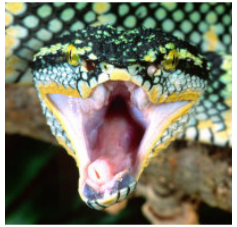
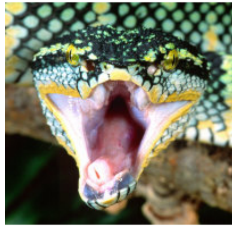


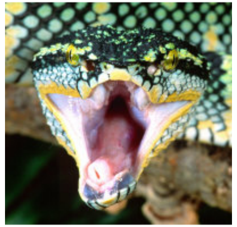
Lab 1:
OpenCPI Application Development

Objectives



1. Create FSK loopback (FPGA internal) OpenCPI Application XML (OAS) using the IDE
2. Run application on Ettus E310 hardware

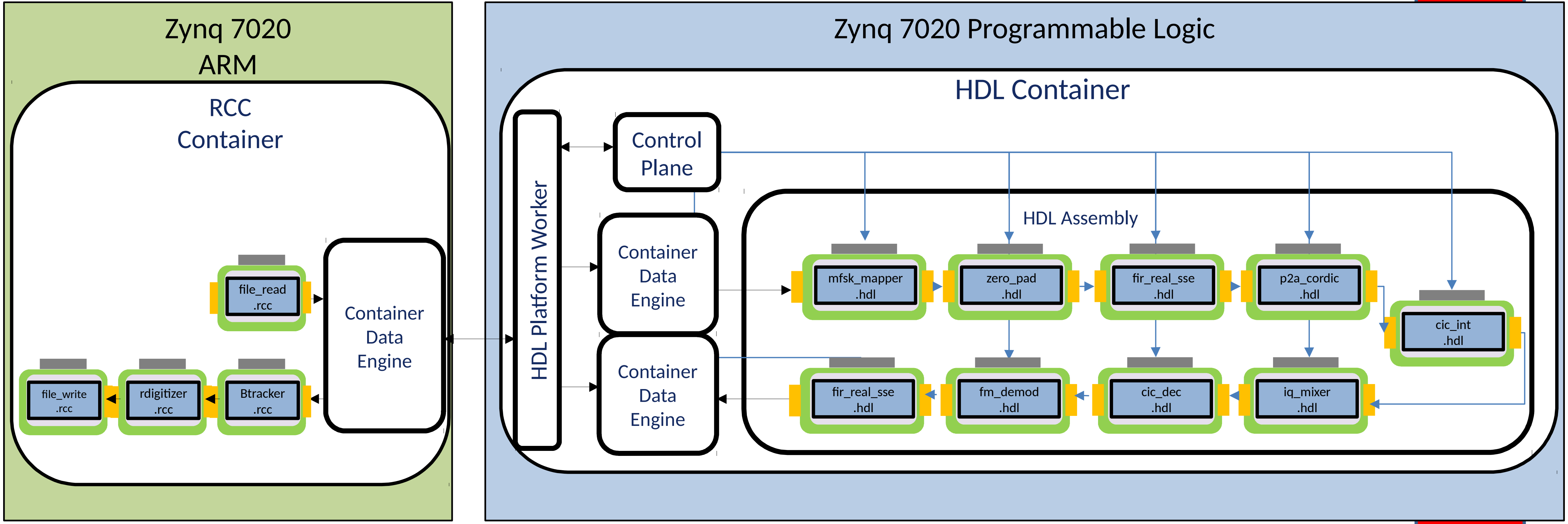
Overview



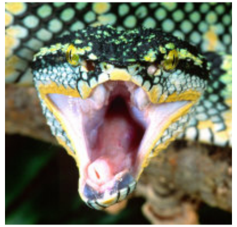
- A common use case for OpenCPI is the reuse of components from multiple libraries to construct applications for heterogeneous systems
- An OpenCPI Application Specification (OAS) XML describes the connections and initial property settings of the components
    - The ANGRYVIPER (AV) IDE helps generate this XML file graphically
- The generated XML is used by the `ocpirun` utility program during the execution of the application on a platform

Overview

- The reference application performs FSK modulation/demodulation
  - Modulation
    - Read Input File → FSK Symbol Mapper → Zero-pad → Pulse Shape → FM Modulate → Interpolate
  - Demodulation
    - Decimate → Demodulate → Filter → Baud Track → Digitize → Write Output File



Using the ocpirun utility

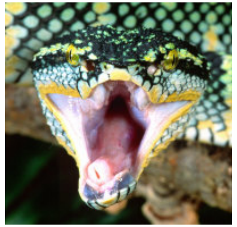


- The utility program ocpirun provides a simple way to execute applications
- Usage is: ocpirun app.xml
  - app.xml is a OAS file like the one which will be generated with the IDE in this lab
- The arguments passed to ocpirun can specify how the application is run

Option	Letter	Description
Dump	d	Dump all readable properties after initialization, and again after execution, to stderr.
Verbose	v	Be verbose in describing what is happening.
Log Level	l	For this execution, set the OpenCPI log level to the given level. 8 and 10 are commonly used.
Time	t	Stop execution after this many seconds. This is useful when there is no definition of "done" for the application.

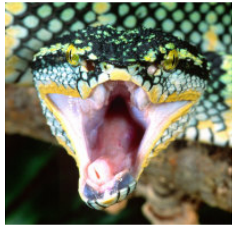
More detail on ocpirun can be found in the **OpenCPI Application Development Guide** document

