OpenCPI - Open Component Portability Infrastructure

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

HDL Assemblies

Concepts and Terminology Review

- Concepts and Terminology Review
- HDL Assemblies

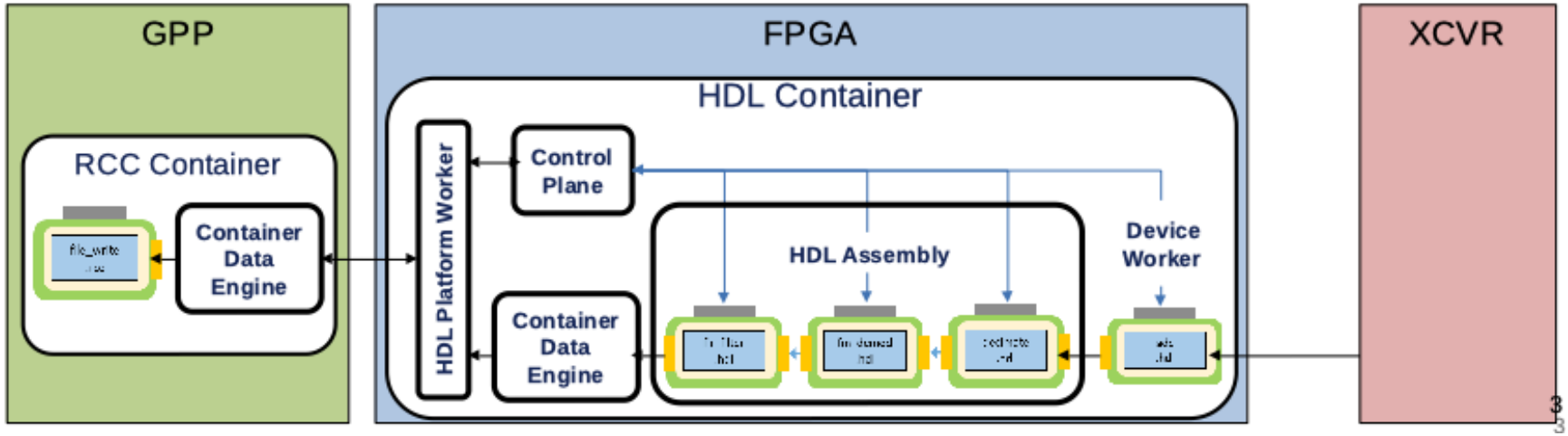
Review: App Worker Development Flow

1. Protocol (OPS): Use pre-existing or create new
2. Component (OCS): Use pre-existing or create new
3. Create new application worker (Modify OWD, Makefile, and source HDL/RCC code)
4. Build the application worker for the target device(s)
5. Create unit test ({component}-test.xml, generate, verify and view scripts)
6. Build unit test – Assembly/Container is generated/built here
7. Run unit test

Review: Building Blocks Terminology

Term: HDL Assembly
Definition: A fixed composition of HDL workers that can act as subset of a heterogeneous OpenCPI application.
Described by: HDL Assembly Description XML (OHAD), **NO VHDL**

```
<HdlAssembly>
  <connection name="in" external="consumer">
    <port instance="decimate" name="in"/>
  </connection>
  <instance component='decimate' connect='fm_demod'/>
  <instance component='fm_demod' connect='fir_filter'/>
  <instance component='fir_filter'/>
  <connection name="out" external="producer">
    <port instance="fir_filter" name="out"/>
  </connection>
</HdlAssembly>
```



