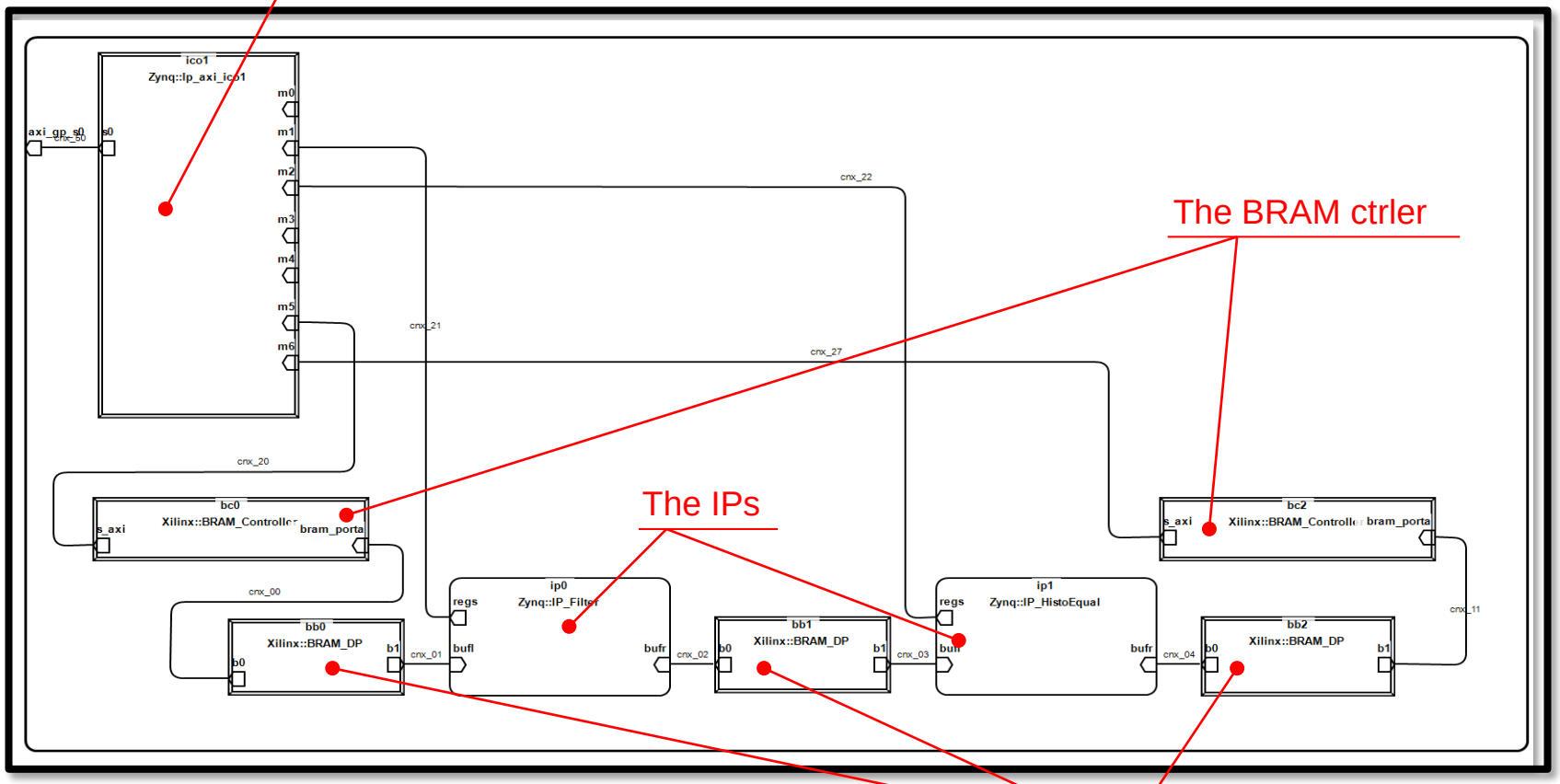


Back

Design alternatives

Alternative #1: *2 IPs, buffers in BRAM*

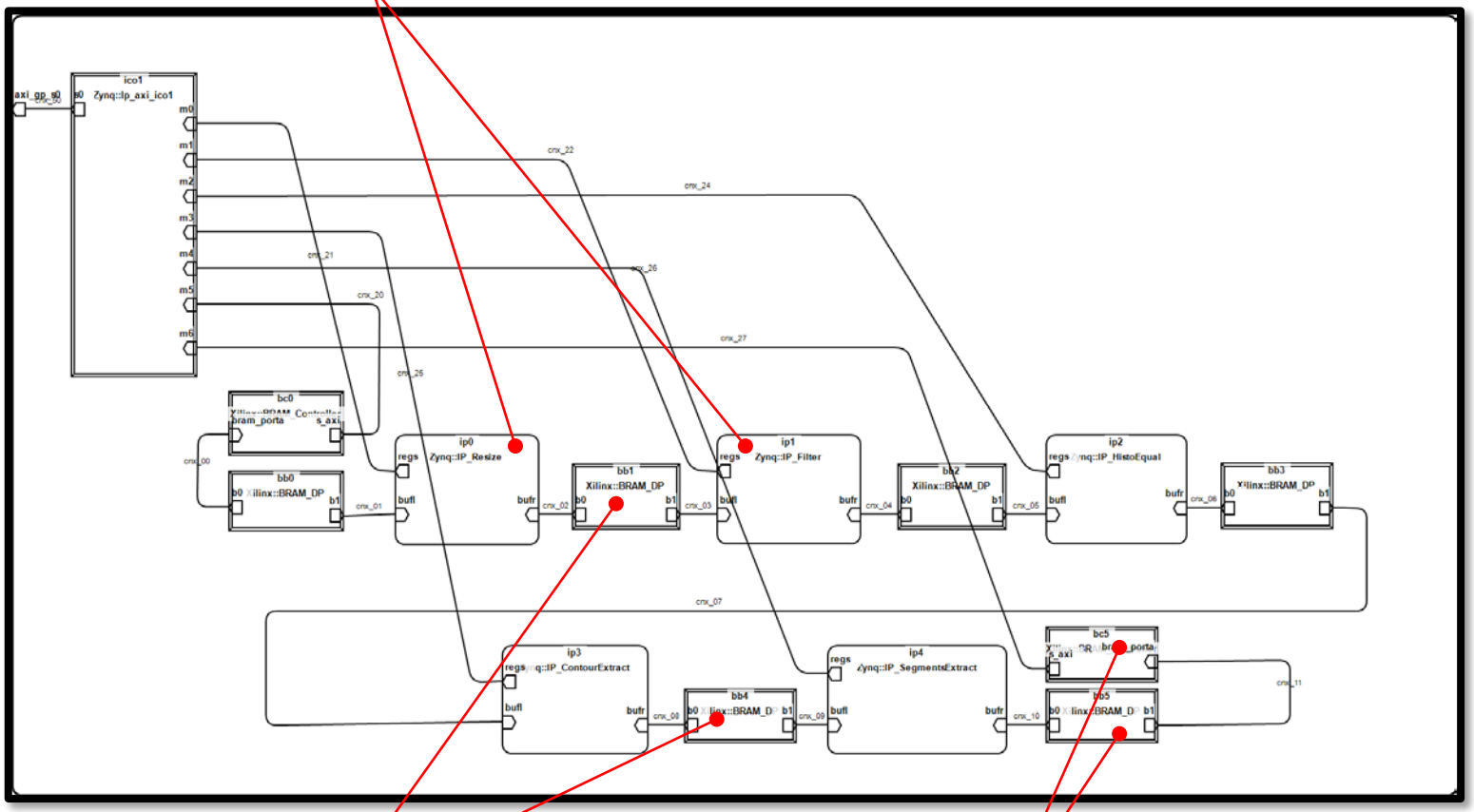
Memory and register access interco.



