Iris Software Radio Architecture

Paul Sutton

19th February 2013

CREW Training Days

Brussels

CTVR | the telecommunications research centre

Outline

- Software Radio
- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- OFDM Specifics

Software Radio

- Software Radio
- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- OFDM Specifics

Software Radio

Software Radios
Survey, Critical Evaluation and Future Directions

J. Mitola, III
Chief Scientist, Electronic Systems
E-Systems, Melpar Division
11225 Waples Mill Road
Fairfax, VA 22030

ABSTRACT

A software radio is a set of Digital Signal Processing (DSP) primitives, a metalevel system for combining the primitives into communications systems functions (transmitter, channel model, receiver . . .) and a set of target processors on which the software radio is hosted for real-time communications. Typical applications include speech/music, modems, packet radio, telemetry and High Definition Television. Low cost high performance DSP chips promote delivery of enhanced communications services as software radios. Time to incorporate a new service into a product is reduced dramatically using this approach. Low costs and new services will continue to increase demand for software radio tool sets and CAD environments.

This paper relates performance of enabling hardware technologies to software radio requirements, portending a decade of shift from hardware radios toward software intensive approaches. Such approaches require efficient use of computational resources through topological consistency of radio functions and host architectures. This leads to a layered

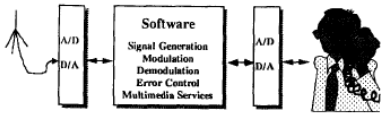


Fig. 1. An Idealized Software Radio

at handset allow all radio transmit, receive, signal generation, modulation/demodulation, timing, control, coding and decoding functions to be performed in software.

This "software" radio of course includes many non-DSP hardware components like RF conversion, RF distribution, anti-aliasing filters, power handling, etc. But the increased performance and continually dropping costs of the enabling technologies of A/D and D/A converters, high speed digital signal distribution, DSP chips and embedded computing are facilitating a shift toward software intensive approaches especially in large scale telesystems applications.

- First IEEE paper
  - Mitola, 1993

The Software Radio Architecture

As communications technology continues its rapid transition from analog to digital, more functions of contemporary radio systems are implemented in software, leading toward the software radio. What distinguishes software radio architectures? What new capabilities are more economically accessible in software radios than digital radios? What are the pitfalls? And the prognosis?

Joe Mitola

We are poised on the threshold of another revolution in radio systems engineering. Throughout the '70s and '80s radio systems migrated from analog to digital in almost every respect from system control to source and channel coding to hardware technology. And now the software radio revolution extends these horizons by liberating radio-based services from chronic dependency on hard-wired characteristics, including frequency band, channel bandwidth, and channel coding. This liberation is accomplished through a combination of techniques that includes multi-band antennas and RF conversion; wideband Analog to Digital (A/D) and Digital to Analog (D/A) conversion (A/D/A conversion); and the implementation of IF, baseband, and bitstream processing functions in general-purpose programmable processors. The resulting software-defined radio (or "software radio") in part extends the evolution of programmable hardware, increasing flexibility via increased programmability. And, in part, it represents an ideal that may never be fully implemented but that nevertheless simplifies and existence of a large commercial base which sometimes fails to emerge in spite of openness. As system complexity increases, architecture becomes more critical because of its power to either simplify and facilitate system development (a "powerful" architecture) or to complicate development and impede progress (a "weak" architecture).

Radio architectures may be plotted in the phase space of network organization versus channel data rate, as shown in Fig. 1. These architectures have evolved from early point-to-point and relatively chaotic peer networks (e.g., citizens band and push-to-talk mobile military radio networks) toward more hierarchical structures with improved service quality. In addition, channel data rates continue to increase through multiplexing and spectrum spreading. In a multiple-hierarchy application, a single radio unit, typically a mobile terminal, participates in more than one network hierarchy. A software radio terminal, for example, could operate in a GSM network, an AMPS network, and a future satellite mobile network. Multiband multimode military radios and future Personal Communications Systems (PCS) that seamlessly integrate multimedia services across such diverse

JOE MITOLA is a consulting scientist with The MITRE Corporation.

SPW is a registered trade-

- IEEE Comms Magazine
  - May 1995
  - Special Issue
  - Mitola Guest Editor

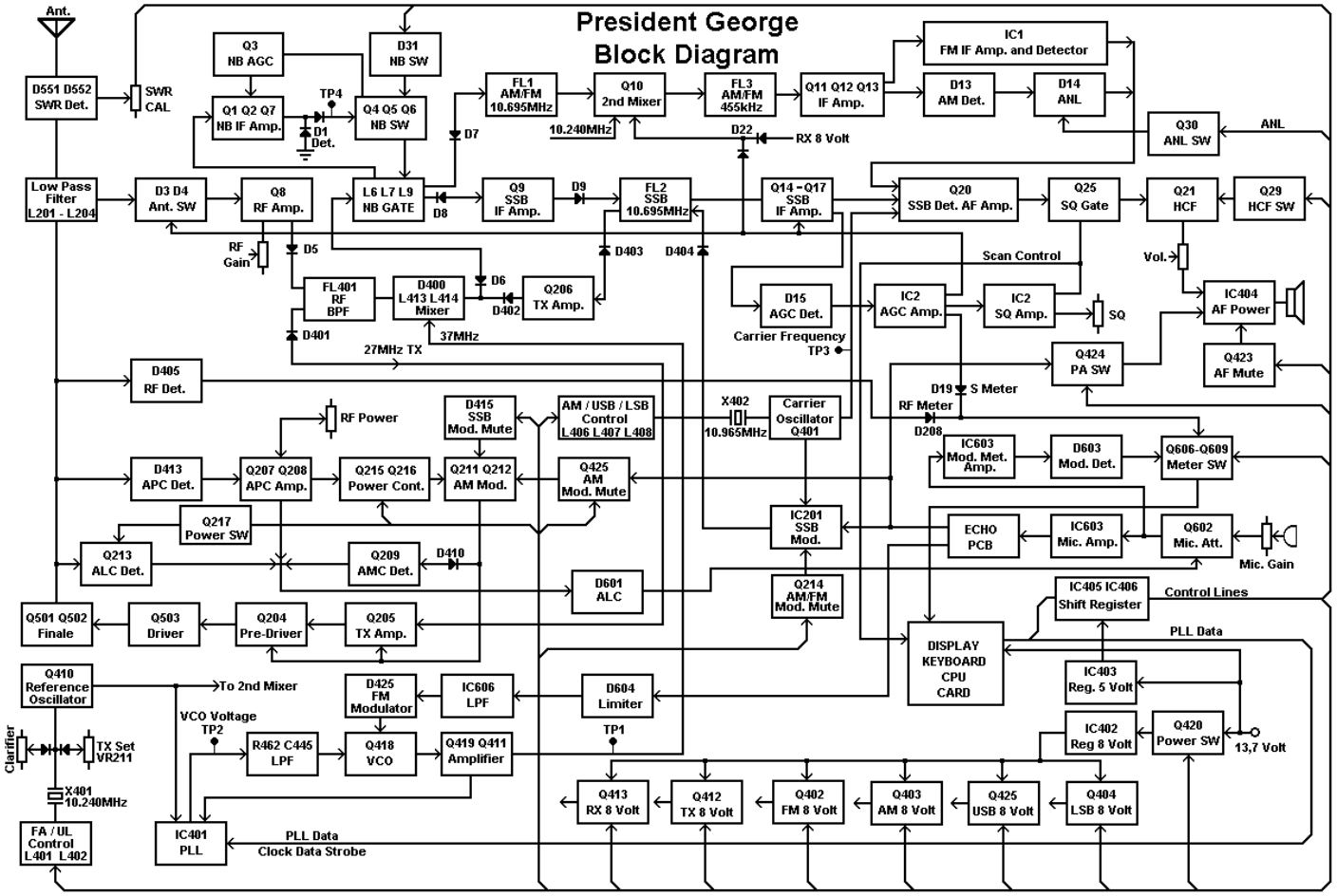
CREW

CTVR | the telecommunications research centre

Analog Radio

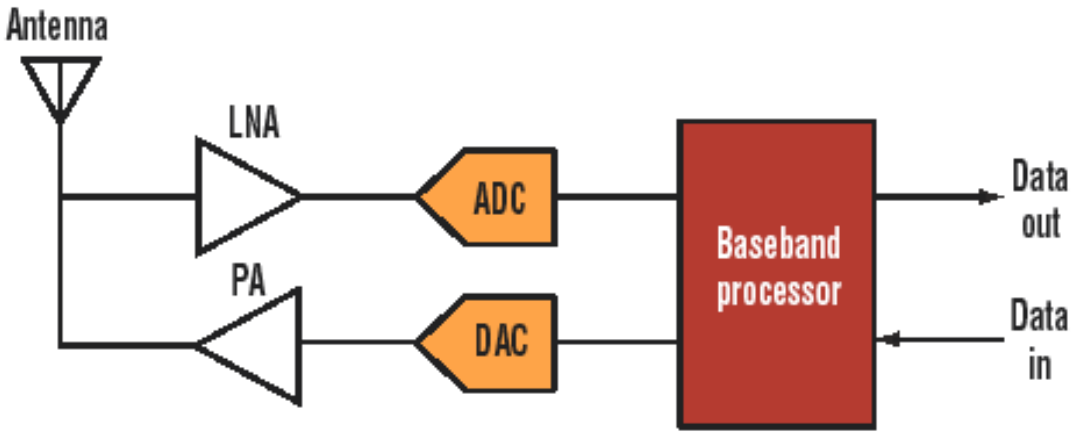
Source: www.flux.utah.edu/~rolke/content/CE-seminar.ppt

Analog Radio



Source: www.flux.utah.edu/~rolke/content/CE-seminar.ppt

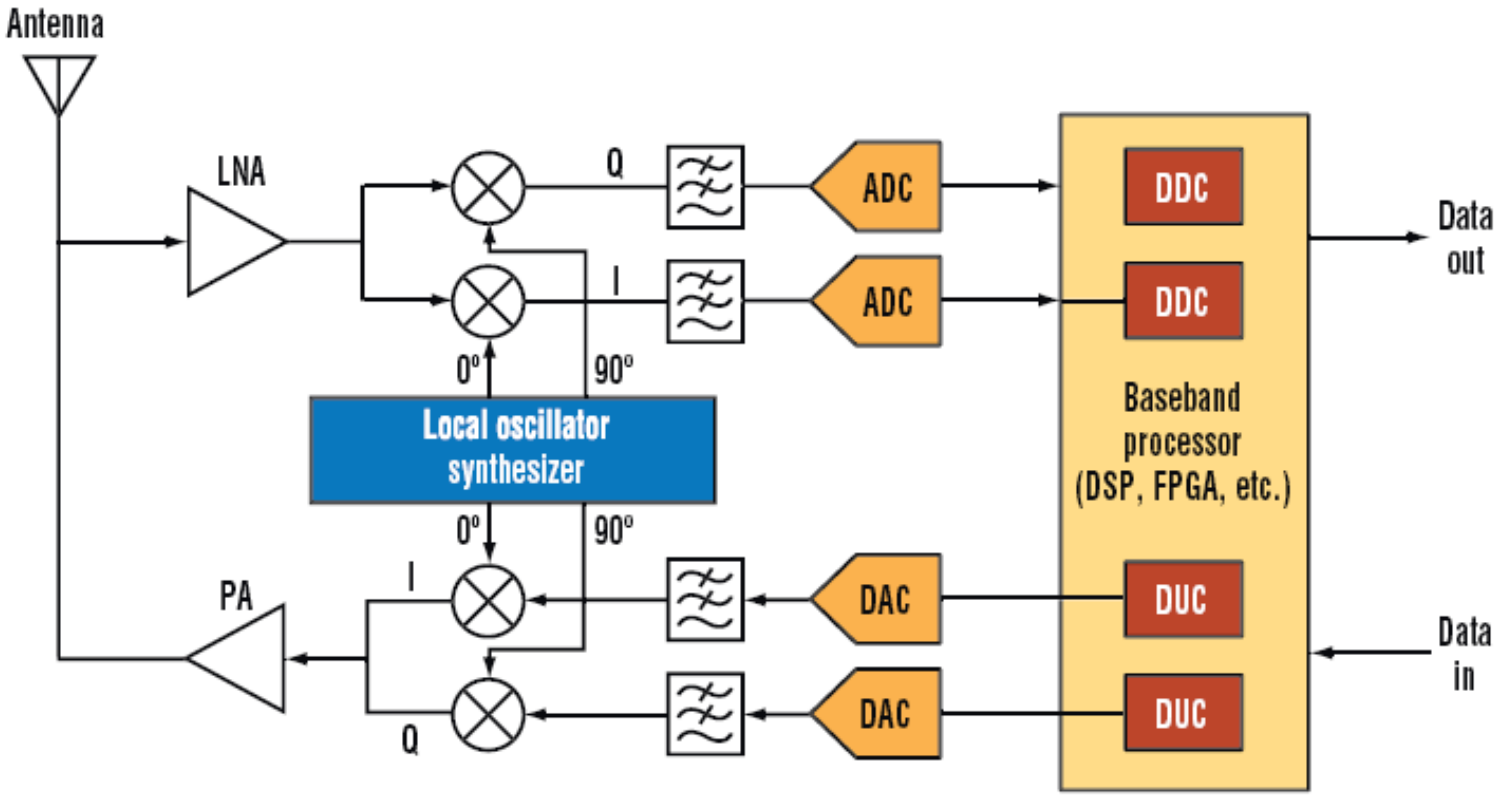
Software Radio



1. In the ideal software-defined radio, the antenna connects directly to the LNA and ADC, or the PA and DAC. The processor handles all radio functions, filtering, up/downconversion, modulation/demodulation, and digital baseband.

Source: http://electronicdesign.com/communications/elusive-software-defined-radio

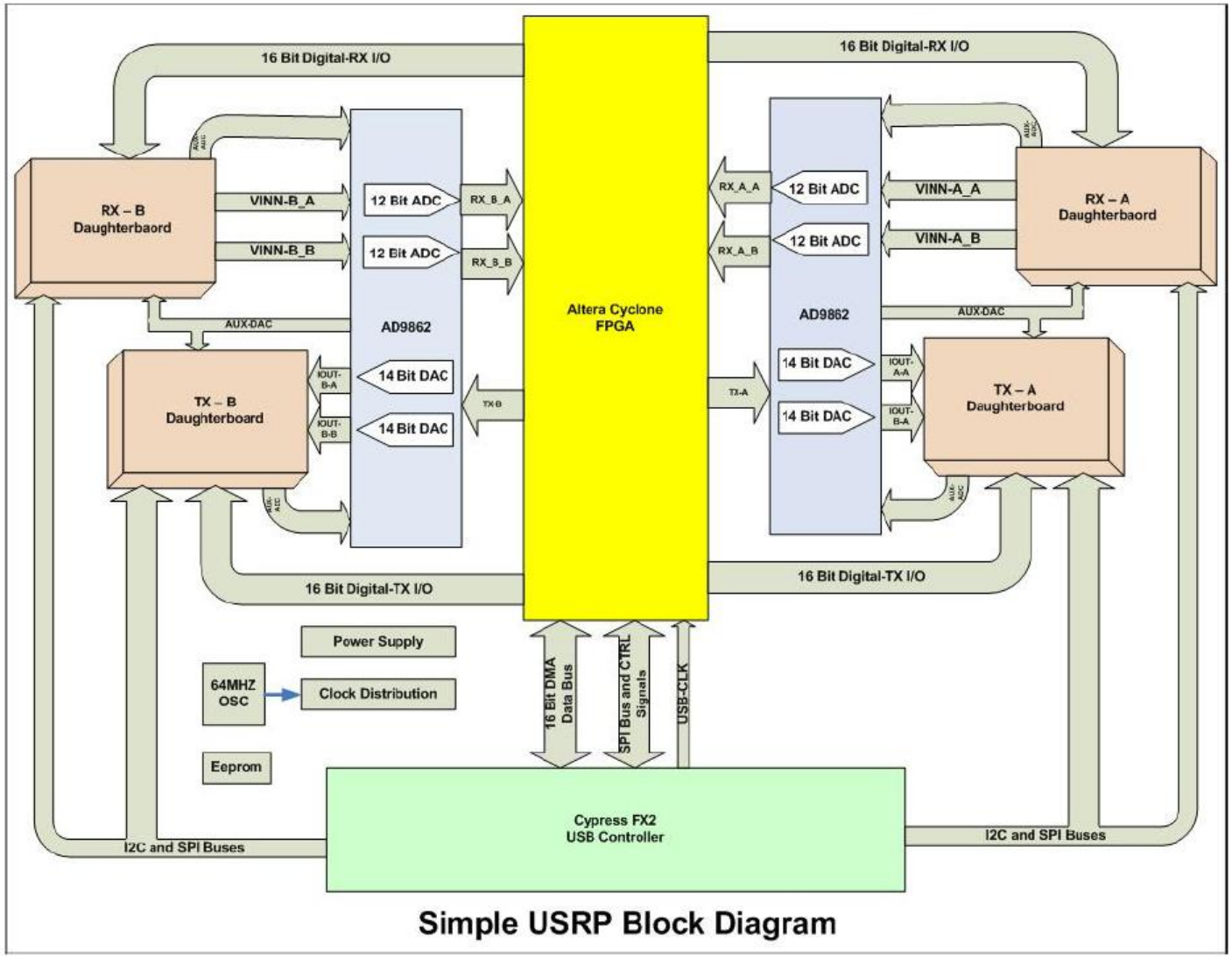
Software Radio



2. Today's SDR architecture features the usual I/Q signal paths. The DDC and DUC functions may be separate ICs, or they're performed in the baseband DSP or FPGA.

Source: http://electronicdesign.com/communications/elusive-software-defined-radio

Software Radio



Source: gnuradio.org/redmine/attachments/128/USRP_Documentation.odt

Iris Overview

- Software Radio
- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- OFDM Specifics

Iris Overview

What is Iris?

Iris Overview

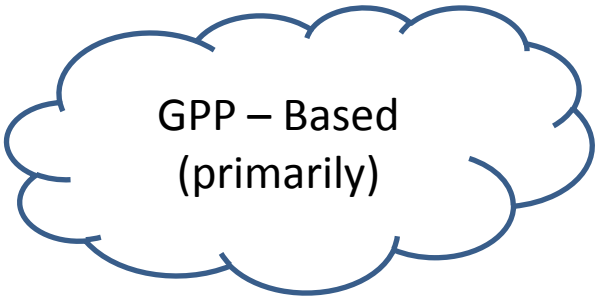
What is Iris?

Iris Overview

What is Iris?

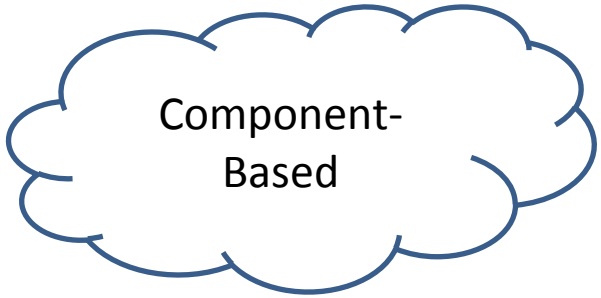
Iris Overview

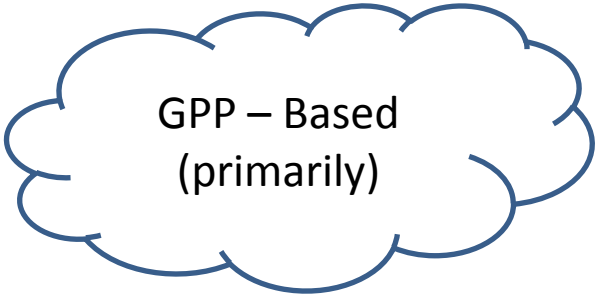
What is Iris?



Iris Overview

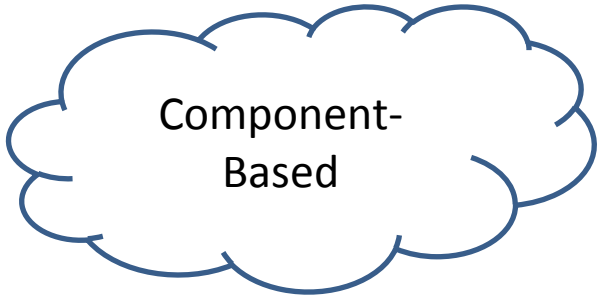
What is Iris?

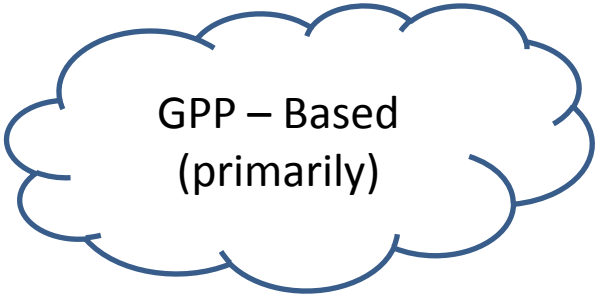


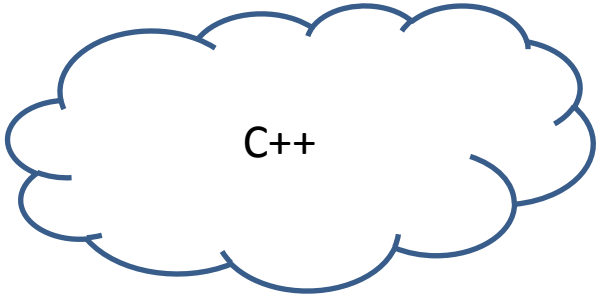


Iris Overview

What is Iris?





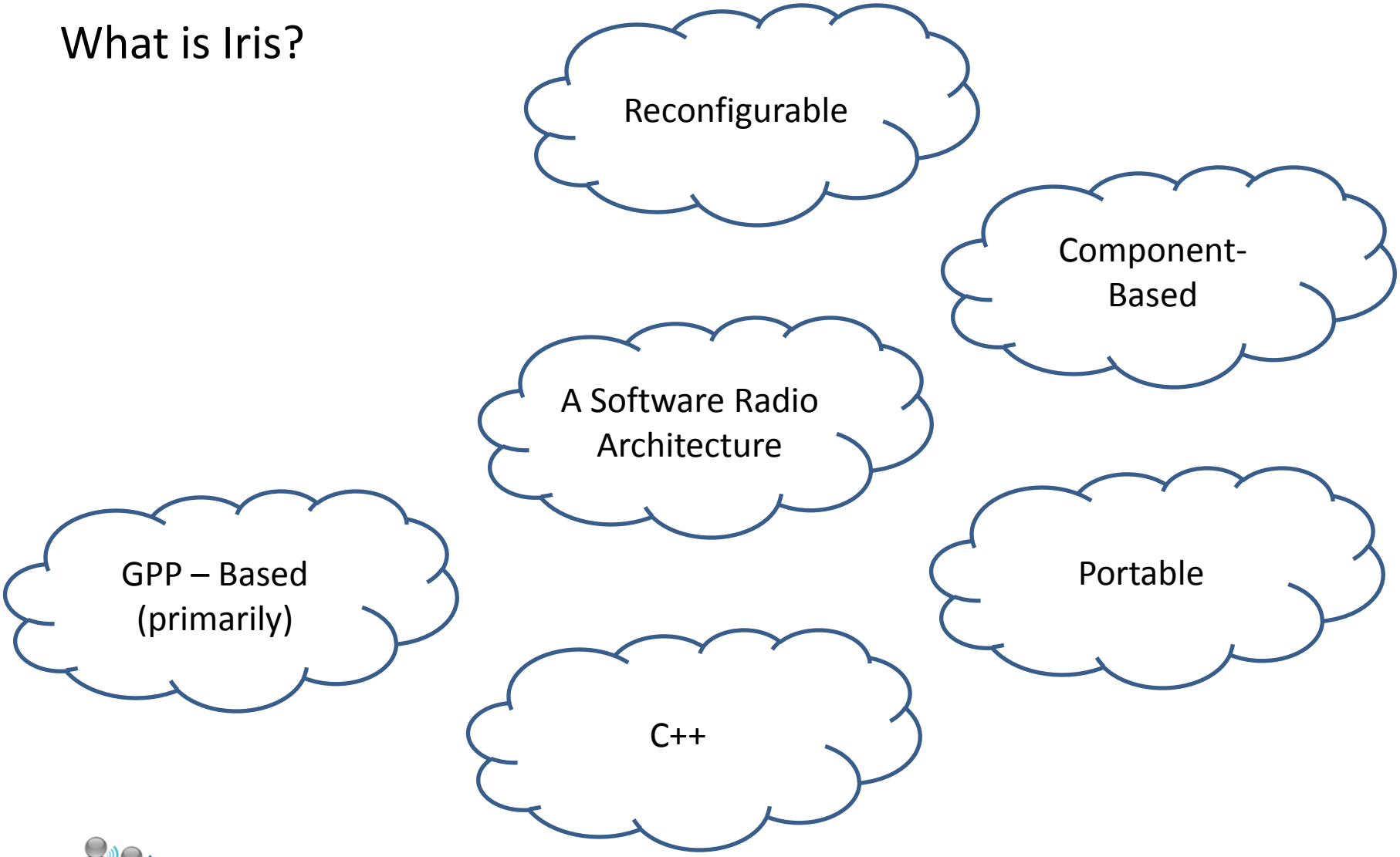


CREW

CTVR | the telecommunications research centre

Iris Overview

What is Iris?