Concepts and Terminology Review

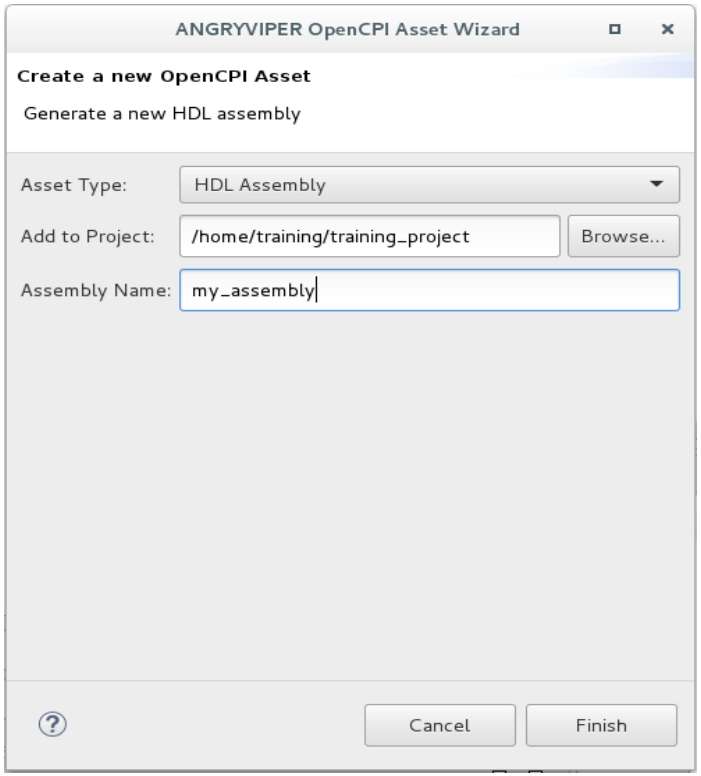
- Concepts and Terminology Review
- HDL Assemblies

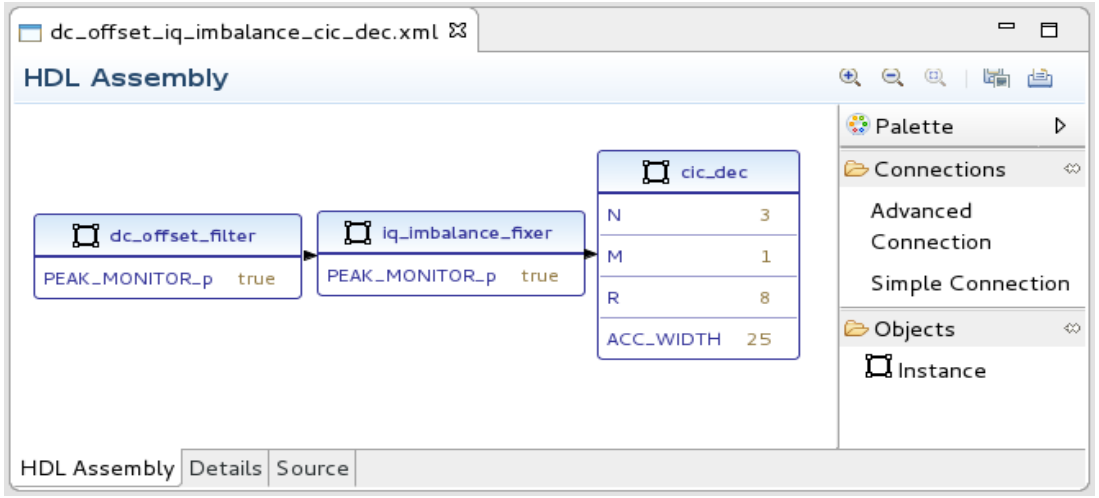
HDL Assemblies

- The OHAD XML provides metadata to the framework describing port interconnections and worker configurations (parameters) which, when built:
  - Generates structural HDL source code
  - Performs incremental compilation using FPGA vendor tools to produce a netlist for the target device
- Directory for the HDL assembly created at *hdl/assemblies/<my_assy>/*
- Assembly builds result in a *single* standalone artifact file per platform
  - *Executable* is the term used for a simulator platform
  - *Bitstream* is the term used for an actual physical FPGA
- The HDL assembly is combined with a platform worker to create an HDL container that is transformed into a bitstream/executable - *<filename>.bitz*

