Speeding Up Robot Control Software Through Seamless Integration With FPGA

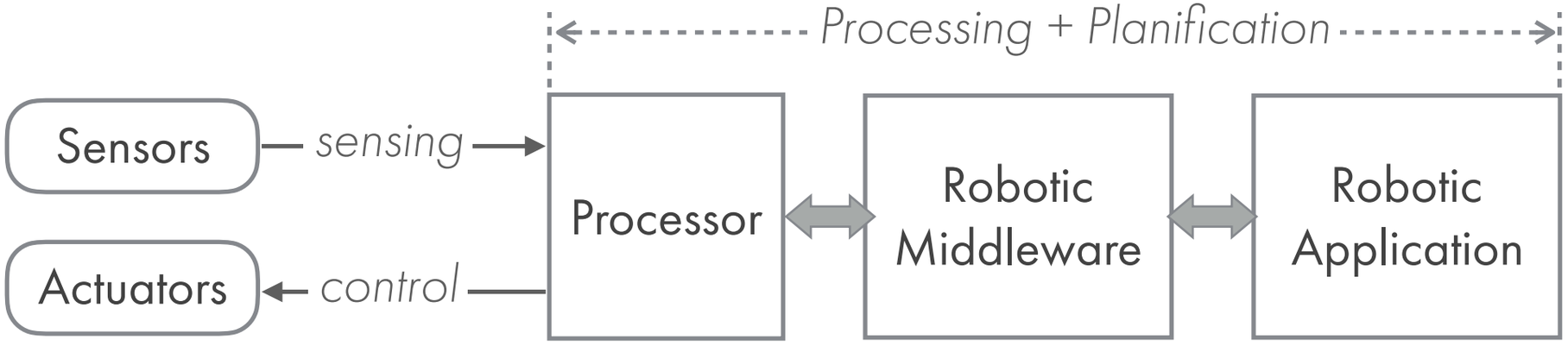
Xuan Sang LE[1,2], Luc Fabresse[1], Jannik Laval[3],
Jean-Christophe Le Lann[2], Loic Lagadec[2] *and* Noury Bouraqadi[1]

[1] Mines Douai-DIA, Univ. Lille
[2] ENSTA Bretagne
[3] DISP Laboratory, University Lumière Lyon 2

Contents

1 Context

Traditional Robotic Computing System



Pros.

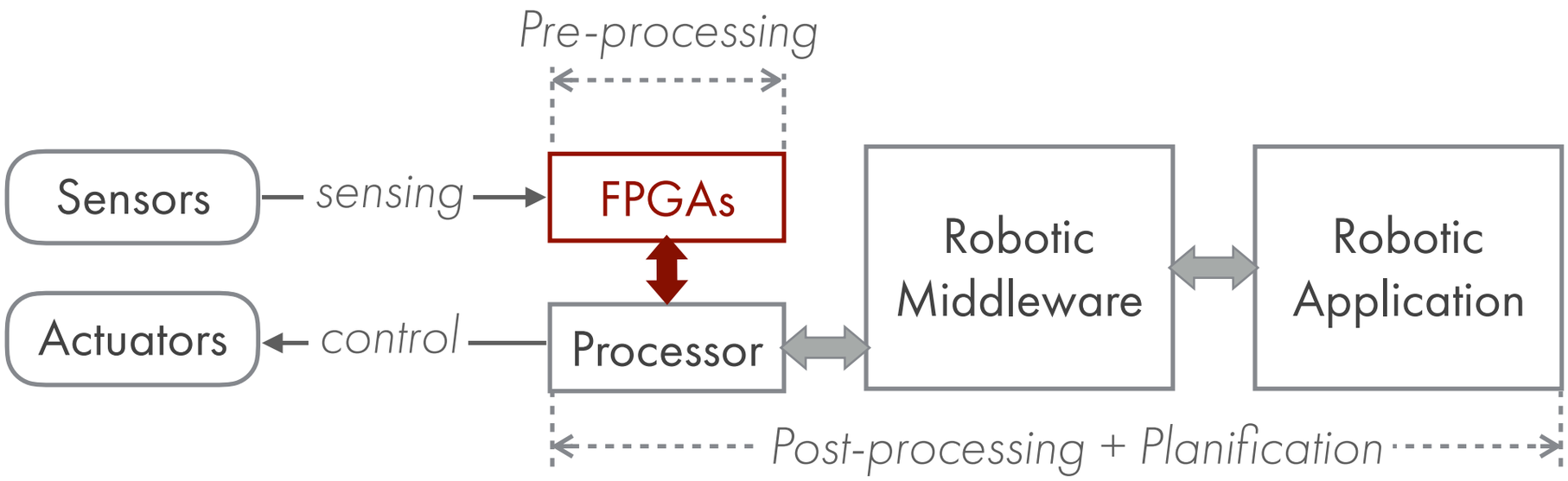
- Accessibility
- Simplicity
- Productivity

vs

Cons.

- Optimization opportunities
- Performance
- Energy Requirement

1 Motivation

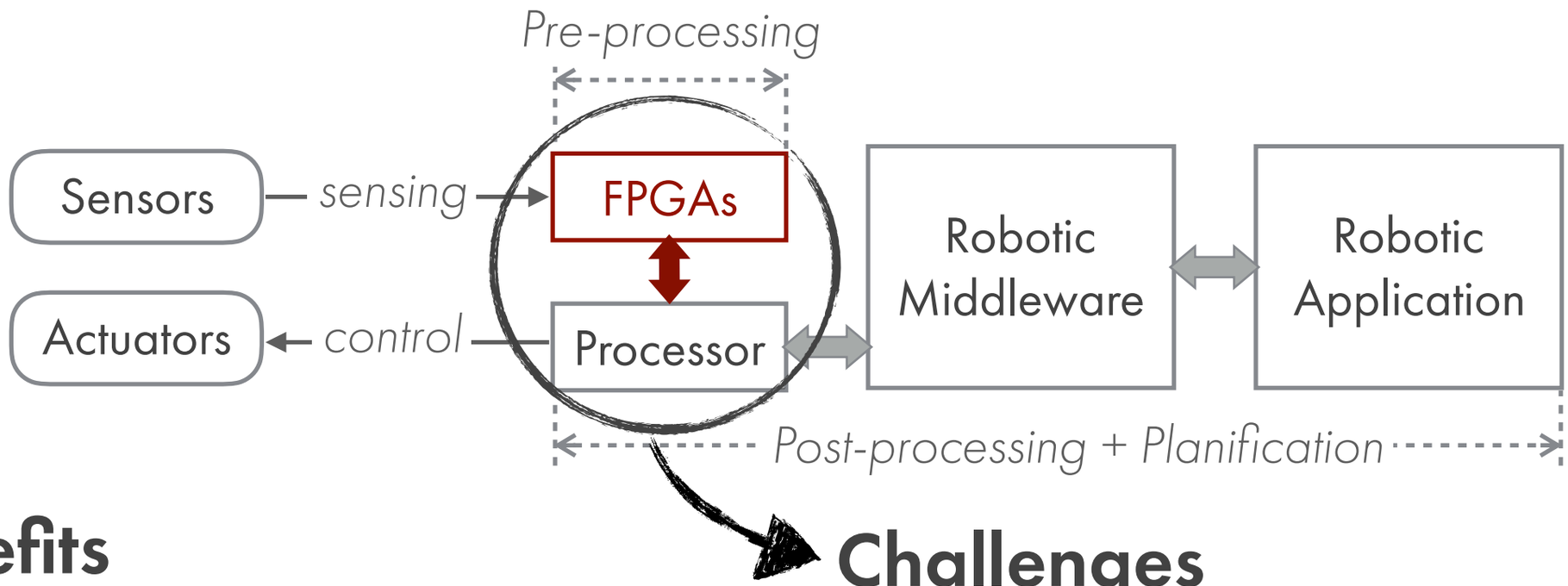


Benefits

- FPGAs
    - Acceleration
    - Hardware Flexibility
- Processor
    - Flexible Software Environment

1 Motivation

FPGAs as Accelerators for Robotic Real Time System



Benefits

- FPGAs
  - Acceleration
  - Hardware Flexibility
- Processor
  - Flexible Software Environment

Challenges

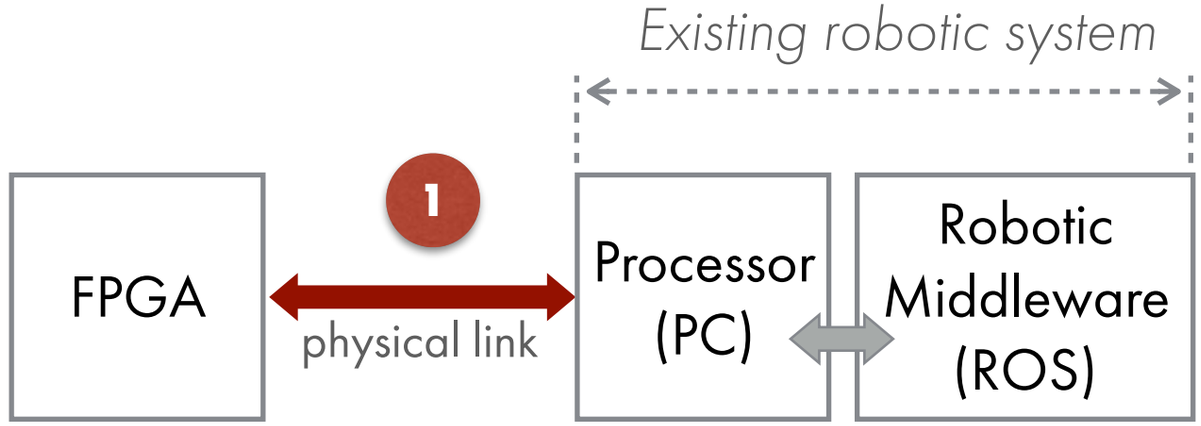
- Require a specific knowledge → loss of productivity
- FPGAs and processor interfacing varies from project to project → development time consuming

A Generic **Software/Hardware Platform** for Easily **Integrating FPGAs** in **Existing Robotic System**

1 Motivation

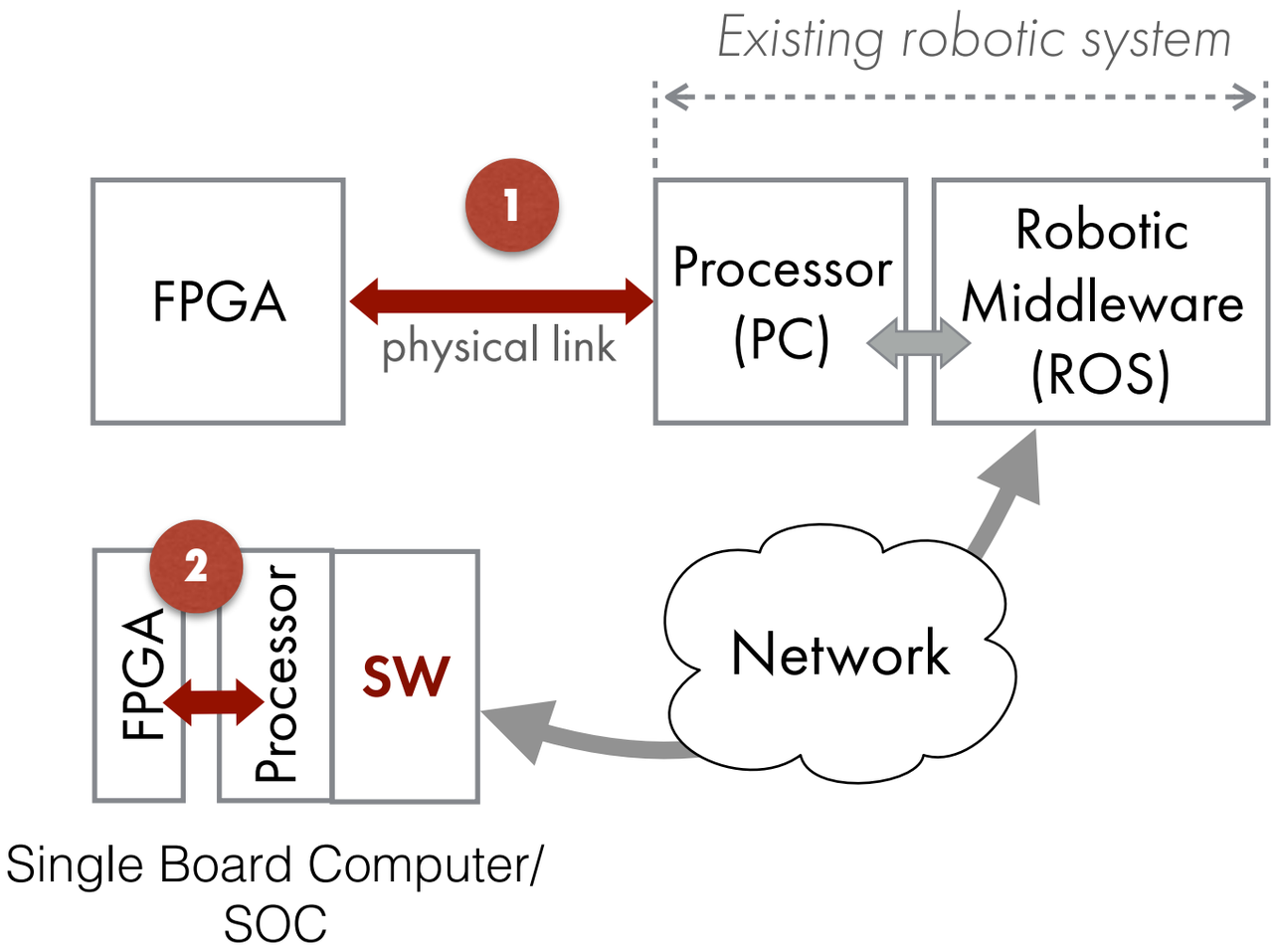
Our Approach

A Generic **Software/Hardware Platform** for Easily **Integrating FPGAs** in **Existing Robotic System**