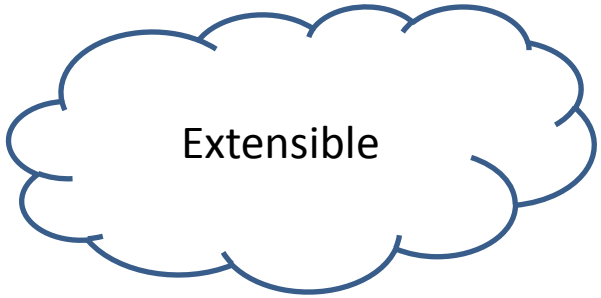


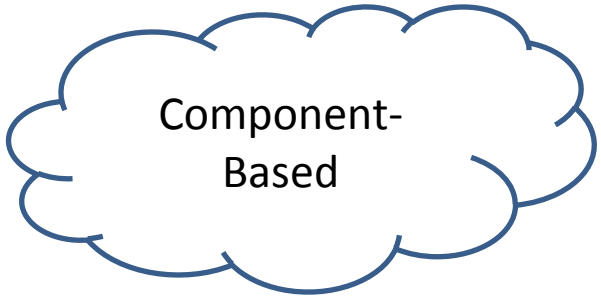
CREW

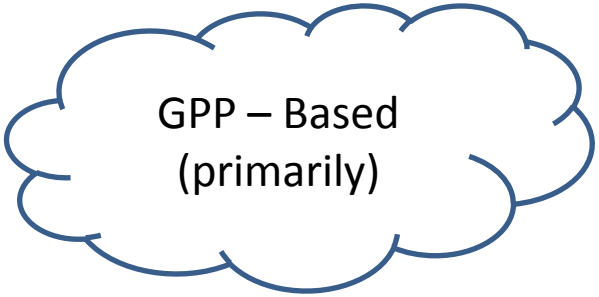
CTVR the telecommunications research centre

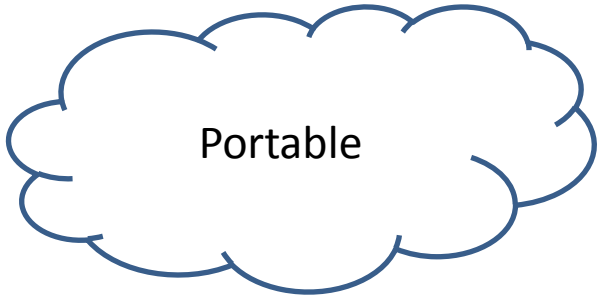
Iris Overview

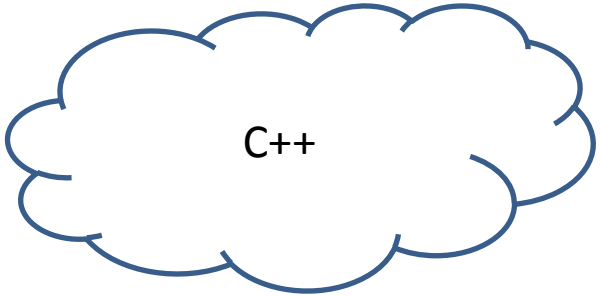
What is Iris?









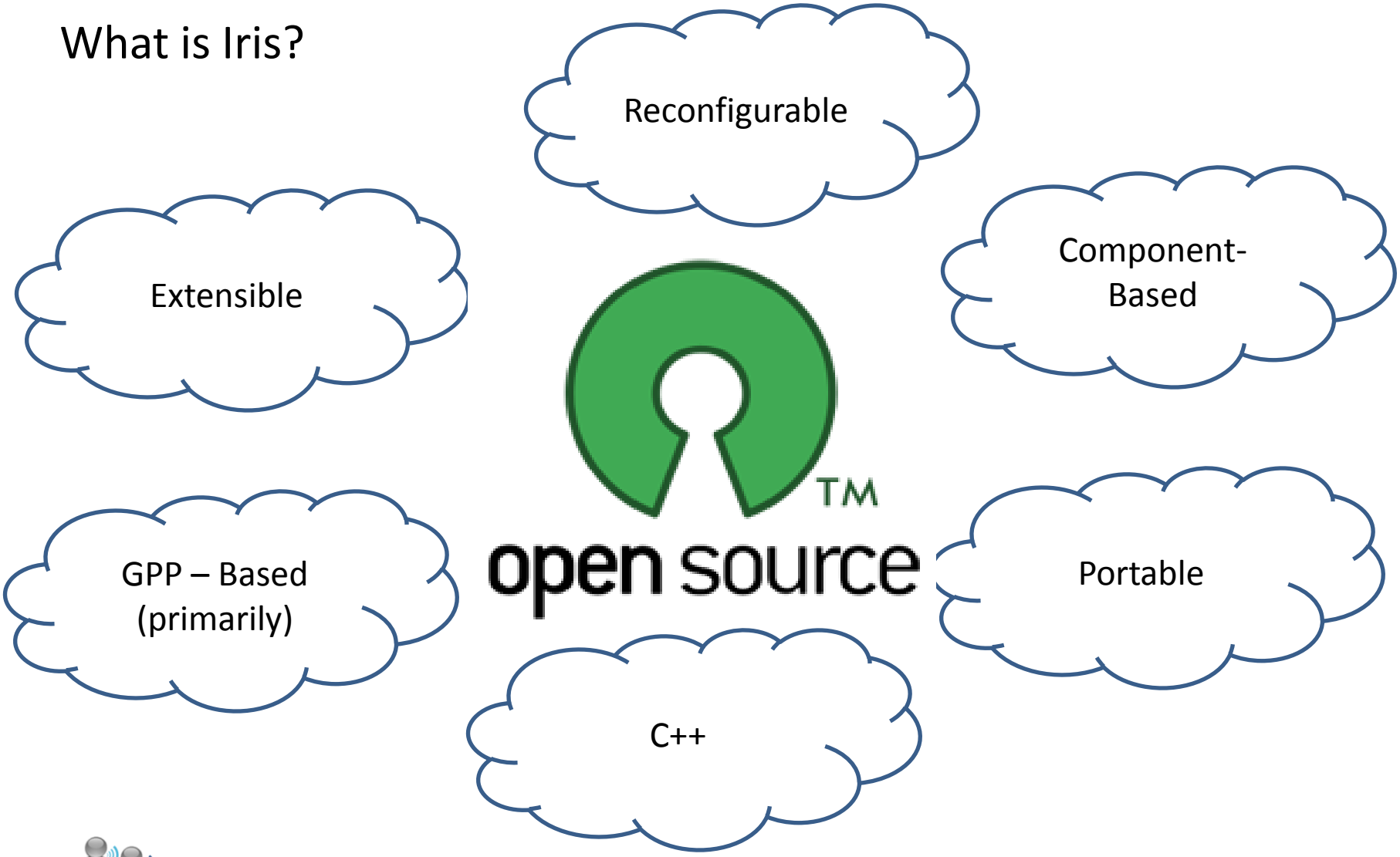


CREW

CTVR the telecommunications research centre

Iris Overview

What is Iris?



CREW

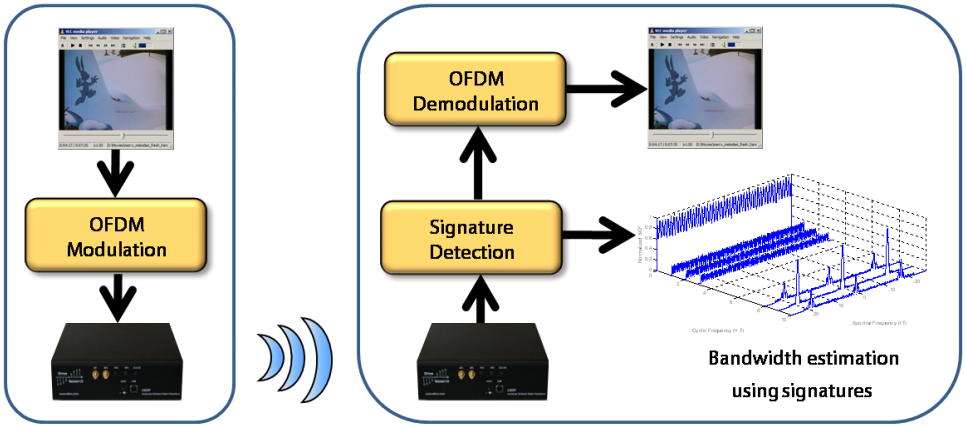
CTVR the telecommunications research centre

Iris Overview

What can I do with Iris?

Iris Overview

What can I do with Iris?

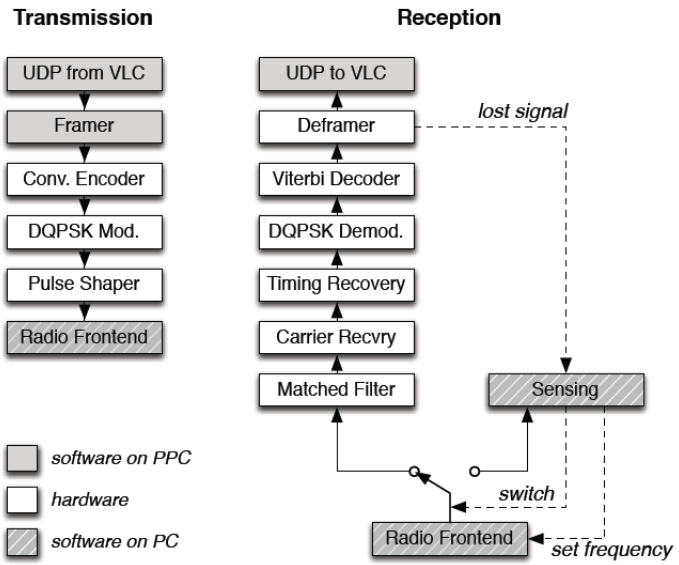


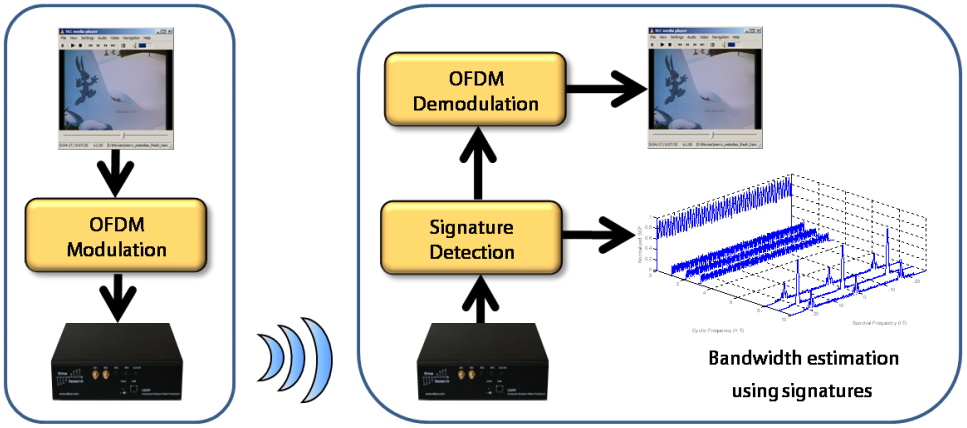
CREW

CTVR the telecommunications research centre

Iris Overview

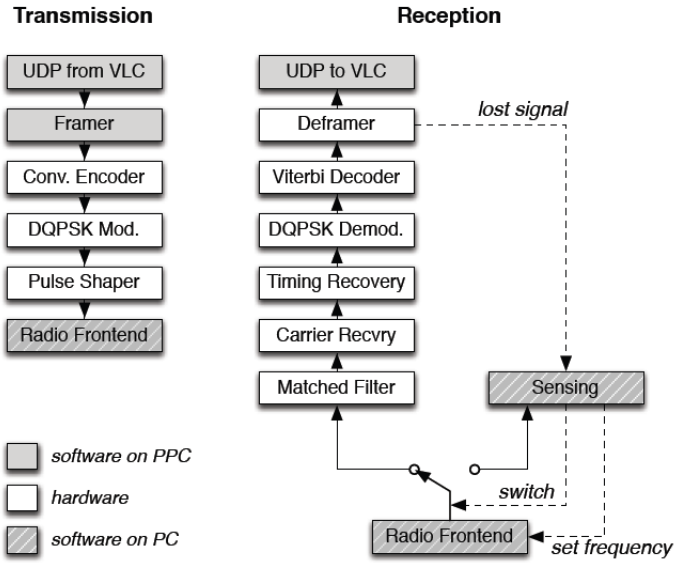
What can I do with Iris?

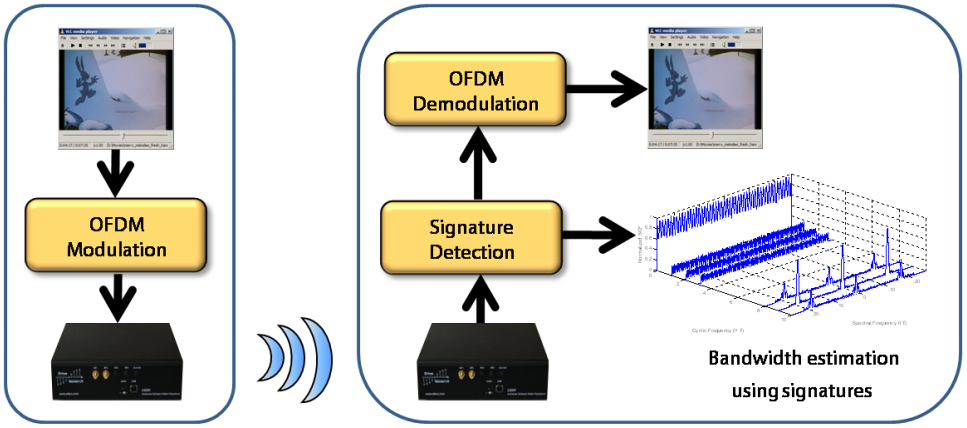


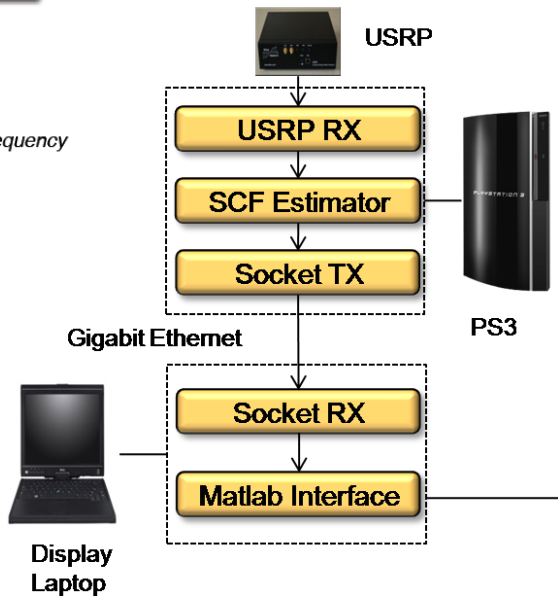


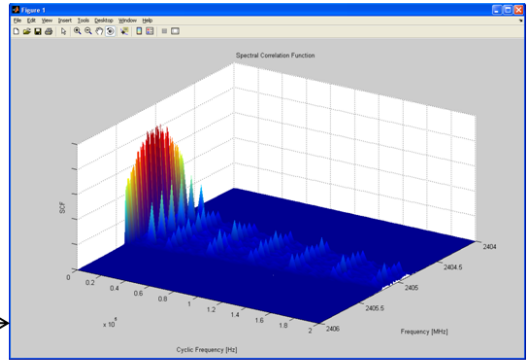
Iris Overview

What can I do with Iris?







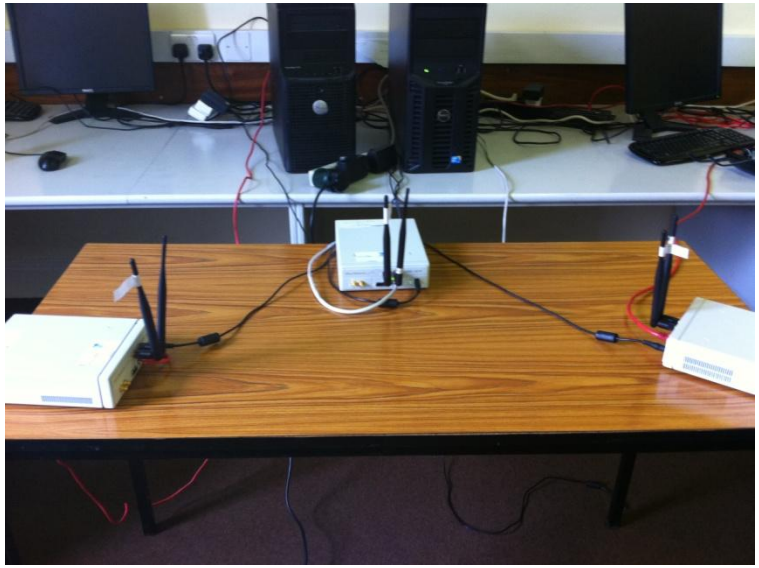


CREW

CTVR the telecommunications research centre

Iris Overview

What can I do with Iris?



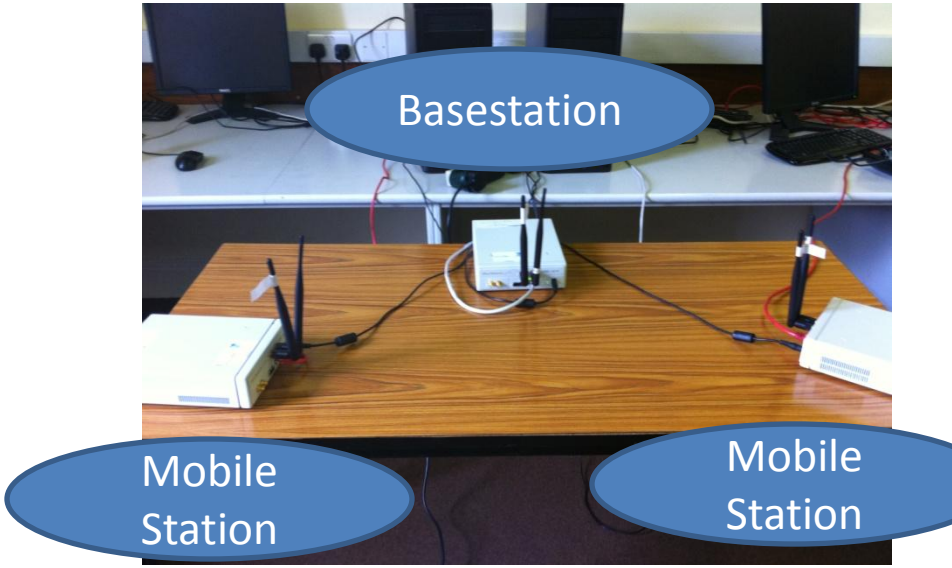
- Jacek Kibilda
- COST Short-Term Scientific Mission
- 2 weeks (no prior knowledge of Iris)
- DSA demo (primary user avoidance)

CREW

CTVR the telecommunications research centre

Iris Overview

What can I do with Iris?



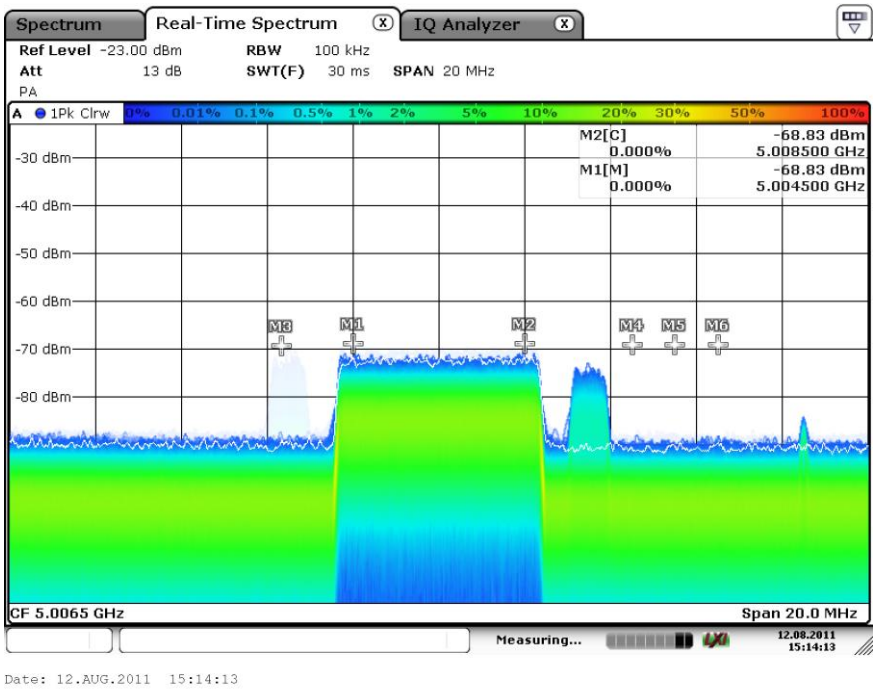
- Jacek Kibilda
- COST Short-Term Scientific Mission
- 2 weeks (no prior knowledge of Iris)
- DSA demo (primary user avoidance)

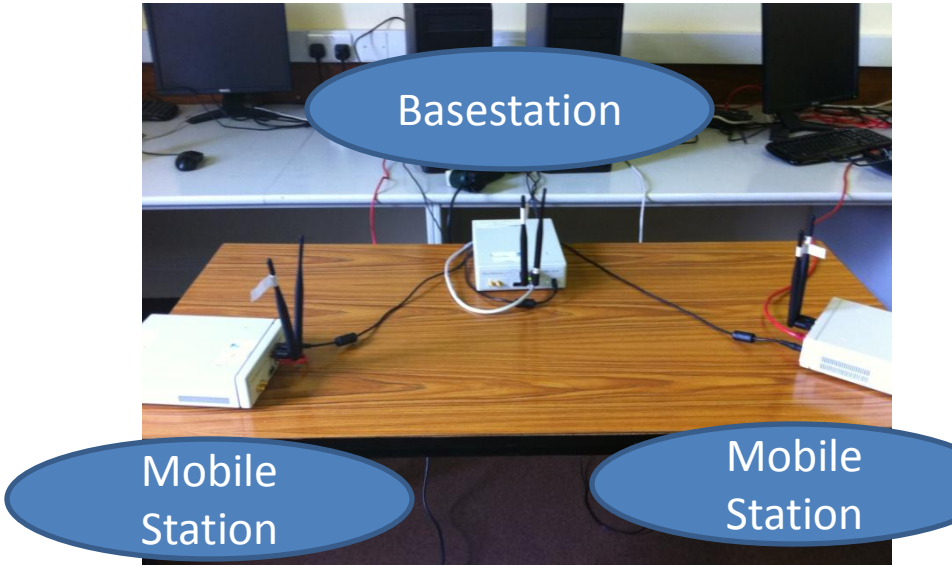
CREW

CTVR the telecommunications research centre

Iris Overview

What can I do with Iris?





