



OpenCPI & GNU Radio Integration for Heterogeneous Processing



David Pocratsky
9/14/2017

Agenda

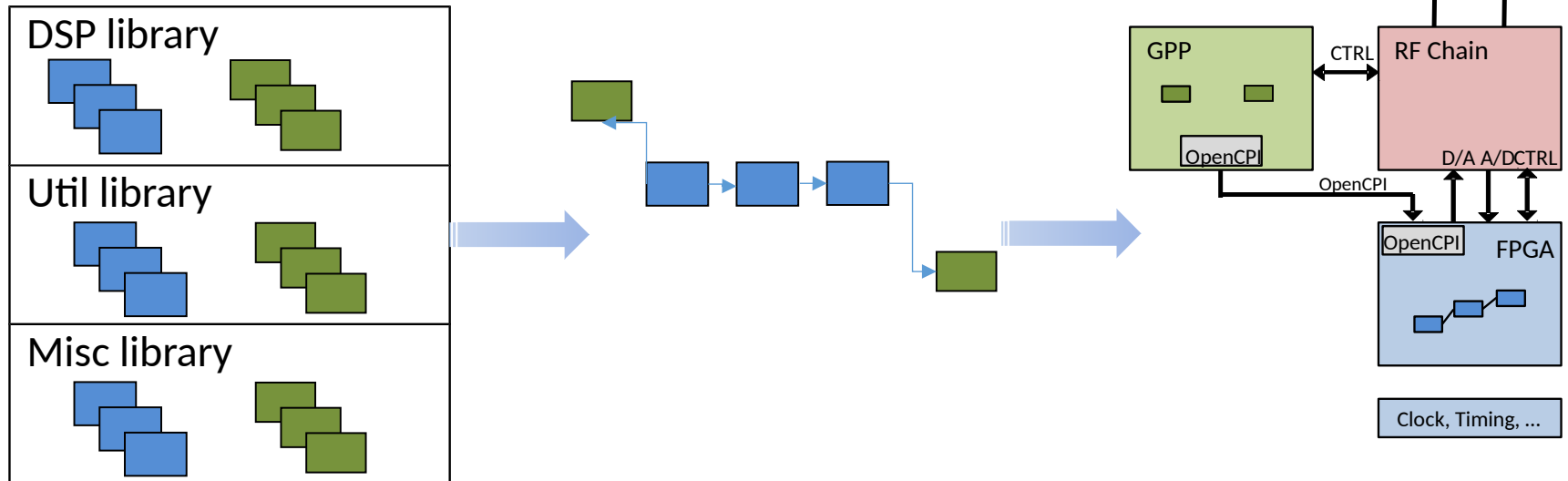
- Introduction to OpenCPI
- Comparisons to Existing Frameworks
- Motivation for GRC Integration
- OpenCPI and GRC (Demo)
- Road Map for Future Development
- How to Get More Information

Introduction to OpenCPI

- **Open Component Portability Infrastructure**
- Component-based framework for developing portable processing applications targeting various processing technologies
- Processing technologies can include multiple field programmable gate arrays (FPGAs), general purpose processors (GPPs), graphical processing units (GPUs)*, and a variety of interconnection technologies
 - Designed to be platform agnostic and easily adaptable to new platforms
- Integrates existing tools for FPGA development such as ISE, Vivado, XSim/ISim, ModelSim, Quartus II, etc.

*in development/coming soon

Introduction to OpenCPI

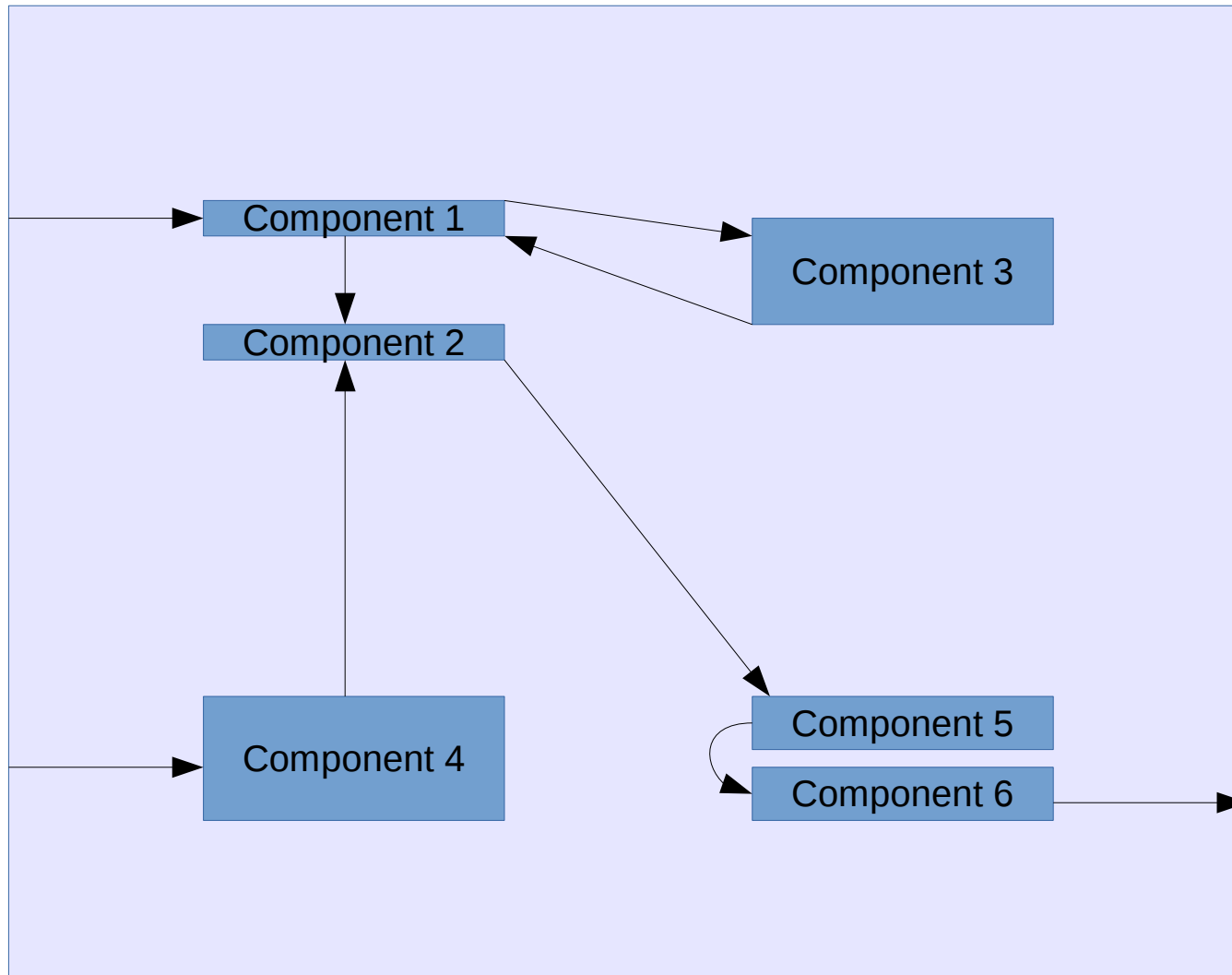


Portable, reusable components are developed in native language for the final target platform (i.e. VHDL/C++) independent of intended application

Applications are constructed from existing components

Application deploys to target platform, with components executing on disparate parts of platform