1 Motivation

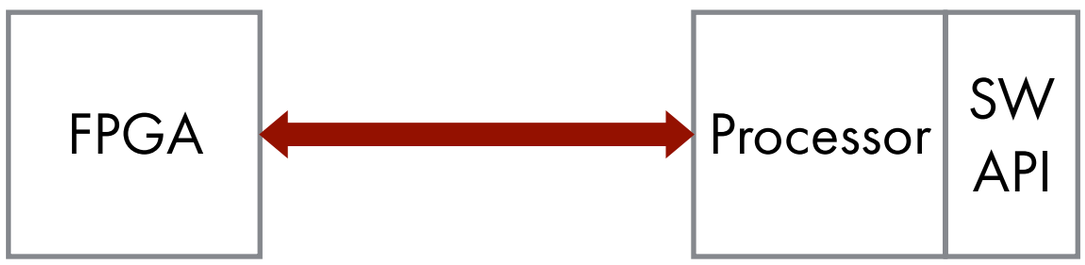
A Generic **Software/Hardware Platform** for Easily **Integrating FPGAs** in **Existing Robotic System**



2 Platform for Software/FPGA Integration

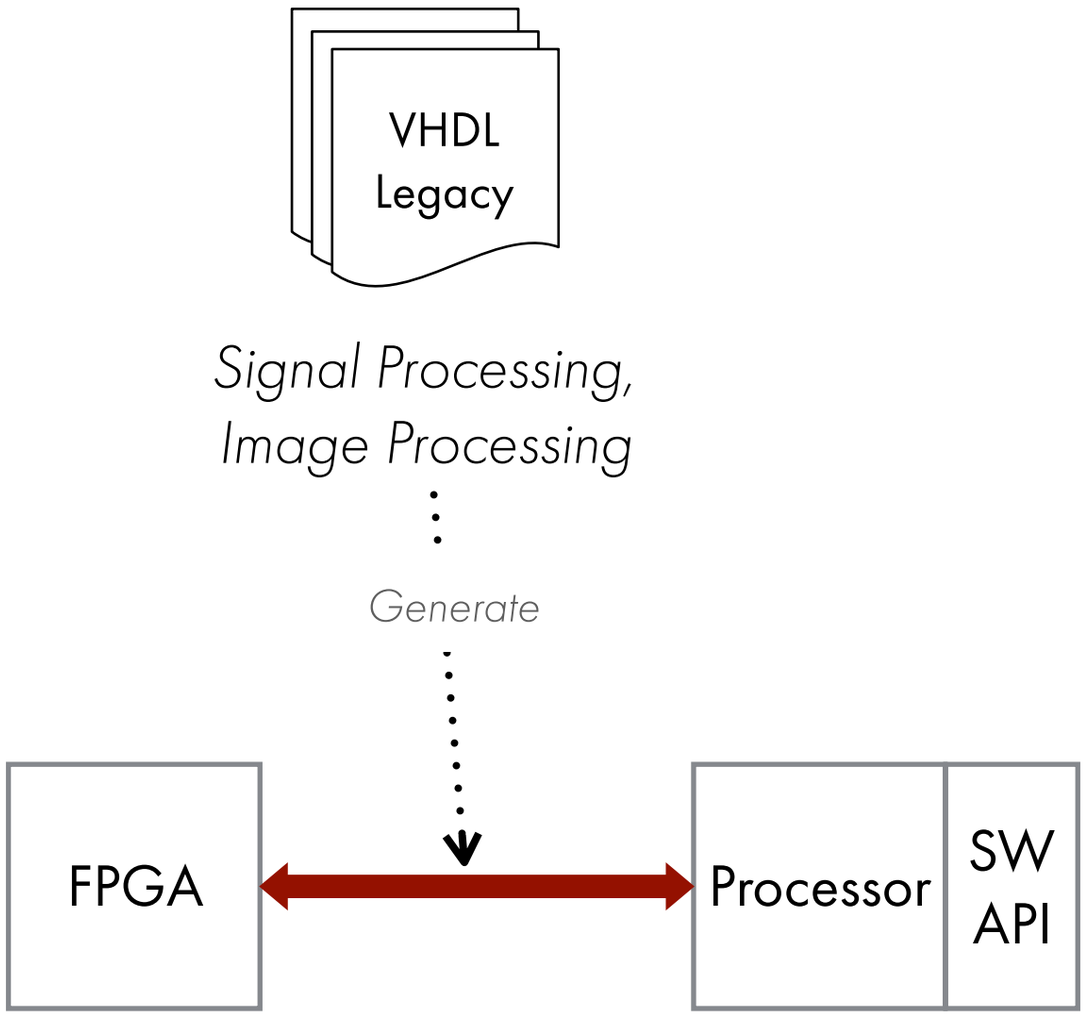
Scenario

Signal Processing,
Image Processing



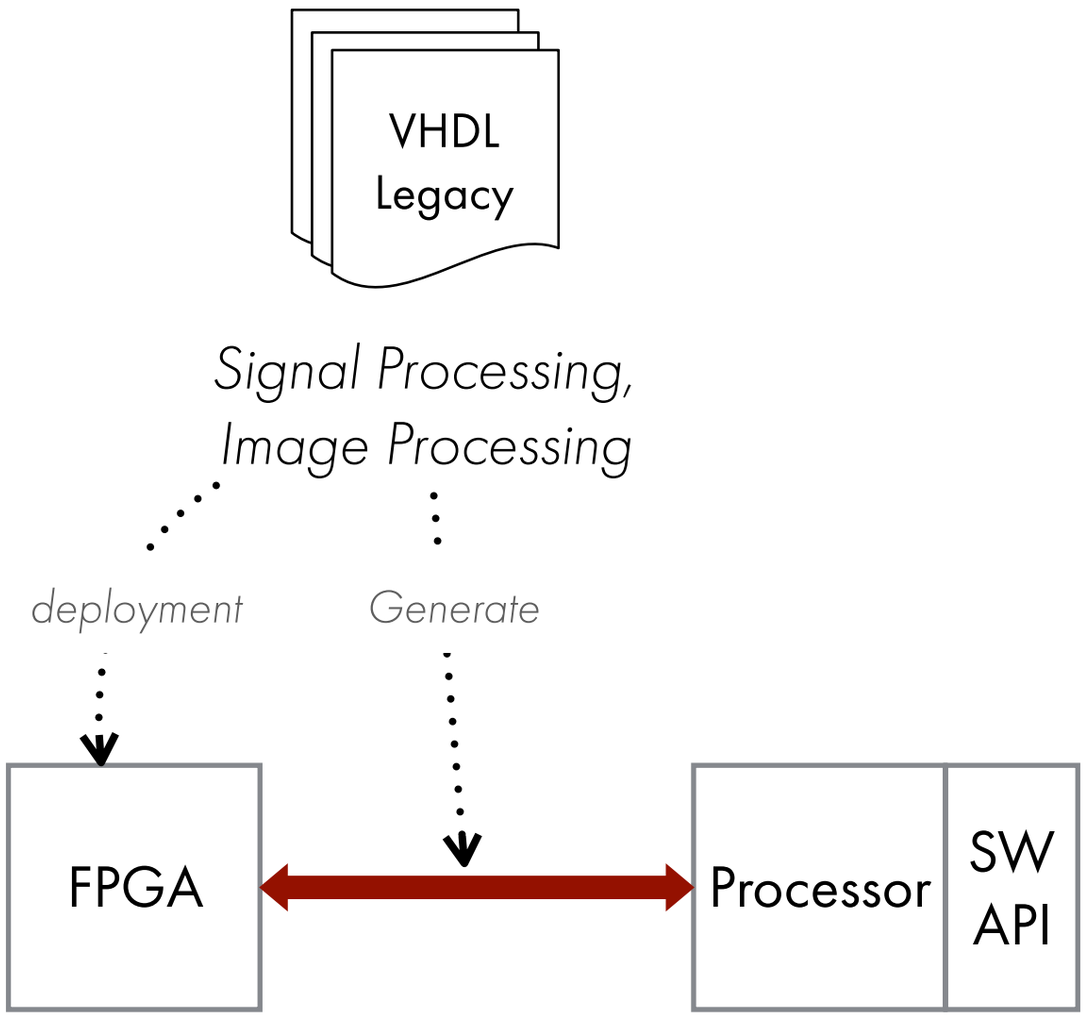
2 Platform for Software/FPGA Integration

Scenario



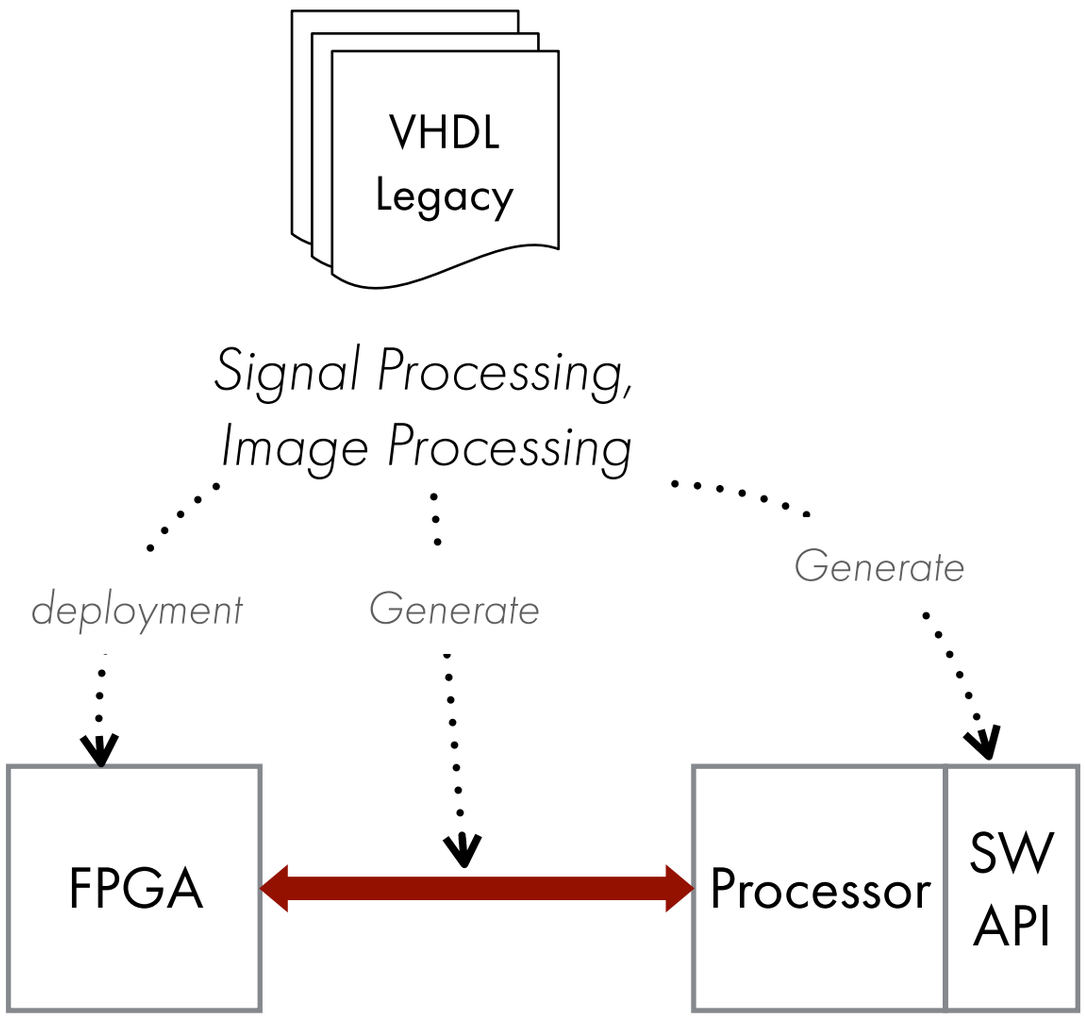
2 Platform for Software/FPGA Integration