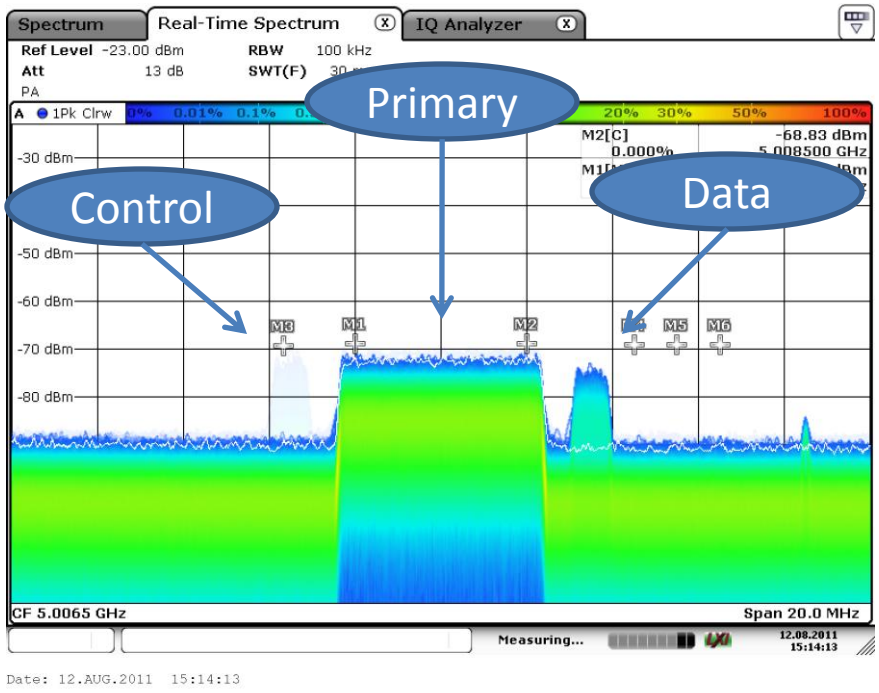
- Jacek Kibilda
- COST Short-Term Scientific Mission
- 2 weeks (no prior knowledge of Iris)
- DSA demo (primary user avoidance)

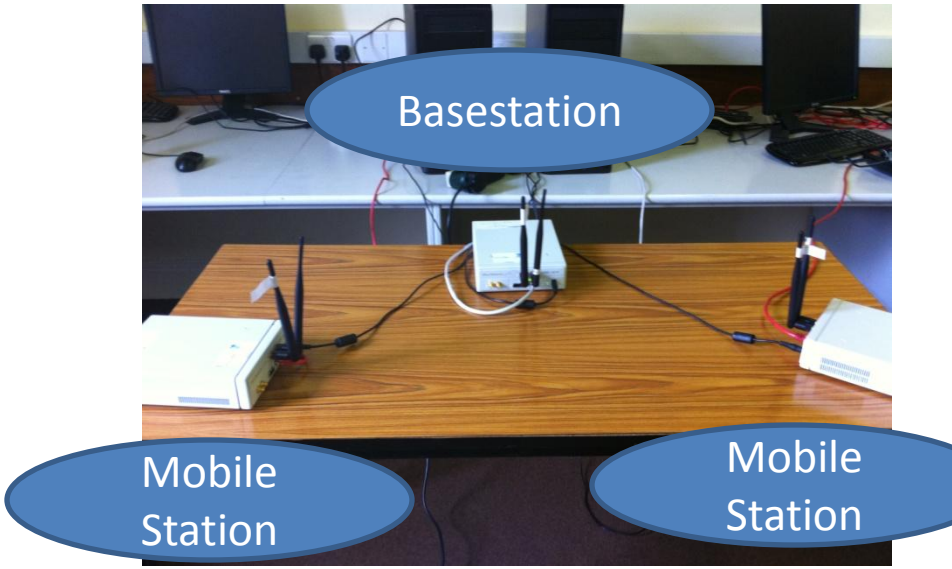
CREW

CTVR the telecommunications research centre

Iris Overview

What can I do with Iris?





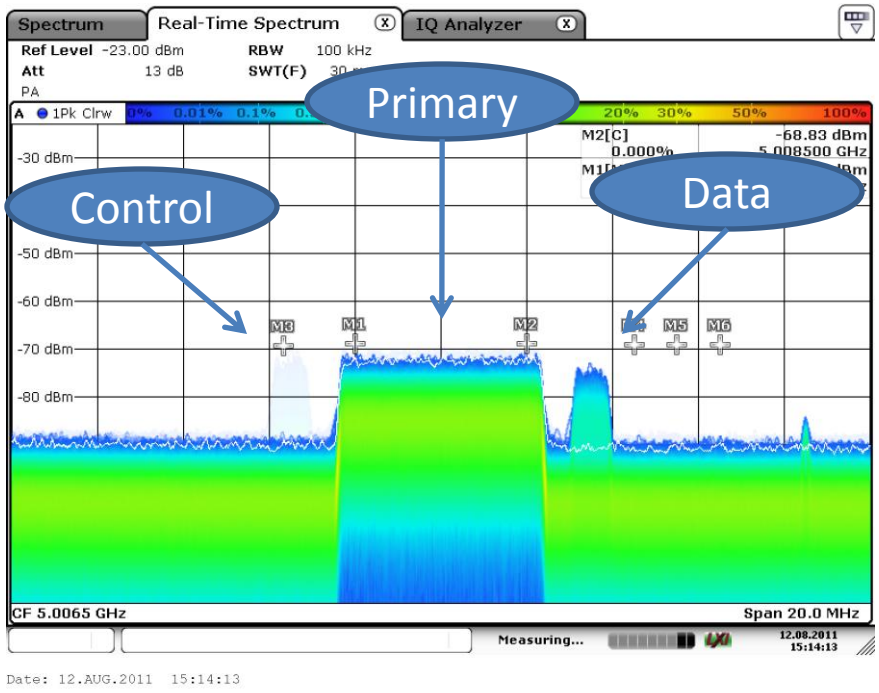
- Jacek Kibilda
- COST Short-Term Scientific Mission
- 2 weeks (no prior knowledge of Iris)
- DSA demo (primary user avoidance)

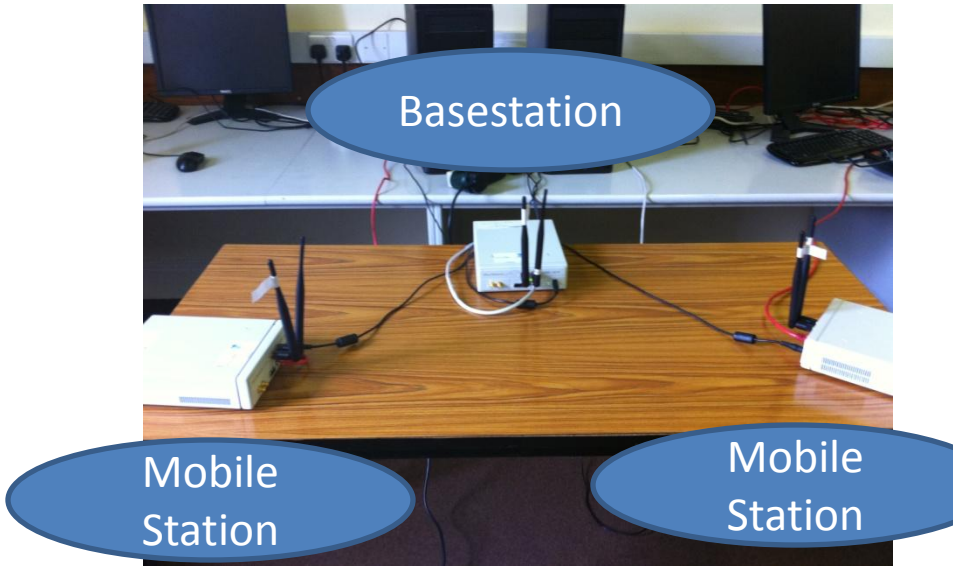
CREW

CTVR the telecommunications research centre

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)

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

CREW

CTVR the telecommunications research centre

Iris Architecture

- Software Radio
- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- OFDM Specifics

The Basics...

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

The Basics...

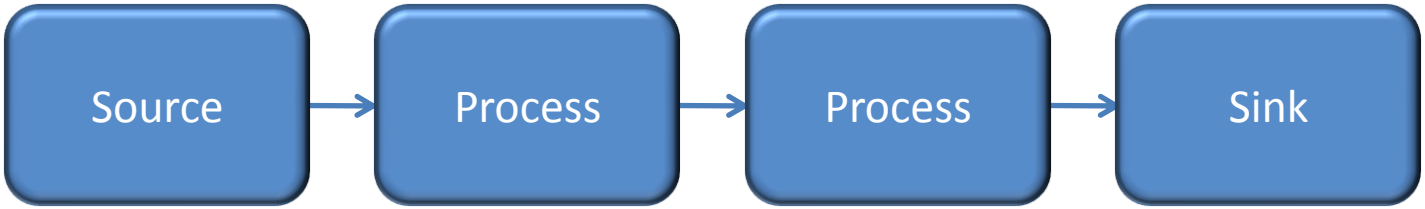
- A GPP-based software radio architecture
  - Fundamental block is the component

- 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

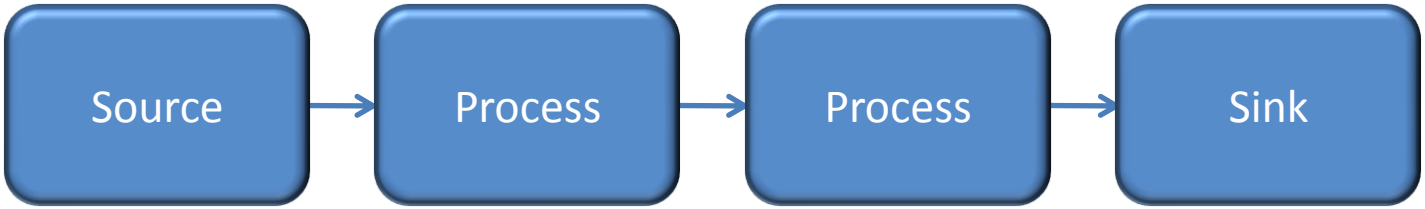
- Most basic configuration :
  - A source component
  - A sink component
  - Some processing components



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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

CTVR the telecommunications research centre

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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

CTVR the telecommunications research centre

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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

CTVR the telecommunications research centre

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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

CTVR the telecommunications research centre

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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

CTVR the telecommunications research centre

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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

CTVR the telecommunications research centre

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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

CTVR the telecommunications research centre

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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

CTVR the telecommunications research centre

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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

CTVR the telecommunications research centre

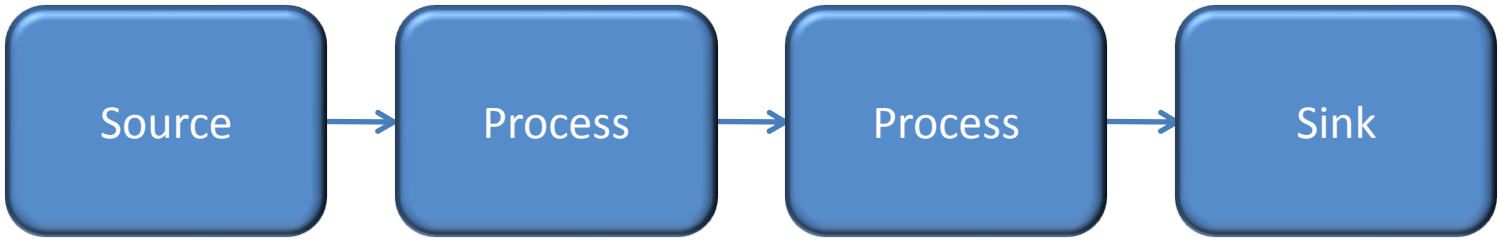
Iris Architecture - The Basics


```
<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="ofdmdemod1" class="ofdmdemodulator">
            <port name="input1" class="input"/>
            <port name="output1" class="output"/>
        </component>

        <component name="filerawwriter1" class="filerawwriter">
            <parameter name="filename" value="out.txt"/>
            <port name="input1" class="input"/>
        </component>

    </engine>

    <link source="filerawreader1.output1" sink="ofdmmod1.input1" />
    <link source="ofdmmod1.output1" sink="ofdmdemod1.input1" />
    <link source="ofdmdemod1.output1" sink="filerawwriter1.input1" />

</softwareradio>
```

CREW

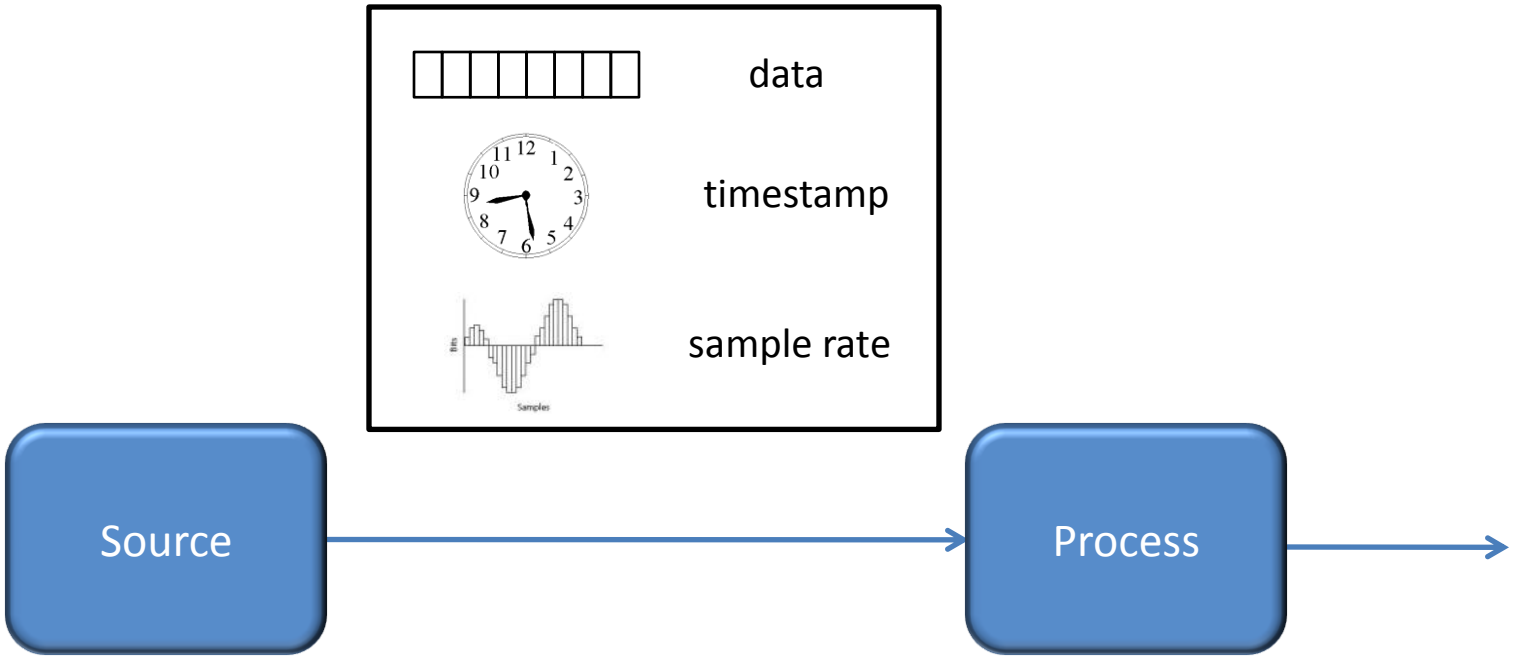
CTVR the telecommunications research centre

