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

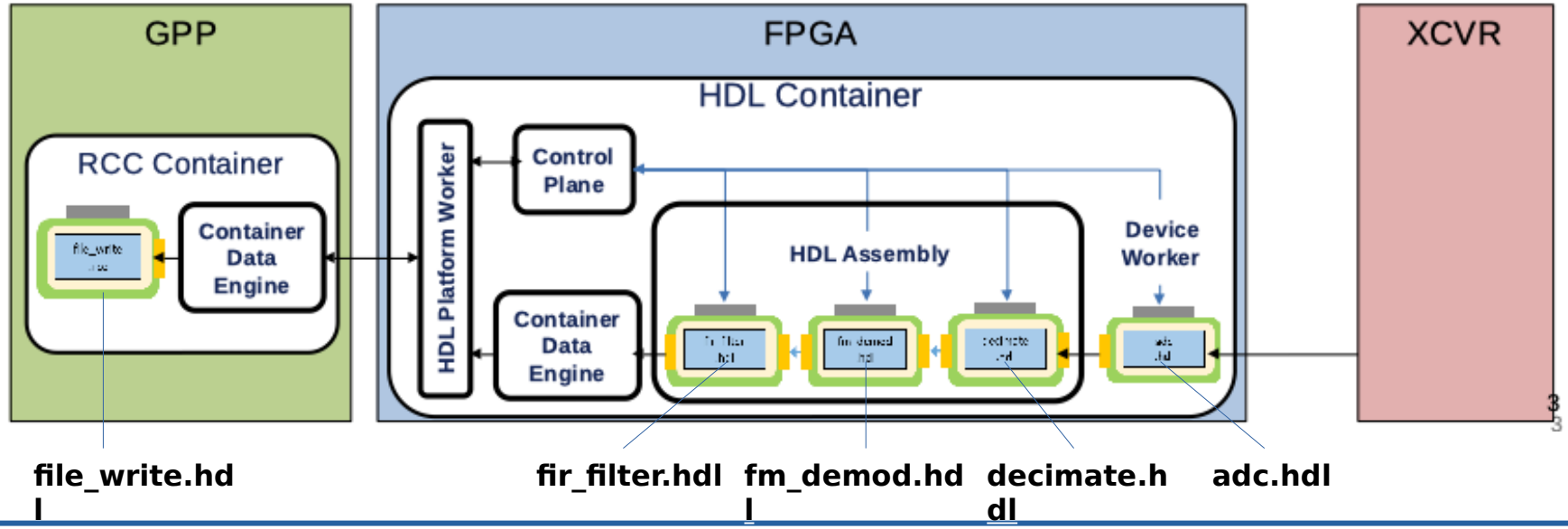
Review: App Worker Development Flow

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 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 OpenCPI HDL Assembly Description (OHAD) 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 HDL Assemblies

With ocpidev:

```
ocpidev create hdl assembly demo_assembly
```

Files created:

```
hdl/assemblies/
|-- assemblies.xml
|-- demo_assembly
    |-- demo_assembly.xml
```

Files of interest:

assemblies.xml: XML for all assemblies
demo_assembly.xml: OHAD that defines the assembly's:
- HDL worker instances
- Property/parameter settings
- Connections
- External ports

