



OpenCPI - Open Component Portability Infrastructure

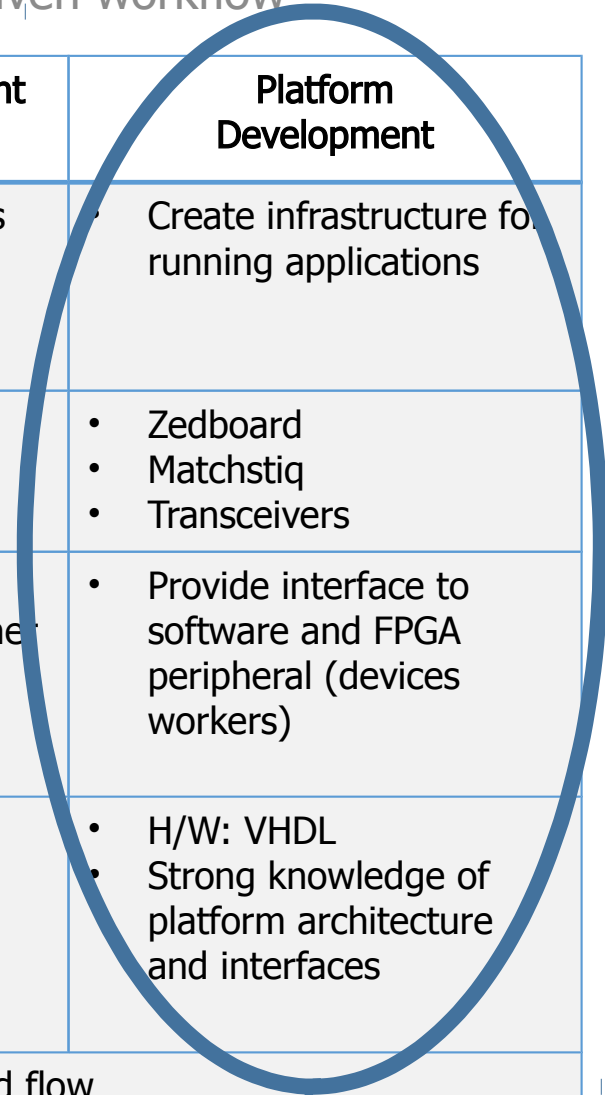
An Open Source Infrastructure for
Heterogeneous/Embedded Component-Based
Systems and Applications

Advanced OpenCPI Platform Development

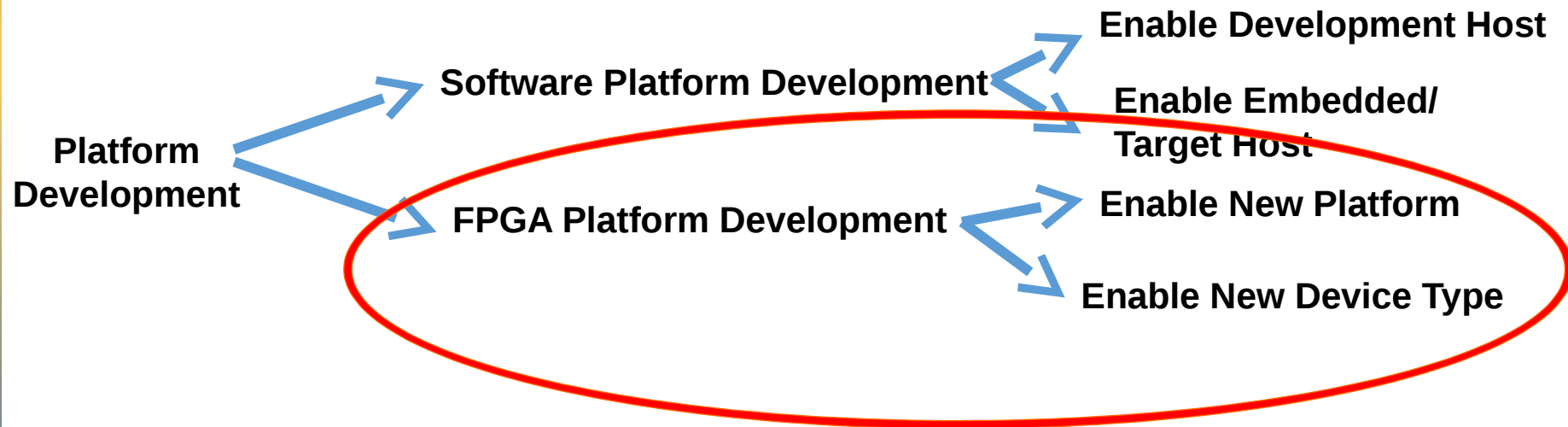
Summary of OpenCPI Development Roles

Three types of development with common XML-driven workflow

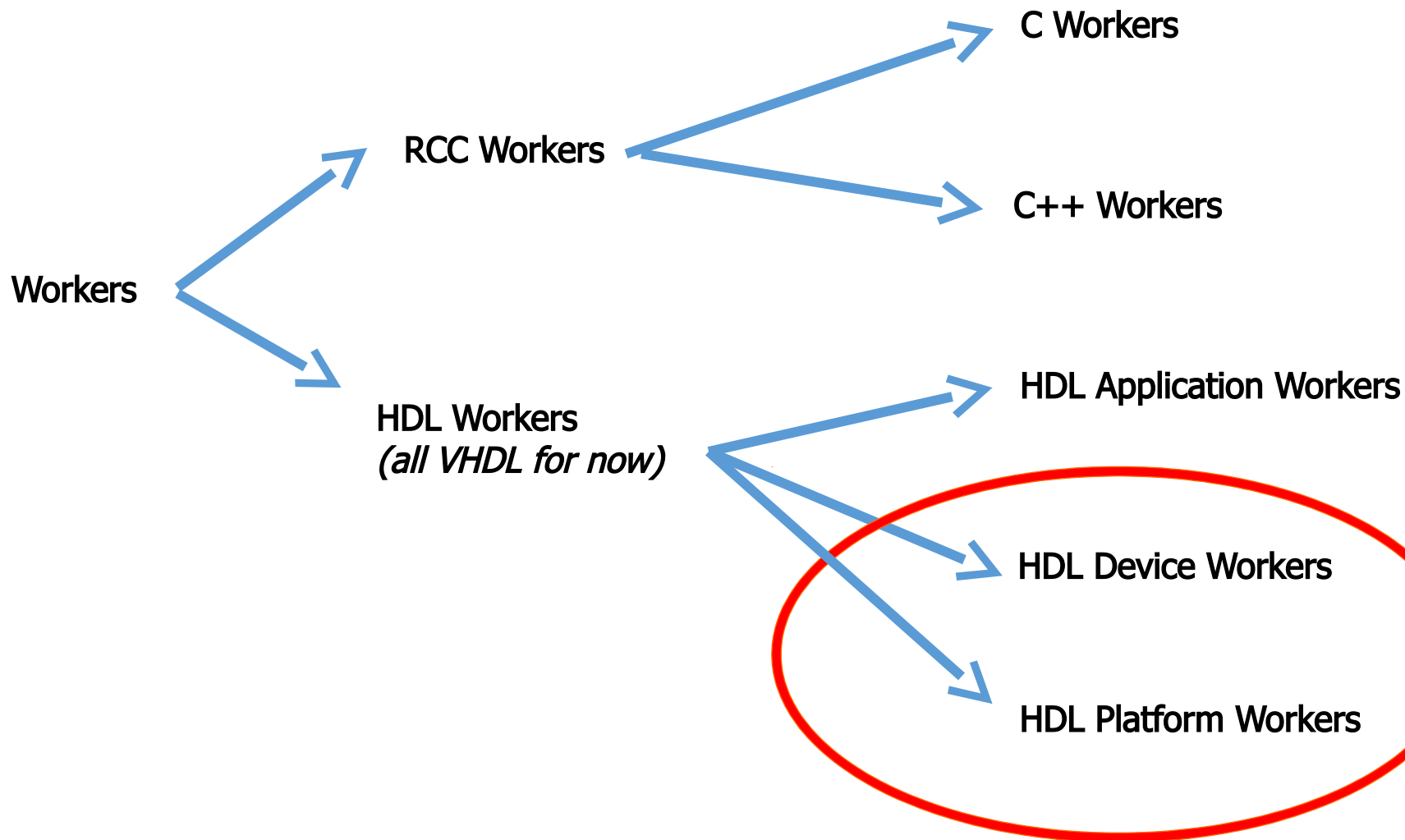
	Application Development	Component Development	Platform Development
Objective	• Create applications using components	• Create building blocks for applications	• Create infrastructure for running applications
Examples	• Tb_bias • FSK app	• Bias • FIR filter	• Zedboard • Matchstiq • Transceivers
Key functions	• Declare components and their connections and properties	• Process data and interface between other components • Vendor agnostic (ideally)	• Provide interface to software and FPGA peripheral (devices workers)
Skills Required	• Familiarity with component library	• S/W: C, C++ • H/W: VHDL	• H/W: VHDL • Strong knowledge of platform architecture and interfaces
Knowledge of OpenCPI build flow			



