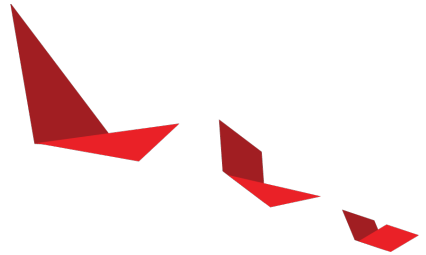
XILINX

ROS 2

Hardware Acceleration Working Group

Meeting #2: ROS 2 architecture for hardware acceleration, use case and ReconROS

Agenda



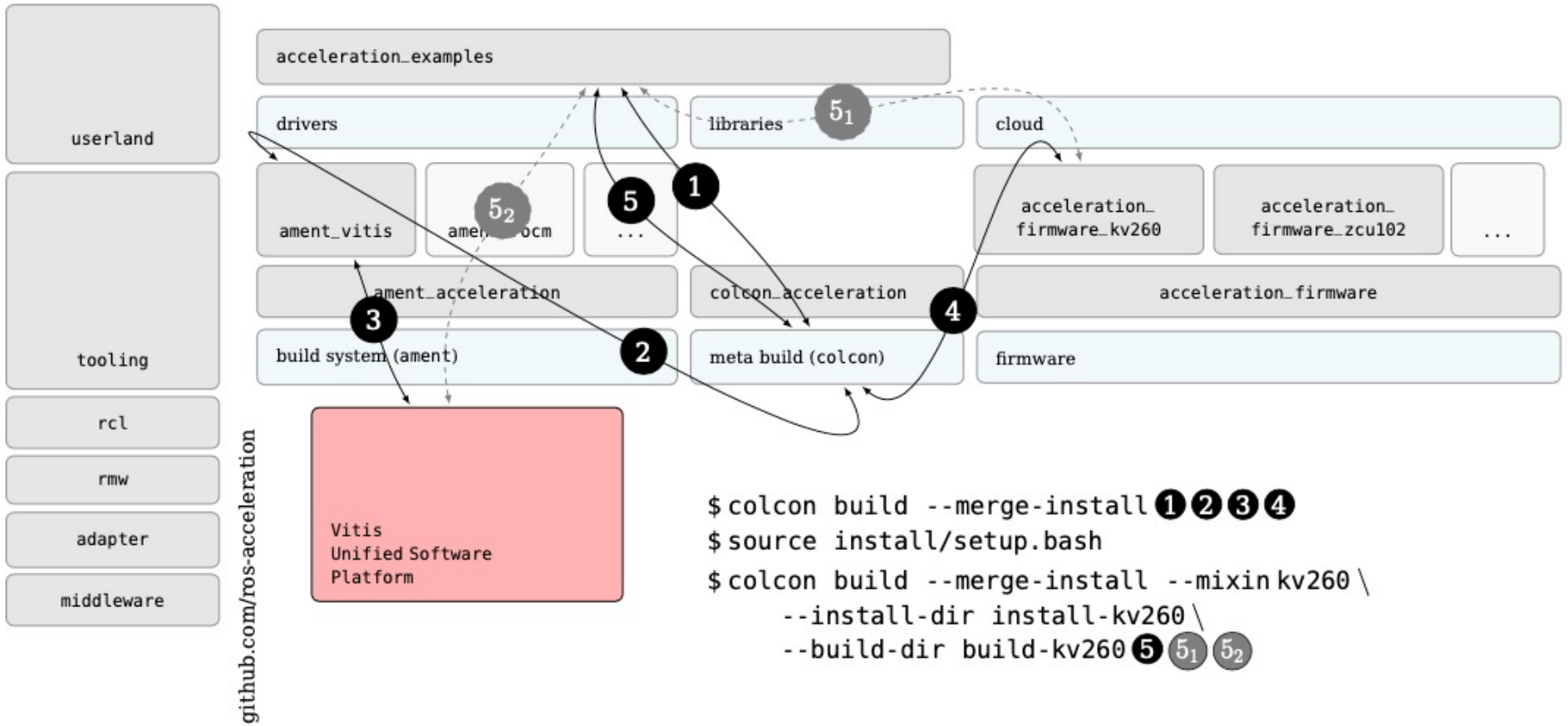
- Welcome and newcomers' intro
- Recap of ROS 2 Hardware Acceleration WG progress
- Adding new boards to the HAWG architecture progress
- Use case
- ReconROS (invited talk)
- Q&A

Recap of ROS 2 Hardware Acceleration WG
resources

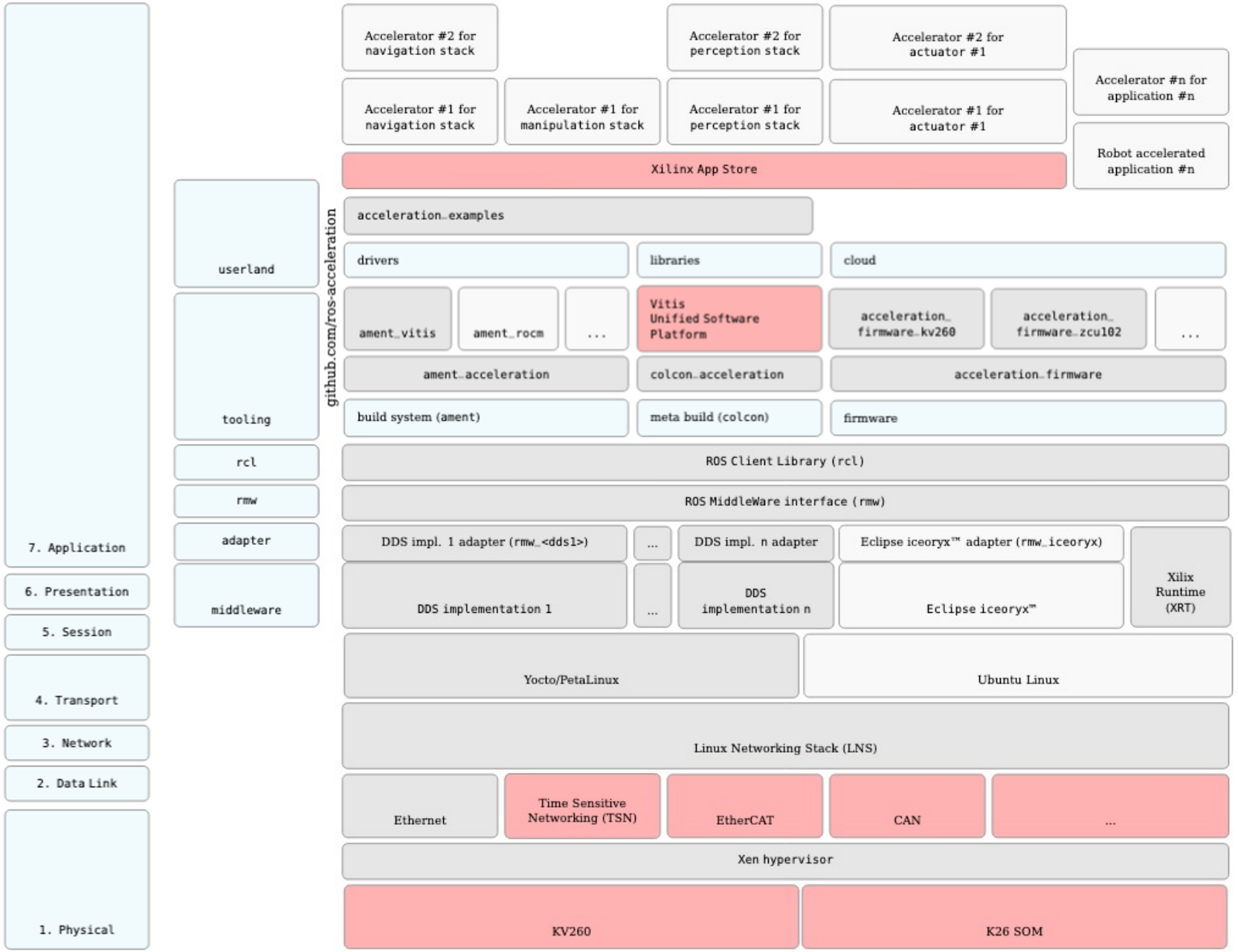
Objective: Drive creation, maintenance and testing of acceleration kernels on top of open standards (C++ and OpenCL) for optimized ROS 2 and Gazebo interactions over different compute substrates (including FPGAs, GPUs and ASICs).

- **Reference hardware platform**: Kria K26 Adaptive SOM
- **Meeting invite group**: ROS 2 Hardware Acceleration WG Google Group
- **Instant messaging**: Matrix community (Matrix is an open network for secure, decentralized communication).
- **Backlog management**: Phase 1: tools, examples, benchmarking and first demonstrators
- **Github organization**: ros-acceleration
- Meeting minutes: HAWG minutes
- **Discourse tag**: wg-acceleration