Back

Design alternatives

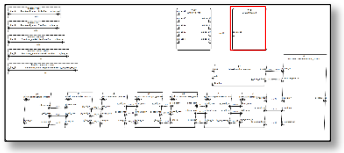
Alternative #2: 5 IPs, buffers in BRAM



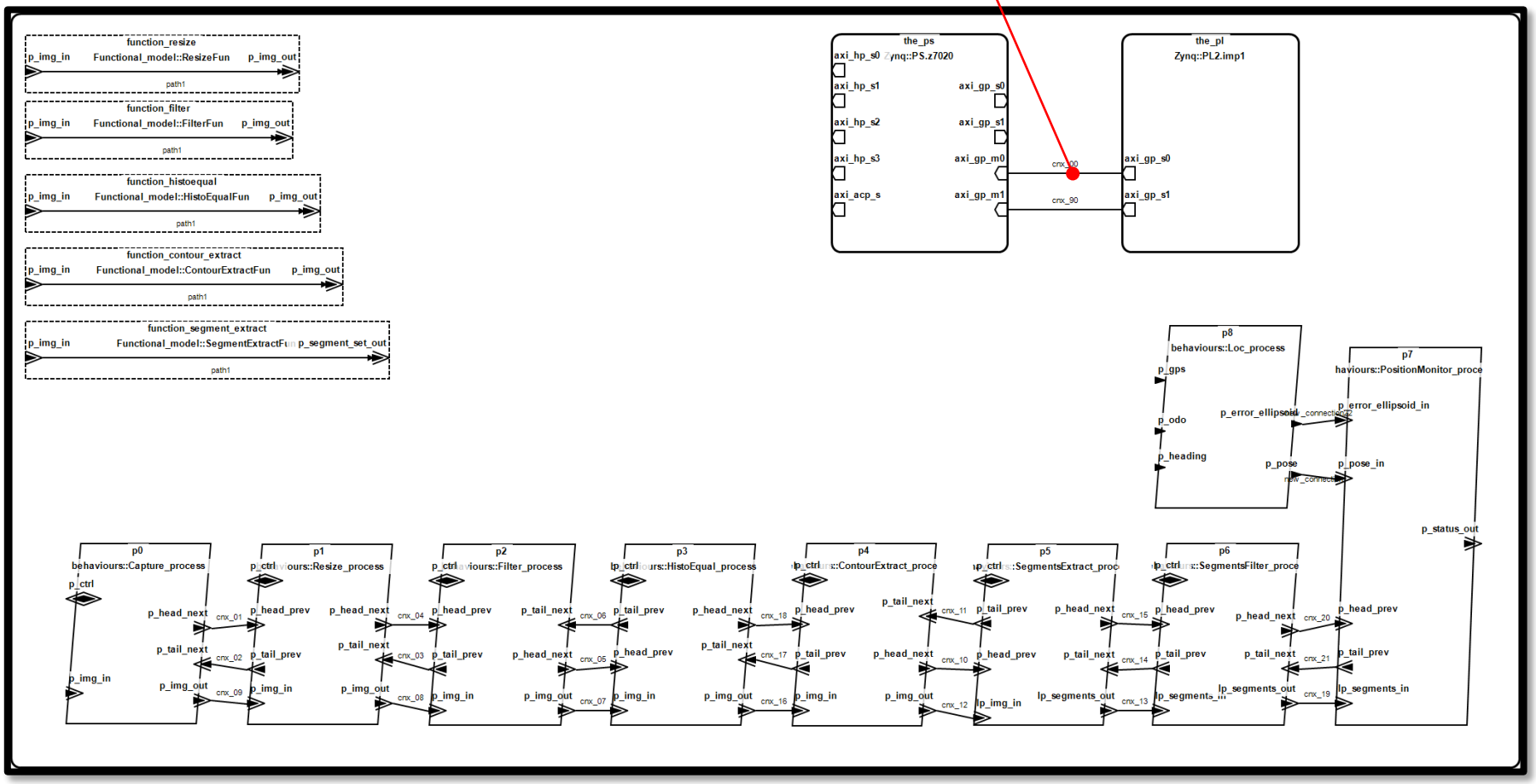
Back

Design alternatives

Alternative #3: 5 IPs, buffers are in BRAM, accessed via AMBA