Enabling new platforms and devices for OpenCPI apps



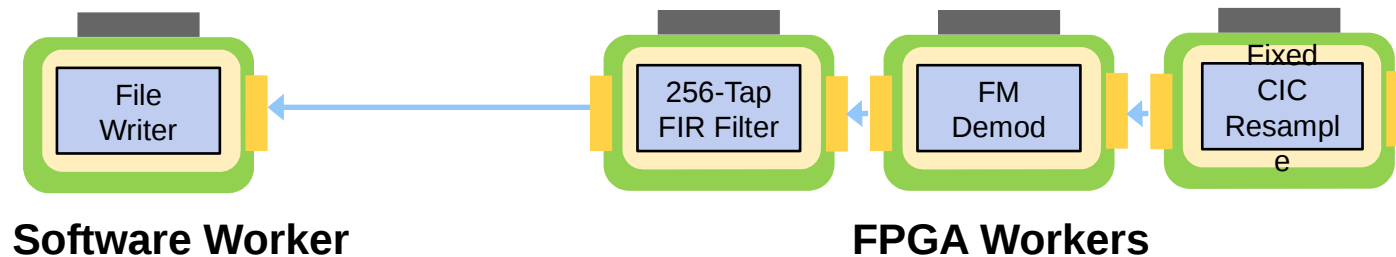
Types of OpenCPI Workers



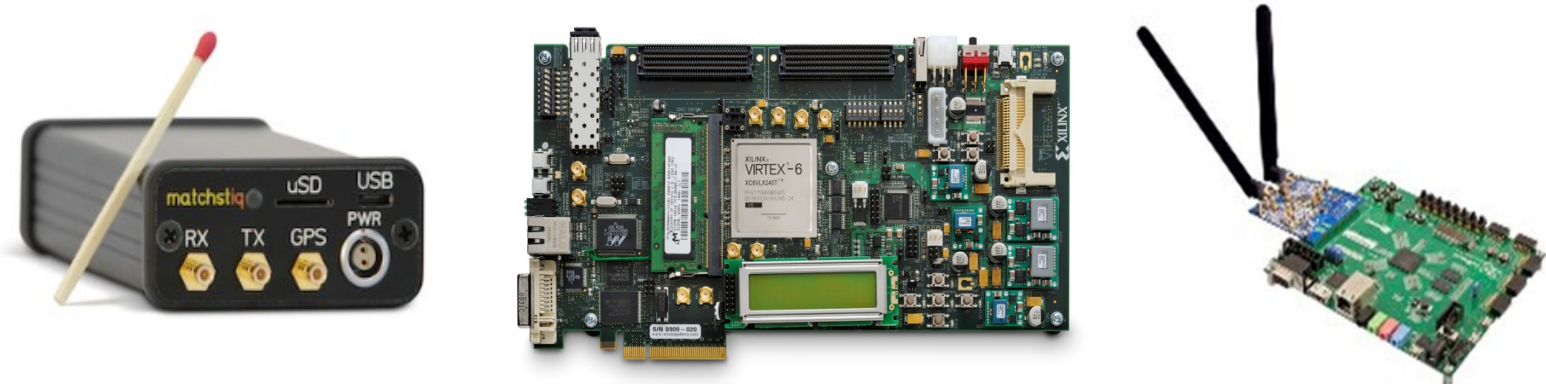
What is an OpenCPI FPGA Platform?

- An **OpenCPI FPGA Platform** is the FPGA, its surrounding infrastructure and attached devices.
- *Enabling the platform* allows it to be used for an OpenCPI application.
- Put another way...

What do I need to run an OpenCPI app?

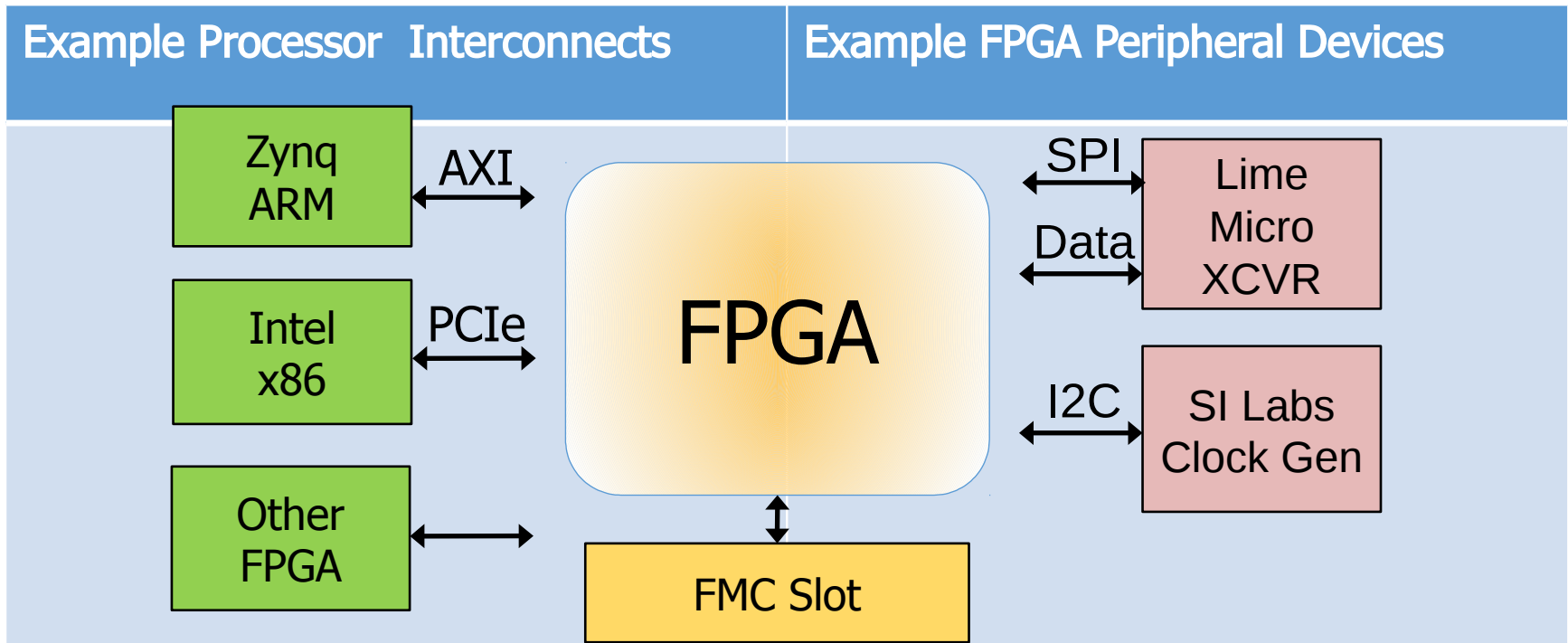


on *my* hardware?