Recap of ROS 2 Hardware Acceleration WG architecture



For more on architecture:
https://github.com/ros-infrastructure/rep/pull/324

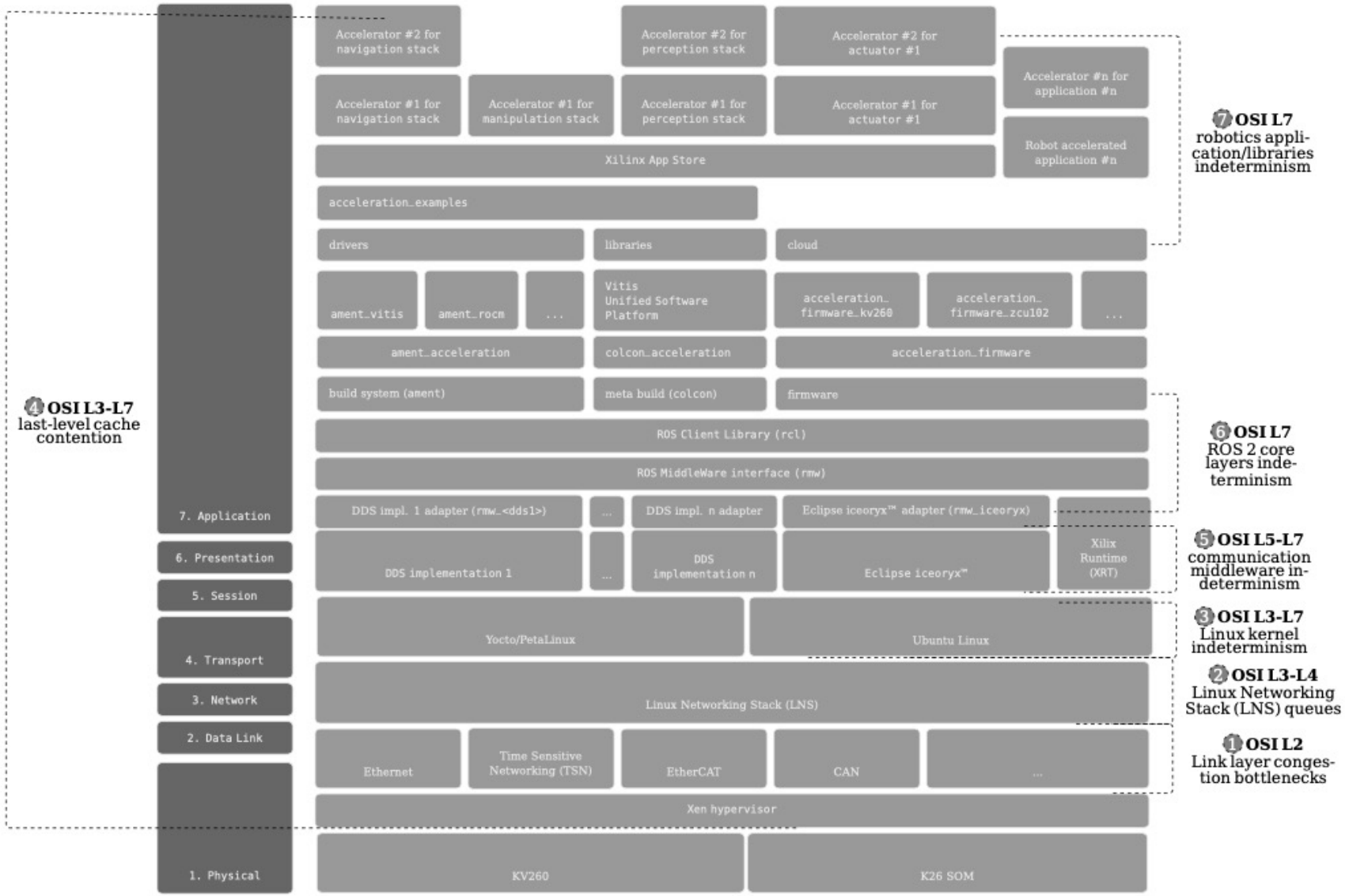


Recap of ROS 2 Hardware Acceleration WG architecture and beyond

XILINX.

Delivering real-time ROS 2

An end-to-end approach from the silicon layer to mitigate all sources of indeterminism across the ROS robotics stack.

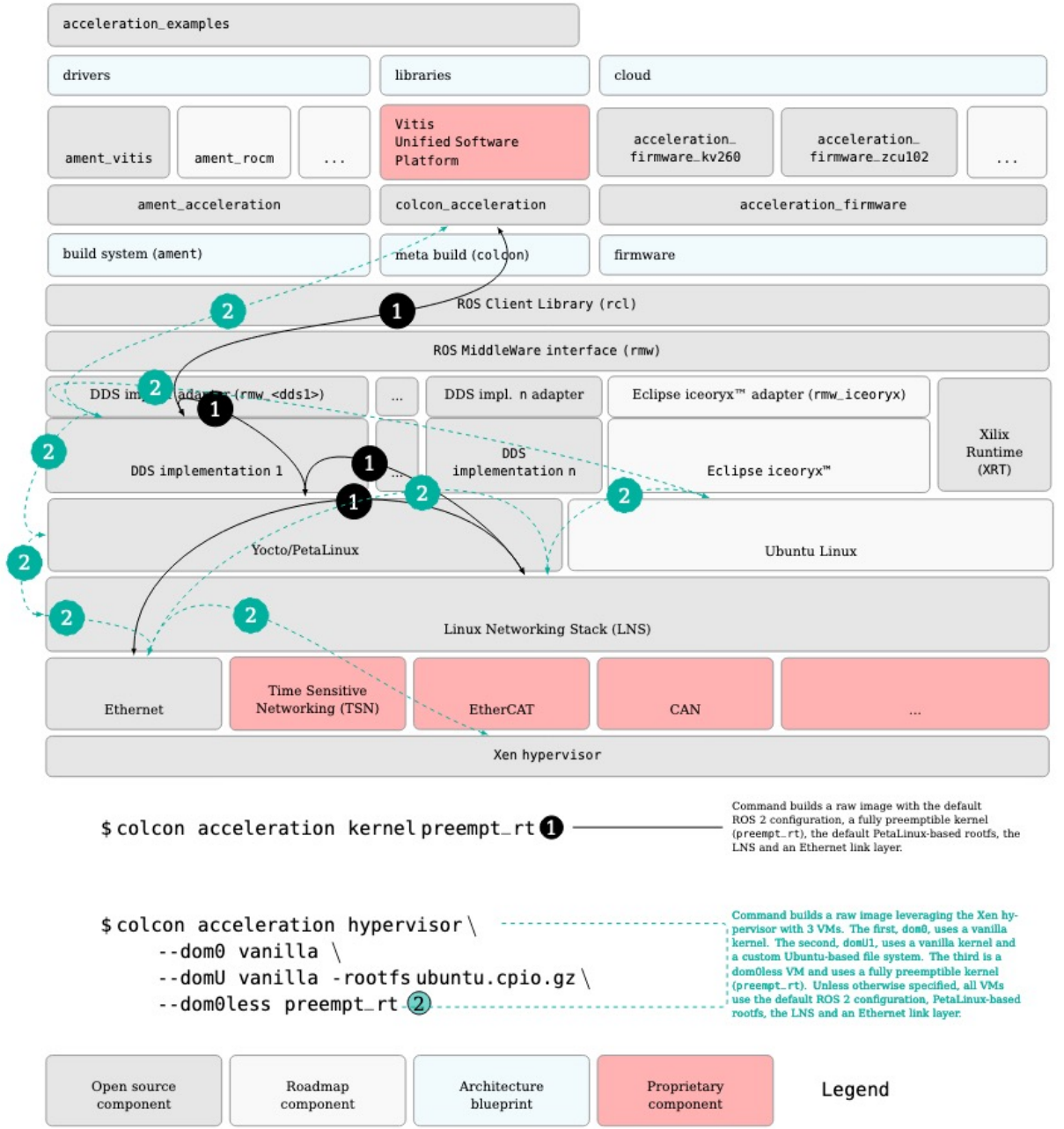


XILINX.

Xilinx facilitates creating ROS 2 mixed critical systems

❶ Command builds a raw image with the default ROS 2 configuration, a fully preemptible kernel (preempt_rt), the default PetaLinux-based rootfs, the LNS and an Ethernet link layer.

❷ Command builds a raw image leveraging the Xen hypervisor with 3 VMs. The first, dom0, uses a vanilla kernel. The second, domU1, uses a vanilla kernel and a custom Ubuntu-based file system. The third is a dom0less VM and uses a fully preemptible kernel (preempt_rt). Unless otherwise specified, all VMs use the default ROS 2 configuration, PetaLinux-based rootfs, the LNS and an Ethernet link layer.



XILINX.

Kria Robotics Stack

Bringing ROS 2 down to silicon

Empowering Hardware Acceleration and next-gen “robot chips”

XILINX

Kria Robotics Stack (KRS)

Bringing ROS 2 down to silicon

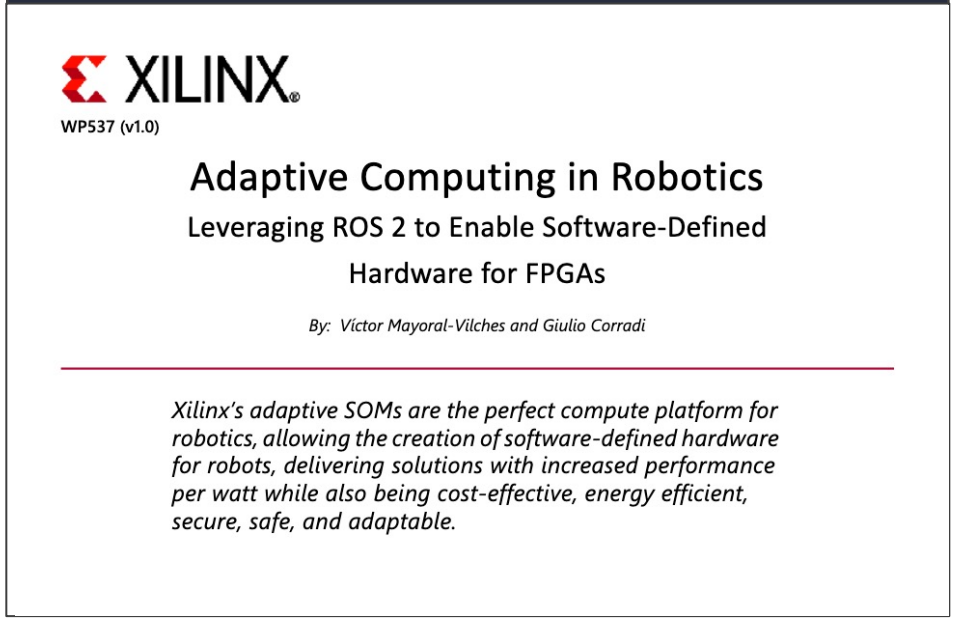
Empowering Hardware Acceleration and next-gen “robot chips”


```
# 0. install Vitis 2020.2.2 and ROS Foxy

# 1. install some dependencies you might be missing
sudo apt-get -y install curl build-essential libssl-dev git wget
                \ ocl-icd-* opencl-headers python3-vcstool
                \ python3-colcon-common-extensions python3-colcon-mixin
                \ kpartx u-boot-tools

# 2. create a new ROS 2 workspace
mkdir -p ~/krs_ws/src; cd ~/krs_ws

# 3. KRS alpha release
cat << 'EOF' > krs_alpha.repos
repositories:
  acceleration/acceleration_firmware:
                type: git
                url: https://github.com/ros-acceleration/acceleration_firmware
                version: 0.4.0
  acceleration/acceleration_firmware_kv260:
                type: zip
                url: https://github.com/ros-acceleration/acceleration_firmware_kv260
                                /releases/download/v0.6.0/acceleration_firmware_kv260.zip
  acceleration/colcon-acceleration:
                type: git
                url: https://github.com/ros-acceleration/colcon-acceleration
                version: 0.1.0
  acceleration/ros2acceleration:
                type: git
                url: https://github.com/ros-acceleration/ros2acceleration
                version: 0.2.0
  acceleration/ament_vitis:
                type: git
                url: https://github.com/ros-acceleration/ament_vitis
                version: 0.5.0
EOF

# 4. import repos of KRS alpha release
vcs import src --recursive < krs_alpha.repos # about 5 mins

# 5. build the workspace and deploy firmware for hardware acceleration
source /tools/Xilinx/Vitis/2020.2/settings64.sh ; source /opt/ros/foxy/setup.bash \
                                ; export PATH="/usr/bin":$PATH
colcon build --merge-install # about 2 mins

# 6. source the overlay to enable all  hardware acceleration features
source install/setup.bash
```


XILINX.

