Iris Software Radio Architecture

Paul Sutton

2nd February 2014
FOSDEM
Outline

- Iris Overview
- Iris Architecture
- Controllers
What is Iris?
What is Iris?

A Software Radio Architecture
What is Iris?

- Reconfigurable
- A Software Radio Architecture
What is Iris?

- Reconfigurable
- A Software Radio Architecture
- GPP – Based (primarily)
What is Iris?

- Reconfigurable
- Component-Based
- A Software Radio Architecture
- GPP – Based (primarily)
Iris Overview

What is Iris?

- Reconfigurable
- Component-Based
- A Software Radio Architecture
- GPP – Based (primarily)
- C++
What is Iris?

- Reconfigurable
- Component-Based
- A Software Radio Architecture
- Portable
- GPP – Based (primarily)
- C++
What is Iris?

- Reconfigurable
- Component-Based
- A Software Radio Architecture
- Portable
- Extensible
- GPP – Based (primarily)
- C++
Iris Overview

What is Iris?

- Reconfigurable
- Component-Based
- Portable
- Extensible
- GPP – Based (primarily)
- C++

open source
What can I do with Iris?
What can I do with Iris?
Iris Overview

What can I do with Iris?

SPECTRUM WARS
An Iris Demonstration

OFDM Modulation

OFDM Demodulation

Signature Detection

Bandwidth estimation using signatures
What can I do with Iris?

SPECTRUM WARS
An Iris Demonstration

OFDM Modulation
OFDM Demodulation
Signature Detection

Bandwidth estimation using signatures

CTVR
the telecommunications research centre
What can I do with Iris?

- Jacek Kibilda
- COST Short-Term Scientific Mission
- 2 weeks (no prior knowledge of Iris)
- DSA demo (primary user avoidance)
What can I do with Iris?

- Jacek Kibliada
- COST Short-Term Scientific Mission
- 2 weeks (no prior knowledge of Iris)
- DSA demo (primary user avoidance)
Iris Overview

What can I do with Iris?

- Jacek Kibilda
- COST Short-Term Scientific Mission
- 2 weeks (no prior knowledge of Iris)
- DSA demo (primary user avoidance)
Iris Overview

What can I do with Iris?

• Jacek Kibilda
• COST Short-Term Scientific Mission
• 2 weeks (no prior knowledge of Iris)
• DSA demo (primary user avoidance)
What can I do with Iris?

- Jacek Kibilda
- COST Short-Term Scientific Mission
- 2 weeks (no prior knowledge of Iris)
- DSA demo (primary user avoidance)

http://ledoyle.wordpress.com/2011/08/14/speedy-creation-of-a-cognitive-radio-demo/
The Basics...

- A GPP-based software radio architecture
  - Fundamental block is the component
The Basics...

- A GPP-based software radio architecture
  - Fundamental block is the component

- Most basic configuration:
  - A source component
  - A sink component
  - Some processing components
The Basics...

- A GPP-based software radio architecture
  - Fundamental block is the component

- Most basic configuration:
  - A source component
  - A sink component
  - Some processing components
The Basics...

• A GPP-based software radio architecture
  – Fundamental block is the component

• Most basic configuration:
  – A source component
  – A sink component
  – Some processing components

• XML document describes radio structure
Iris Architecture - The Basics

<softwareradio name="Radio1">

<engine name="phyengine1" class="phyengine">

<component name="filerawreader1" class="filerawreader">
  <parameter name="filename" value="testdata.txt"/>
  <port name="output1" class="output"/>
</component>

<component name="ofdmmod1" class="ofdmmodulator">
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="signalscaler1" class="signalscaler">
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="usrptx1" class="usrptx">
  <parameter name="frequency" value="5010000000"/>
  <parameter name="rate" value="1000000"/>
  <port name="input1" class="input"/>
</component>

<link source="filerawreader1.output1" sink="ofdmmod1.input1"/>
<link source="ofdmmod1.output1" sink="signalscaler1.input1"/>
<link source="signalscaler1.output1" sink="usrptx1.input1"/>