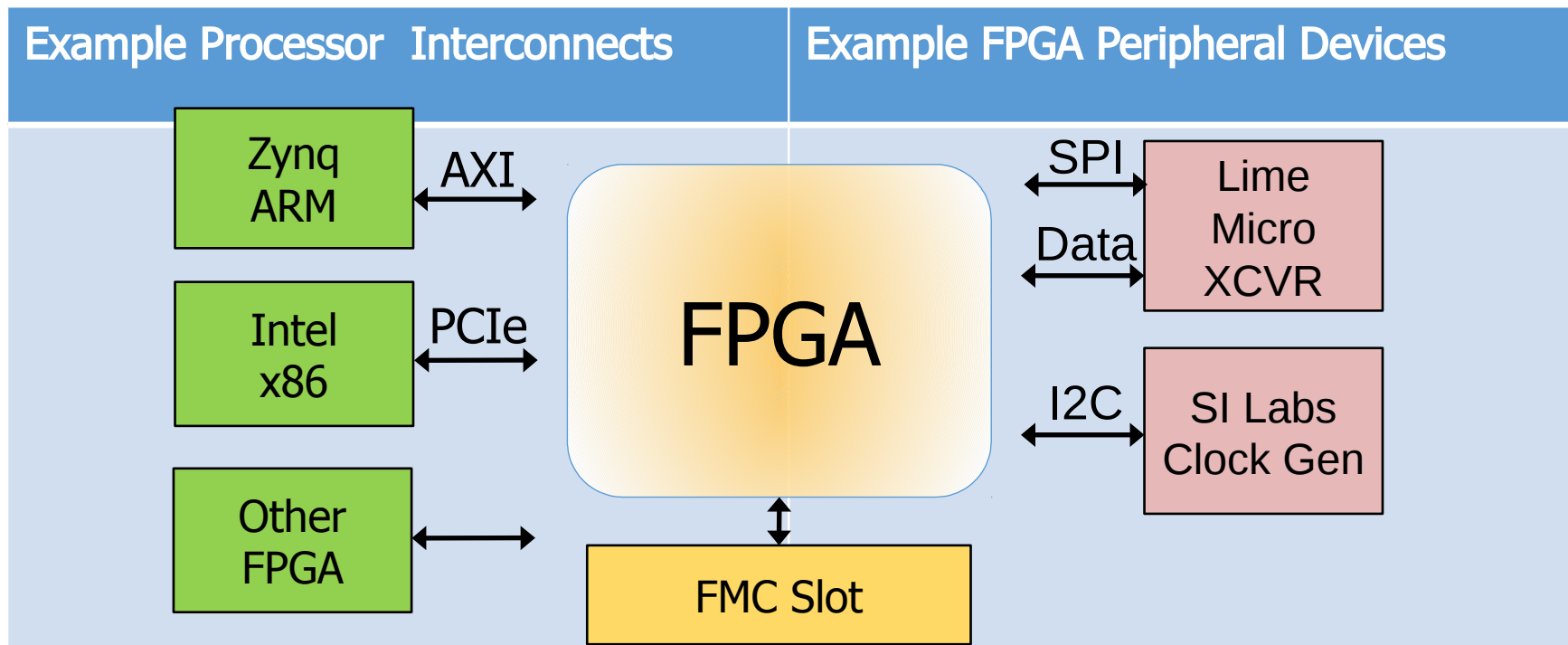
What Does an OpenCPI FPGA Platform Consist of?

1. **The FPGA:** a place where HDL application workers may execute
2. **Interconnects:** off-chip connections to processor(s) or other FPGA platforms
3. **Devices:** peripherals attached to the FPGA, useful to applications
4. **Slots:** hardware physical interfaces where *cards* (with devices) are plugged in

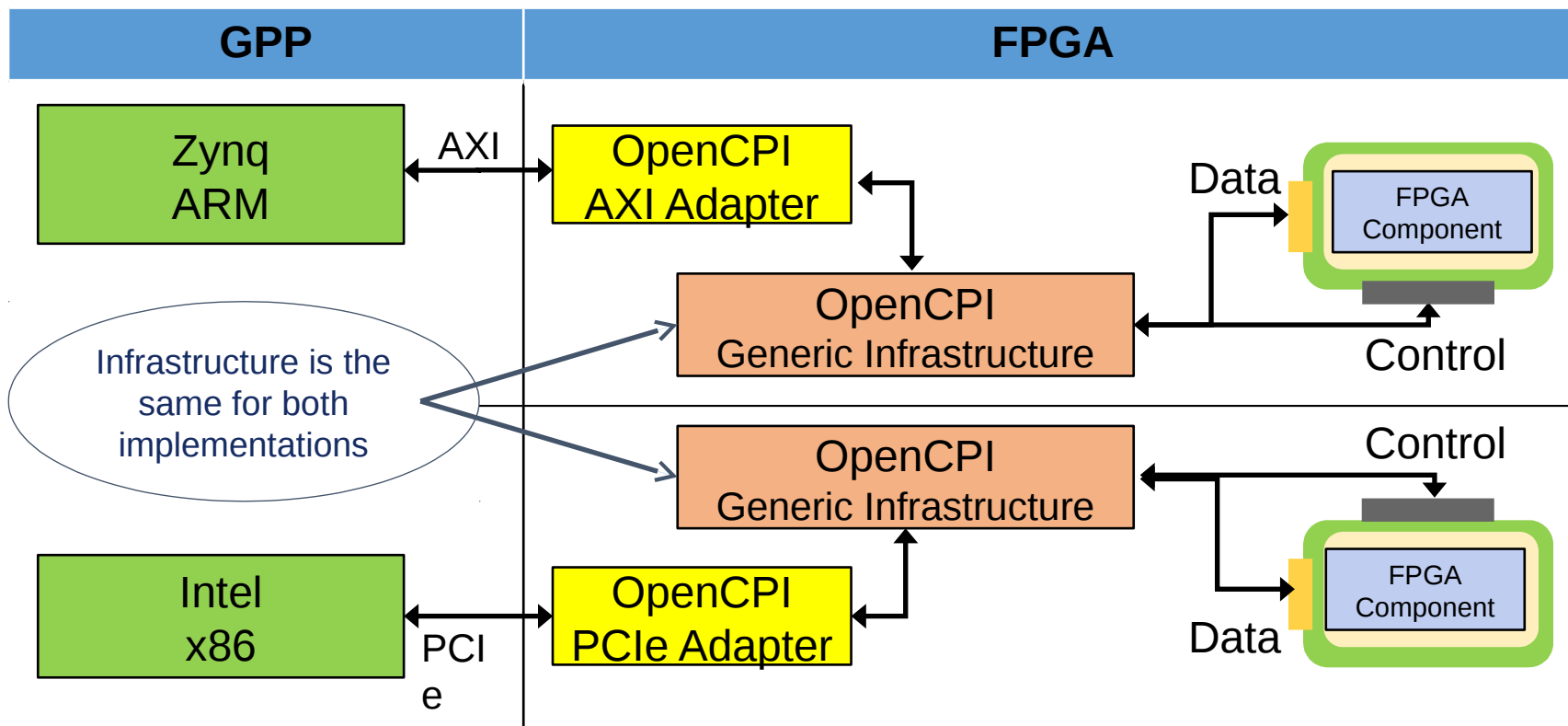


What is Needed to Enable an OpenCPI FPGA Platform?

1. The OCPI on-chip *clocks*, *control* and *data planes* must be adapted to the external (off-FPGA) resources available in the platform: create the *HDL platform worker*.
 - **Control Plane:** mechanisms to allow the processor to control and configure FPGA workers
 - **Data Plane:** mechanisms to allow messages to flow between on-chip workers and workers elsewhere
 - **Clocks:** the mechanisms to provide clocking and time-of-day to FPGA workers
2. The attached devices must be supported by specialized workers acting as device drivers: create or reuse *HDL device workers*.