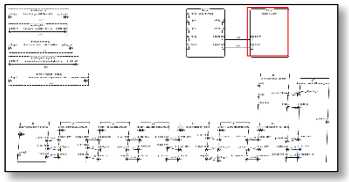
IPs are slaves

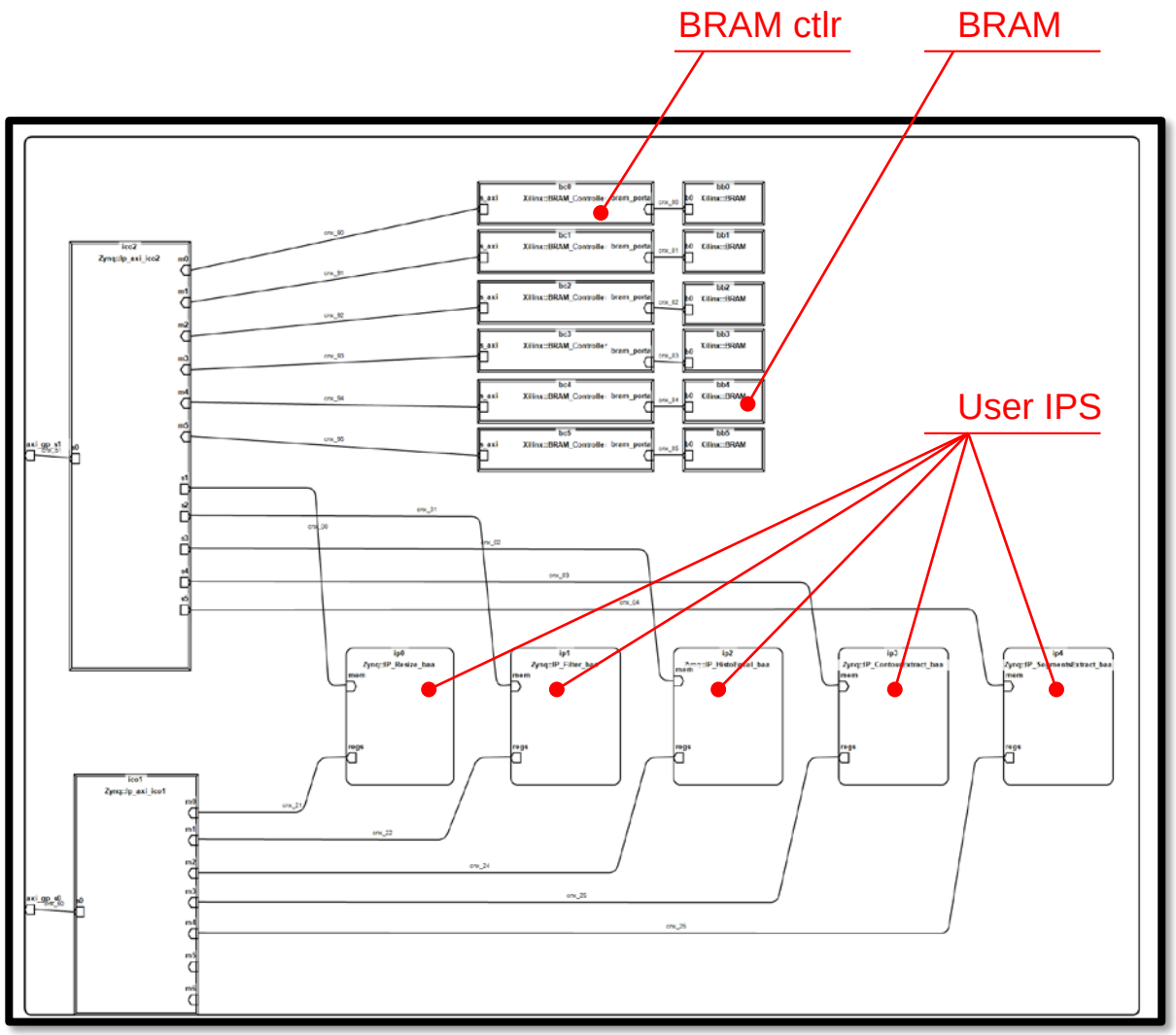


Back

Design alternatives

Alternative #3: 5 IPs, buffers are in BRAM, accessed via AMBA

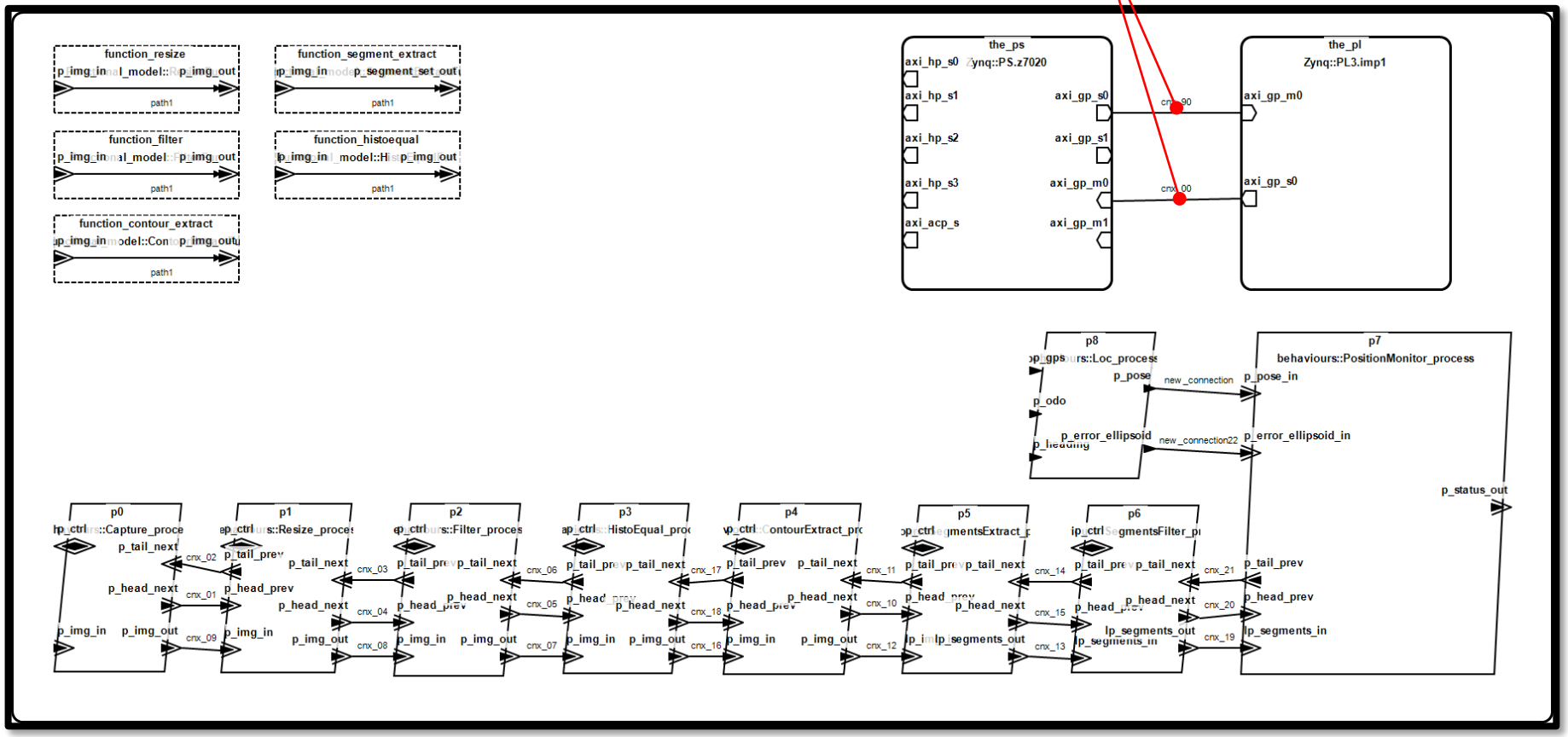




Back

Design alternatives