HDL Assembly OHAD Structure

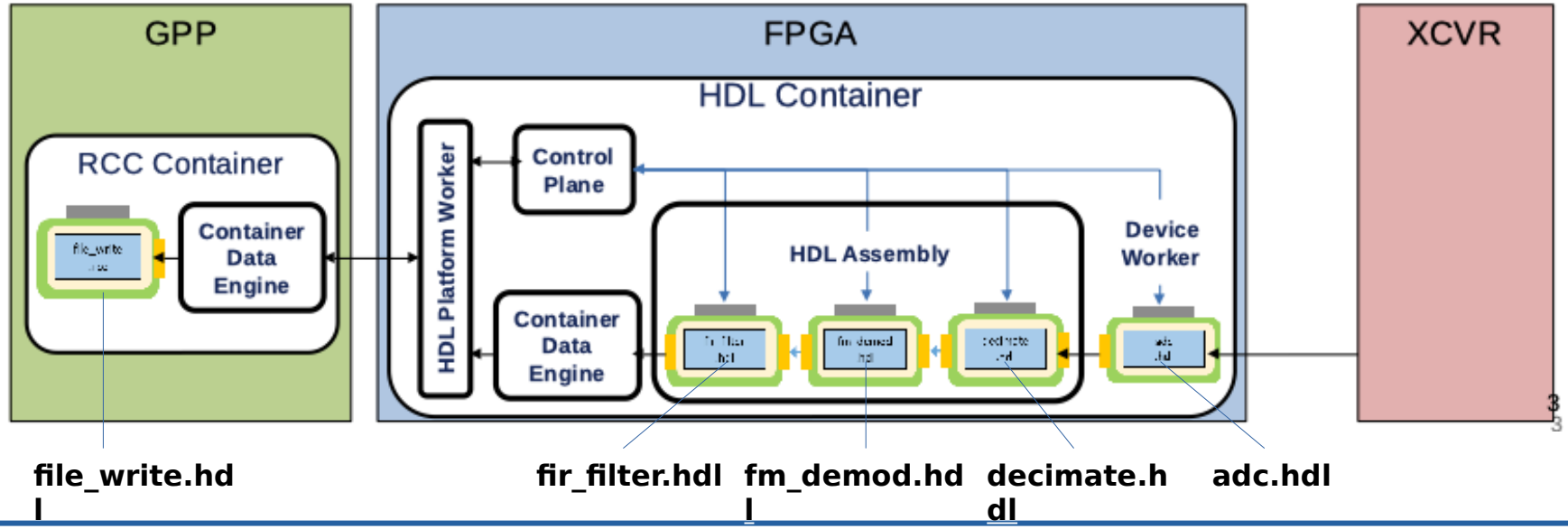
- Worker instances reference HDL workers in a component library (worker's OWD name)
  - Instance names are optional; when not specified, they are either the same as the worker name, without any package prefix (when there is only one instance of that worker in the assembly), or the worker name followed directly by a zero-based decimal ordinal (when there is more than one instance of the same worker)
- **`connect`** elements define connections among worker data ports
- The **`connect`** *attribute* of an instance indicates the instance's *only* output is connected to the *only* input of another instance
- External ports, where the data is flowing into or out of the assembly itself, can be specified with:
  - The **`external`** element, when the external port name is different from the worker port it's connected to
  - The **`external`** attribute of the instance, when the worker port name and external port name are the same
  - The **`externals`** boolean, to specify that all unconnected worker ports are external ports

HDL Assembly XML Attributes for Building

- Optional top-level XML attributes used when an HDL assembly is built
  - **`ComponentLibraries`**
    - A list of component libraries in which to find the workers to make the HDL assembly
    - Can also be set in the hdl/assemblies directory XML file "assemblies.xml" to apply to all the assemblies in that directory or in the project's "Project.xml" 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
- See the *OpenCPI HDL Development Guide* for details

