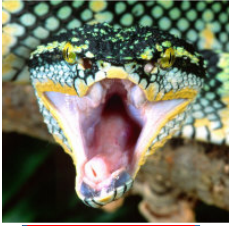


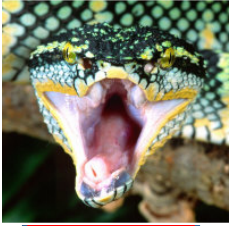
Lab 1

OpenCPI Application Development



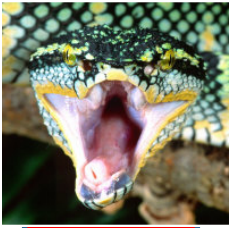
Objectives

1. Create FSK loopback (FPGA internal) OpenCPI Application XML (OAS) using the IDE
2. Run application on Matchstiq Z1 hardware



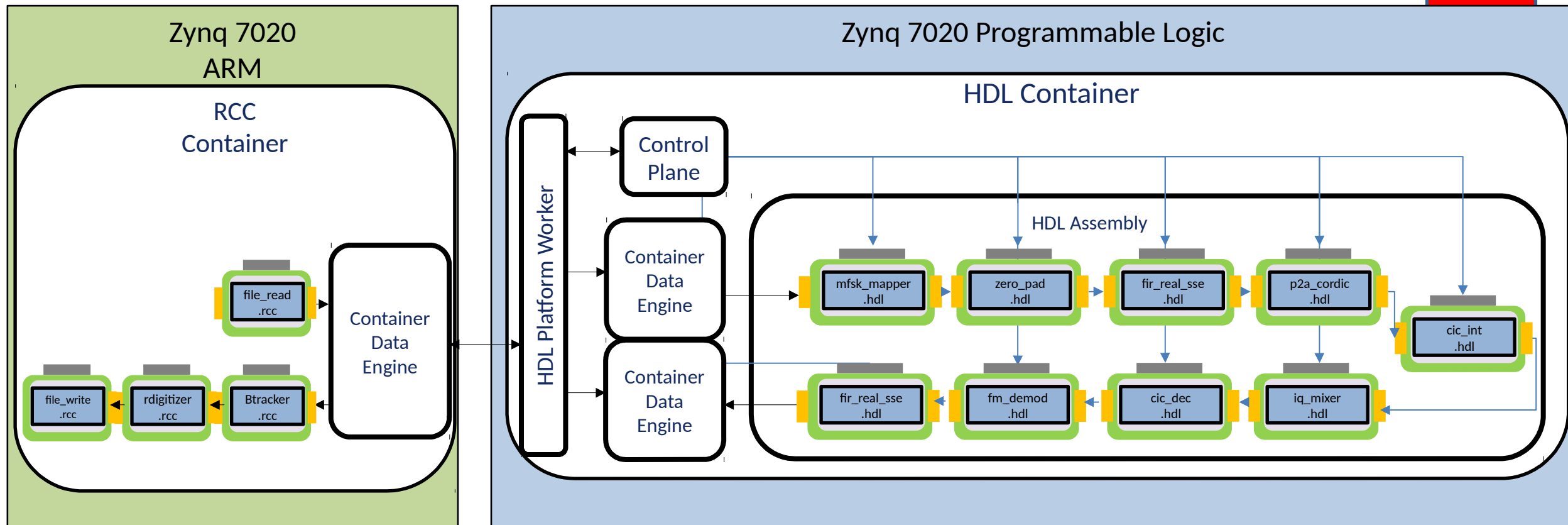
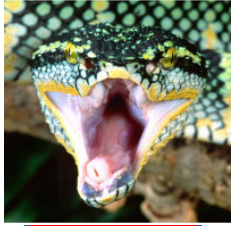
Overview

- 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

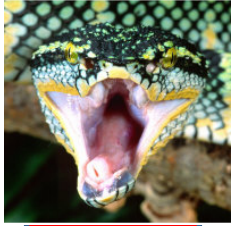


Overview

- 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



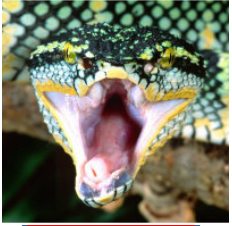
- The utility program ocpirun provides a simple way to execute applications
- Usage is: ocpirun app.xml
 - app.xml is a OAS file like the one which will be generated with the IDE in this lab
- The arguments passed to ocpirun can specify how the application is run