OpenCPI – Applications and Components

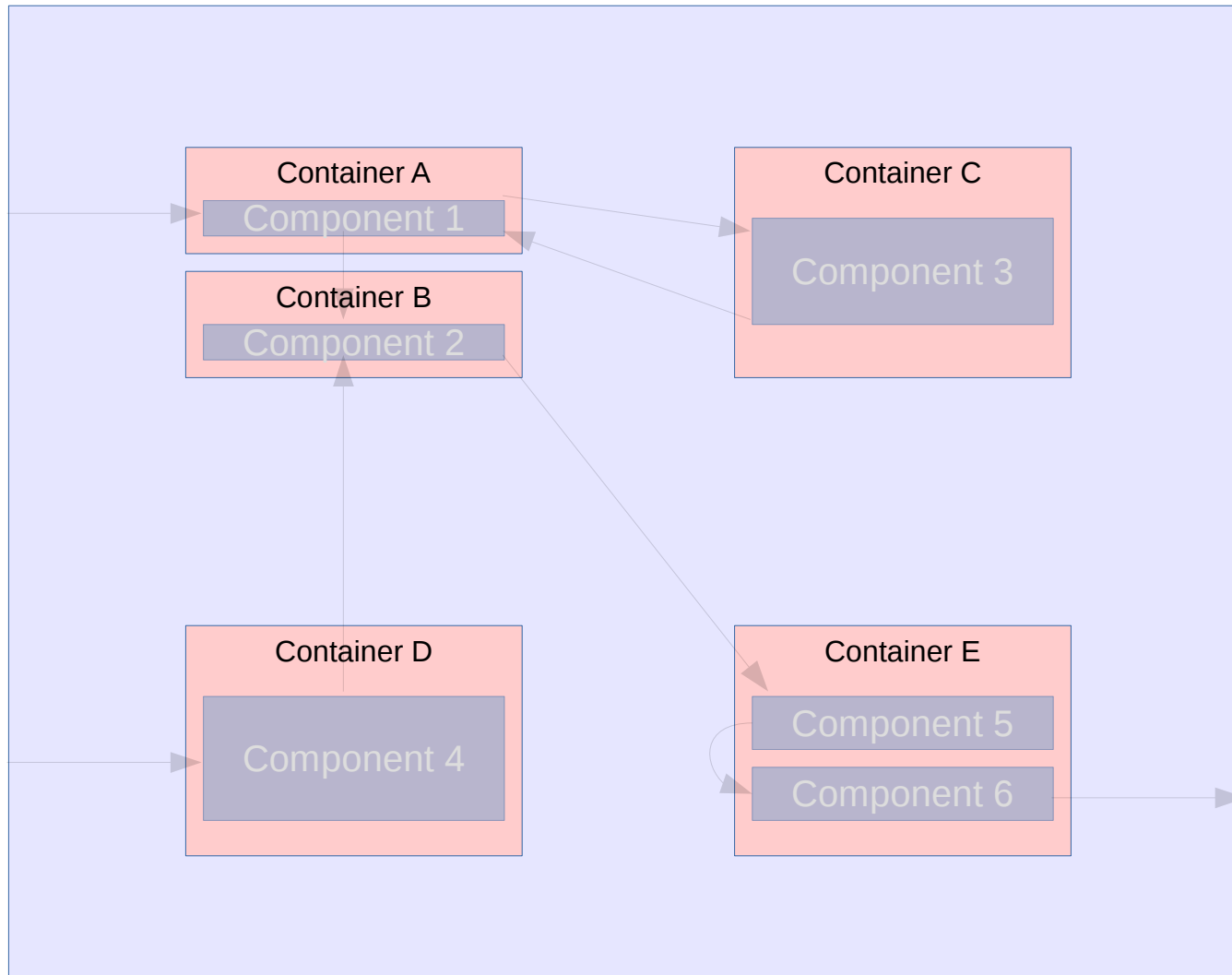


Application – A collection of interconnected component specs

Component – A specification of a function or operation implemented by Workers to run on specific platforms

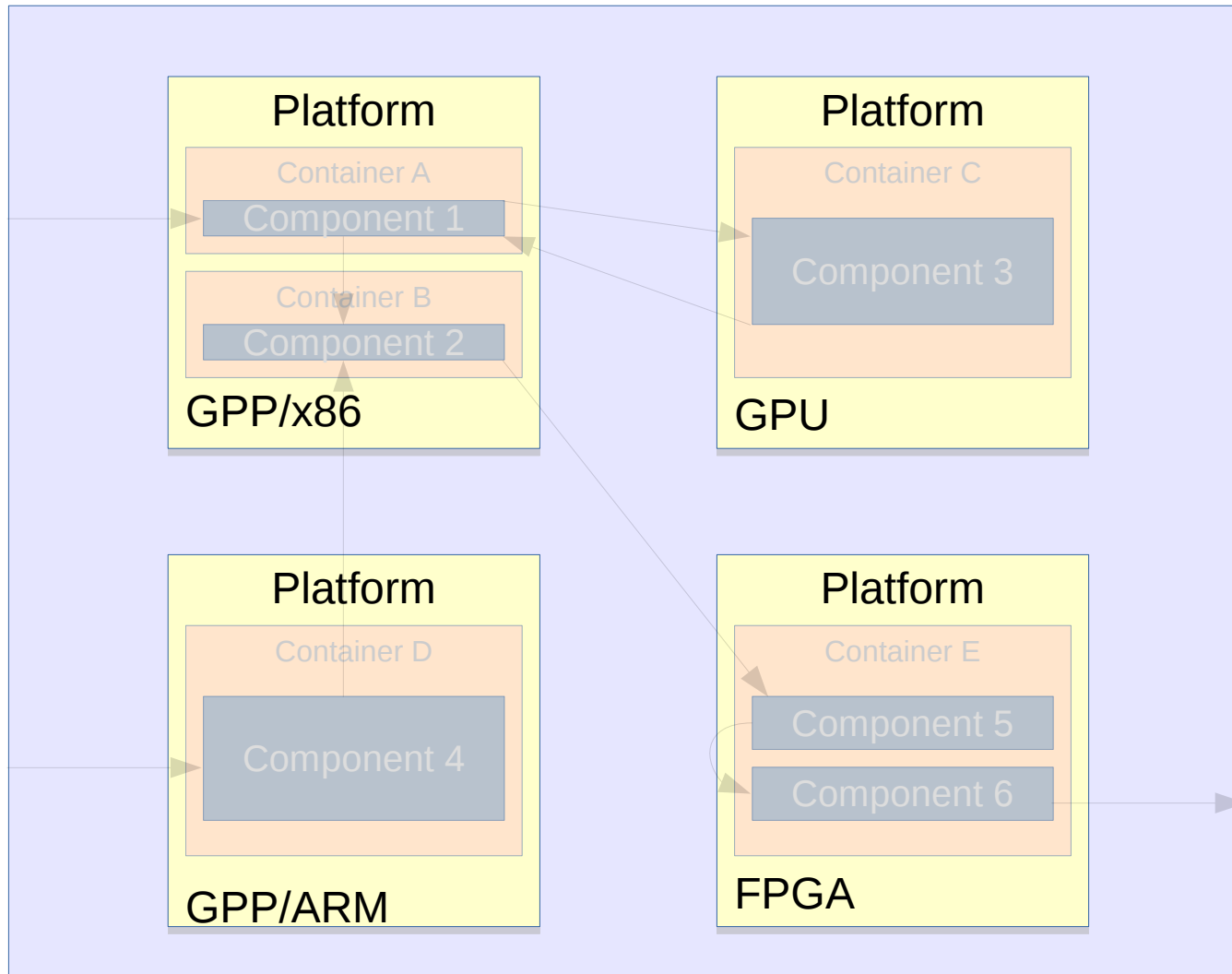
Worker – A concrete implementation of a component

OpenCPI – Containers and Assemblies



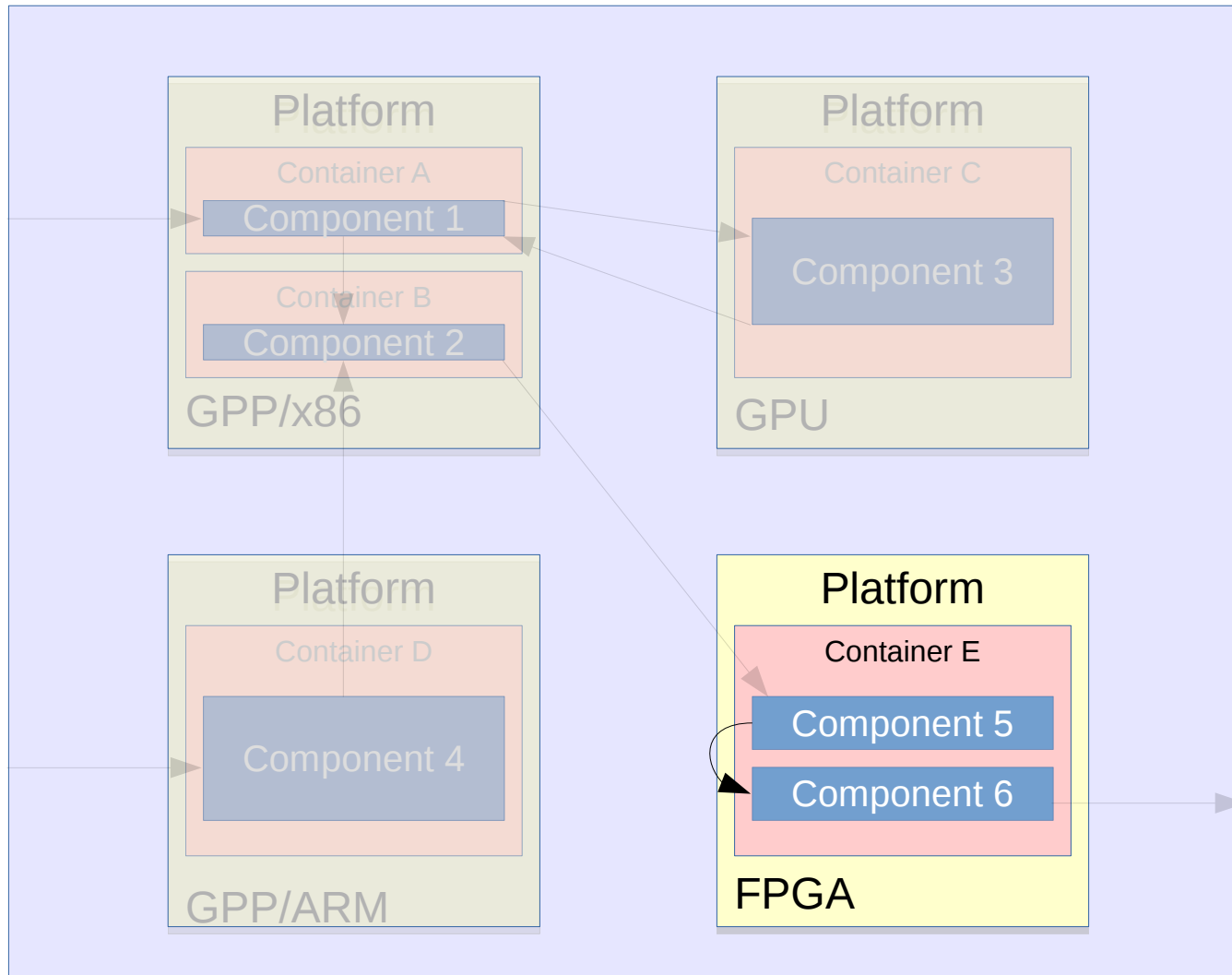
Container –
Execution
environment that
runs on some
platform that
executes workers