HDL Container

- Written in XML
- 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

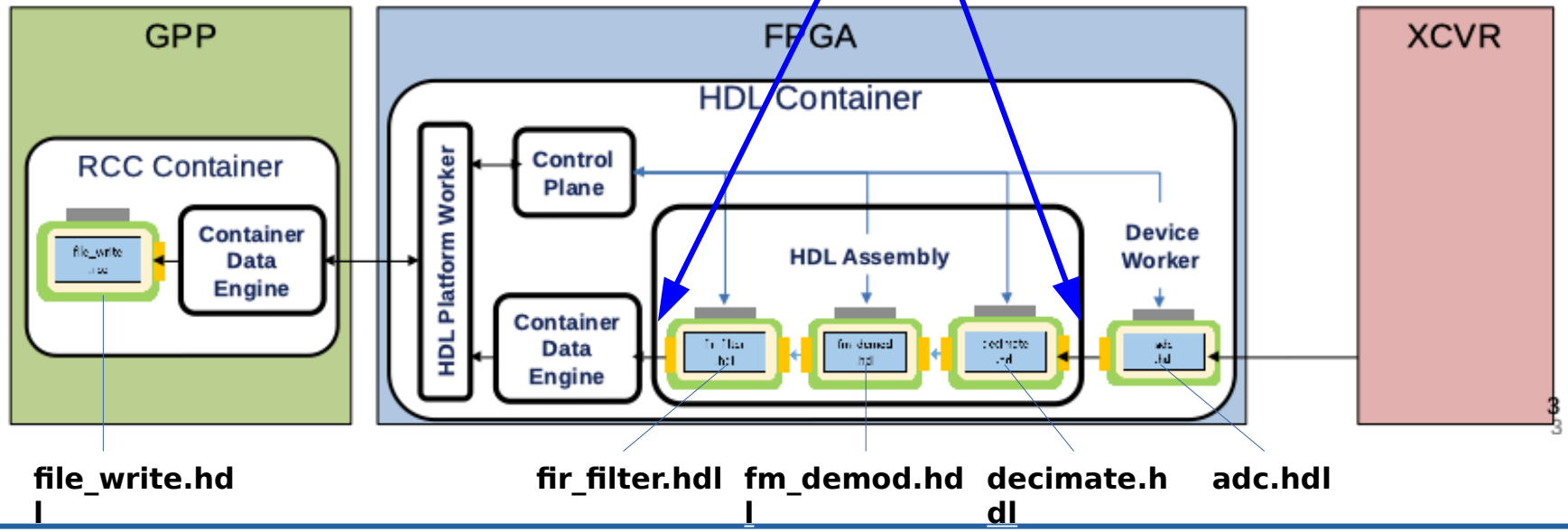


HDL Container (cont.)