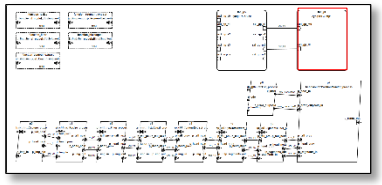
Alternative #3: buffers are in DDR



Back

Design alternatives

Alternative #3: *5 IPs, data in DDR*



Register access interco.

The IPs

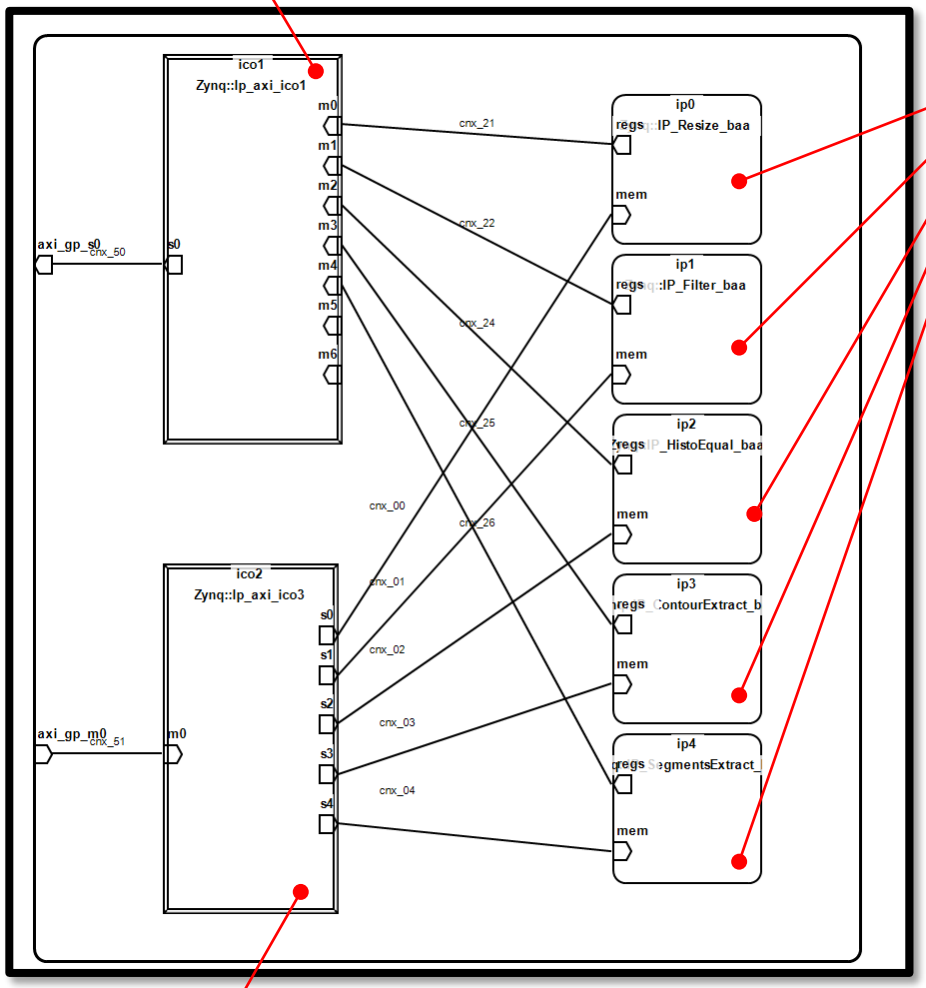
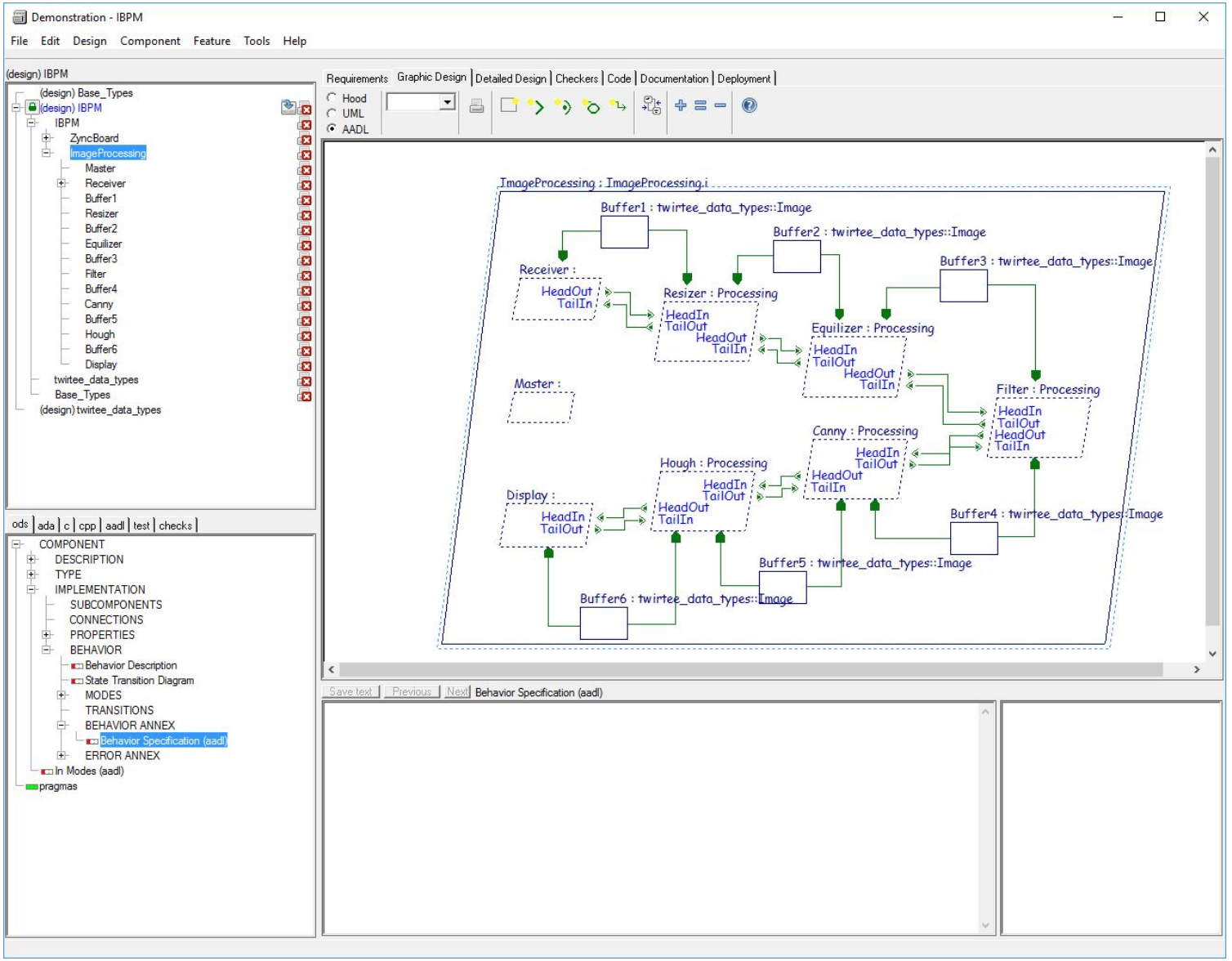