OpenCPI - Platforms



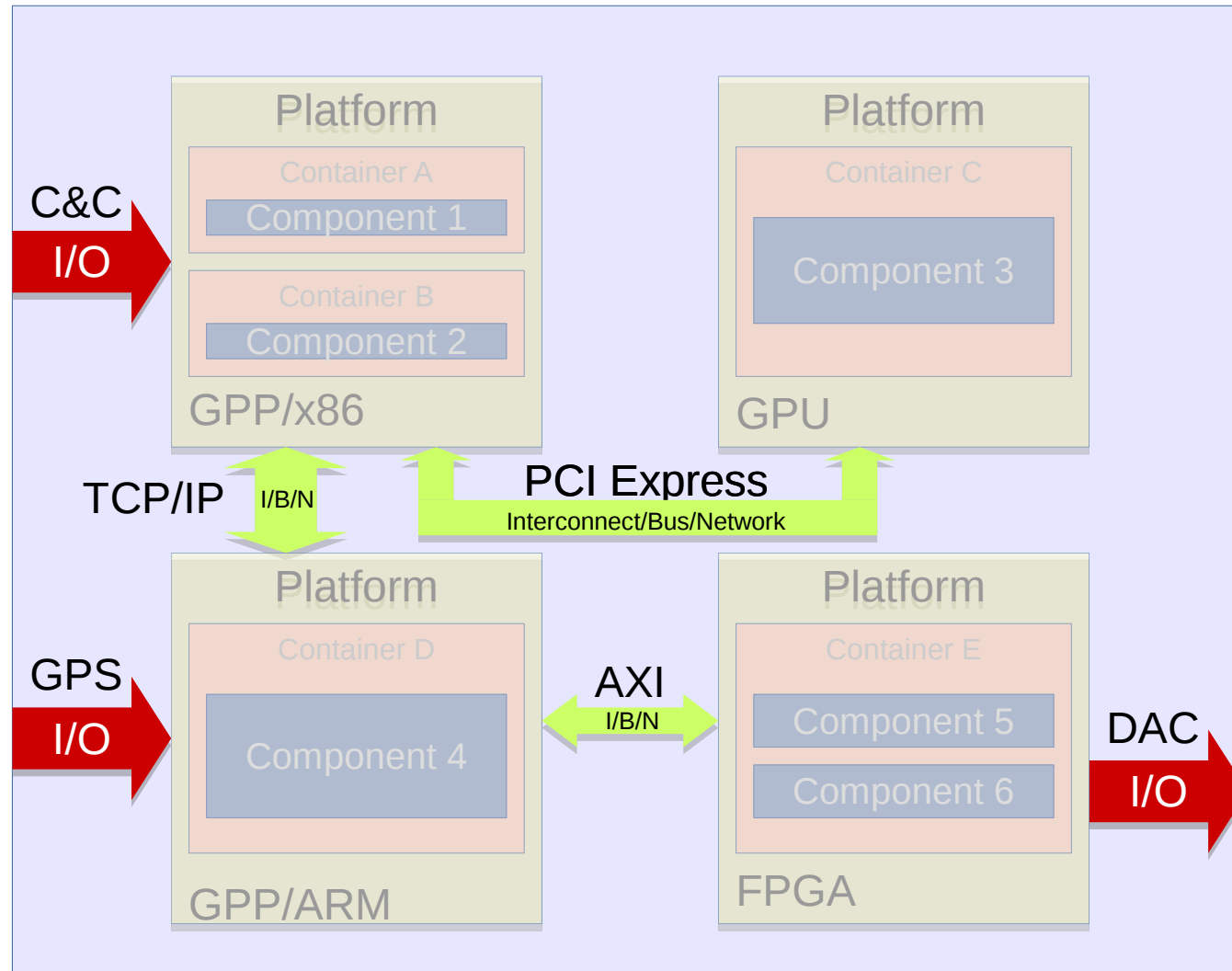
Platform –
Physical device
which houses one
or more
interconnected
Containers and
associated I/O

OpenCPI - Platforms



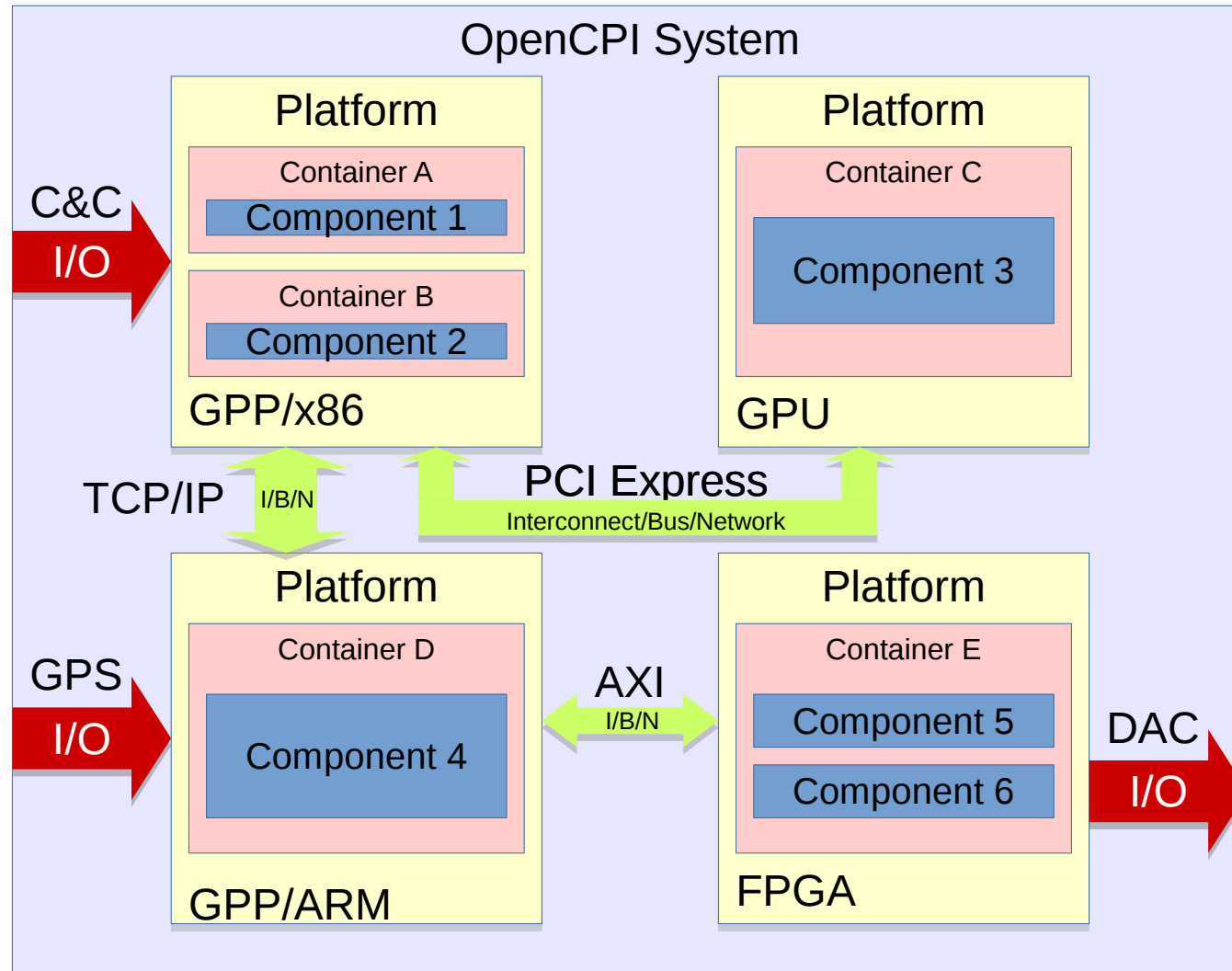
Assembly –
Collocated subset
of an application
built from one or
more HDL workers

OpenCPI – Interconnect/Bus/Network & I/O



Interconnect/Bus/
Network & I/O –
Physical
connections
between Platforms
that is typically
abstracted away
from the developer

OpenCPI – OpenCPI System



System – A collection of interconnected OpenCPI-enabled Platforms that can be used to execute Applications