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

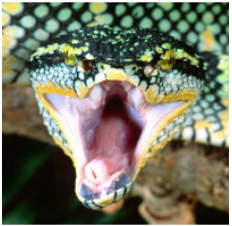
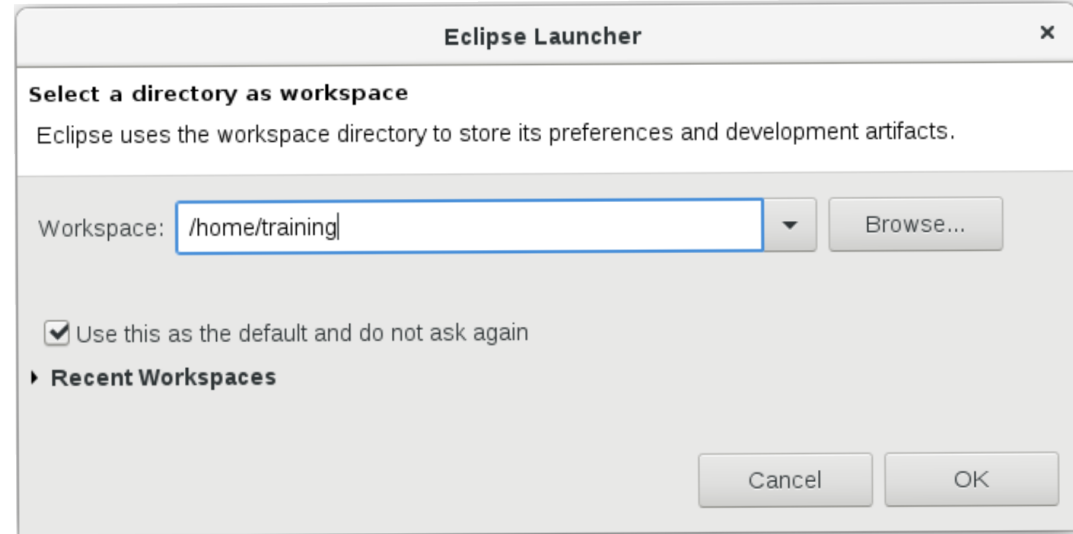
Application Development Flow

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



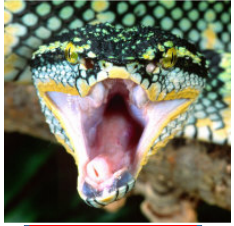
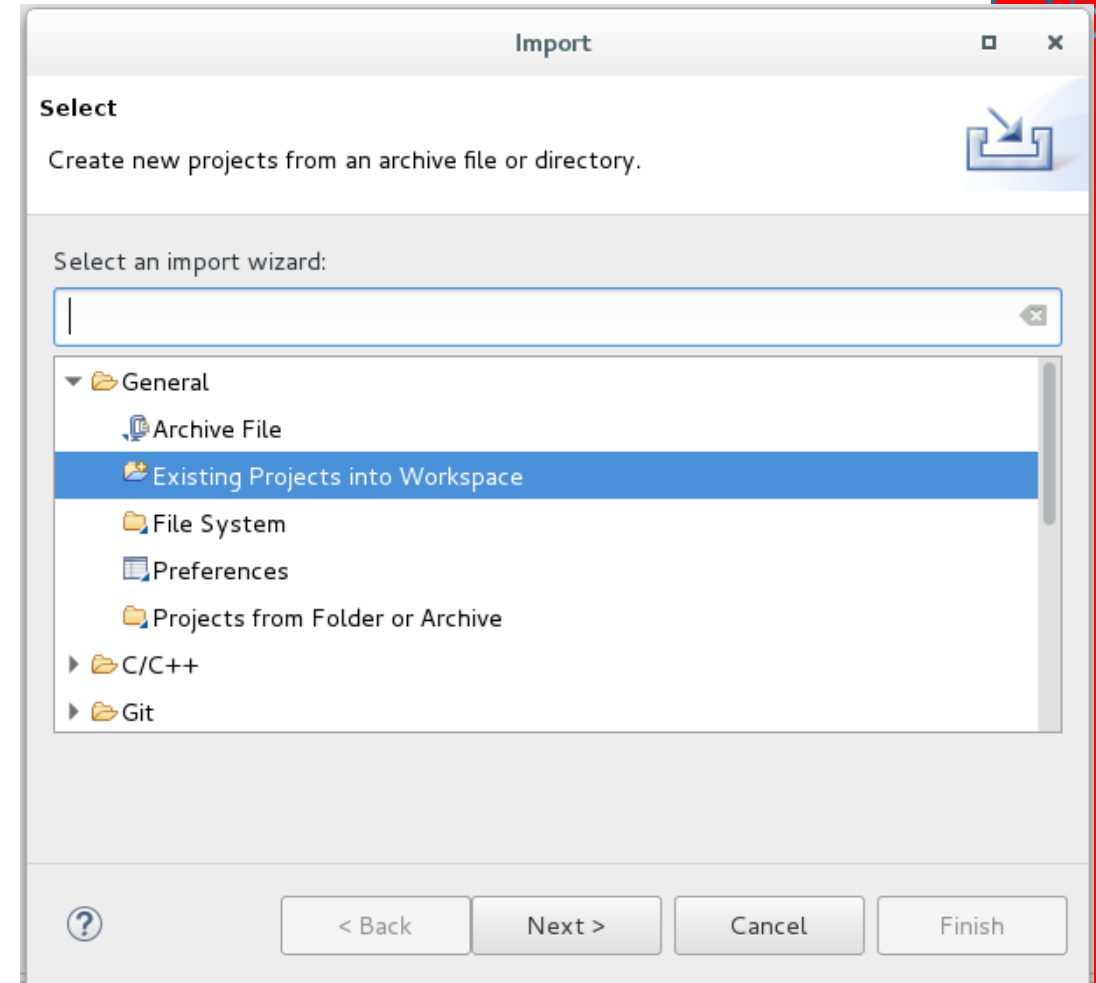
Step 1