Iris Architecture - The Basics



Iris Architecture - The Basics

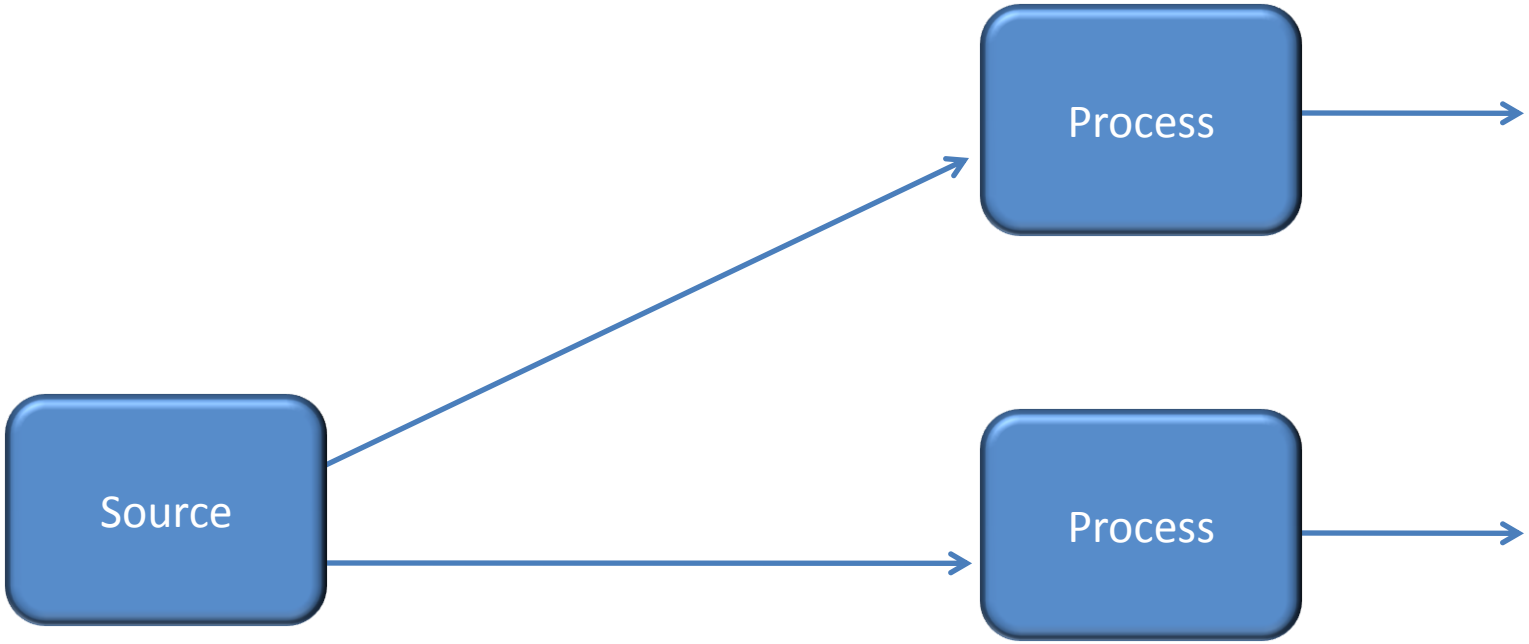
Iris Architecture - The Basics



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

Iris Architecture - The Basics

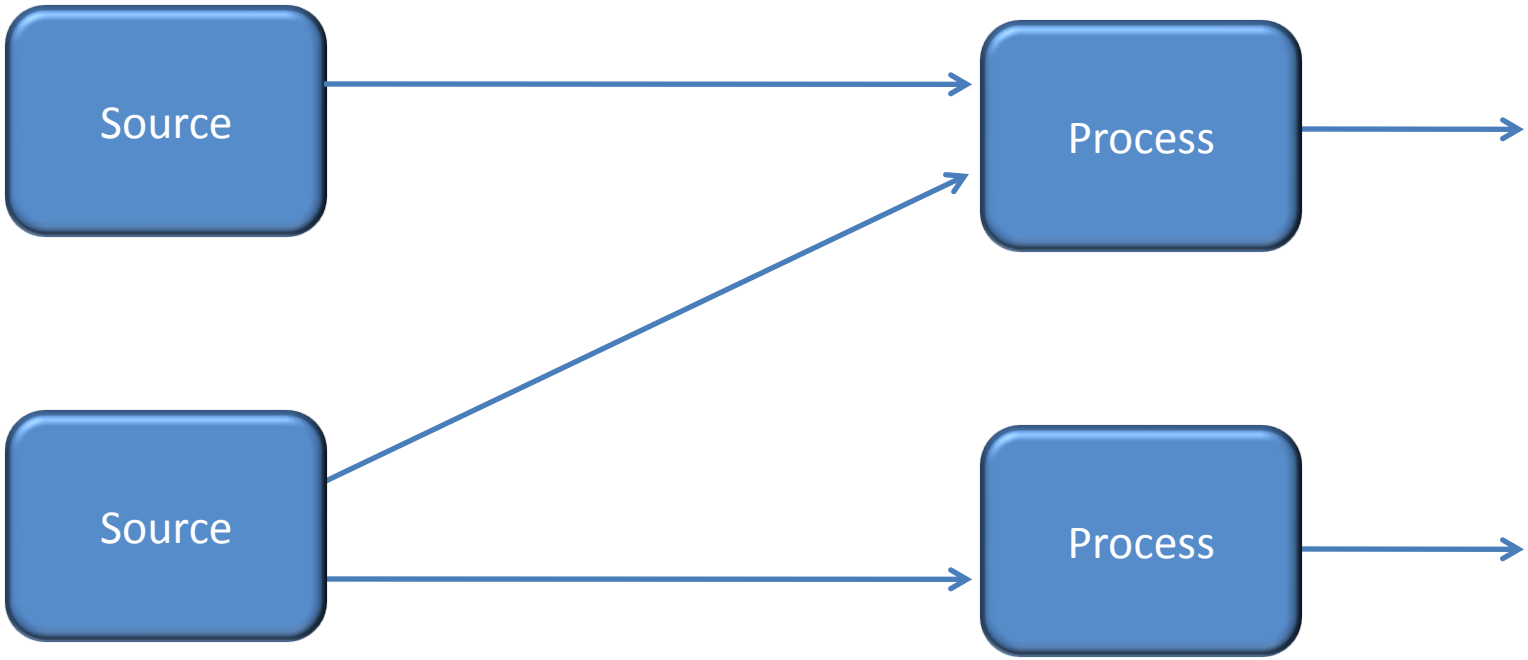
Iris Architecture - The Basics



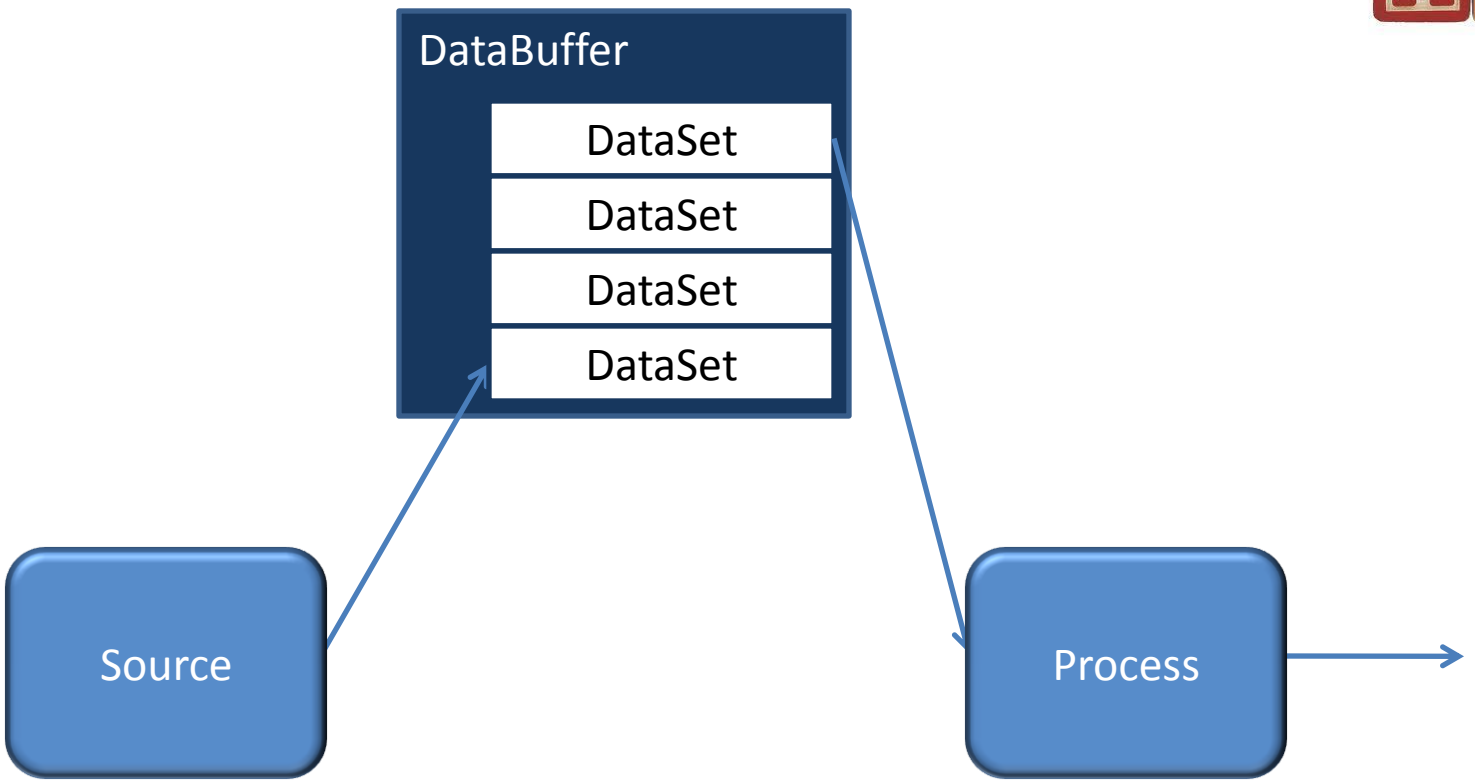
CREW

CTVR the telecommunications research centre

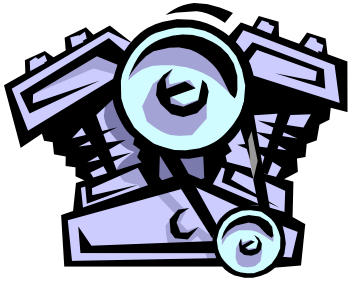
Iris Architecture - The Basics



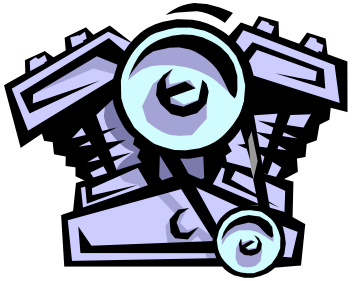
Iris Architecture - The Basics



Engines

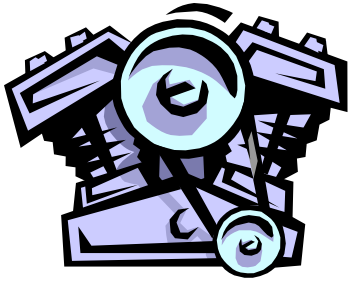


Engines



- 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

Engines

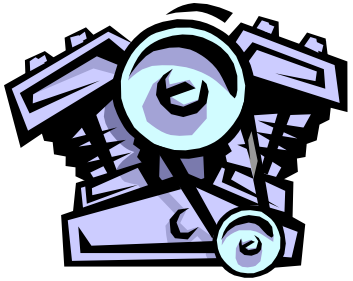


- An *engine*
  - The environment within which one more components operates
  - Defines its own data-flow mechanism
  - Defines its own reconfiguration mechanisms
  - Runs one or more of its own threads
  - Provides a clean interface for the Iris system

Executes a section of the flow graph

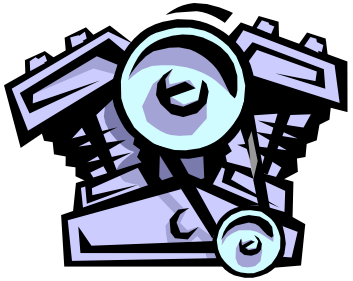
Completely up to the engine how that's done

Iris Architecture - Engines



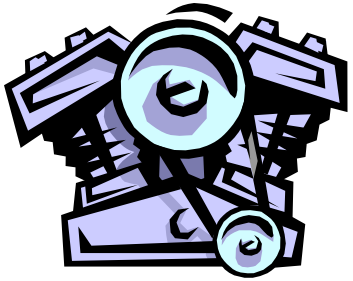
- Two engine types:
  - PHY Engine
  - Stack Engine

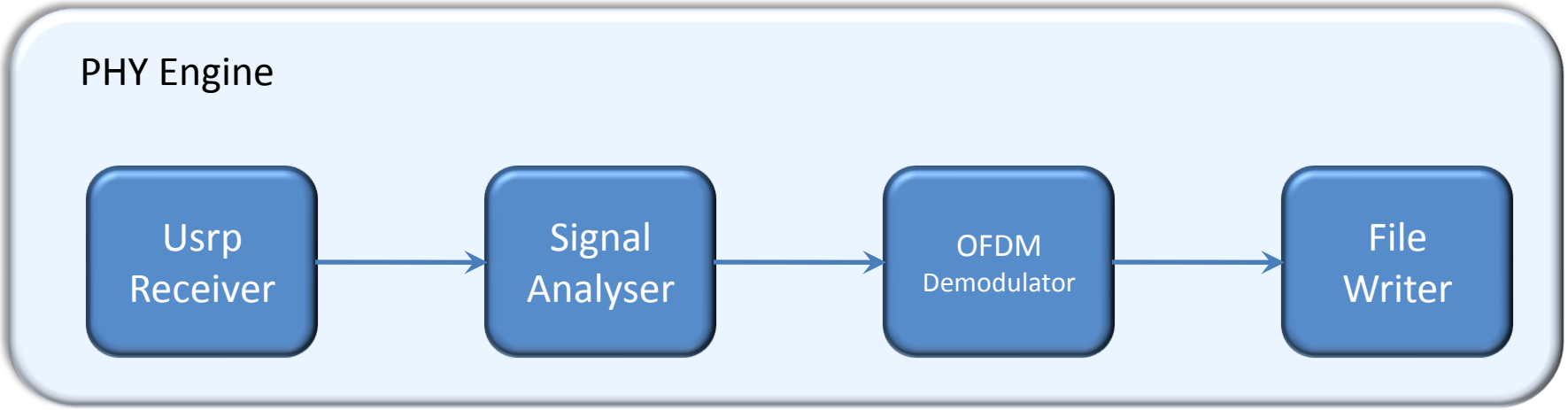
Iris Architecture - Engines



- 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 blocksizes

Iris Architecture - Engines

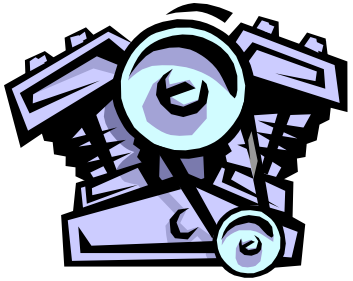




CREW

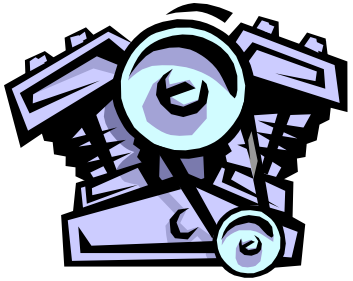
CTVR the telecommunications research centre

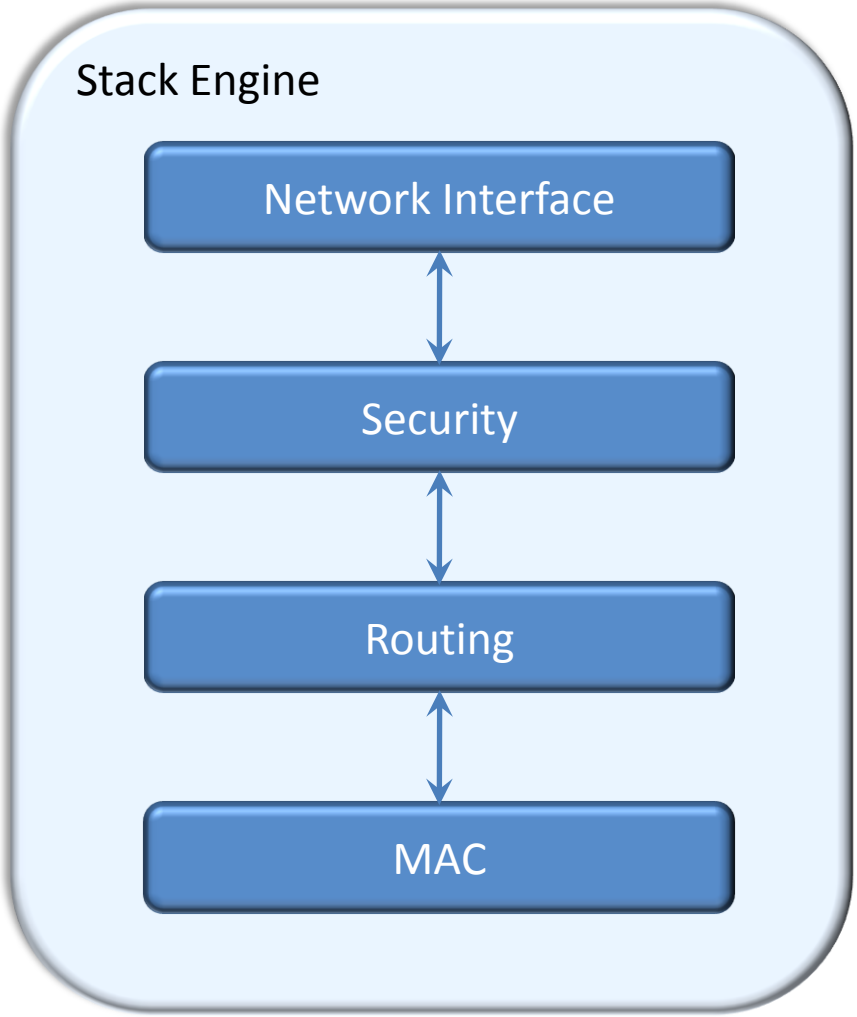
Iris Architecture - Engines



- 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



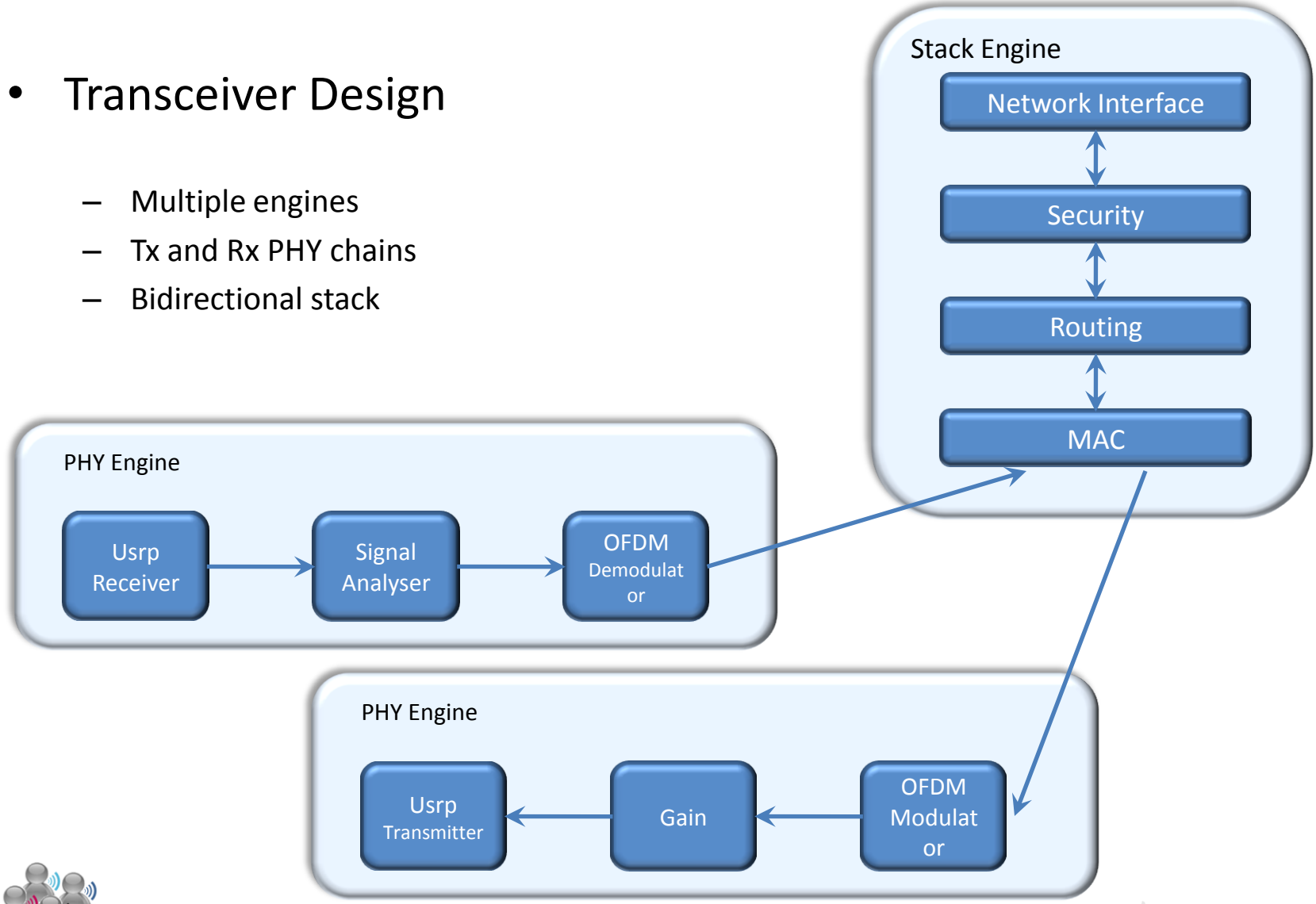


CREW

CTVR the telecommunications research centre

Iris Architecture - Engines

- Transceiver Design
  - Multiple engines
  - Tx and Rx PHY chains
  - Bidirectional stack



Iris Architecture - PHY Components

A PHY Component

A PHY Component

ExampleComponent

```
ExampleComponent(std::string name);

void calculateOutputTypes(
  std::map<std::string, int>& inputTypes,
  std::map<std::string, int>& outputTypes);

void registerPorts();

void initialize();

void process();
```

Iris Architecture - PHY Components

```
ExampleComponent::ExampleComponent(std::string name)
  : PhyComponent(name,                          // component name
                 "example",                     // component type
                 "An example phy component",    // description
                 "Paul Sutton",                 // author
                 "0.1")                         // version
{
  registerParameter(
      "exampleparameter",                       // name
      "An example parameter",                   // description
      "0",                                      // default value
      true,                                     // dynamic?
      example_x,                                // parameter
      Interval<uint32_t>(0,5));                 // allowed values
}
```

CREW

CTVR | the telecommunications research centre

Iris Architecture - PHY Components

exampleparameter

```
ExampleComponent::ExampleComponent(std::string name)
  : PhyComponent(name,                          // component name
                 "example",                     // component type
                 "An example phy component",    // description
                 "Paul Sutton",                 // author
                 "0.1")                         // version
{
  registerParameter(
      "exampleparameter",                       // name
      "An example parameter",                   // description
      "0",                                      // default value
      true,                                     // dynamic?
      example_x,                                // parameter
      Interval<uint32_t>(0,5));                 // allowed values
}
```

CREW

CTVR the telecommunications research centre

Iris Architecture - PHY Components

exampleparameter

```
void ExampleComponent::registerPorts()
{
  registerInputPort("input1", TypeInfo< uint32_t>::identifier);
  registerOutputPort("output1", TypeInfo< uint32_t >::identifier);
}

void ExampleComponent::calculateOutputTypes(
    std::map<std::string,int>& inputTypes,
    std::map<std::string,int>& outputTypes)
{
  //One output type - always uint32_t
  outputTypes["output1"] = TypeInfo< uint32_t >::identifier;
}
```

CREW

CTVR the telecommunications research centre

Iris Architecture - PHY Components

exampleparameter

input1

output1

```
void ExampleComponent::registerPorts()
{
  registerInputPort("input1", TypeInfo< uint32_t>::identifier);
  registerOutputPort("output1", TypeInfo< uint32_t >::identifier);
}

void ExampleComponent::calculateOutputTypes(
    std::map<std::string,int>& inputTypes,
    std::map<std::string,int>& outputTypes)
{
  //One output type - always uint32_t
  outputTypes["output1"] = TypeInfo< uint32_t >::identifier;
}
```

CREW

CTVR the telecommunications research centre

Iris Architecture - PHY Components

exampleparameter

input1

output1

```
void ExampleComponent::process()
{
  DataSet<uint32_t>* readDataSet = NULL;
  getInputDataSet("input1", readDataSet);
  std::size_t size = readDataSet->data.size();

  DataSet<uint32_t>* writeDataSet = NULL;
  getOutputDataSet("output1", writeDataSet, size);

  copy(readDataSet->data.begin(), readDataSet->data.end(),
       writeDataSet->data.begin());

  writeDataSet->timeStamp = readDataSet->timeStamp;
  writeDataSet->sampleRate = readDataSet->sampleRate;

  releaseInputDataSet("input1", readDataSet);
  releaseOutputDataSet("output1", writeDataSet);
}
```

CREW

CTVR the telecommunications research centre

Iris Architecture - Stack Components

A Stack Component

CREW

CTVR the telecommunications research centre

A Stack Component

```
ExampleComponent(std::string name);

void initialize();

void start();

void stop();

void processMessageFromAbove(boost::shared_ptr<StackDataSet> set);

void processMessageFromBelow(boost::shared_ptr<StackDataSet> set);
```

Iris Architecture - Stack Components

```
ExampleComponent::ExampleComponent(std::string name)
  : StackComponent(name,                              // Component name
                   "example",                         // Component type
                   "An example stack component",      // Description
                   "Paul Sutton",                     // Author
                   "0.1")                             // Version
{
  registerParameter("exampleparameter",
                    "An example parameter",
                    "0",
                    true,
                    example_x,
                    Interval<uint32_t>(0,5));
}
```

CREW

CTVR the telecommunications research centre

Iris Architecture - Stack Components

```
ExampleComponent::ExampleComponent(std::string name)
  : StackComponent(name,                              // Component name
                   "example",                         // Component type
                   "An example stack component",      // Description
                   "Paul Sutton",                     // Author
                   "0.1")                             // Version
{
  registerParameter("exampleparameter",
                    "An example parameter",
                    "0",
                    true,
                    example_x,
                    Interval<uint32_t>(0,5));
}
```

exampleparameter

Iris Architecture - Stack Components

```
void ExampleComponent::processMessageFromAbove(
        boost::shared_ptr<StackDataSet> set)
{
  sendDownwards(set); // Simply send the message on
}

void ExampleComponent::processMessageFromBelow(
        boost::shared_ptr<StackDataSet> set)
{
  sendUpwards(set);  // Simply send the message on
}
```

exampleparameter

CREW

CTVR the telecommunications research centre

Iris Architecture - Stack Components

```
void ExampleComponent::processMessageFromAbove(
        boost::shared_ptr<StackDataSet> set)
{
  sendDownwards(set); // Simply send the message on
}

void ExampleComponent::processMessageFromBelow(
        boost::shared_ptr<StackDataSet> set)
{
  sendUpwards(set);  // Simply send the message on
}
```

exampleparameter

CREW

CTVR the telecommunications research centre

Getting Started

- Software Radio
- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- OFDM Specifics

Getting Started

- Code: https://github.com/softwareradiosystems
- Redmine: http://www.softwareradiosystems.com/redmine/projects/iris
- Mailing Lists: http://www.softwareradiosystems.com/mailman/listinfo

Getting Started

- Demo 1
  - Checkout core from github
  - Explore directories
  - CMake configuration & dependencies
  - Build, test & install

  - Checkout modules (switch to demo branch)
  - Explore directories
  - CMake configuration & dependencies
  - Build, test, benchmark & install

Getting Started

- Demo 2
  - Remote OFDM demo
  - Command line options
  - Parametric reconfiguration via XML
  - Display on analyser

Controllers

- Software Radio
- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- OFDM Specifics

Controllers

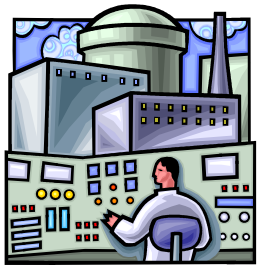
- So far...
  - We can create a radio
  - and reconfigure it manually

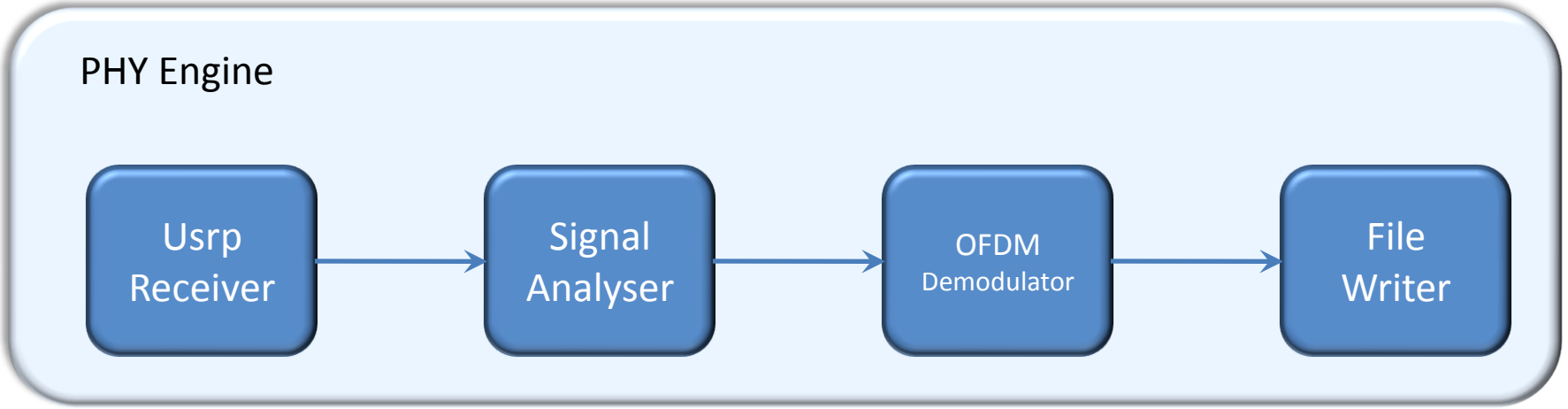
Controllers

- So far...
  - We can create a radio
  - and reconfigure it manually

- How to reconfigure dynamically?