Comparisons to Existing Frameworks

	GNU Radio	RF NoC	OpenCL	REDHAWK	OpenCPI
Software Support	X	X	X	X	X
FPGA Support		X	Supported with vendor specific extensions	Treats FPGA as black box	X
Access FPGA I/O (ADC/DAC)		X			X
GPU Support			X		Planned for future release
Platform Agnostic			X		X
Leverages Native Platform Language	X	X		X	X
Distributed				X	Support for distributed processing across embedded targets planned for future release

Motivation for GRC Integration

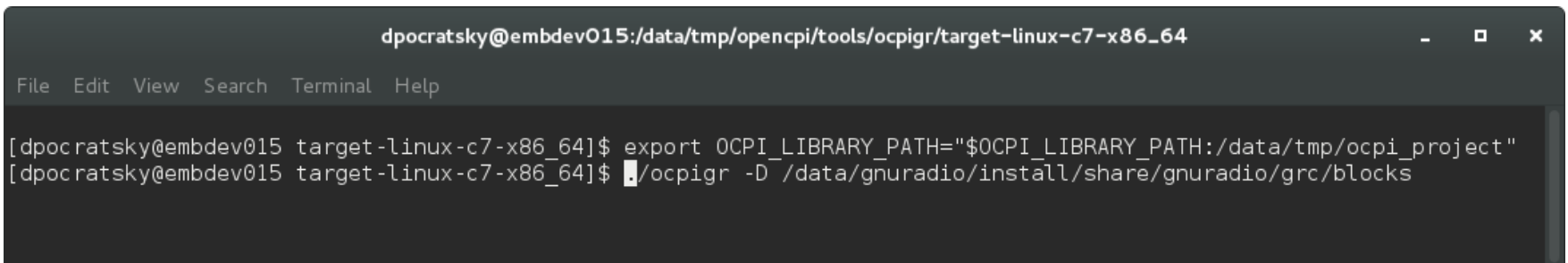
- Provide a front end GUI for the OpenCPI Application Developer to quickly test, design, and run OpenCPI Applications or Assemblies – Mostly work currently done by hand
- Leverage the existing work that was done in GRC that does not fall within OpenCPI's domain
- Introduce OpenCPI to the GNU Radio community in a way that is familiar and easy to use
- Enhance GRC to be able to run certain blocks on various hardware platforms such as FPGAs, GPPs, GPUs, etc

OpenCPI and GRC – General Outline

- Install OpenCPI and build the assets for the desired platforms
 - (Optional) Write and build your own OpenCPI workers specific to your application
 - Run ocpigr to import the workers into GRC as GRC blocks
 - Create flowgraph as you normally would with vanilla GRC
 - Set the deployment to target the desired containers for each block
 - Build/Run the application
- 
- The diagram consists of two blue curly braces on the right side of the slide. The top brace groups the first three bullet points and is labeled 'OpenCPI Documentation'. The bottom brace groups the last four bullet points and is labeled 'Covered in this presentation'.
- OpenCPI Documentation
- Covered in this presentation

OpenCPI and GRC – ocpigr Tool

- Tool in the OpenCPI repo that will parse all OpenCPI workers/components found in the OCPI_LIBRARY_PATH and translate their XML into GRC block xml
- Creates an OpenCPI Container Platform block for each platform that was encountered during parsing
- Takes advantage of the GRC Domain concept by making a domain for each Platform encountered during parsing

A terminal window with a dark background and light text. The title bar at the top reads 'dpocratsky@embdev015:/data/tmp/opencpi/tools/ocpigr/target-linux-c7-x86_64'. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows two lines of command execution. The first line is '[dpocratsky@embdev015 target-linux-c7-x86_64]\$ export OCPI_LIBRARY_PATH="\$OCPI_LIBRARY_PATH:/data/tmp/ocpi_project"'. The second line is '[dpocratsky@embdev015 target-linux-c7-x86_64]\$./ocpigr -D /data/gnuradio/install/share/gnuradio/grc/blocks'.

```
dpocratsky@embdev015:/data/tmp/opencpi/tools/ocpigr/target-linux-c7-x86_64
File Edit View Search Terminal Help
[dpocratsky@embdev015 target-linux-c7-x86_64]$ export OCPI_LIBRARY_PATH="$OCPI_LIBRARY_PATH:/data/tmp/ocpi_project"
[dpocratsky@embdev015 target-linux-c7-x86_64]$ ./ocpigr -D /data/gnuradio/install/share/gnuradio/grc/blocks
```

OpenCPI and GRC – ocpigr Tool

The screenshot displays the GNU Radio Companion (GRC) interface. The title bar reads "*untitled - GNU Radio Companion". The menu bar includes File, Edit, View, Run, Tools, and Help. The toolbar contains various icons for file operations, editing, and execution. On the left, there are two panels: "Options" and "Variable". The "Options" panel shows "ID: top_block" and "Generate Options: QT GUI". The "Variable" panel shows "ID: samp_rate" and "Value: 32k". On the right, a block tree is visible, showing a hierarchy of blocks. The "OpenCPI" block is expanded, revealing a list of sub-blocks: lab, local, ocp, bias, capture, comparator_complex, comparator_real, cons, cons_cc, Consumer, copy, devices, emulator, file_read, and file_read_msg. The "file_read" block is highlighted in blue. An orange callout box points to the "OpenCPI" block in the tree, containing the text: "The ocpigr tool imported the OpenCPI Component Specs under the 'OpenCPI' Block Tree".

*untitled - GNU Radio Companion

File Edit View Run Tools Help

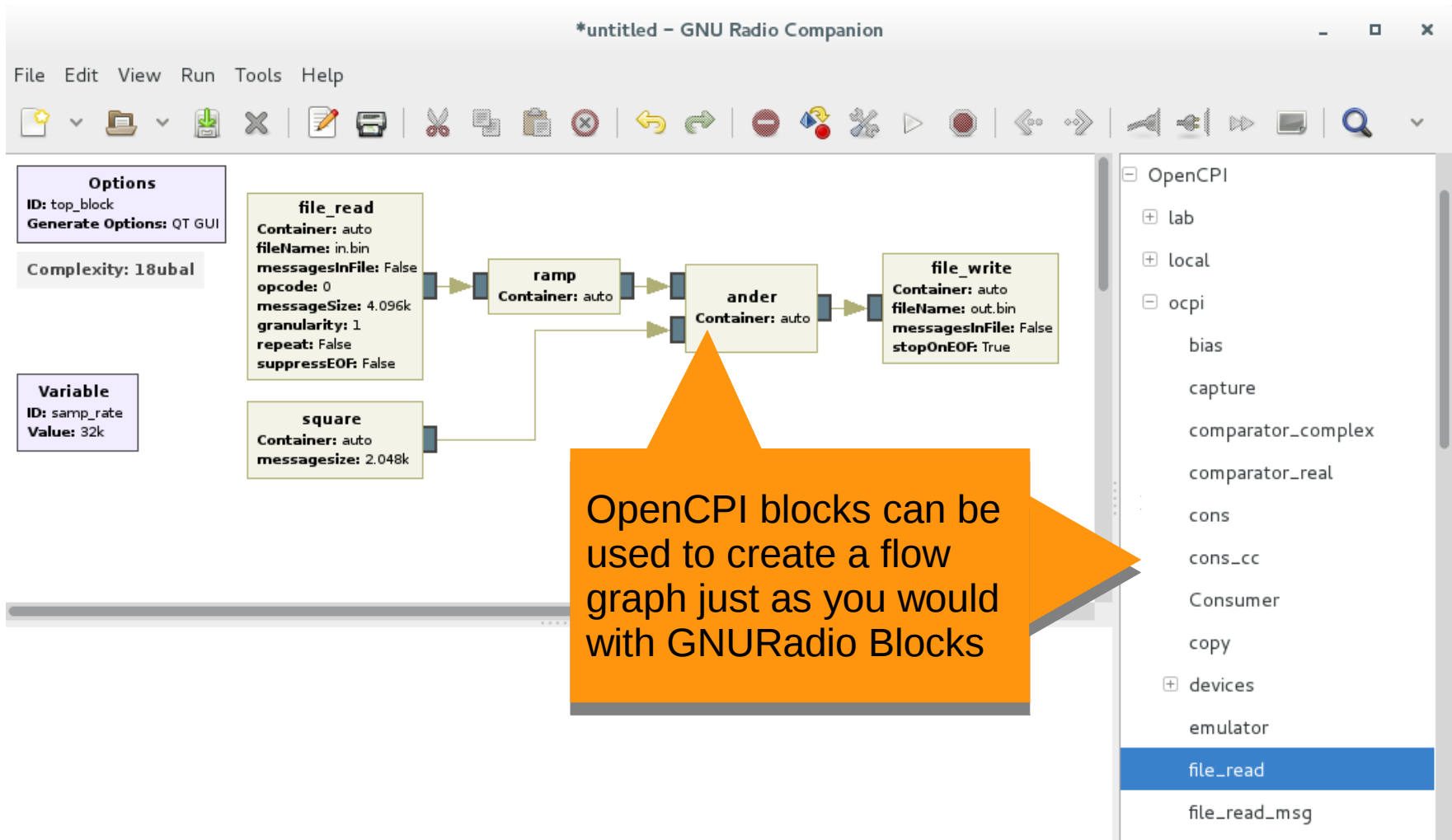
Options
ID: top_block
Generate Options: QT GUI
Complexity: 18ubal