Application Development Flow

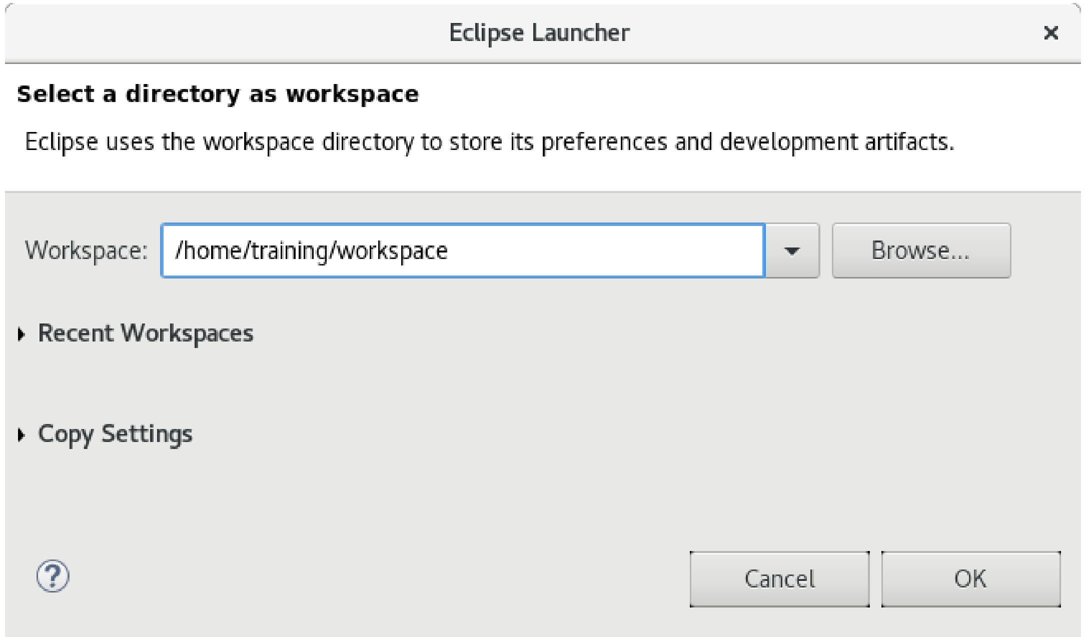


1. Add components to the OAS
2. Specify non-default properties for the components
3. Make connections between the components
4. Setup deployment platform
5. Run and test the application

Step 1



- Start AV IDE and set the default workspace to:
  - /home/training/workspace
  - Note: Don' t deviate from this path, this will be used in the remainder of the labs.
- Exit the welcome screen
- Launch the "perspective"
  - Window → Perspective → Open Perspective → Other…
  - Choose "ANGRYVIPER Perspective"



Eclipse Launcher

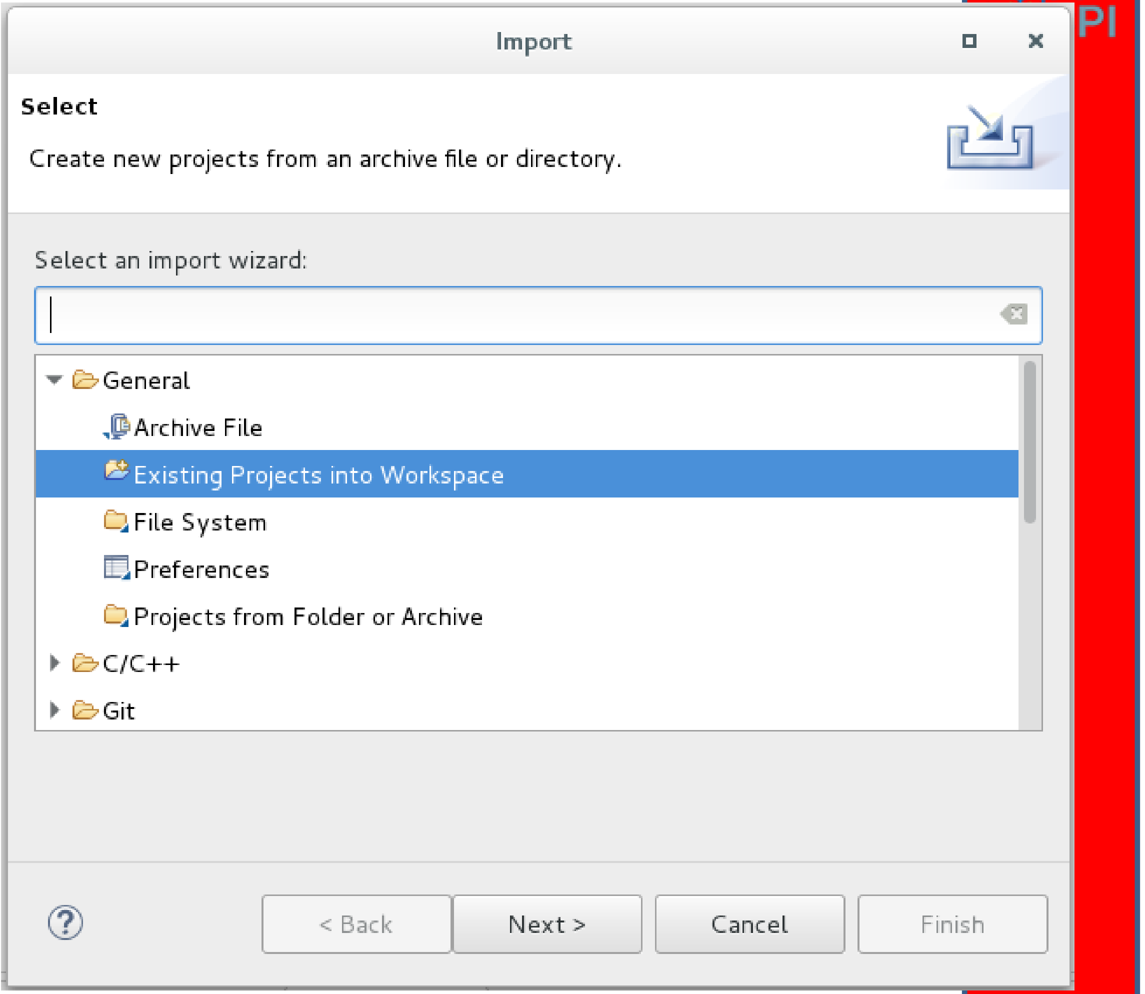
Select a directory as workspace

Eclipse uses the workspace directory to store its preferences and development artifacts.

Workspace: /home/training/workspace

Step 2

- Import pre-built projects:
  - **core:** basic components(read and write files)
  - **assets:** various components libraries
  - **assets_ts:** components for timestamping
  - **bsp_e310:** unique device workers, card definition and the E310 BSP
- Pre-built projects are located at:
  - ~/training/ocpi_projects/core
  - ~/training/ocpi_projects/assets
  - ~/training/ocpi_projects/assets_ts
  - ~/training/ocpi_projects/bsp_e310
- To import project into eclipse:
  - File → Import…
    - "Existing Projects into Workspace"



