Iris Software Radio Architecture

Paul Sutton

14th January 2014

CREW Training Days

Ghent, Belgium

Software Radio Systems

CTVR the telecommunications research centre

Outline

- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- Case Study - OFDM
- Interesting Applications

Iris Overview

- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- Case Study - OFDM
- Interesting Applications

Iris Overview

What is Iris?

Iris Overview

What is Iris?

A Software Radio Architecture

Iris Overview

What is Iris?

Reconfigurable

A Software Radio Architecture

Iris Overview

What is Iris?

Reconfigurable

A Software Radio Architecture

GPP – Based (primarily)

Iris Overview

What is Iris?

Reconfigurable

Component-Based

A Software Radio Architecture

GPP – Based (primarily)

Iris Overview

What is Iris?

Reconfigurable

Component-Based

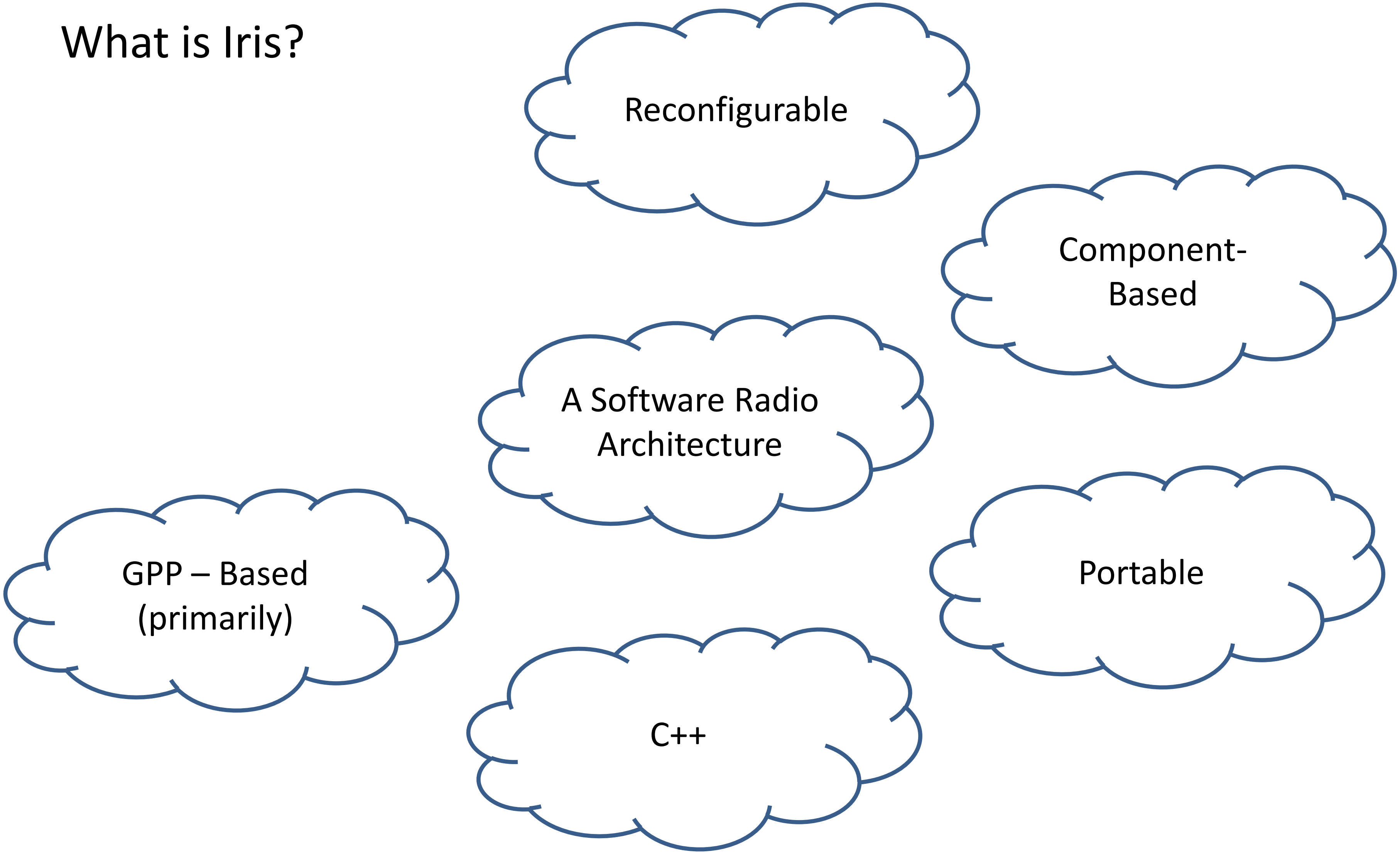
A Software Radio Architecture

GPP – Based (primarily)

C++

Iris Overview

What is Iris?



Reconfigurable

Component-Based

A Software Radio Architecture

GPP – Based (primarily)

Portable

C++

Iris Overview

What is Iris?

Reconfigurable

Extensible

Component-Based

A Software Radio Architecture

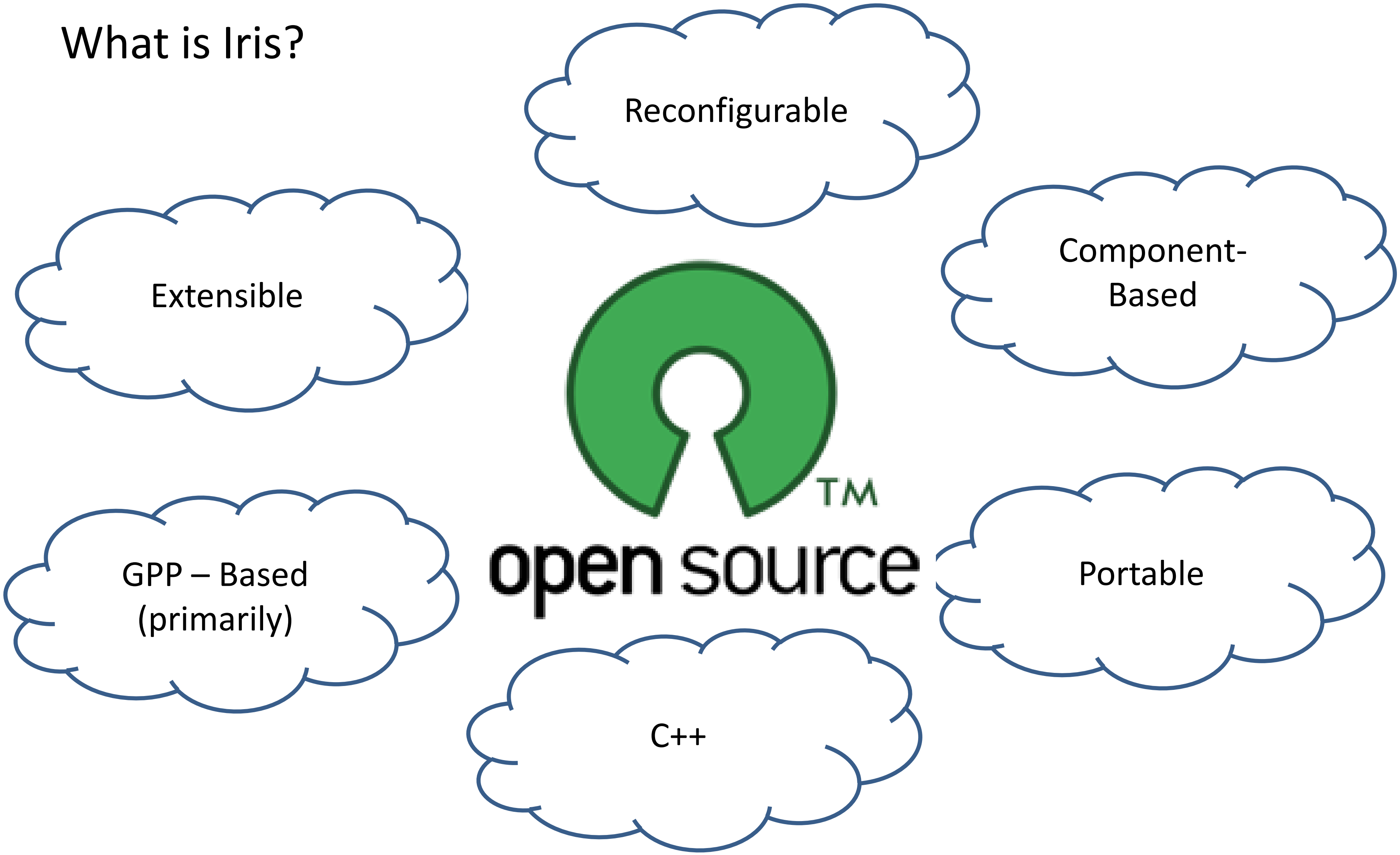
GPP – Based (primarily)

Portable

C++

Iris Overview

What is Iris?



Reconfigurable

Extensible

Component-Based

open source

GPP – Based (primarily)

Portable

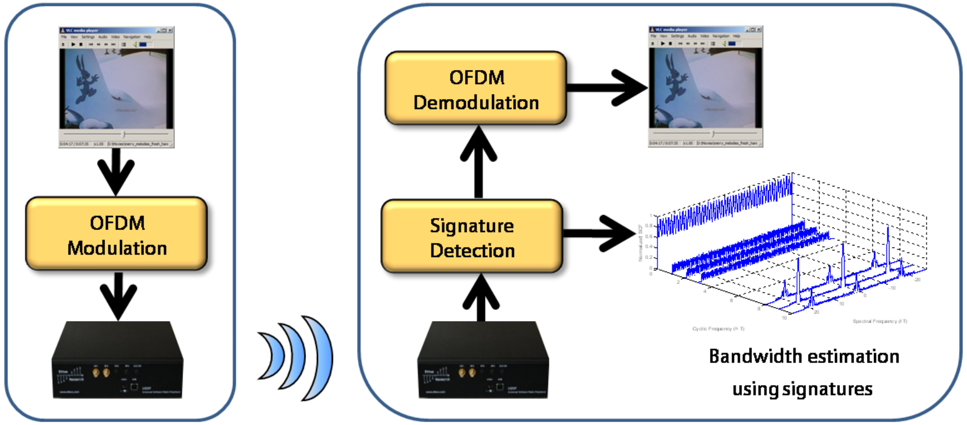
C++

Iris Overview

What can I do with Iris?

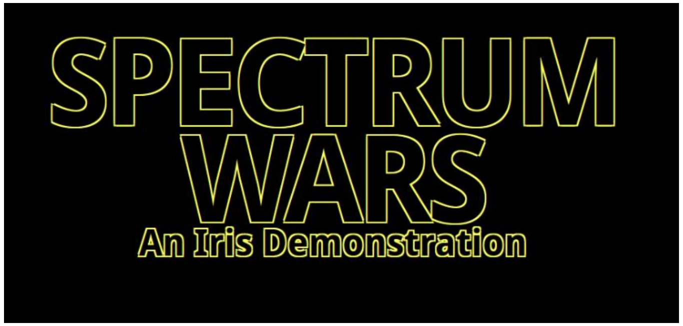
Iris Overview

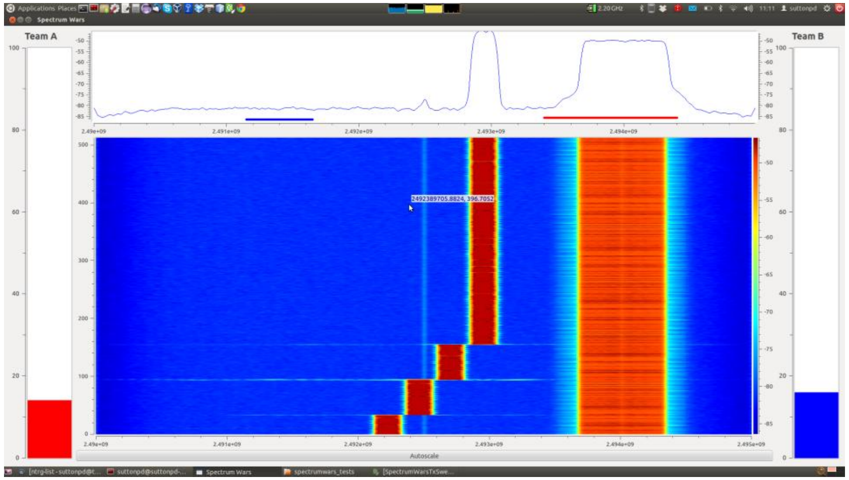
What can I do with Iris?

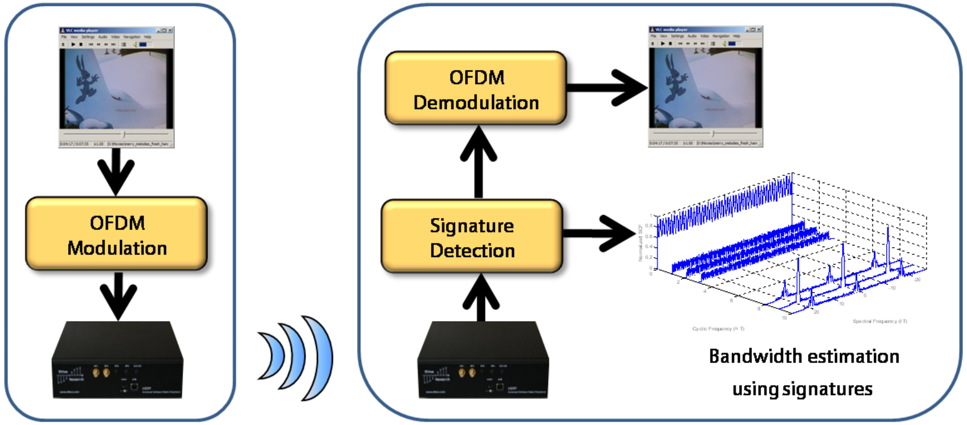


Iris Overview

What can I do with Iris?

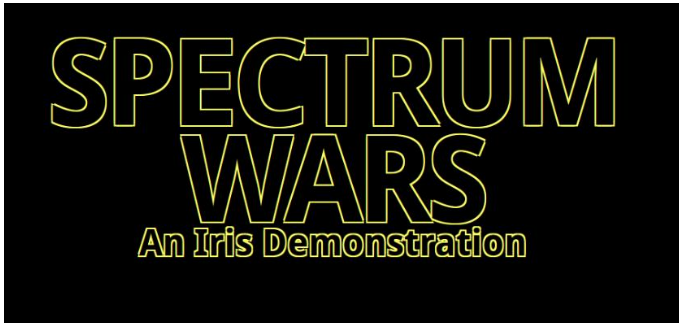


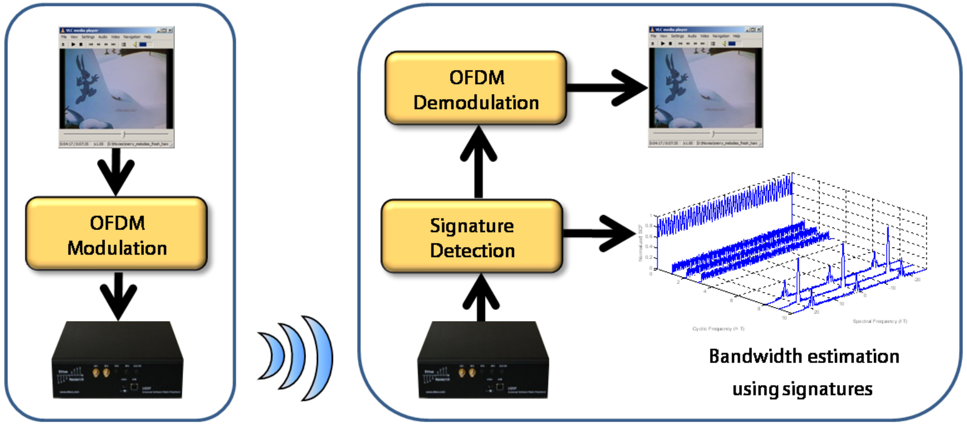


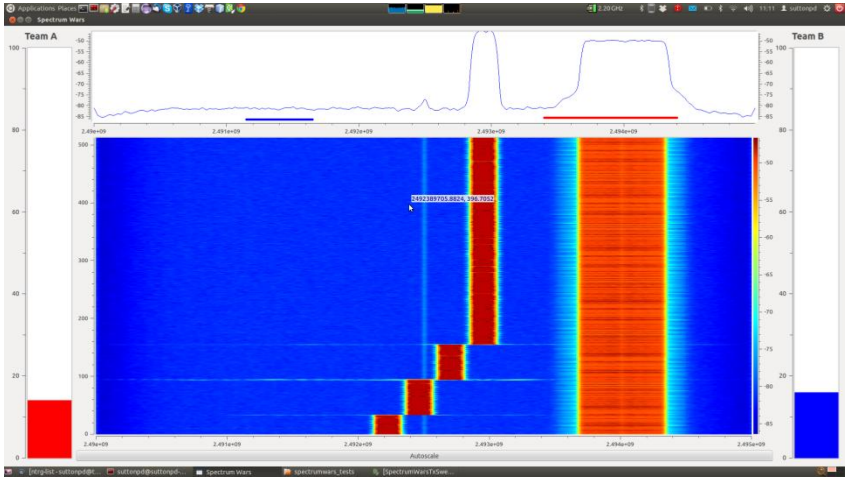


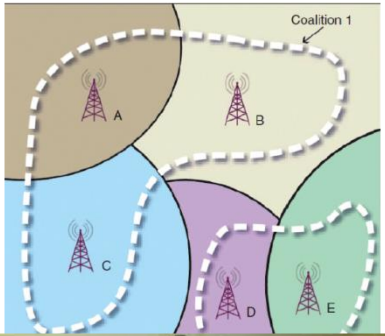
Iris Overview

What can I do with Iris?