Image access interco.

Back

Evaluation using Ellidiss' Marzhin



Back

Evaluation using Ellidiss' Marzhin

Configuration #1

Configuration #2

Configuration #3

Back

Early validation with AADL
Achievements / lessons

On going work

- AADL behavioral model being developed for 2 parts:
  - Image processing function
  - Power supply function
- Modeling using the BA, simulation using Marzhin.

Perspectives

- Improvements of Marzhin (incl. coverage of the BA)?
- Execution semantics for hardware parts?
- Transformation to other simulators?

Back

Experiments with SpaceStudio

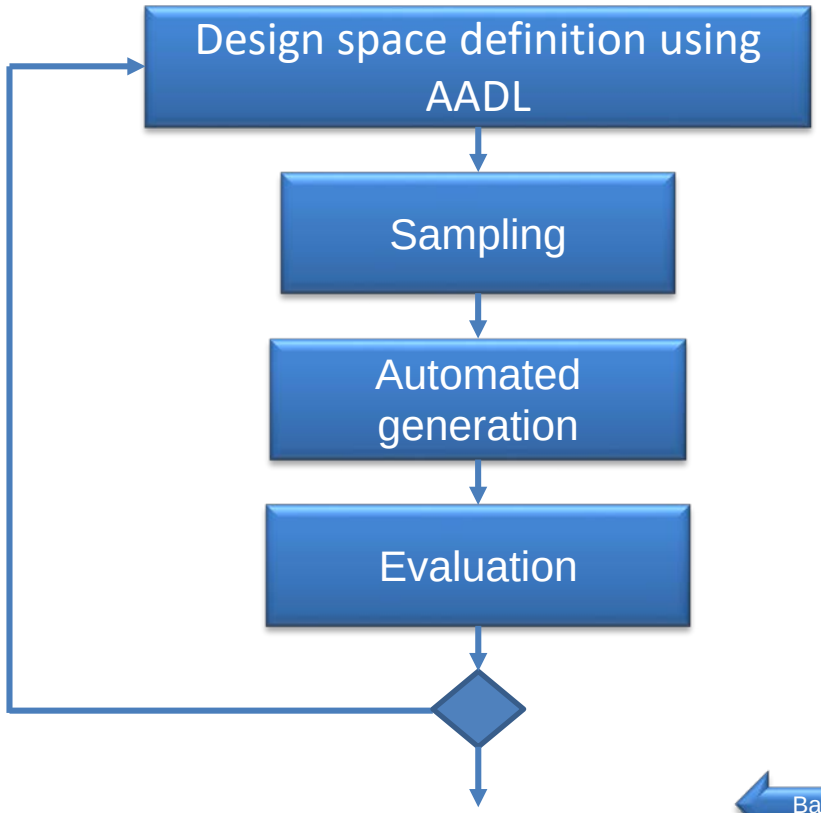
Achievements - Lessons - Perspectives

Achievements

- Transformation AADL to SpaceCo operational but limited

Perspectives

- Handling of regular data patterns (*a la* MARTE/RSM)?
- **Complete automation:**
  - Express the acceptable configurations using constraint language (LUTE?)
  - Generate valid configurations
  - Generate virtual pf configuration scripts
  - Execute model
  - Measure
  - *loop*