Controllers





Controllers

- Parameters

Parametric reconfiguration
e.g. Number of subcarriers

PHY Engine

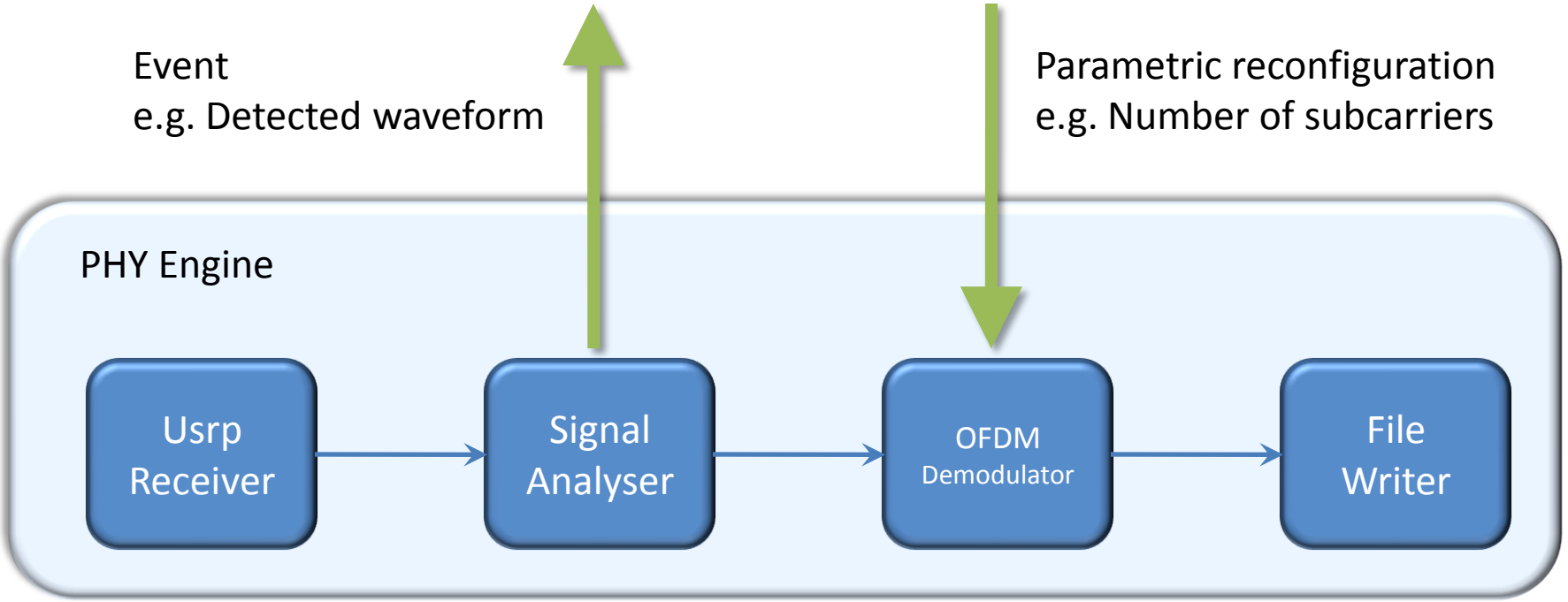
Usrp Receiver → Signal Analyser → OFDM Demodulator → File Writer

CREW

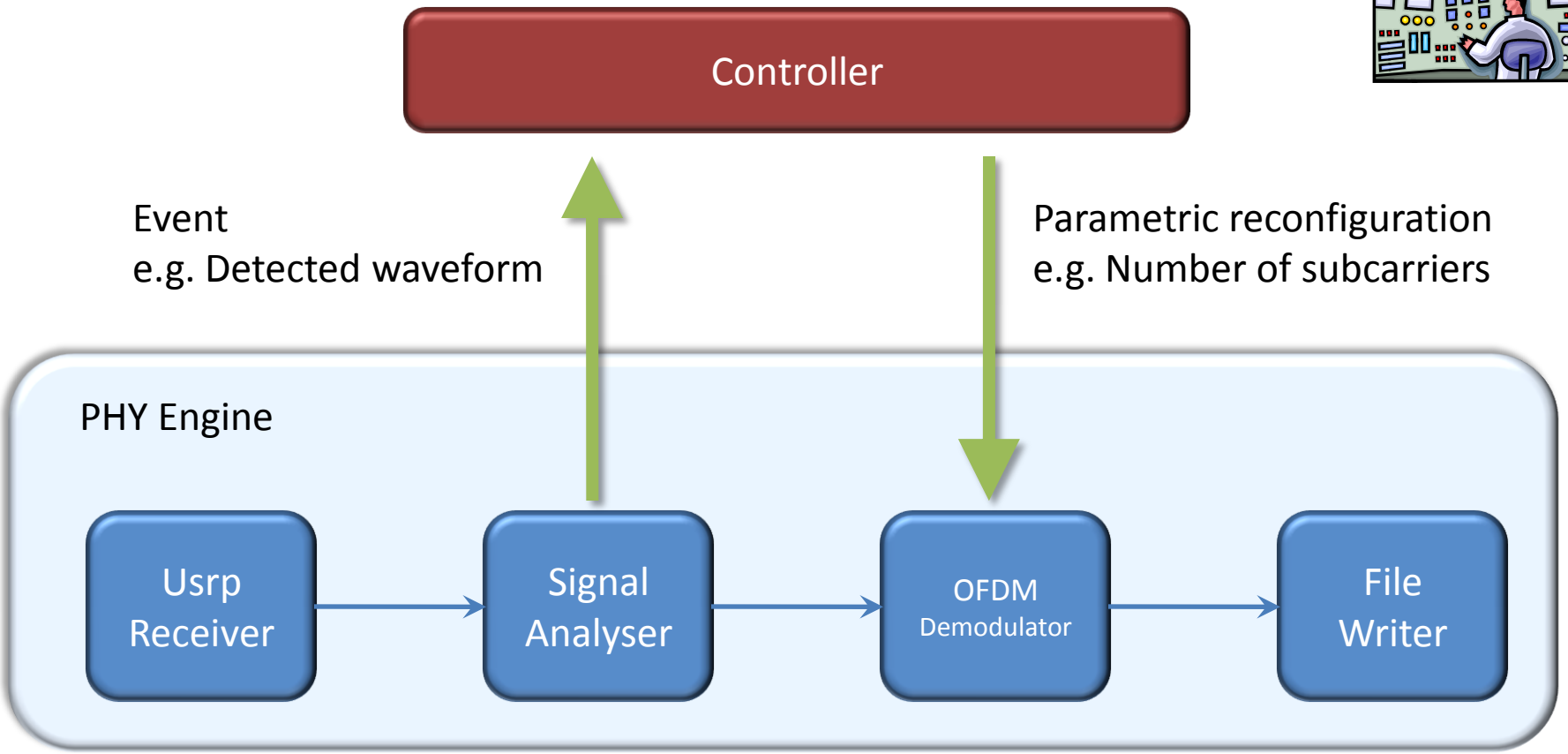
CTVR the telecommunications research centre

Controllers

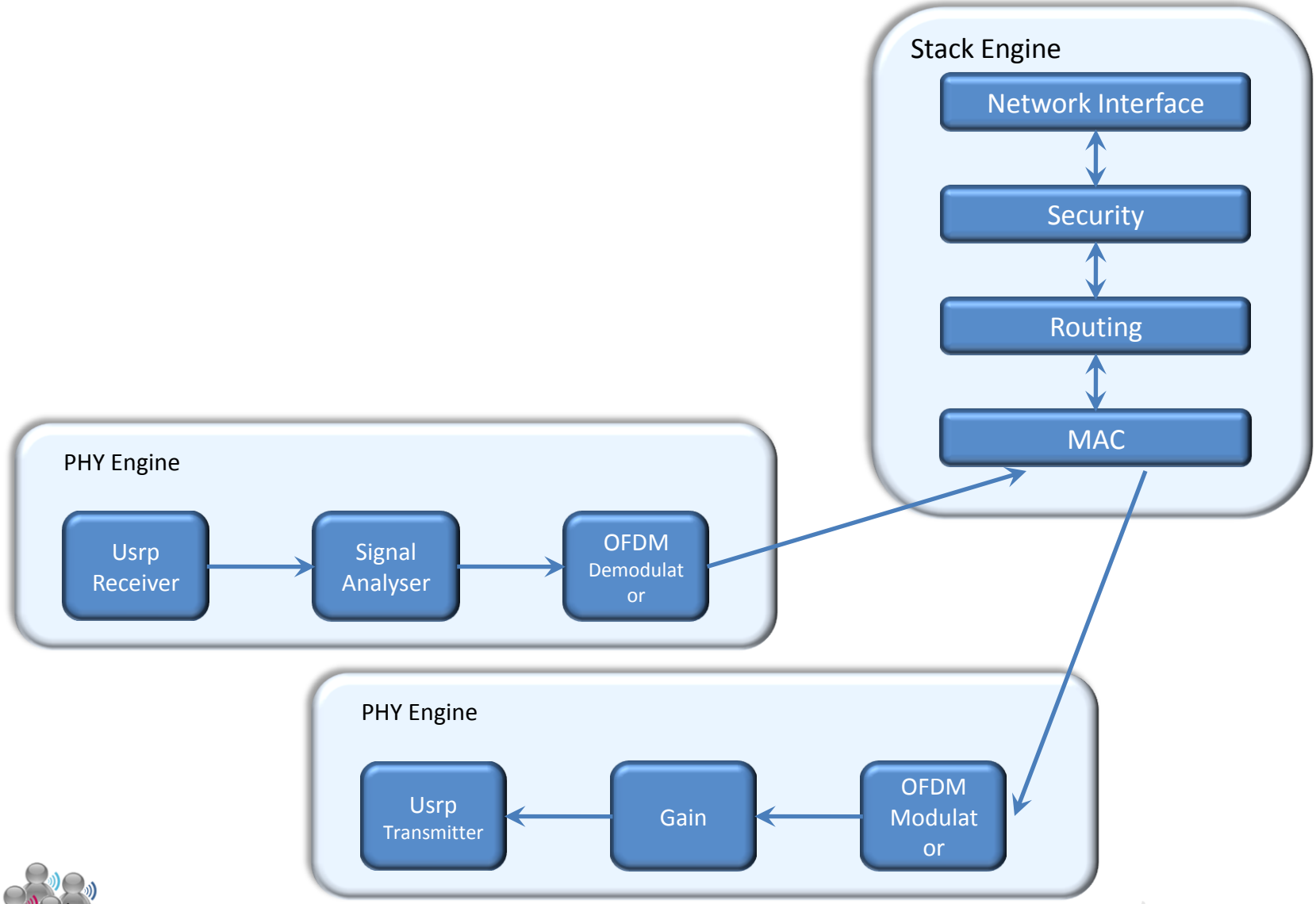
- Events



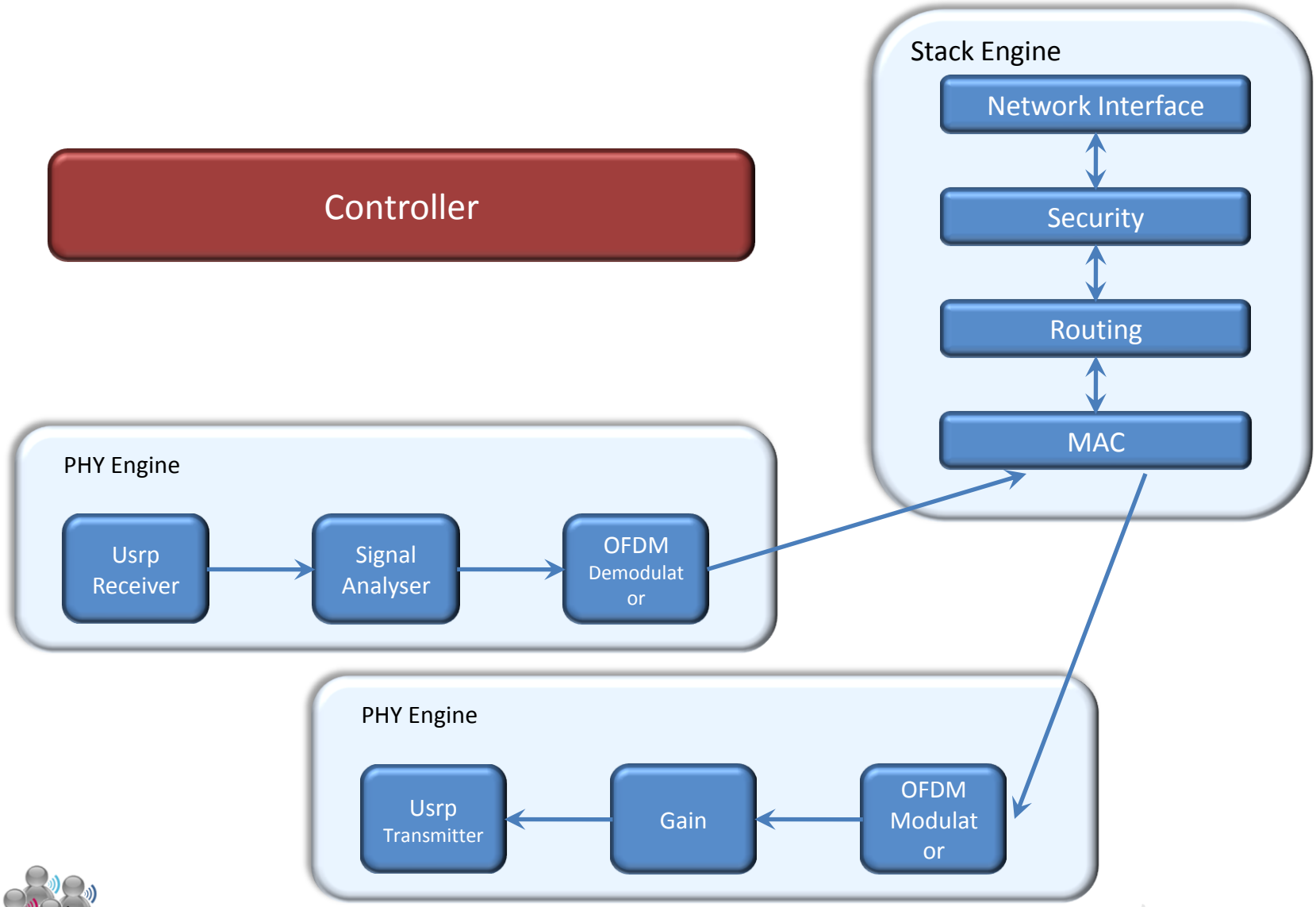
Controllers



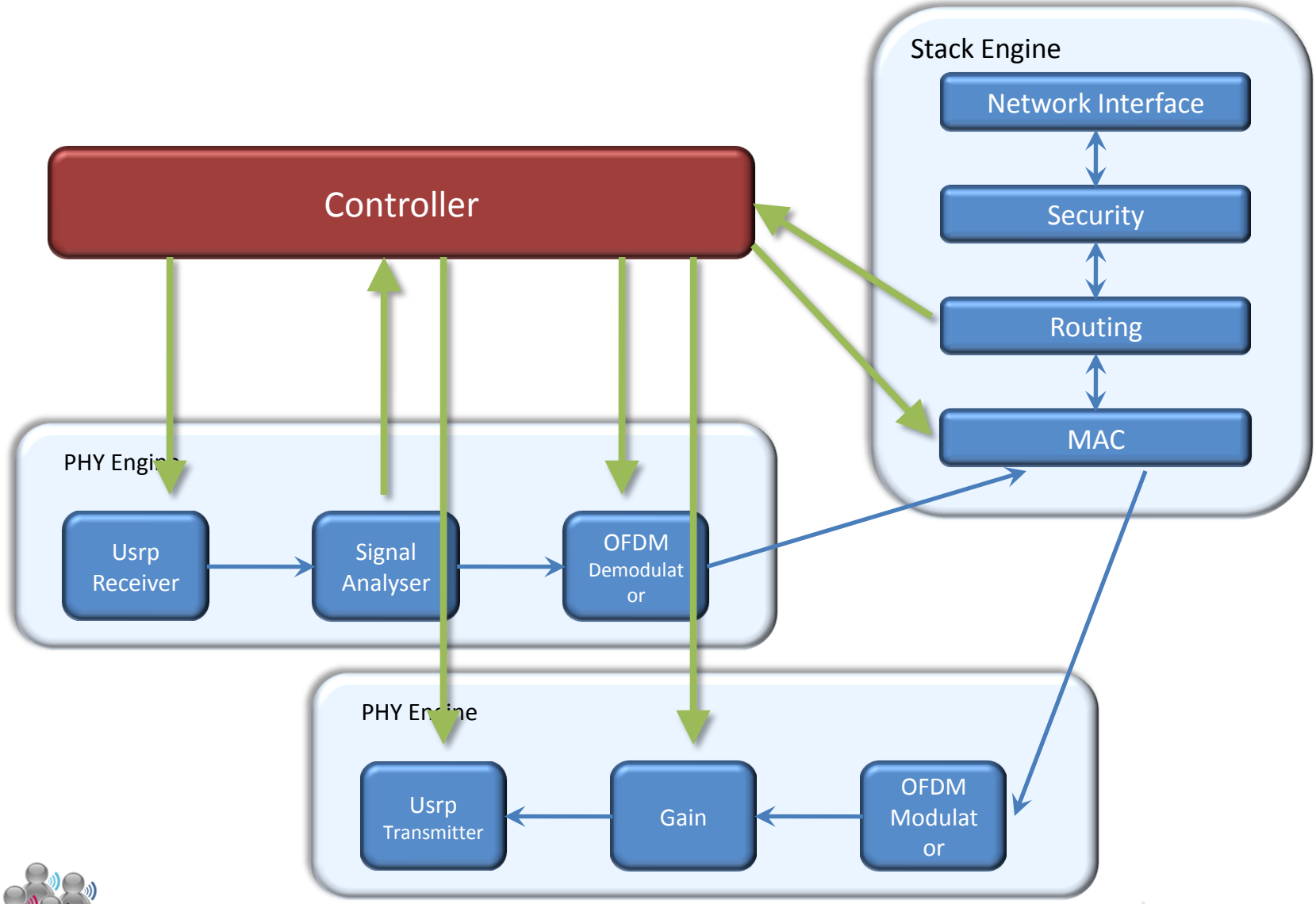
Controllers



Controllers



Controllers



Controllers

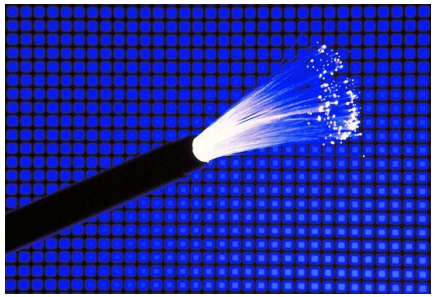
- Demo 3
  - Remote OFDM demo with controller
  - Display on analyser

OFDM Specifics

- Software Radio
- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- OFDM Specifics