Import

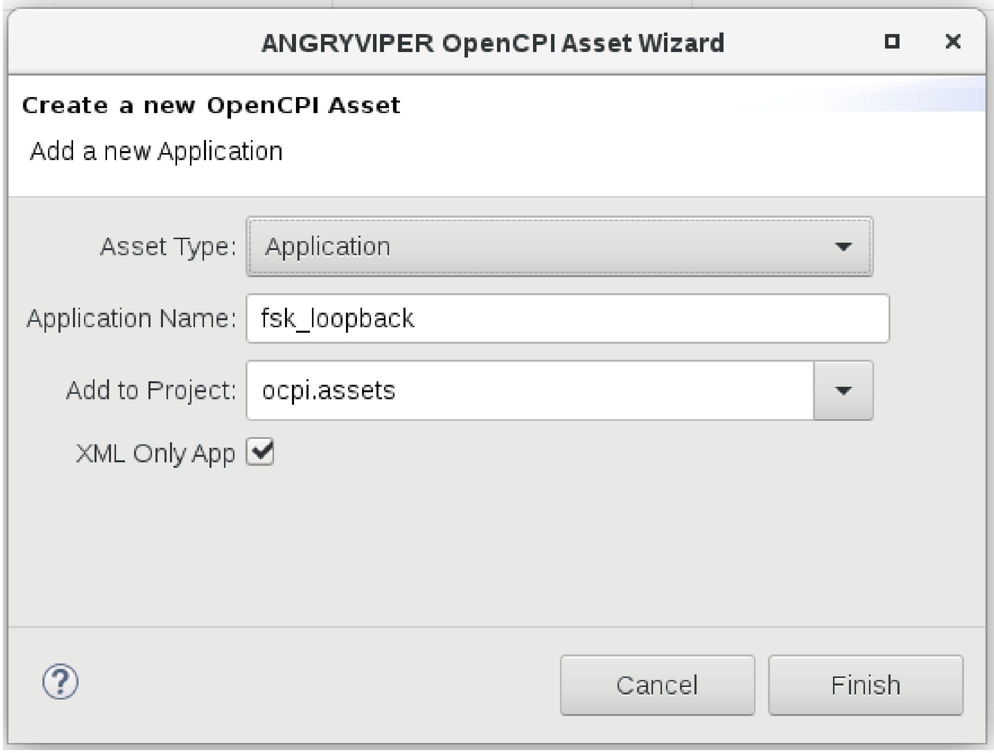
Select

Create new projects from an archive file or directory.

Select an import wizard:

Step 3

- Create new application in an existing project
- To create an application
  - In OpenCPI Projects, right click assets:
    - Asset Wizard
  - Asset Type: Application
  - Application Name: fsk_loopback
  - Add to Project: ocpi.assets
  - XML only: Yes


ANGRYVIPER OpenCPI Asset Wizard

Create a new OpenCPI Asset

Add a new Application

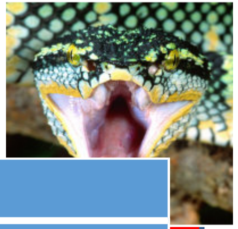
Asset Type: Application

Application Name: fsk_loopback

Add to Project: ocpi.assets

XML Only App ☑

Step 4



- **Delete the ocpi.core.nothing component**
  - This worker is automatically placed by the framework to ensure the generated OAS can be executed without editing the generated file
- **To add components**
  1. Within the Project Explorer tab and using the provided table, navigate into the "specs" directory of the appropriate Project:Library
  2. Drag spec file onto Application Editor
  3. Recommended: Name component

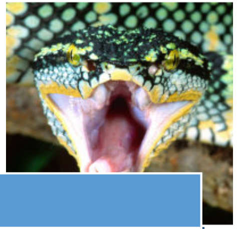
     *** There are 2 instances of fir_real_sse-spec.xml. To distinguish the instances, name one "tx_fir" and the other "rx_fir". Addressed in the following slides.**

Component Specs Required	
Name	**Project : Library**
file_read_spec.xml	Core : components
mfsk_mapper-spec.xml	Assets : components/comms_comps
zero_pad-spec.xml	Assets : components/util_comps
fir_real_sse-spec.xml*	Assets : components/dsp_comps
phase_to_amp_cordic-spec.xml	Assets : components/dsp_comps
cic_int-spec.xml	Assets : components/dsp_comps
complex_mixer-spec.xml	Assets : components/dsp_comps
cic_dec-spec.xml	Assets : components/dsp_comps
rp_cordic-spec.xml	Assets : components/dsp_comps
fir_real_sse-spec.xml*	Assets : components/dsp_comps
baudTracking-spec.xml	Assets : components/dsp_comps
real_digitizer-spec.xml	Assets : components/dsp_comps
file_write_spec.xml	Core : components

Step 5

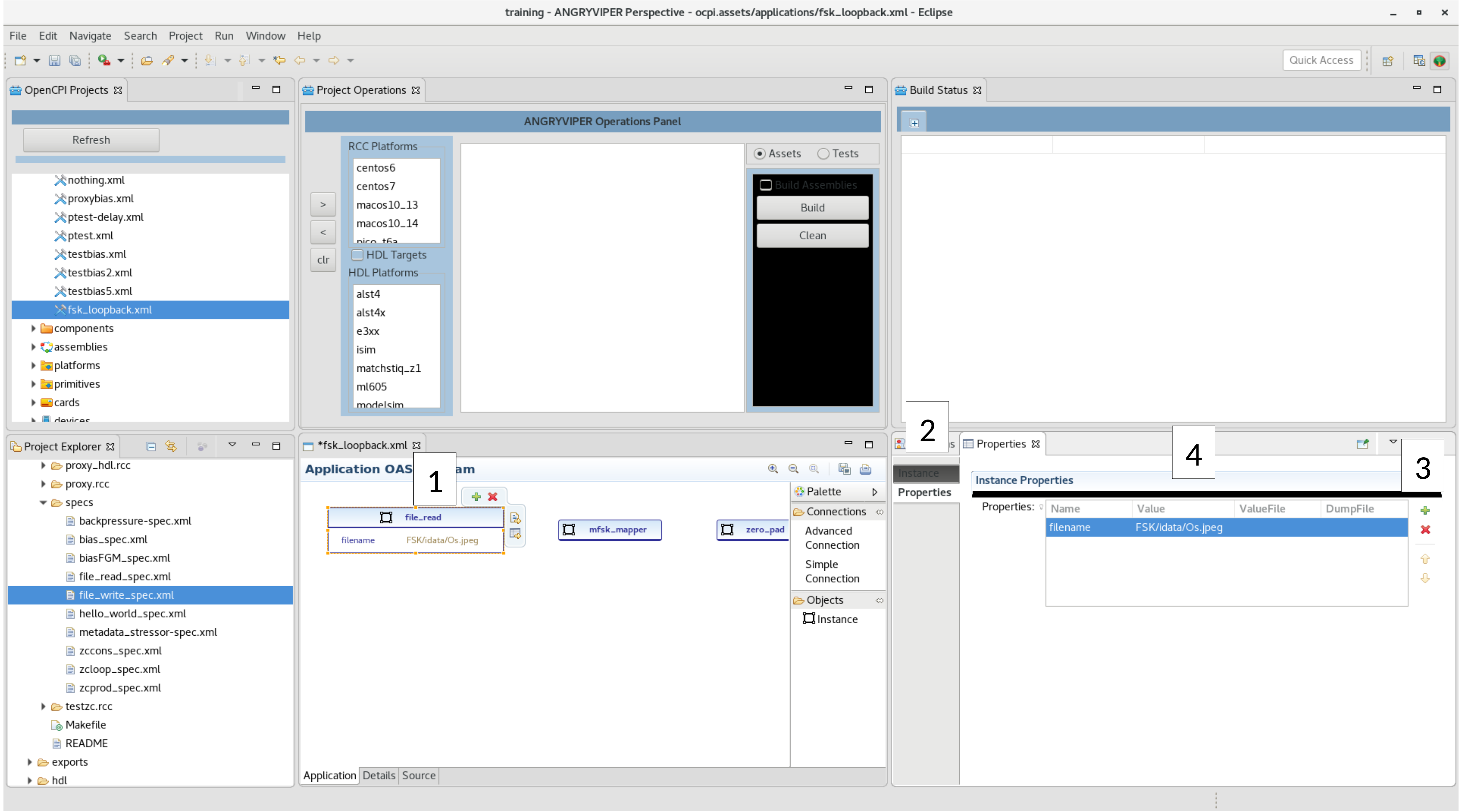
- Set property values
  - To specify a property value **(diagram on next slide)**