</engine>

</softwareradio>
<softwareradio name="Radio1">

<engine name="phyengine1" class="phyengine">

  <component name="filerawreader1" class="filerawreader">
    <parameter name="filename" value="testdata.txt"/>
    <port name="output1" class="output"/>
  </component>

  <component name="ofdmmod1" class="ofdmmodulator">
    <port name="input1" class="input"/>
    <port name="output1" class="output"/>
  </component>

  <component name="signalscaler1" class="signalscaler">
    <port name="input1" class="input"/>
    <port name="output1" class="output"/>
  </component>

  <component name="usrptx1" class="usrptx">
    <parameter name="frequency" value="5010000000"/>
    <parameter name="rate" value="1000000"/>
    <port name="input1" class="input"/>
  </component>

</engine>

<link source="filerawreader1.output1" sink="ofdmmod1.input1"/>
<link source="ofdmmod1.output1" sink="signalscaler1.input1"/>
<link source="signalscaler1.output1" sink="usrptx1.input1"/>

</softwareradio>
<softwareradio name="Radio1">

<engine name="phyengine1" class="phyengine">
    <component name="filerawreader1" class="filerawreader">
        <parameter name="filename" value="testdata.txt"/>
        <port name="output1" class="output"/>
    </component>

    <component name="ofdmmod1" class="ofdmmodulator">
        <port name="input1" class="input"/>
        <port name="output1" class="output"/>
    </component>

    <component name="signalscaler1" class="signalscaler">
        <port name="input1" class="input"/>
        <port name="output1" class="output"/>
    </component>

    <component name="usrptx1" class="usrptx">
        <parameter name="frequency" value="5010000000"/>
        <parameter name="rate" value="1000000"/>
        <port name="input1" class="input"/>
    </component>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1"/>
    <link source="ofdmmod1.output1" sink="signalscaler1.input1"/>
    <link source="signalscaler1.output1" sink="usrptx1.input1"/>

</engine>

</softwareradio>
<softwareradio name="Radio1">

<engine name="phyengine1" class="phyengine">
<component name="filerawreader1" class="filerawreader">
  <parameter name="filename" value="testdata.txt"/>
  <port name="output1" class="output"/>
</component>

<component name="ofdmmod1" class="ofdmmodulator">
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="signalscaler1" class="signalscaler">
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="usrptx1" class="usrptx">
  <parameter name="frequency" value="5010000000"/>
  <parameter name="rate" value="1000000"/>
  <port name="input1" class="input"/>
</component>

</engine>

<link source="filerawreader1.output1" sink="ofdmmod1.input1"/>
<link source="ofdmmod1.output1" sink="signalscaler1.input1"/>
<link source="signalscaler1.output1" sink="usrptx1.input1"/>

</softwareradio>
<softwareradio name="Radio1">

<engine name="phyengine1" class="phyengine">

<component name="filerawreader1" class="filerawreader">
  <parameter name="filename" value="testdata.txt"/>
  <port name="output1" class="output"/>
</component>

<component name="ofdmmod1" class="ofdmmodulator">
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="signalscaler1" class="signalscaler">
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="usrptx1" class="usrptx">
  <parameter name="frequency" value="5010000000"/>
  <parameter name="rate" value="1000000"/>
  <port name="input1" class="input"/>
</component>

<link source="filerawreader1.output1" sink="ofdmmod1.input1" />
<link source="ofdmmod1.output1" sink="signalscaler1.input1" />
<link source="signalscaler1.output1" sink="usrptx1.input1" />
</engine>
</softwareradio>
<softwareradio name="Radio1">

<engine name="phyengine1" class="phyengine">

<component name="filerawreader1" class="filerawreader">
    <parameter name="filename" value="testdata.txt"/>
    <port name="output1" class="output"/>
</component>

<component name="ofdmmod1" class="ofdmmodulator">
    <port name="input1" class="input"/>
    <port name="output1" class="output"/>
</component>

<component name="signalscaler1" class="signalscaler">
    <port name="input1" class="input"/>
    <port name="output1" class="output"/>
</component>

<component name="usrptx1" class="usrptx">
    <parameter name="frequency" value="5010000000"/>
    <parameter name="rate" value="1000000"/>
    <port name="input1" class="input"/>
</component>