- Start AV IDE and set the default workspace to:
 - /home/training/
 - 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”



Step 2

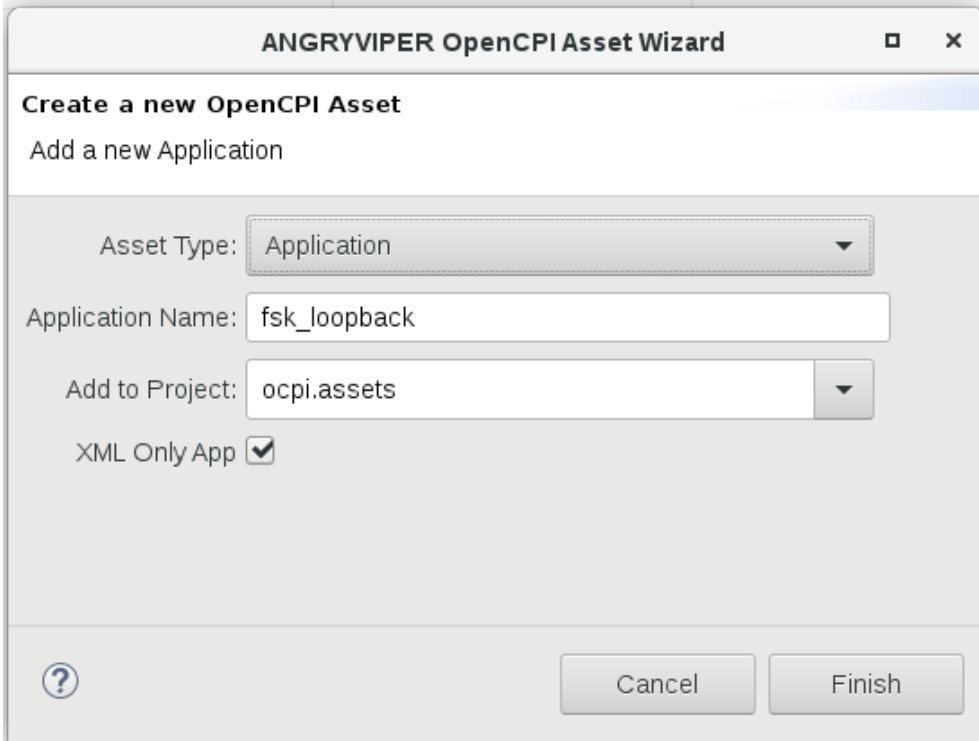
- Import pre-built projects: core and assets
 - The **core** project contains some basic components, including workers to read and write files.
 - Another project called **assets** is included. It contains a number of components used in this lab.
 - Pre-built projects are located at:
 - ~/core
 - ~/assets
- To import project into eclipse:
 - File → Import...
 - “Existing Projects into Workspace”



Open
CPI

Step 3

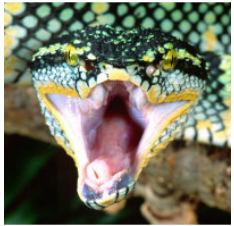
- Create new application in an existing project
- To create an application
 - In OpenCPI Projects, right click assets:
 - Asset Wizard
 - Asset Type: Application
 - Application Name: fsk_loopback
 - Add to Project: ocpi.assets
 - XML only: Yes



The screenshot shows a dialog box titled "ANGRYVIPER OpenCPI Asset Wizard". Inside, the main heading is "Create a new OpenCPI Asset" with the subtitle "Add a new Application". The form contains the following fields and controls:

- Asset Type:** A dropdown menu with "Application" selected.
- Application Name:** A text input field containing "fsk_loopback".
- Add to Project:** A dropdown menu with "ocpi.assets" selected.
- XML Only App:** A checkbox that is checked.