Variable
ID: samp_rate
Value: 32k

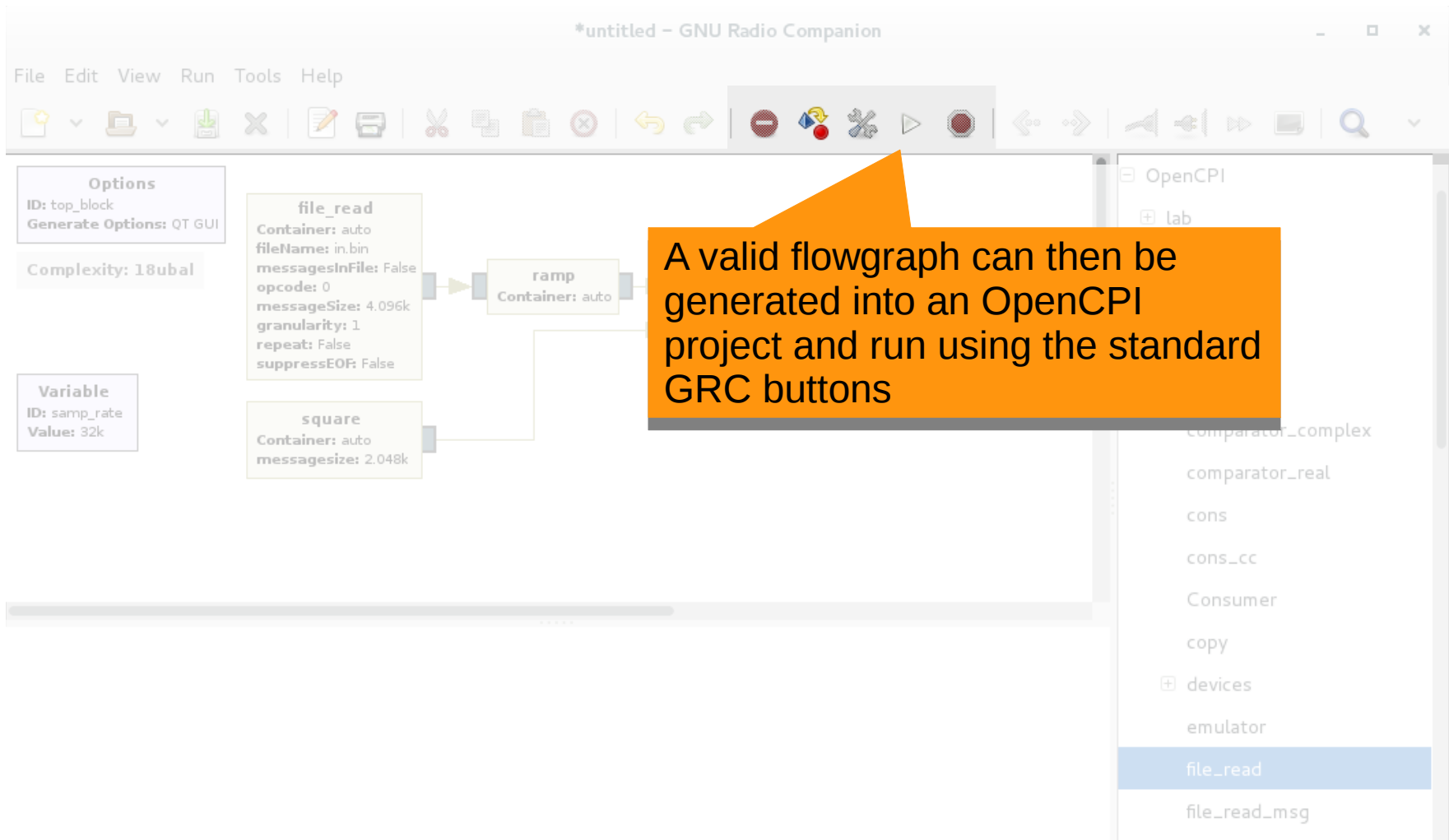
The ocpigr tool imported the OpenCPI Component Specs under the "OpenCPI" Block Tree

- OpenCPI
 - lab
 - local
 - ocpi
 - bias
 - capture
 - comparator_complex
 - comparator_real
 - cons
 - cons_cc
 - Consumer
 - copy
 - devices
 - emulator
 - file_read
 - file_read_msg

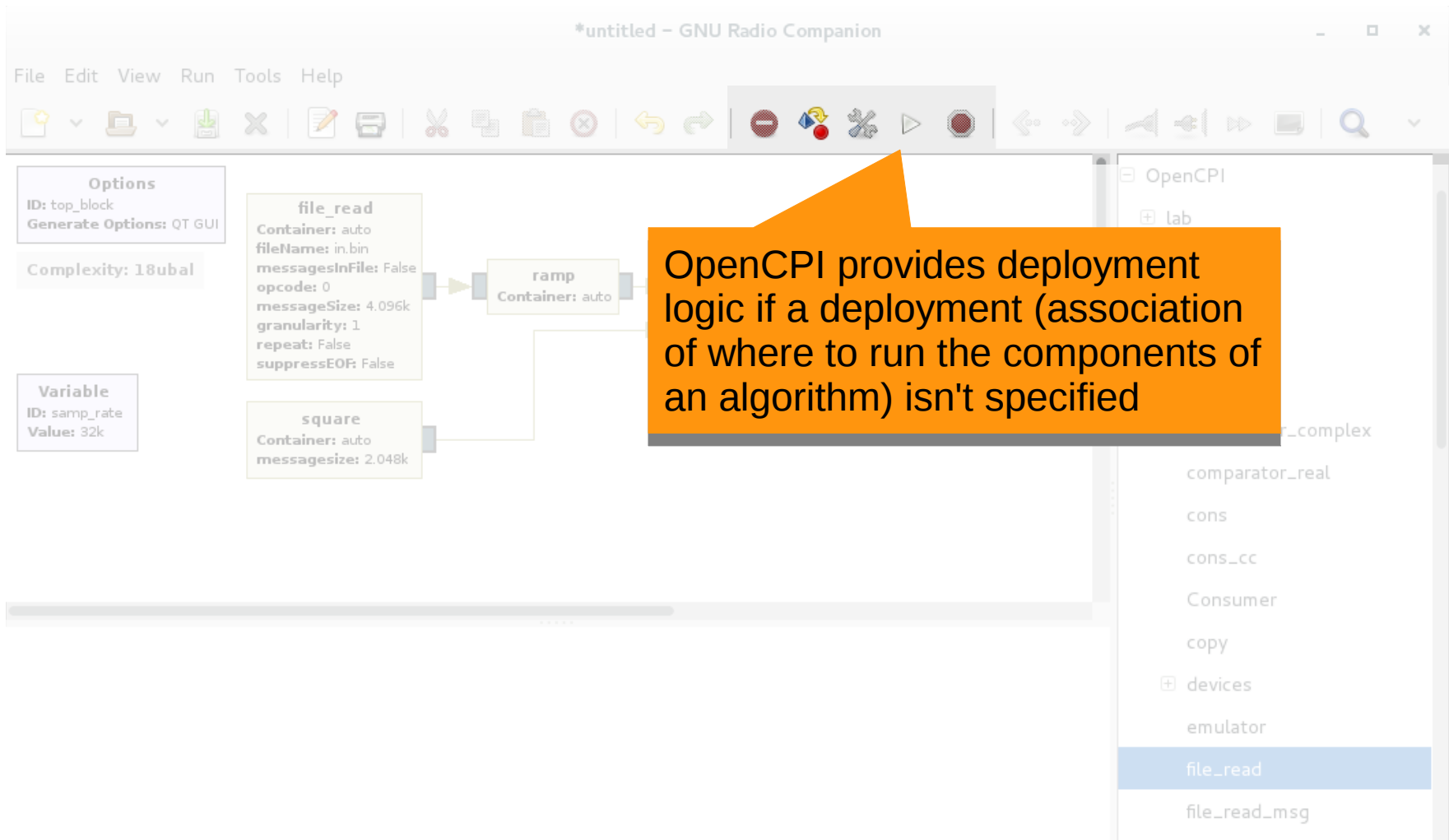
OpenCPI and GRC – Creating a Flowgraph



OpenCPI and GRC – Running a Flowgraph



OpenCPI and GRC – Running a Flowgraph



OpenCPI and GRC – Running a Flowgraph

untitled.grc - /data/gnuradio/install/bin - GNU Radio Companion

File Edit View Run Tools Help

Options
ID: top_block
Generate Options: QT GUI

file_read
Container: auto
fileName: in.bin

ander
Container: auto

file_write
Container: auto
fileName: out.bin
messagesInFile: False
stopOnEOF: True

GRC will autodetect OpenCPI blocks and use the appropriate generate and run options

Auto-detected OpenCPI block. Generate options have been overridden with the OpenCPI generator.
Generating: '/data/gnuradio/install/bin/top_block.xml'

Auto-detected OpenCPI block. Generate options have been overridden with the OpenCPI generator.
Executing: /opt/opencpi/cdk/bin/linux-c7-x86_64/ocpirun /data/gnuradio/install/bin/top_block_deployment.xml

OpenCPI

- lab
- local
- ocpi
 - bias
 - capture
 - comparator_complex
 - comparator_real
 - cons
 - cons_cc
 - Consumer
 - copy
- devices
 - emulator
 - file_read
 - file_read_msg

OpenCPI and GRC – Set Deployment

The screenshot shows the GNU Radio Companion (GRC) interface with a signal flow graph. The graph includes the following blocks:

- file_read** (green box): Container: rcc0, fileName: in.bin, messagesInFile: False, opcode: 0, messageSize: 4.096k, granularity: 1, repeat: False, suppressEOF: False.
- ramp** (orange box): Container: lsim_isim, ocpi_debug: False, ocpi_endian: little.
- square** (orange box): Container: lsim_isim, messagesize: 2.048k, ocpi_debug: False, ocpi_endian: little.
- ander** (orange box): Container: lsim_isim, ocpi_debug: False, ocpi_endian: little.
- file_write** (green box): Container: rcc0, fileName: out.bin, messagesInFile: False, stopOnEOF: True.