Creating an Assembly with the IDE

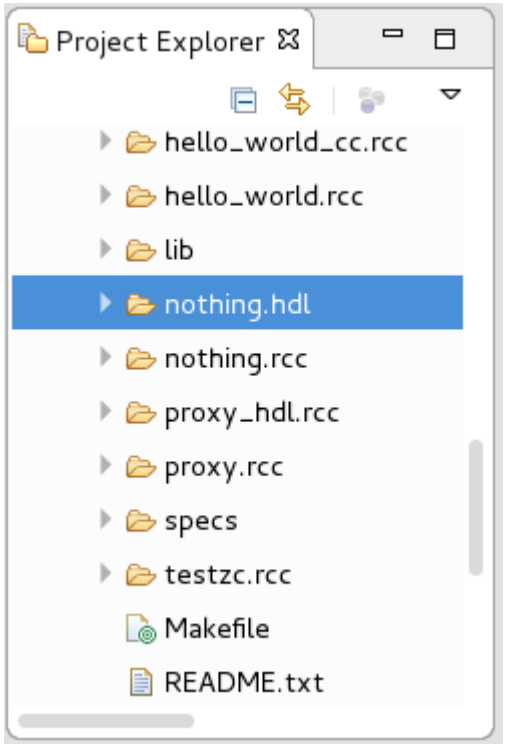
HDL assembly creation and building is supported in the IDE

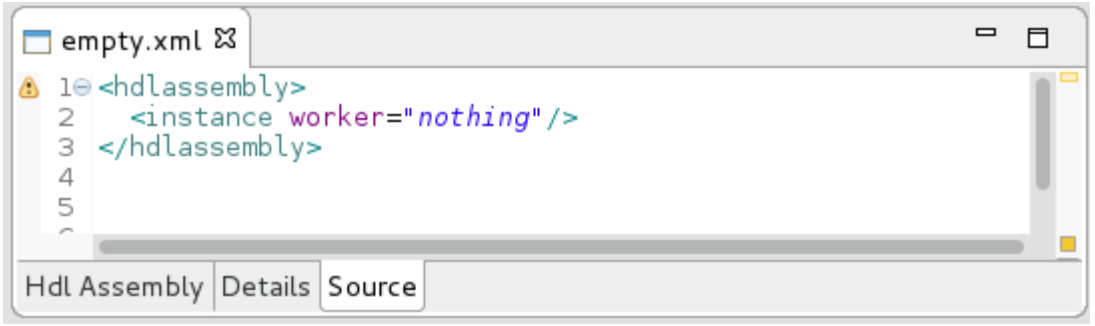




HDL Assembly OHAD Examples

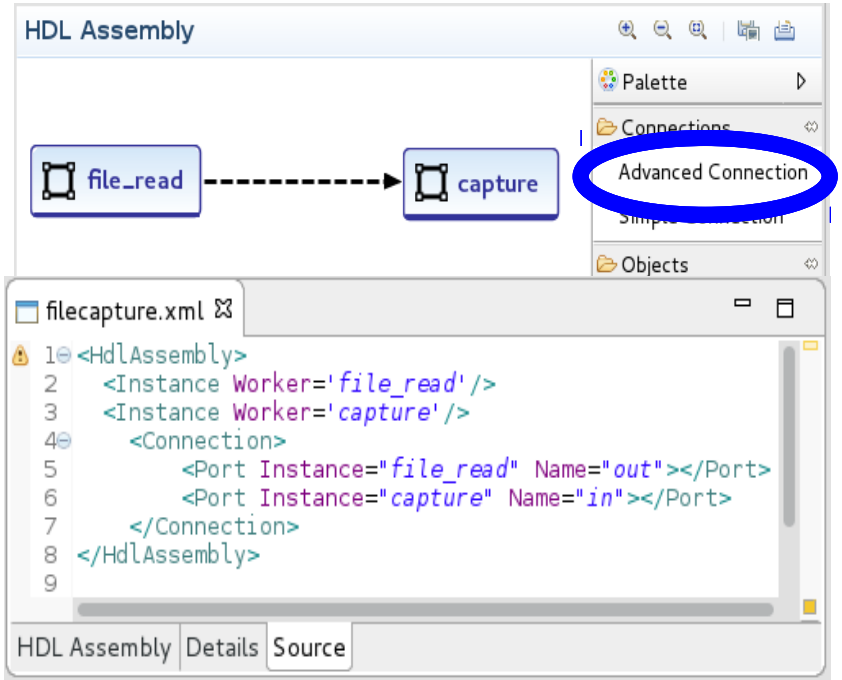
Worker instances reference HDL workers in a component library (worker's OWD name)