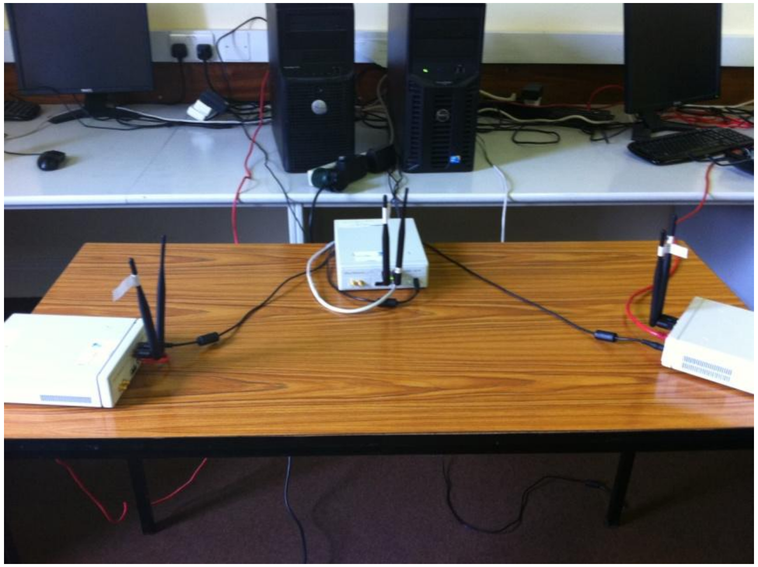


Software Radio Systems

CTVR the telecommunications research centre

Iris Overview

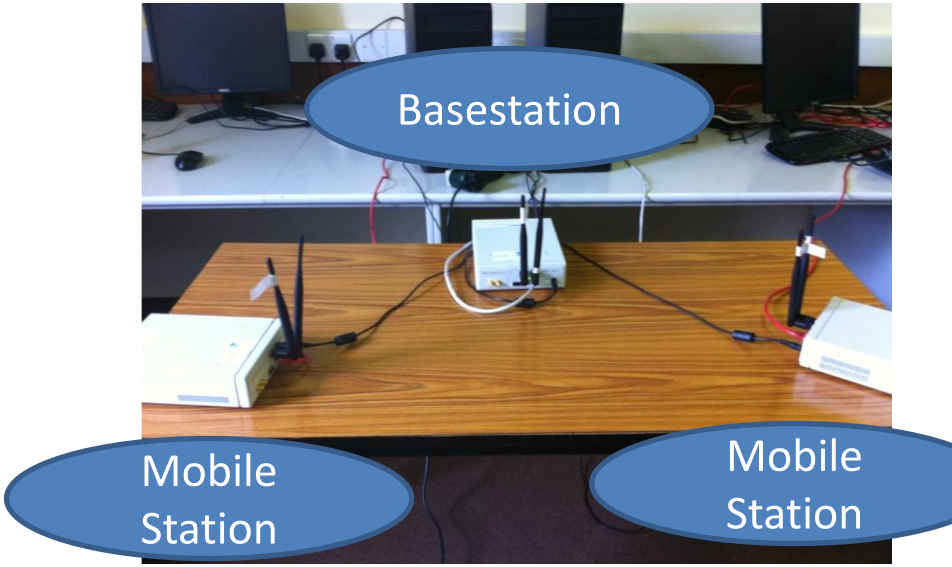
What can I do with Iris?



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

Iris Overview

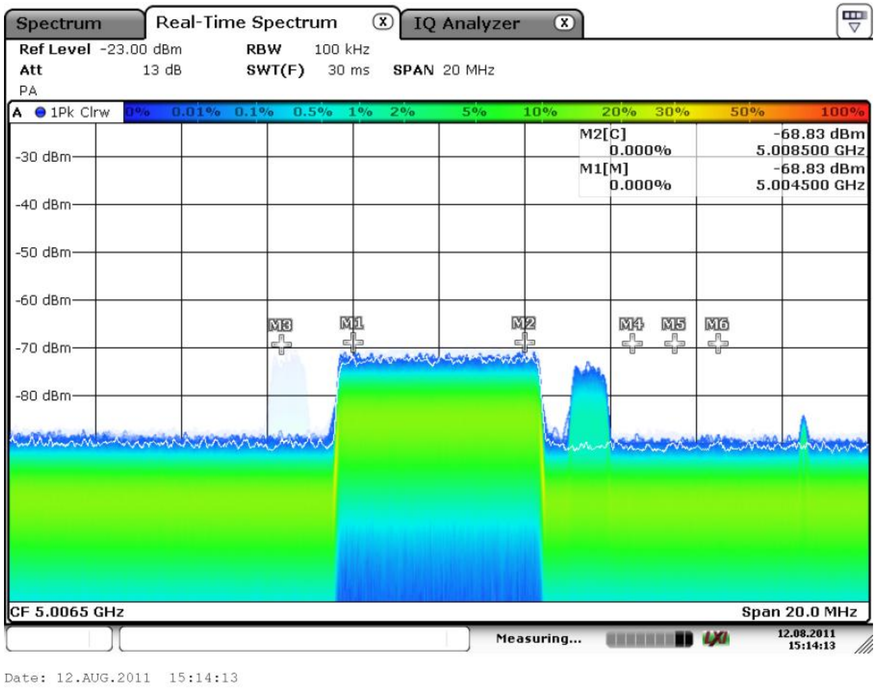
What can I do with Iris?



- 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?



Spectrum | Real-Time Spectrum ⓧ | IQ Analyzer ⓧ

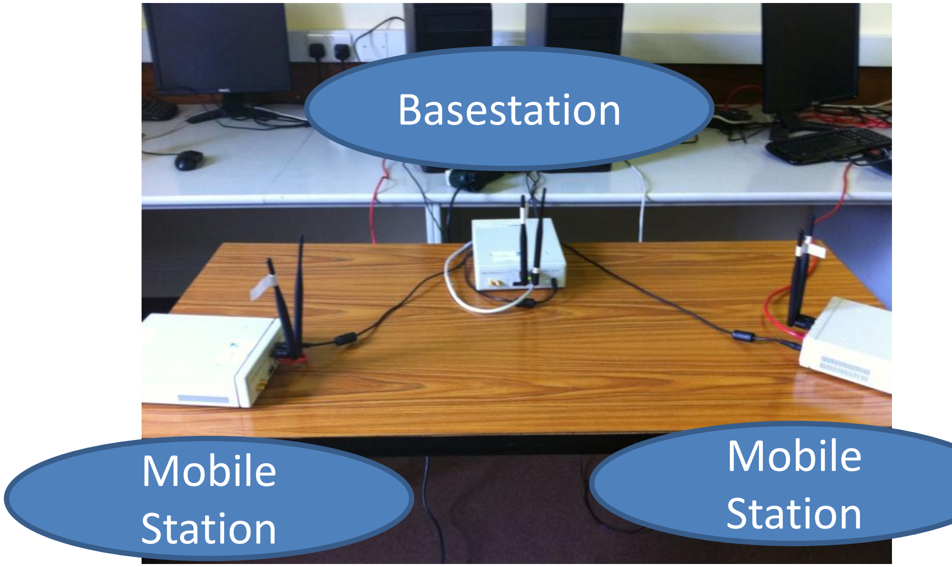
Ref Level -23.00 dBm RBW 100 kHz
Att 13 dB SWT(F) 30 ms SPAN 20 MHz
PA

M2[C] -68.83 dBm
0.000% 5.008500 GHz
M1[M] -68.83 dBm
0.000% 5.004500 GHz

CF 5.0065 GHz Span 20.0 MHz

Measuring... 12.08.2011 15:14:13

Date: 12.AUG.2011 15:14:13



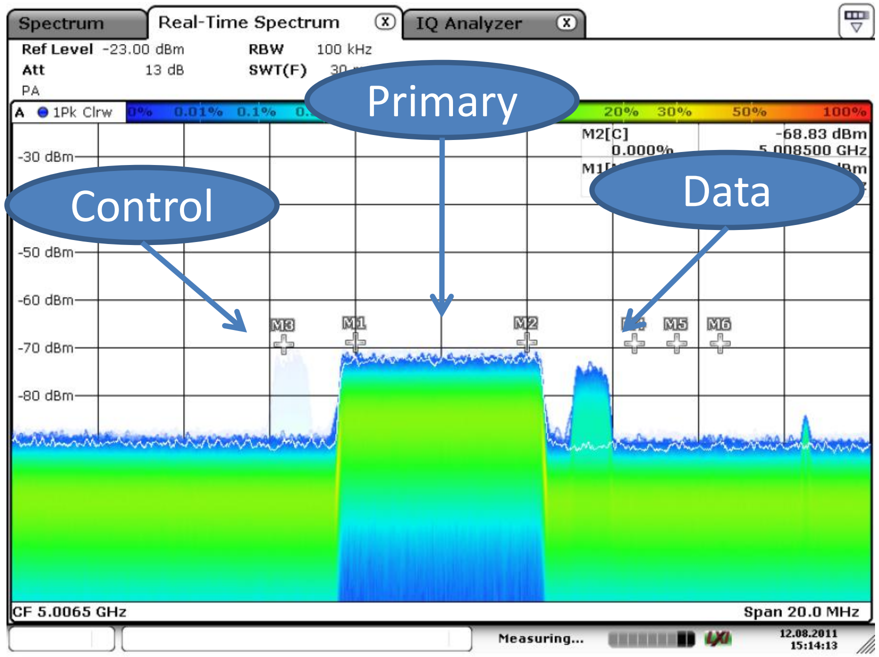
- Jacek Kibilda
- COST Short-Term Scientific Mission
- 2 weeks (no prior knowledge of Iris)
- DSA demo (primary user avoidance)

SRS Software Radio Systems

CTVR the telecommunications research centre

Iris Overview

What can I do with Iris?



Ref Level -23.00 dBm RBW 100 kHz

Att 13 dB SWT(F) 30

PA

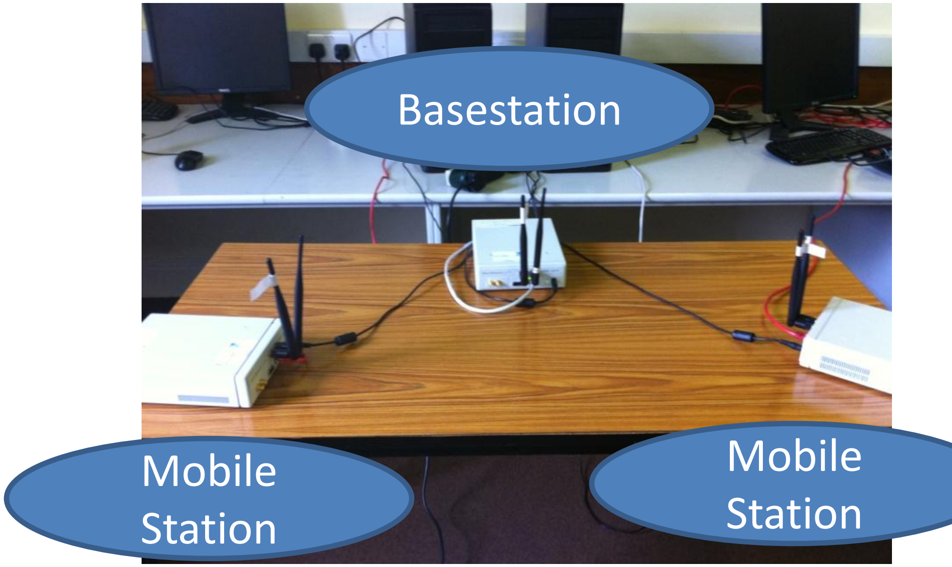
A 1Pk Clrw

M2[C] 0.000% -68.83 dBm 5.008500 GHz

CF 5.0065 GHz Span 20.0 MHz

12.08.2011 15:14:13

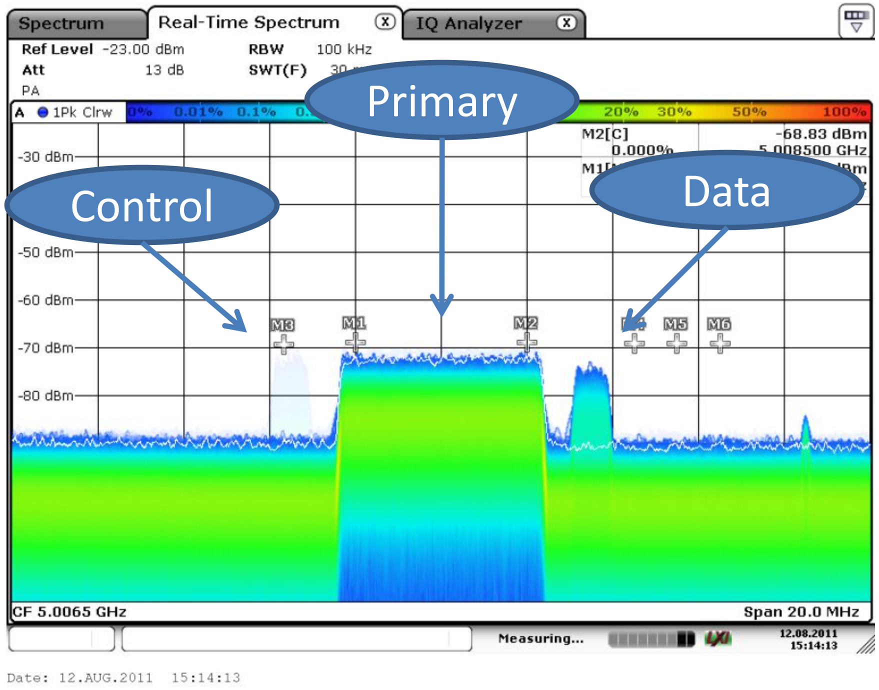
Date: 12.AUG.2011 15:14:13



- 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?



Spectrum | Real-Time Spectrum | IQ Analyzer

Ref Level -23.00 dBm RBW 100 kHz

Att 13 dB SWT(F) 30

PA

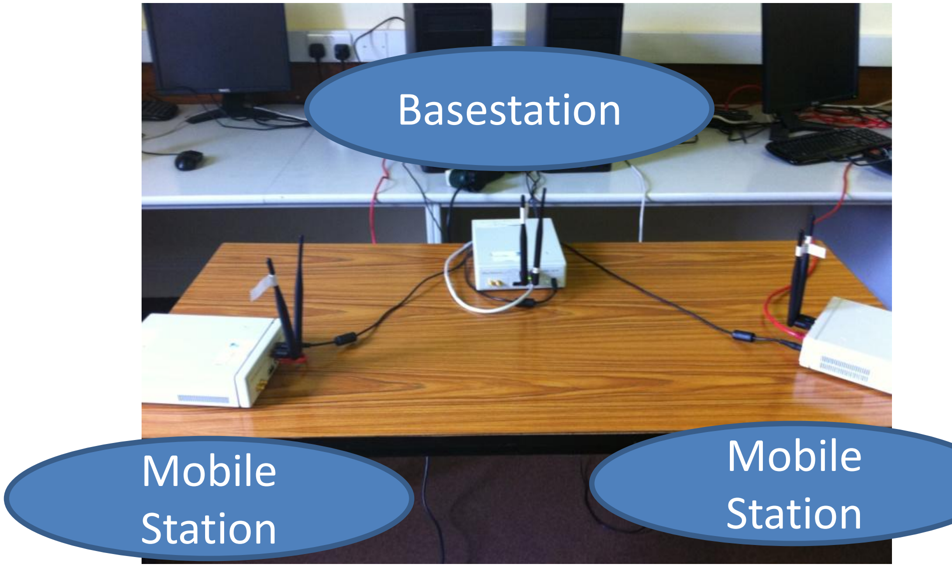
M2[C] 0.000% -68.83 dBm 5.008500 GHz

CF 5.0065 GHz Span 20.0 MHz

Measuring...