At the bottom left is a help icon (a question mark in a circle). At the bottom right are two buttons: "Cancel" and "Finish".



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 a component
 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'

Component Specs Required	
Name	Project : Library
file_read_spec.xml	Core Project : 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 Project : 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
file_read	messageSize	2048
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

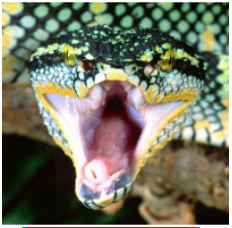
The screenshot displays the ANGRYVIPER Perspective in Eclipse, titled "training - ANGRYVIPER Perspective - ocpi.assets/applications/fsk_loopback.xml - Eclipse". The interface is divided into several panels:

- OpenCPI Projects:** Shows a list of projects including "ocpi.core" and "ocpi.assets".
- Project Operations:** Contains the "ANGRYVIPER Operations Panel" with sections for "RCC Platforms" (listing xilinx13_3, centos6, xilinx13_4, centos7, macos10_13), "HDL Targets" (with a checkbox), and "HDL Platforms" (listing e3xx, alst4, modelsim, picoflexor_s1t6a, matchstiq_z1, xsim, zed). It also includes "Assets" and "Tests" radio buttons, a "Build Label (optional)" text field, a "Build Assemblies" checkbox, and "Build" and "Clean" buttons.
- Build Status:** A panel for monitoring build progress.
- Project Explorer:** Shows the file structure of the "fsk_loopback.xml" project, including files like "tx_event_test", "zipper_i2c_test", "bias.xml", "copy.xml", "devbias.xml.hold", "file-bias-capture.xml", "fsk_loopback.xml" (selected), "hello.xml", "Makefile", "nothing.xml", "out.out", and "pattern-bias-file.xml".
- fsk_loopback.xml OAS Diagram:** Displays the Open Architecture Specification (OAS) diagram. It features a "file_read" component with properties "fileName" (FSK/ldata/Os.jpeg) and "messageSize" (2048), and a "mtsk_mapper" component with a "symbols" property set to "-32768,32767".
- Properties:** A panel showing the "Instance Properties" for the selected component. It includes a table with the following data:

Name	Value	ValueFile	DumpFile
fileName	FSK/ldata/Os.jpeg		
messageSize	2048		

Numbered callouts indicate key areas: 1 points to the "file_read" component in the OAS diagram; 2 points to the "Instance Properties" tab; 3 points to the "Properties" section header; and 4 points to the "Value" column in the properties table.

Step 6



- Set property ValueFiles
- The fir_real_sse components 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
rx_fir	taps	FSK/idata/rx_rrcos_taps.dat
tx_fir	taps	FSK/idata/tx_rrcos_taps.dat

Specifying Property ValueFiles

The screenshot displays the Eclipse IDE interface with the ANGRYVIPER tool. The title bar indicates the project is 'training - ANGRYVIPER Perspective - ocpi.assets/applications/fsk_loopback.xml - Eclipse'.

OpenCPI Projects: Shows a tree view with 'ocpi.core' and 'ocpi.assets'.

Project Operations: The 'ANGRYVIPER Operations Panel' is active, showing 'RCC Platforms' (xilinx13_3, centos6, xilinx13_4, centos7) and 'HDL Platforms' (e3xx, alst4, modelsim, picoflexor_s1t6a, matchstiq_z1, xsim). The 'Assets' tab is selected, and the 'Build' button is visible.

Project Explorer: Shows the project structure with 'fsk_loopback.xml' selected.

Application OAS Diagram: A diagram showing a 'zero_pad' block connected to a 'tx_fir' block. The 'tx_fir' block has a property 'taps' with a value of 'FSK/ldata/tx_rrcos_taps.dat'. A red box highlights the 'taps' property, and a red arrow points to it from the 'Instance Properties' table.