Learn more about KRS

Adaptive Computing in Robotics


XILINX

WP537 (v1.0)

Adaptive Computing in Robotics

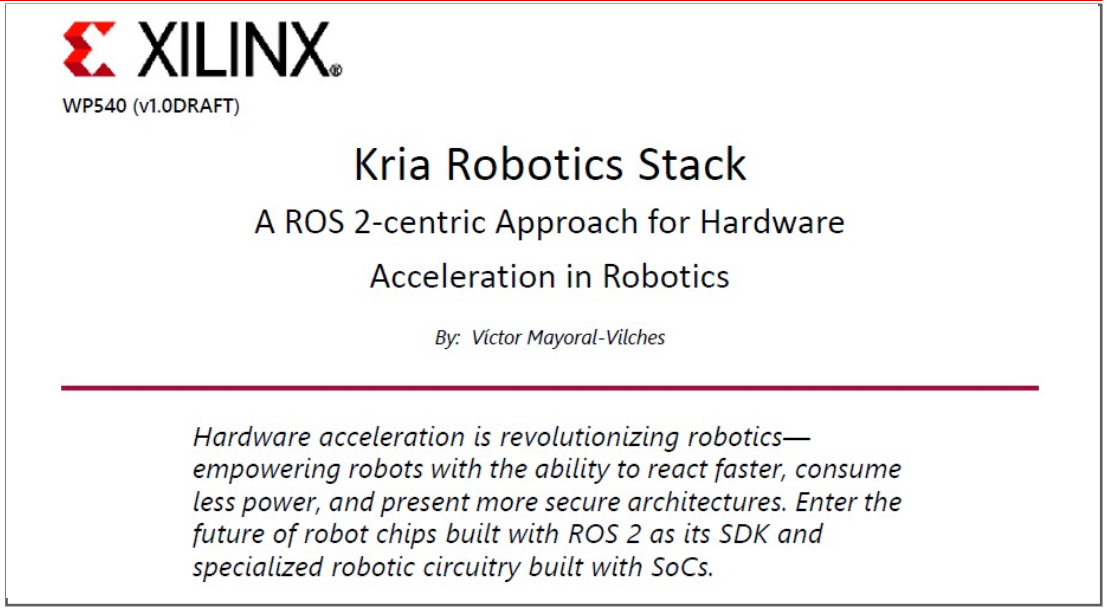
Leveraging ROS 2 to Enable Software-Defined Hardware for FPGAs

By: Víctor Mayoral-Vilches and Giulio Corradi

Xilinx's adaptive SOMs are the perfect compute platform for robotics, allowing the creation of software-defined hardware for robots, delivering solutions with increased performance per watt while also being cost-effective, energy efficient, secure, safe, and adaptable.

xilinx.com/krs

The Kria Robotics Stack (KRS)


XILINX

WP540 (v1.0DRAFT)

Kria Robotics Stack

A ROS 2-centric Approach for Hardware Acceleration in Robotics

By: Víctor Mayoral-Vilches

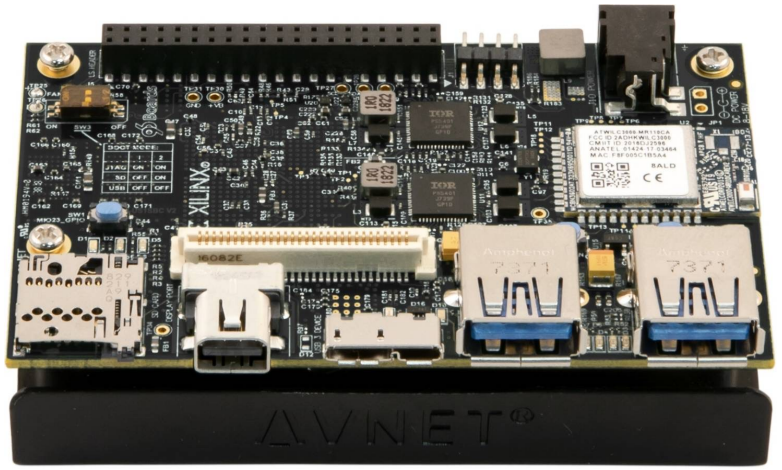
Hardware acceleration is revolutionizing robotics—empowering robots with the ability to react faster, consume less power, and present more secure architectures. Enter the future of robot chips built with ROS 2 as its SDK and specialized robotic circuitry built with SoCs.

xilinx.com/krs

Access KRS alpha release **here** (documentation).
EA of KRS beta

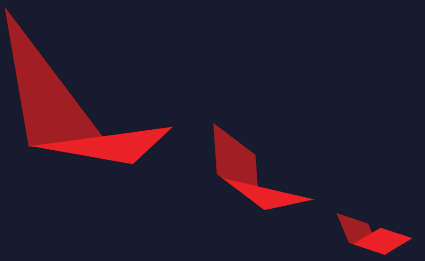
Adding new boards to the HAWG – process

1. Check out the Initial draft of REP-2008 - ROS 2 Hardware Acceleration Architecture and Conventions
2. Create your own firmware repository for the corresponding board (see acceleration_firmware_kv260 for an example)
3. Assess the capabilities of the hardware according to REP-2008 and create a table in the README.md that argues about it (see example here)
4. Submit a PR to ros-acceleration/community to add your board to the community, with the corresponding support level according to REP-2008



Ultra96-v2

Led by @pimartos, check it out at
https://github.com/ros-acceleration/community/issues/1



ROS 2 hardware acceleration use case

robotics architect creating an accelerator for a faster ROS 2 publisher

Accelerating ROS 2

ROS 2 Hardware acceleration "hello world" – doublevadd_publisher[1]

1 https://github.com/ros-acceleration/acceleration_examples/tree/main/doublevadd_publisher

```
int main(int argc, char * argv[]) {
 rclcpp::init(argc, argv);
 auto node = rclcpp::Node::make_shared("doublevadd_publisher");
 auto publisher = node->create_publisher<
     std_msgs::msg::String>("vector", 10);
 auto publish_count = 0;
 std_msgs::msg::String message;
 rclcpp::WallRate loop_rate(100ms);          Goal: 10 Hz
 ...
 while (rclcpp::ok()) {
  ...
  vadd(in1, in2, out, DATA_SIZE);
  check_vadd(in1, in2, out);
  ...
  try {
   publisher->publish(message);
   rclcpp::spin_some(node);
  } catch (const rclcpp::exceptions::RCLError & e) {
   RCLCPP_ERROR(
    node->get_logger(),
    "unexpectedly failed with %s",
    e.what());
  }
  loop_rate.sleep();
 }
 rclcpp::shutdown();
 return 0;
}
```

Leo "the roboticist"

```
void vadd(
      const unsigned int *in1,   // Read-Only Vector 1
      const unsigned int *in2,   // Read-Only Vector 2
      unsigned int *out,         // Output Result
      int size                   // Size in integer
      )
{
   for (int j = 0; j < size; ++j) {  // stupidly iterate
                                     // to generate "double" load
      for (int i = 0; i < size; ++i) {
        out[i] = in1[i] + in2[i];
      }
   }
}
```

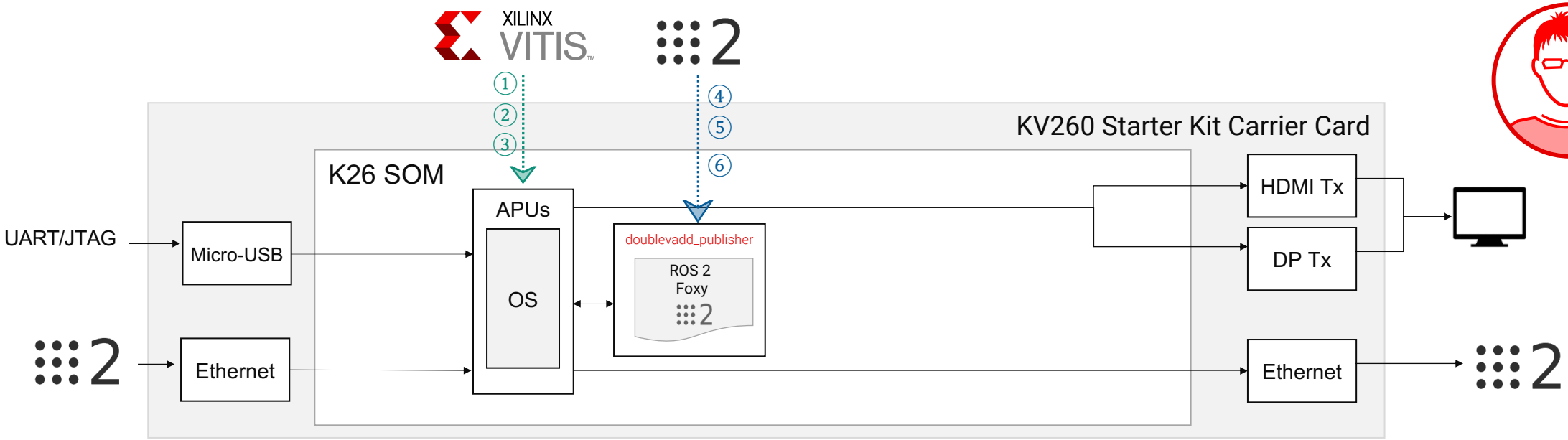
Accelerating ROS 2

ROS 2 Hardware acceleration "hello world" – doublevadd_publisher[1]

1 https://github.com/ros-acceleration/acceleration_examples/tree/main/doublevadd_publisher

- ① Fetch BSP, ② include ROS 2 Yocto recipes, ③ and build rootfs
- Build ROS 2 doublevadd_publisher package:
  - **Cross-compile ROS 2 workspace**: ④ Set up cross-compilation toolchain, ⑤ sysroot, and ⑥ build/cross-compile ROS 2 workspace
  - **Yocto recipe**: ④ Create Yocto recipe for doublevadd_publisher, build package
  - **Native build**: ④ Integrate in rootfs build tools, ⑤ re-build rootfs, and ⑥ build doublevadd_publisher natively.