12.08.2011 15:14:13

Date: 12.AUG.2011 15:14:13



- 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/

Iris Architecture

- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- Case Study - OFDM
- Interesting Applications

Iris Architecture - The Basics

The Basics...

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

Iris Architecture - The Basics

The Basics...

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

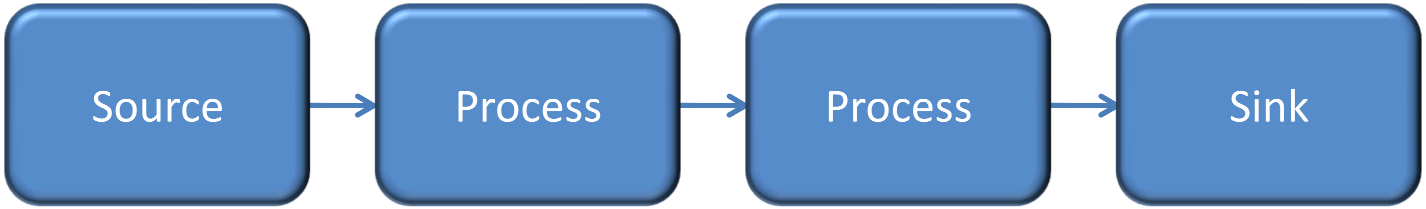
- Most basic configuration :
  - A source component
  - A sink component
  - Some processing components

Iris Architecture - The Basics

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

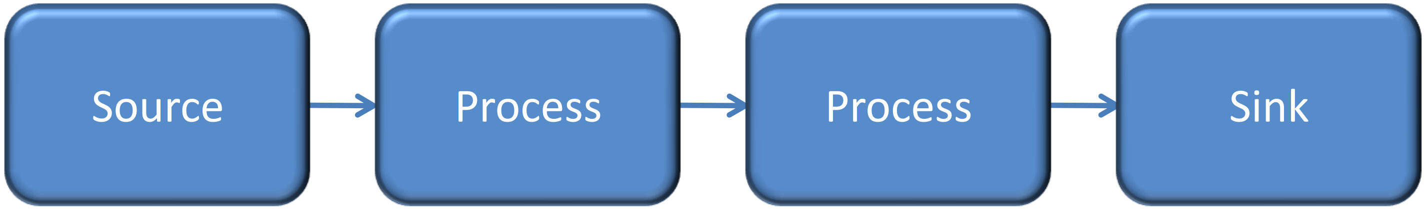


Iris Architecture - The Basics

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>

    </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

```
<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

```
<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

```
<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

```
<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

```
<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

```
<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

```
<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>
```

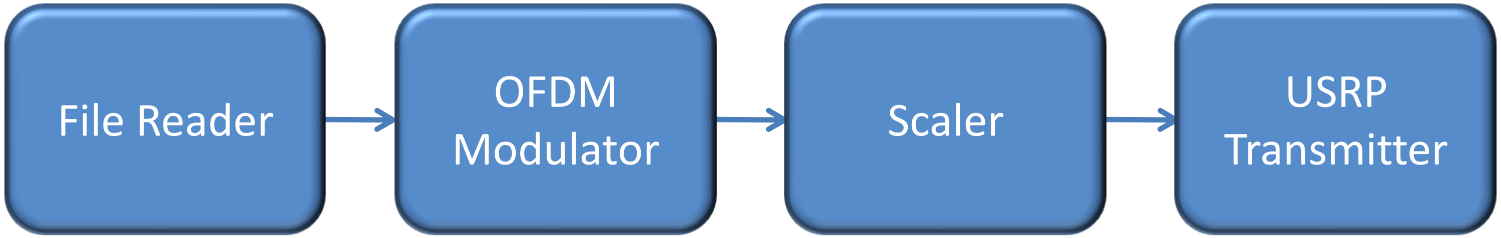
Iris Architecture - The Basics

```
<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>
```

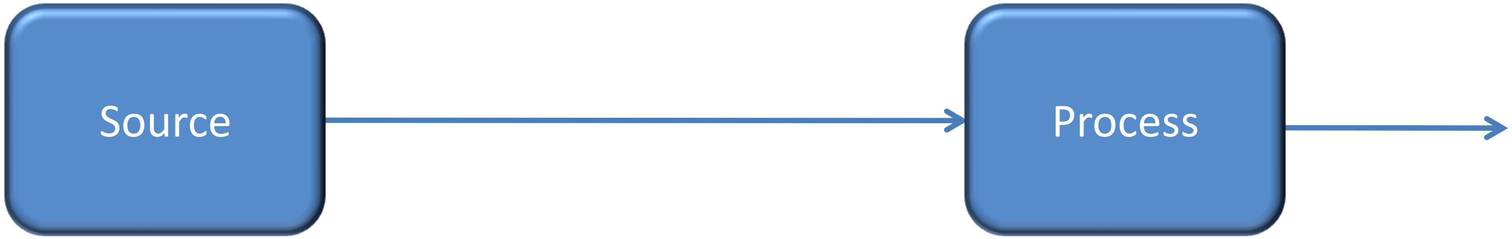
Iris Architecture - The Basics

```
<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>
```

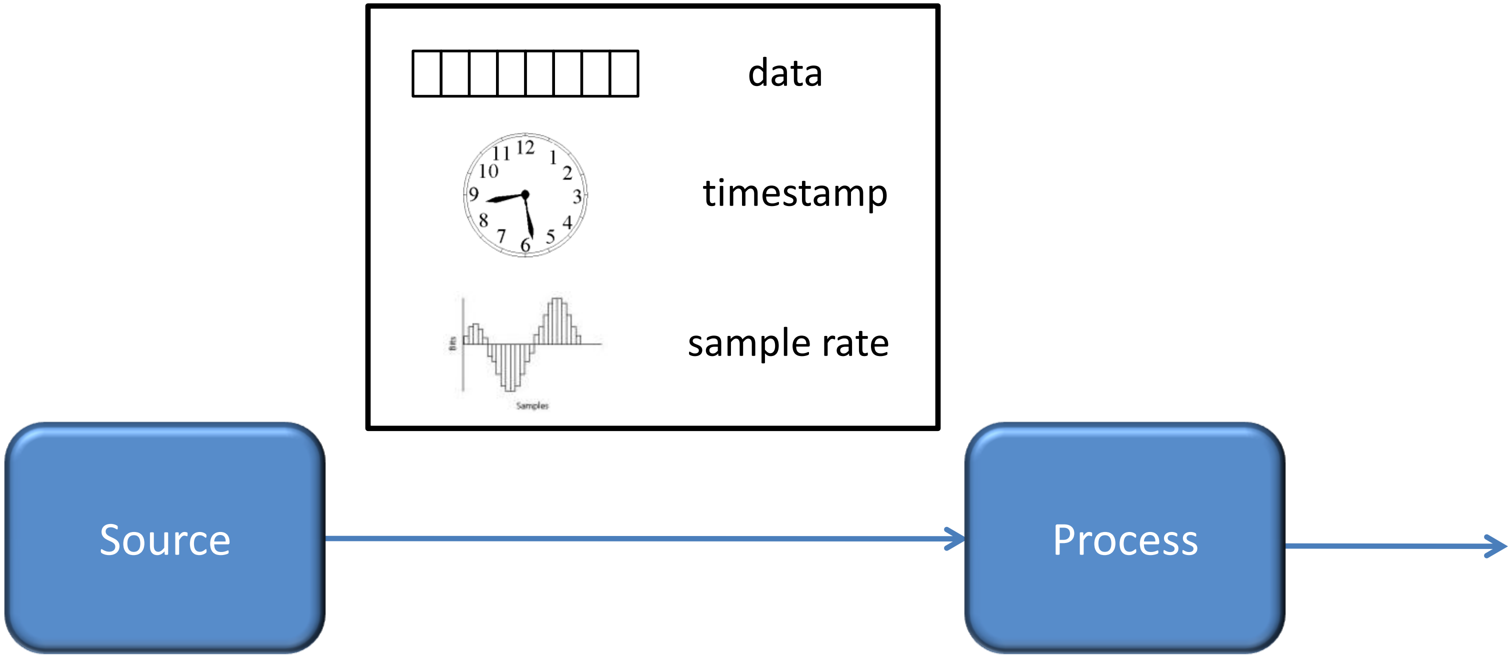
Iris Architecture - The Basics



Iris Architecture - The Basics

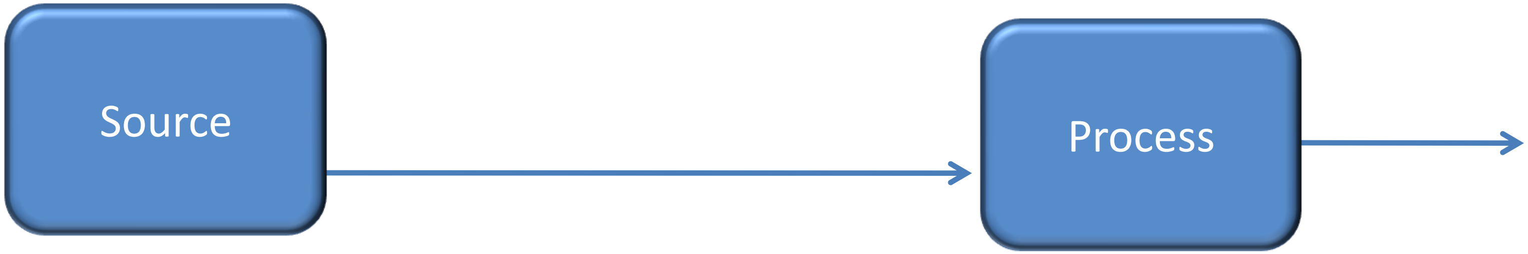


Iris Architecture - The Basics

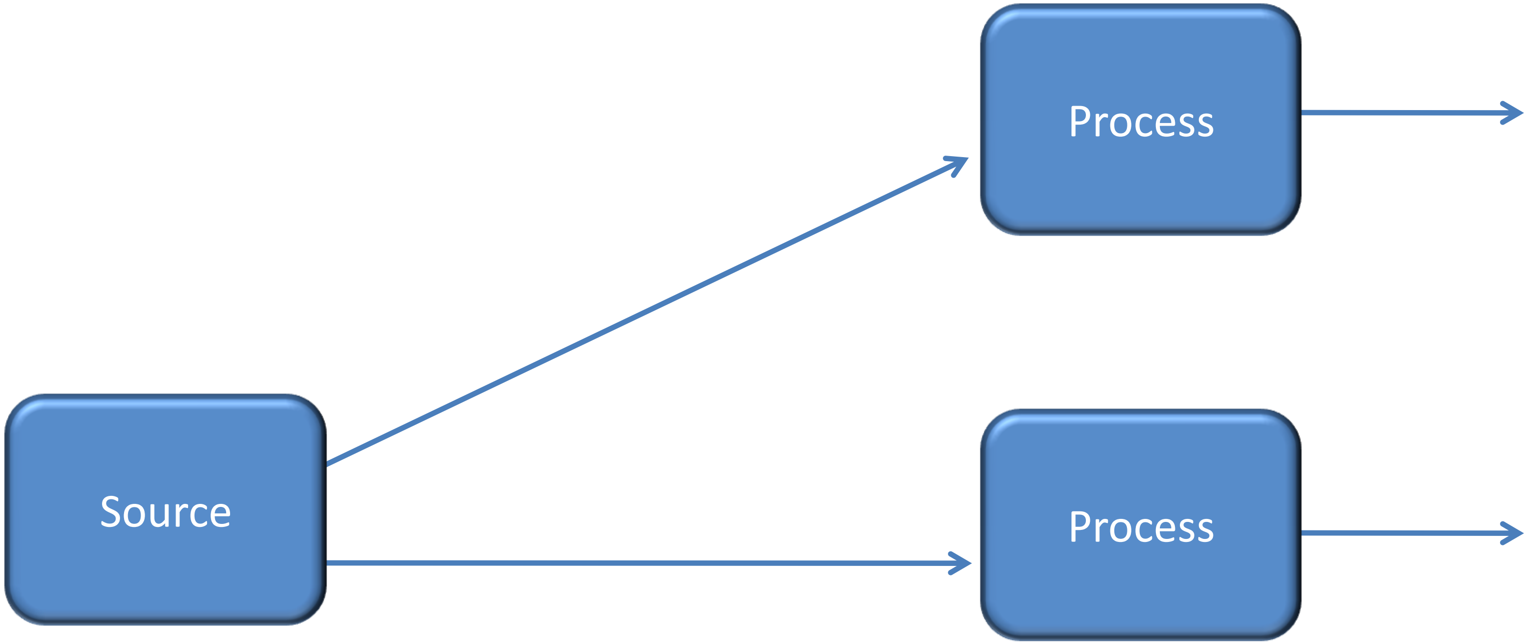


- Data is passed between components in blocks – a DataSet
- Vector of data samples
- Metadata – e.g. timestamp, sample rate

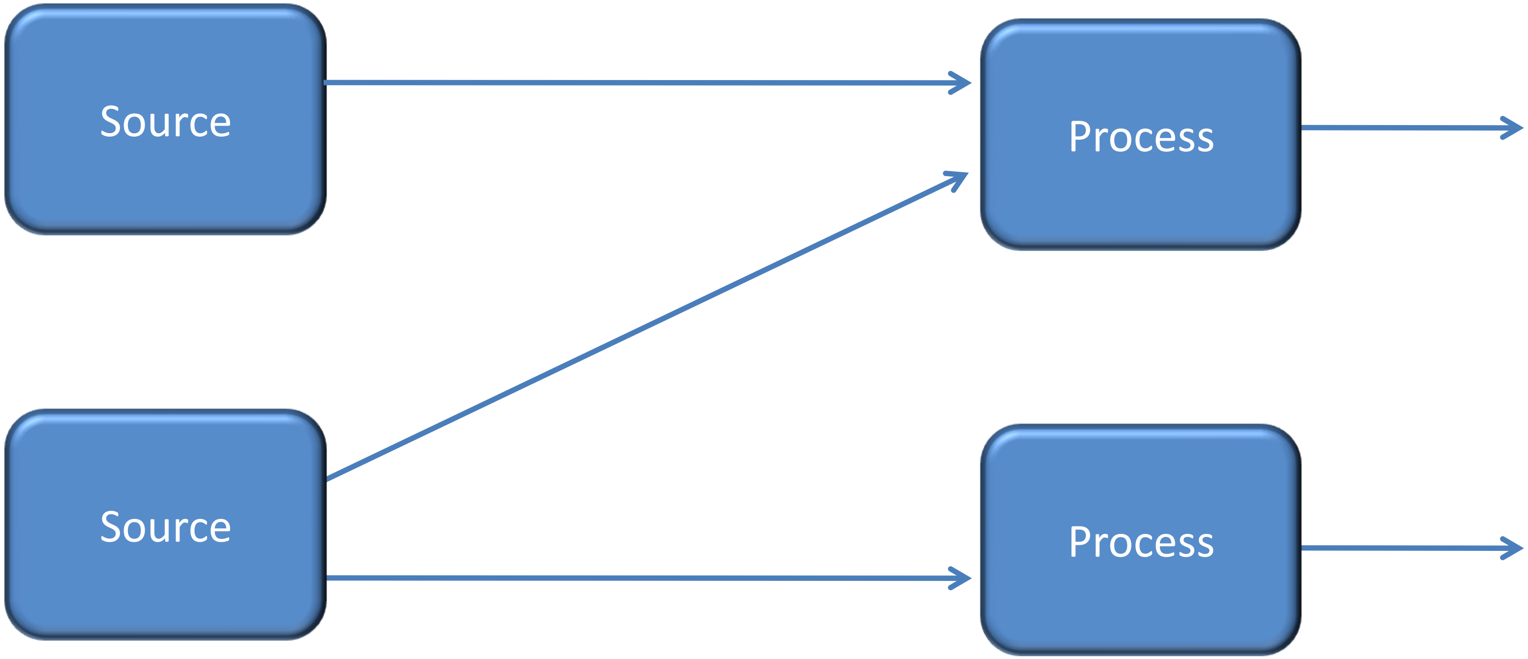
Iris Architecture - The Basics



Iris Architecture - The Basics

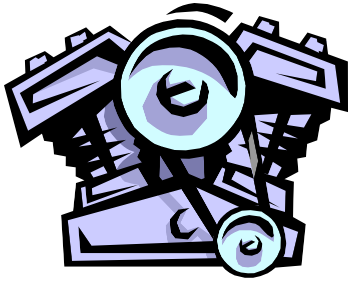


Iris Architecture - The Basics



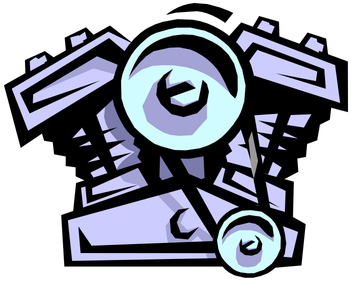
Iris Architecture - Engines

Engines



Iris Architecture - Engines

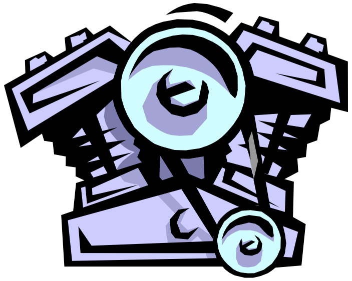
Engines



- 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

Iris Architecture - Engines

Engines

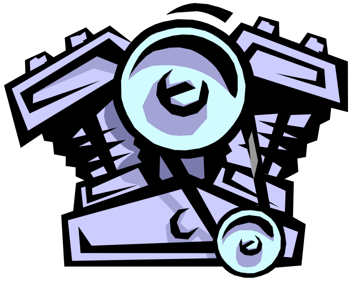


- An *engine*
  - The environment within which one more components operates
  - Defines its own data-flow mechanism
  - Defines its own reconfiguration mechanisms
  - Runs one or more of its own threads
  - Provides a clean interface for the Iris system