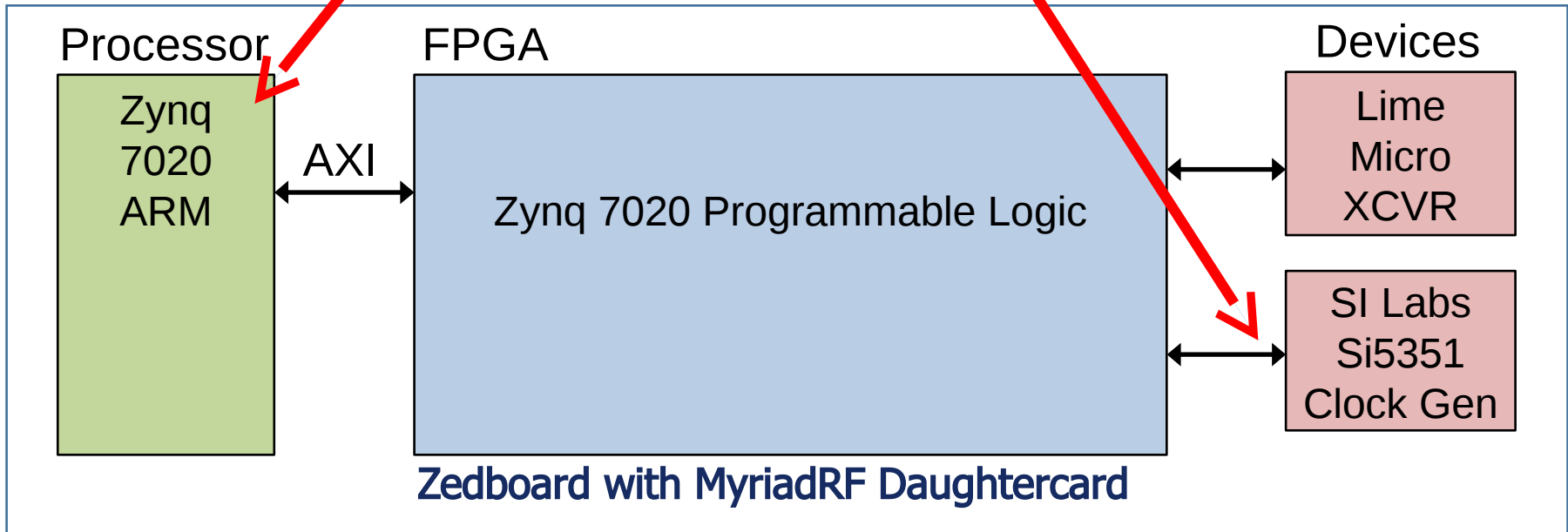
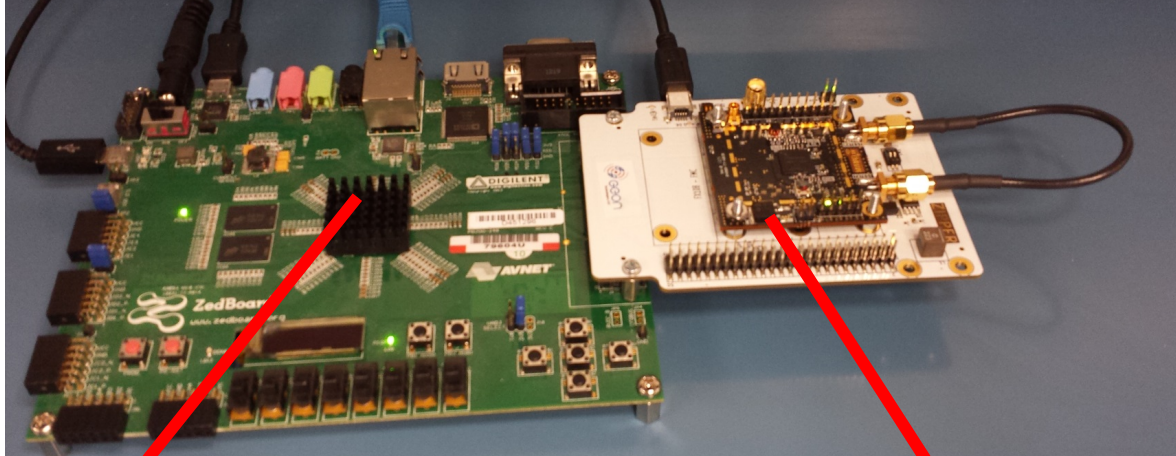


- Generic Infrastructure Modules
 - For controlling FPGA worker registers/properties from software (control plane)
 - For data transfer from FPGA workers to/from other platforms and software (data plane)
 - These modules are instanced automatically as needed
- Some existing **HDL device workers**/drivers for controlling certain devices (on any platform).
- Adapter logic for different processor/FPGA interconnects



Case Study: Zedboard with MyriadRF

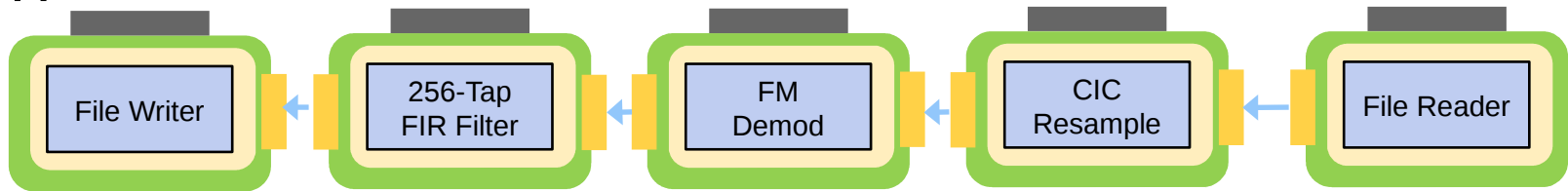
Treated as one board, ignoring the card/slot aspect for now



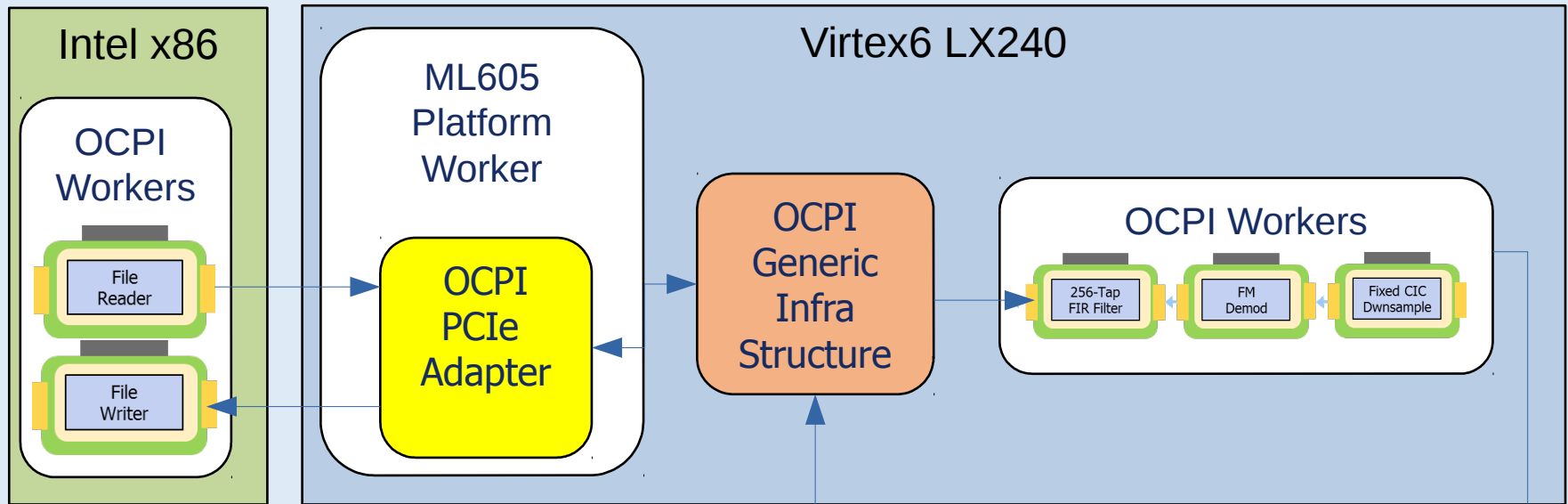
Case Study: What We Started With

- Virtex6 PCIe development kit: ML605
- OpenCPI adapter logic for Virtex6 PCIe
- Existing application using file based input and output
- *No devices used*