Leo "the roboticist"



XILINX

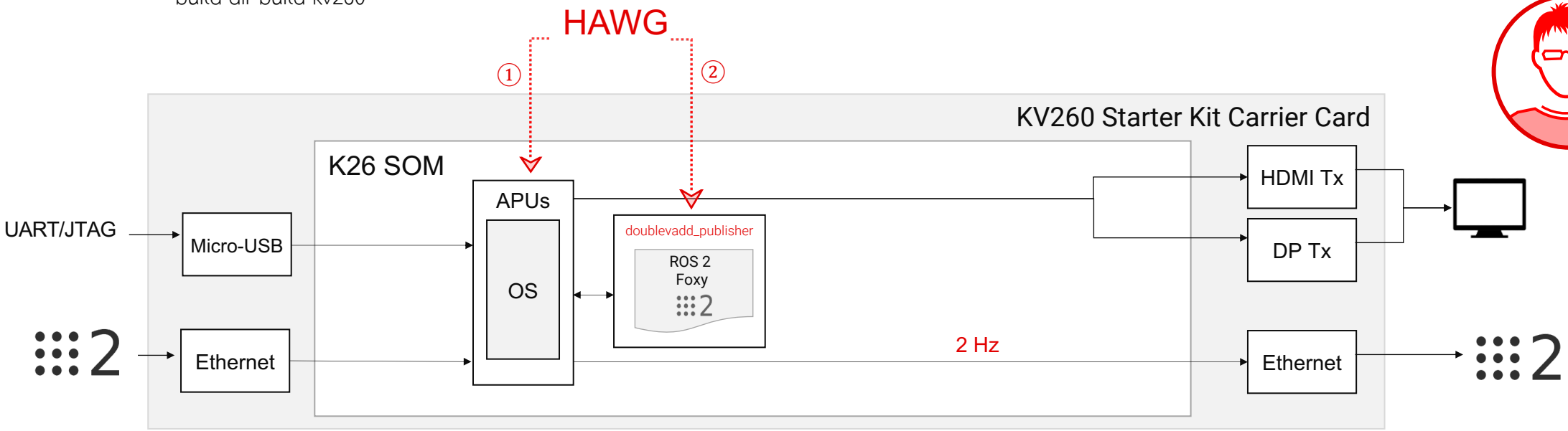
Accelerating ROS 2 – with HAWG

ROS 2 Hardware acceleration "hello world" – doublevadd_publisher[1]

1 https://github.com/ros-acceleration/acceleration_examples/tree/main/doublevadd_publisher

```
# with KRS
$ colcon acceleration select kv260                  # select KV260 firmware
$ colcon acceleration linux vanilla                 # ① create rootfs with ROS 2
$ colcon build --merge-install                      # build KRS ROS 2 packages
$ source install/setup.bash                         # apply the overlay
$ colcon build --merge-install --mixin kv260  \     # ② cross-compile doublevadd_publisher
                --install-dir install-kv260 \
                --build-dir build-kv260
```

Leo "the roboticist"



XILINX

Accelerating ROS 2 – with HAWG

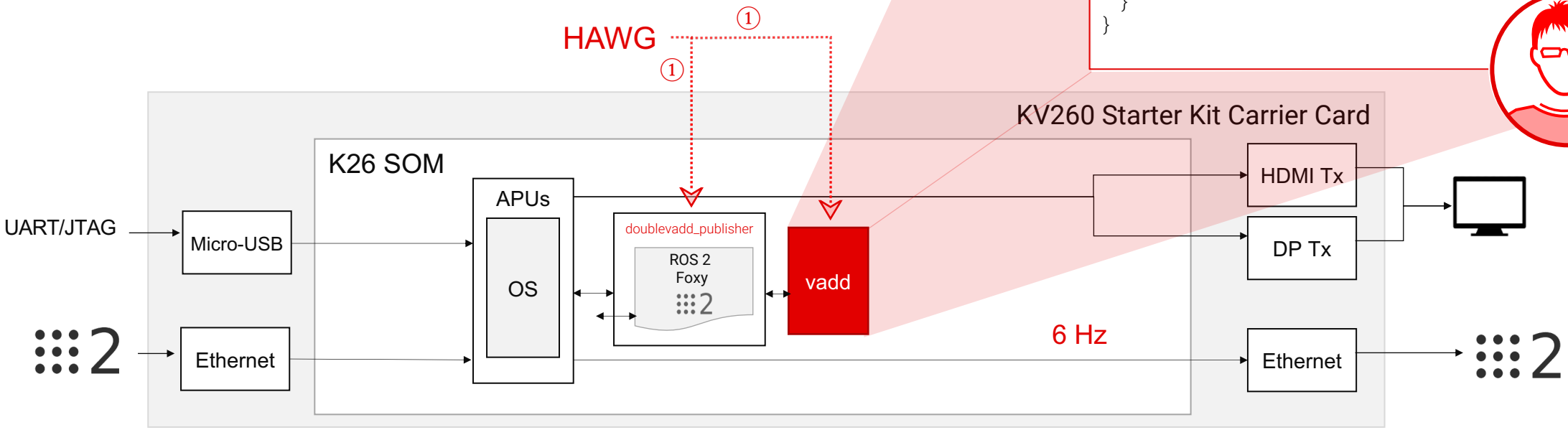
ROS 2 Hardware acceleration "hello world" – accelerated_doublevadd_publisher[1]

1 https://github.com/ros-acceleration/acceleration_examples/tree/main/accelerated_doublevadd_publisher

```
# rebuild ① doublevadd_publisher, including kernels
$ colcon build --merge-install --mixin kv260  \
               --install-dir install-kv260 \
               --build-dir build-kv260
```

```
void vadd(
      const unsigned int *in1,   // Read-Only Vector 1
      const unsigned int *in2,   // Read-Only Vector 2
      unsigned int *out,         // Output Result
      int size                   // Size in integer
      )
{

#pragma HLS INTERFACE m_axi port=in1 bundle=aximm1
#pragma HLS INTERFACE m_axi port=in2 bundle=aximm2
#pragma HLS INTERFACE m_axi port=out bundle=aximm1

   for (int j = 0; j < size; ++j) {  // stupidly iterate
                                     // to generate "double" load
      for (int i = 0; i < size; ++i) {
        out[i] = in1[i] + in2[i];
      }
   }
}
```

Leo "the roboticist"



XILINX.

Accelerating ROS 2 – with HAWG

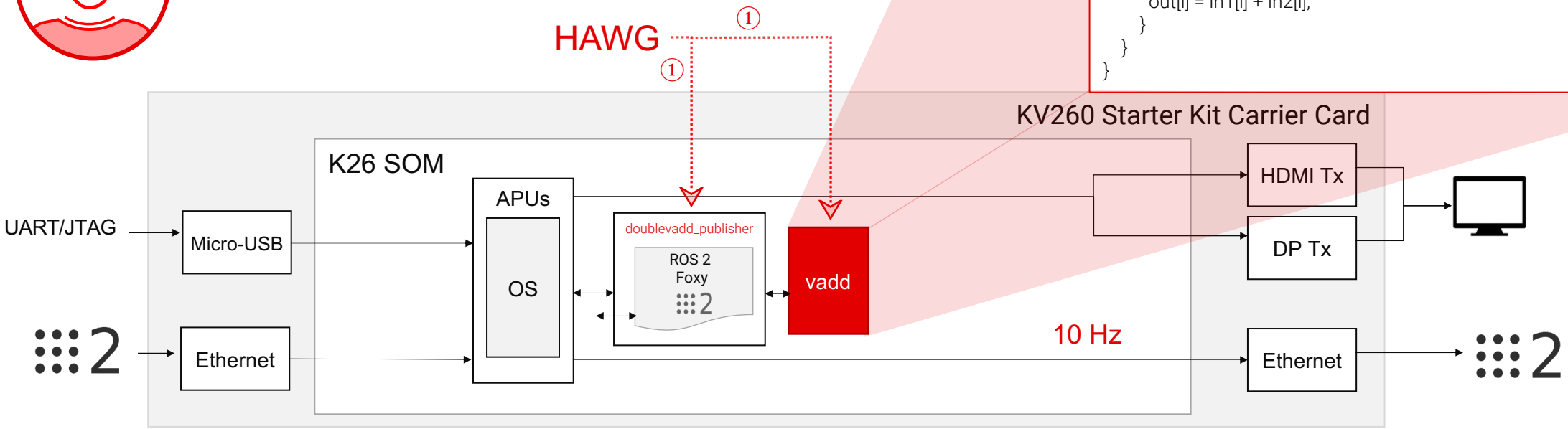
ROS 2 Hardware acceleration "hello world" – faster_doublevadd_publisher[1]

1 https://github.com/ros-acceleration/acceleration_examples/tree/main/faster_doublevadd_publisher

Leo "the roboticist"

```
# rebuild ① doublevadd_publisher, including kernels
$ colcon build --merge-install --mixin kv260  \
               --install-dir install-kv260 \
               --build-dir build-kv260
```

```
void vadd(
      const unsigned int *in1,   // Read-Only Vector 1
      const unsigned int *in2,   // Read-Only Vector 2
      unsigned int *out,         // Output Result
      int size                   // Size in integer
      )
{

#pragma HLS INTERFACE m_axi port=in1 bundle=aximm1
#pragma HLS INTERFACE m_axi port=in2 bundle=aximm2
#pragma HLS INTERFACE m_axi port=out bundle=aximm1

   for (int j = 0; j < size; ++j) {  // stupidly iterate
                                     // to generate "double" load
      for (int i = 0; i < size; ++i) {
      for (int i = 0; i<(size/16)*16; ++i) {
#pragma HLS UNROLL factor=16
        out[i] = in1[i] + in2[i];
      }
   }
}
```



Faster ROS 2 interactions with KRS

XILINX.