Executes a section of the flow graph

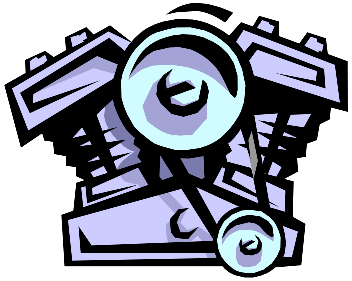
Completely up to the engine how that's done

Iris Architecture - Engines



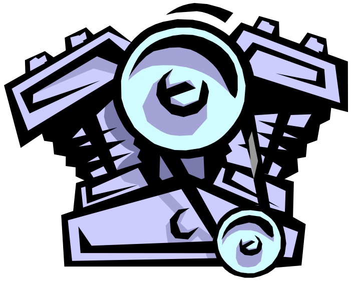
- Two engine types:
  - PHY Engine
  - Stack Engine

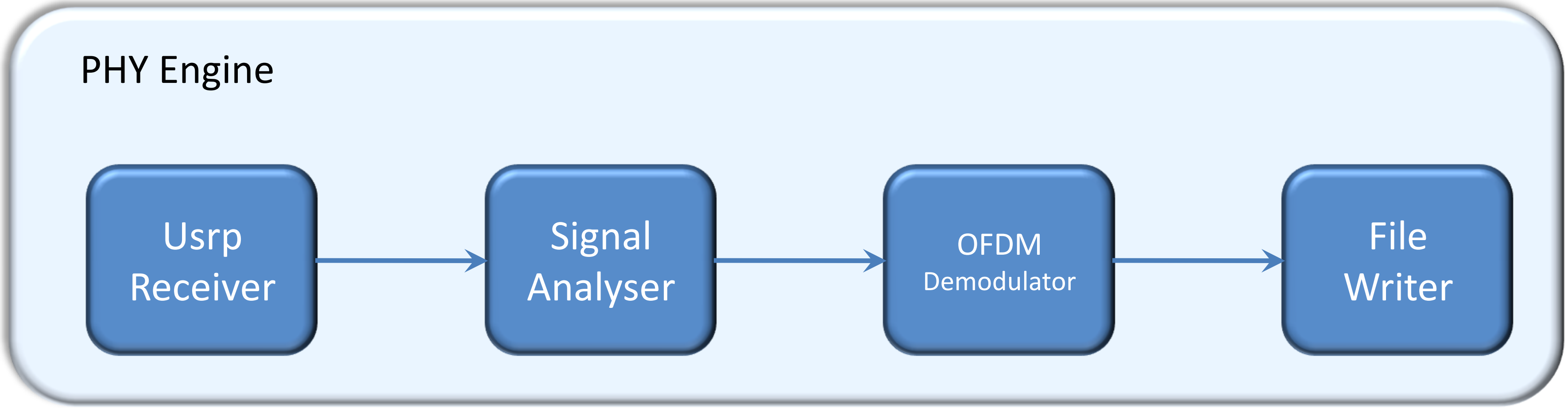
Iris Architecture - Engines



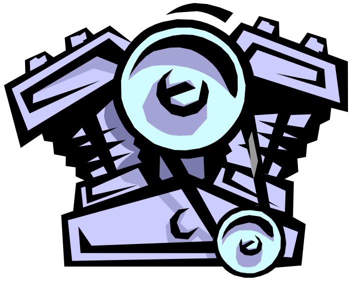
- PHY Engine
  - Maximum flexibility
  - One thread per engine
  - Data-driven execution
  - One or more components per engine
  - Multiple component inputs / outputs
  - Unidirectional data flow
  - No fixed relationship between the inputs and outputs of a component
  - Flexible blocksizes

Iris Architecture - Engines



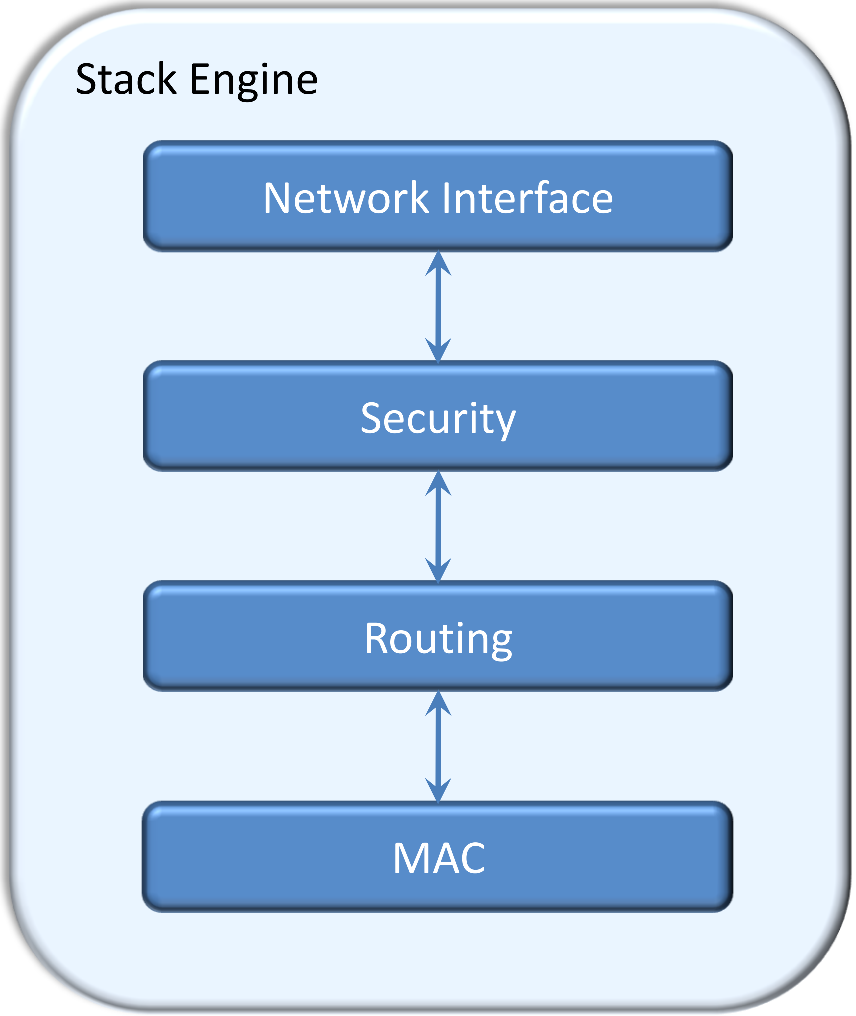


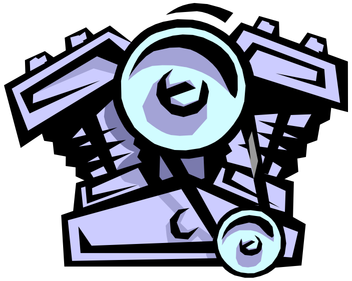
Iris Architecture - Engines



- 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

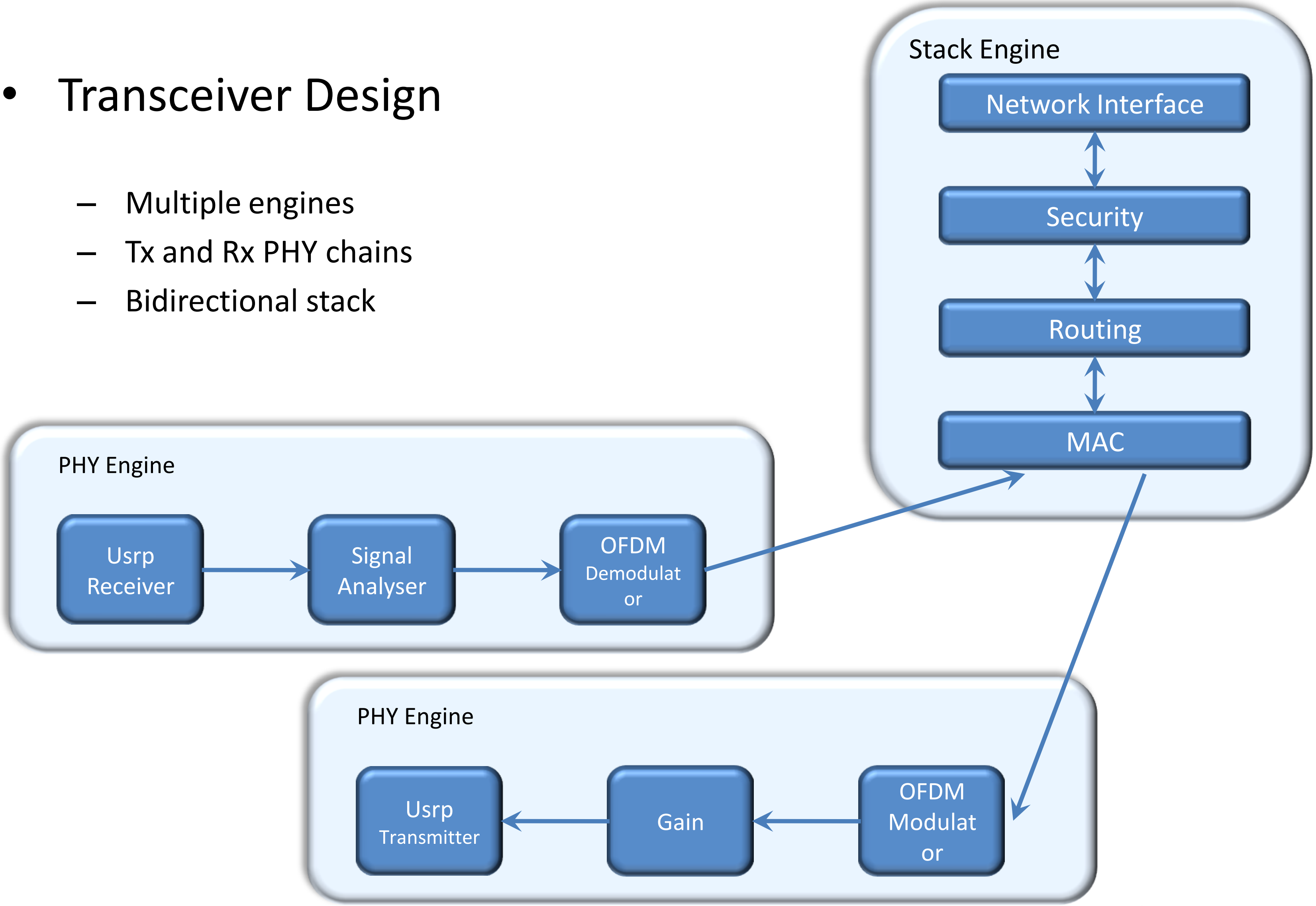
Iris Architecture - Engines





Iris Architecture - Engines

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



Software Radio Systems

CTVR the telecommunications research centre

Iris Architecture - PHY Components

A PHY Component

Iris Architecture - PHY Components

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

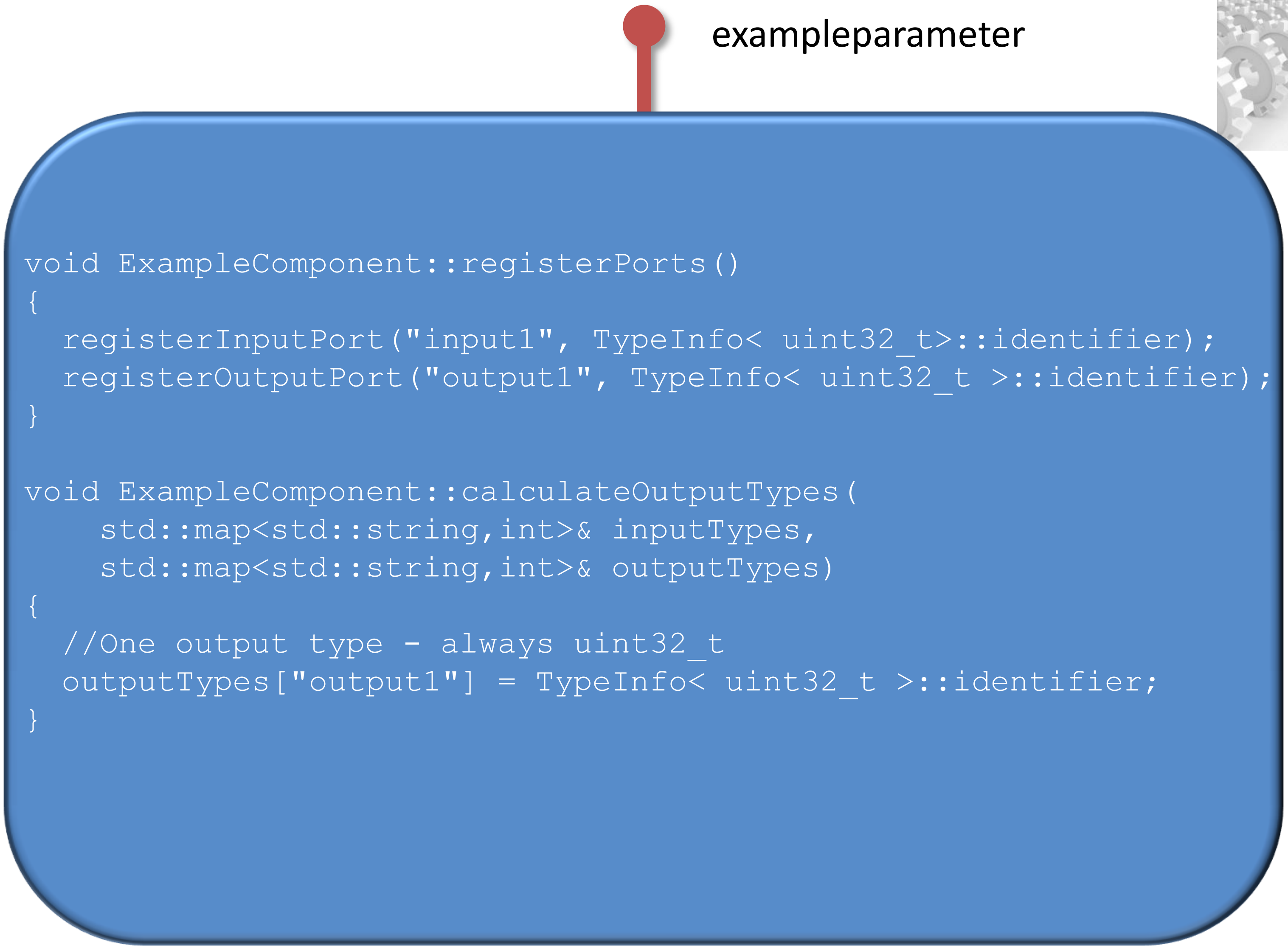
```
ExampleComponent::ExampleComponent(std::string name)
  : PhyComponent(name,                        // component name
                "example",                    // component type
                "An example phy component", // description
                "Paul Sutton",                // author
                "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
}
```