- Written in XML

```
<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

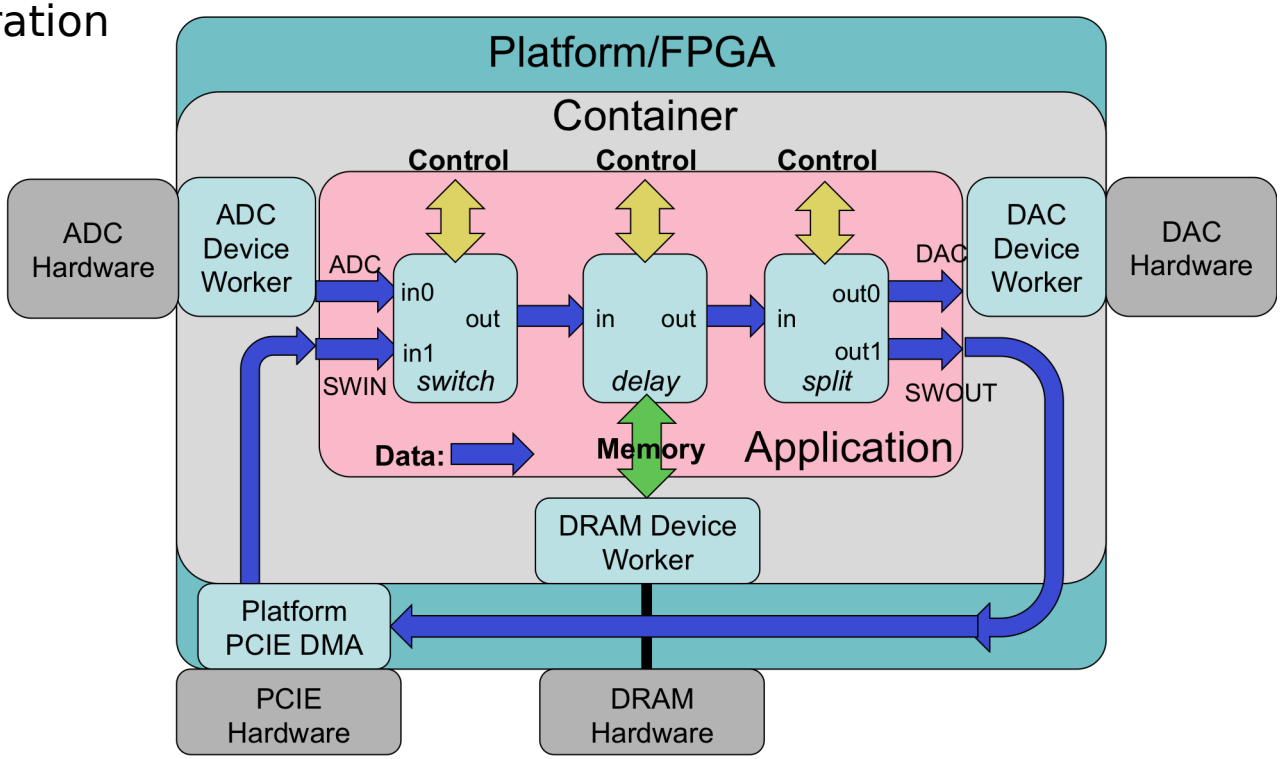
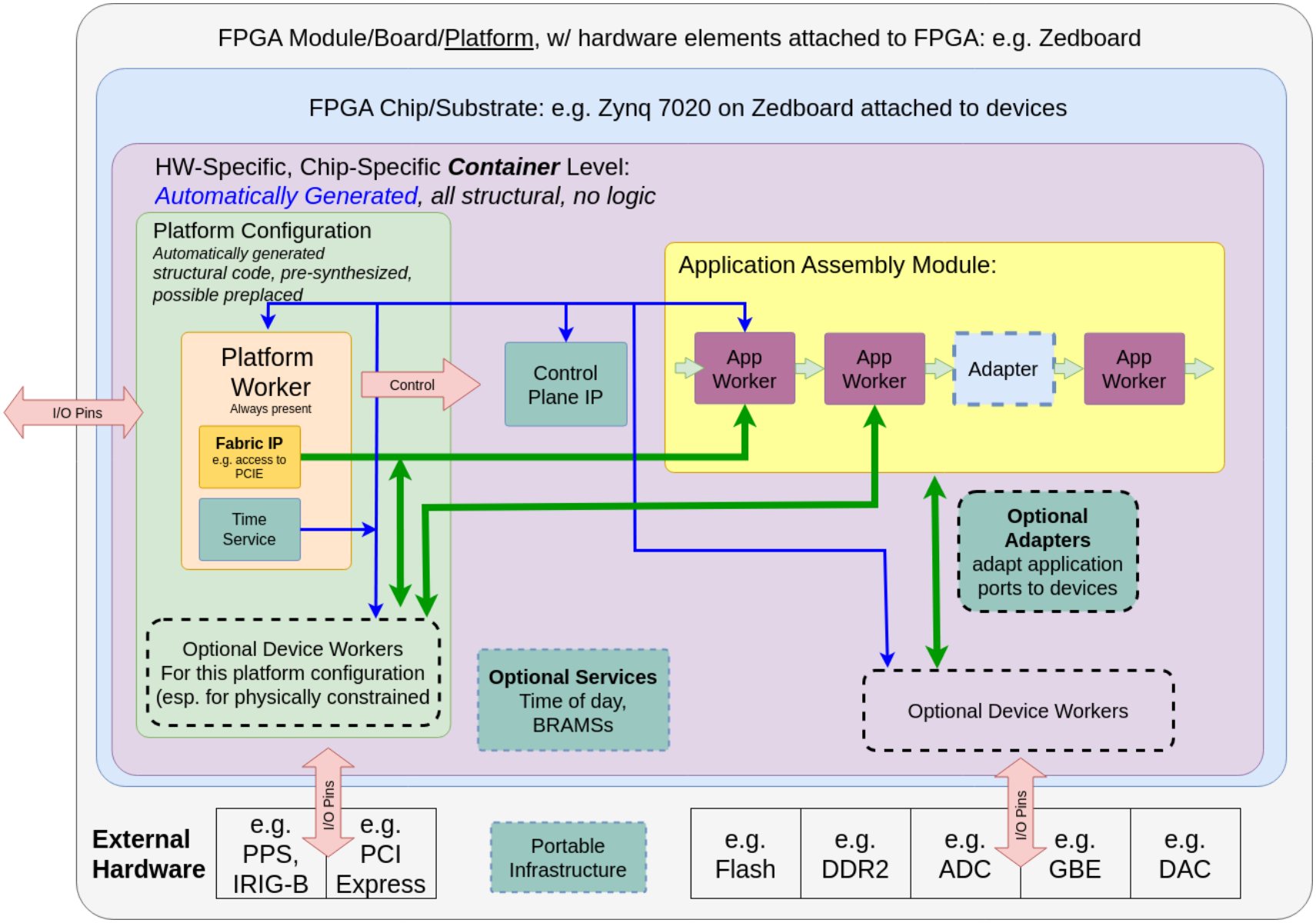
- Steps taken to go from the HDL assembly OHAD to a bitstream/executable
  1. Describe the assembly in XML, specifying workers and connections
  2. Select the platform configuration that the assembly will be implemented on
  3. Specify how the assembly's external ports connect to the platform (e.g. to an interconnect like PCIe for off-platform connections, or to local devices). This is "defining the container"
  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 top-level assembly XML attribute)
  - 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`** (optional top-level assembly XML attribute)
  - 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.)