Scenario



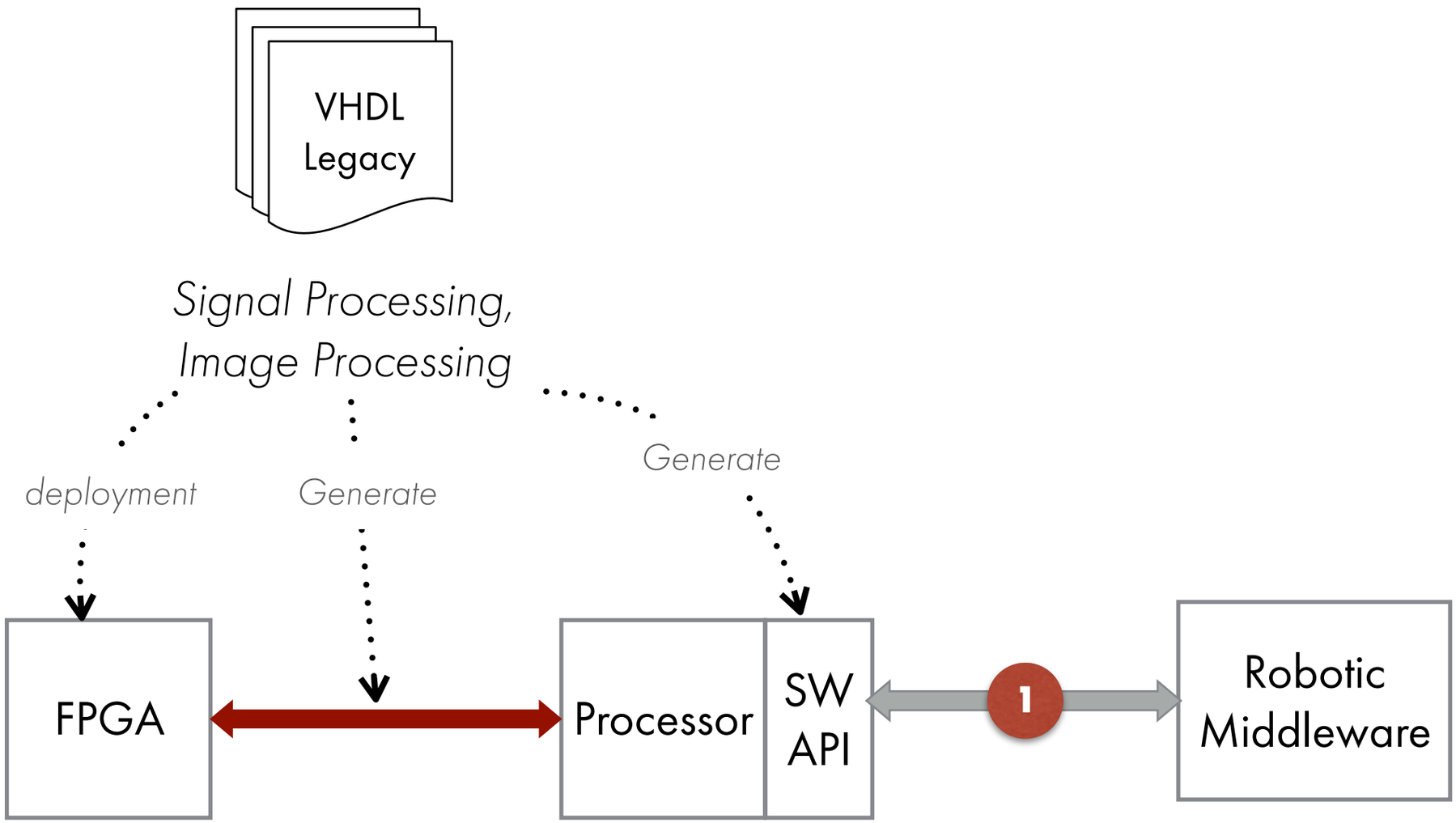
2 Platform for Software/FPGA Integration

Scenario



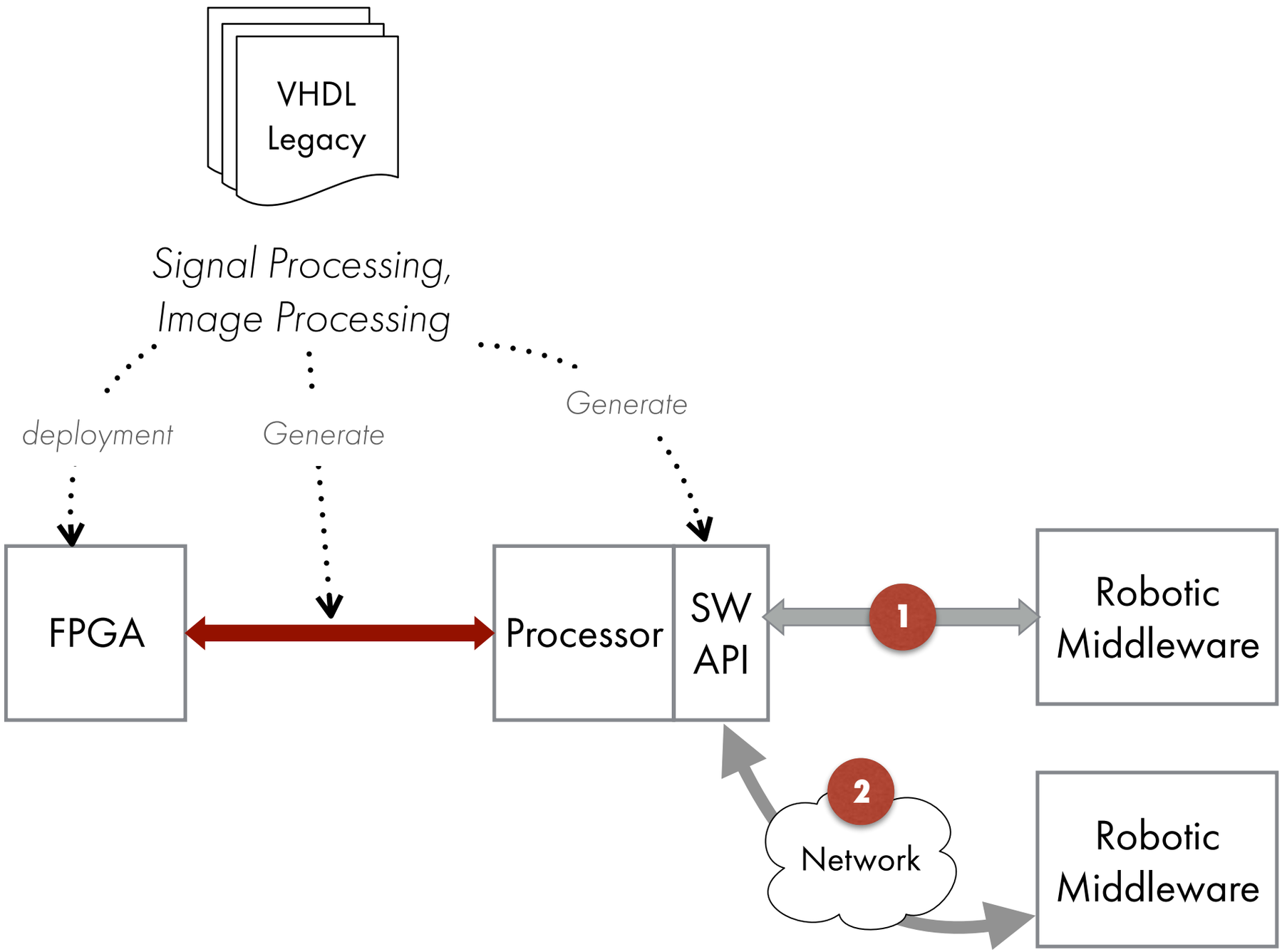
2 Platform for Software/FPGA Integration

Scenario



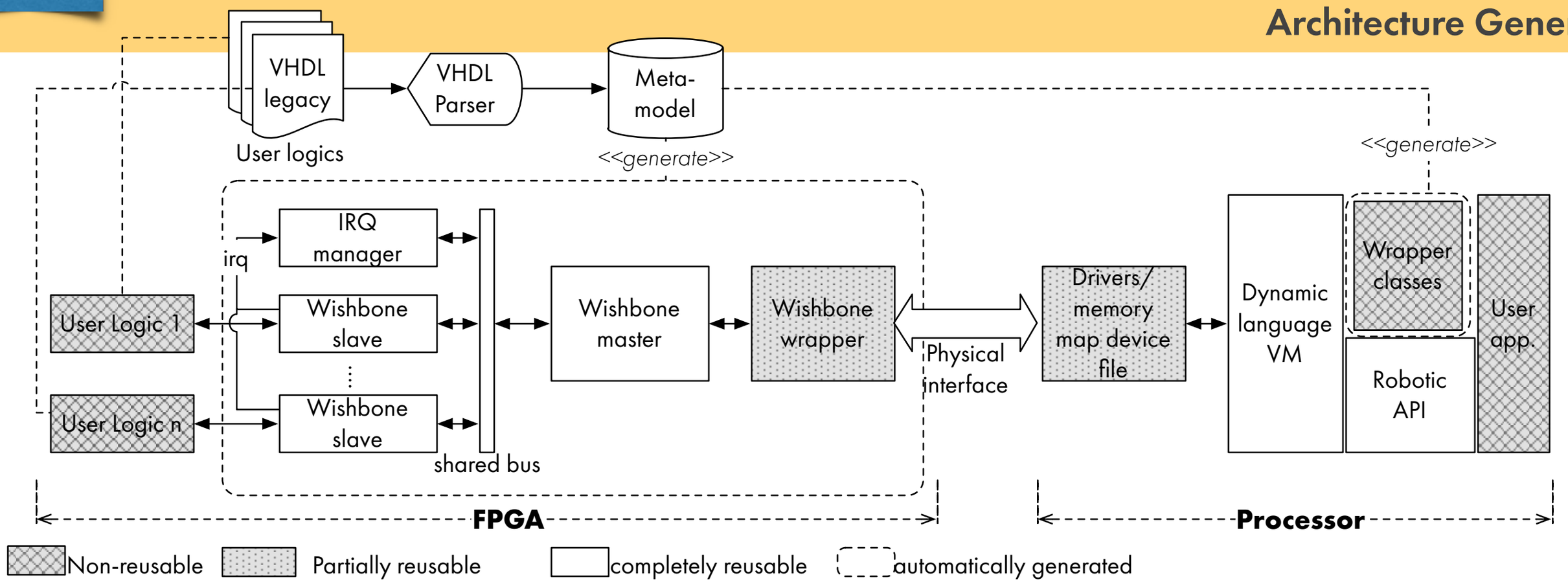
2 Platform for Software/FPGA Integration