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
}
```

Iris Architecture - PHY Components



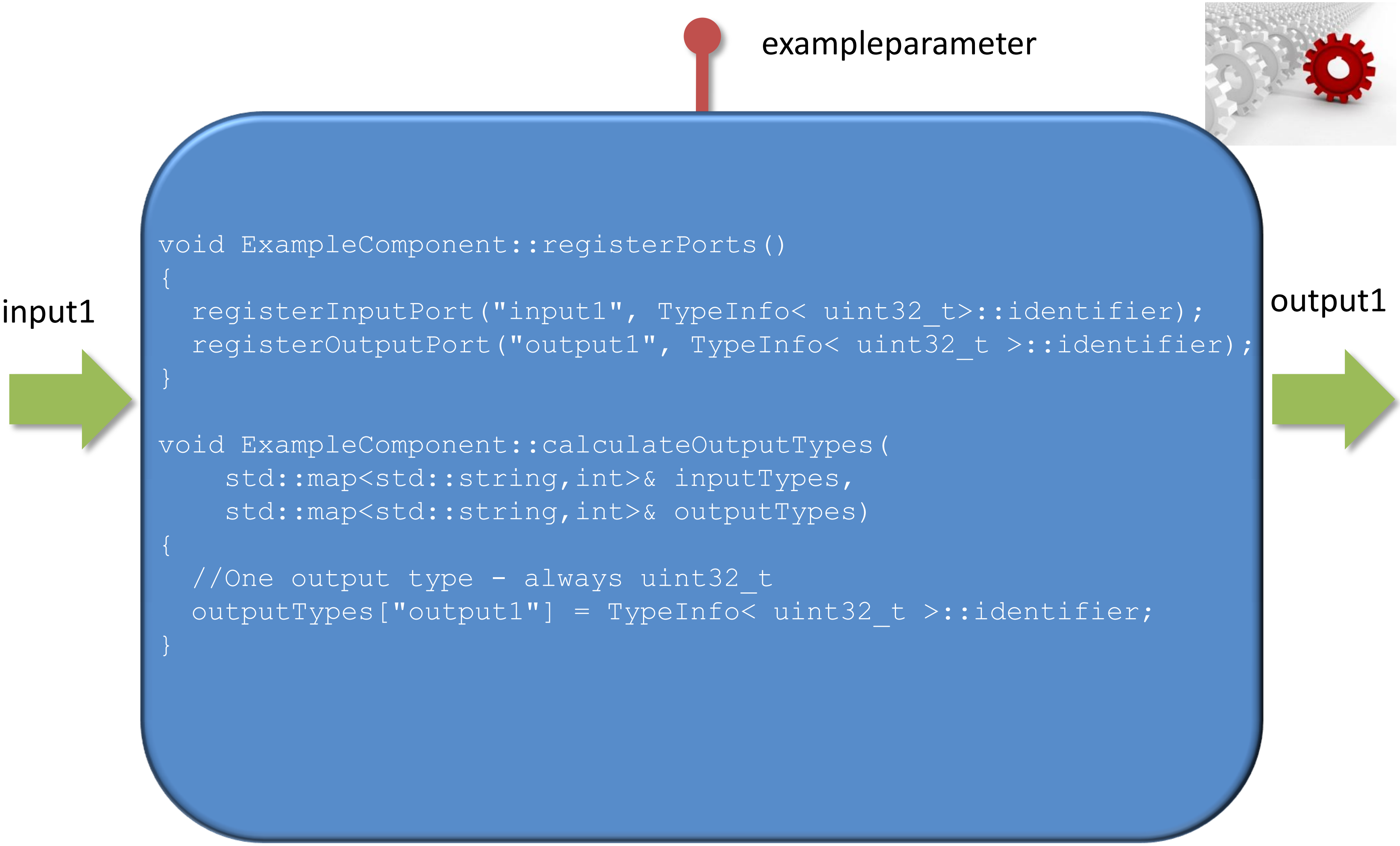
exampleparameter

```
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;
}
```

Software Radio Systems

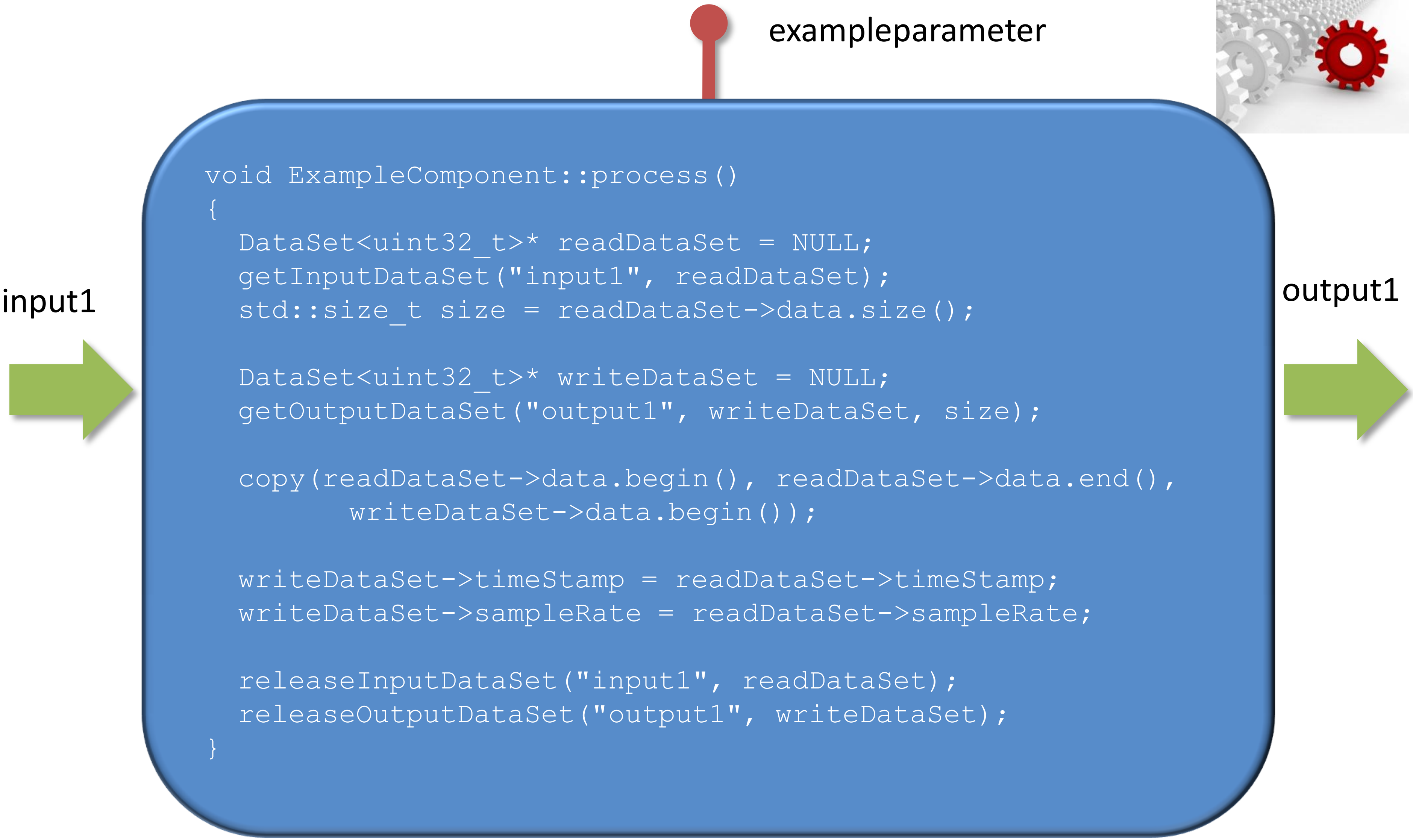
CTVR the telecommunications research centre

Iris Architecture - PHY Components



```
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;
}
```

Iris Architecture - PHY Components



```
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);
}
```

Iris Architecture - Stack Components

A Stack Component

Iris Architecture - Stack Components

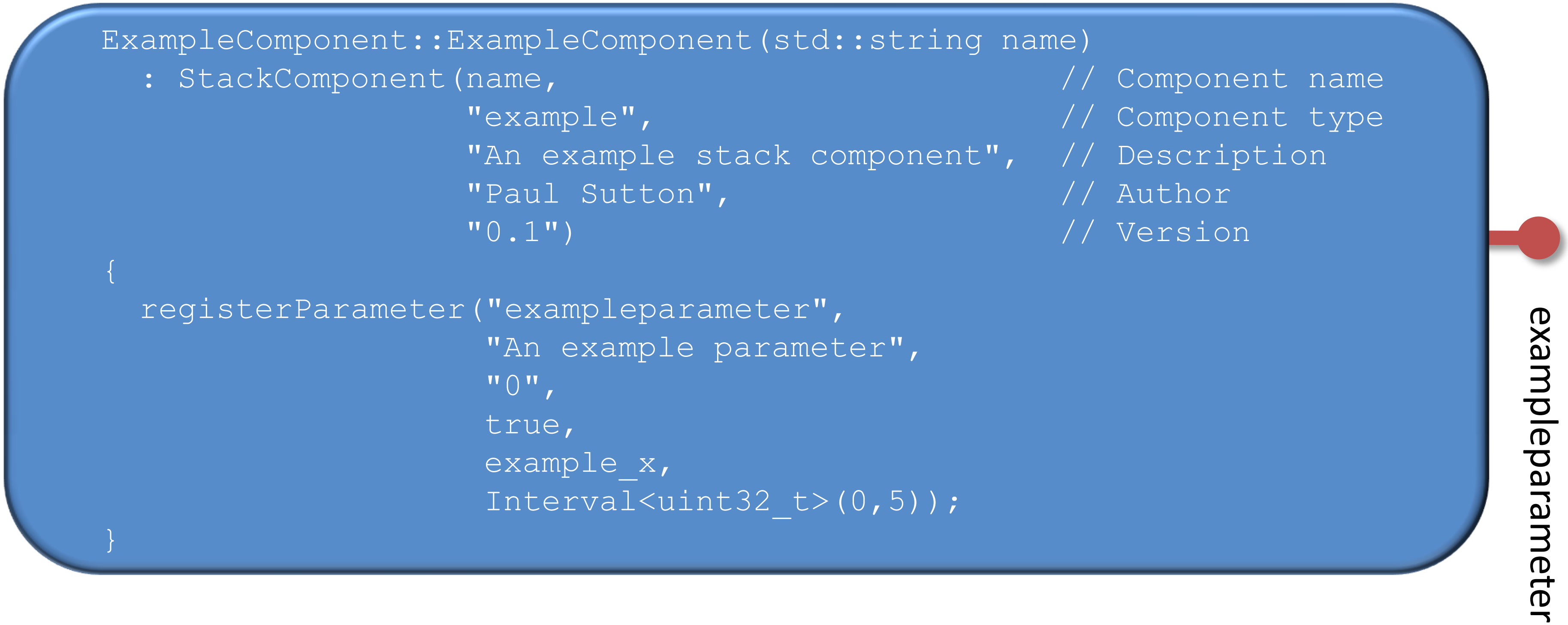
A Stack Component

```
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

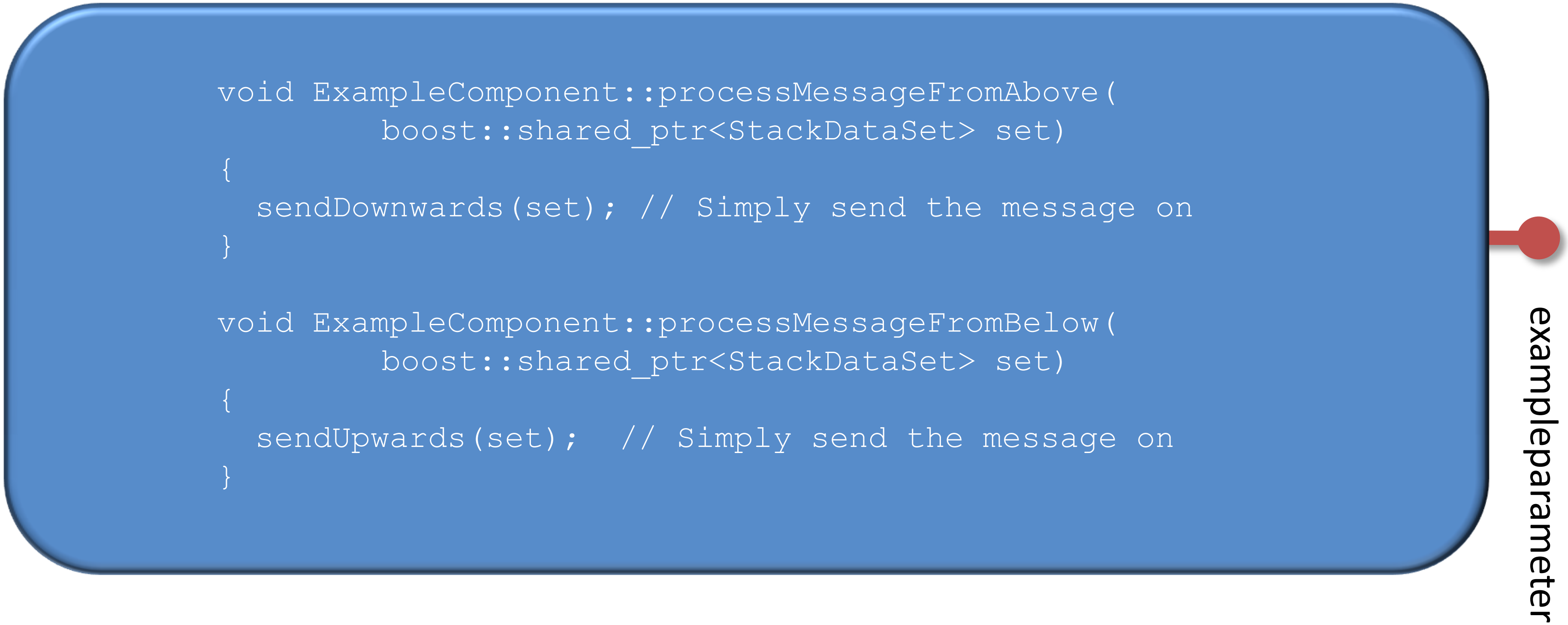
```
ExampleComponent::ExampleComponent(std::string name)
  : StackComponent(name,                          // Component name
                   "example",                     // Component type
                   "An example stack component",  // Description
                   "Paul Sutton",         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

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));
}
```