1) Right click on instance → “Show in Properties View”
2) Click Properties Tab → Properties
3) Click green plus sign on right side of tab → Instance Property
4) Add “Name” and “Value”



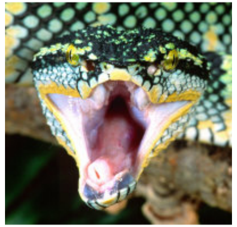
Property Values Required		
Component	Property Name	Value
file_read	fileName	FSK/idata/Os.jpeg
mfsk_mapper	symbols	-32768, 32767
zero_pad	num_zeros	38
phase_to_amp_cordic	magnitude	20000
phase_to_amp_cordic	STAGES	16
cic_int	R	16
cic_int	ACC_WIDTH	28
complex_mixer	enable	False
cic_dec	R	16
cic_dec	ACC_WIDTH	28
baudTracking	SPB	39
baudTracking	BaudAvrCount	10
file_write	fileName	out.out

Specifying Property Values

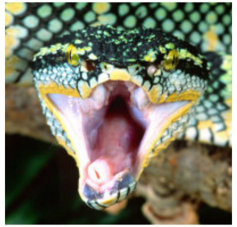


Properties:

Name	Value	ValueFile	DumpFile
filename	FSK/idata/Os.jpeg		



Step 6

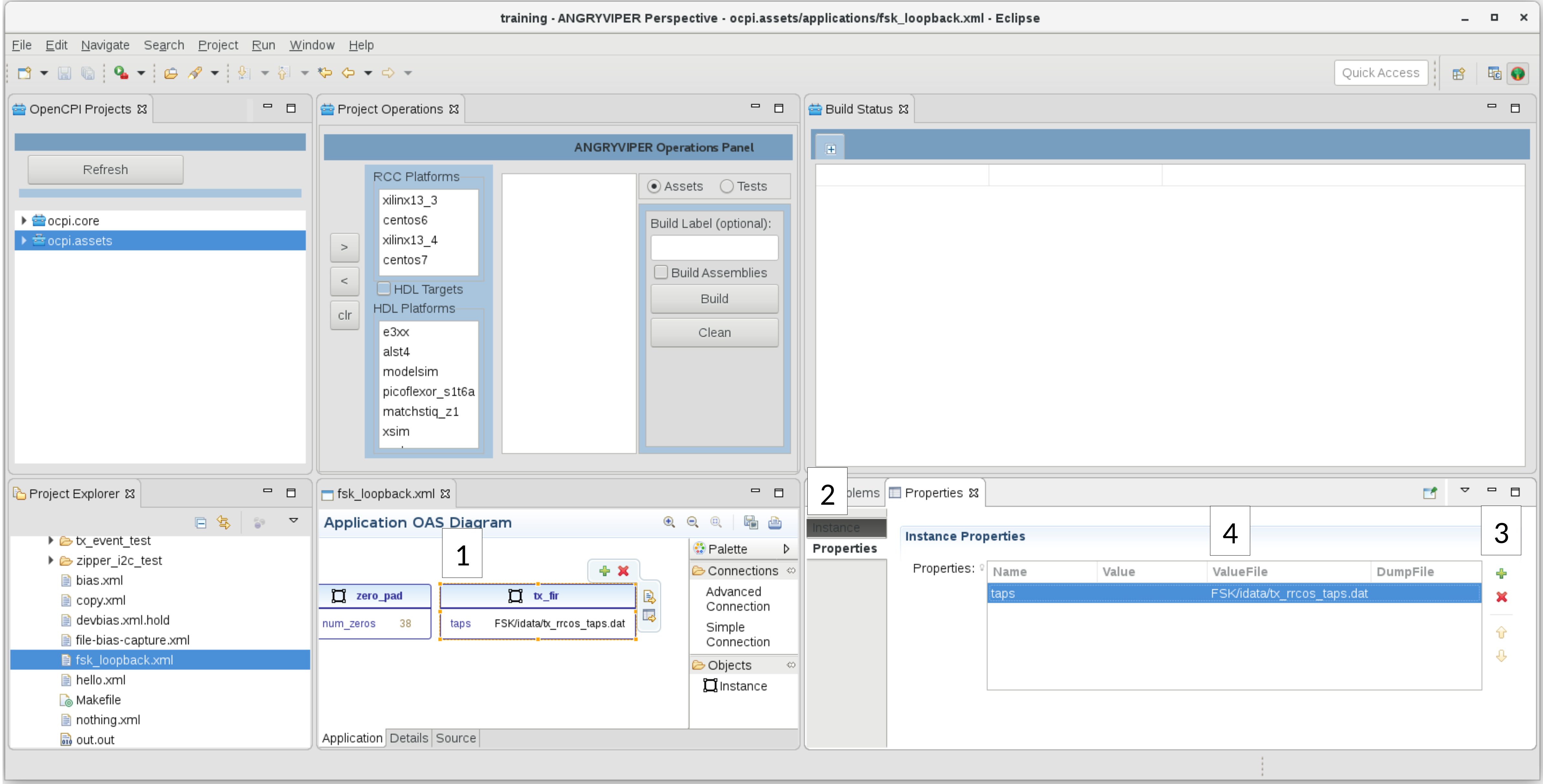


- Set property **ValueFile**
- The two instances of the *fir_real_sse* component used in this application have a property called “taps” which are arrays of 64
- Instead of specifying all 64 values in the IDE, we can set an attribute called **ValueFile** which points to a file which contains the values
- To specify a property **ValueFile** **(diagram on next slide)**