Connections: file_read connects to ramp, square connects to ramp, ramp connects to ander, and ander connects to file_write.

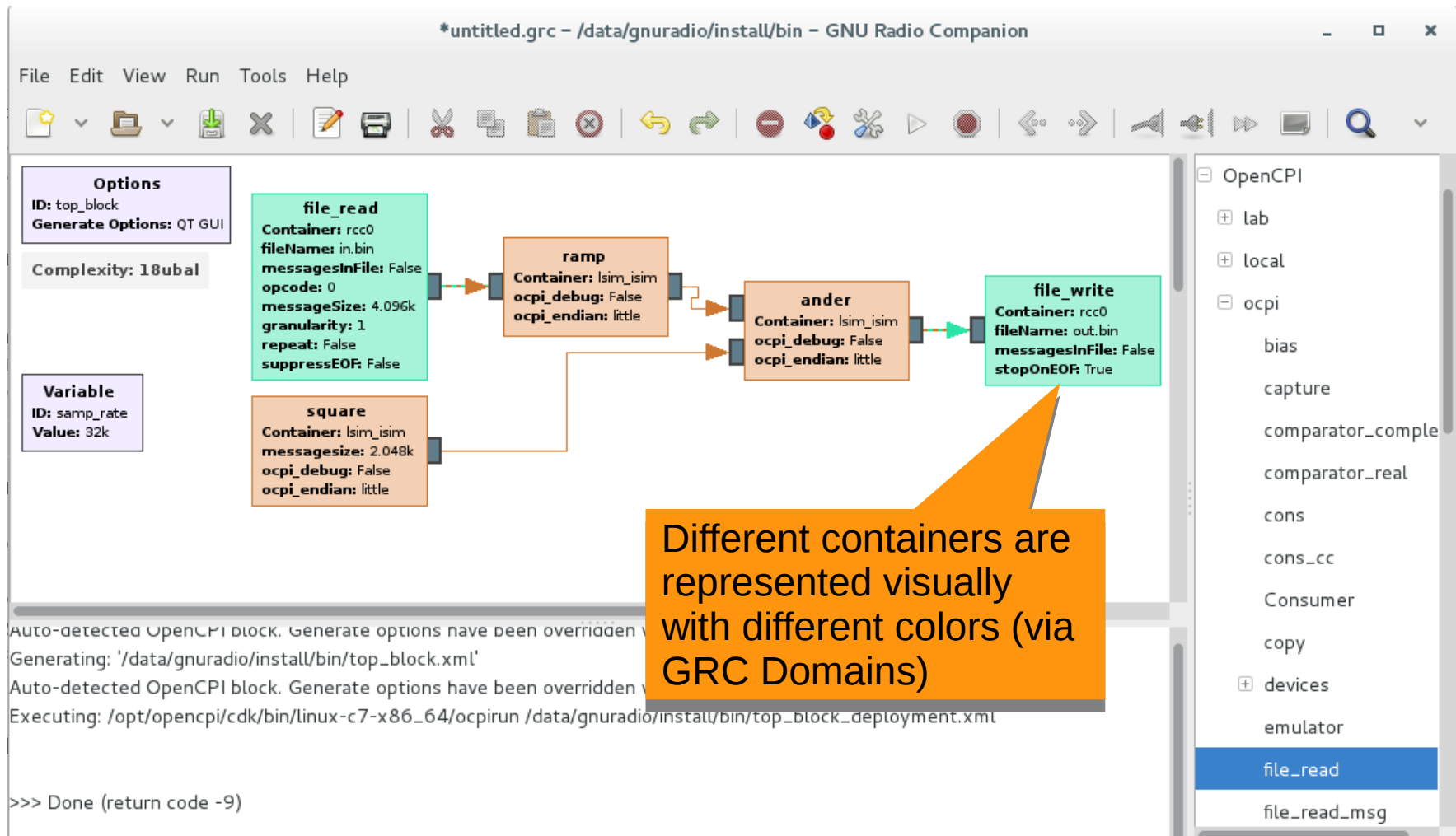
On the left, there are panels for **Options** (ID: top_block, Generate Options: QT GUI, Complexity: 18ubal) and **Variable** (ID: samp_rate, Value: 32k).

An orange callout box points to the 'Container' property of the 'square' block, containing the text: "Each GRC block has a 'Container' Property which dictates on which container in the targeted system the block should execute".

On the right, a sidebar shows a tree view of OpenCPI components, including lab, local, ocpi (with sub-items like bias, capture, comparator_comple, comparator_real, cons, cons_cc, Consumer, copy), devices, emulator, file_read (highlighted), and file_read_msg.

At the bottom, the terminal shows: >>> Done (return code -9).

OpenCPI and GRC – Set Deployment



OpenCPI and GRC – Build Assembly

*untitled.grc – /data/gnuradio/install/bin – GNU Radio Companion

File Edit View Run Tools Help

Options
ID: top_block
Generate Options: QT GUI
Complexity: 188ubal

Variable
ID: samp_rate
Value: 32k

file_read
Container: rcc0
fileName: in.bin
messagesInFile: False
opcode: 0
messageSize: 4.096k
granularity: 1
repeat: False
suppressEOF: False

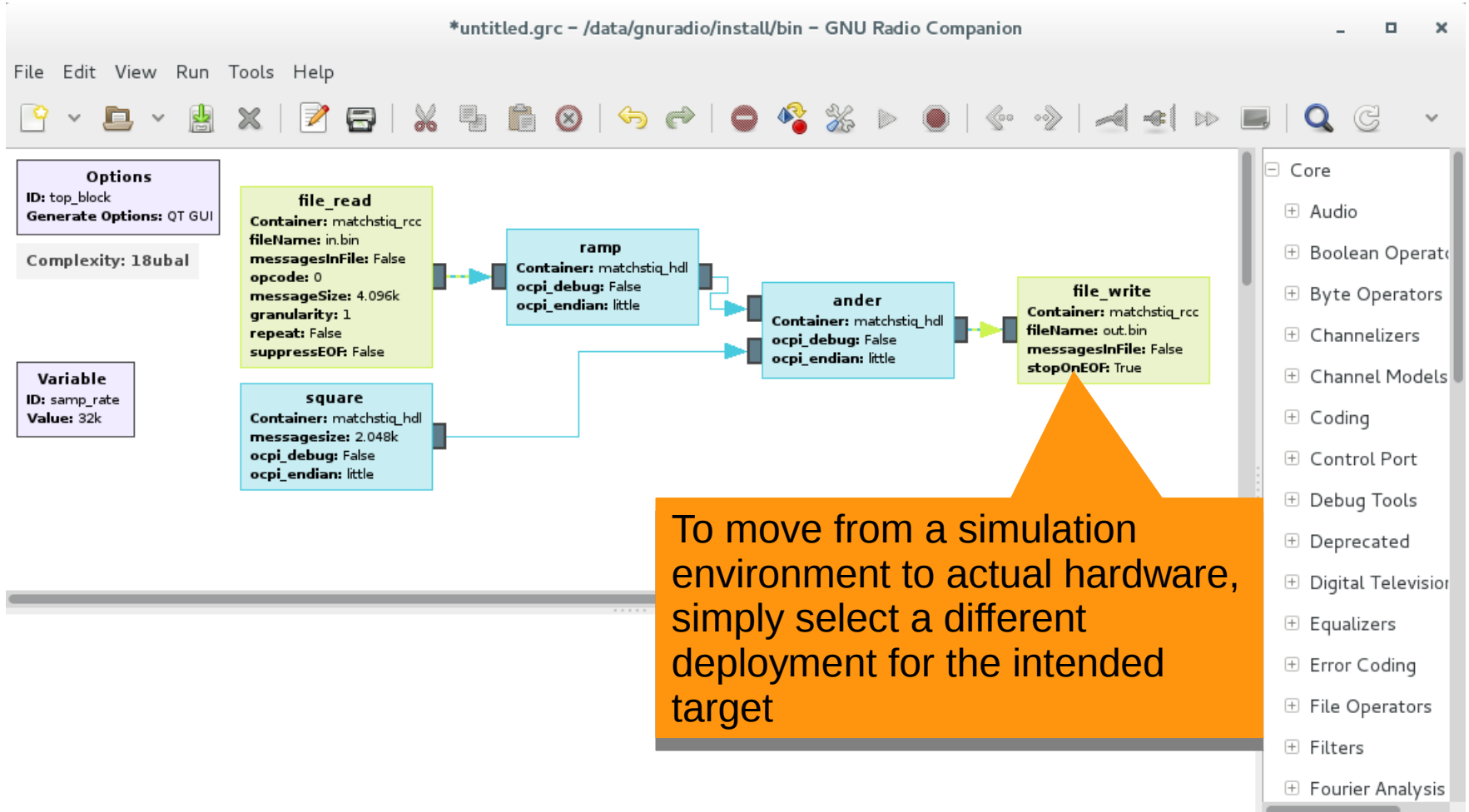
square
Container: lsim_isim
messageSize: 2.048k
ocpi_debug: False
ocpi_endian: little