OFDM Specifics

DAB digital radio



DSL

Wi Fi™

wimax

Lte™

DVB® Digital Video Broadcasting

Powerline Communications

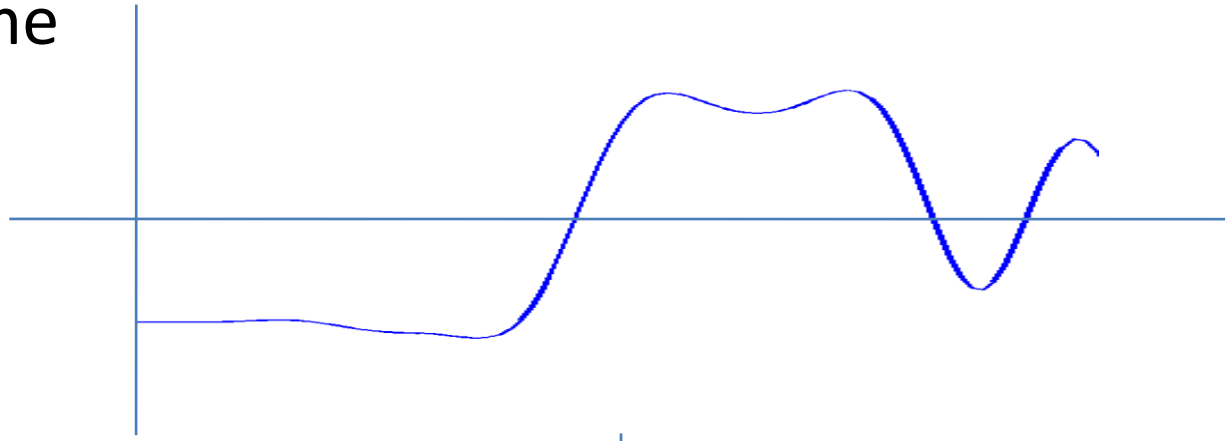
CREW

CTVR the telecommunications research centre

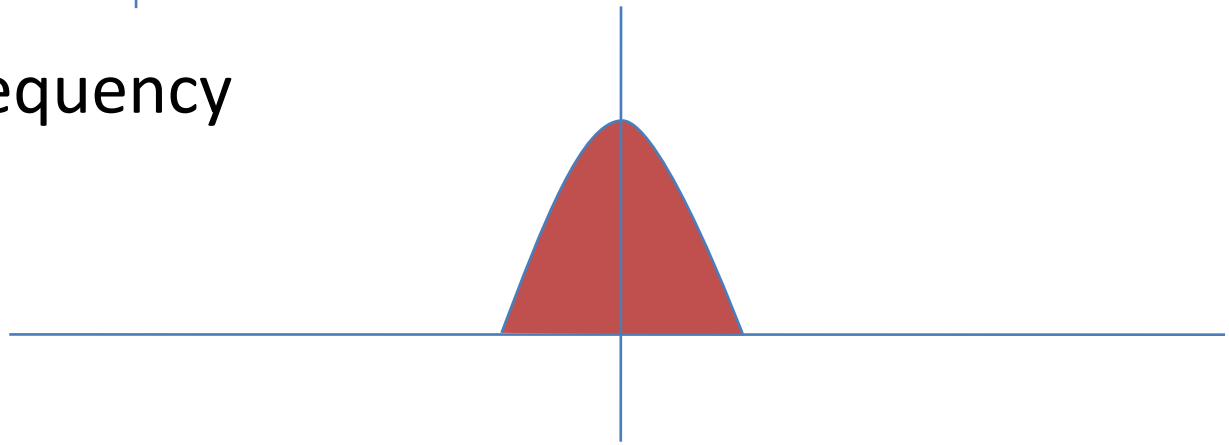
Single-Carrier Systems

Single-Carrier Systems

Time

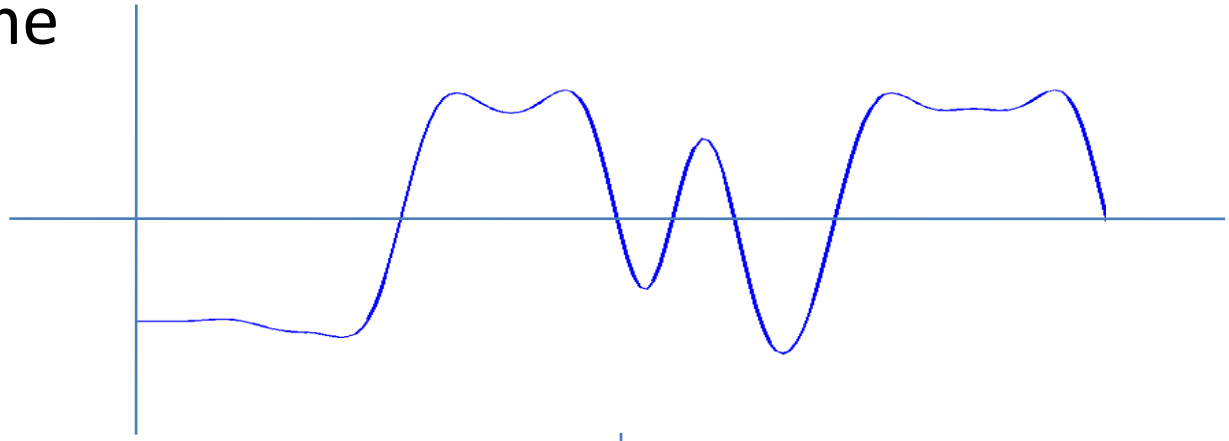


Frequency

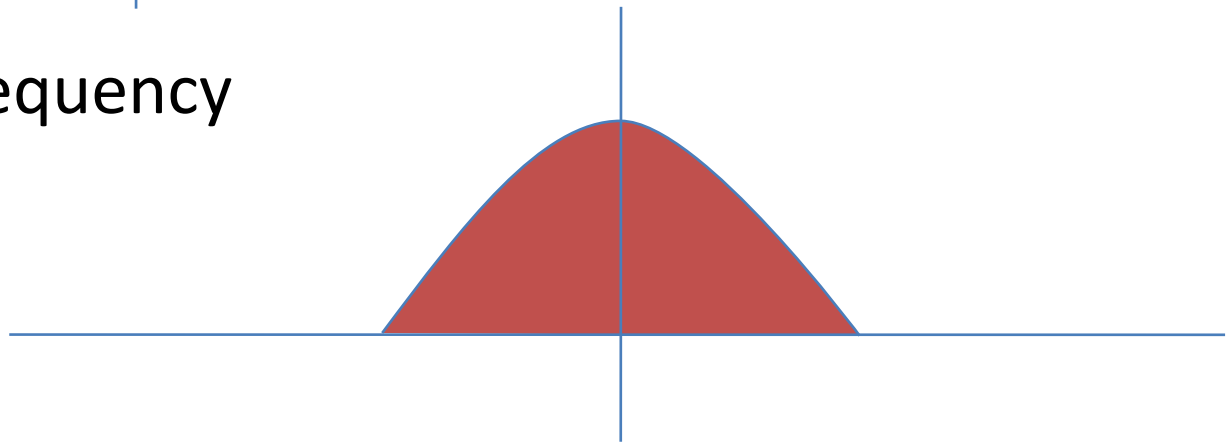


Single-Carrier Systems

Time

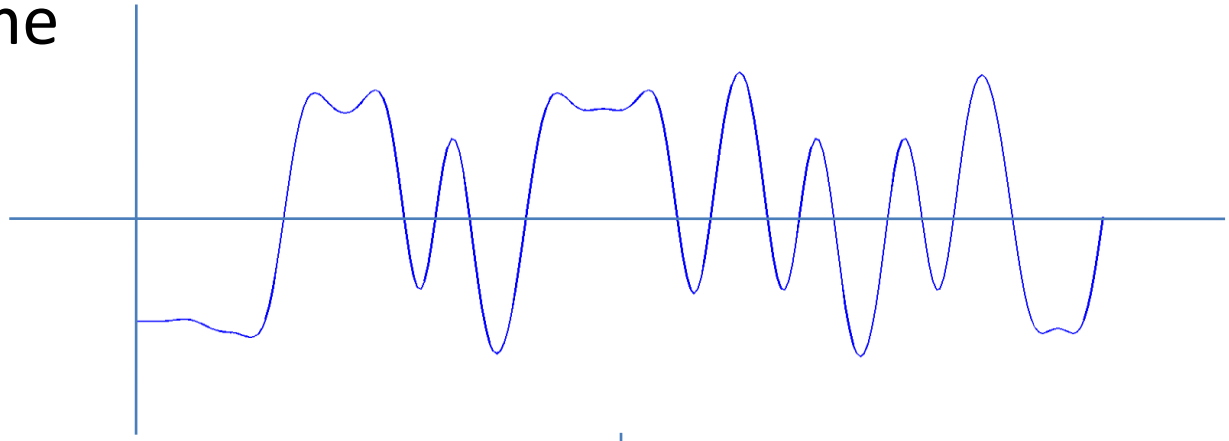


Frequency

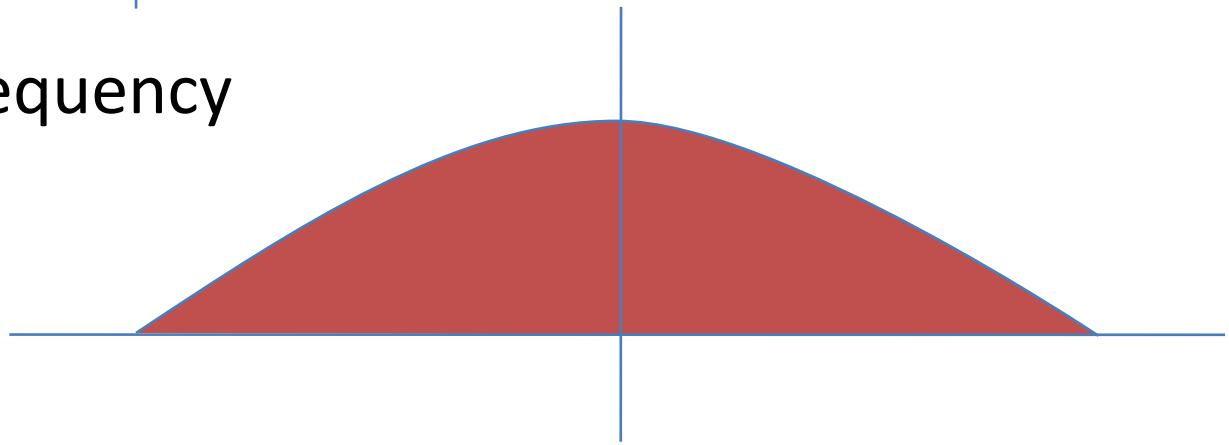


Single-Carrier Systems

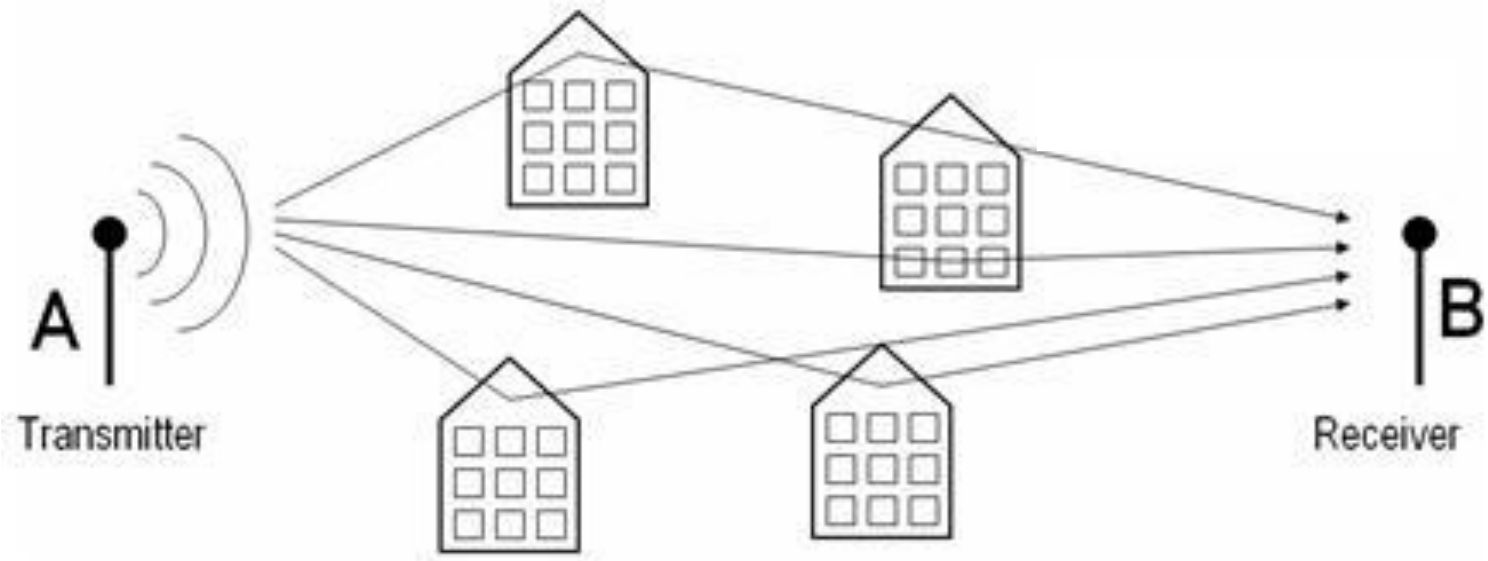
Time



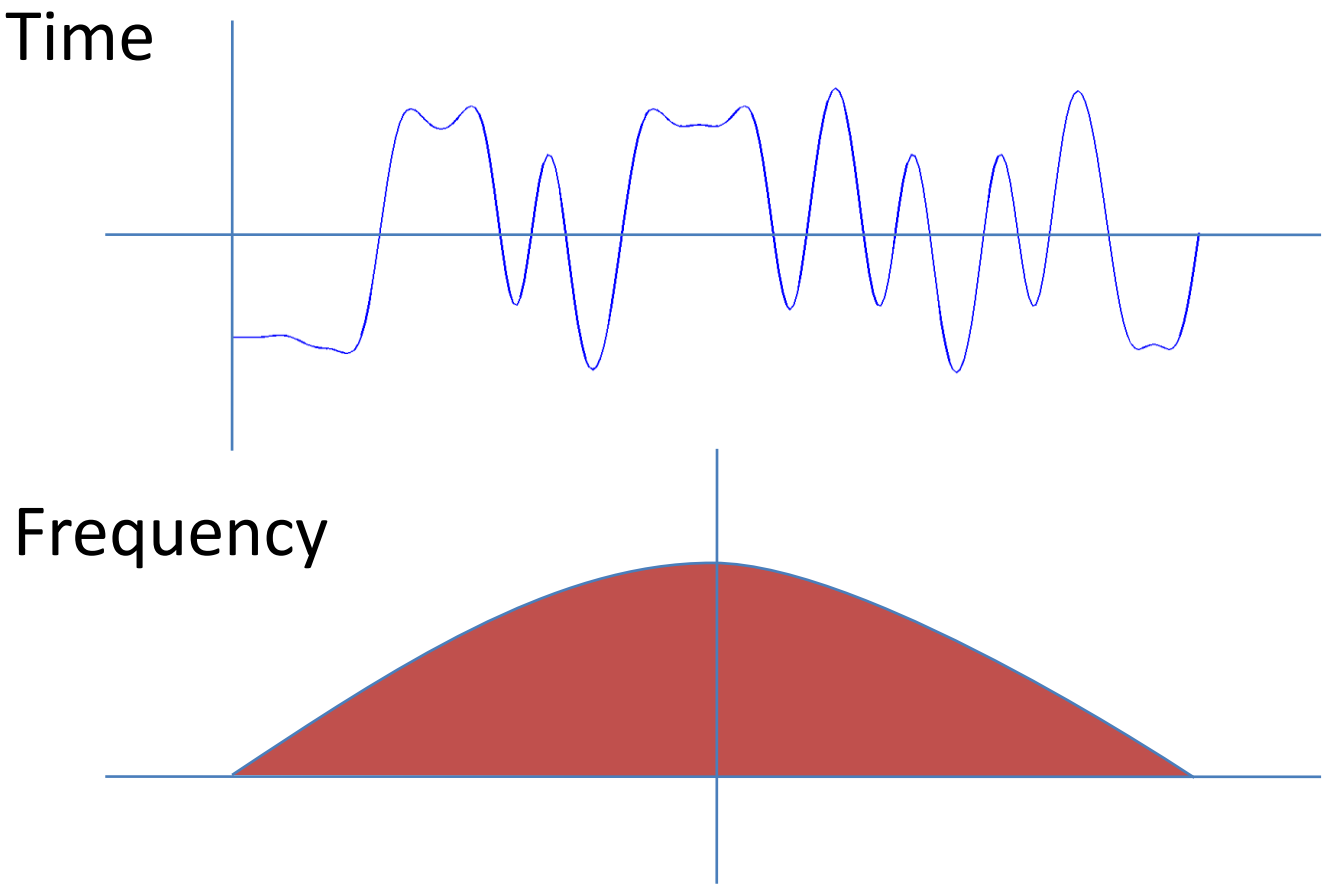
Frequency



Multipath Propogation

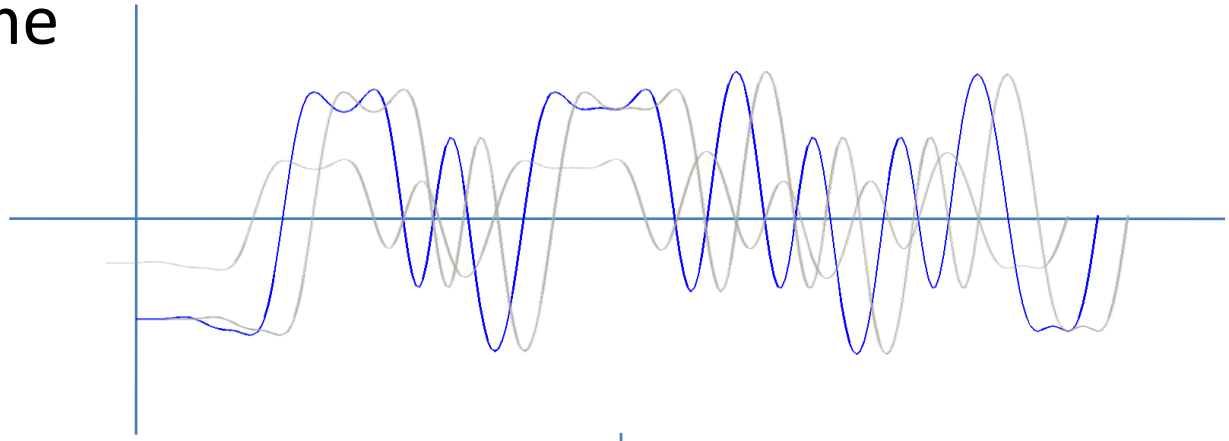


OFDM Specifics

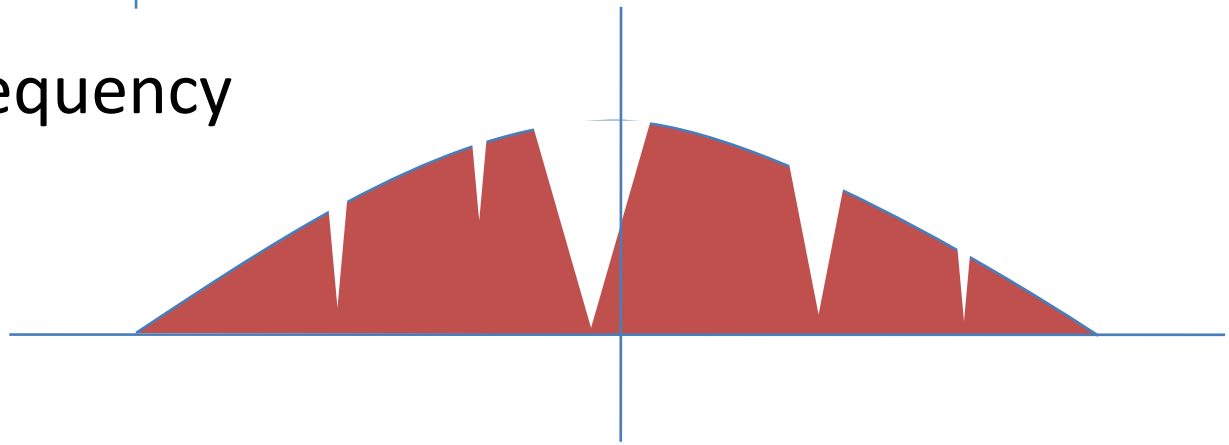


OFDM Specifics

Time



Frequency

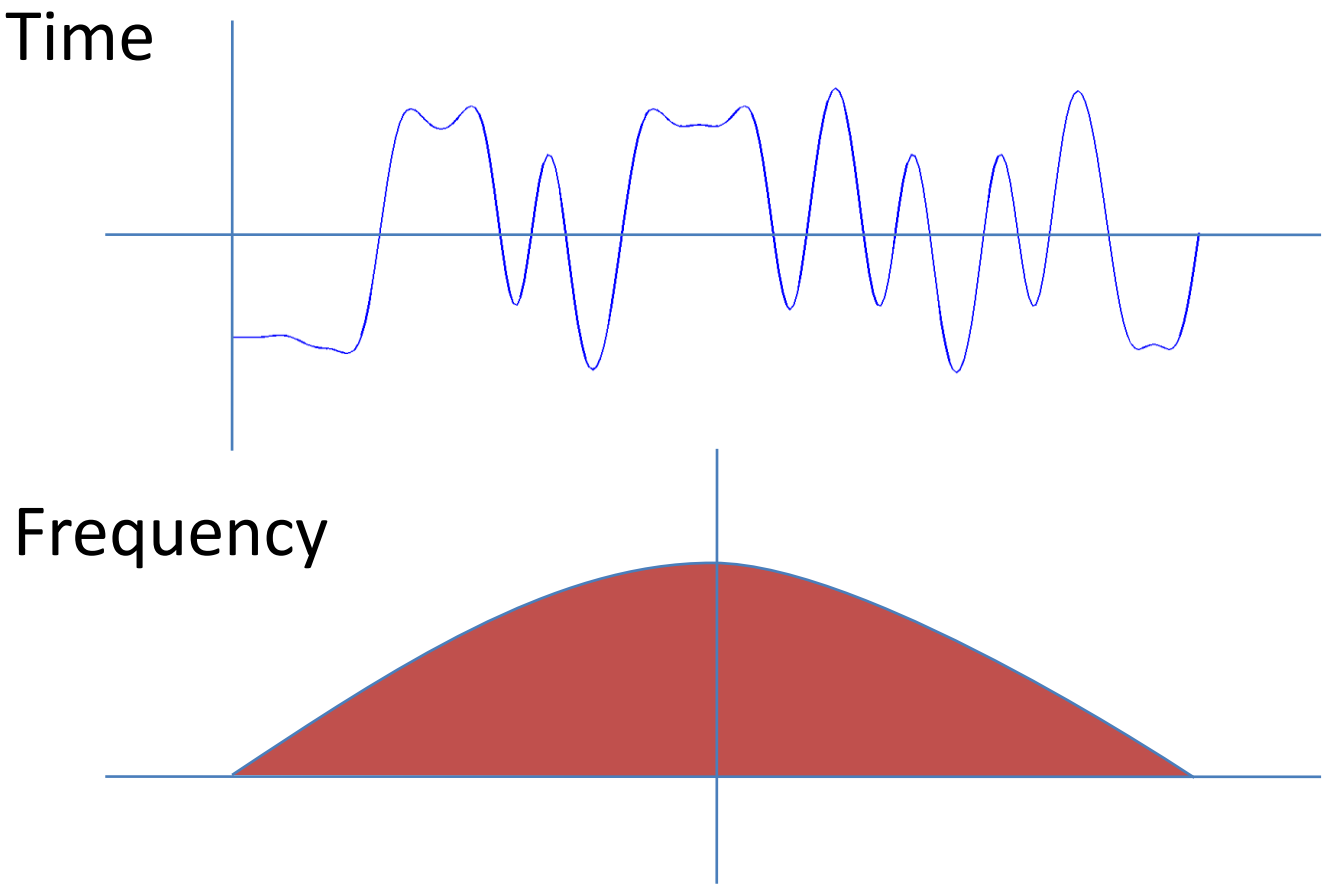


CREW

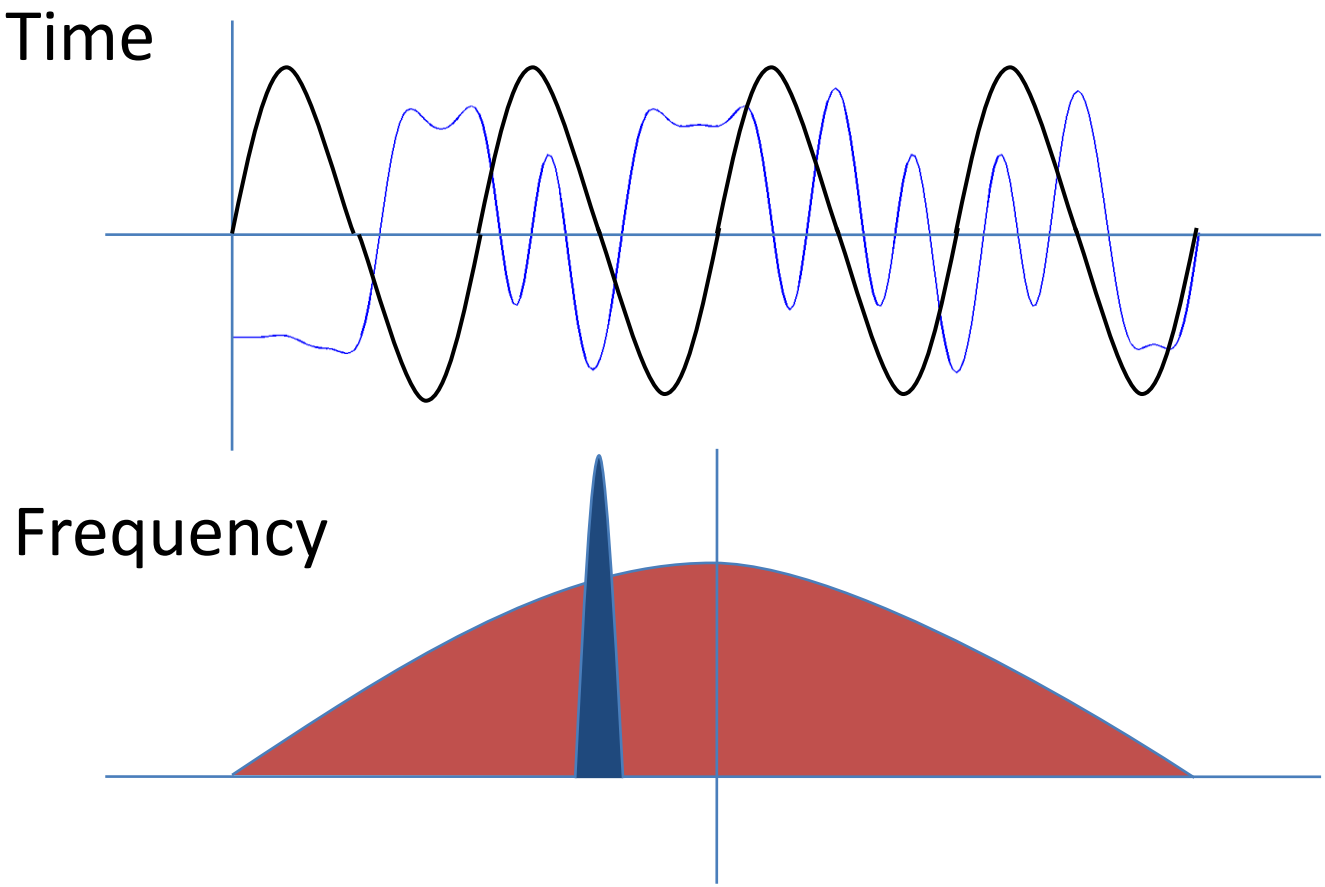
CTVR the telecommunications research centre