Instance Properties: A table showing the properties of the selected instance. The table has columns: Name, Value, ValueFile, and DumpFile. The row for 'taps' is highlighted, showing the ValueFile 'FSK/ldata/tx_rrcos_taps.dat'.

Annotations: Numbered boxes 1, 2, 3, and 4 highlight specific elements: 1 points to the 'taps' property in the diagram; 2 points to the 'Instance Properties' tab; 3 points to the 'ValueFile' column header; 4 points to the 'ValueFile' value 'FSK/ldata/tx_rrcos_taps.dat'.

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

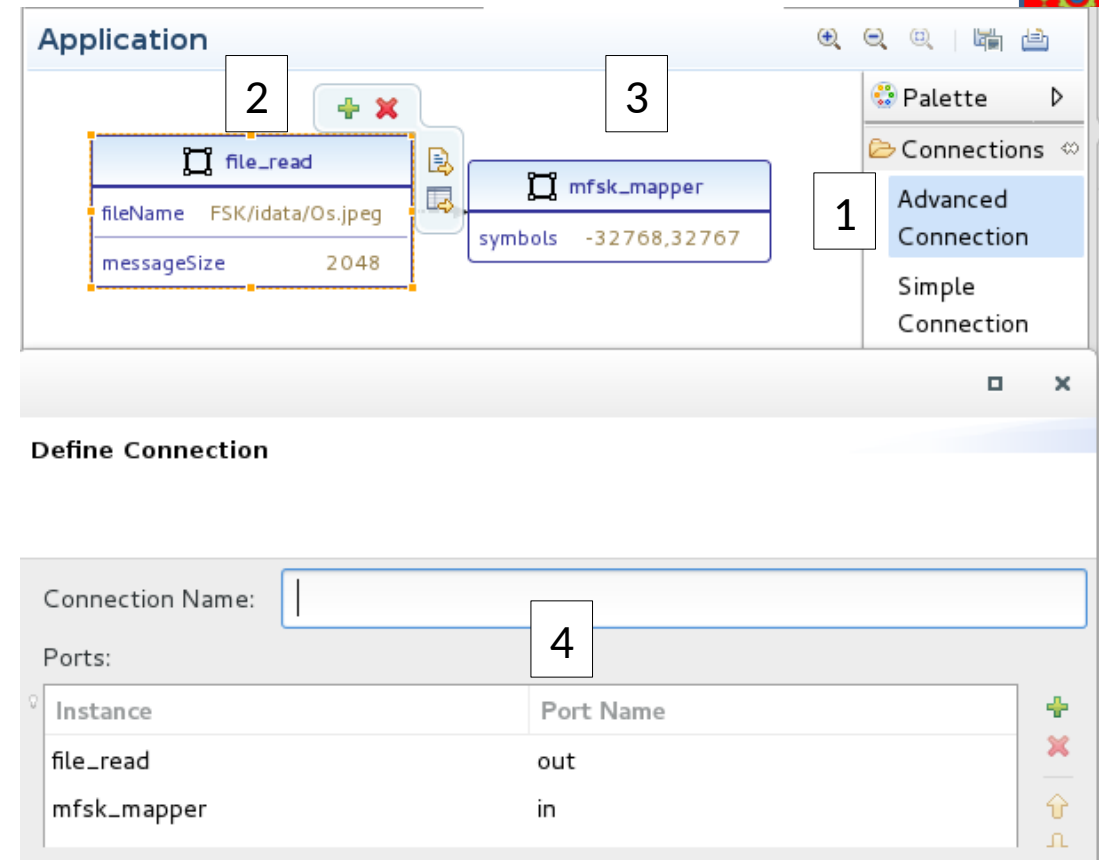
Step 7

- Specifying Top Level Attributes in OAS
 - Configure the OAS to 'be done' 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