Back

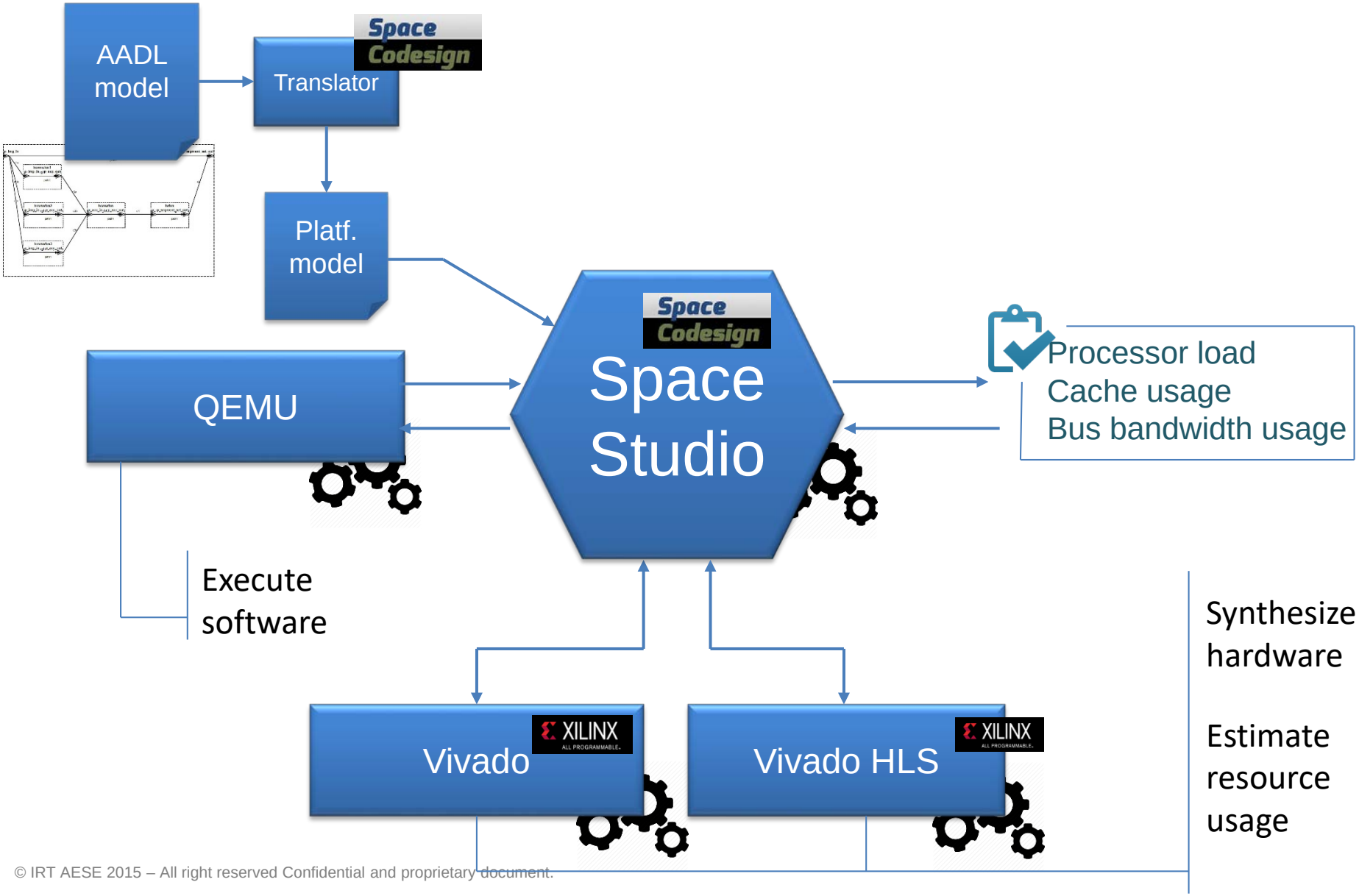
SW and HW design

Design exploration using SpaceCoDesign SpaceStudio

No, thanks…

Design Space Exploration

SpaceStudio



AADL for Design Space exploration

From AADL to SpaceStudio

```
-- The Zynq implementation system.
system implementation main.zynq_2
    subcomponents
        cpu1: processor AADL2SpacecodesignPlatform::core.arm;
        cpu2: processor AADL2SpacecodesignPlatform::core.arm;
        Capture: process AADL2SpacecodesignSoftware::Capturep.impl;
        Resize: process AADL2SpacecodesignSoftware::Resizep.impl;
        Filter: process AADL2SpacecodesignSoftware::Filterp.impl;
        HistoEqual: process AADL2SpacecodesignSoftware::HistoEqualp.impl;
        CountourExtract: process AADL2SpacecodesignSoftware::CountourExtractp.impl;
        Segmentextraction: process AADL2SpacecodesignSoftware::HoughThreadp.impl;
        SegmentFiltering: process AADL2SpacecodesignSoftware::SegmentFilteringp.impl;
        PositionMonitoring: process AADL2SpacecodesignSoftware::PositionMonitoringp.impl;
        busamba: bus AADL2SpacecodesignPlatform::MemoryBus;
        mem: memory AADL2SpacecodesignPlatform::DDR;

    connections
        c_p_img_in_2: port Capture.p_img_out -> Resize.p_img_in;
        c_p_img_in_3: port Resize.p_img_out -> Filter.p_img_in;
        c_p_img_in_4: port Filter.p_img_out -> HistoEqual.p_img_in;
        c_p_img_in_5: port HistoEqual.p_img_out -> Segmentextraction.the_contour;
        c_segment_1: port Segmentextraction.the_segment_set -> SegmentFiltering.p_segment_set_in;
        c_segment_2: port SegmentFiltering.p_segment_set_out -> PositionMonitoring.p_segment_set_in;
        b0: bus access busamba <-> cpu1.bus_ba;
        b00: bus access busamba <-> cpu2.bus_ba;
        b1: bus access busamba <-> mem.Mem_Bus;

    properties
        Actual_Processor_Binding => (reference (cpu1)) applies to Capture;
        Actual_Processor_Binding => (reference (cpu1)) applies to Resize;
        Actual_Processor_Binding => (reference (cpu1)) applies to Filter;
        Actual_Processor_Binding => (reference (cpu1)) applies to HistoEqual;
        Actual_Processor_Binding => (reference (cpu1)) applies to CountourExtract;
        Actual_Processor_Binding => (reference (cpu2)) applies to SegmentExtraction;
        Actual_Processor_Binding => (reference (cpu2)) applies to SegmentFiltering;
        Actual_Processor_Binding => (reference (cpu2)) applies to PositionMonitoring;
```

```
--
-- Cores
--

-- A generic core specification
processor core
    features
        bus_ba: requires bus access MemoryBus;
end core;

-- An ARM core
processor implementation core.arm
    properties
        SpaceStudio::Subtype_component => armCortexA9;
end core.arm;

-- An MicroBlaze soft core
processor implementation core.microblaze
    properties
        SpaceStudio::Subtype_component => uBlaze;
end core.microblaze;
```

```
--
-- Buses

bus MemoryBus
    properties
        SpaceStudio::Subtype_component => AMBA_AXIBus_LT;
end MemoryBus;
```

```
-- The Hough process
process HoughThreadp
    features
        the_contour: in data port Domain::Image;
        the_segment_set: out data port Domain::SegmentSet;
end HoughThreadp;

process implementation HoughThreadp.impl
    subcomponents
        P1: thread HoughThread.impl;
    connections
        c_the_contour: port the_contour -> P1.the_contour;
        c_the_segment_set: port P1.the_segment_set -> the_segment_set;
end HoughThreadp.impl;
```