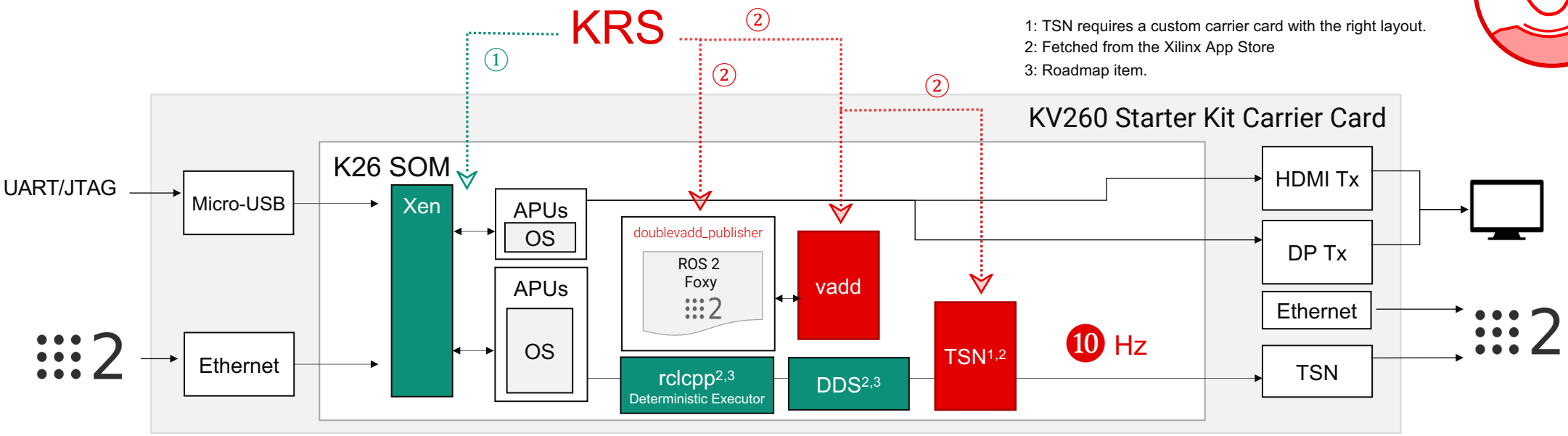
Accelerating ROS 2 – with KRS

ROS 2 Hardware acceleration "hello world" – doublevadd_publisher

```
# rebuild rootfs for determinism ①
$ colcon acceleration hypervisor  \
        --dom0 vanilla \
        --domU preempt_rt
```

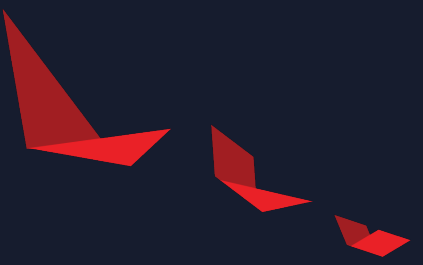
```
# rebuild ② doublevadd_publisher, including kernels
$ colcon build --merge-install --mixin kv260  \
        --install-dir install-kv260 \
        --build-dir build-kv260
```

Leo "the roboticist"



Deterministic real-time ROS 2 interactions with KRS with high performance

XILINX



ReconROS

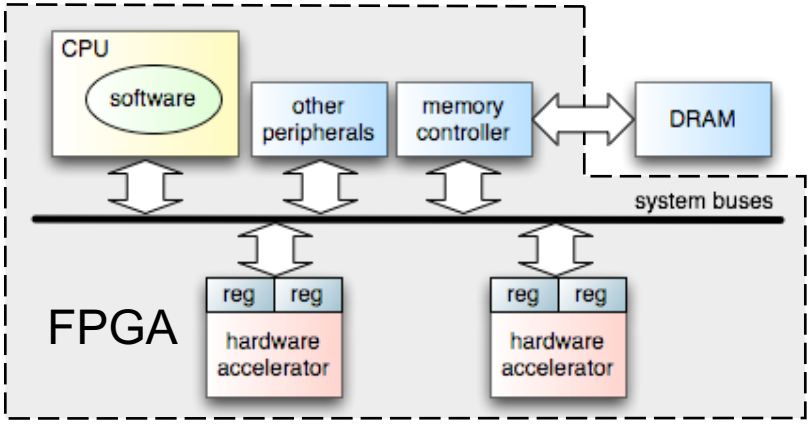
ReconROS
(ReconOS + ROS)

Flexible Hardware Acceleration for ROS2 Applications

Christian Lienen, Marco Platzner
Paderborn University
September 29, 2021

Operating System Integration of FPGAs

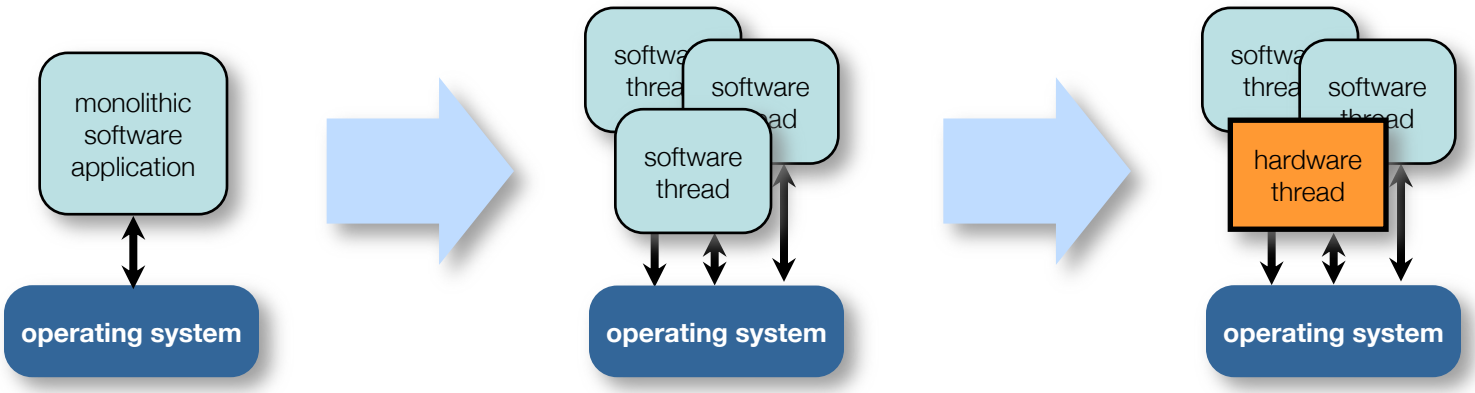
- Traditionally, hardware accelerators integrated as coprocessors
  - software (CPU) explicitly communicates with hardware accelerator
  - memory-mapped registers
  - remote procedure call



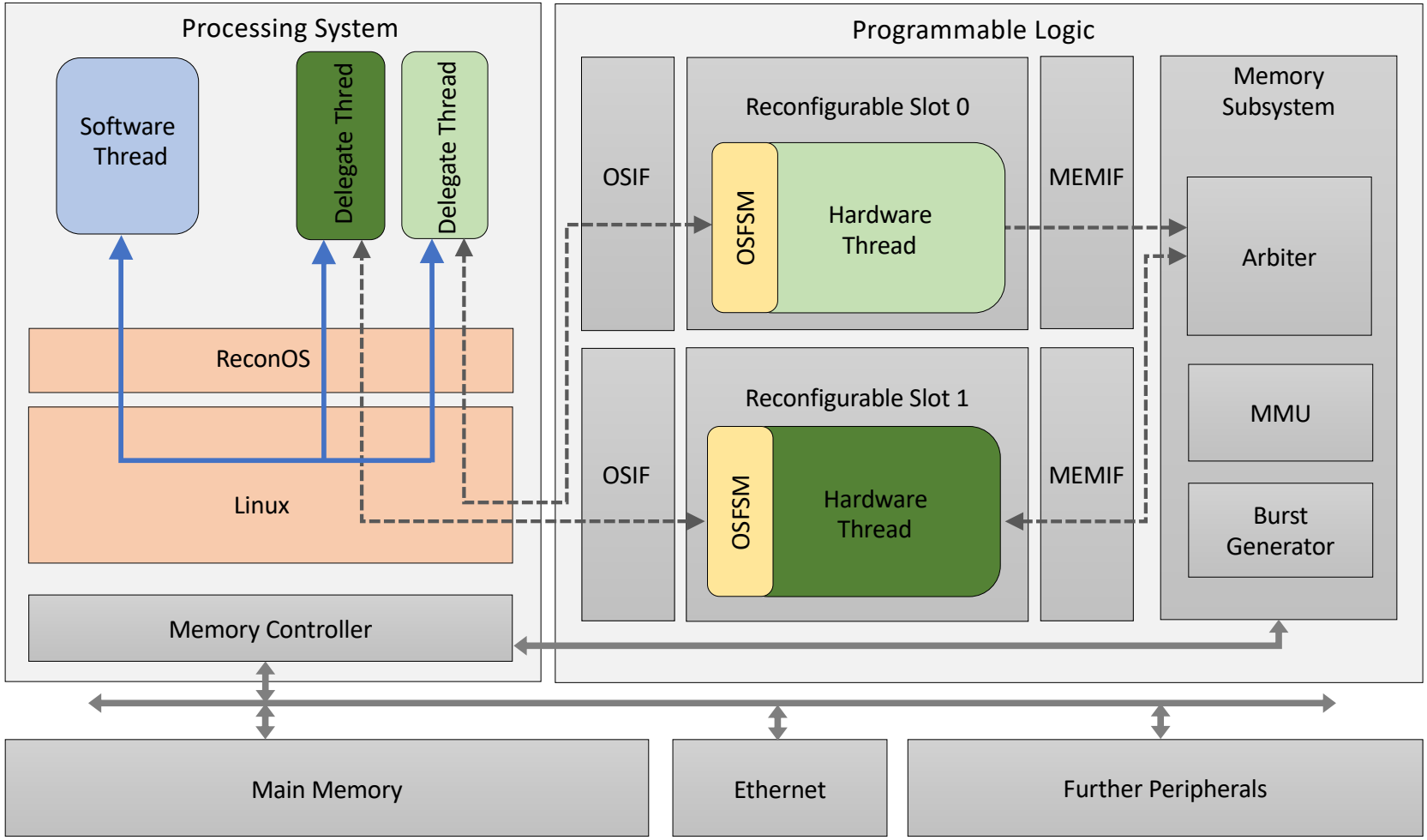

```
int main() {
    ...
    data = do_something();
    io_out32(HW_ACCEL_DATA_REG, data);
    io_out32(HW_ACCEL_CMD_REG, CMD_GO);
    while (io_in32(HW_ACCEL_STATUS_REG) != DONE)
       sleep();
    result = io_in32(HW_ACCEL_RESULT_REG);
    ...
}
```