When Targeting Containers that require Artifacts that need to be built (ex. HDL bit file), the “build” button becomes active and will orchestrate the project generation and building for you

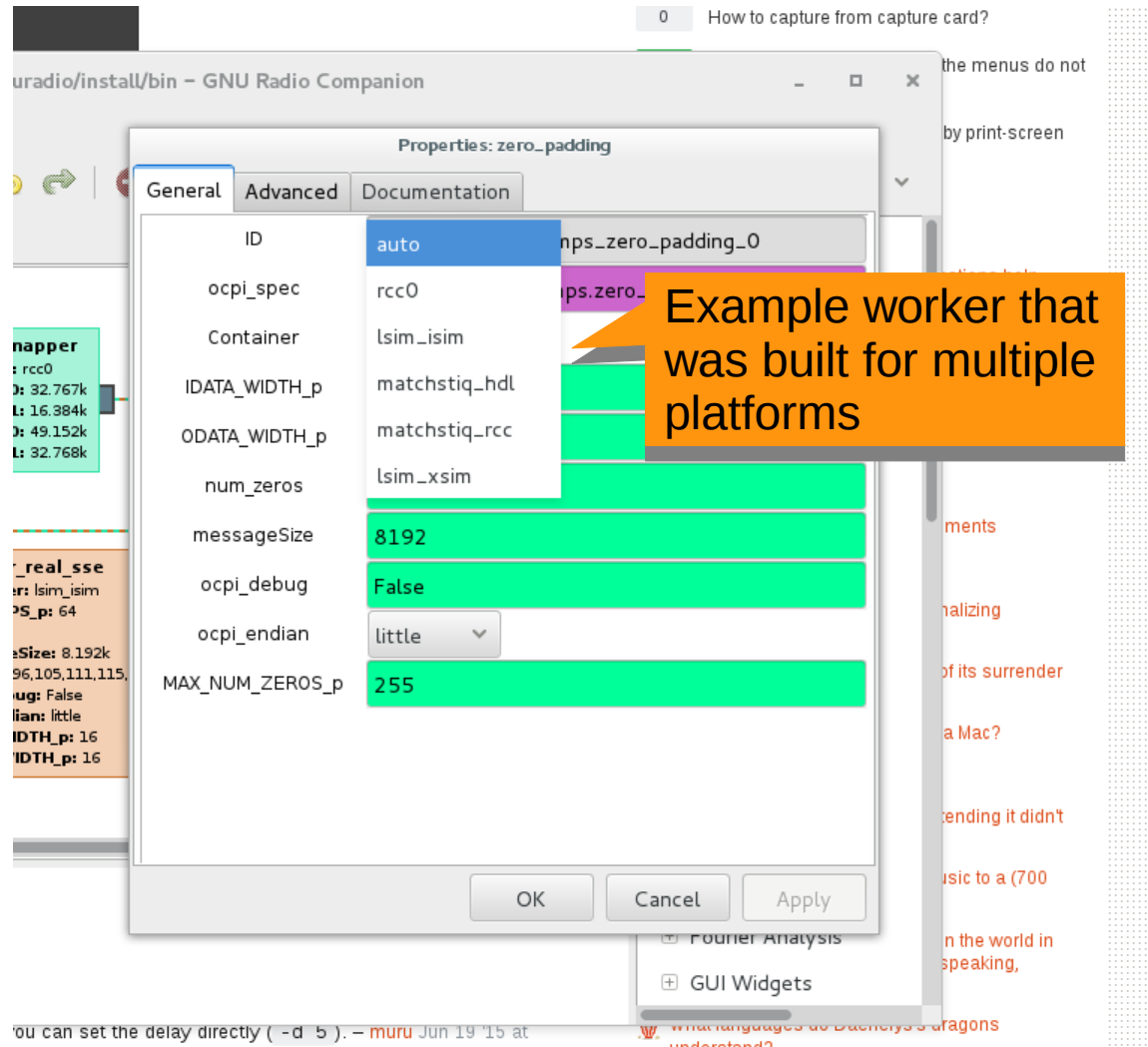
OpenCPI
+ lab
+ local
ocpi
bias
capture
comparator_comple
comparator_real
cons
cons_cc
Consumer
copy
+ devices
emulator
file_read
file_read_msg

Auto-detected OpenCPI block. Generate options have been overridden with the OpenCPI generator.
Generating: '/data/gnuradio/install/bin/top_block.xml'
Auto-detected OpenCPI block. Generate options have been overridden with the OpenCPI generator.
Executing: /opt/opencpi/cdk/bin/linux-c7-x86_64/ocpirun /data/gnuradio/install/bin/top_block_deployment.xml
>>> Done (return code -9)