App

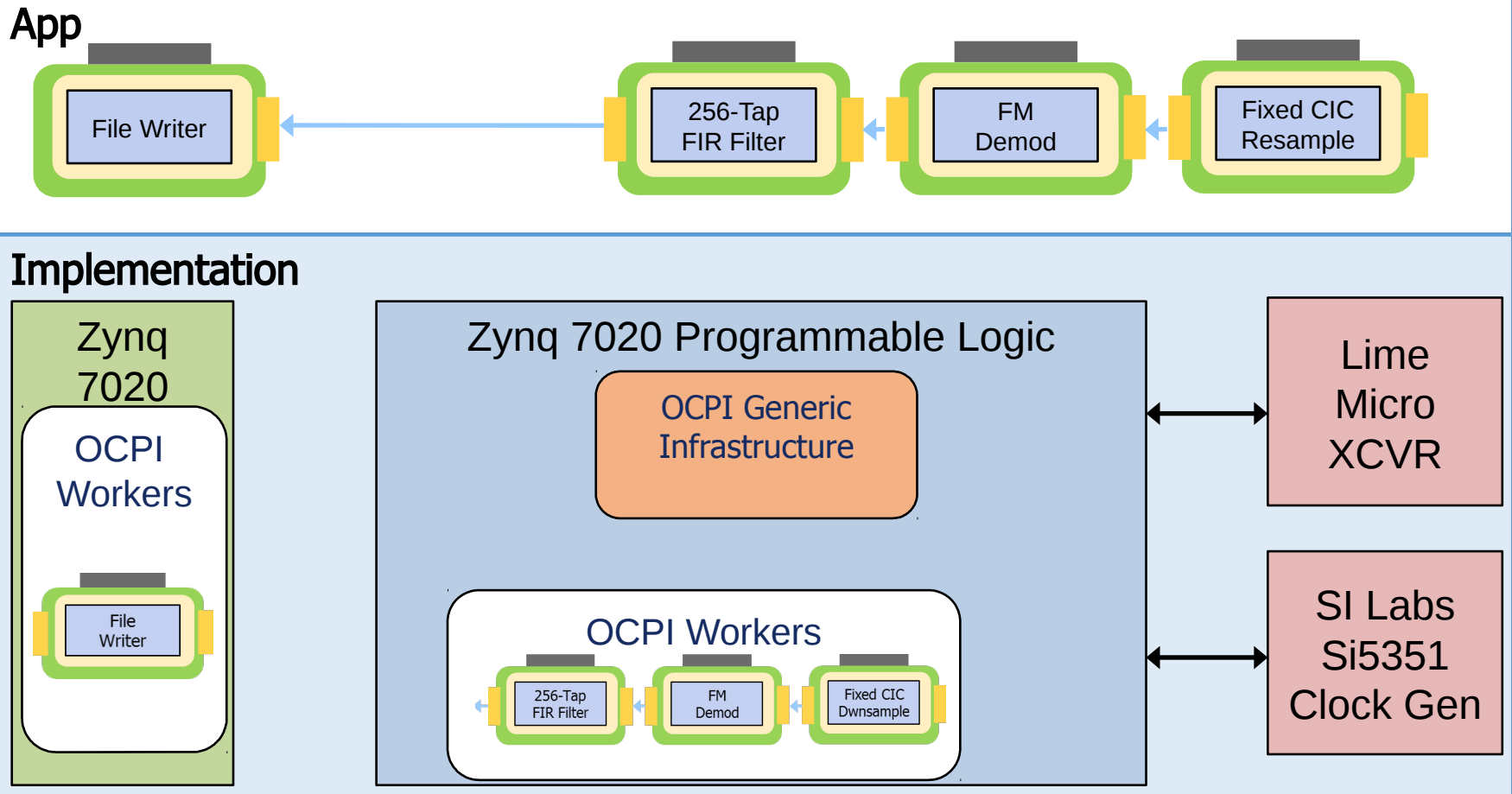


Implementation



Case Study: What Could Be Re-used?

- What could be reused?
 - Generic infrastructure
 - HDL and software workers
 - Workers must be recompiled for Zynq/ARM. No code changes needed



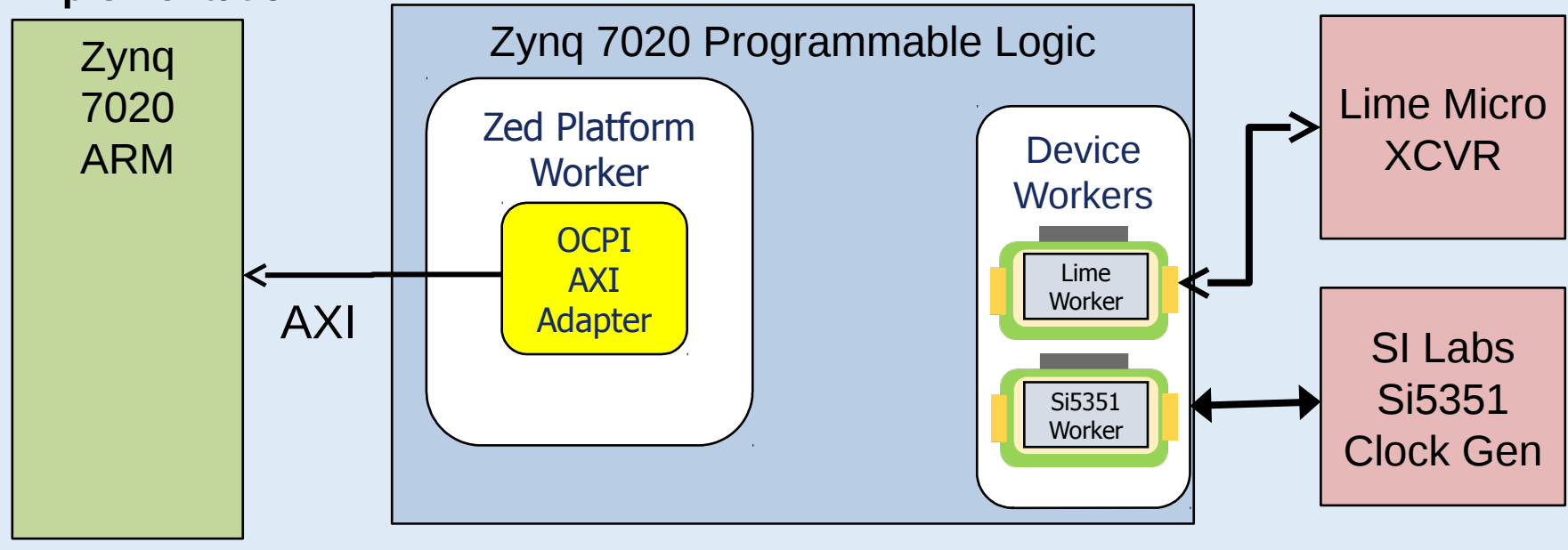
Case Study: What was Needed?

