OpenCPI - Open Component Portability Infrastructure

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

HDL Primitives

HDL Primitives

- HDL primitives and primitive libraries are the preferred way in OpenCPI to store/present reusable HDL for use in multiple workers
- Using HDL primitives in a worker or in another primitive requires mentioning them in the using primitive's XML or worker's OWD file with the **`Libraries`** or **`Cores`** attributes
- May also be referenced in project and component library XML files with the **`HdlLibraries`** attribute to make them available to all HDL workers in the project/component library
- Using HDL primitives is not required, but is recommended practice
  - Alternatively, if used in a single worker, source could be added to the worker's directory and to the **`SourceFiles`** attribute in the worker's OWD
- Must ***not*** have the same name as an HDL application worker

HDL Primitives (cont.)

- HDL Primitive Library
  - Collection of modules
  - Built from HDL source code (ex: FIR, CIC, etc.)
- HDL Primitive Core
  - Single modules built from HDL source code or generated by vendor tools
    - Vendor dependent (ex. Xilinx/Coregen or Vivado, Altera/MegaWizard)
  - Prebuilt Cores from third party (.qxp/.edf/.ngc)
- Primitives may depend on each other
  - But *circular* dependencies are ***not*** supported
  - Primitive libraries depend on primitive libraries
    - When primitive libraries depend on each other, use the **`Libraries`** attribute in the "primitives.xml" file in the hdl/primitives directory
    - When one primitive depends on another, use the top-level **`Libraries`** attribute in the primitive's XML file (e.g. "<myprimitive>.xml")
  - The **`ocpi.core`** project depends on primitive libraries

Creating HDL Primitive Libraries

With ocpidev:

```
ocpidev create hdl primitive library my_prims
```

- Directory for the primitive library created at hdl/primitives/<my_prims>/
- Libraries build in per-target directories named "target-<hdl-target>"
- Library must include the VHDL file <libname>_pkg.vhd used for component declarations and unique type declarations
- Can have multiple *_pkg.vhd files and separate package body *-body.vhd files

Creating HDL Primitive Libraries (cont.)

- VHDL package name doesn't have to be the same as the library's name
- Ex: Single package for a primitive library mylib
  - library mylib; use mylib.mylib.all;
- Ex: Multiple packages in the primitive library mylib; different package names
  - library mylib; use mylib.mypkg.all;
- `SourceFiles` attribute
  - used to define build dependency order within a primitive library
  - required when a multi-level directory structure is used within the primitive library
- Log output of tools found in target-<hdl-target>/<libname>-<tool>.out
- Exportable results for all primitive libraries & cores found in hdl/primitives/lib

Creating HDL Primitive Cores

With ocpidev:

```
ocpidev create hdl primitive core my_cores
```

- Directory for the primitive core created at hdl/primitives/<my_cores>/
- Can be a mix of source and prebuilt files
  - Pre-synthesized core files (Xilinx Vivado *.edf, Xilinx ISE *.ngc or Altera *.qxp)
- VHDL instantiation must have a package file <corename>_pkg.vhd
- Verilog instantiation must have a "black box" empty module definition file <corename>_bb.v
- Optional XML attributes
  - **`Top`** (**`ocpidev -M`**) - Specifies the top module name of core when different than **`ocpidev`** name <my_core>
  - **`PreBuiltCore`** (**`ocpidev -B`**) - Specifies a file that is not source code