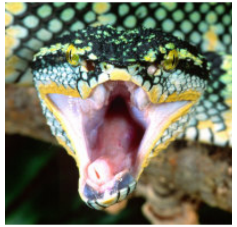
1) Right click on instance → “Show in Properties View”
2) Click Properties Tab → Properties
3) Click green plus sign on right side of tab → Instance Property
4) Add “Name” and “ValueFile”

Property Values Required		
Component	**Property Name**	**ValueFile**
tx_fir	taps	FSK/idata/tx_rrcos_taps.dat
rx_fir	taps	FSK/idata/rx_rrcos_taps.dat

Specifying Property **ValueFile**

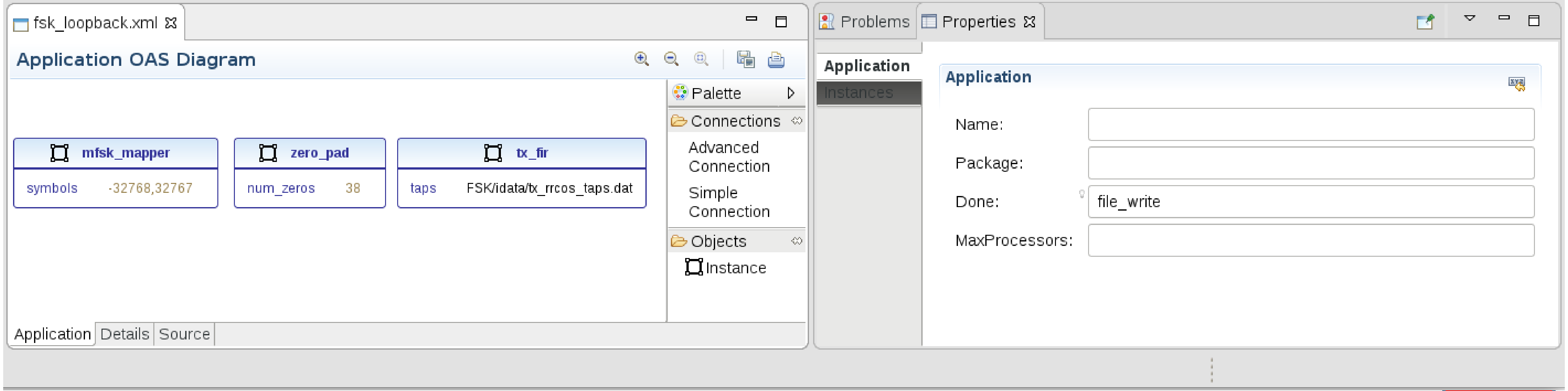


Name	Value	ValueFile	DumpFile
taps		FSK/idata/tx_rrcos_taps.dat	



Step 7

- Specifying Top Level Attributes in OAS
  - Configure the OAS to quit when the file_write component received End-of-File. There is a "top-level" attribute for OAS XML called "Done" used for this purpose
- To set top level OAS attribute:
  1. Click on the white space in between instances so none are selected
  2. In the property tab, fill in the "Done" field with the desired worker name



fsk_loopback.xml

Application OAS Diagram

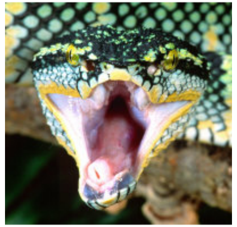
mfsk_mapper	
symbols	-32768,32767

zero_pad	
num_zeros	38

tx_fir	
taps	FSK/idata/tx_rrcos_taps.dat

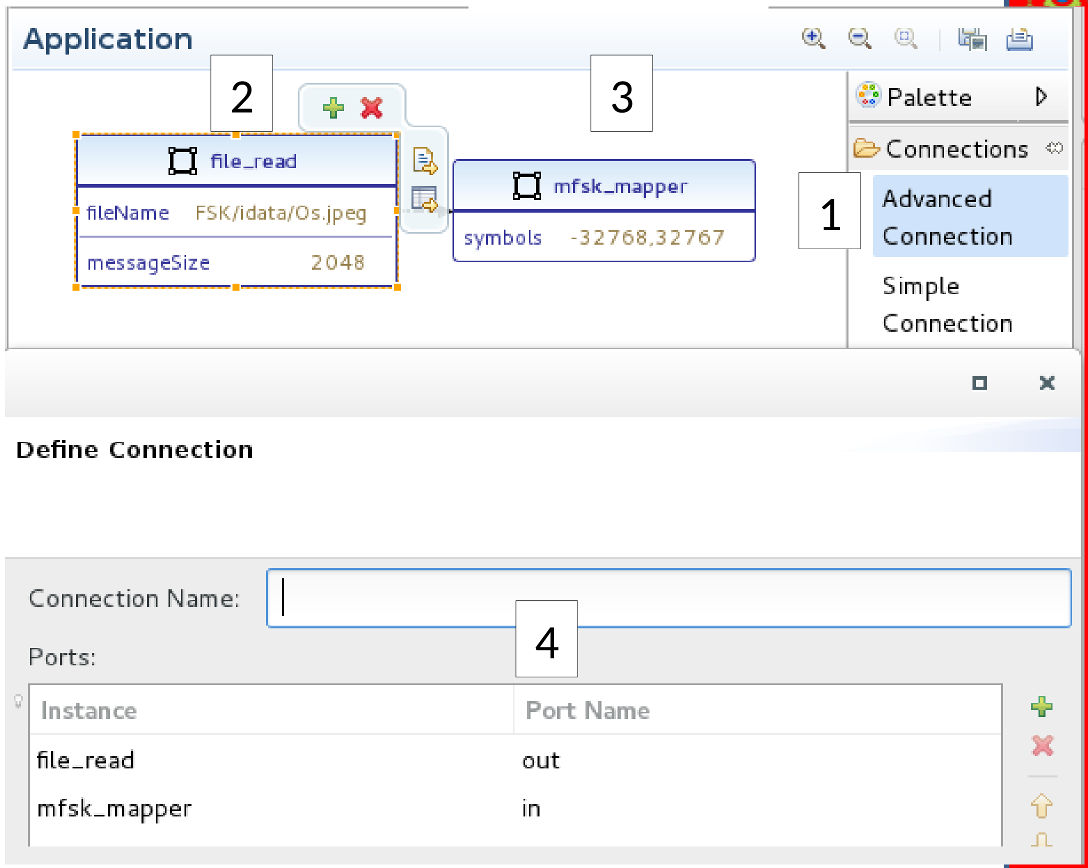
Application

Name:	
Package:	
Done:	file_write
MaxProcessors:	