AADL for Design Space exploration

From AADL to SpaceStudio

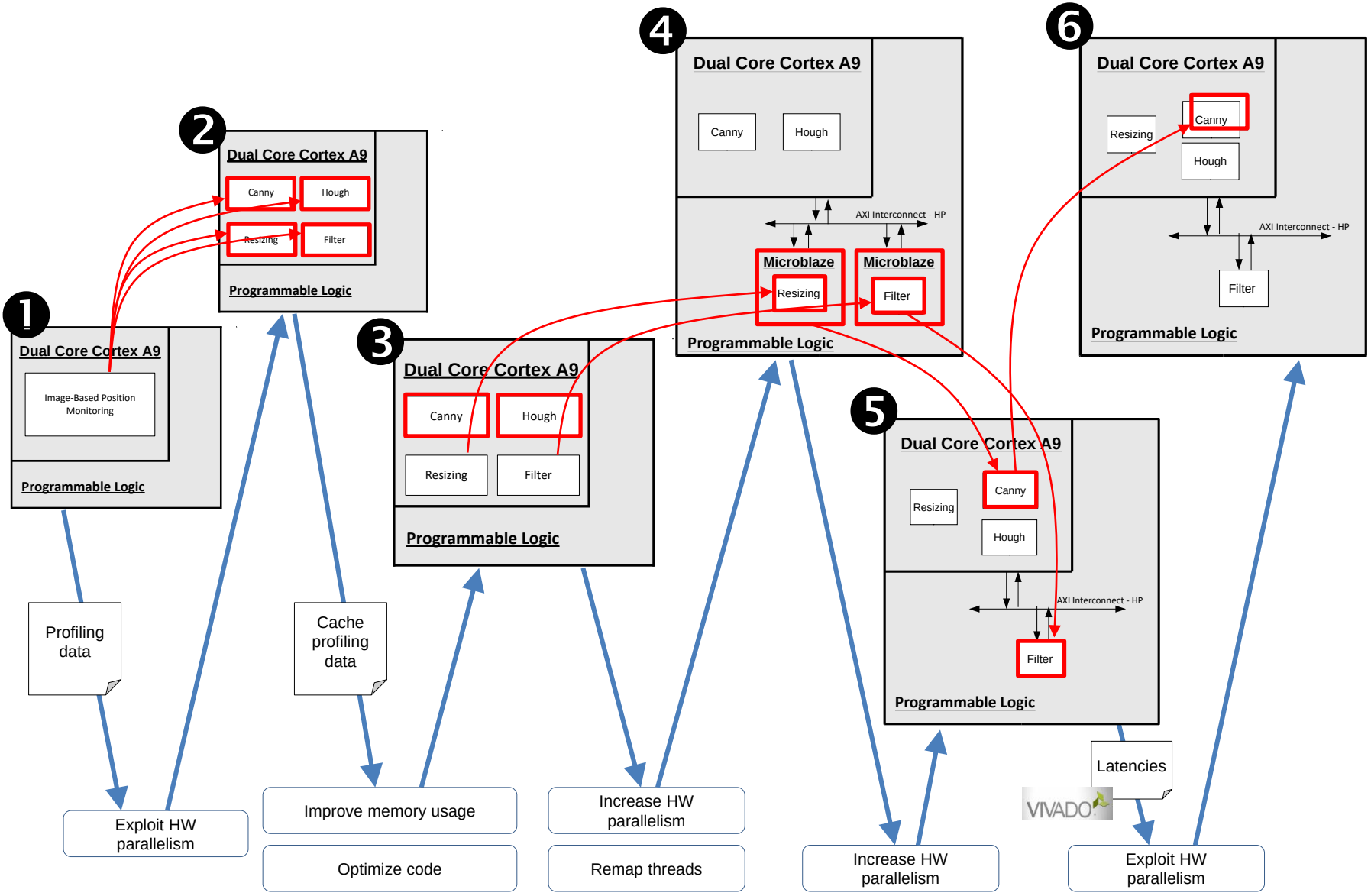
```
-- ----------------
-- A Zynq processing unit with a 2-core Xillinx
of RAM
-- ----------------
system implementation IBPM_Pf.zynq_2
    subcomponents
        the_cpu: system multicore.zynq_2;
        the_mem: memory hw_components::DDR {
        SpaceStudio::Subtype_component => gaisle
        the_mem_bus: bus { SpaceStudio::Subtype_
        AMBA_AXIBus_LT; };
    [...]
end IBPM_Pf.zynq_2;
```

```
PROJECT_NAME = 'main.zynq_2'
BUS_NAMES = [('busamba','AMBA_AXIBus_LT')]
MEMORY_NAMES = [('DDR','gaislerddr','busamba')]
# processor (processor_name, processor_type, simulation_model, binding_bus,
scheduling_protocol)
PROCESSOR_NAMES = [('cpu1','armCortexA9','None','busamba','round_robin'), ('cpu2',
'armCortexA9','None','busamba','round_robin')]
# module (module_name, binding_processor, priority, dispatch_protocol, period)
MODULE_NAMES = [('system_Capture_P1','cpu1',18,'none',0), ('system_Resize_P1','cpu1',18,
'none',0), ('system_Filter_P1','cpu1',18,'none',0), ('system_HistoEqual_P1','cpu1',18,
'none',0), ('system_CountourExtract_P1','cpu1',18,'none',0), ('system_Segmentextraction_P1'
,'cpu2',18,'none',0), ('system_SegmentFiltering_P1','cpu2',18,'none',0), (
'system_PositionMonitoring_P1','cpu2',18,'none',0)]
DEVICE_NAMES = []
```

```
-- Hough: hand-coded C implementation
subprogram implementation Hough.hc
    properties
        Source_Text => ("hough.cpp");
        Source_Name => "hough";
end Hough.hc;

--  A generic Hough thread
thread HoughThread
    features
        the_contour: in data port Domain::Image;
        the_segment_set: out data port Domain::SegmentSet;
end HoughThread;

-- A generic implementation of a Hough thread
thread implementation HoughThread.gen
    calls
        c1: {s1: subprogram Hough;};
end HoughThread.gen;
```

```
////////////////////////////////////////////////
///////////////   Constructor   ////////////////
////////////////////////////////////////////////
system_Segmentextraction_P1::system_Segmentextraction_P1(sc_module_name zName, double dClockPeriod,
sc_time_unit ClockPeriodUnit, unsigned char ucID, unsigned char ucPriority, bool bVerbose)
: SpaceBaseModule(zName, dClockPeriod, ClockPeriodUnit, ucID, ucPriority, bVerbose)
{
                    SC_THREAD(thread);
}

////////////////////////////////////////////////
/////////////////   Thread   ///////////////////
////////////////////////////////////////////////
void system_Segmentextraction_P1::thread(void) // Can be either SW or HW
{
unsigned int initValue = 0;
unsigned int timerValue;

    while(1)
    {
        //Execution between 10 and 40 ms

        Hough();


        #if defined(SPACE_SIMULATION_RELEASE) || defined(SPACE_SIMULATION_MONITORING) ||
        defined(SPACE_SIMULATION_DEBUG)
            DeviceRead(XILINXTIMER1_ID, XILINX_TIMER_OFFSET_TCR0, &timerValue);
            if (timerValue - initValue < 0 * 1000)
                deadline::getInstance().addOc(system_Segmentextraction_P1);
            else
                deadline::getInstance().addDl(system_Segmentextraction_P1);
            DeviceRead(XILINXTIMER1_ID, XILINX_TIMER_OFFSET_TCR0, &initValue);
        #endif
    }
}
```