ReconOS Approach

- Extend multithreaded programming model to hardware
  - hardware accelerators are turned into hardware threads
  - threads communicate and synchronize using programming model primitives, e.g., semaphores, mutexes, mailboxes, shared memory read/write
  - established software abstraction (e.g., POSIX pthreads) provides a standarized interface between software and hardware



Example ReconOS Hardware Architecture



- Hardware threads use cooperative multitasking
- Hardware threads can be loaded / removed by partial reconfiguration

ReconOS Versions

Version 1.0
- eCos / PowerPC
- Virtex-2, Virtex-2Pro, Virtex-4

Version 2.0
- Linux, eCos / PowerPC
- Virtex-2Pro, Virtex-4
- virtual memory support

Version 3.0
- Linux, xilkernel / MicroBlaze
- Virtex-6

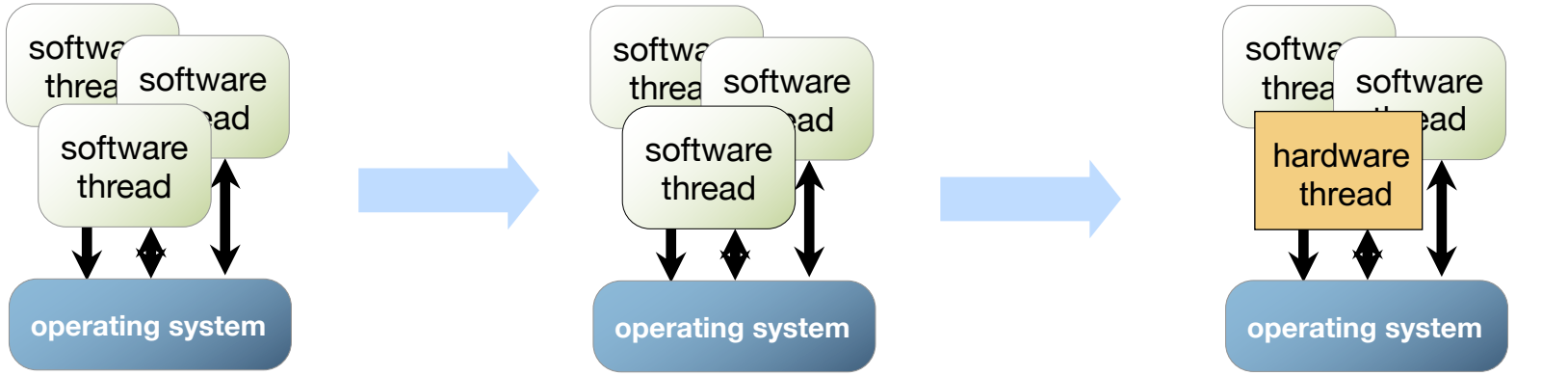
Version 3.1
- Linux / ARM
- Xilinx Zynq
- ISE

Version 4.0
- Linux / ARM
- Xilinx Zynq
- Vivado, HLS for hardware thread design, flexible and dynamic clocks