- What was needed?
 - Zed platform worker with OCPI adapter
 - Device workers to ingest data and control devices

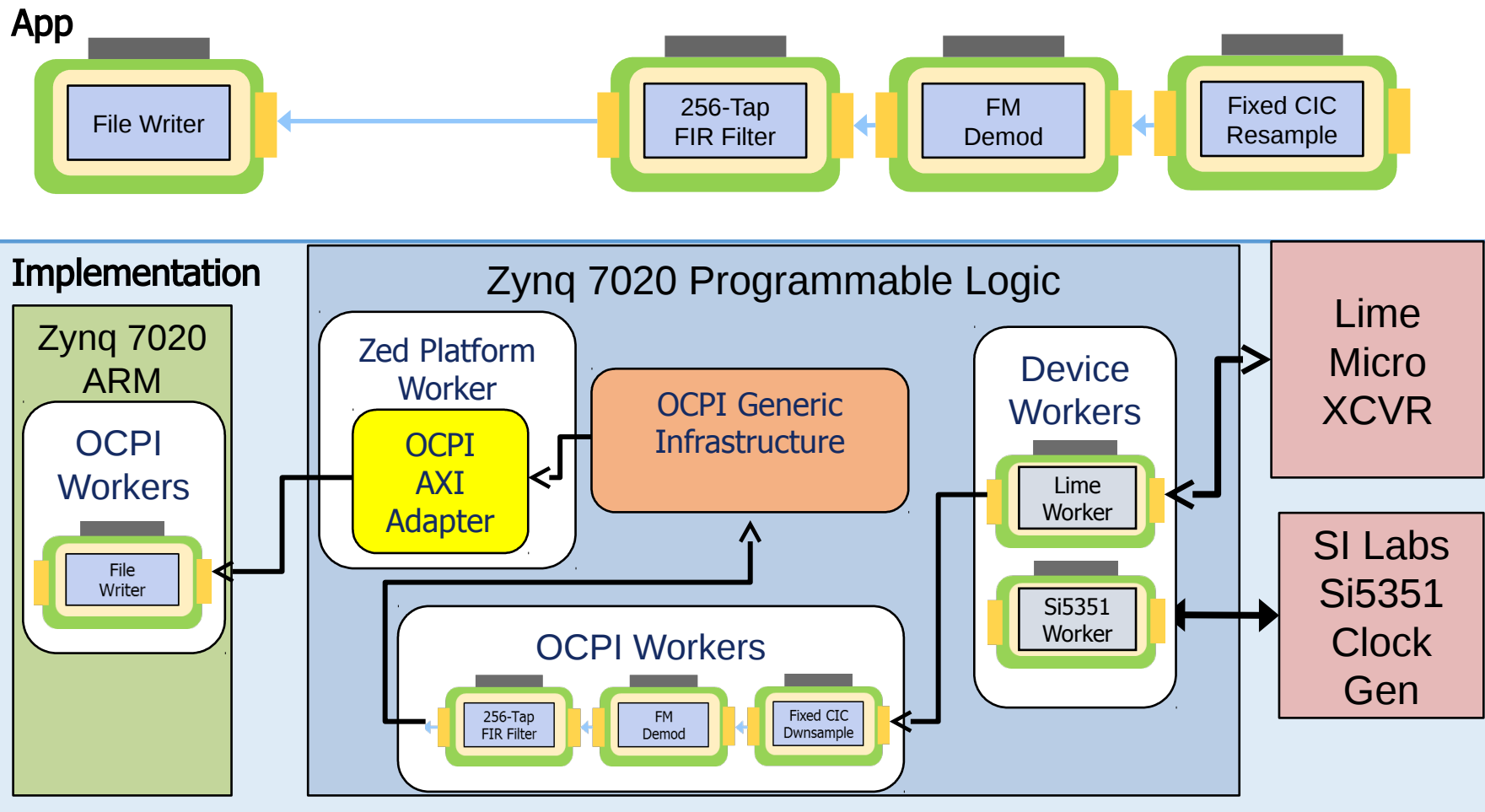
App



Implementation

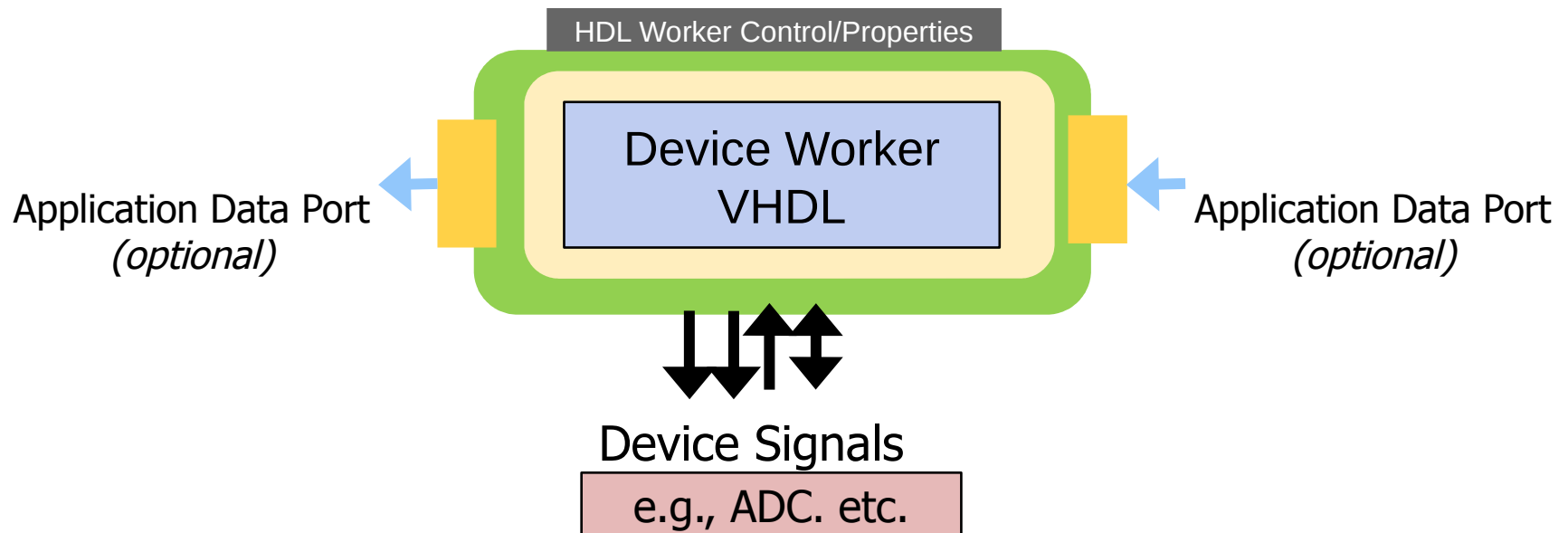


Case Study: Final Implementation



FPGA (HDL) Device Worker

- Acts as the controlling logic for an attached device: the *device driver*
- A superset of an application worker, with normal properties and ports
- Connects directly to the I/O pins for signals between FPGA and device
- HDL Device Worker OWD/XML has element: `<Signal>`
 - Declares signals to/from the device, with their direction, width, I/O attributes
 - These signals are made directly available in the VHDL architecture worker code