Scenario



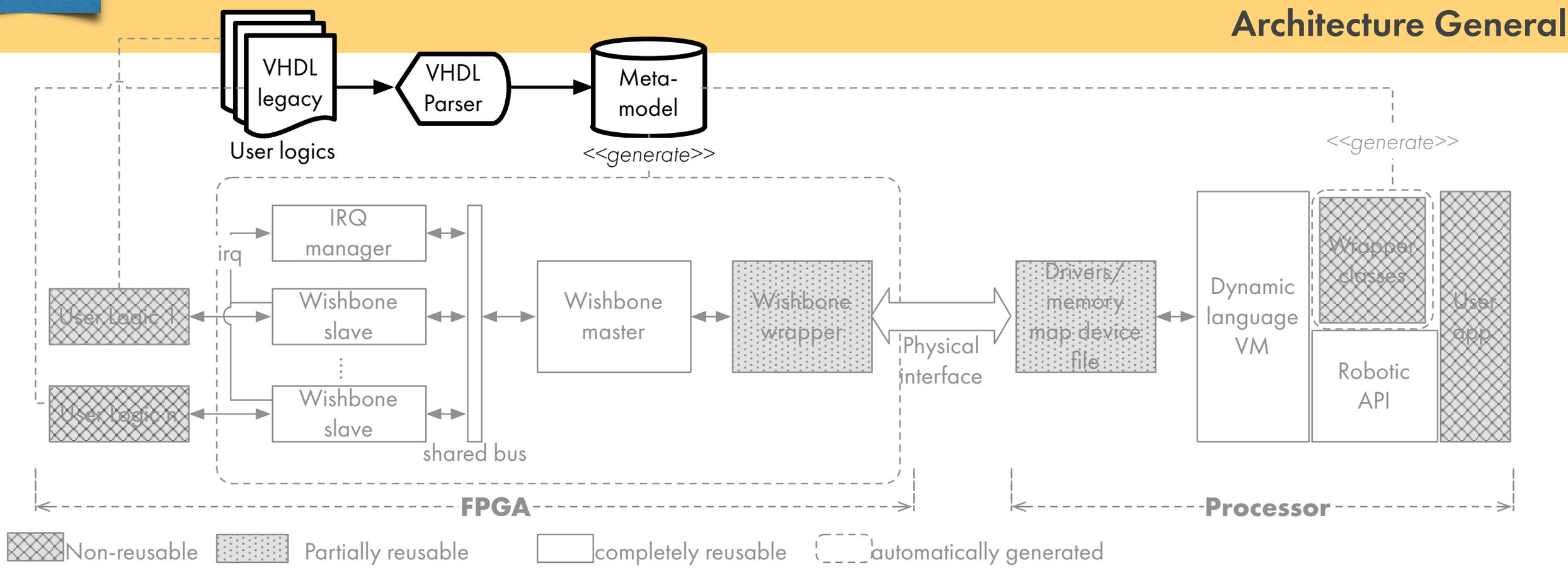
2 Platform for Software/FPGA Integration

Architecture General

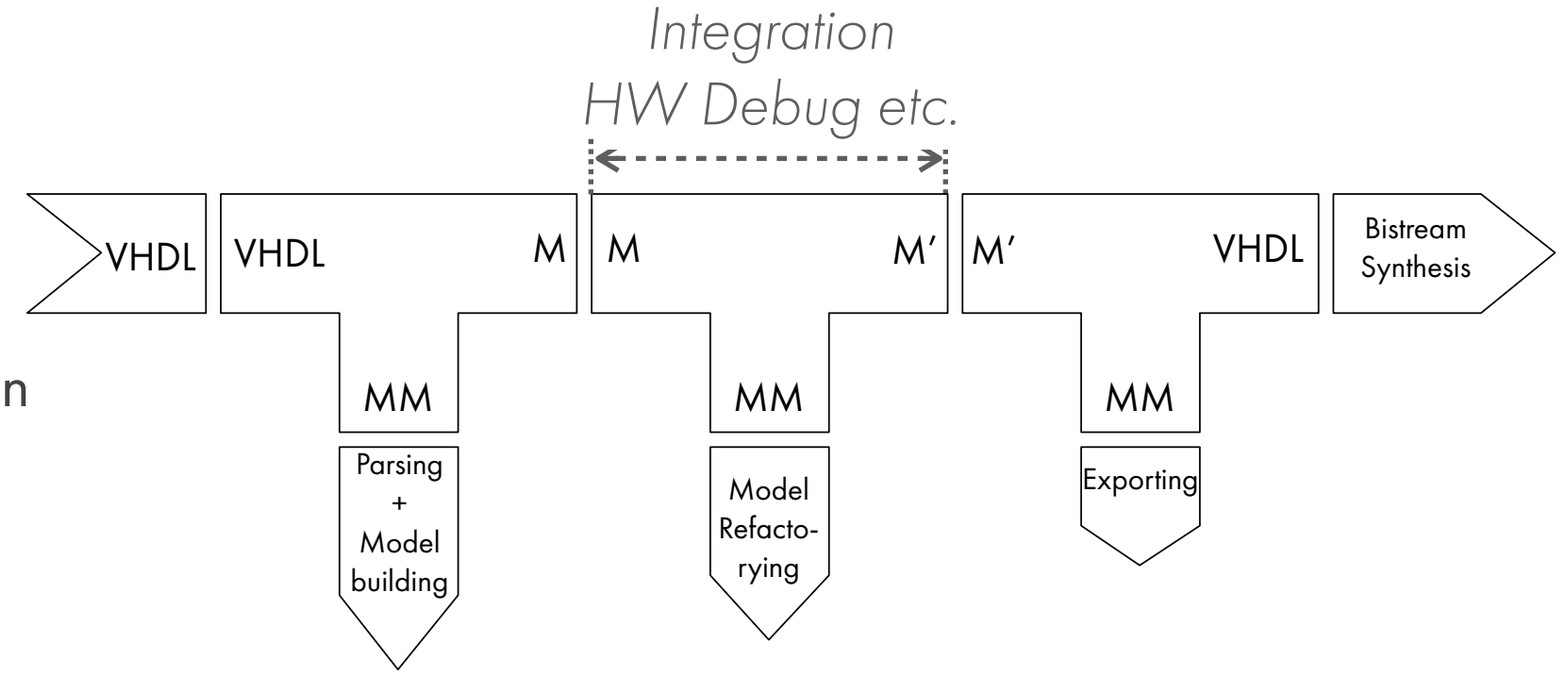


2 Platform for Software/FPGA Integration

Architecture General

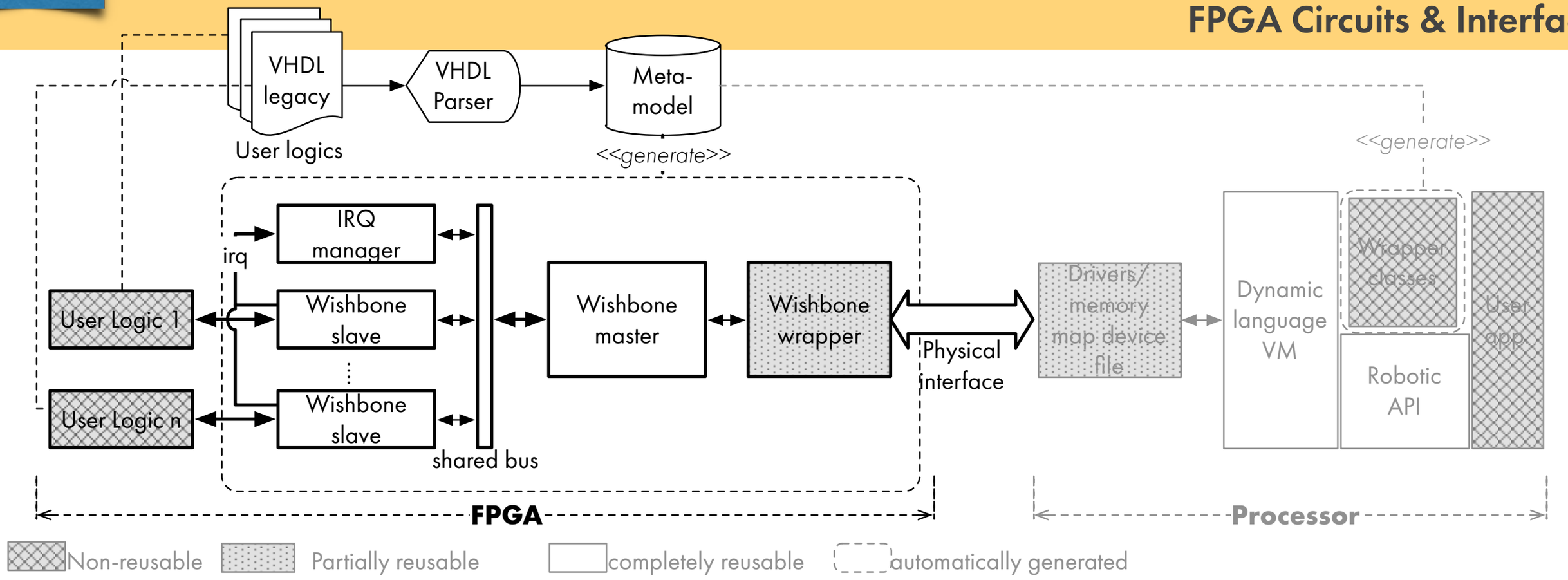


- Meta-model
  - Capture a subset of synthesizable VHDL
  - Oriented-Object circuit description
  - Circuit Model refactoring
  - Implemented in Pharo/Smalltalk



2 Platform for Software/FPGA Integration

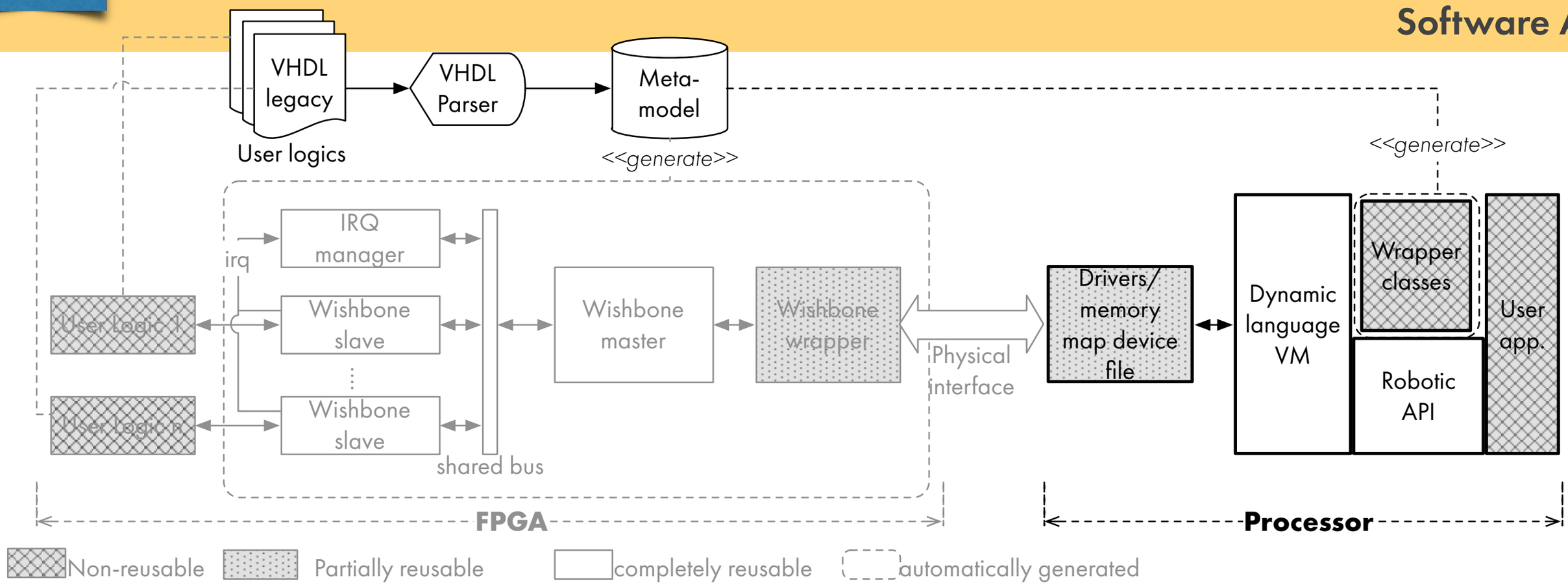
FPGA Circuits & Interface



- **Automatic interface generation**
  - Address/data IO interface supports memory mapping mechanism of SW
  - Wishbone is used for the interface
  - Each user circuit connects to a slave
  - Circuit's registers can be accessed via a virtual address (provided by SW)
  - IRQ support software handlers
  - Automatically circuit registers addressing