OpenCPI and GRC – Set Deployment

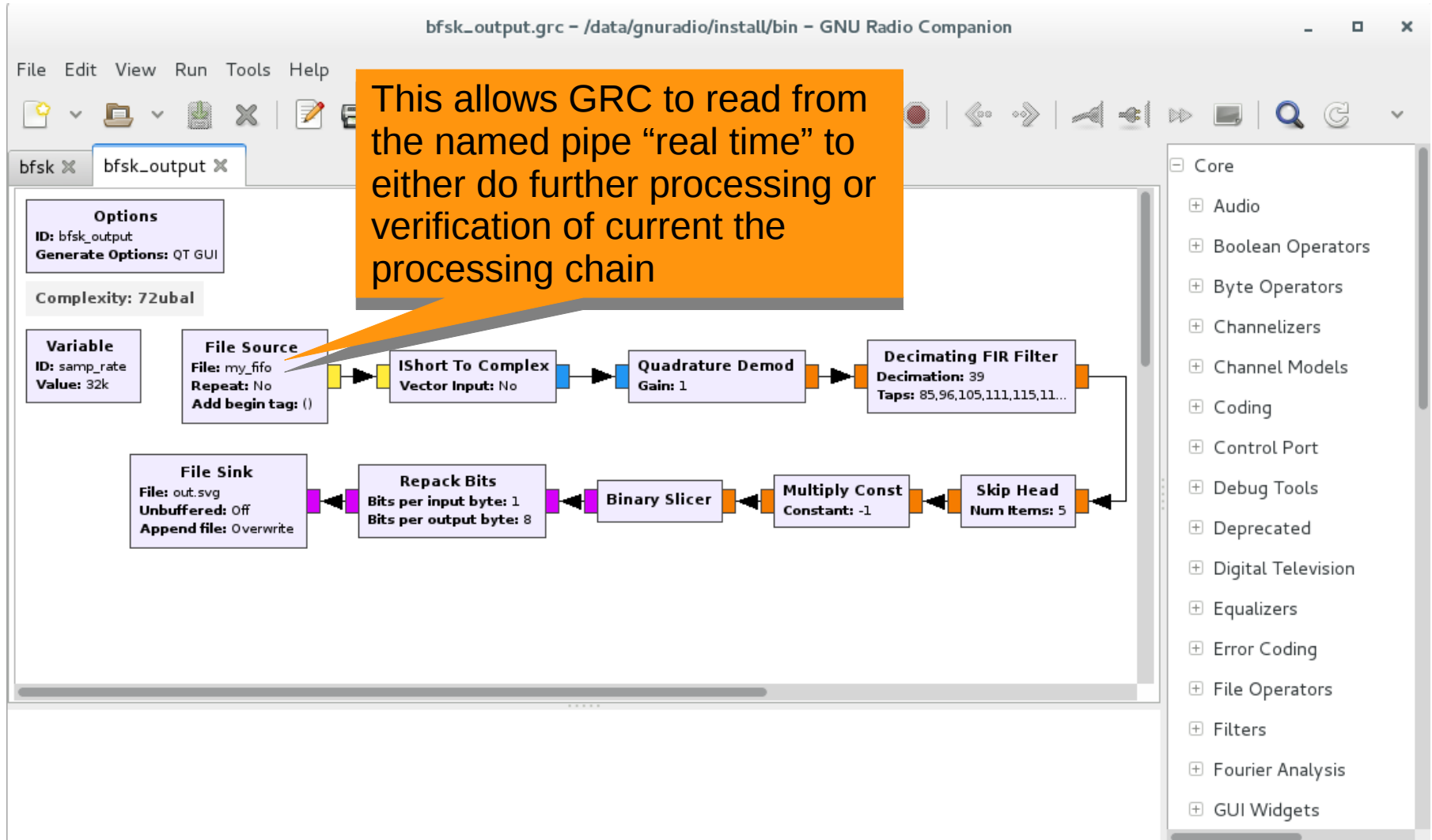


OpenCPI and GRC – Set Deployment

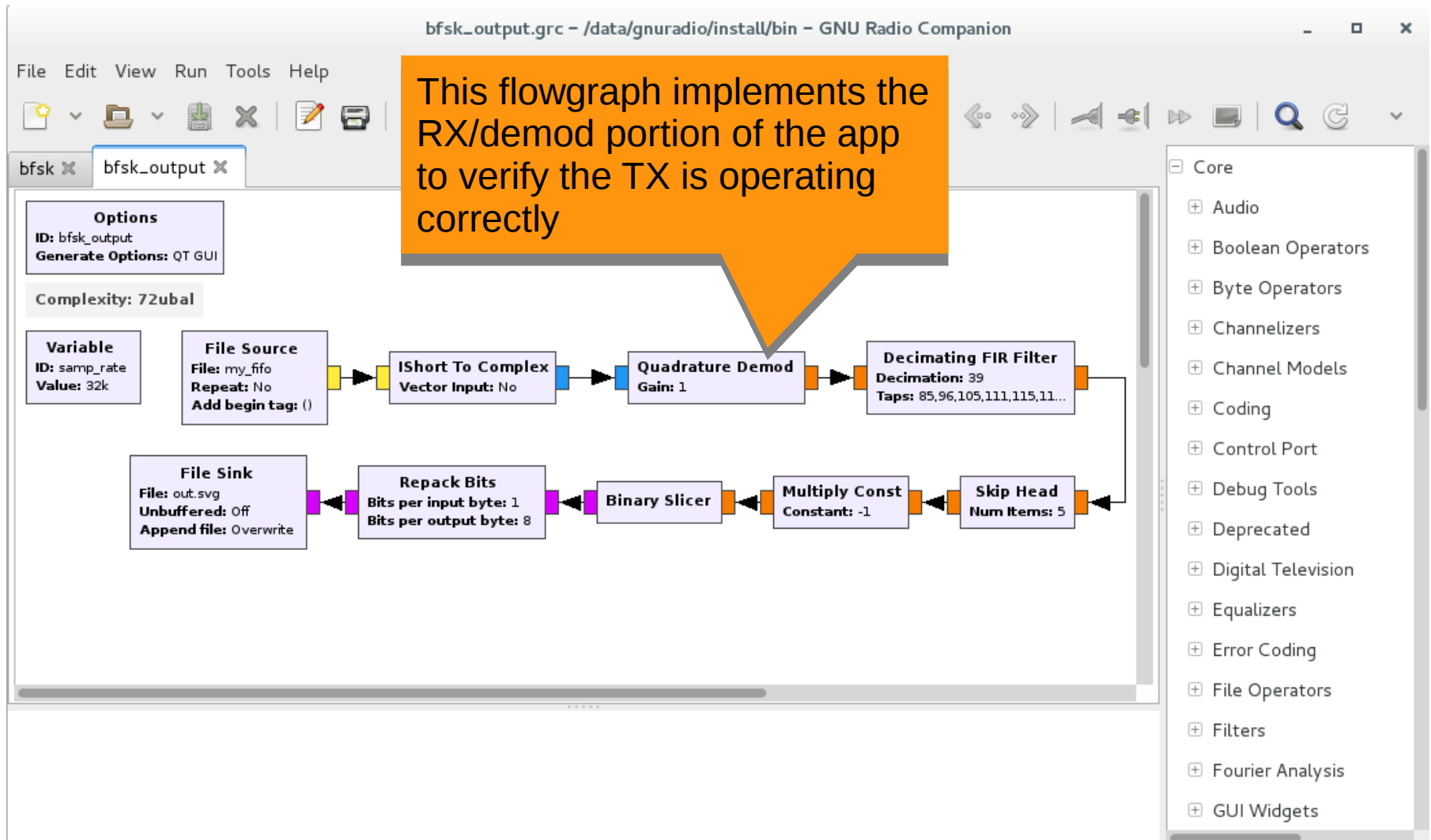




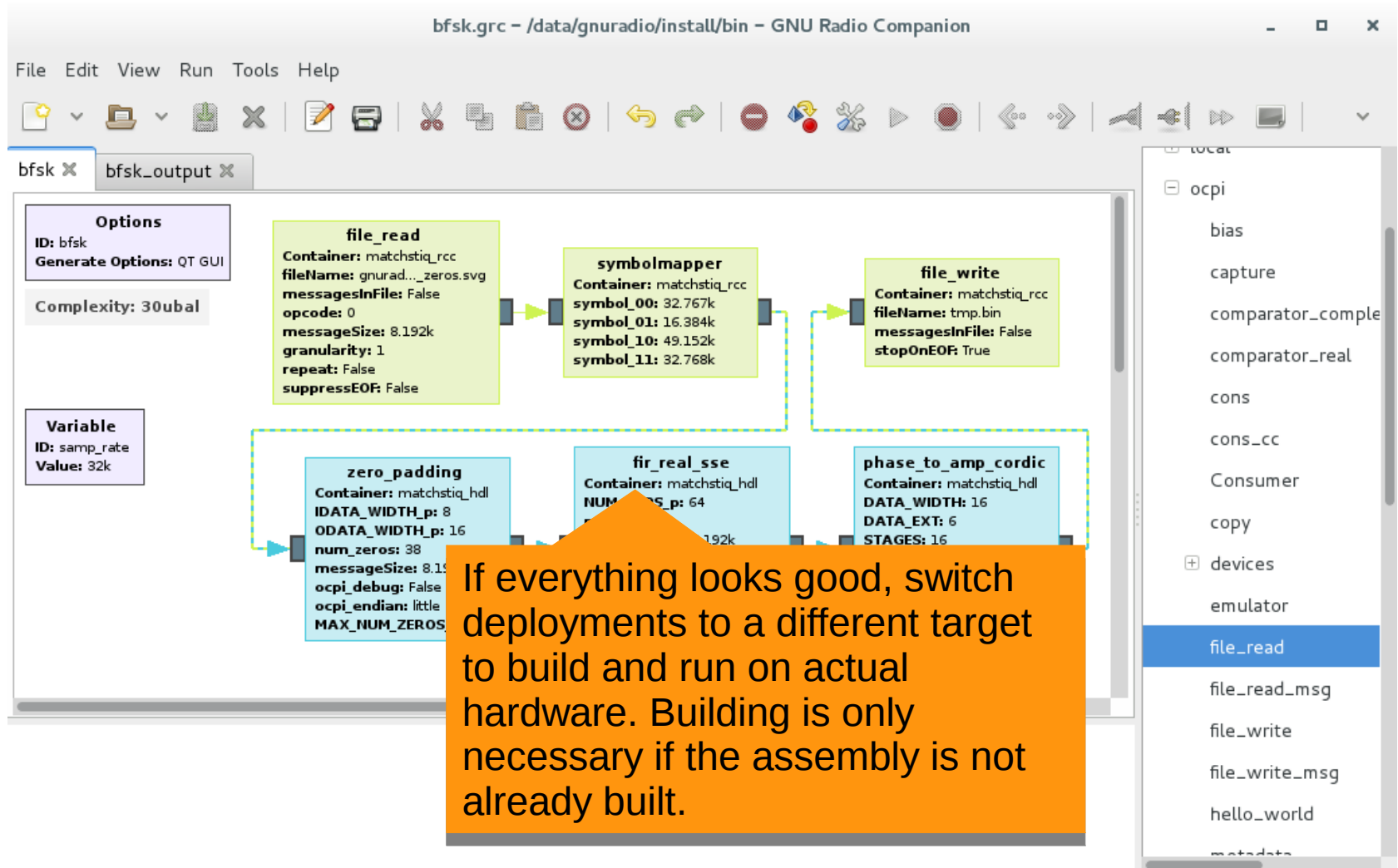
OpenCPI and GRC – Example BFSK App



OpenCPI and GRC – Example BFSK App



OpenCPI and GRC – Example BFSK App



Road Map for Future Development

- Have GRC and OpenCPI blocks talk directly or through a translation block
- Enable GRC to determine if the required artifacts are available and automatically build them if they are not
- Expose more features of OpenCPI to the GRC user
 - Select specific worker
 - Customize HDL Container
- Allow deployment across multiple systems on a network
- OpenCL/GPU support
- Support more platforms “out of the box”

How to Get More Information

- OpenCPI
 - <https://www.opencpi.org>
 - <https://github.com/opencpi/opencpi> (branch 2017.Q2)
 - issues@opencpi.org



Backup