<link source="filerawreader1.output1" sink="ofdmmod1.input1"/>
<link source="ofdmmod1.output1" sink="signalscaler1.input1"/>
<link source="signalscaler1.output1" sink="usrptx1.input1"/>
</engine>
</softwareradio>
<softwareradio name="Radio1">

<engine name="phyengine1" class="phyengine">

<component name="filerawreader1" class="filerawreader">
<parameter name="filename" value="testdata.txt"/>
<port name="output1" class="output"/>
</component>

<component name="ofdmmod1" class="ofdmmodulator">
<port name="input1" class="input"/>
<port name="output1" class="output"/>
</component>

<component name="signalscaler1" class="signalscaler">
<port name="input1" class="input"/>
<port name="output1" class="output"/>
</component>

<component name="usrptx1" class="usrptx">
<parameter name="frequency" value="5010000000"/>
<parameter name="rate" value="1000000"/>
<port name="input1" class="input"/>
</component>

</engine>

<link source="filerawreader1.output1" sink="ofdmmod1.input1" />
<link source="ofdmmod1.output1" sink="signalscaler1.input1" />
<link source="signalscaler1.output1" sink="usrptx1.input1" />

</softwareradio>
<softwareradio name="Radio1">

<engine name="phyengine1" class="phyengine">

  <component name="filerawreader1" class="filerawreader">
    <parameter name="filename" value="testdata.txt"/>
    <port name="output1" class="output"/>
  </component>

  <component name="ofdmmod1" class="ofdmmodulator">
    <port name="input1" class="input"/>
    <port name="output1" class="output"/>
  </component>

  <component name="signalscaler1" class="signalscaler">
    <port name="input1" class="input"/>
    <port name="output1" class="output"/>
  </component>

  <component name="usrptx1" class="usrptx">
    <parameter name="frequency" value="5010000000"/>
    <parameter name="rate" value="1000000"/>
    <port name="input1" class="input"/>
  </component>

  <link source="filerawreader1.output1" sink="ofdmmod1.input1"/>
  <link source="ofdmmod1.output1" sink="signalscaler1.input1"/>
  <link source="signalscaler1.output1" sink="usrptx1.input1"/>

</engine>

</softwareradio>
<softwareradio name="Radio1">

<engine name="phyengine1" class="phyengine">

<component name="filerawreader1" class="filerawreader">
    <parameter name="filename" value="testdata.txt"/>
    <port name="output1" class="output"/>
</component>

<component name="ofdmmod1" class="ofdmmodulator">
    <port name="input1" class="input"/>
    <port name="output1" class="output"/>
</component>

<component name="signalscaler1" class="signalscaler">
    <port name="input1" class="input"/>
    <port name="output1" class="output"/>
</component>

<component name="usrptx1" class="usrptx">
    <parameter name="frequency" value="5010000000"/>
    <parameter name="rate" value="1000000"/>
    <port name="input1" class="input"/>
</component>

<link source="filerawreader1.output1" sink="ofdmmod1.input1"/>
<link source="ofdmmod1.output1" sink="signalscaler1.input1"/>
<link source="signalscaler1.output1" sink="usrptx1.input1"/>
</engine>
</softwareradio>
<softwareradio name="Radio1">
  <engine name="phyengine1" class="phyengine">
    <component name="filerawreader1" class="filerawreader">
      <parameter name="filename" value="testdata.txt"/>
      <port name="output1" class="output"/>
    </component>
    <component name="ofdmmod1" class="ofdmmodulator">
      <port name="input1" class="input"/>
      <port name="output1" class="output"/>
    </component>
    <component name="signalscaler1" class="signalscaler">
      <port name="input1" class="input"/>
      <port name="output1" class="output"/>
    </component>
    <component name="usrptx1" class="usrptx">
      <parameter name="frequency" value="5010000000"/>
      <parameter name="rate" value="1000000"/>
      <port name="input1" class="input"/>
    </component>
  </engine>
  <link source="filerawreader1.output1" sink="ofdmmod1.input1"/>
  <link source="ofdmmod1.output1" sink="signalscaler1.input1"/>
  <link source="signalscaler1.output1" sink="usrptx1.input1"/>
</softwareradio>
Iris Architecture - The Basics