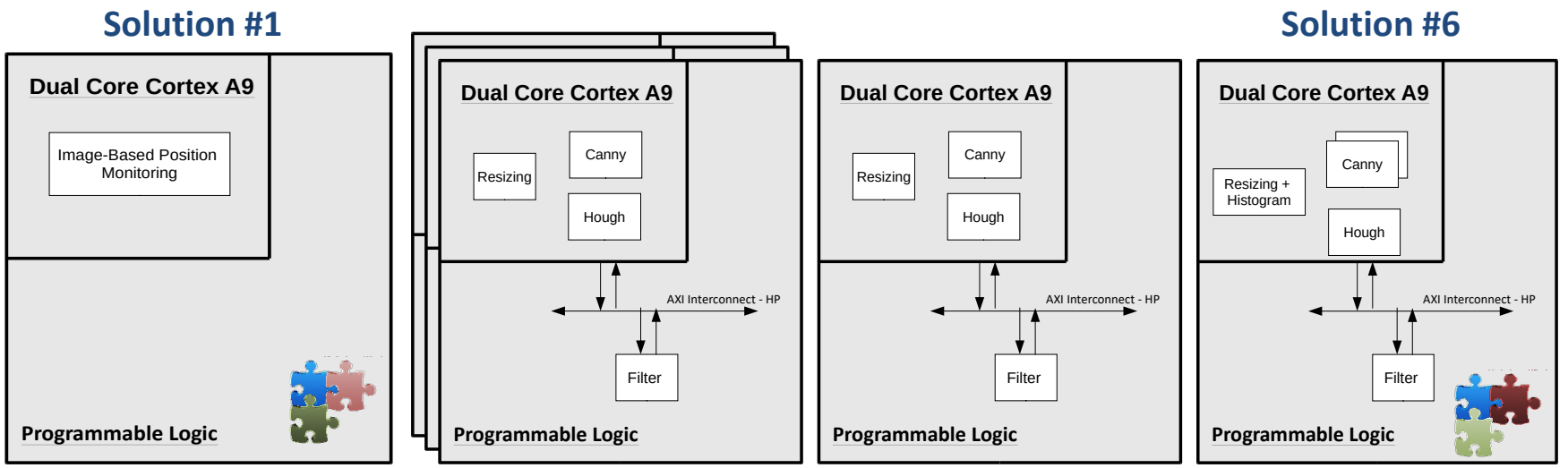
Evaluation process



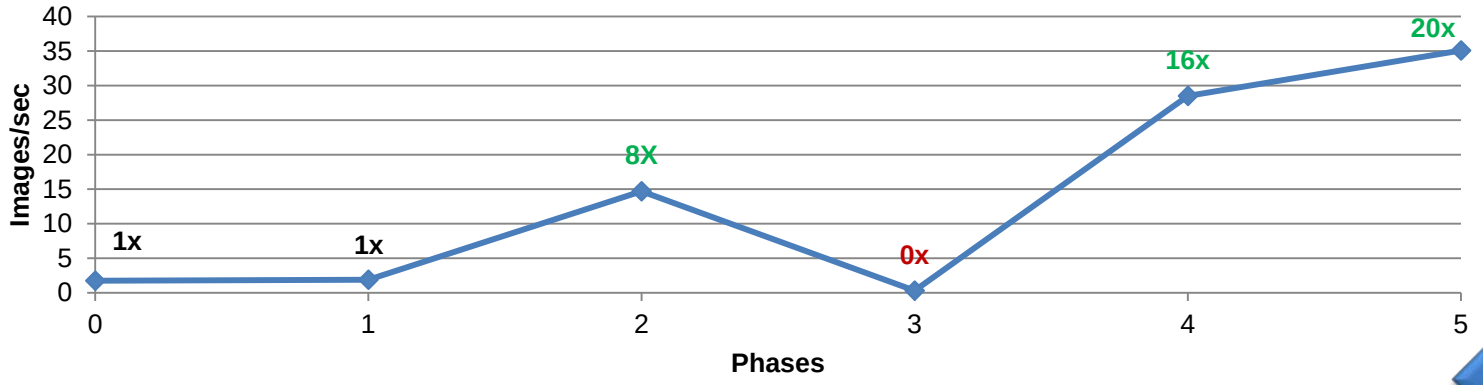
Design space exploration

Experiment with SpaceStudio

Solutions



Speed-up of Image Processing Chain algorithm over the design space exploration



Go back