Narrowband Interference

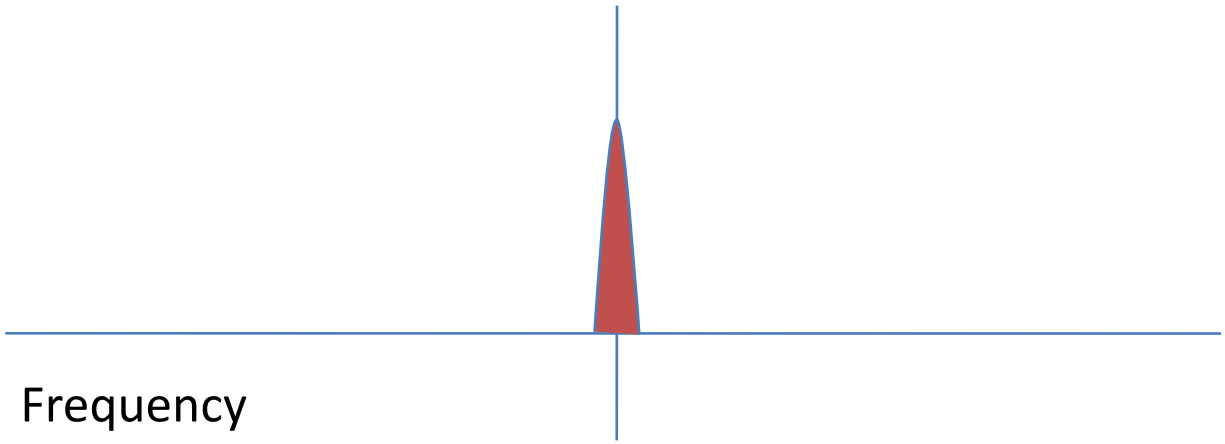
OFDM Specifics



OFDM Specifics

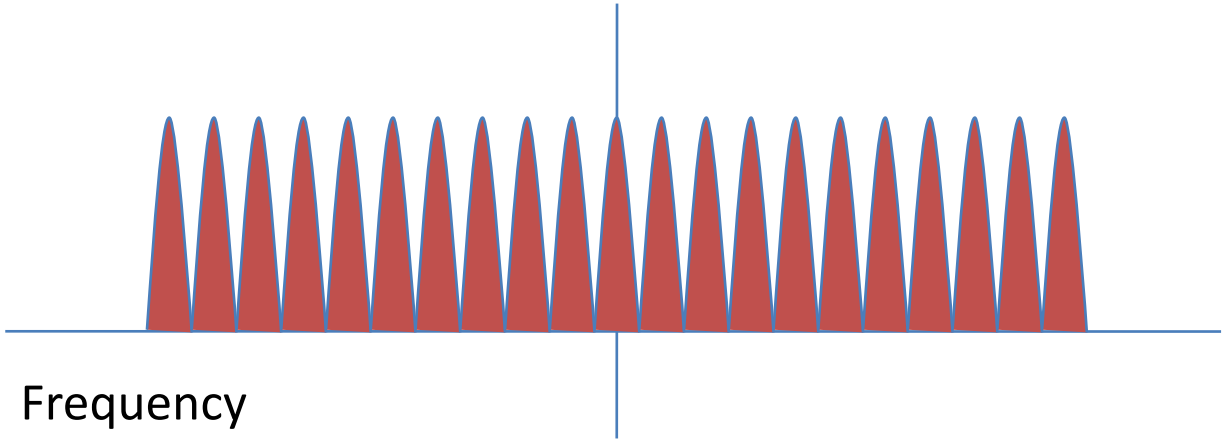


Single-Carrier System

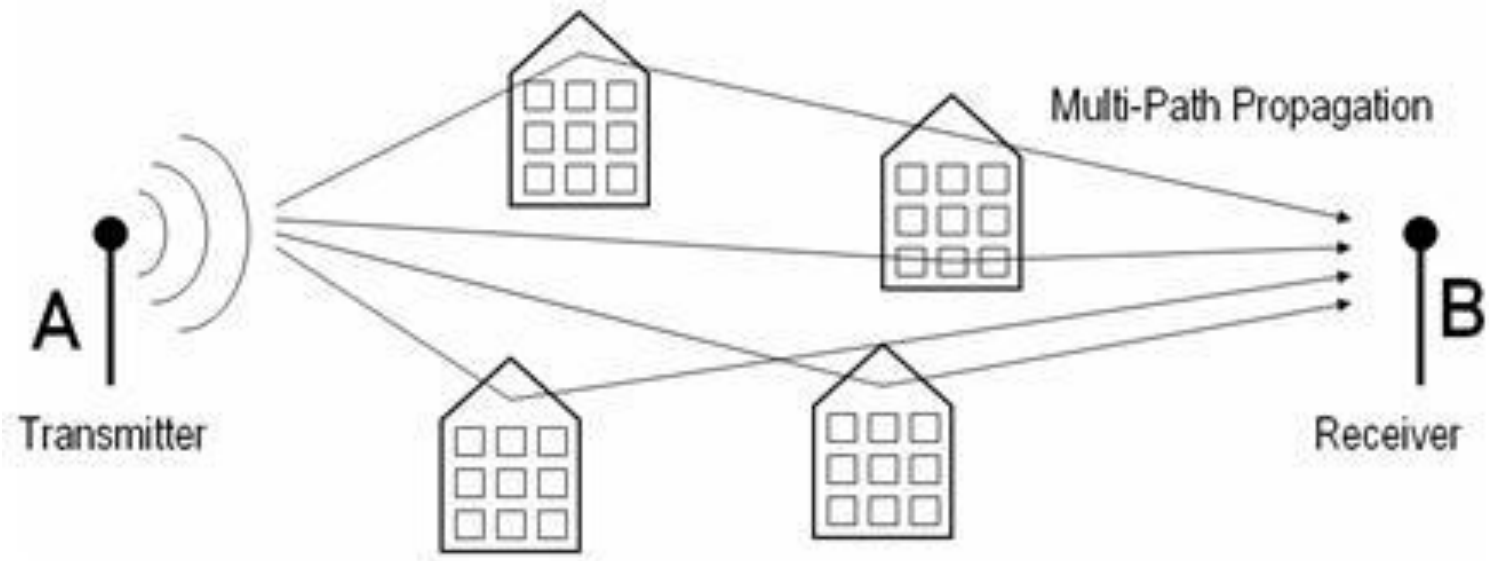


Multi-Carrier System

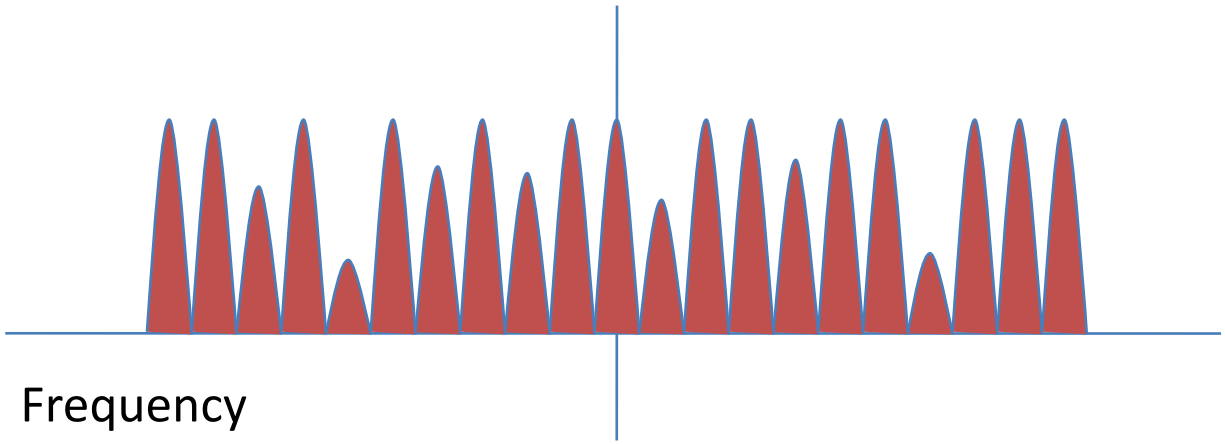
Frequency Division Multiplexing (FDM)



OFDM Specifics

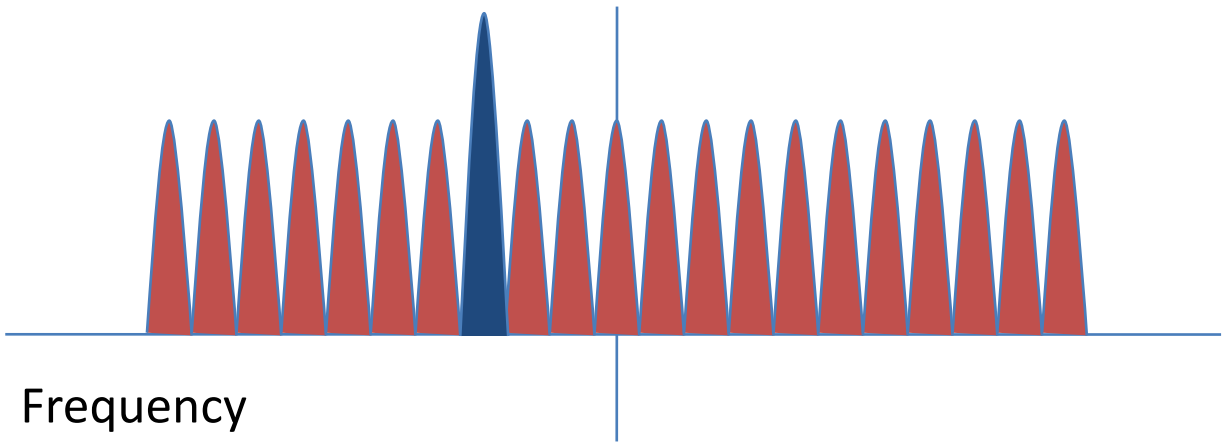


OFDM Specifics

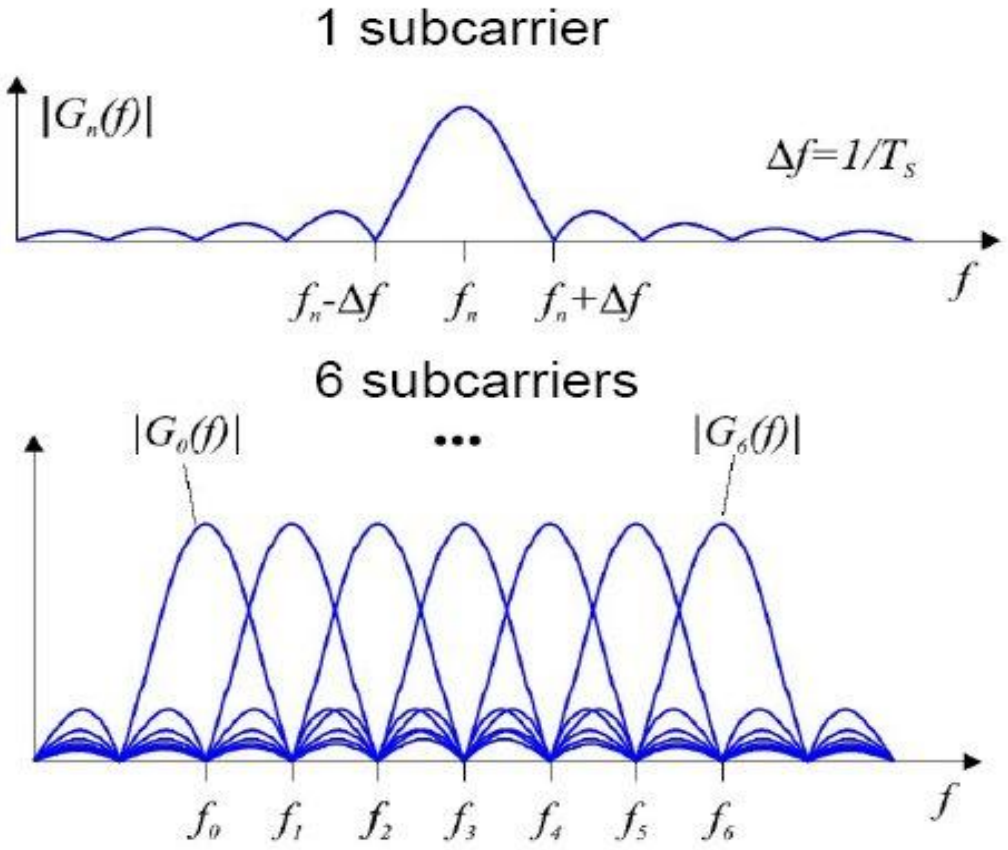


OFDM Specifics

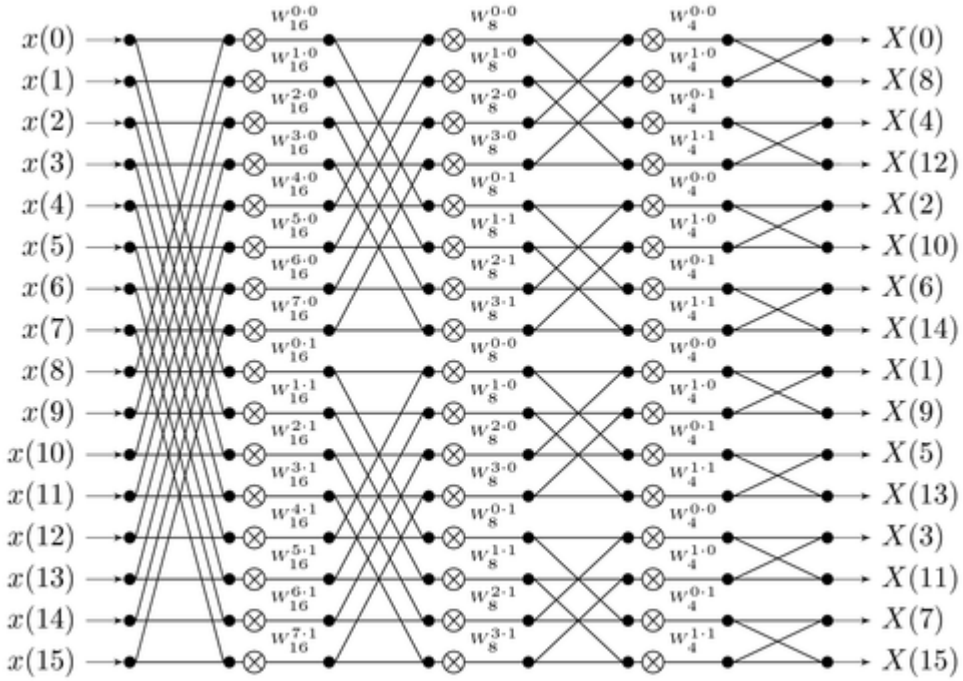
OFDM Specifics



Orthogonal Frequency Division Multiplexing

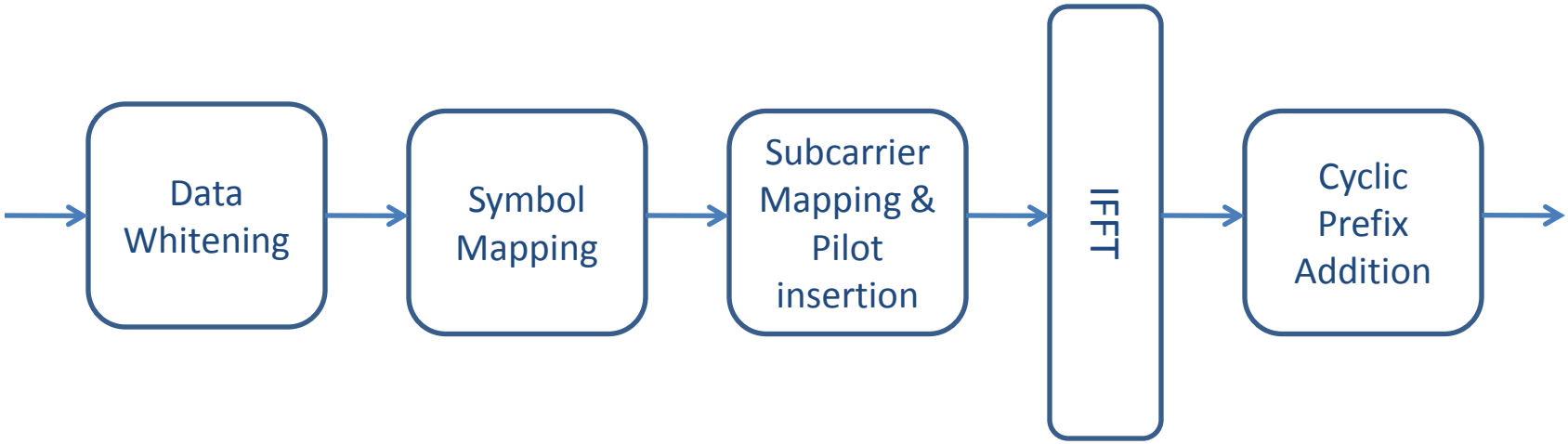


Fast Fourier Transform



OFDM Specifics

- OFDM Modulation



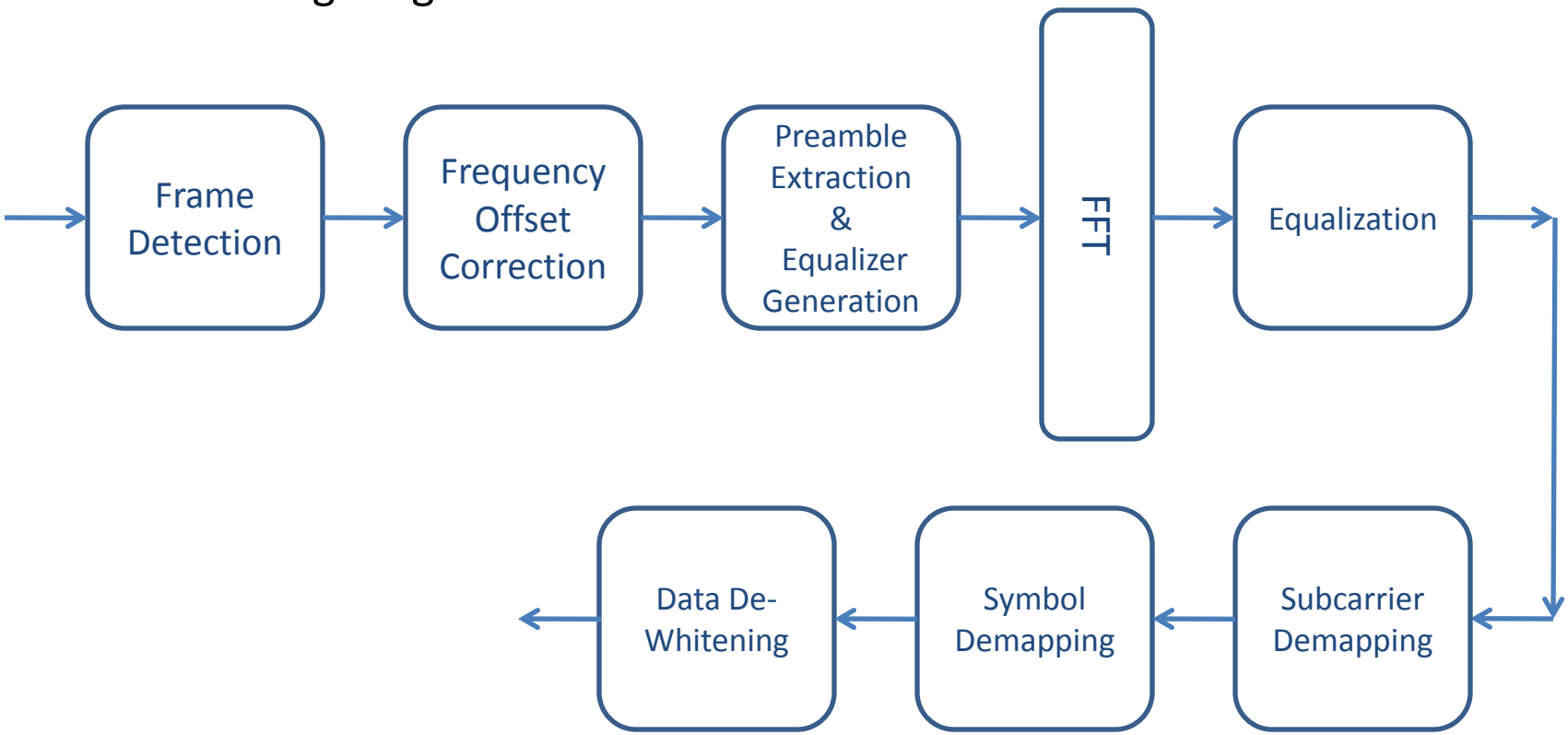
OFDM Specifics

- Demo 4
  - OFDM Modulator

OFDM Specifics

- OFDM Frame Structure

- OFDM Demodulation
  - Receiving a signal?

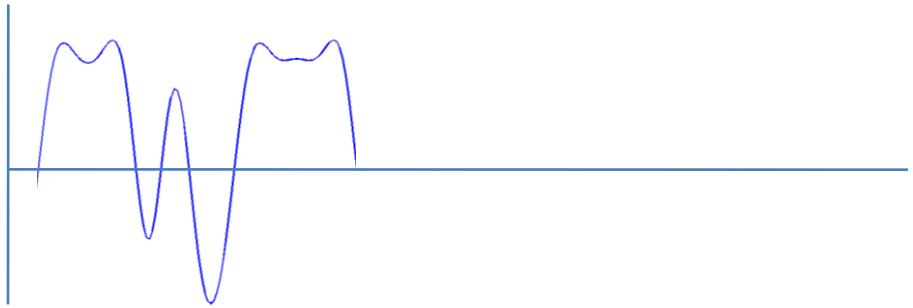


OFDM Specifics

- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization

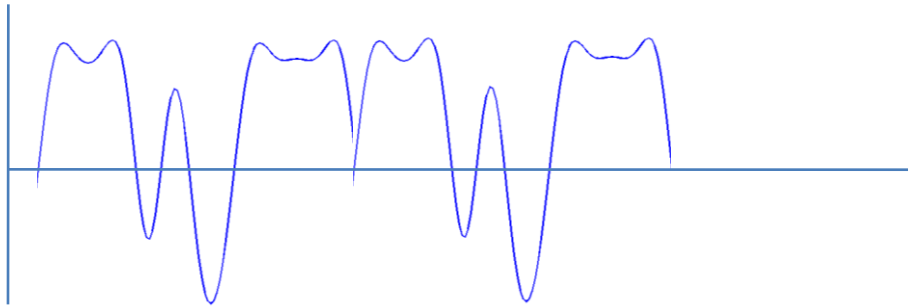
OFDM Specifics

- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization



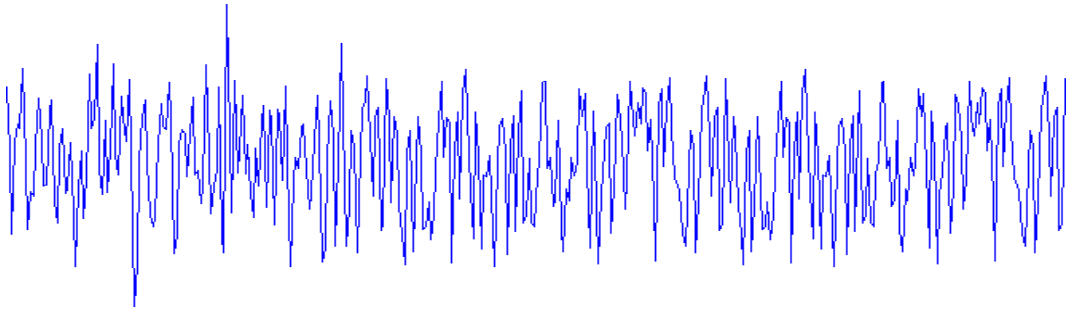
OFDM Specifics

- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization



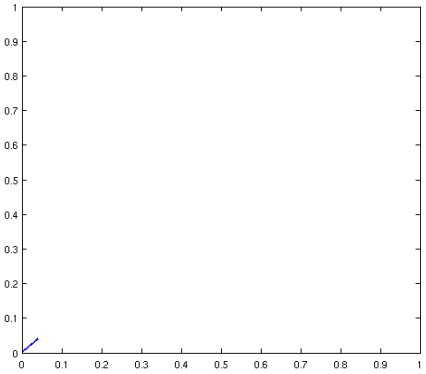
OFDM Specifics

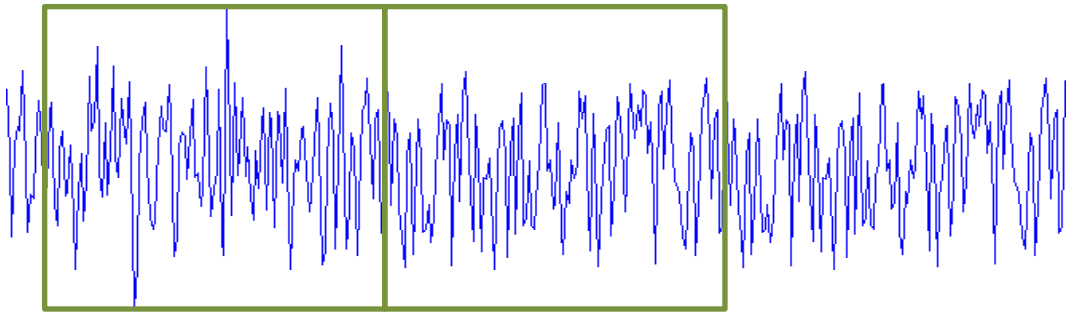
- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization

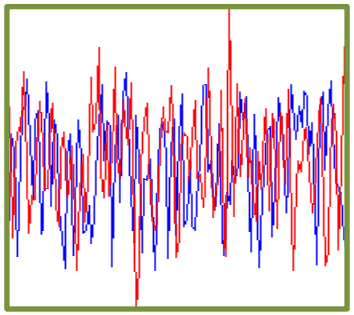


OFDM Specifics

- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization





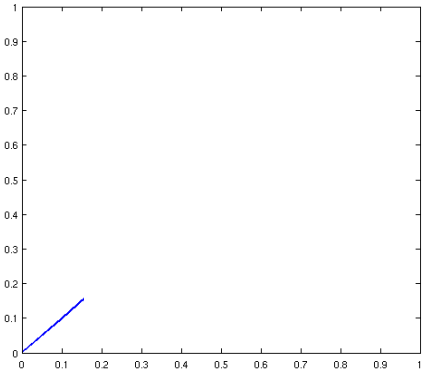


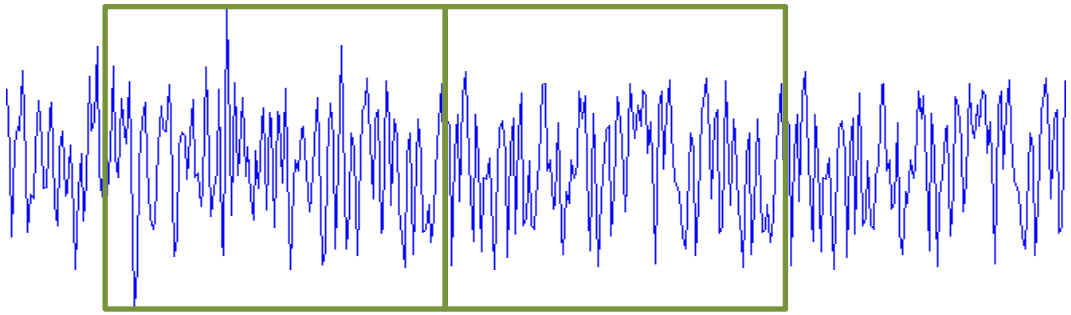
CREW

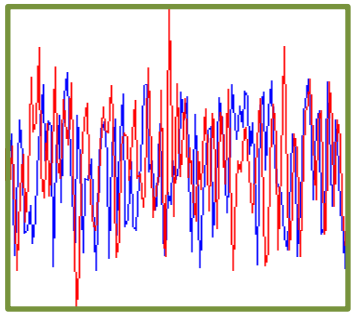
CTVR the telecommunications research centre

OFDM Specifics

- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization





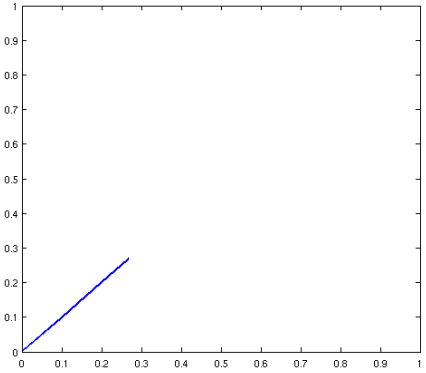


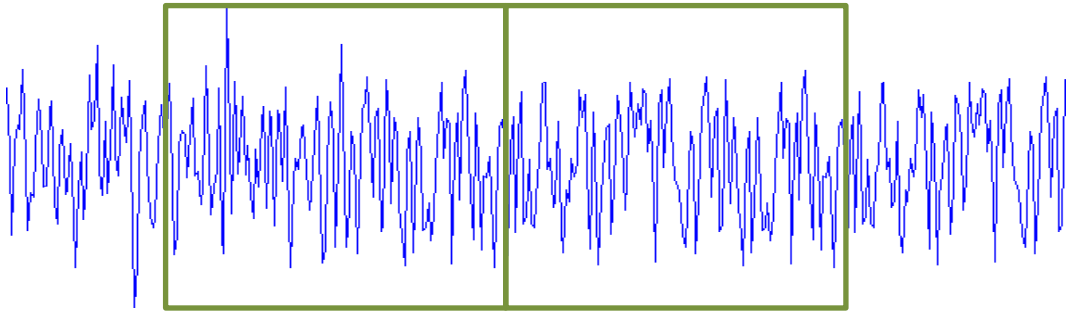
CREW

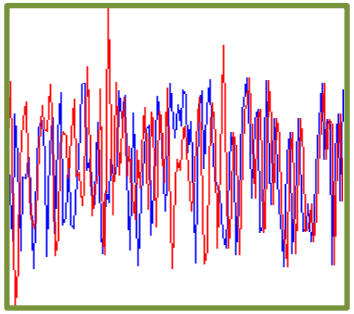
CTVR the telecommunications research centre

OFDM Specifics

- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization





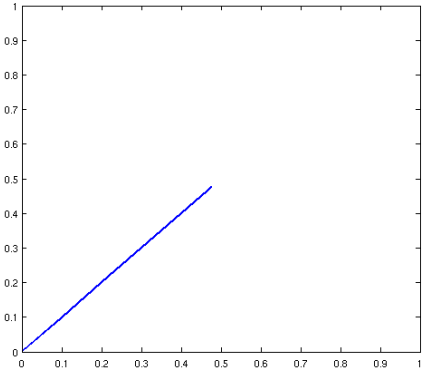


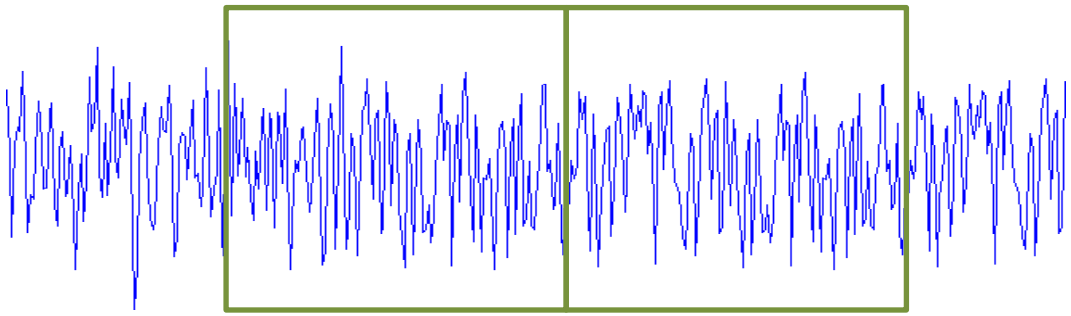
CREW

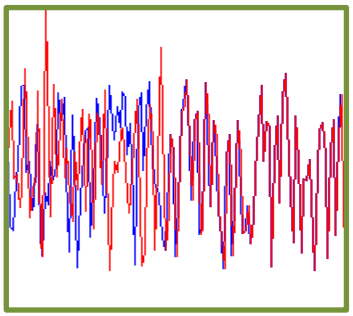
CTVR the telecommunications research centre

OFDM Specifics

- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization





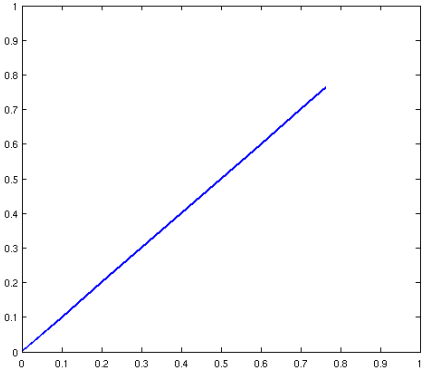


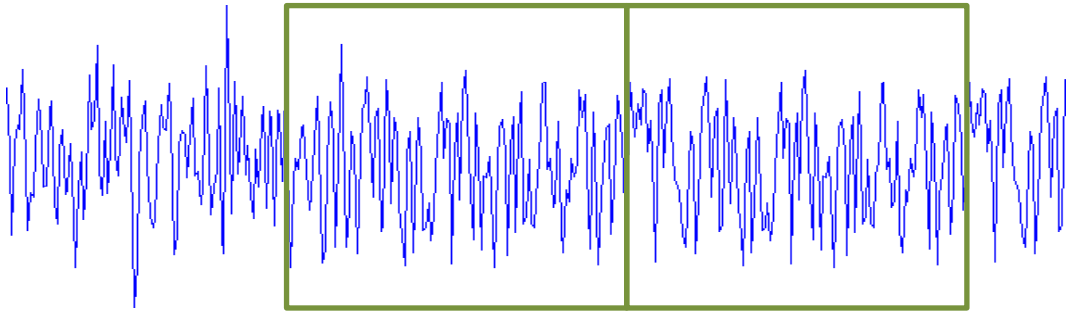
CREW

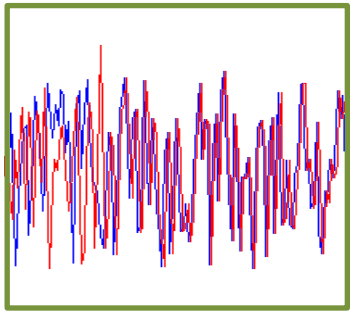
CTVR the telecommunications research centre

OFDM Specifics

- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization





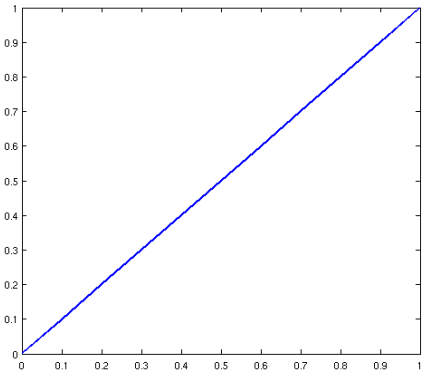


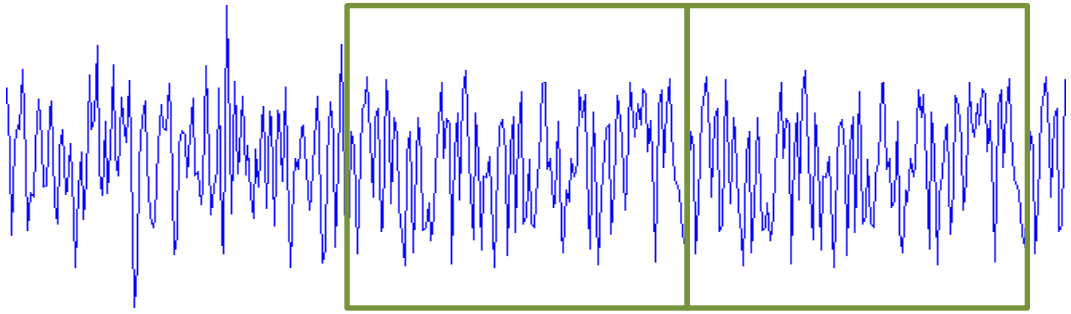
CREW

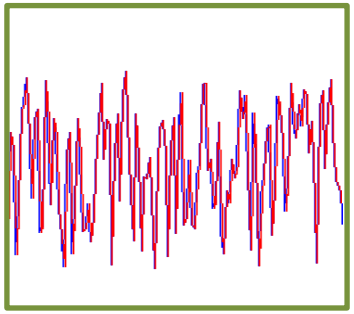
CTVR the telecommunications research centre

OFDM Specifics

- OFDM Demodulation
  - Time synchronization
  - Frequency synchronization







CREW

CTVR the telecommunications research centre

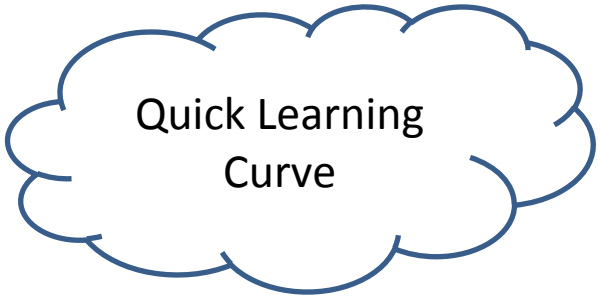
OFDM Specifics

- Demo 5
  - OFDM Demodulator

OFDM Specifics

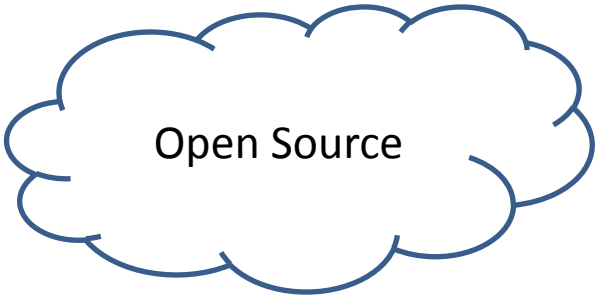
Why use Iris?

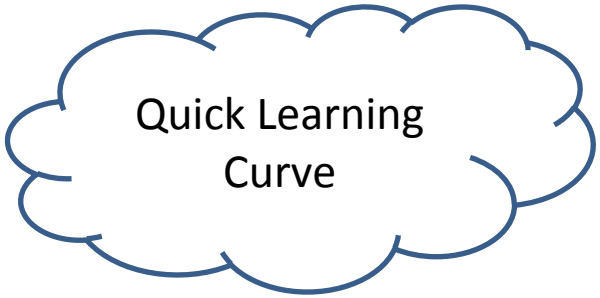
Why use Iris?



OFDM Specifics

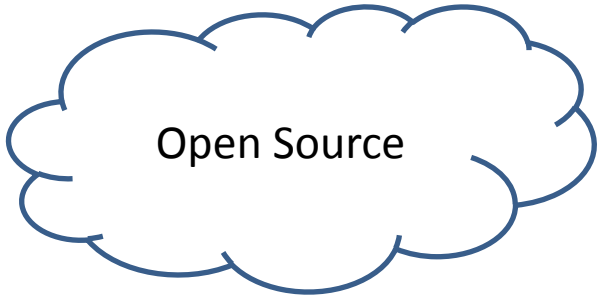
Why use Iris?

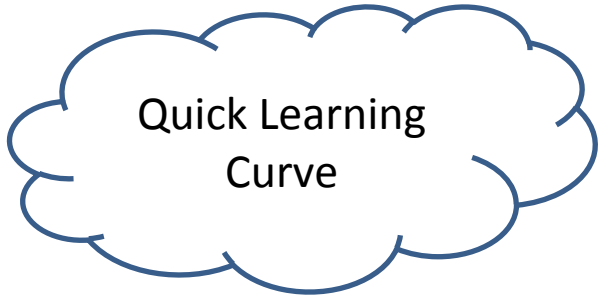




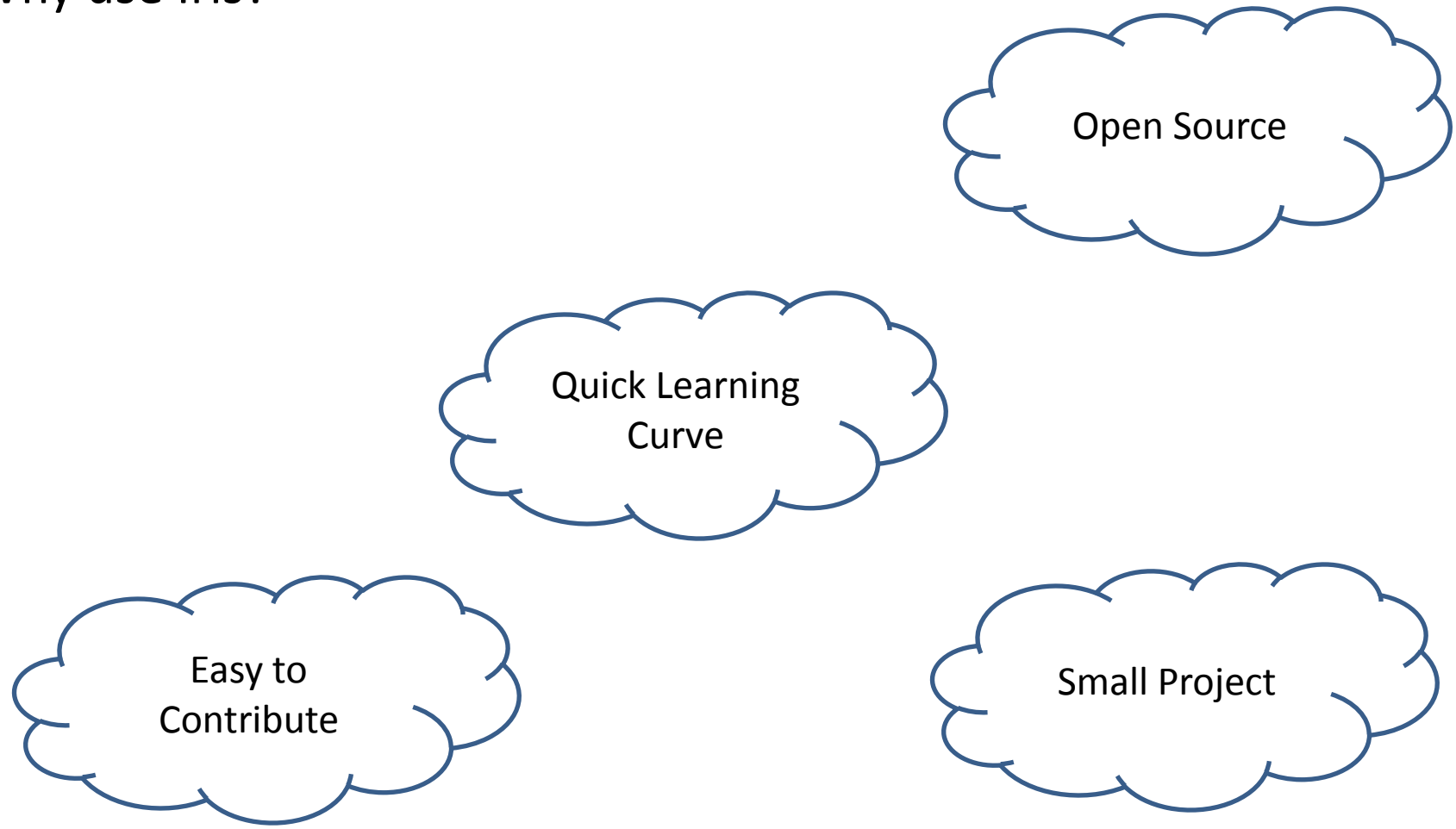
OFDM Specifics

Why use Iris?

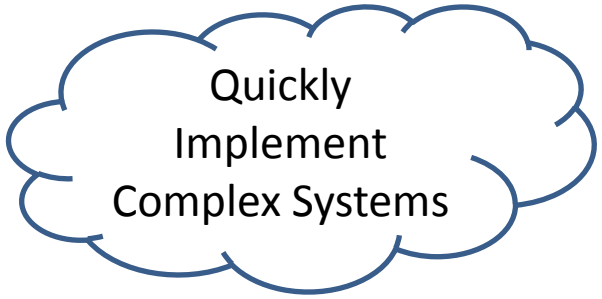


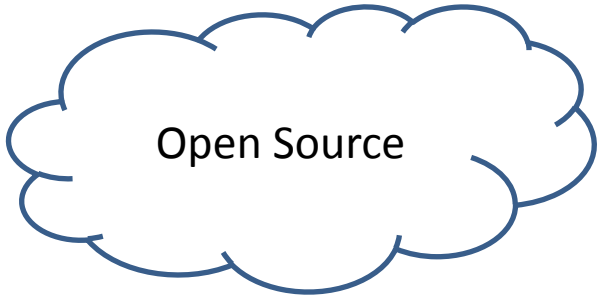


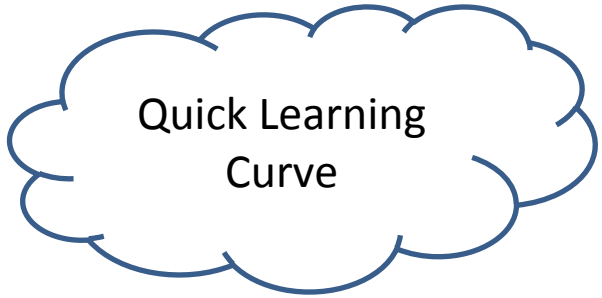
Why use Iris?


Open Source
Quick Learning Curve
Easy to Contribute
Small Project

Why use Iris?


Quickly
Implement
Complex Systems


Open Source


Quick Learning
Curve


Easy to
Contribute


Small Project

Try it out

https://github.com/softwareradiosystems

Thank you

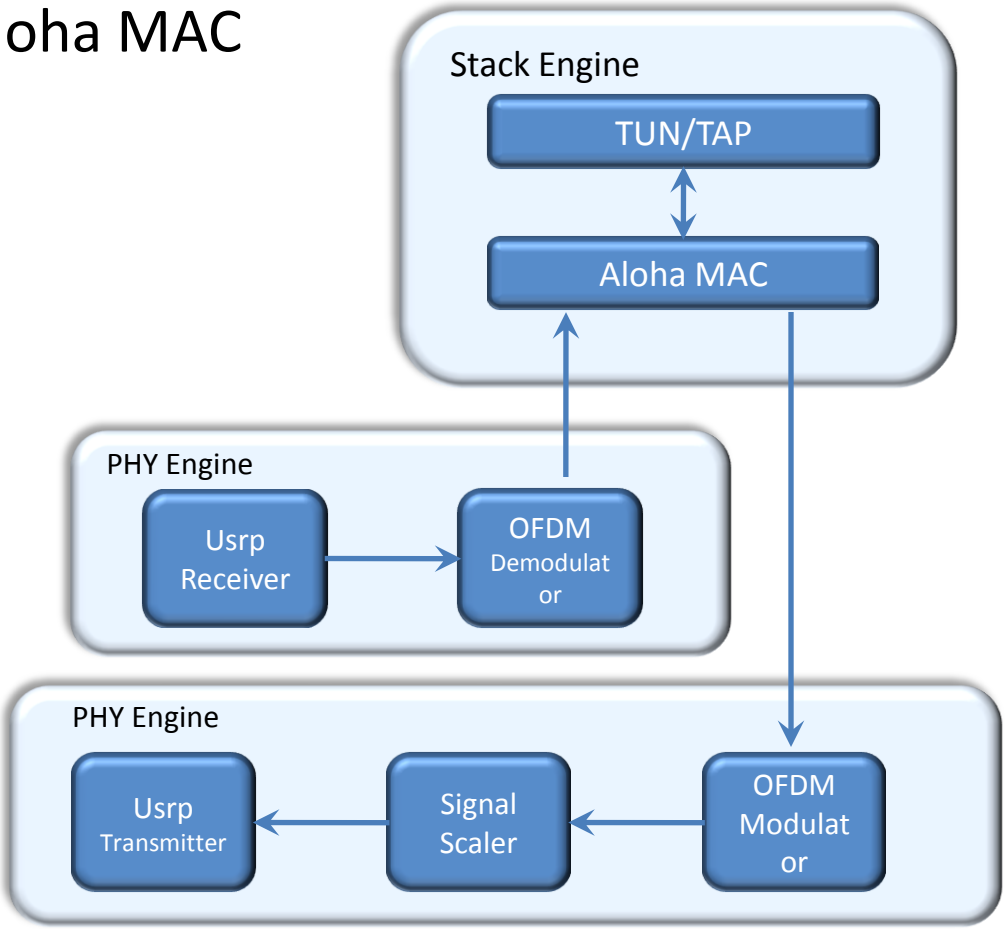
suttonpd@tcd.ie

paul@softwareradiosystems.com

Additional Material

MAC Development with Iris

- Simple Aloha MAC



http://www.puschmann.net/page/?p=156

MAC Development with Iris

- Simple Aloha MAC
  - Send packet
  - Wait for ACK
  - Receive ACK / Timeout
  - Resend packet

http://www.puschmann.net/page/?p=156

MAC Development with Iris

- Simple Aloha MAC
  - Send packet
  - Wait for ACK
  - Receive ACK / Timeout
  - Resend packet

Andre Puschmann
andrepuschmann

Ilmenau University of Technology
Ilmenau, Germany
andre.puschmann@tu-ilmenau.de
http://www.tu-ilmenau.de/ics

Contributions | Repositories | Public Activity | Follow

Filter andrepuschmann's repositories... All Sources Forks Mirrors

iris_modules
C++ ★ 1 1
forked from softwareradiosystems/iris_modules
The main Iris modules repository.
Last updated 7 days ago

iris_core
C++ ★ 1 1
forked from softwareradiosystems/iris_core
The core Iris software radio architecture.
Last updated 9 days ago

http://www.puschmann.net/page/?p=156

CREW

CTVR the telecommunications research centre