File Reader → OFDM Modulator → Scaler → USRP Transmitter
Iris Architecture - The Basics

- Data is passed between components in blocks – a **DataSet**
- Vector of data samples
- Metadata – e.g. timestamp, sample rate
Iris Architecture - The Basics
Iris Architecture - The Basics

Diagram showing the process from Source to Process.
Iris Architecture - The Basics

Source → Process

Source → Process
Engines
Engines

• An *engine*

  – The environment within which one or more components operates
  – Defines its own data-flow mechanism
  – Defines its own reconfiguration mechanisms
  – Runs one or more of its own threads
  – Provides a clean interface for the Iris system
Iris Architecture - Engines

Engines

- An *engine*
  - The environment within which one more components operates
  - Defines its own data-flow mechanism
  - Defines its own reconfiguration mechanisms
  - Runs one or more of its own threads
  - Provides a clean interface for the Iris system

Executes a section of the flow graph
Completely up to the engine how that’s done
Iris Architecture - Engines

- Two engine types:
  - PHY Engine
  - Stack Engine
Iris Architecture - Engines

• **PHY Engine**
  
  – Maximum flexibility
  – One thread per engine
  – Data-driven execution
  – One or more components per engine
  – Multiple component inputs / outputs
  – Unidirectional data flow
  – No fixed relationship between the inputs and outputs of a component
  – Flexible block sizes
Iris Architecture - Engines

PHY Engine

- Usrp Receiver
- Signal Analyser
- OFDM Demodulator
- File Writer
• Stack Engine
  
  – Network stack architecture
  
  – Components are layers within the stack
  
  – Each component runs its own thread
  
  – Bidirectional data flow
  
  – Supports e.g. MAC layer implementations
**Iris Architecture - Engines**

- **Transceiver Design**
  - Multiple engines
  - Tx and Rx PHY chains
  - Bidirectional stack
• So far...
  – We can create a radio
  – and reconfigure it manually
• So far...
  – We can create a radio
  – and reconfigure it manually

• How to reconfigure **dynamically**?
Controllers

- **Usrp Receiver**
- **Signal Analyser**
- **OFDM Demodulator**
- **File Writer**
Controllers

- Parameters

**PHY Engine**

- **Usrp Receiver**
- **Signal Analyser**
- **OFDM Demodulator**
- **File Writer**

Parametric reconfiguration e.g. Number of subcarriers
Controllers

• Events

Event
e.g. Detected waveform

Parametric reconfiguration
e.g. Number of subcarriers

PHY Engine

Usrp Receiver → Signal Analyser → OFDM Demodulator → File Writer
Controllers

Event
e.g. Detected waveform

Parametric reconfiguration
e.g. Number of subcarriers

PHY Engine

Usrp Receiver

Signal Analyser

OFDM Demodulator

File Writer

Controller
Why use Iris?
Why use Iris?

Quick Learning Curve
Why use Iris?

- Quick Learning Curve
- Open Source
Why use Iris?

- Open Source
- Quick Learning Curve
- Easy to Contribute
Why use Iris?

- Open Source
- Quick Learning Curve
- Easy to Contribute
- Small Project
Why use Iris?

- Quickly Implement Complex Systems
- Quick Learning Curve
- Easy to Contribute
- Open Source
- Small Project
Getting Started

- Code: https://github.com/softwareradiosystems
- Mailing Lists: http://www.softwareradiosystems.com/mailman/listinfo
- Blog: http://irissoftwareradio.wordpress.com/
Try it out

https://github.com/softwareradiosystems

Thank you

suttonpd@tcd.ie

paul@softwareradiosystems.com
Additional Material
Release 1.1.0
Release 1.1.0

- Liquid-DSP Components

```text
liquid-dsp

Software-Defined Radio Digital Signal Processing Library

liquid-dsp is a free and open-source digital signal processing (DSP) library designed specifically for software-defined radios on embedded platforms. The aim is to provide a lightweight DSP library that does not rely on a myriad of external dependencies or proprietary and otherwise cumbersome frameworks. All signal processing elements are designed to be flexible, scalable, and dynamic, including filters, filter design, oscillators, modems, synchronizers, and complex mathematical operations.
```