www.reconos.de

Experience with ReconOS

1. ReconOS supports a step-by-step application design process

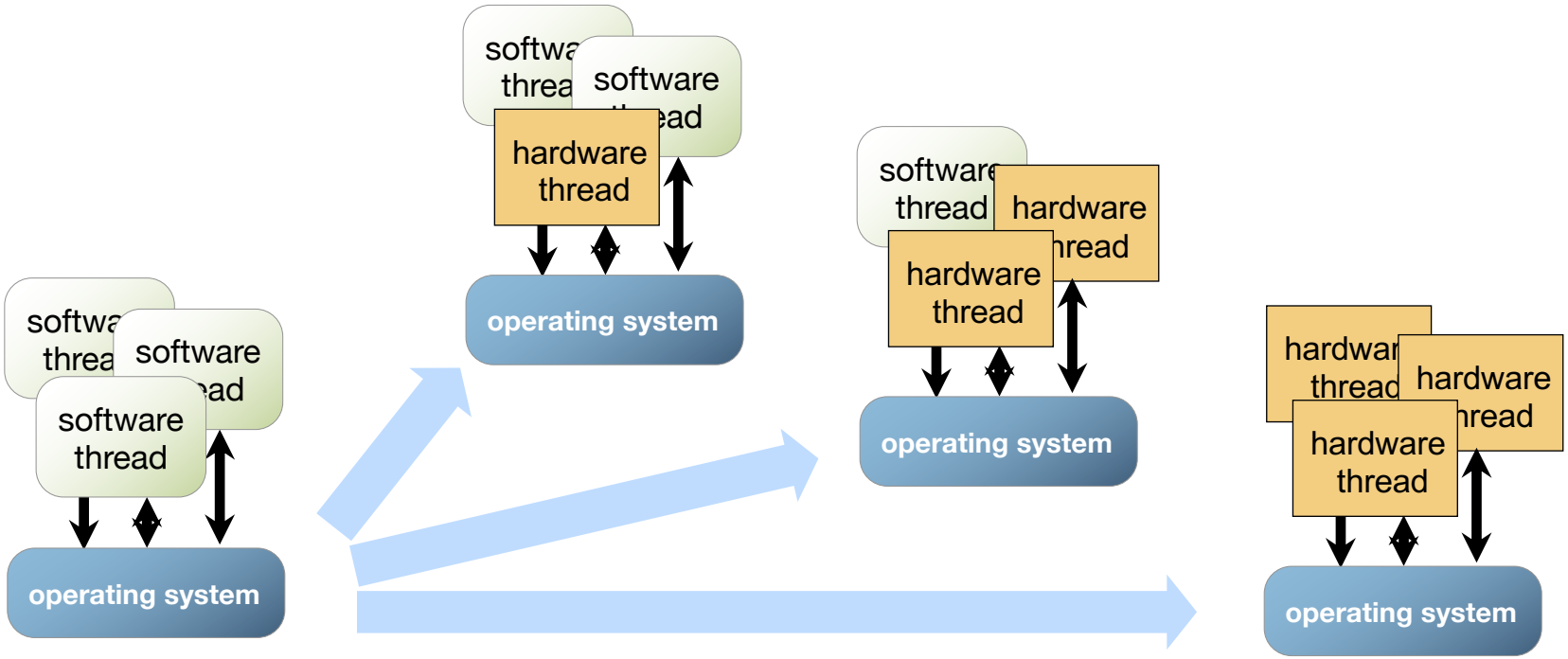


1. desktop/Linux

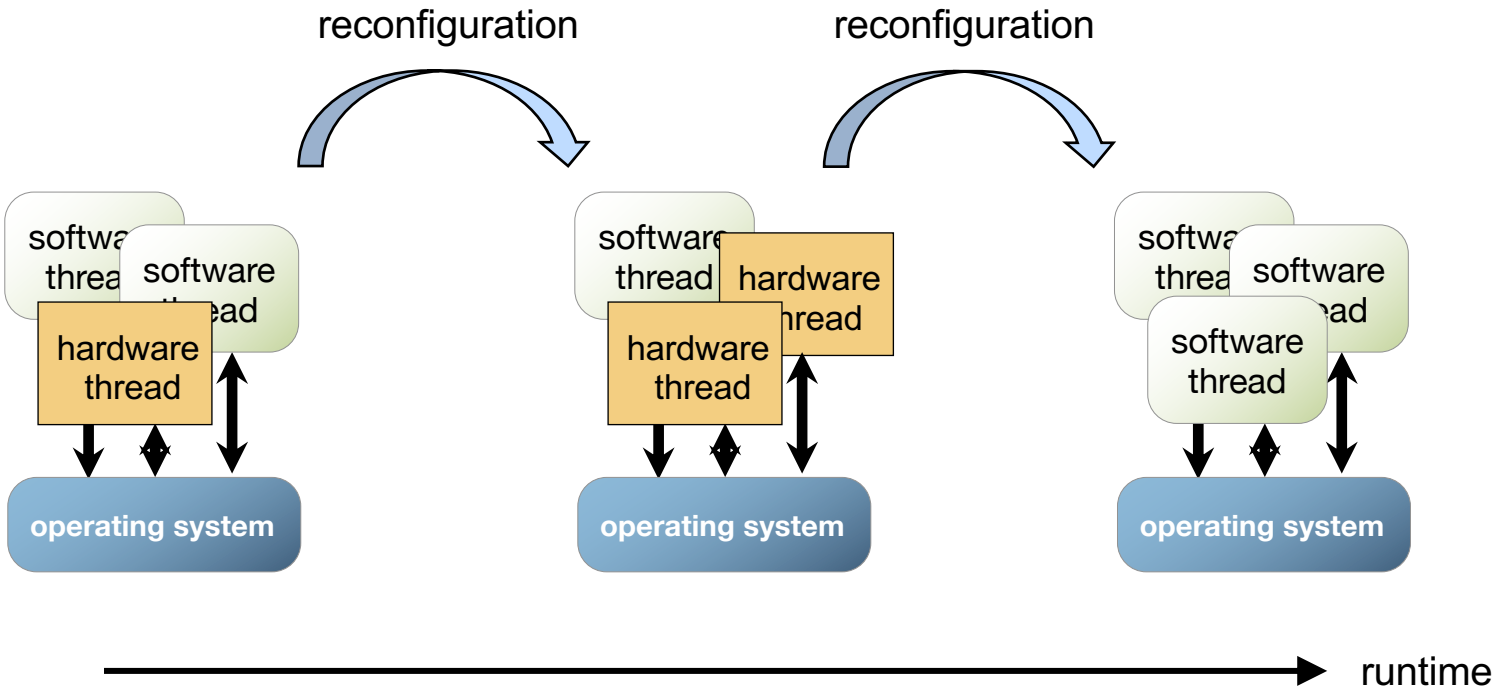
2. platform FPGA/Linux

3. platform FPGA/Linux
+ hardware accelerators

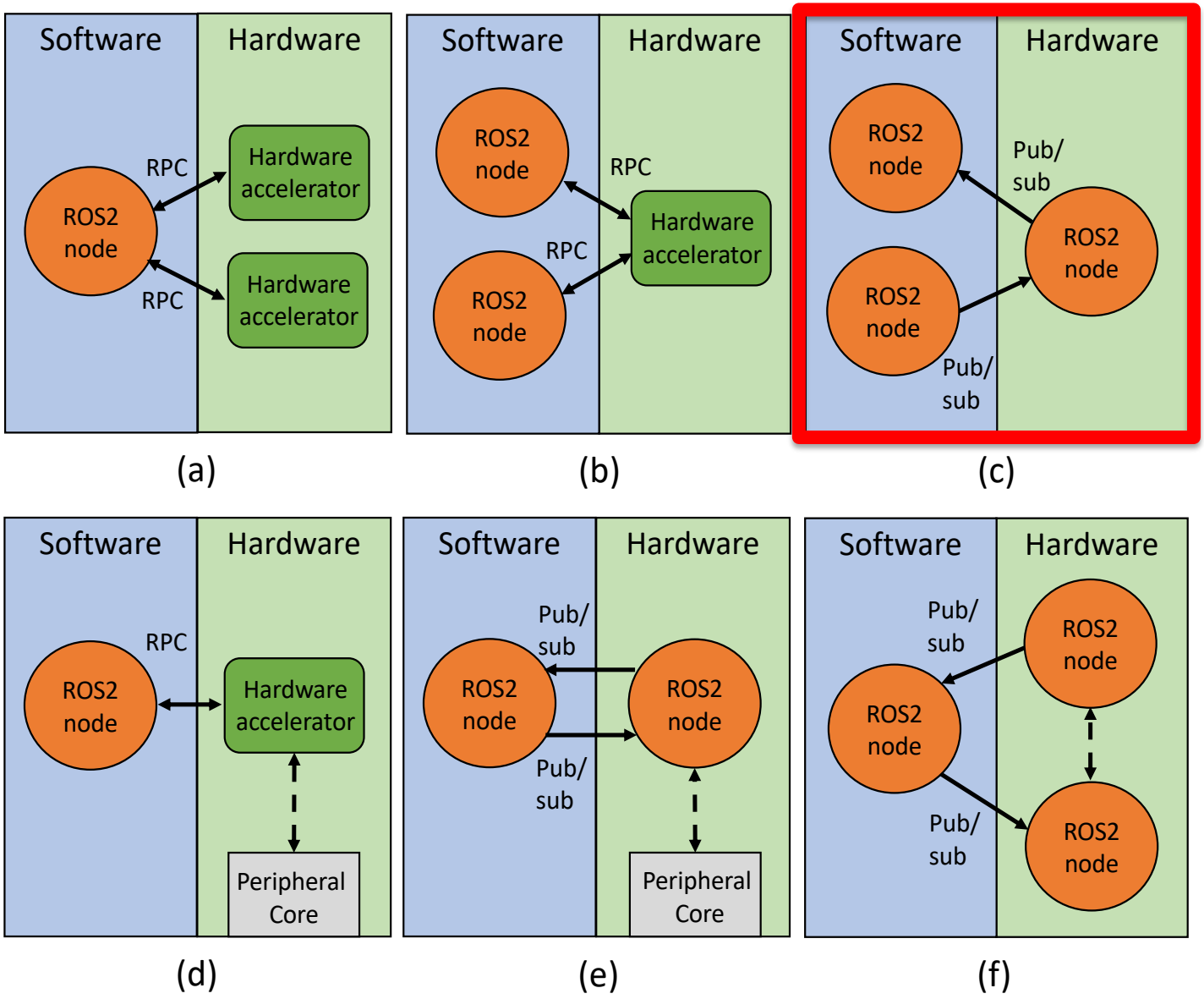
2. ReconOS facilitates fast design space exploration



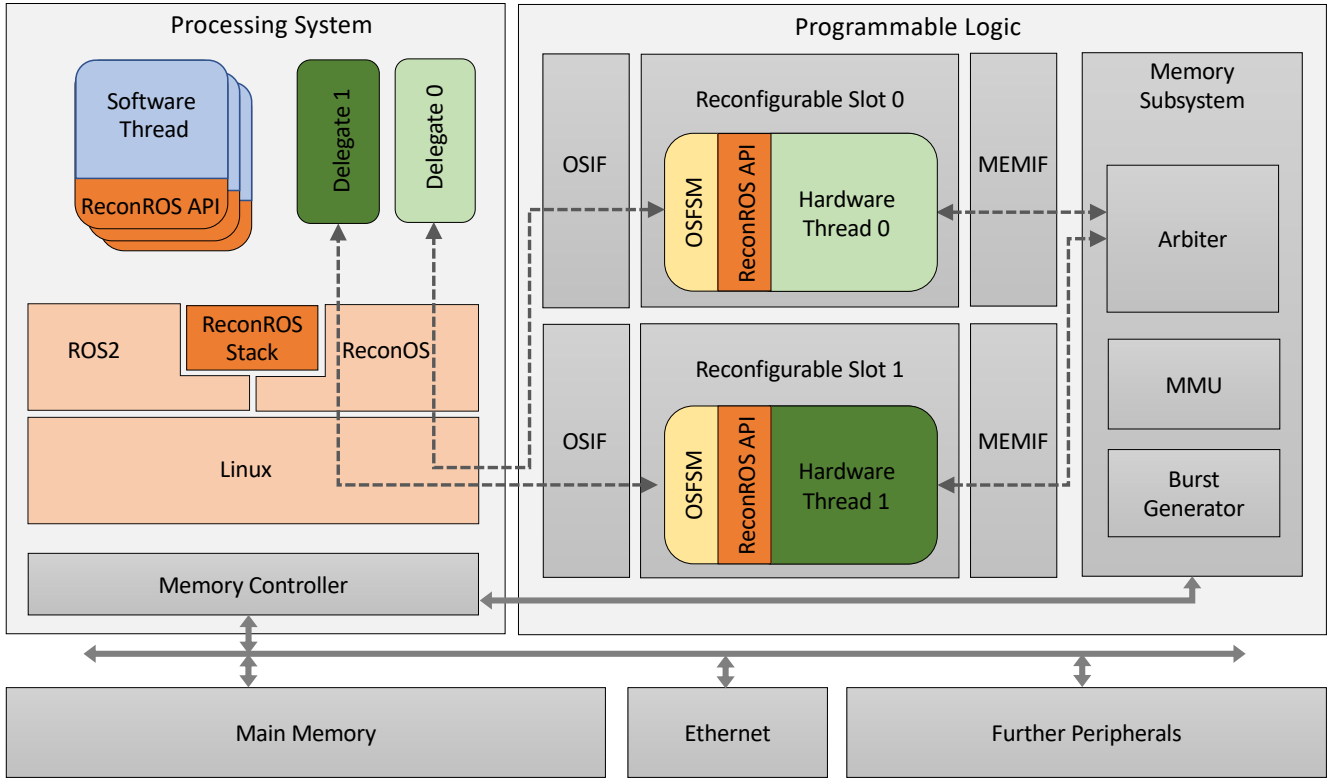
3. ReconOS enables migration between software and hardware at runtime → (self-)adaptive systems



Hardware Acceleration Options for ROS Nodes

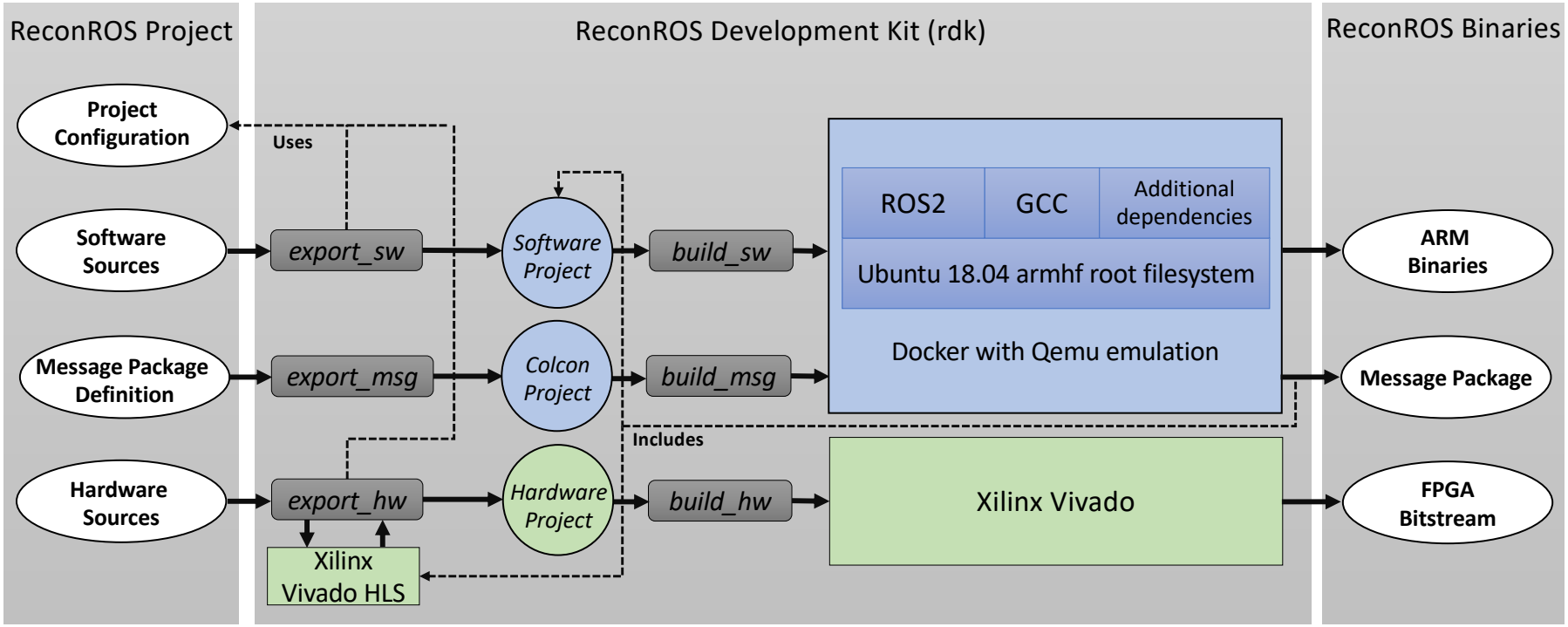


ReconROS Architecture



- ReconROS API
  - provides access to ReconROS stack for hardware and software threads
  - easily extendable and slightly abstracted subset of ROS 2 – *rcl* API
- ReconROS stack
  - ROS 2-related objects added as ReconOS primitives (e.g. ROS 2 nodes, pub/sub, services, actions...)
  - stack includes initializations and memory management