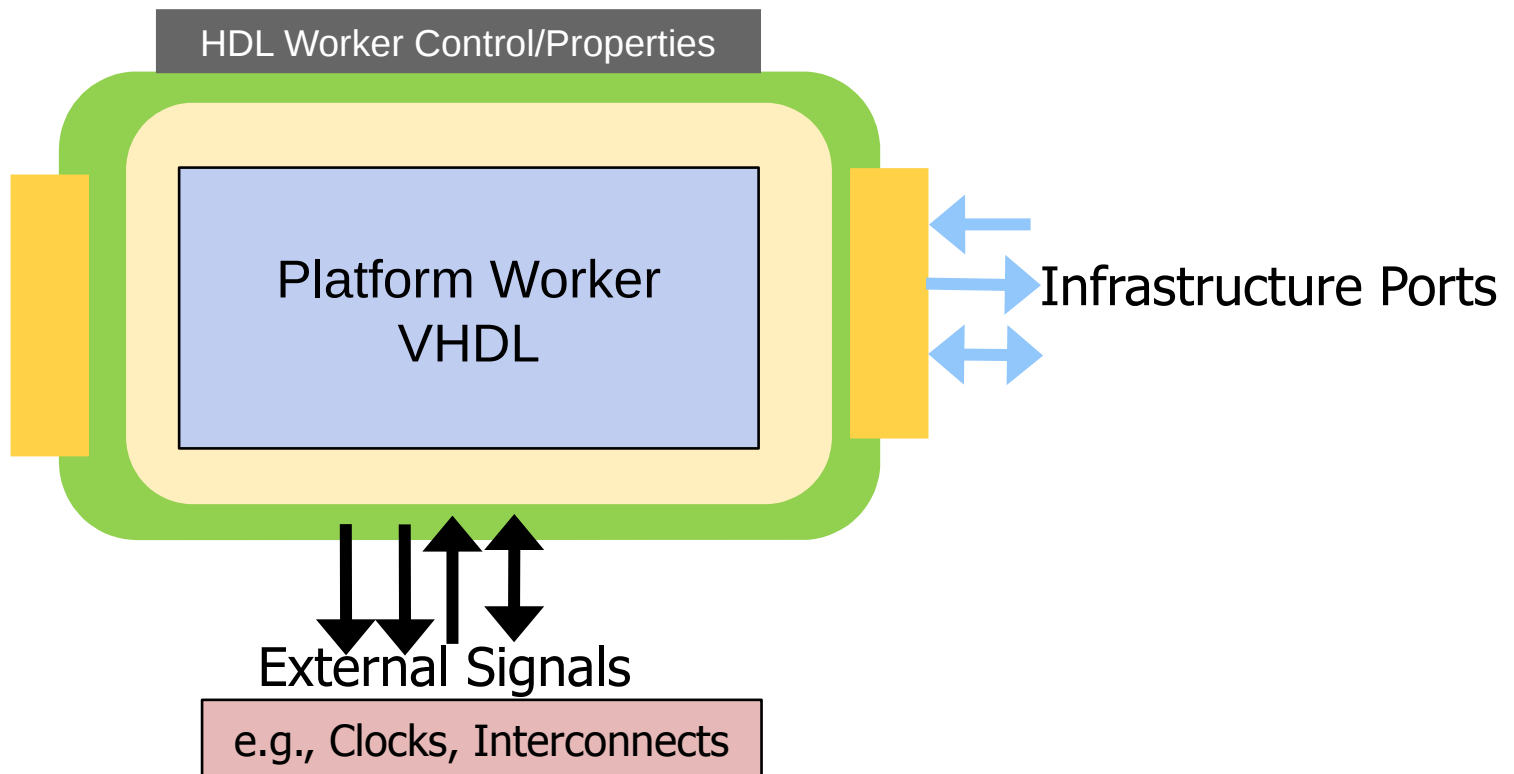


FPGA Device Worker OWD XML Example

```
<!-- Example ADC worker -->
<HdlDevice language="vhdl" firststrawproperty='dc_regval' spec='qadc-spec'>
  <ControlInterface Timeout="1024"/> <!-- timeout long enough for the SPI
access -->
  <Property name='USE_CTL_CLK_p' type='bool' default='1'/>
  <Property name='source' type='enum' enums='adc,count,loopback' initial='1'/>
  <!-- Properties in registers -->
  <property name='dc_regval' type='uchar' volatile='true'/>
  <property name='rccal_lpfcal' type='uchar' volatile='true'/>
  <property name='dc_cntval' type='uchar' writable='true' volatile='true'/>
  <!-- Ports -->
  <StreamInterface Name="OUT" producer='true' DataWidth="32"/>
  <!-- Signals to/from the device -->
  <Signal Output="RX_CLK"/>
  <Signal Input="RX_IQ_SEL"/>
  <Signal Input="RXD" width="12"/>
  <Signal Input="RX_CLK_IN"/>
  <Signal Output="SEN"/>
  <Signal Input="SDO"/>
  <Signal Output="SDIO"/>
  <Signal Output="SCLK"/>
  <Signal Output="RESET"/>
</HdlDevice>
```

FPGA (HDL) Platform Worker

- A special type of HDL device worker
- Based on the "platform-spec" OCS, has properties
- As a device worker, has external hardware/pin signals via <signal>
- Has "infrastructure" ports to connect to generic infrastructure



FPGA Platform Worker Must Provide:

1. A suitable clock for the control plane
 - Currently this is widely used as the "default clock" for many purposes
2. A suitable clock for timekeeping
 - Typically the highest quality/stability
3. A path for an external processor to perform control access ops
 - Typically allow processor to perform memory mapped load/store access
 - Adapt this path to the OpenCPI "control plane master" interface.
4. A bus-mastering/DMA path for the system interconnect(s)
 - Typically allow OCPI generic infrastructure to be bus master.
 - Adapt this path to the OCPI generic "data plane" interface

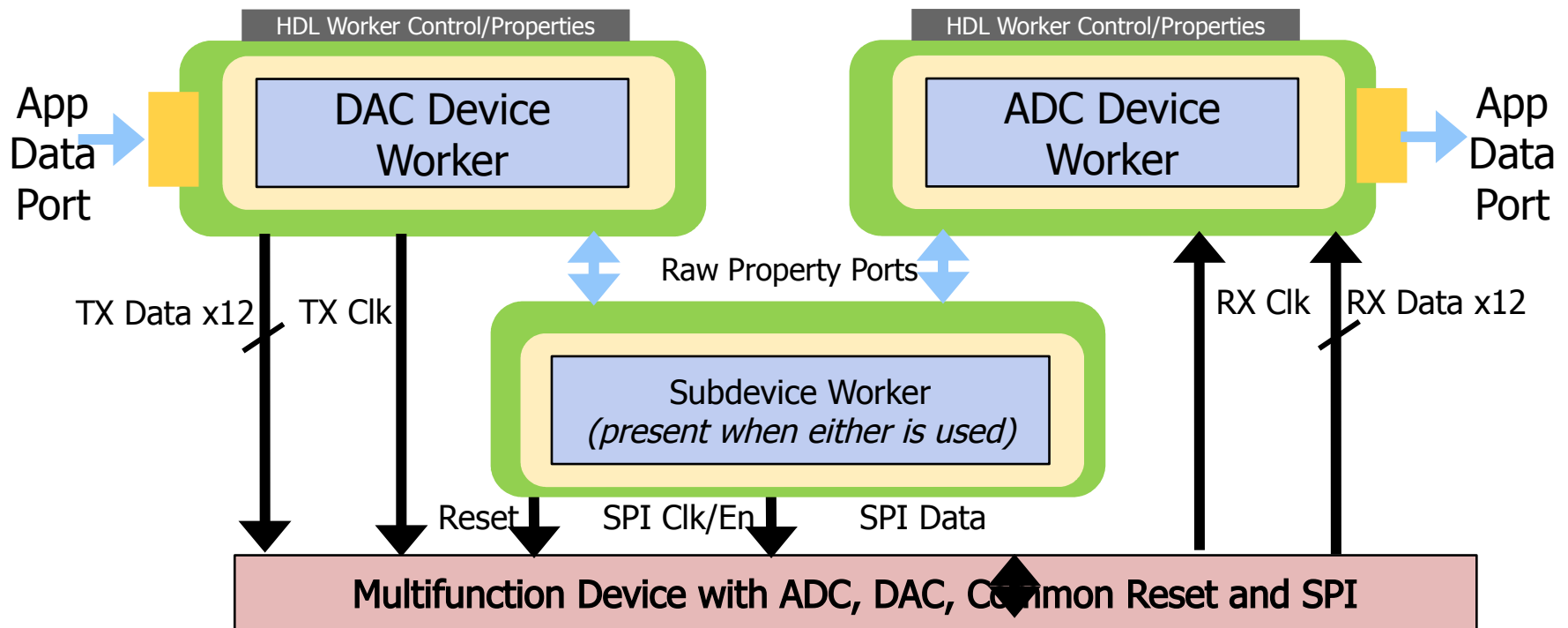
- Top level element is <HdlPlatform>
- Spec is "platform-spec"
- Declare infrastructure ports using these elements:
 - <Timebase> for timekeeping clock
 - <Metadata> for access to in-bit-stream compressed artifact metadata
 - <CpMaster> for access to generic control plane infrastructure
 - <unoc> or <sdp> for access to generic data plane infrastructure
- Declare attached devices using <device> elements
 - including any build parameters and settings required for this platform
- Declare available slots using <slot> elements.
 - including any non-standard slot pin signal names