Iris Architecture - Stack Components

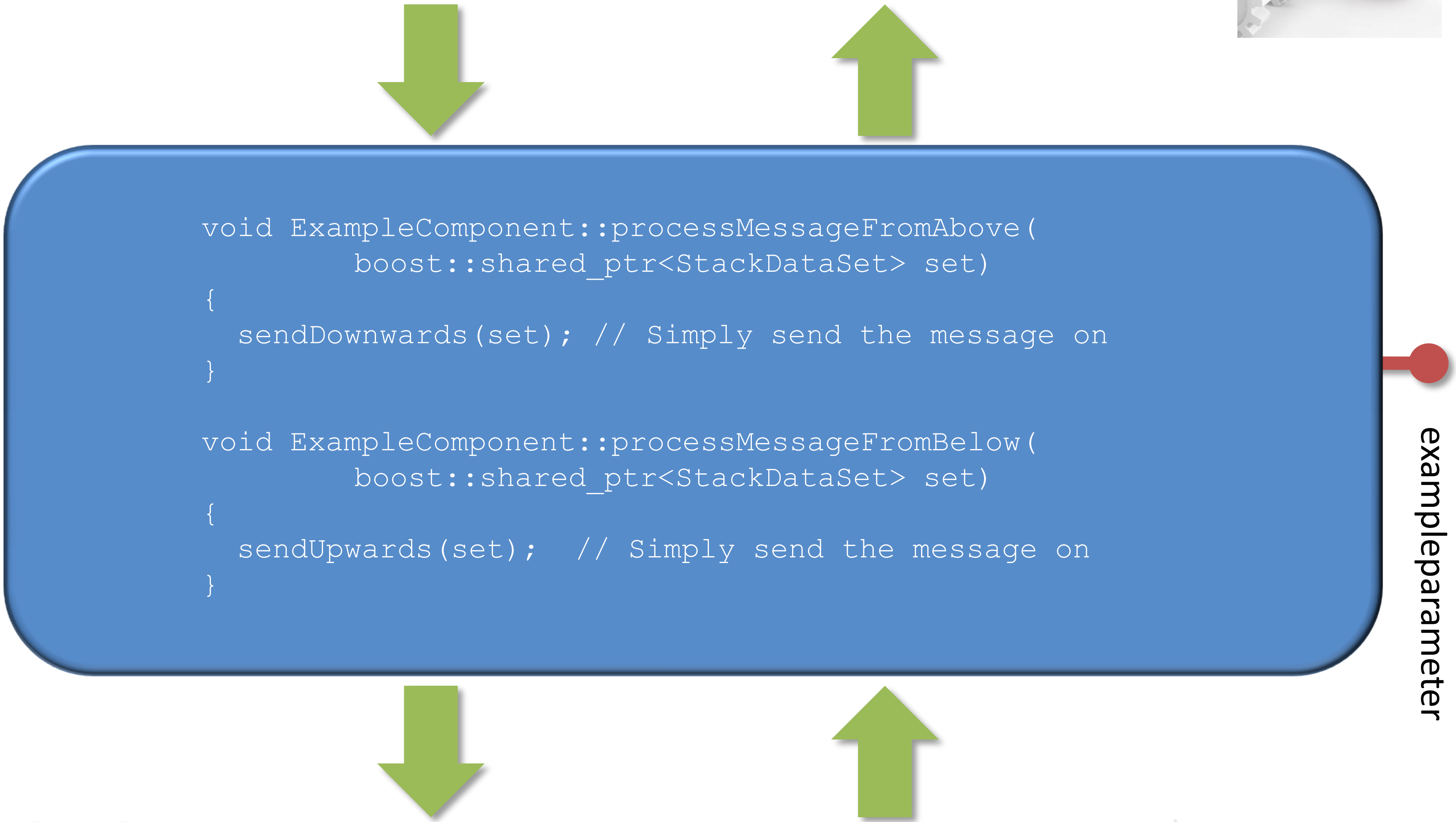


```
void ExampleComponent::processMessageFromAbove(
        boost::shared_ptr<StackDataSet> set)
{
  sendDownwards(set); // Simply send the message on
}

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

Software Radio Systems

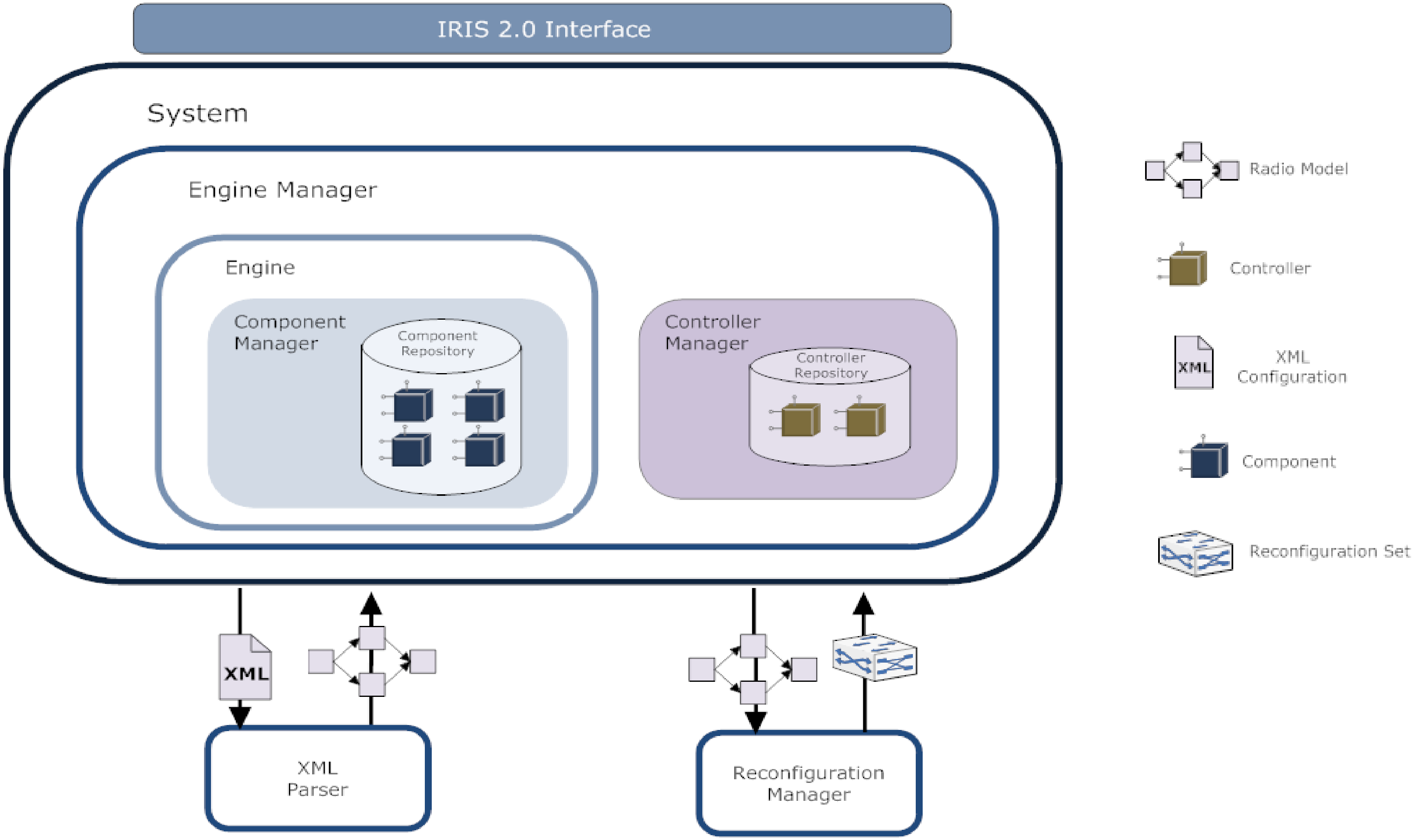
CTVR the telecommunications research centre

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
}
```

Iris Architecture - Core



SRS Software Radio Systems

CTVR the telecommunications research centre

Getting Started

- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- Case Study - OFDM
- Interesting Applications

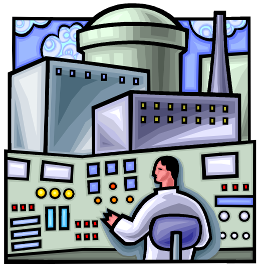
Getting Started

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

Controllers

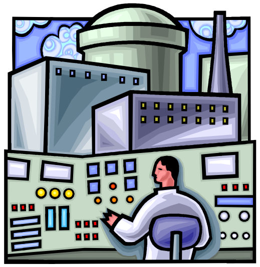
- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- Case Study - OFDM
- Interesting Applications

Controllers



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

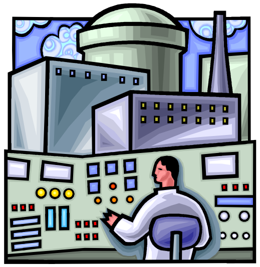
Controllers

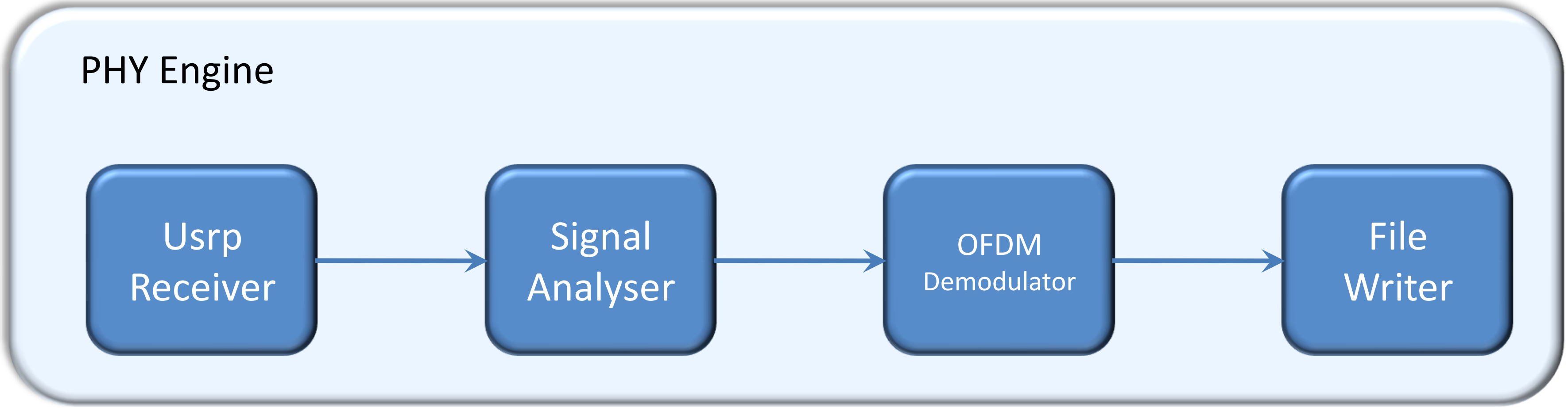


- So far...
  - We can create a radio
  - and reconfigure it manually

- How to reconfigure dynamically?