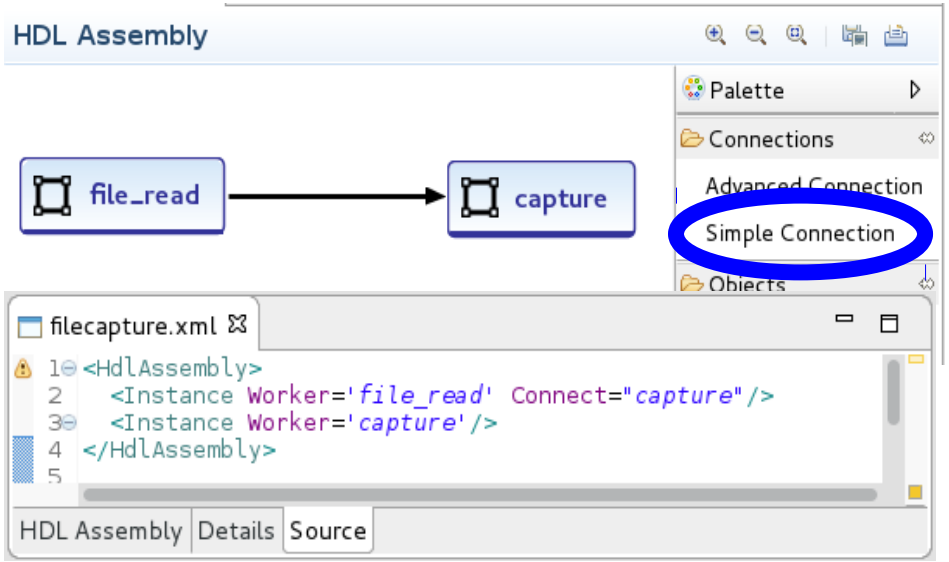




HDL Assembly OHAD Examples

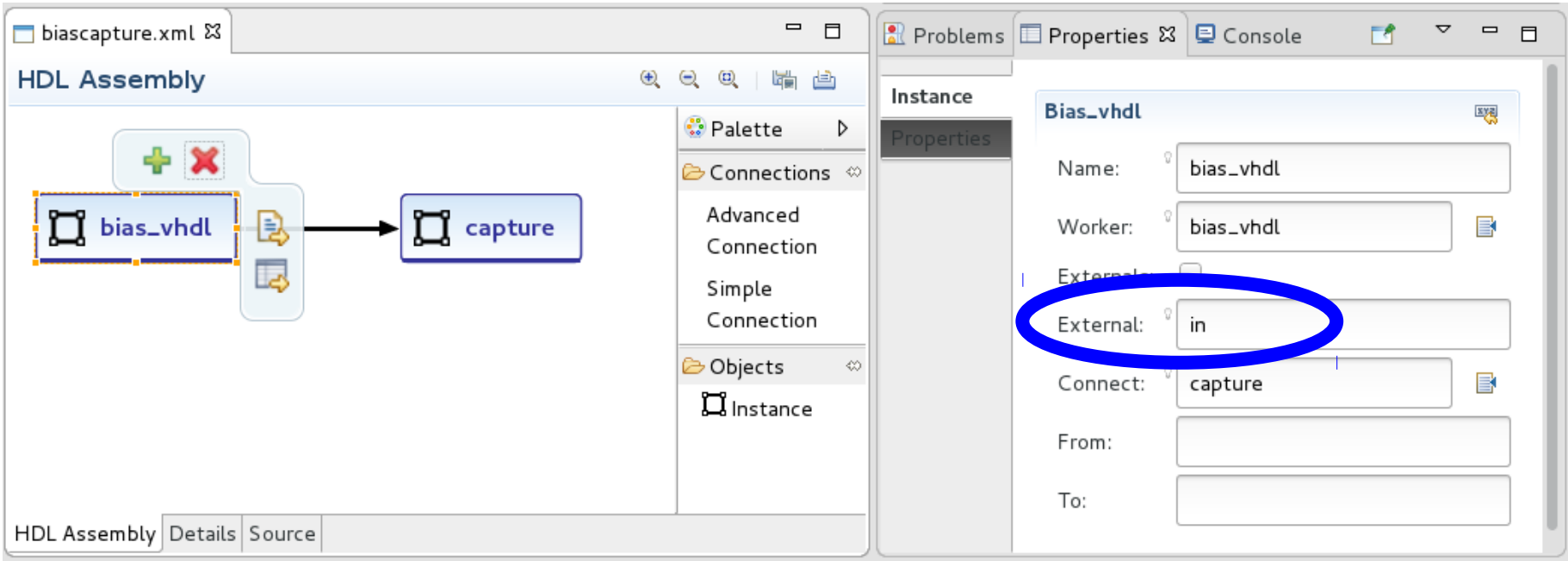
- Advanced uses *connection* XML elements
  - Necessary when there are multiple input/output ports, otherwise optional
- Simple uses a *connect attribute* of the instance, indicating the instance's *only* output is connected to the *only* input of another instance