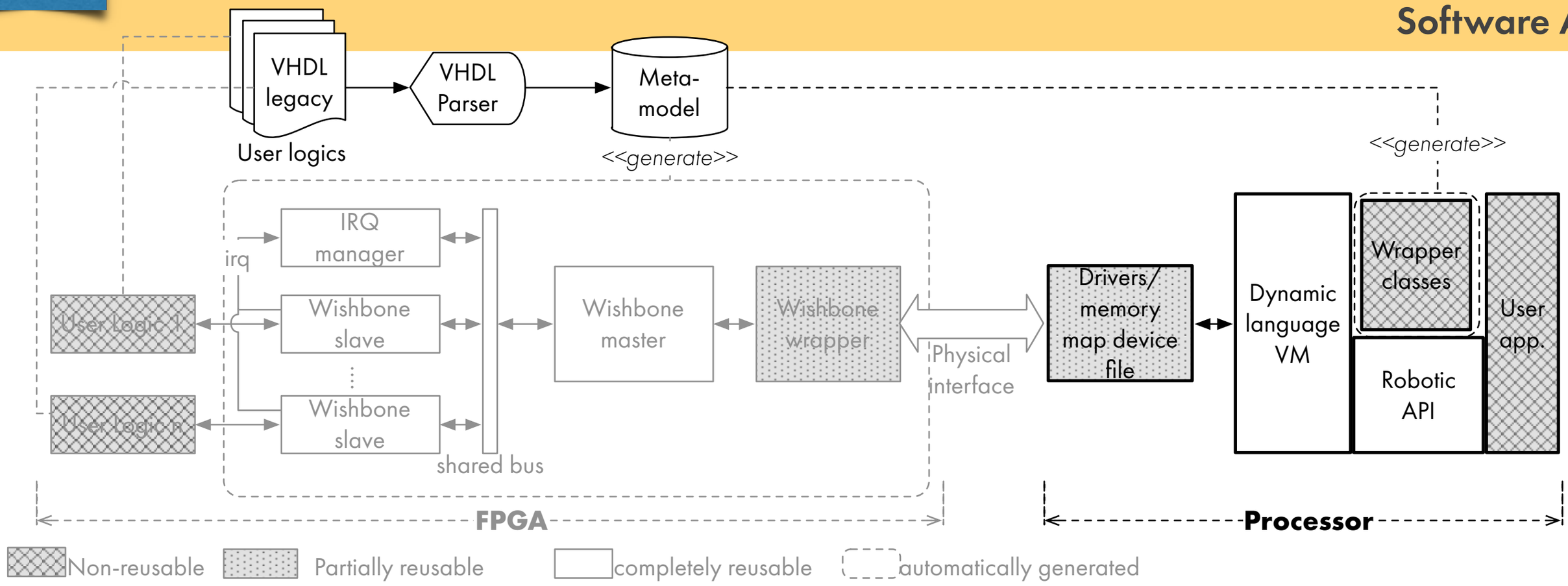
2 Platform for Software/FPGA Integration

Software API



2 Platform for Software/FPGA Integration

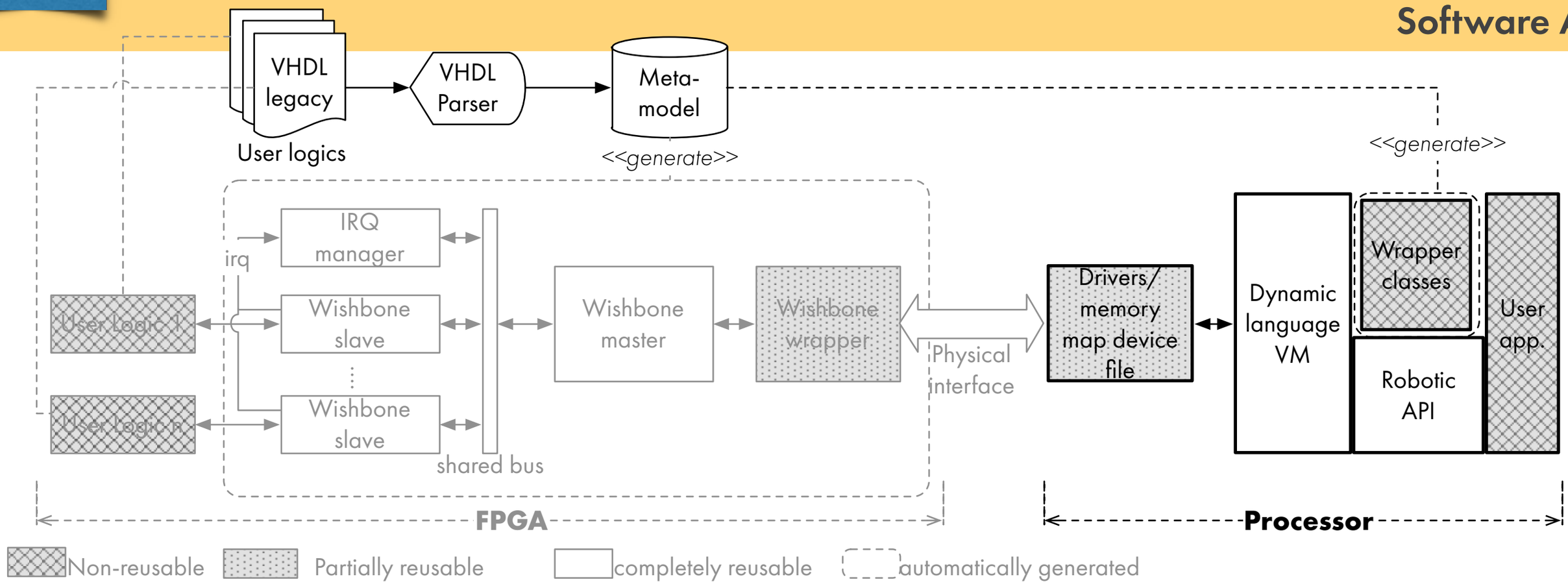
Software API



- **System Driver Layer**
  - User space IO Driver
  - Users circuits are mapped to a virtual memory region
  - Physical interface specific
  - Compliant with Wishbone interface on FPGA

2 Platform for Software/FPGA Integration

Software API



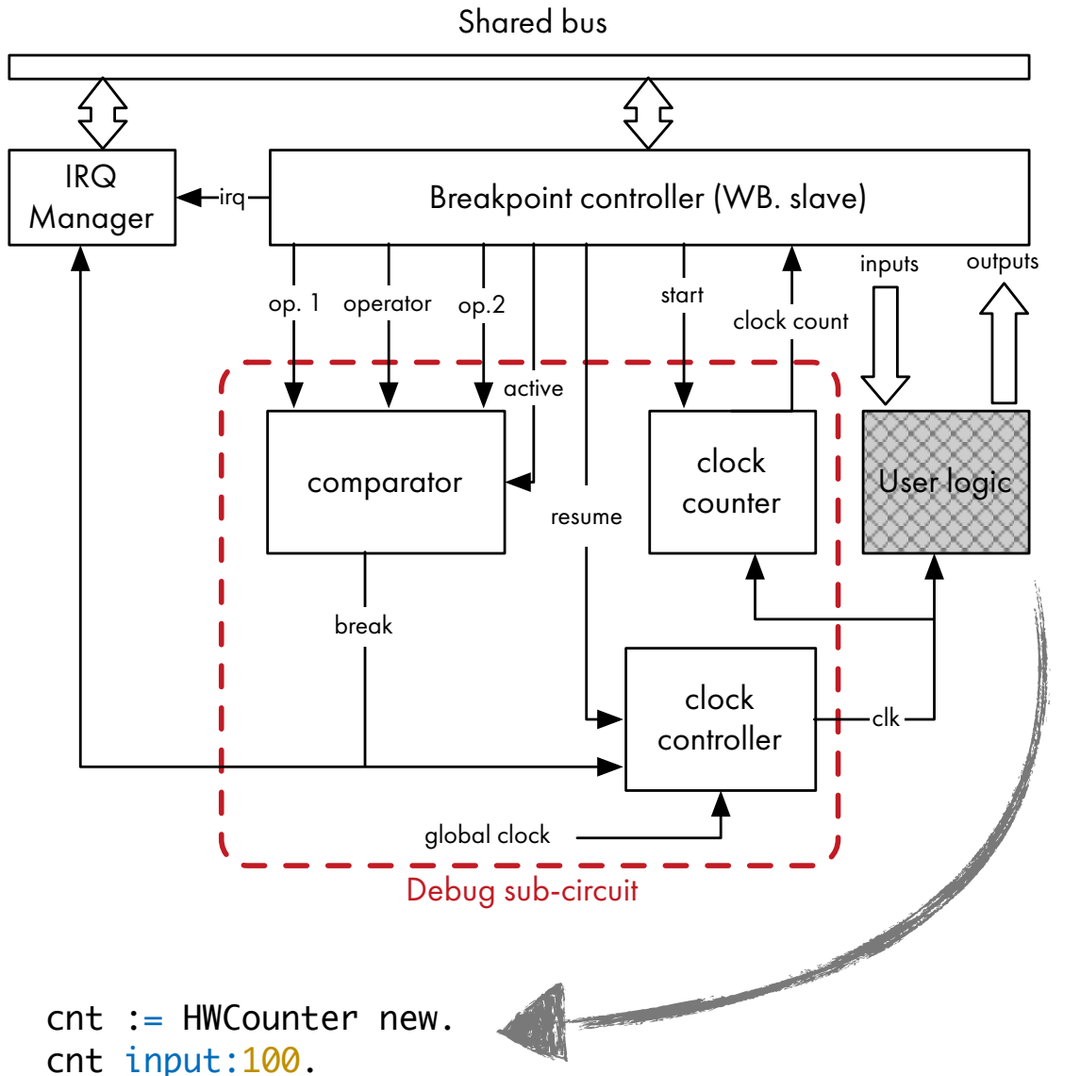
- **System Driver Layer**
  - User space IO Driver
  - Users circuits are mapped to a virtual memory region
  - Physical interface specific
  - Compliant with Wishbone interface on FPGA

- **API Layer**
  - FPGA circuits registers are accessed via virtual memory address
  - Smalltalk VM supports memory mapping at primitives level
  - Wrapper classes consider circuits as Smalltalk Object
  - IRQ handler support
  - Support high level robotic middleware (ROS, REST, etc)

2 Platform for Software/FPGA Integration

Controllability and Debugging

- Hardware Breakpoint
  - A breakpoint controller is injected automatically
  - Clock-gating technique is used to control the user circuit
  - Breakpoint condition can be set from software
  - Execution is resumable
- Draw back
  - Vendor specific (Digital Clock Manager)

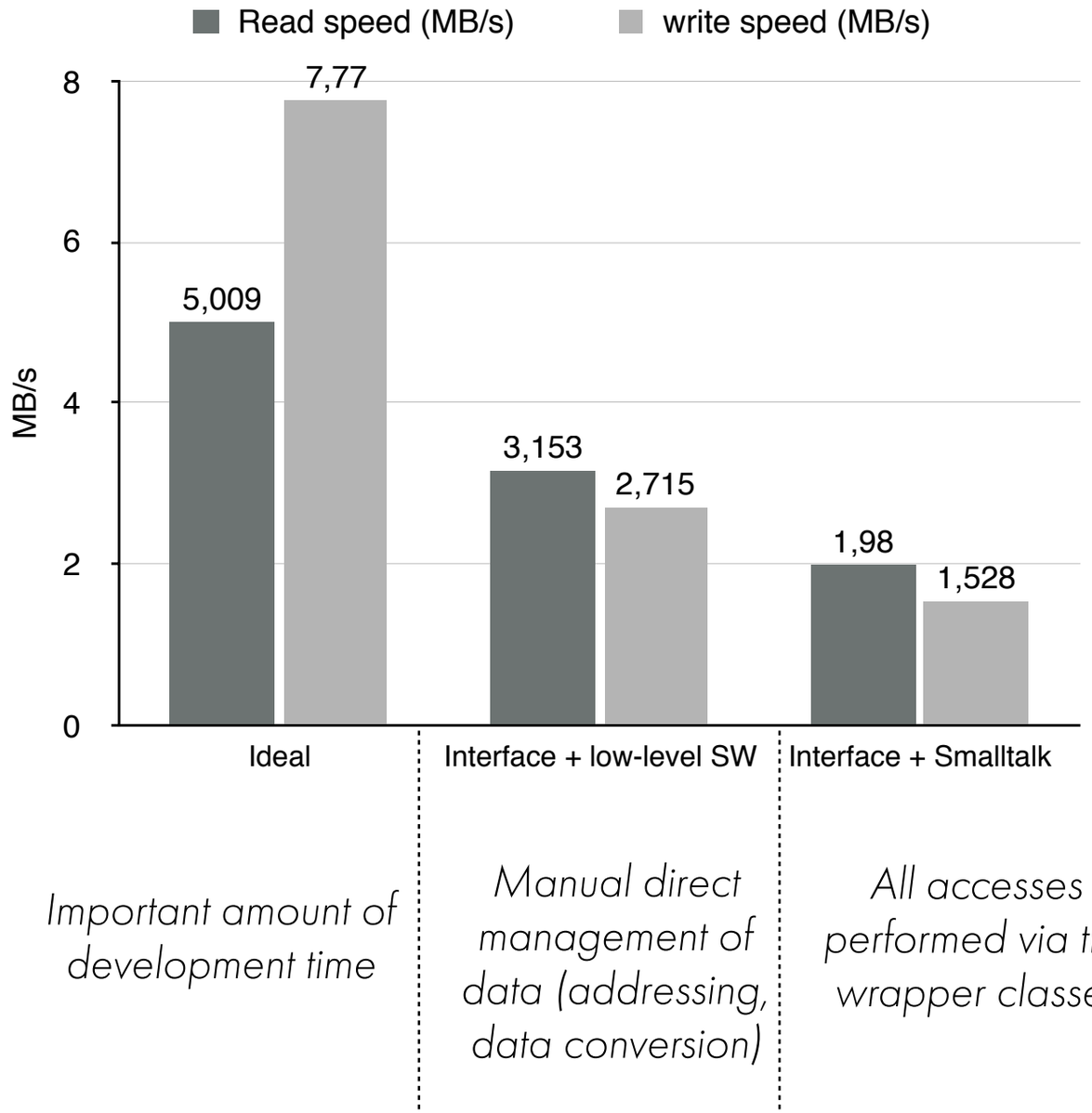



```
cnt := HWCounter new.
cnt input:100.
cnt setBreakpointOn:#output forValue:50 condition:#=.
cnt start:true.
cnt waitForIRQ:[
     cnt bpActive ifTrue:[
          ('Stop at: ', cnt output asString) print.
          ('Steps: ', cnt clockCount asString) print.
          cnt resume:true.
     ] ifFalse:[
          'Execution done' print.
     ]
] timeOut:10 milliseconds.
```