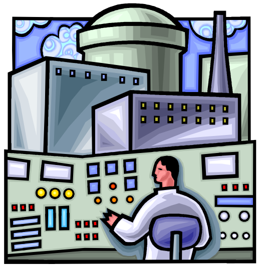
Controllers

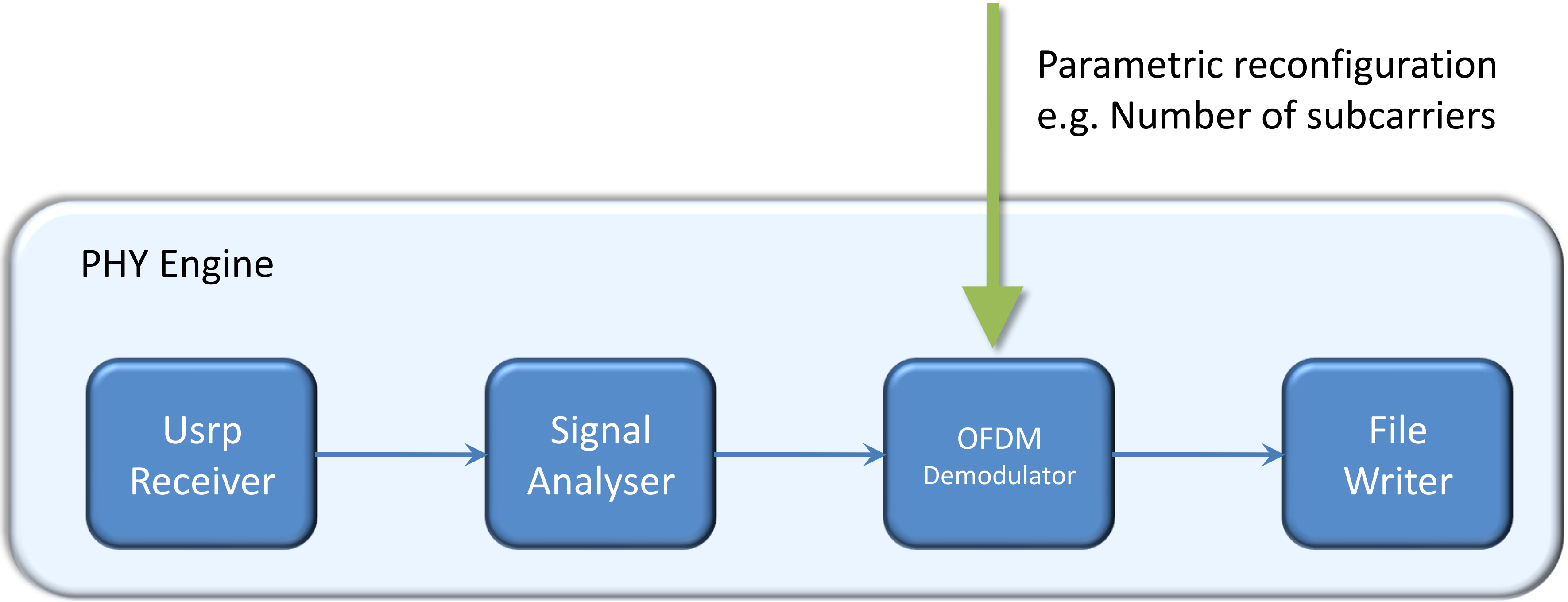




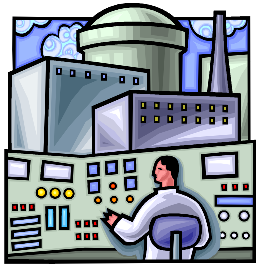
Controllers

- Parameters

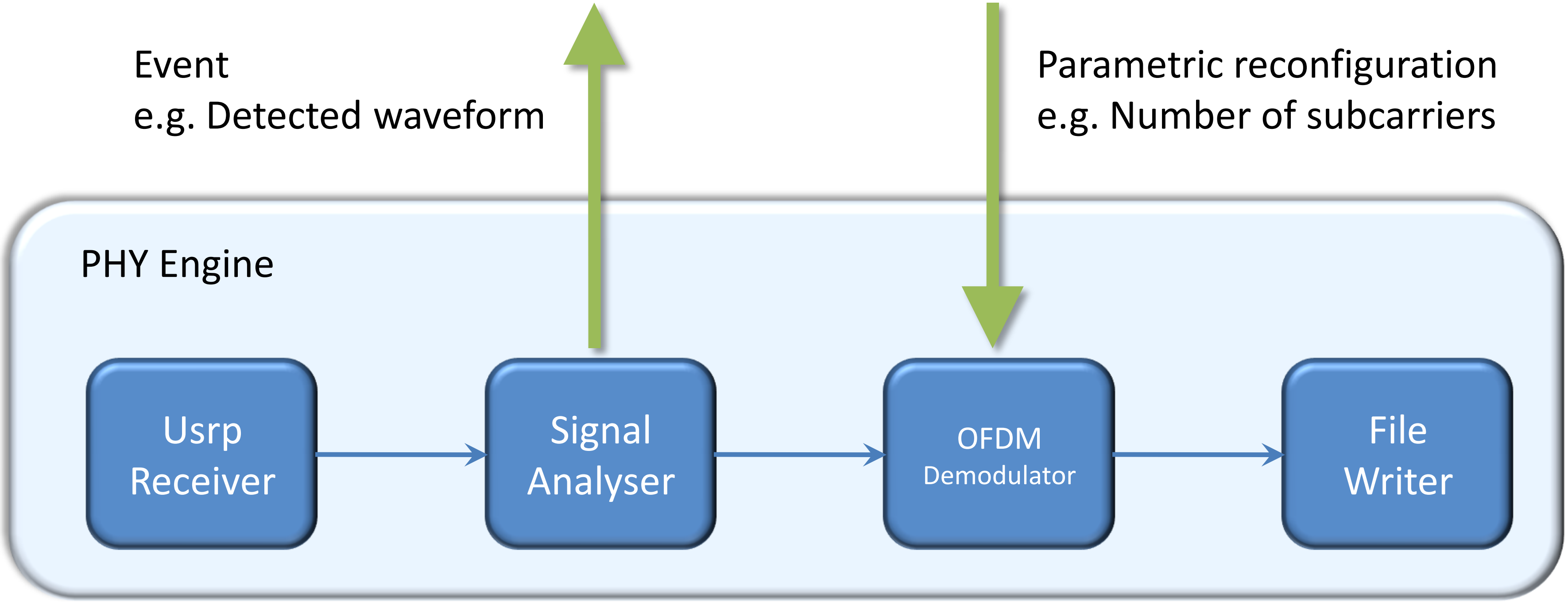




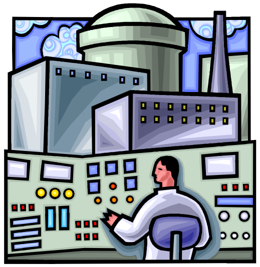
Controllers

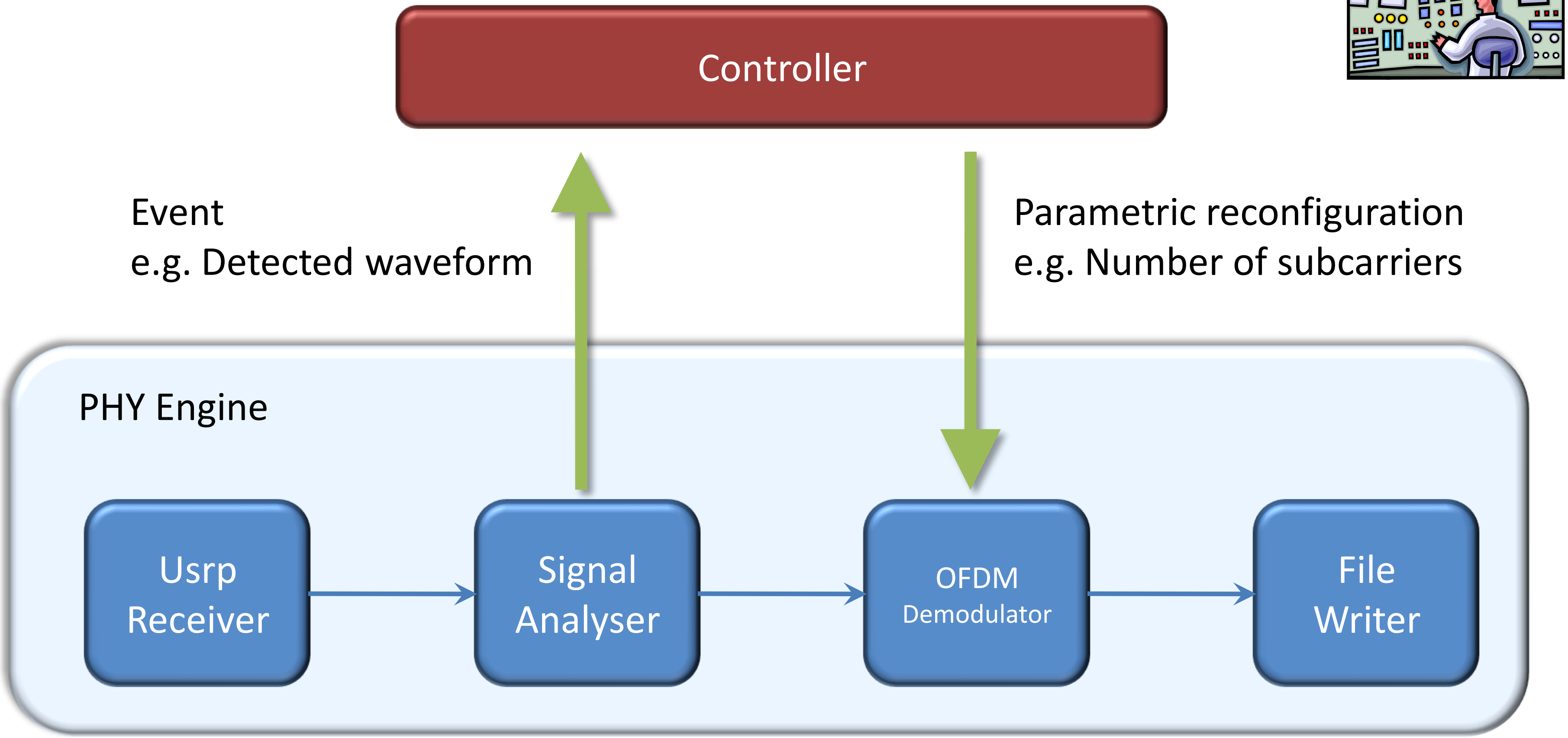


- Events

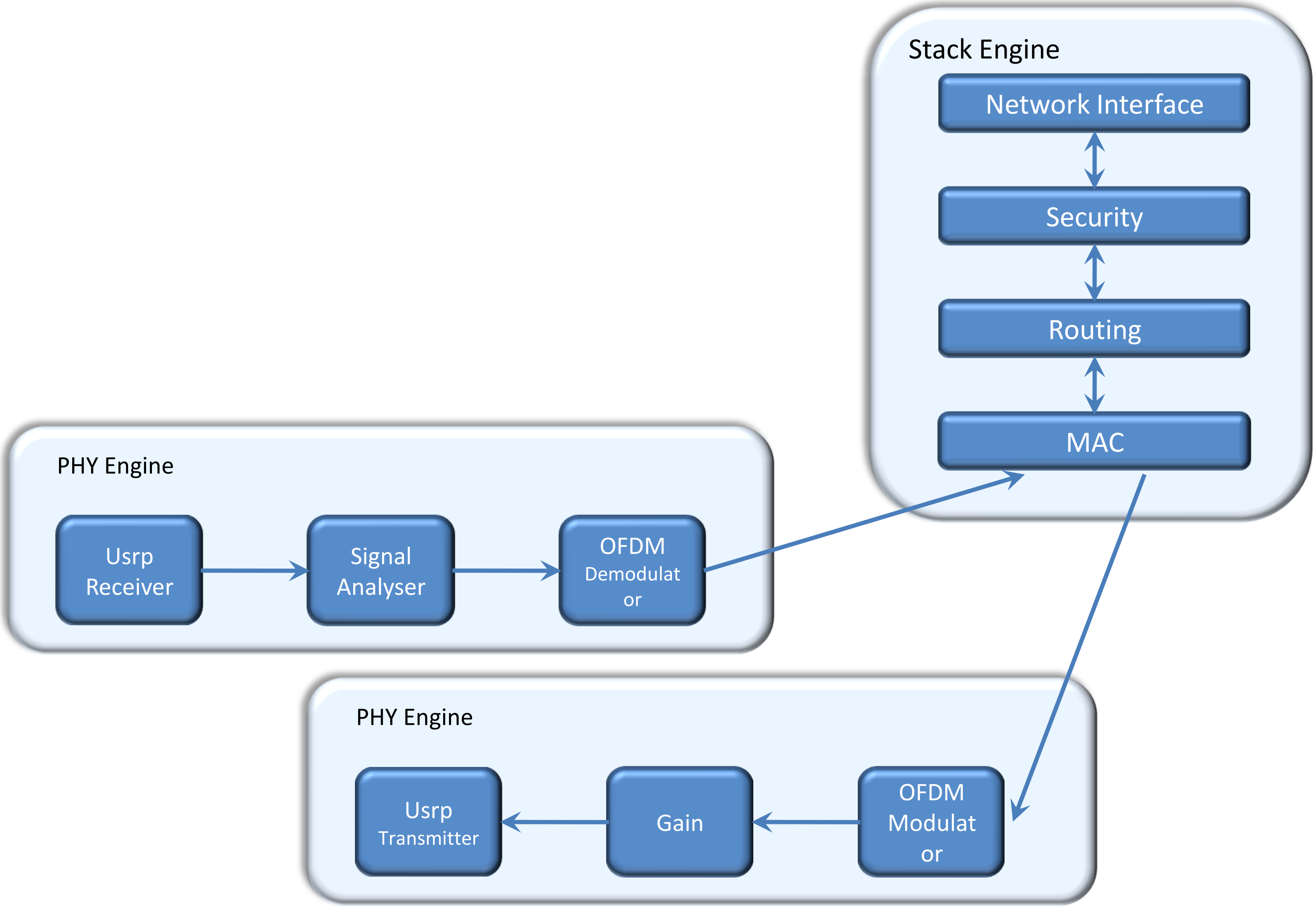


Controllers

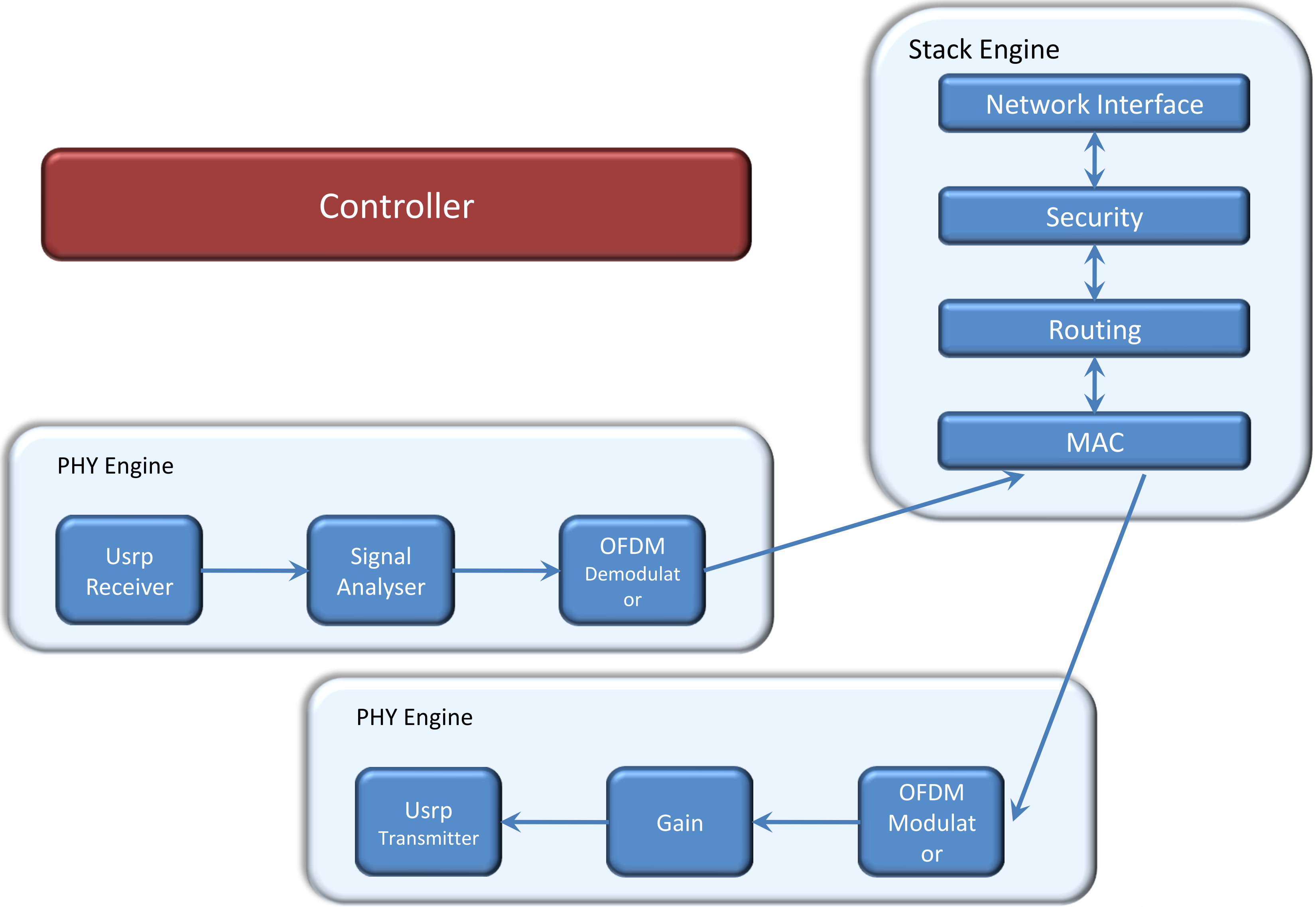




Controllers



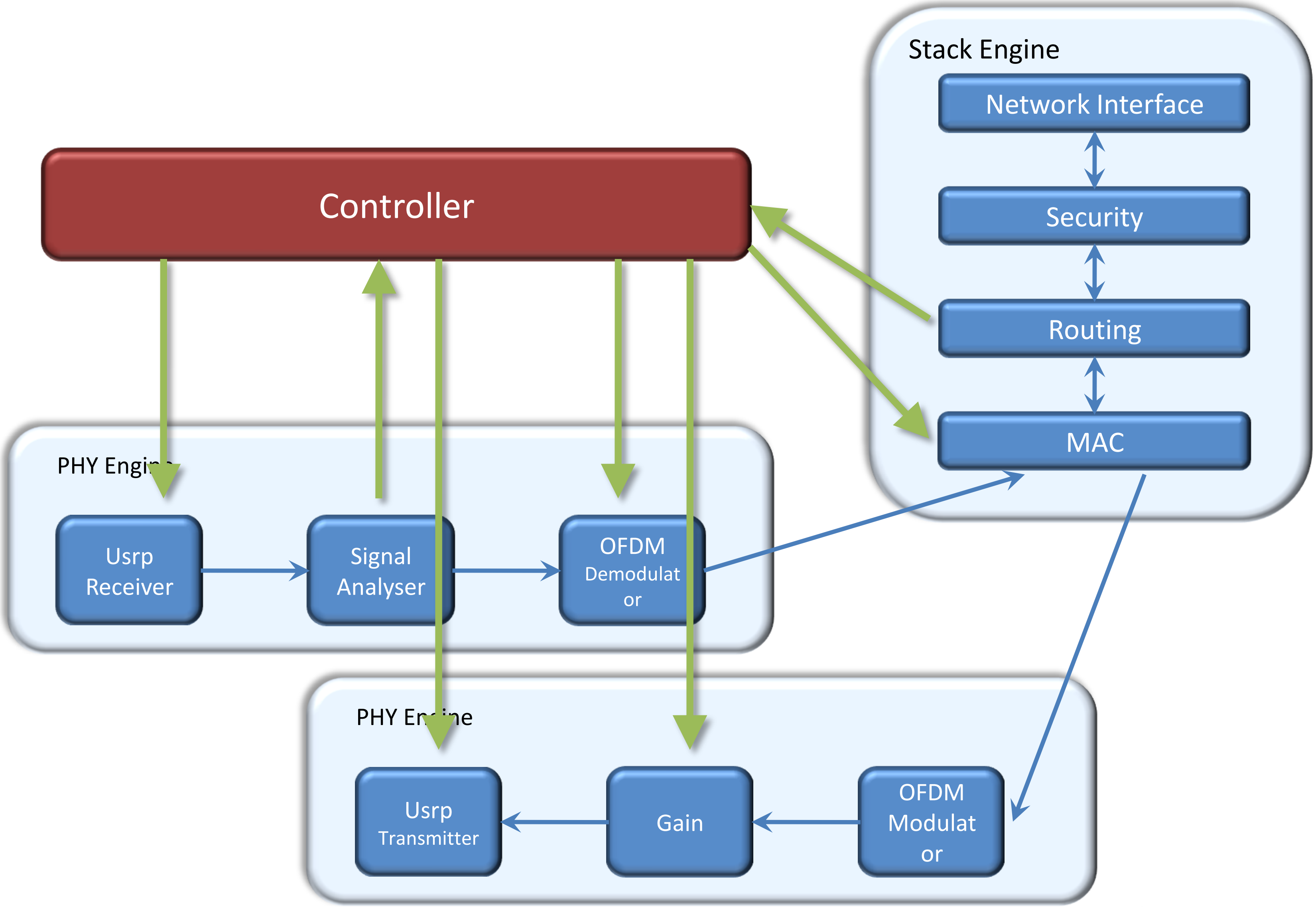
Controllers



SRS Software Radio Systems

CTVR the telecommunications research centre

Controllers



SRS Software Radio Systems

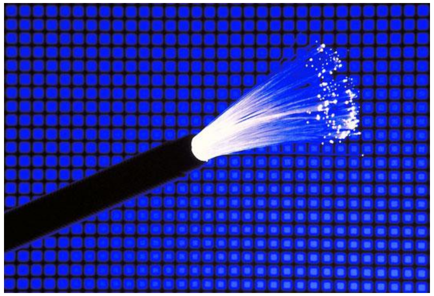
CTVR the telecommunications research centre

OFDM Specifics

- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- Case Study - OFDM
- Interesting Applications