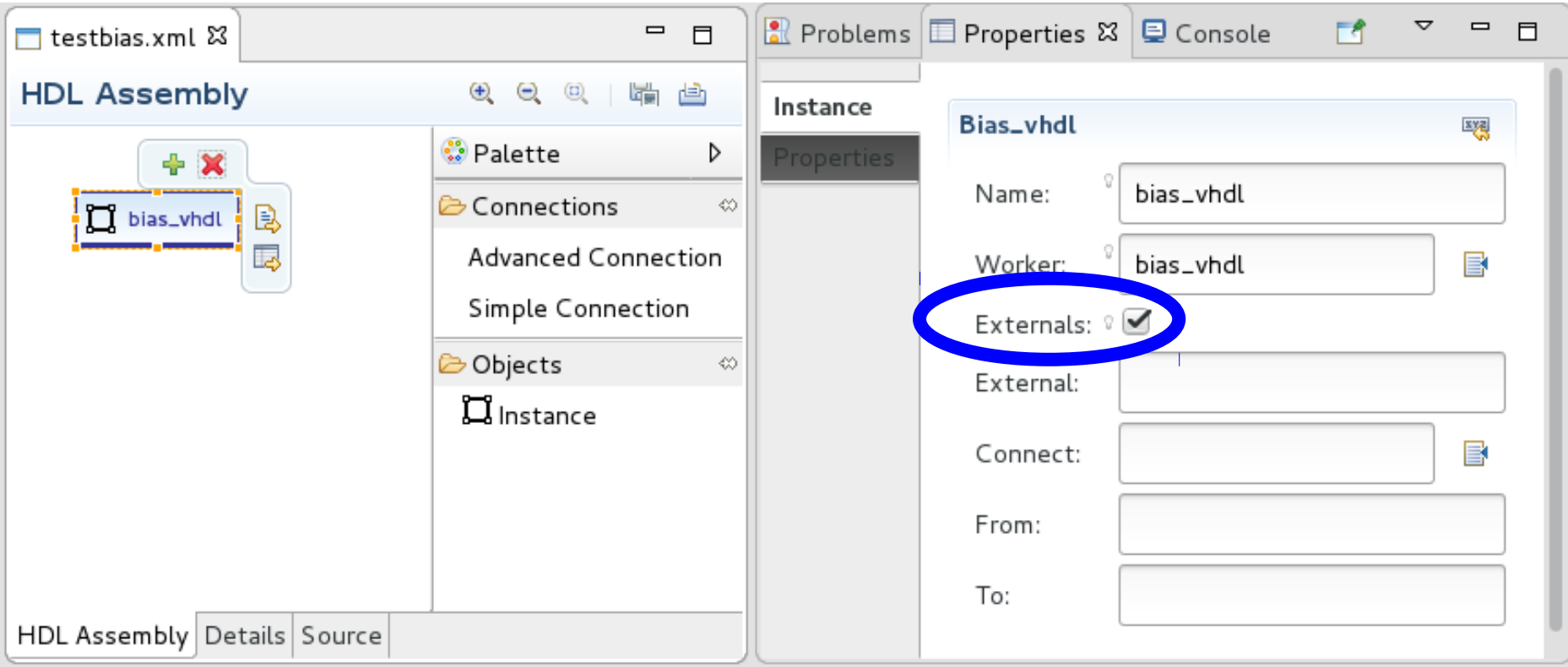
HDL Assembly OHAD Examples

- External ports in assemblies describe connections to/from the assembly
  - When the external port name is the same as the worker's port name, the **external** attribute of the instance may be used