2 Platform for Software/FPGA Integration

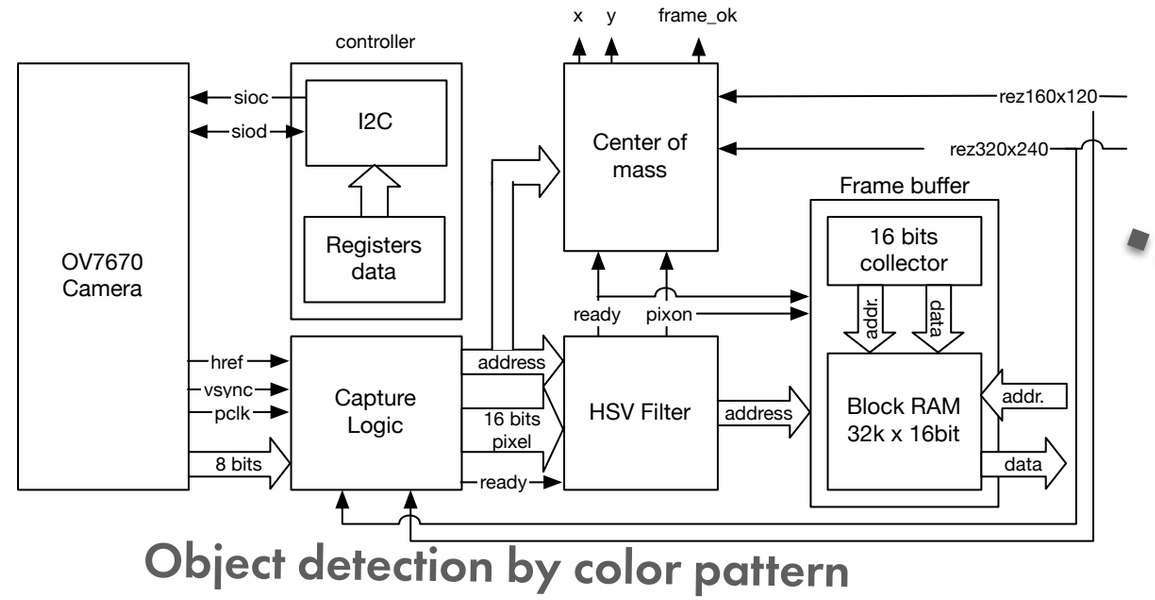
Impact of Different Software Layers to the Performance of the Link

- Experiments : Read/Write to a Single Clock Block Ram. <u>3 scenarios</u>:
  - *Ideal, without our platform*
  - *With the generated interface + our low level software API*
  - *With the generated interface + our high level software API (Smalltalk)*
- Device: APF 51
  - FPGA : Xilinx Spartan 6 (LX9)
  - Processor : Freescale iMX515 (Cortex A8 @ 800 Mhz)
  - Physical Interface: 16 bit WEIM (Wireless External Interface Module)

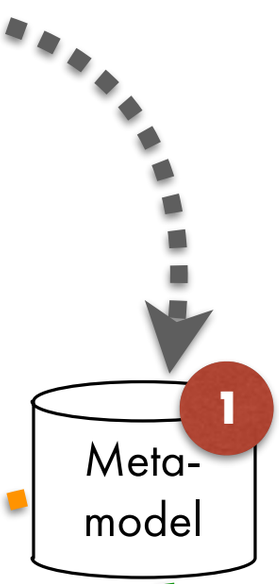


3 Applications

Robot Follower Using Camera for Object Tracking

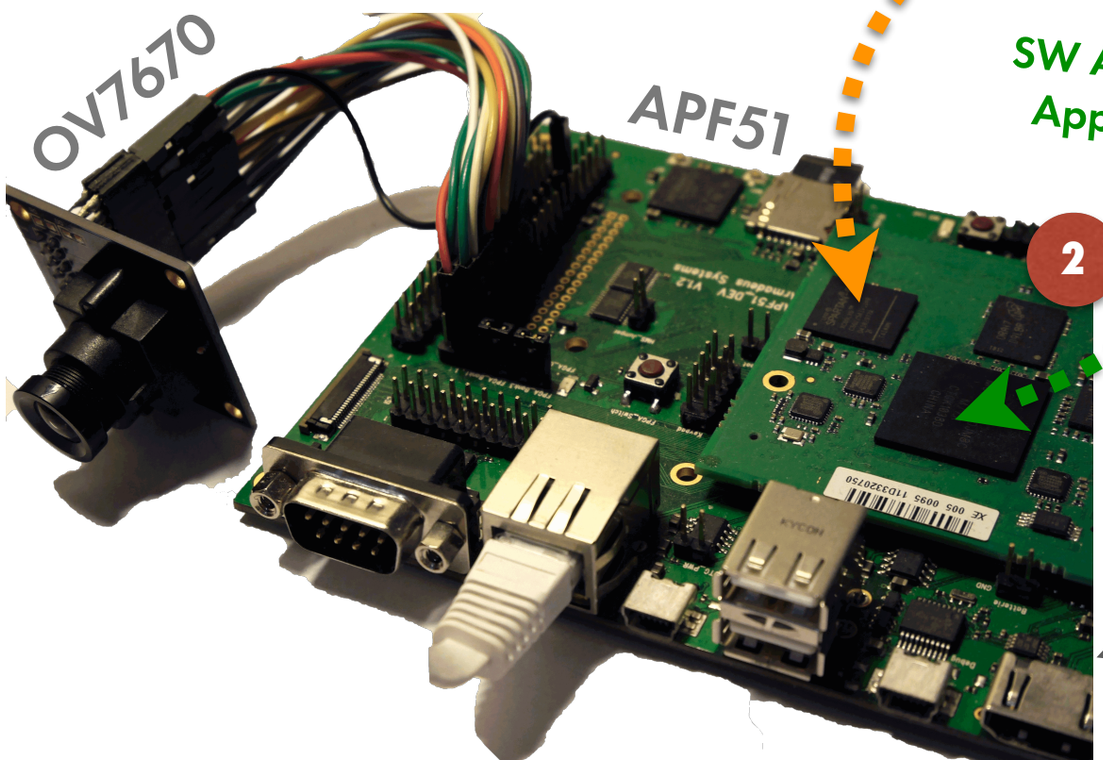


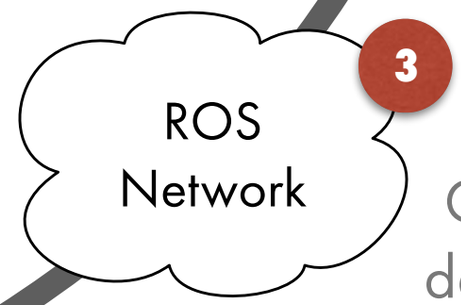
Object detection by color pattern



Interface + Circuit

SW API + user Application





Communicate detected object position (10 lines of code)

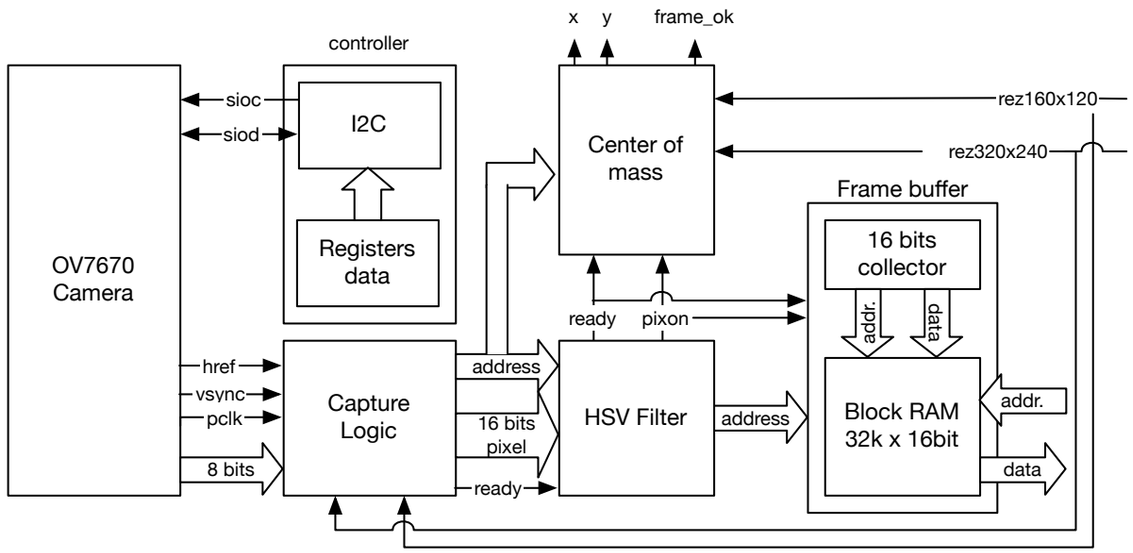
3 Applications

Robot Follower Using Camera for Object Tracking

- Problem with VGA images
  - The original design work only with images QQVGA (160x120)
- Simulation & Debug (VGA)
  - Simulation of **HSV Filter Unit** → *14 cycles/ pixel*
  - **Capture Logic Unit**, hard to simulate

→ Hardware Breakpoint

- Stop the circuit when a line (640 pixels) is captured
- *2556 cycles/ 640 pixels →4 cycles/pixel*



Original design

Optimised design

3 Applications

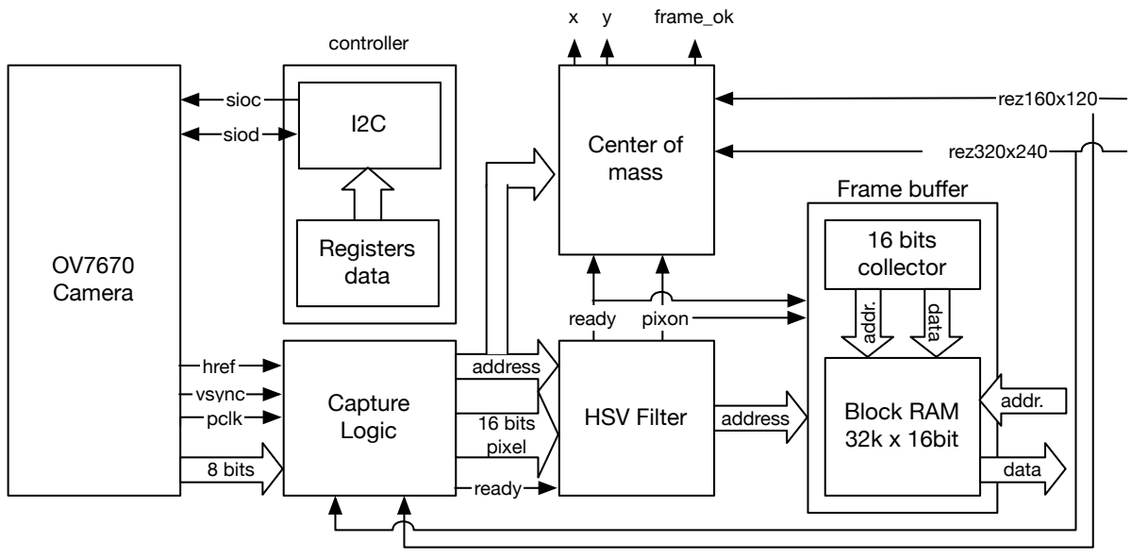
Robot Follower Using Camera for Object Tracking