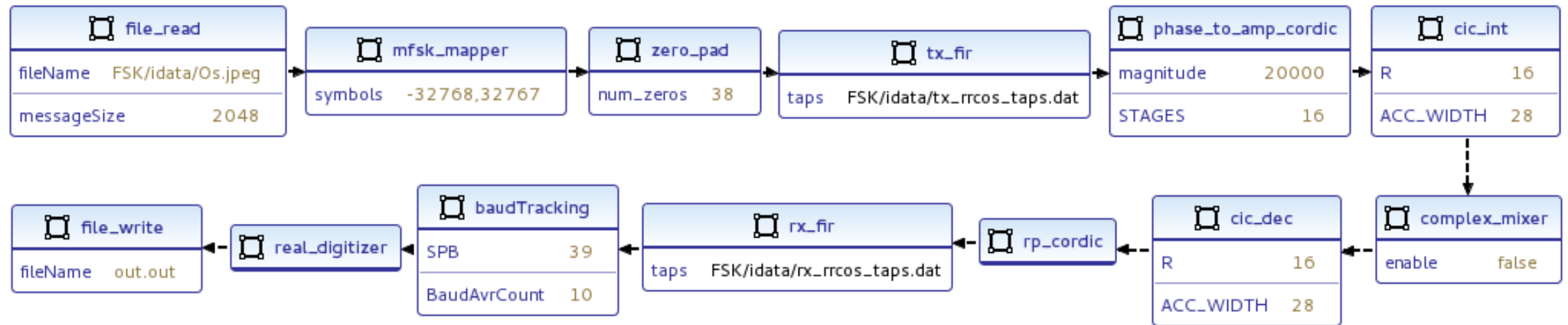
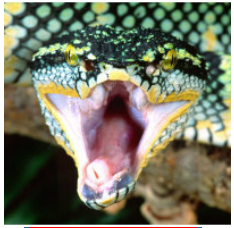
The screenshot displays the OpenCPN software interface. The main window shows the 'Application OAS Diagram' for a file named 'fsk_loopback.xml'. The diagram contains three component instances: 'mfsk_mapper' (with parameters 'symbols' and '-32768,32767'), 'zero_pad' (with parameters 'num_zeros' and '38'), and 'tx_fir' (with parameters 'taps' and 'FSK/ldata/tx_rrcos_taps.dat'). A right-hand sidebar contains a 'Palette' with 'Connections' and 'Objects' sections, and a 'Properties' tab. The 'Properties' tab is active, showing the 'Application' section with fields for 'Name', 'Package', 'Done', and 'MaxProcessors'. The 'Done' field is currently set to 'file_write'. The bottom of the window has tabs for 'Application', 'Details', and 'Source'.