Step 8

- Make connections
  - See next slide for diagram of required connections
  - Maximizing OAS pane helps
- To make a connection
  1. Click “Advanced Connection” on Palette Menu
  2. Click originating instance
  3. Click destination instance
  4. Populate "Port Name" fields
     - All workers in this lab use the default “out” and “in” Port Names
- **Save your work!**

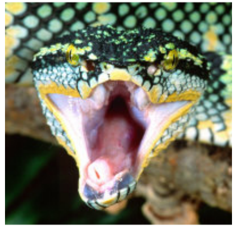


Define Connection

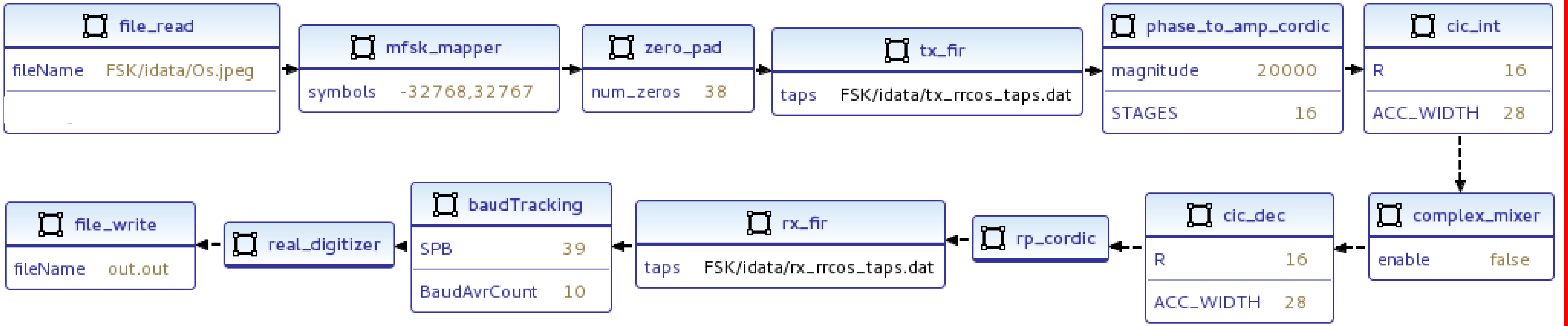
Connection Name:

Ports:

Instance	Port Name
file_read	out
mfsk_mapper	in



End Result



file_read
fileName FSK/idata/Os.jpeg

mfsk_mapper
symbols -32768,32767

zero_pad
num_zeros 38

tx_fir
taps FSK/idata/tx_rrcos_taps.dat

phase_to_amp_cordic
magnitude 20000
STAGES 16

cic_int
R 16
ACC_WIDTH 28

complex_mixer
enable false

cic_dec
R 16
ACC_WIDTH 28

rp_cordic

rx_fir
taps FSK/idata/rx_rrcos_taps.dat

baudTracking
SPB 39
BaudAvrCount 10

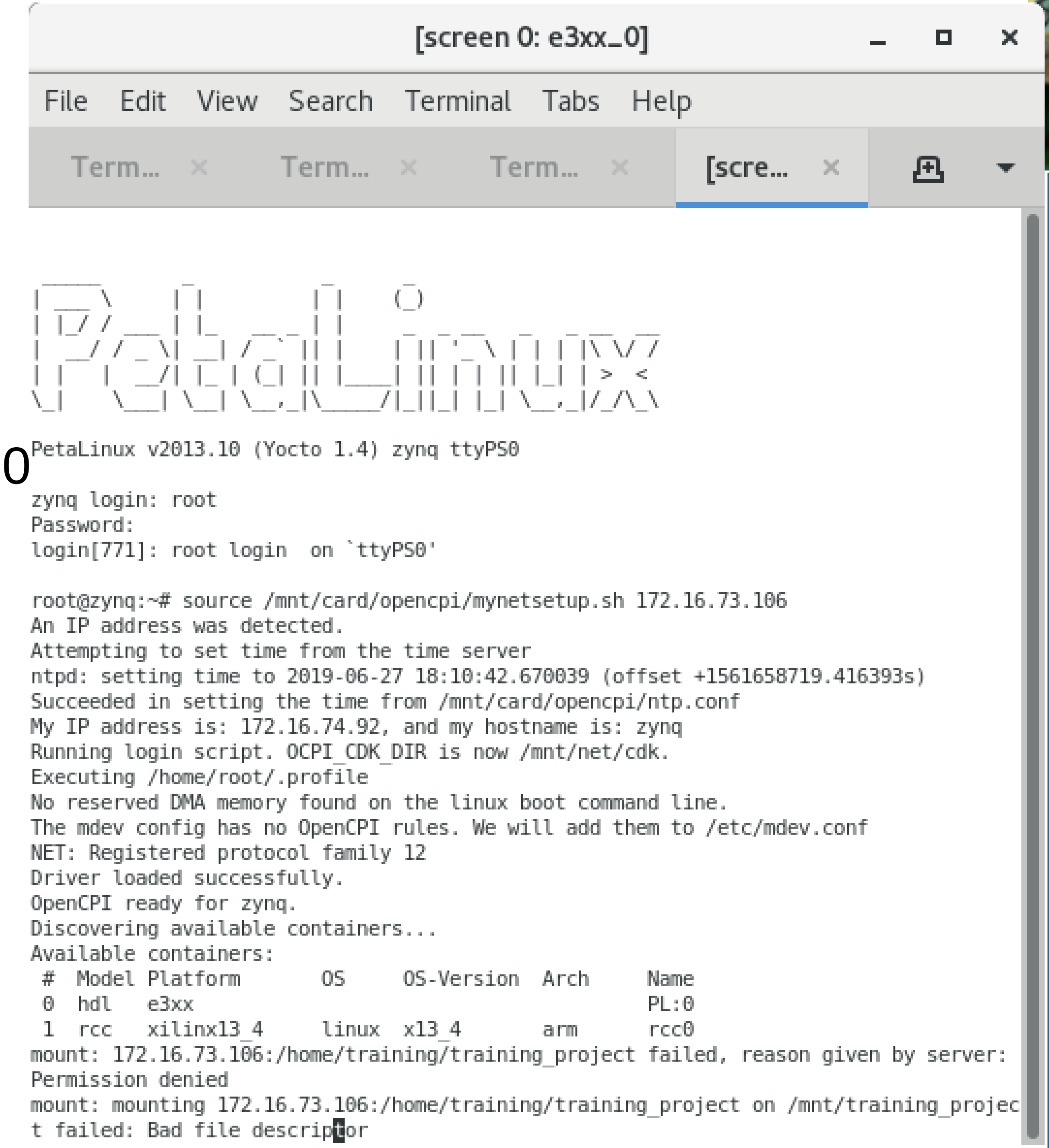
real_digitizer

file_write
fileName out.out

Step 9

- Setup deployment platform
  1. Connect to serial port via USB on rear of Ettus E310 on Host
     - “screen /dev/e3xx_0 115200”
  2. Boot and login into Petalinux on E310
     - User/Password = root:root
  3. Verify Host and E310 have valid IP addresses
     - For training, they should both be on the same subnet
  4. Run setup script on E310
     - “source /mnt/card/opencpi/mynetsetup.sh <Host ip address>”