HDL Assembly OHAD Examples

- The ***externals*** attribute is used to specify that all unconnected worker ports should be made external ports



HDL Assembly Makefile

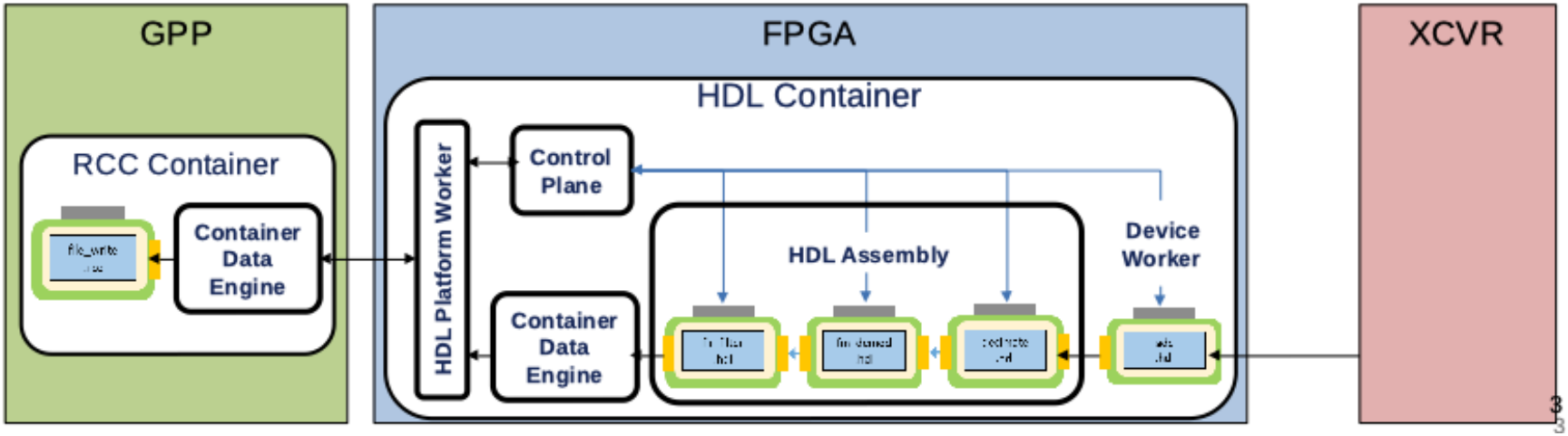
- Located in *hdl/assemblies/<my_assy>/*
- Makefile contains *include $(OCPI_CDK_DIR)/include/hdl/hdl-assembly.mk*
- Optional makefile variables
  - *ComponentLibraries*
    - A list of component libraries to find the workers in the HDL assembly
    - Declared in the HDL Assembly(ies) Makefile, or per project in the Project.mk file
  - *OnlyPlatforms*
    - An exclusive list of platforms for which this assembly (and default containers) should be built
  - *ExcludePlatforms*
    - A list of platforms for this assembly that should NOT be built
  - *Containers/DefaultContainers*
    - A list of containers for building the assembly
- Built using the *ocpidev build--hdl-platform <platform>* to identify the HDL platform(s) to target

HDL Assemblies Makefile

- Located in *hdl/assemblies/*
- Makefile contains *include $(OCPI_CDK_DIR)/include/hdl/hdl-assemblies.mk*
- Optional makefile variables
  - *ComponentLibraries*
    - A list of component libraries to find the workers in the HDL assembly
    - Declared in the HDL Assembly(ies) Makefile, or per project in the Project.mk file
  - *OnlyPlatforms*
    - An exclusive list of platforms for which this set of assemblies (and default containers) should be built
  - *ExcludePlatforms*
    - A list of platforms for this set of assemblies that should NOT be built
  - *Assemblies/ExcludeAssemblies*
    - A list of assemblies to build or exclude from being built, respectively (all built by default)
  - *Containers/DefaultContainers*
    - A list of containers for building the assembly
- Built using the *ocpidev build --hdl-platform <platform>* to identify the HDL platform(s) to target