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!**



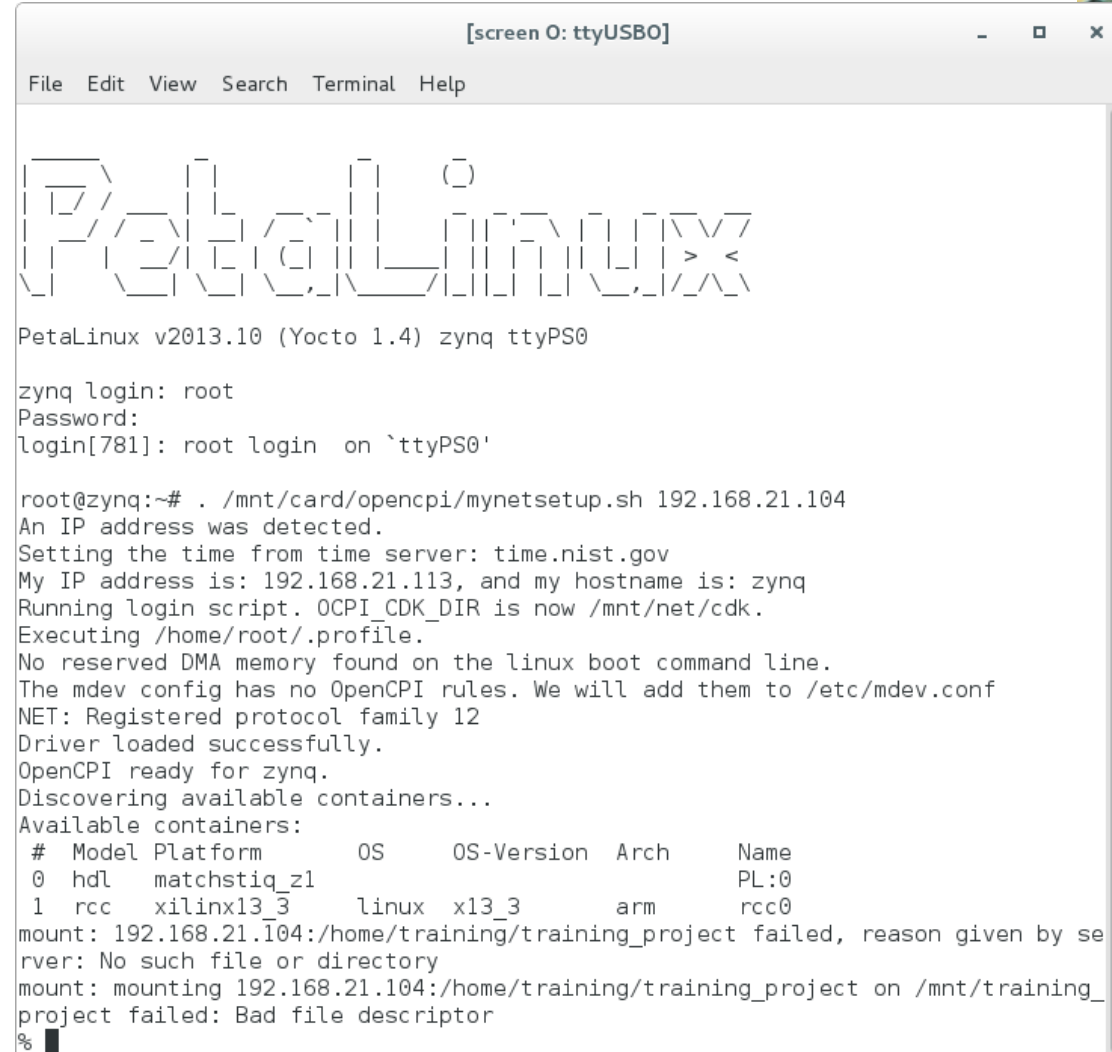
End Result



Step 9

- Setup deployment platform
 1. Connect to serial port via USB on rear of Matchstiq Z1 using Host
 - 'screen /dev/matchstiq_z1_0 115200'
 2. Boot and login into PetaLinux
 - User/Password = root:root
 3. Verify Host and Matchstiq Z1 have valid IP addresses
 - For training, they should both be on the same subnet
 4. Run setup script on Matchstiq Z1
 - 'source /mnt/card/opencv/mynetsetup.sh <Host IP address>'

More detail on this process can be found in the **Matchstiq_Z1 Getting Started Guide** document



The image shows a terminal window titled "[screen 0: ttyUSB0]" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output displays the PetaLinux boot sequence, including the PetaLinux logo, version information (v2013.10, Yocto 1.4), and login details (zynq login: root, Password: root). It then shows the execution of the mynetsetup.sh script, which sets the IP address to 192.168.21.104, sets the time from time.nist.gov, and displays the system's IP address (192.168.21.113) and hostname (zynq). The script also shows the execution of the login script, the OpenCPI setup, and the discovery of available containers. A table of available containers is shown, with columns for #, Model, Platform, OS, OS-Version, Arch, and Name. The table lists two containers: 0 (hdl, matchstiq_z1, PL:0) and 1 (rcc, xilinx13_3, linux, x13_3, arm, rcc0). The script then attempts to mount the training project directory, but fails with the error "mount: mounting 192.168.21.104:/home/training/training_project on /mnt/training_project failed: Bad file descriptor".

```
[screen 0: ttyUSB0]
File Edit View Search Terminal Help

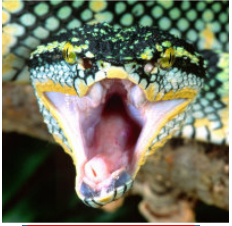
PetaLinux

PetaLinux v2013.10 (Yocto 1.4) zynq ttyPS0

zynq login: root
Password:
login[781]: root login on `ttyPS0'

root@zynq:~# . /mnt/card/opencv/mynetsetup.sh 192.168.21.104
An IP address was detected.
Setting the time from time server: time.nist.gov
My IP address is: 192.168.21.113, 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 matchstiq_z1 PL:0
1 rcc xilinx13_3 linux x13_3 arm rcc0
mount: 192.168.21.104:/home/training/training_project failed, reason given by se
rver: No such file or directory
mount: mounting 192.168.21.104:/home/training/training_project on /mnt/training_
project failed: Bad file descriptor
%
```

Step 10



- Setup environment on Matchstiq Z1 using OCPI_LIBRARY_PATH
 - The OCPI_LIBRARY_PATH environment variable is used to locate deployable artifacts
 - To deploy this application, 5 artifacts are needed
 - 4 software worker .so files
 1. file_read.so
 2. file_write.so
 3. Baudtracking_simple.so
 4. real_digitizer.so
 - 1 HDL container .bitz
 1. fsk_filerw_matchstiq_base.bitz
 - To set OCPI_LIBRARY_PATH on Matchstiq Z1
 - 'export OCPI_LIBRARY_PATH=/mnt/ocpi_core/exports:/mnt/ocpi_assets/exports'
 - These component instances were added from these component libraries.
 - The directories in this path are searched recursively, so this variable can be as specific or as broad as needed as long as the artifacts are in the path. 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