Complications for Device Workers

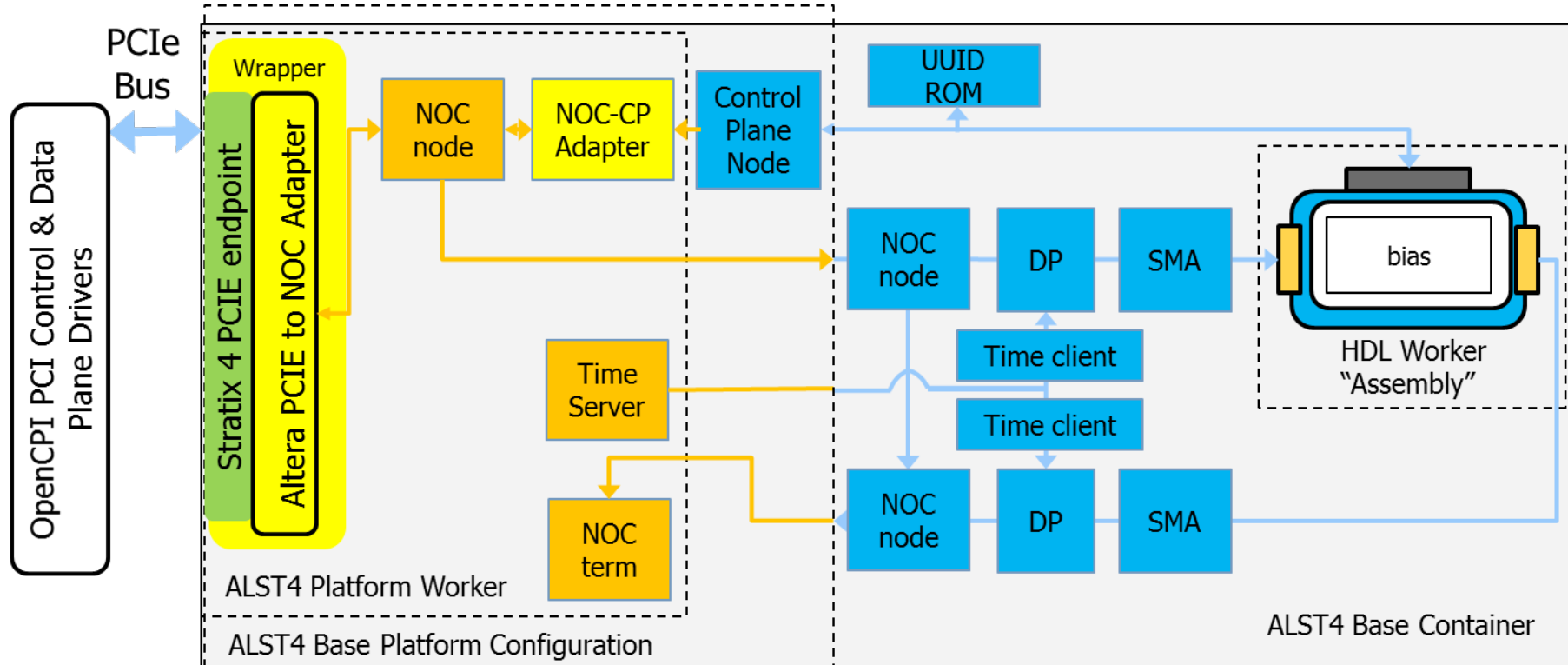
- Multiple devices may share some external signals
 - SPI and I2C configuration busses
 - Clocks, resets
- Complex multifunction devices should act as separate simple devices
 - Don't want to waste logic for unused functions
 - Want functional modularity of device workers to be consistent
- Solution is "subdevices", which are special device workers for:
 - Encapsulating signal sharing among device workers
 - Instantiated automatically when any of its supported devices are used.
 - May or may not have their own properties

Subdevice Worker Example

- ADC and DAC have separate data paths and clocks
- They share an SPI and master reset



- Platform modules can be found in the hdl/platforms directory
- They have implementation XML and source code
 - Just like other workers
- Responsible for instancing OpenCPI adapter logic and some generic infrastructure



SMA – Streaming to Message Adapter Component

DP – Data Plane Worker

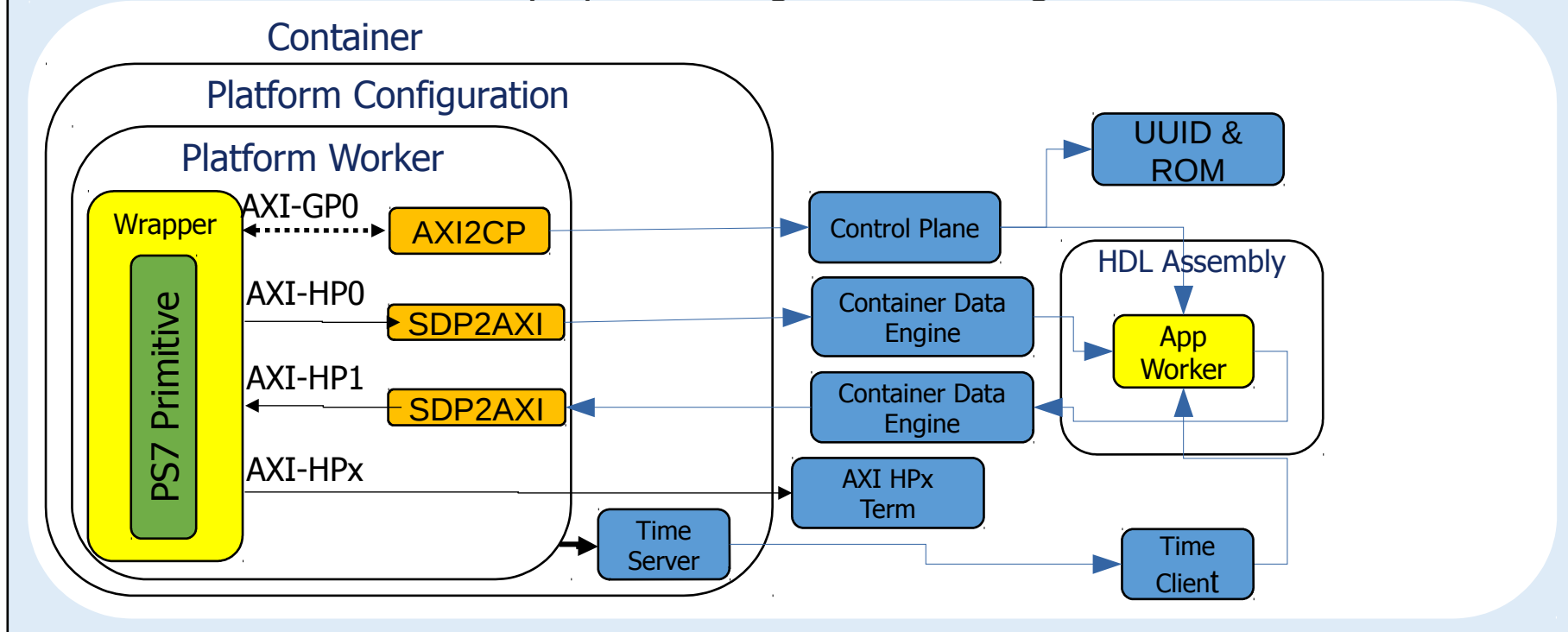
Generic Infrastructure included with OpenCPI – instanced by platform developer

Generic Infrastructure included with OpenCPI – automatically connected and instanced, as needed

Vendor supplied hard IP block

Custom code required for platform enablement

Zynq 7020 Programmable Logic



Generic Infrastructure included with OpenCPI – instanced by platform developer

Generic Infrastructure included with OpenCPI – automatically instanced and connected, as needed, by the code-generation tool

Vendor supplied hard IP block

Custom code required for platform enablement

Where Can I Find More Information?

- *OpenCPI Platform Development Guide*
 - https://opencpi.gitlab.io/releases/develop/docs/OpenCPI_Platform_Development_Guide.pdf
- *Matchstiq BSP Case Study*
 - https://opencpi.gitlab.io/releases/develop/docs/assets/Matchstiq_BSP_Case_Study.pdf
- *Matchstiq Platform Worker Data Sheet*
 - https://opencpi.gitlab.io/releases/develop/docs/assets/Matchstiq_Z1_Platform_Worker.pdf