HDL Container

- Written in XML (Currently not supported by the IDE)
- Top-level module that implements the assembly on the platform
- Metadata for the framework to compile an Application Assembly (netlist) + Platform Configuration (netlist) + additional optional Device Workers (netlist) into a bitstream for execution on an FPGA
- Separating the assembly from the container keeps the assembly portable
- Assembly's external input/output connections are unspecified until implemented in a container on the platform

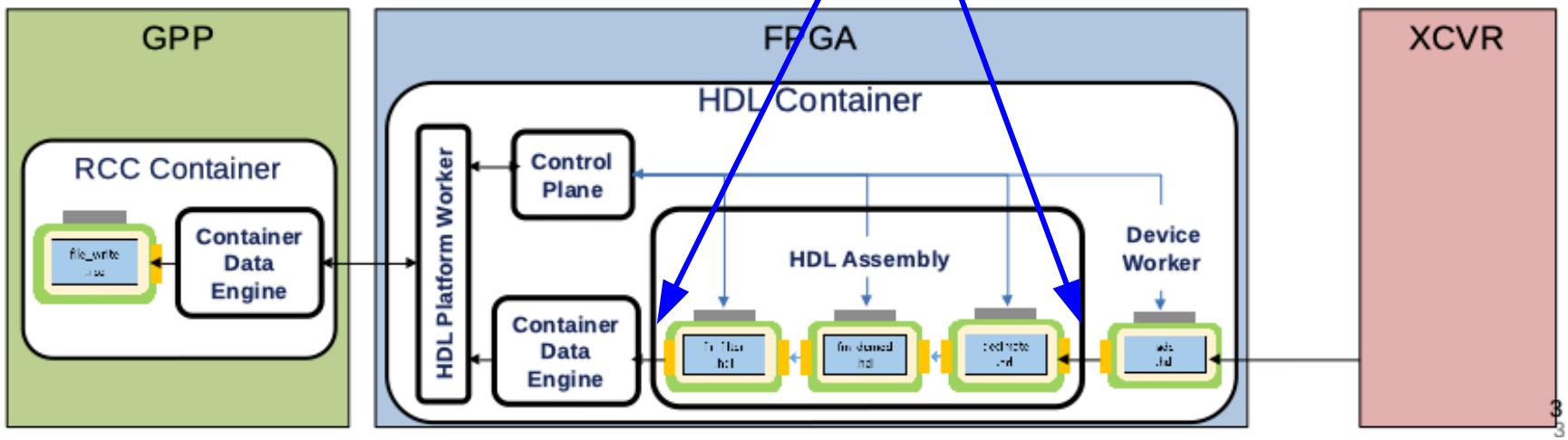


HDL Container (cont.)

- Written in XML (Currently not supported by the IDE)

```
<HdlContainer Platform="matchstiq_z1">
  <Connection External="in"  Port="out" Device="lime_adc"/>
  <Connection External="out" Interconnect="zynq"/>
</HdlContainer>
```