ReconROS Design Flow

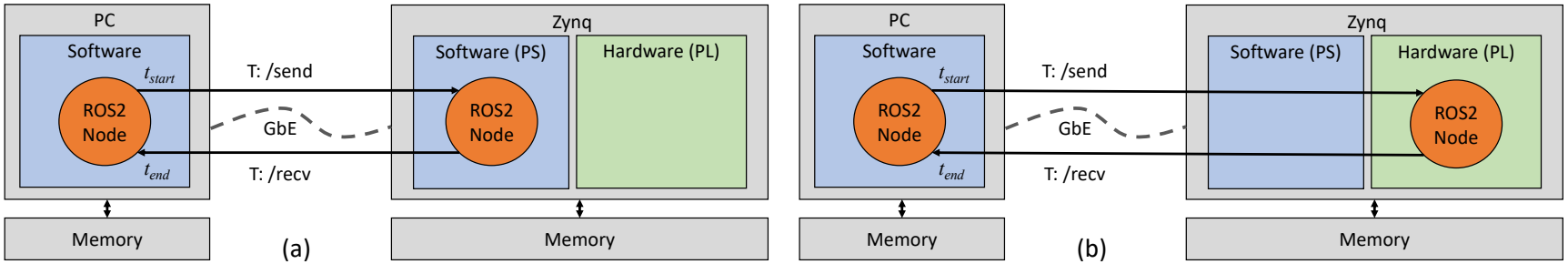


- Software / message tool chain
  - support of C/C++ source files and ROS 2 message definition files
  - uses cross compilation to ARM for easier handling of dependencies
- Hardware tool chain
  - support of C/C++ HLS sources and VHDL / Verilog descriptions
  - message definitions available in VIVADO HLS as C structs

Benefits of Mapping ROS Nodes to Hardware

- Higher performance
  - raw speedup depends on node function and the effort spent for optimization
  - overall speedup depends also on overall ROS application (Amdahl's law)
  - hardware nodes execute in parallel to CPU cores → lower CPU load and/or higher performance for overall application
- Potentially lower jitter
- Power/energy savings
- Pre/post-processing in hardware
  - when sensor/actuator is directly connected to FPGA

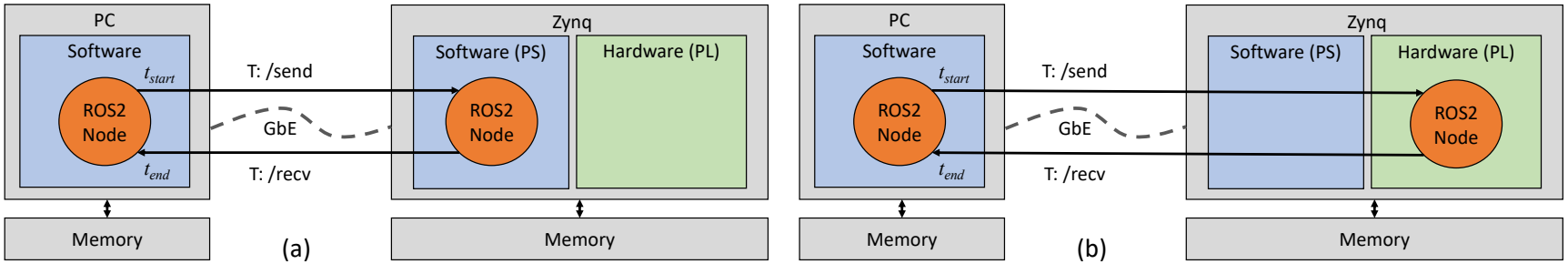
Evaluation



Message size	$t_{pp-echo-SW}$ [ms]	$t_{pp-echo-HW}$ [ms]	$S_{pp-echo}$
4 Byte	0.81	1.2	0.68
8 KiB	10.65	10.48	1.02
1 MiB	52.21	52.16	1.01
6 MiB	363.91	363.30	1.00
10 MiB	630.37	624.02	1.01

- ‘Echo’ application
  - ROS nodes just pass message pointers between subscribe() and publish(), but there is no real computation
  - no overhead for using hardware (underlying ReconOS signaling)

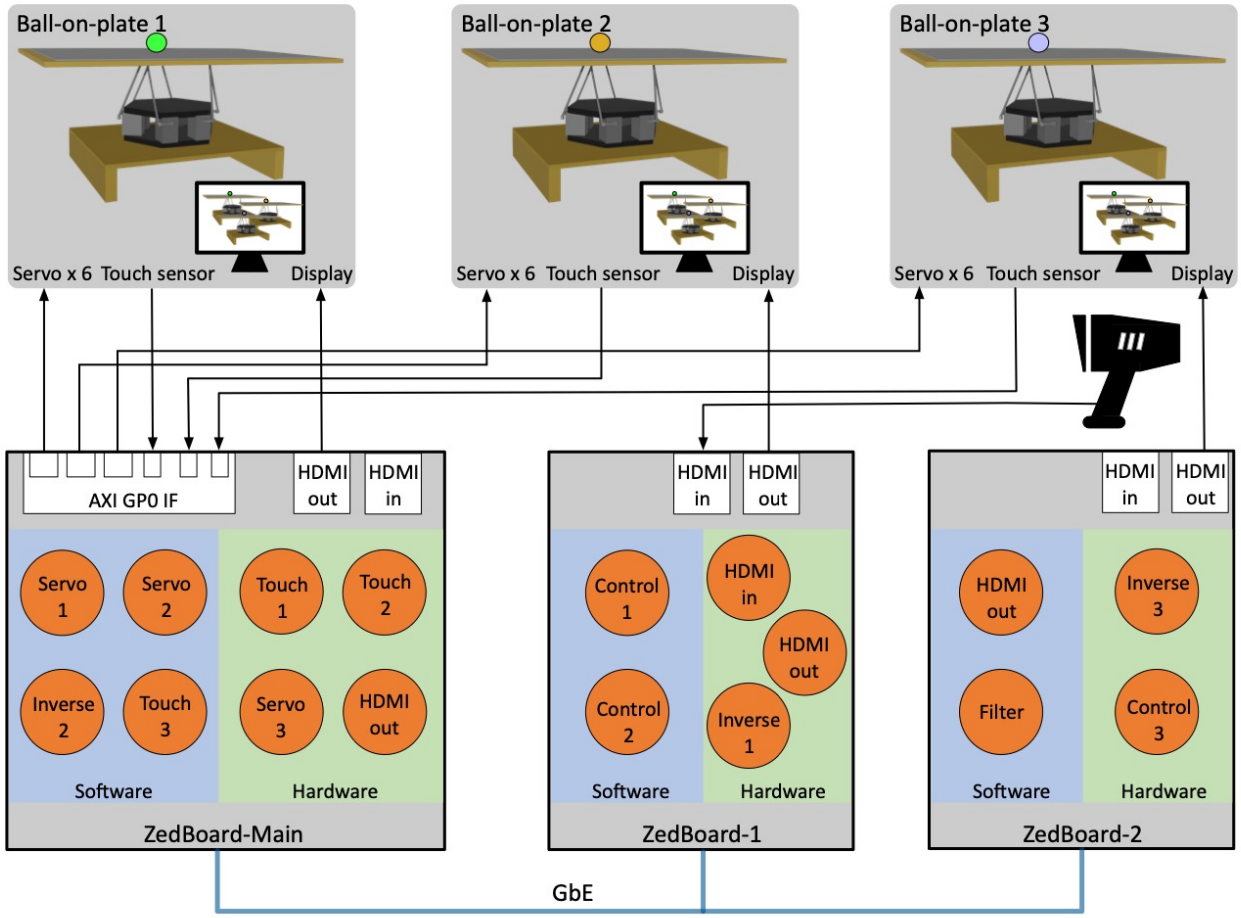
Evaluation



RECONROS application	Message Size In/Out	t_{raw-SW} [ms]	t_{raw-HW} [ms]	S_{raw}	t_{pp-SW} [ms]	t_{pp-HW} [ms]	S_{pp}
Inverse	4/4 Byte	1.20	0.19	6.32	7.70	6.64	1.16
Sorting	8/8 KiB	17.44	35.11	0.50	24.42	42.08	0.58
Sobel	900/900 KiB	37.53	22.28	1.68	83.39	68.54	1.22
MNIST	850/4 Byte	88.03	30.74	2.86	98.58	41.25	2.39

- ‘Smaller’ applications
  - hardware nodes created with high-level synthesis (HLS) from C/C++ without any optimization

- Mechatronics application
  - easy to create new system configurations with different distributions of nodes between compute boards and between software and hardware



Summary, Sources and Literature

ReconROS enables ROS 2 application developers to use flexible hardware acceleration with FPGAs through a consistent programming model across the hardware/software boundary.

ReconOS  https://github.com/ReconOS/reconos

[1] Enno Lübbers and Marco Platzner. "ReconOS: "Multithreaded programming for reconfigurable computers", ACM Transactions on Embedded Computing Systems, 9(1):1-33, 2009.

[2] Andreas Agne, Markus Happe, Ariane Keller, Enno Lübbers, Bernhard Plattner, Marco Platzner and Christian Plessl. "ReconOS: An Operating System Approach for Reconfigurable Computing", IEEE Micro, 34(1):60-71, 2014.

ReconROS  https://github.com/Lien182/ReconROS
(Code, Demos, Board Images,…)

[3] Christian Lienen, Marco Platzner and Bernhard Rinner. "ReconROS: Flexible Hardware Acceleration for ROS2 Applications", International Conference on Field Programmable Technology (ICFPT), 2020.

[4] Christian Lienen and Marco Platzner. "Design of Distributed Reconfigurable Robotics Systems with ReconOS", ArXiv paper under review, 2021.

Q&A