- Problem with VGA images
  - The original design work only with images QQVGA (160x120)
- Simulation & Debug (VGA)
  - Simulation of **HSV Filter Unit** → *14 cycles/ pixel*
  - **Capture Logic Unit**, hard to simulate

→ Hardware Breakpoint

- Stop the circuit when a line (640 pixels) is captured
- *2556 cycles/ 640 pixels →4 cycles/pixel*

The filter takes more time to process a pixel than the time needed to produce a pixel



Original design

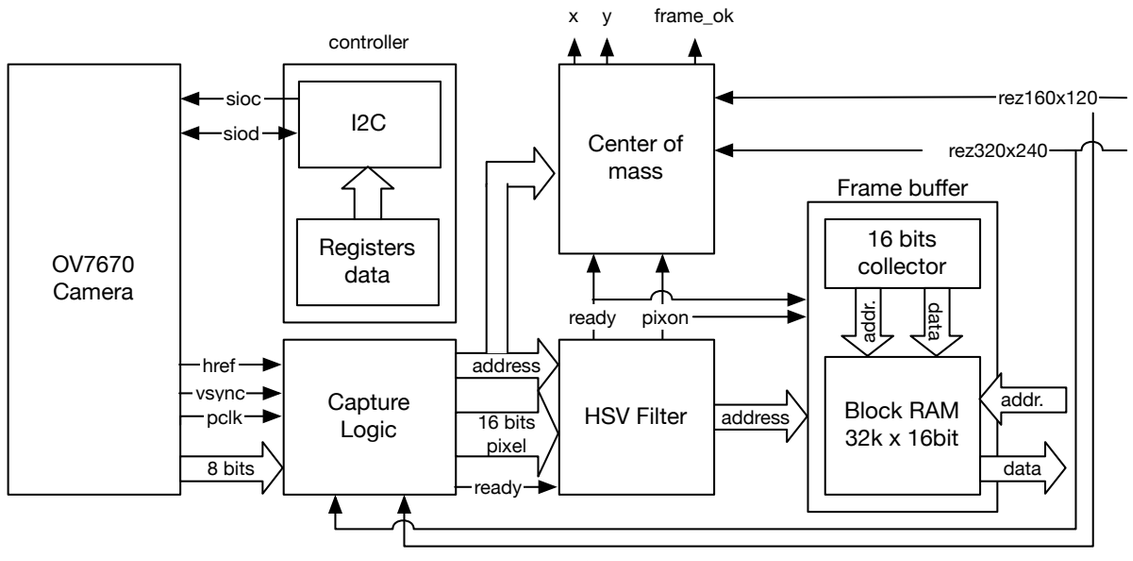
Optimised design

3 Applications

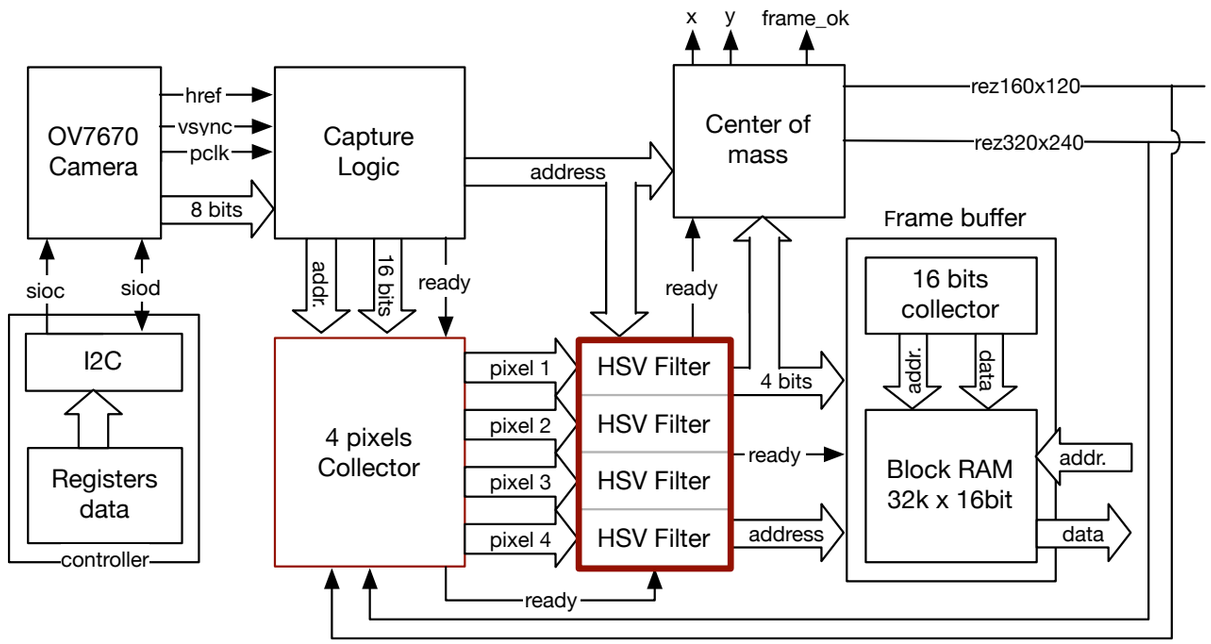
Robot Follower Using Camera for Object Tracking

- Problem with VGA images
  - The original design work only with images QQVGA (160x120)
- Simulation & Debug (VGA)
  - Simulation of **HSV Filter Unit** → *14 cycles/ pixel*
- **Capture Logic Unit**, hard to simulate → Hardware Breakpoint
  - Stop the circuit when a line (640 pixels) is captured
  - *2556 cycles/ 640 pixels →4 cycles/pixel*

The filter takes more time to process a pixel than the time needed to produce a pixel

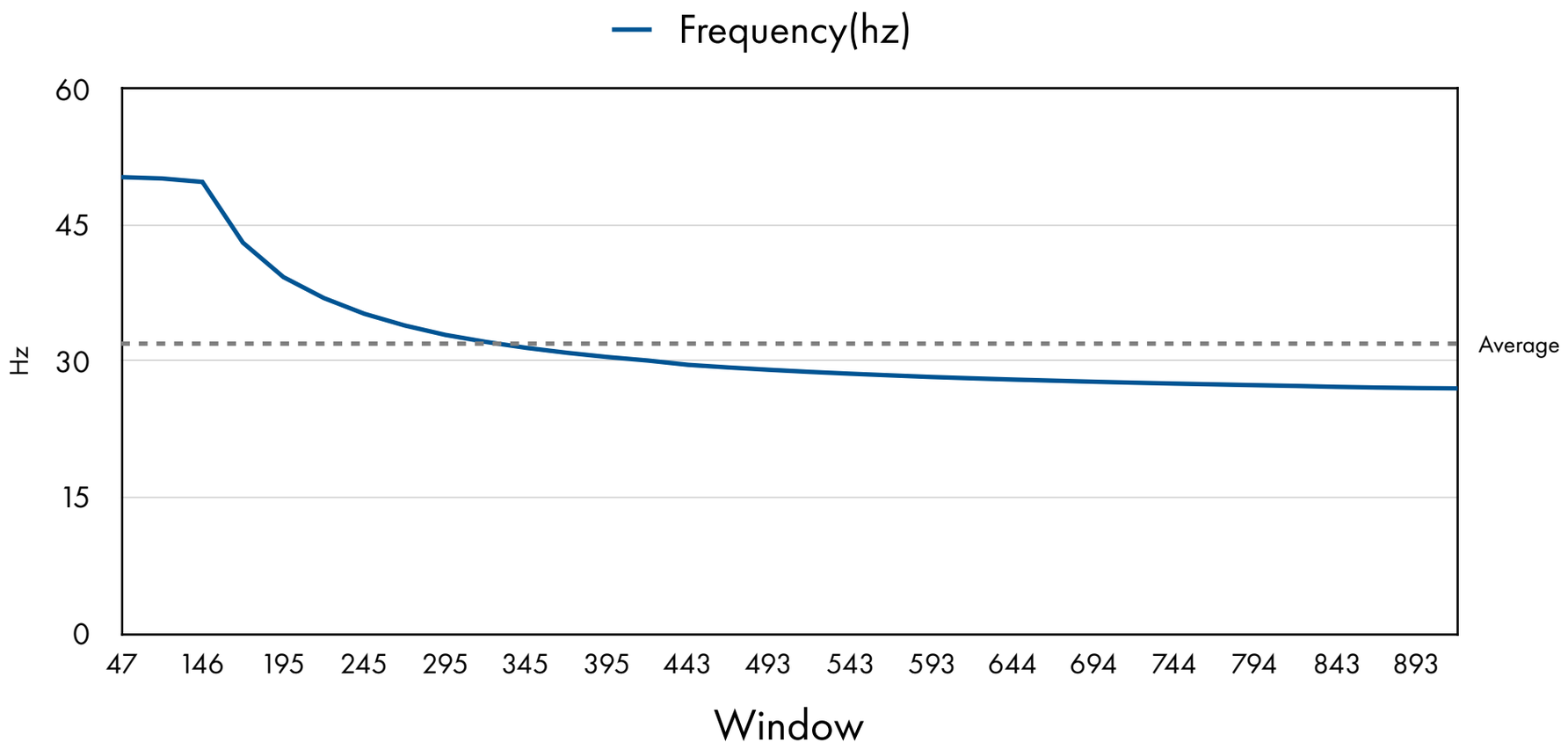


Original design



Optimised design

Robot Follower Using Camera for Object Tracking



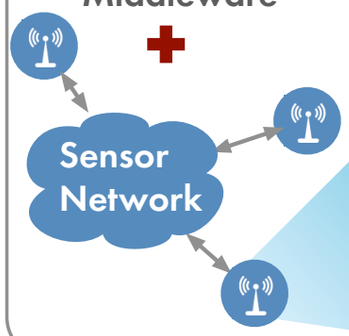
Publishing rate of the ROS topic

3 Applications

Reconfigurable IP-Based Smart Sensor Network

Context

IP-based Middleware



Flexible software env.
Ease of reconfiguration

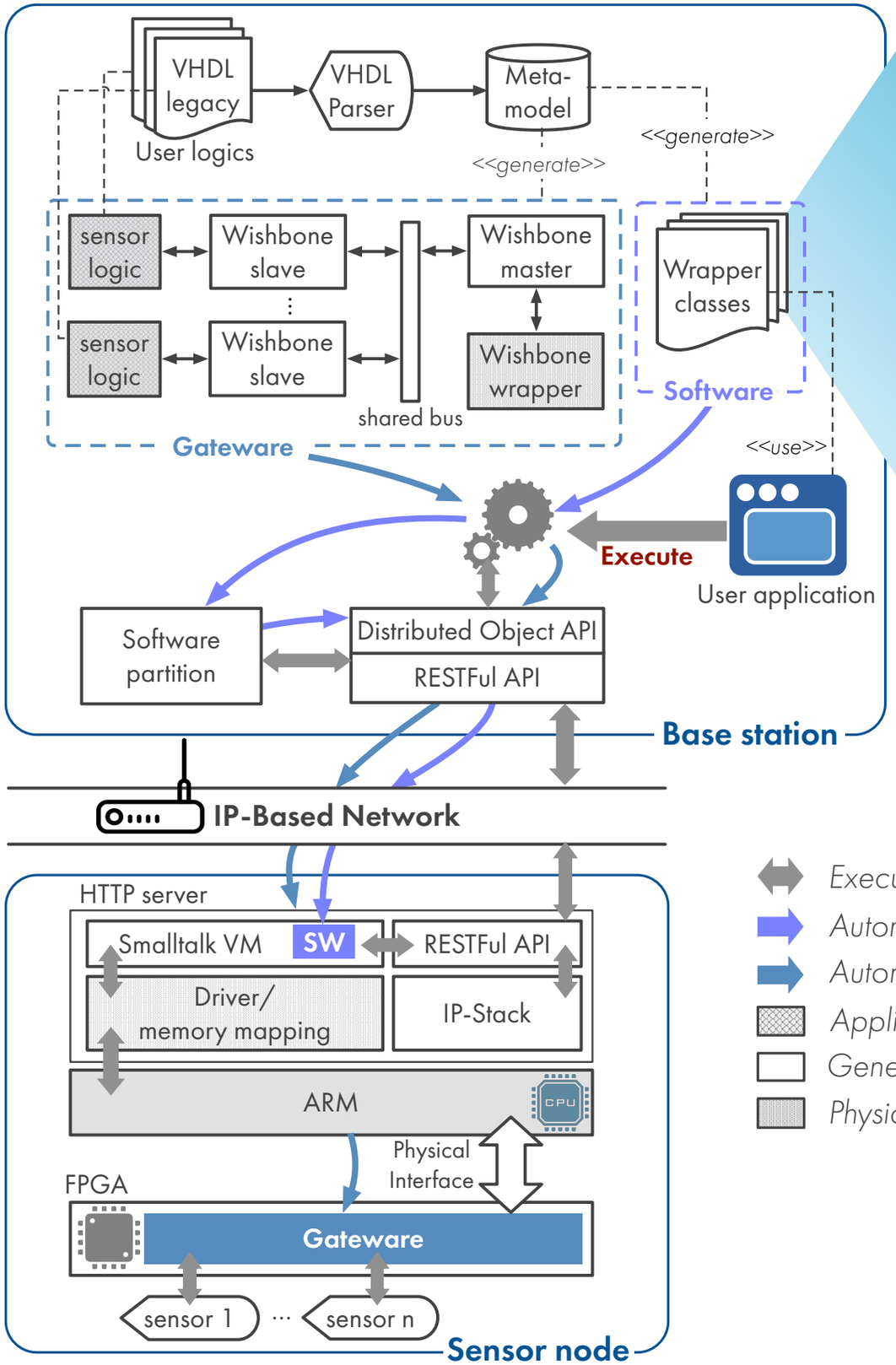
Hardware flexibility
Performance/energy

Challenges

Complex hybrid software/hardware :