More detail on this process can be found in the **E3xx Getting Started Guide** document



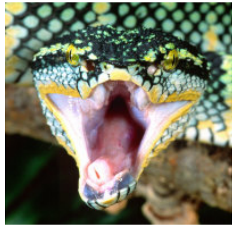
```
______     _        _     _
| ___ \   | |      | |   (_)
| |_/ /__ | |_ __ _| |    _ _ __  _   ___  __
|  __/ _ \| __/ _` | |   | | '_ \| | | \ \/ /
| | |  __/| || (_| | |___| | | | | |_| |>  <
\_|  \___| \__\__,_|_____/_|_| |_|\__,_/_/\_\

PetaLinux v2013.10 (Yocto 1.4) zynq ttyPS0

zynq login: root
Password:
login[771]: root login  on `ttyPS0'

root@zynq:~# source /mnt/card/opencpi/mynetsetup.sh 172.16.73.106
An IP address was detected.
Attempting to set time from the time server
ntpd: setting time to 2019-06-27 18:10:42.670039 (offset +1561658719.416393s)
Succeeded in setting the time from /mnt/card/opencpi/ntp.conf
My IP address is: 172.16.74.92, and my hostname is: zynq
Running login script. OCPI_CDK_DIR is now /mnt/net/cdk.
Executing /home/root/.profile
No reserved DMA memory found on the linux boot command line.
The mdev config has no OpenCPI rules. We will add them to /etc/mdev.conf
NET: Registered protocol family 12
Driver loaded successfully.
OpenCPI ready for zynq.
Discovering available containers...
Available containers:
 #  Model Platform       OS     OS-Version  Arch     Name
 0  hdl   e3xx                                       PL:0
 1  rcc   xilinx13_4     linux  x13_4       arm      rcc0
mount: 172.16.73.106:/home/training/training_project failed, reason given by server:
Permission denied
mount: mounting 172.16.73.106:/home/training/training_project on /mnt/training_projec
t failed: Bad file descriptor
```

Step 10

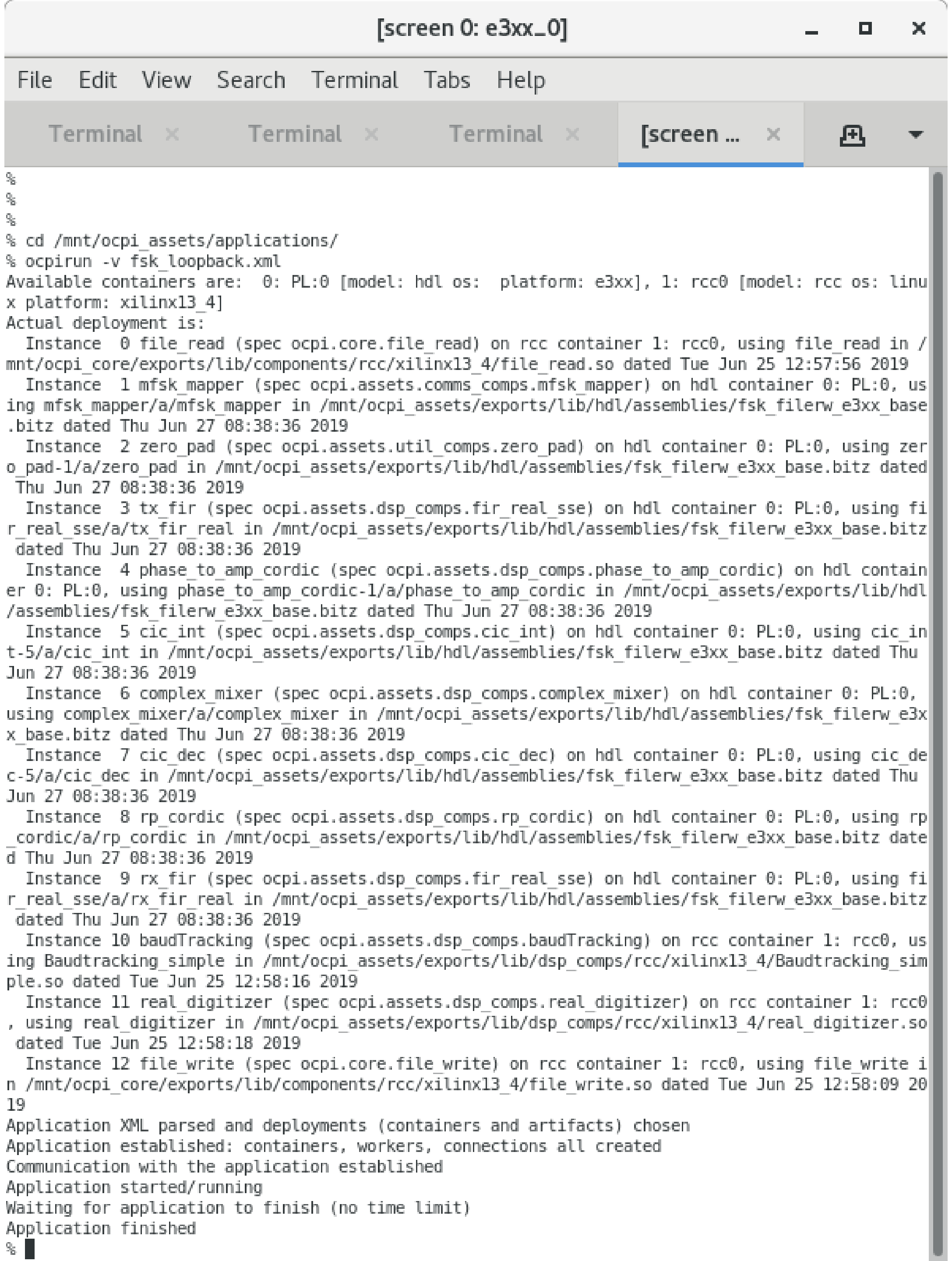


- Configure run-time artifact search path on target platform, i.e. OCPI_LIBRARY_PATH=
  - At run-time, applications must locate artifacts that satisfy its requirements, as defined in the OAS XML
    - Software worker .so files
    - HDL container .bitz files
  - To deploy this application, 5 artifacts are needed: 4 software worker .so files, 1 HDL container .bitz
    - file_read.so
    - file_write.so
    - Baudtracking_simple.so
    - real_digitizer.so
    - fsk_filerw_e3xx_base.bitz
  - The OCPI_LIBRARY_PATH environment variable defines the search path for locating deployable artifacts
    - Path are searched recursively, so this variable can be as very specific or as broad as needed for locating the artifacts.
      - Broader paths lead to longer search times when running an application
    - The exports directory at the top level of project contains links to artifacts contained in the project
    - Component instances were added from the component libraries contained within these projects.
    - Set OCPI_LIBRARY_PATH on target platform