Describes connections external to assembly



HDL Assemblies Build Process

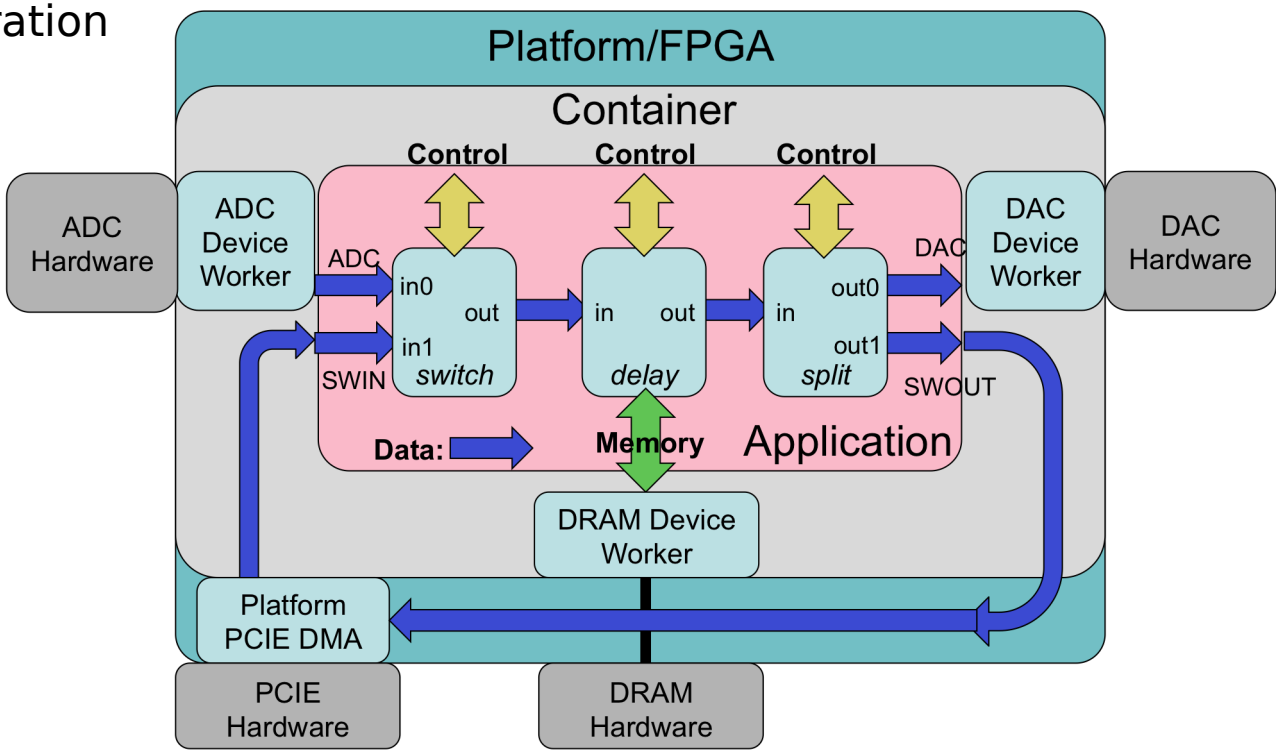
- Steps taken to go from the HDL assembly OHAD to a bitstream/executable
  1. Describe the assembly in XML, specifying workers and connections
  2. Select the platform configuration that the assembly will be implemented on
  3. Specify how the assembly's external ports connect to the platform (e.g. to an interconnect like PCIe for off-platform connections, or to local devices). This is "defining the container" **(currently not supported by the IDE)**
  4. Generate the bitstream/executable
- **In many cases, steps 2 & 3 may use defaults**
- Under the hood, the scripts and tools take the following steps
  1. **Generate** the Verilog/VHDL code that structurally implements the assembly
  2. Build/synthesize the assembly module, that has some "external ports"
  3. **Generate** the Verilog/VHDL container code that structurally combines the assembly and the platform configuration, as well as any necessary adapters and device workers
  4. Build/synthesize the container code, incorporating the assembly and platform configuration. This is the top level module
  5. Run the final tool steps to build the bitstream (map, place, route, etc.)

HDL Default Container

- *DefaultContainers* (optional)
  - Connects all assembly external ports to the platform's interconnect
  - Lists platform configurations for which default containers should be automatically generated for this assembly
  - Items in the list are <platform> or <platform>/<configuration>
  - If defined as empty (DefaultContainers=) there are no default containers for this assembly
  - If not defined (the default) will generate default containers for whatever platform the assembly is built for
    - This is the base platform configuration with no device workers

HDL Containers

- *Containers*
  - Specifies a list of containers that should be built in addition to those indicated by DefaultContainers
  - Does not suppress building default containers (must set DefaultContainers=)
  - Used to add device workers to the container that are not part of the platform configuration



HDL Containers (cont.)