- Development/ reconfiguration ?
- Deployement ?

- IP based Sensor Network:
  - Integration of FPGA to the nodes
  - Compliant with IoT Standard
  - Distributed Objected programming on sensor nodes using a dynamic language
  - Over-the-air reconfiguration of HW/SW on the nodes




```
CameraUnitWrapper » x
  <remote>
  ^self int32At:24

CameraUnitWrapper » y
  <remote>
  ^self int32At:28

CameraUnitWrapper » position
  <remote>
  ^ { self x. self y}

CameraUnitWrapper » positionDo:aBlock
  <remote:#aBlock>
  500 timesRepeat:[
    aBlock value: self position.
    100 milliseconds wait]

CameraUnitWrapper » stream
  self positionDo:[:p| p print]
```

- *Execution flow*
- *Automatic software reconfiguration*
- *Automatic gateware reconfiguration*
- *Application specific*
- *Generic or automatically generated*
- *Physical interface specific/partial reusable*

3 Applications

Reconfigurable IP-Based Smart Sensor Network

Proof of concept of the hybrid system :

- Base station: Implemented using Pharo Smalltalk
- Sensor node:
  - ARM: Httpd + Smalltalk VM + REST API
  - FPGA + camera: object tracking using a color pattern
- The base station fetches the object position from the node

S.W Memory footprint on a node (KB)

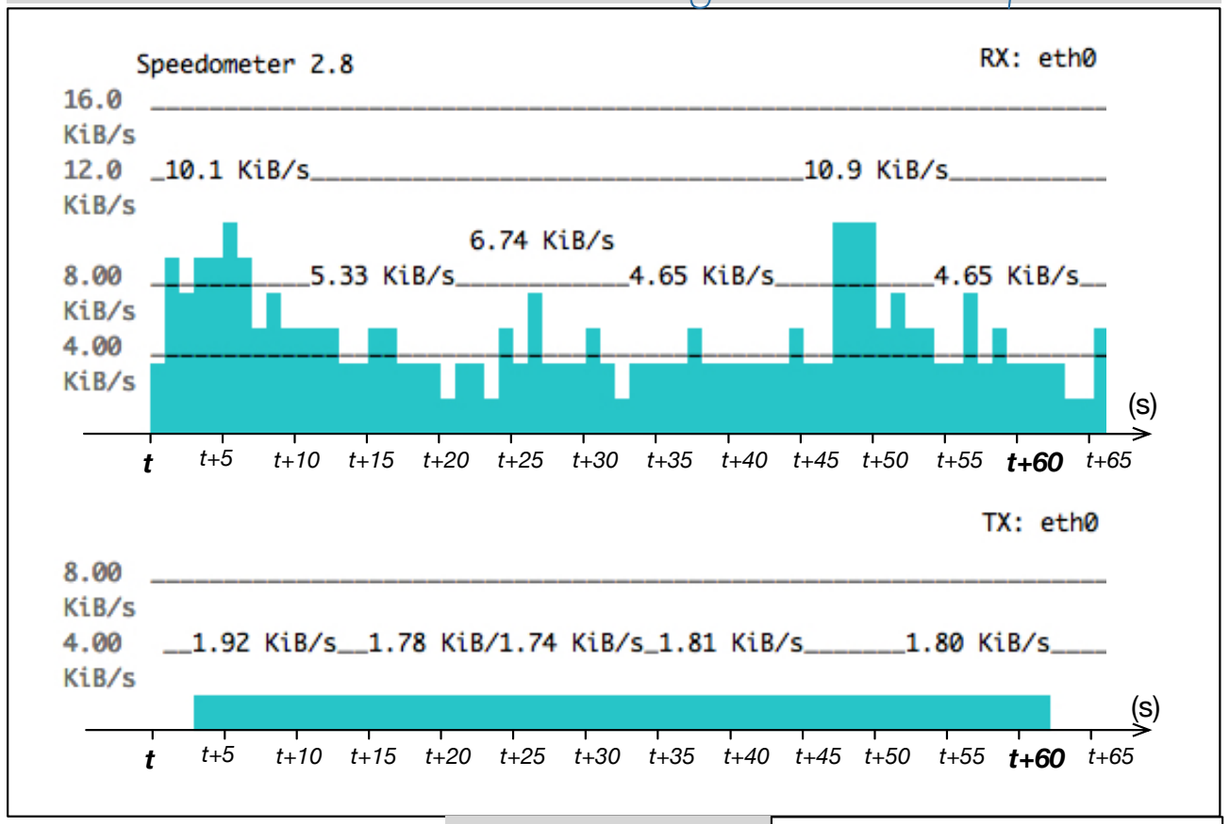
Module	RSS[1]	Shared Memory
HTTPD	640	544
REST+VM	532	80

[1] *Residence Set Size*

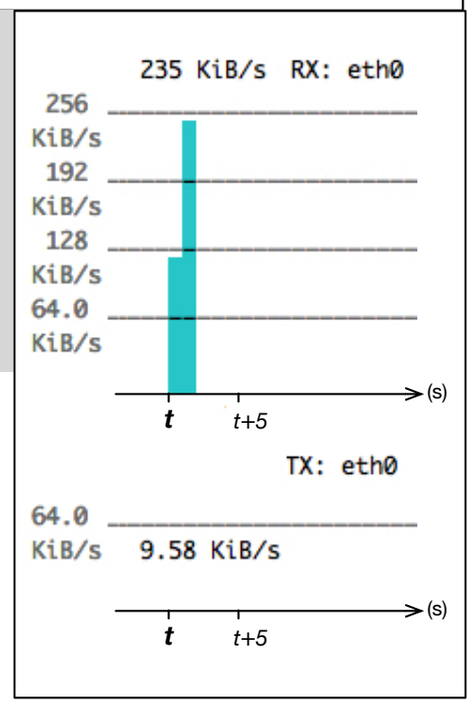
Resource used on a sensor node at different stages

Resource	Idle	SW. reconfig	GW. reconfig.	streaming
% Memory	0.5	0.5	0.5	0.5
% CPU	0	4.7	26.2	5.3

Network load[2] of streamming 500 data samples



Network load[2] of the software/ gateware reconfiguration



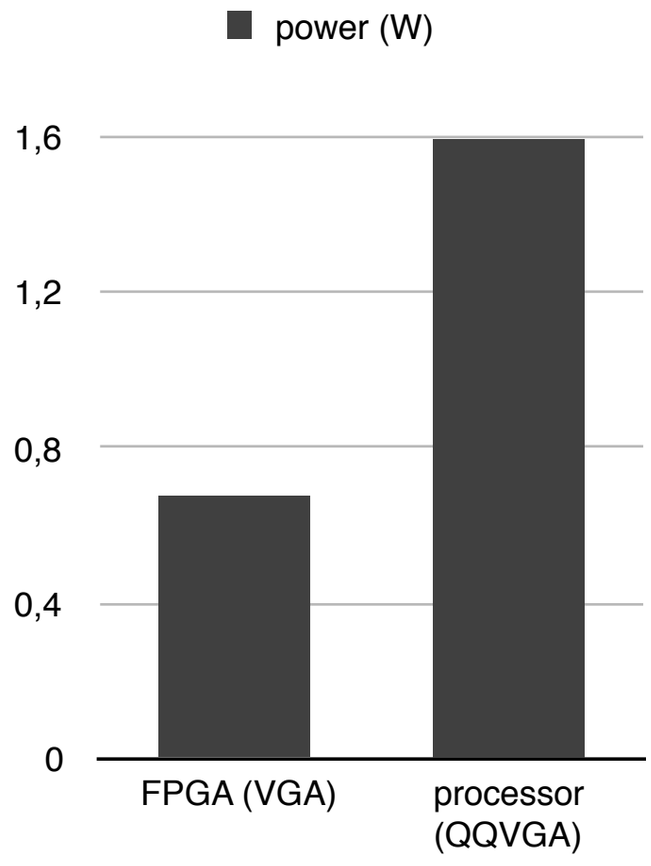
4 Conclusion

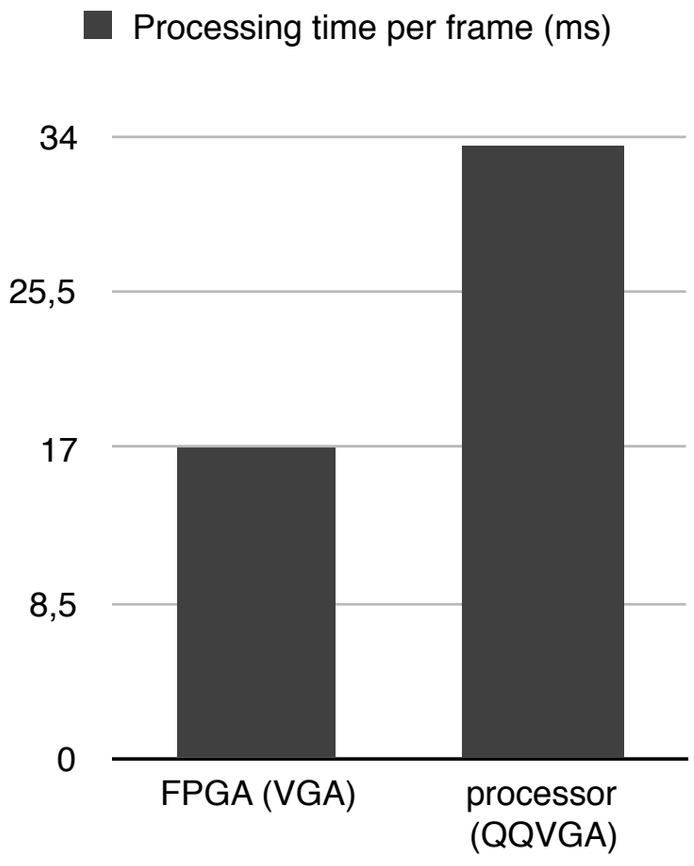
- A platform for interfacing FPGA to High level robotic software
- Easy integration of legacy circuit design using an auto-generation code approach
- Debugging using hardware breakpoint
- Use cases
- Futur works
  - Improvement of the meta-model
  - Support more robotic middleware in our software API

4 Annexe

FPGA vs Processor

- Device: APF 51
- Scenario 1:
  - FPGA used to acquire image
  - The object detection algorithm is implemented in C (on Processor)
  - Image size QQVGA
- Scenario 2:
  - The object detection algorithm is implemented on FPGA
  - FPGA communicates object position with processor
  - Image size VGA





4 Annexe

- The VHDL Parser is tested again some set of VHDL Benchmark
  - The ANTLR set: standard VHDL Package
  - IWLS 2005 Benchmarks
  - ITC'99 Benchmarks
  - Gaisler Research Benchmarks