      "**export OCPI_LIBRARY_PATH=/mnt/ocpi_core/exports:/mnt/ocpi_assets/exports**”

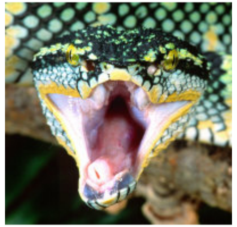
Step 11

- Run application on E310 using ocpirun
    - ocpirun is a utility program provided with the Component Development Kit (CDK) for running applications described by OAS XML
- To run application on E310:
    1. Navigate to OAS XML:
        - "cd /mnt/ocpi_assets/applications"
    2. Pass OAS XML to ocpirun:
        - "ocpirun -v fsk_loopback.xml"
        - ocpirun is a utility program provided with the CDK for running the application
            - Optional arguments on previous slides
        - Problems? See next slide
    3. View output image on Host
        - "cd /home/training/ocpi_projects/assets/applications"
        - "eog out.out"



```
%
%
%
% cd /mnt/ocpi_assets/applications/
% ocpirun -v fsk_loopback.xml
Available containers are:  0: PL:0 [model: hdl os:  platform: e3xx], 1: rcc0 [model: rcc os: linu
x platform: xilinx13_4]
Actual deployment is:
  Instance  0 file_read (spec ocpi.core.file_read) on rcc container 1: rcc0, using file_read in /
mnt/ocpi_core/exports/lib/components/rcc/xilinx13_4/file_read.so dated Tue Jun 25 12:57:56 2019
  Instance  1 mfsk_mapper (spec ocpi.assets.comms_comps.mfsk_mapper) on hdl container 0: PL:0, us
ing mfsk_mapper/a/mfsk_mapper in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_filerw_e3xx_base
.bitz dated Thu Jun 27 08:38:36 2019
  Instance  2 zero_pad (spec ocpi.assets.util_comps.zero_pad) on hdl container 0: PL:0, using zer
o_pad-1/a/zero_pad in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_filerw_e3xx_base.bitz dated
 Thu Jun 27 08:38:36 2019
  Instance  3 tx_fir (spec ocpi.assets.dsp_comps.fir_real_sse) on hdl container 0: PL:0, using fi
r_real_sse/a/tx_fir_real in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_filerw_e3xx_base.bitz
 dated Thu Jun 27 08:38:36 2019
  Instance  4 phase_to_amp_cordic (spec ocpi.assets.dsp_comps.phase_to_amp_cordic) on hdl contain
er 0: PL:0, using phase_to_amp_cordic-1/a/phase_to_amp_cordic in /mnt/ocpi_assets/exports/lib/hdl
/assemblies/fsk_filerw_e3xx_base.bitz dated Thu Jun 27 08:38:36 2019
  Instance  5 cic_int (spec ocpi.assets.dsp_comps.cic_int) on hdl container 0: PL:0, using cic_in
t-5/a/cic_int in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_filerw_e3xx_base.bitz dated Thu
Jun 27 08:38:36 2019
  Instance  6 complex_mixer (spec ocpi.assets.dsp_comps.complex_mixer) on hdl container 0: PL:0,
using complex_mixer/a/complex_mixer in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_filerw_e3x
x_base.bitz dated Thu Jun 27 08:38:36 2019
  Instance  7 cic_dec (spec ocpi.assets.dsp_comps.cic_dec) on hdl container 0: PL:0, using cic_de
c-5/a/cic_dec in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_filerw_e3xx_base.bitz dated Thu
Jun 27 08:38:36 2019
  Instance  8 rp_cordic (spec ocpi.assets.dsp_comps.rp_cordic) on hdl container 0: PL:0, using rp
_cordic/a/rp_cordic in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_filerw_e3xx_base.bitz date
d Thu Jun 27 08:38:36 2019
  Instance  9 rx_fir (spec ocpi.assets.dsp_comps.fir_real_sse) on hdl container 0: PL:0, using fi
r_real_sse/a/rx_fir_real in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_filerw_e3xx_base.bitz
 dated Thu Jun 27 08:38:36 2019
  Instance 10 baudTracking (spec ocpi.assets.dsp_comps.baudTracking) on rcc container 1: rcc0, us
ing Baudtracking_simple in /mnt/ocpi_assets/exports/lib/dsp_comps/rcc/xilinx13_4/Baudtracking_sim
ple.so dated Tue Jun 25 12:58:16 2019
  Instance 11 real_digitizer (spec ocpi.assets.dsp_comps.real_digitizer) on rcc container 1: rcc0
, using real_digitizer in /mnt/ocpi_assets/exports/lib/dsp_comps/rcc/xilinx13_4/real_digitizer.so
 dated Tue Jun 25 12:58:18 2019
  Instance 12 file_write (spec ocpi.core.file_write) on rcc container 1: rcc0, using file_write i
n /mnt/ocpi_core/exports/lib/components/rcc/xilinx13_4/file_write.so dated Tue Jun 25 12:58:09 20
19
Application XML parsed and deployments (containers and artifacts) chosen
Application established: containers, workers, connections all created
Communication with the application established
Application started/running
Waiting for application to finish (no time limit)
Application finished
%
```

Common Errors / Debugging



1. "No acceptable implementations found"
   - OCPI_LIBRARY_PATH incorrect; try "-l 8"
   - Typo in OAS; check "Source" Tab and check spelling
     - Log 8 would say something like: Rejected: initial property "your_typo" not found
2. "No containers were found for deploying instance"
   - OCPI_LIBRARY_PATH incorrect
   - Have instructor check project exports
3. "...produced an error during the "start" control operation"
   - Follow diagnostics given, *e.g*. mistyped fileName entry
4. "Can't process file..."
   - Follow diagnostics given, *e.g*. mistyped ValueFile entry