Step 11

- Run application on Matchstiq Z1 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 Matchstiq Z1:
 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/assets/applications'
 - 'eog out.out'



```
File Edit View Search Terminal Help
% cd /mnt/ocpi_assets/applications/
% ocpirun -v fsk_loopback.xml
Available containers are: 0: PL:0 [model: hdl os: platform: matchstiq_z1], 1: rcc
0 [model: rcc os: linux platform: xilinx13_3]
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/linux-x13_3-arm/file_read_s.
  so dated Wed Feb 14 09:38:37 2018
  Instance 1 mfsk_mapper (spec ocpi.assets.comms_comps.mfsk_mapper) on hdl contain
  er 0: PL:0, using mfsk_mapper/a/mfsk_mapper in /mnt/ocpi_assets/exports/lib/hdl/ass
  embles/fsk_filerw_matchstiq_z1_base.bitz dated Tue Feb 13 16:44:59 2018
  Instance 2 zero_pad (spec ocpi.assets.util_comps.zero_pad) on hdl container 0: P
  L:0, using zero_pad-1/a/zero_pad in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk
  _filerw_matchstiq_z1_base.bitz dated Tue Feb 13 16:44:59 2018
  Instance 3 tx_fir (spec ocpi.assets.dsp_comps.fir_real_sse) on hdl container 0:
  PL:0, using fir_real_sse/a/tx_fir_real in /mnt/ocpi_assets/exports/lib/hdl/assembli
  es/fsk_filerw_matchstiq_z1_base.bitz dated Tue Feb 13 16:44:59 2018
  Instance 4 phase_to_amp_cordic (spec ocpi.assets.dsp_comps.phase_to_amp_cordic)
  on hdl container 0: PL:0, using phase_to_amp_cordic-1/a/phase_to_amp_cordic in /mnt
  /ocpi_assets/exports/lib/hdl/assemblies/fsk_filerw_matchstiq_z1_base.bitz dated Tue
  Feb 13 16:44:59 2018
  Instance 5 cic_int (spec ocpi.assets.dsp_comps.cic_int) on hdl container 0: PL:0
  , using cic_int-5/a/cic_int in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_file
  rw_matchstiq_z1_base.bitz dated Tue Feb 13 16:44:59 2018
  Instance 6 complex_mixer (spec ocpi.assets.dsp_comps.complex_mixer) on hdl conta
  iner 0: PL:0, using complex_mixer/a/complex_mixer in /mnt/ocpi_assets/exports/lib/h
  dl/assemblies/fsk_filerw_matchstiq_z1_base.bitz dated Tue Feb 13 16:44:59 2018
  Instance 7 cic_dec (spec ocpi.assets.dsp_comps.cic_dec) on hdl container 0: PL:0
  , using cic_dec-5/a/cic_dec in /mnt/ocpi_assets/exports/lib/hdl/assemblies/fsk_file
  rw_matchstiq_z1_base.bitz dated Tue Feb 13 16:44:59 2018
  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/fs
  k_filerw_matchstiq_z1_base.bitz dated Tue Feb 13 16:44:59 2018
  Instance 9 rx_fir (spec ocpi.assets.dsp_comps.fir_real_sse) on hdl container 0:
  PL:0, using fir_real_sse/a/rx_fir_real in /mnt/ocpi_assets/exports/lib/hdl/assembli
  es/fsk_filerw_matchstiq_z1_base.bitz dated Tue Feb 13 16:44:59 2018
  Instance 10 baudTracking (spec ocpi.assets.dsp_comps.baudTracking) on rcc contain
  er 1: rcc0, using Baudtracking_simple in /mnt/ocpi_assets/exports/lib/dsp_comps/rcc
  /linux-x13_3-arm/Baudtracking_simple_s.so dated Tue Feb 13 15:16:23 2018
  Instance 11 real_digitizer (spec ocpi.assets.dsp_comps.real_digitizer) on rcc con
  tainer 1: rcc0, using real_digitizer in /mnt/ocpi_assets/exports/lib/dsp_comps/rcc/
  linux-x13_3-arm/real_digitizer_s.so dated Tue Feb 13 15:16:24 2018
  Instance 12 file_write (spec ocpi.core.file_write) on rcc container 1: rcc0, usin
  g file_write in /mnt/ocpi_core/exports/lib/components/rcc/linux-x13_3-arm/file_writ
  e_s.so dated Wed Feb 14 09:38:40 2018
Application XML parsed and deployments (containers and implementations) 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