Background

DAB digital radio



DSL

WiFi™

wimax

Lte™

DVB®
Digital Video Broadcasting

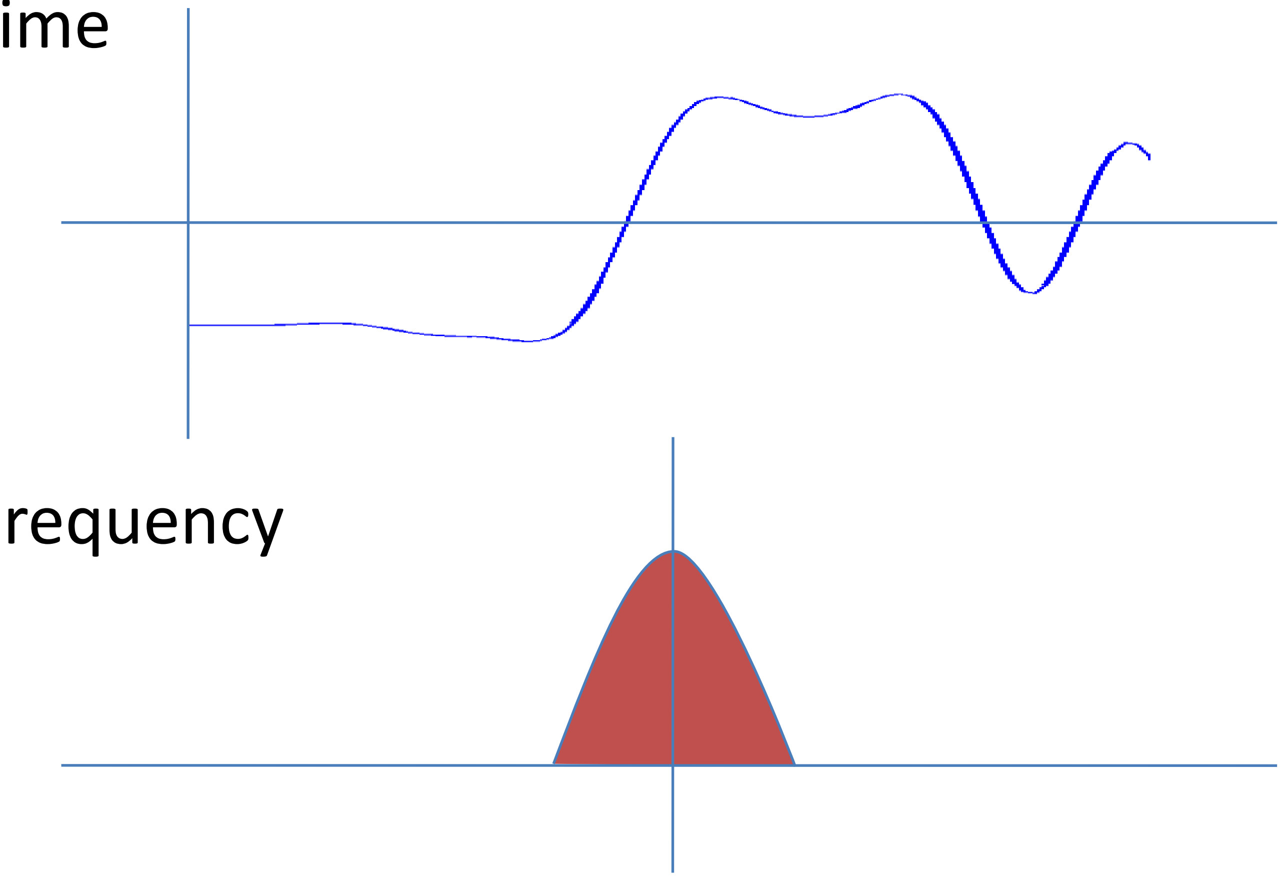
Powerline Communications

Background

Single-Carrier Systems

Background

Single-Carrier Systems



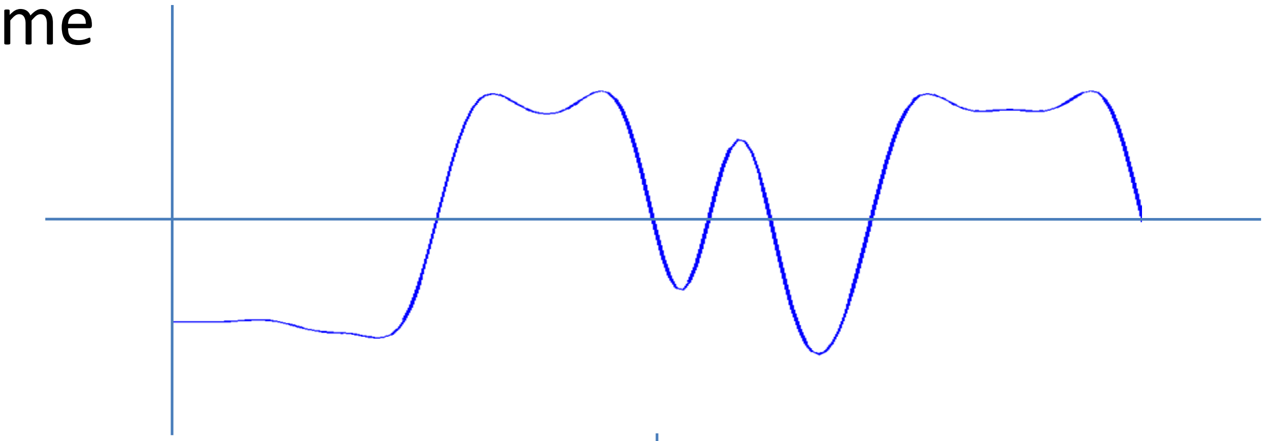
Time

Frequency

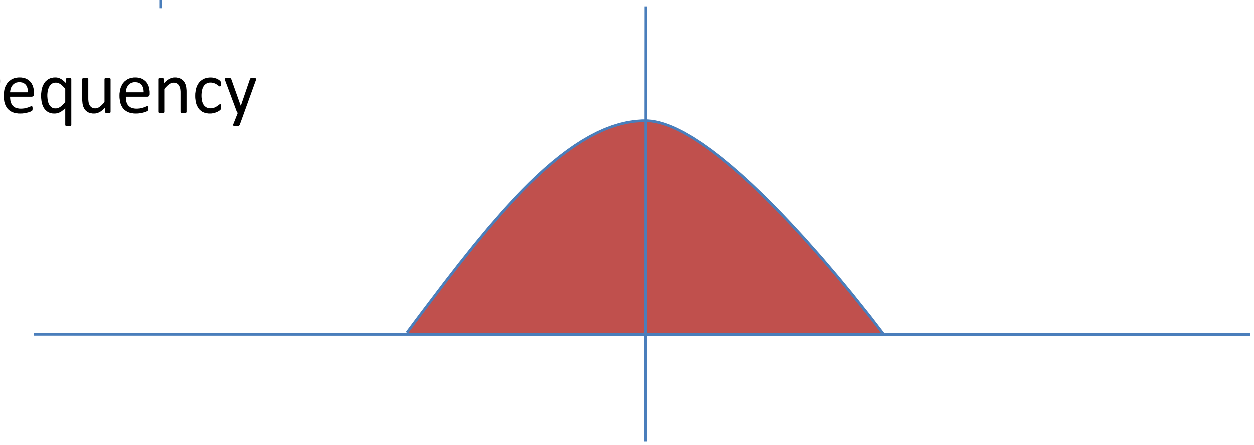
Background

Single-Carrier Systems

Time

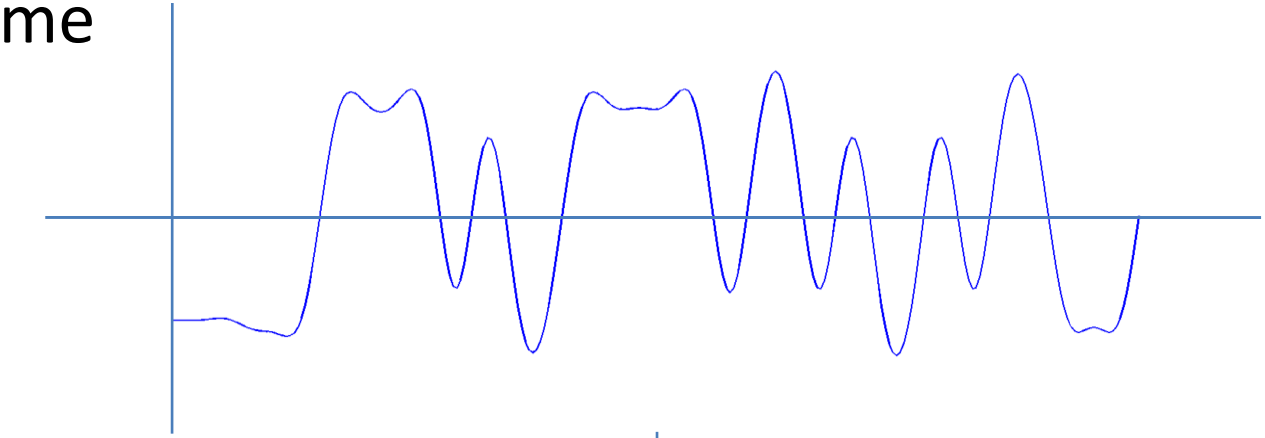


Frequency

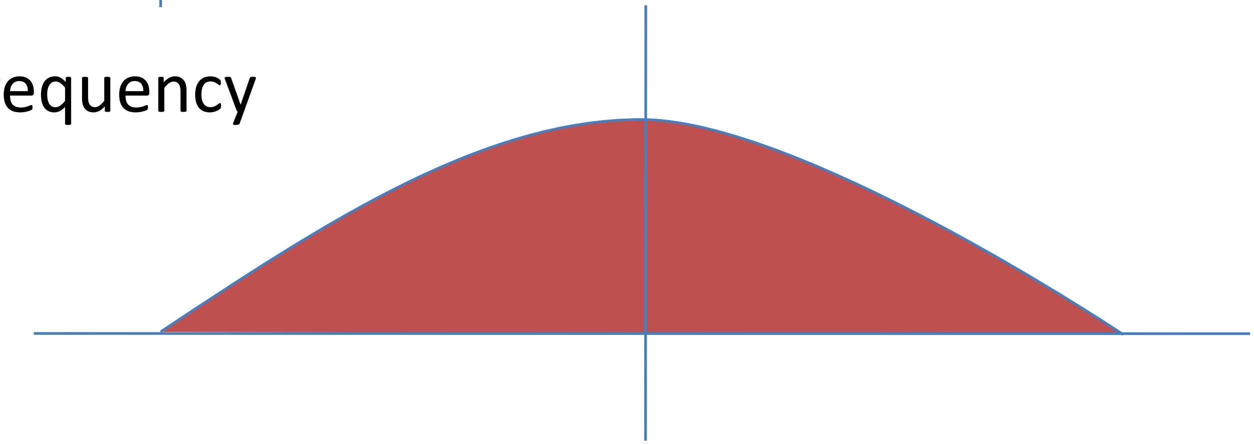


Single-Carrier Systems

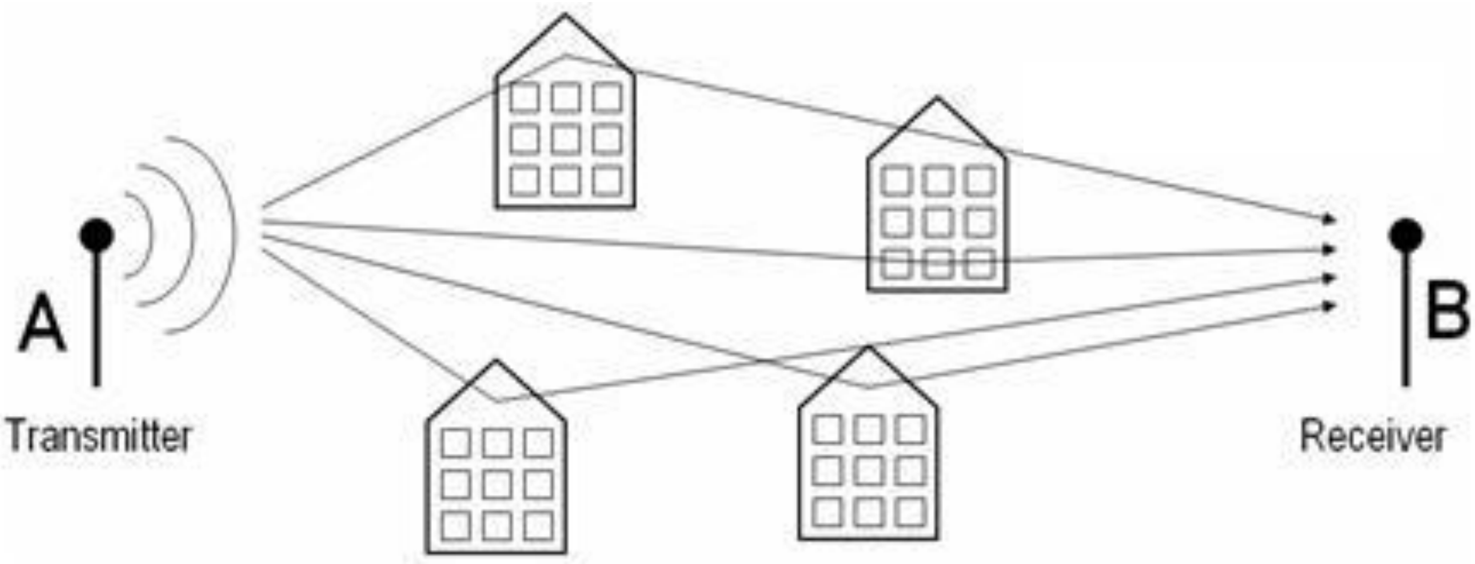
Time



Frequency

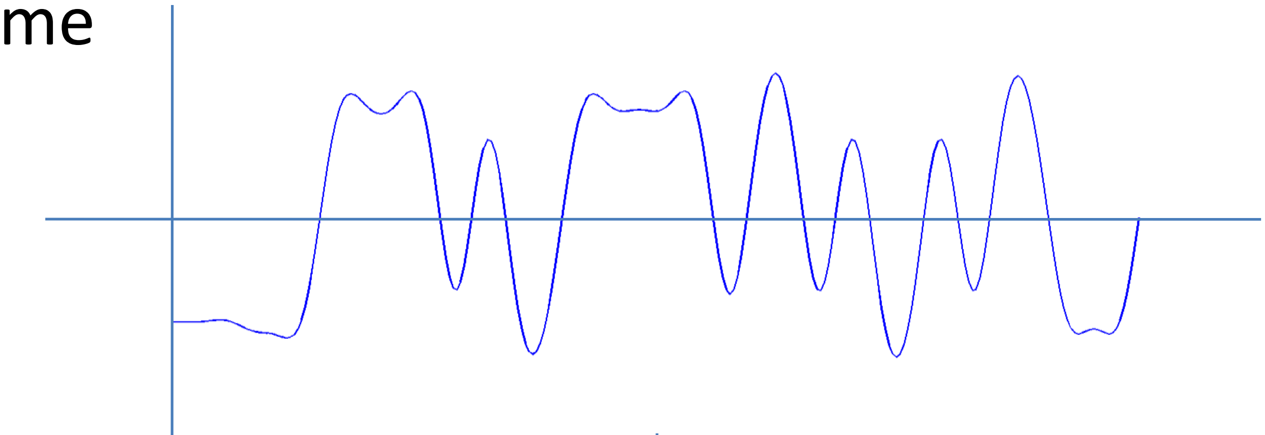


Multipath Propogation

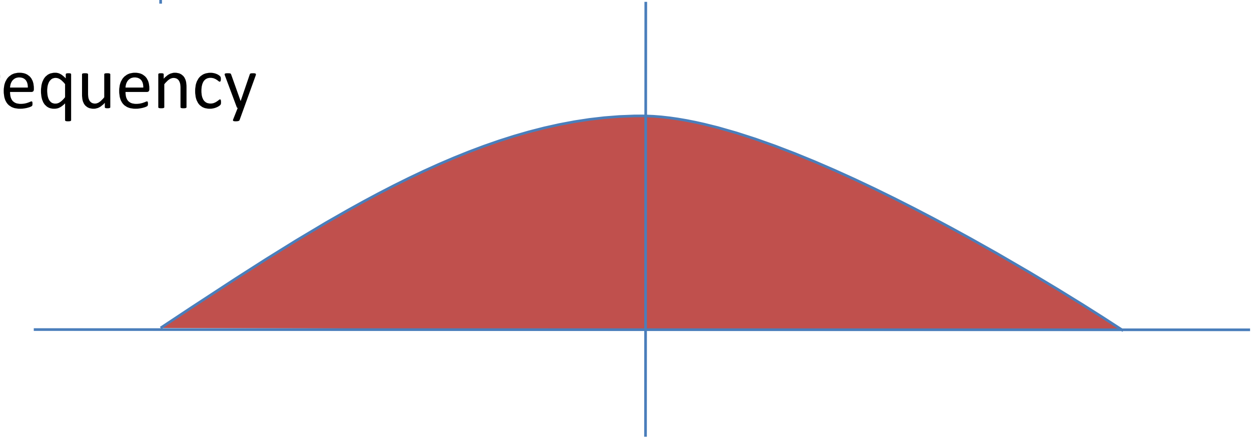


Background

Time

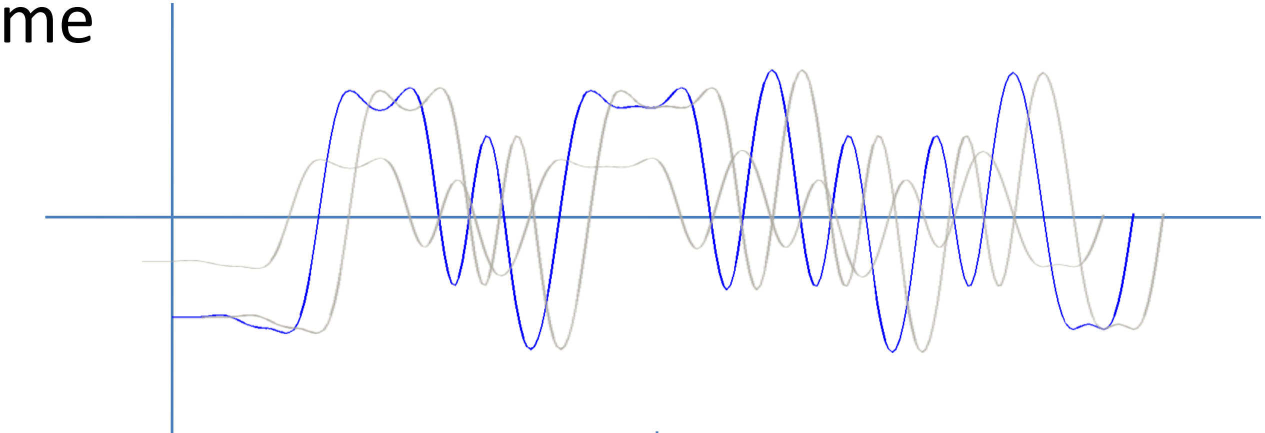


Frequency

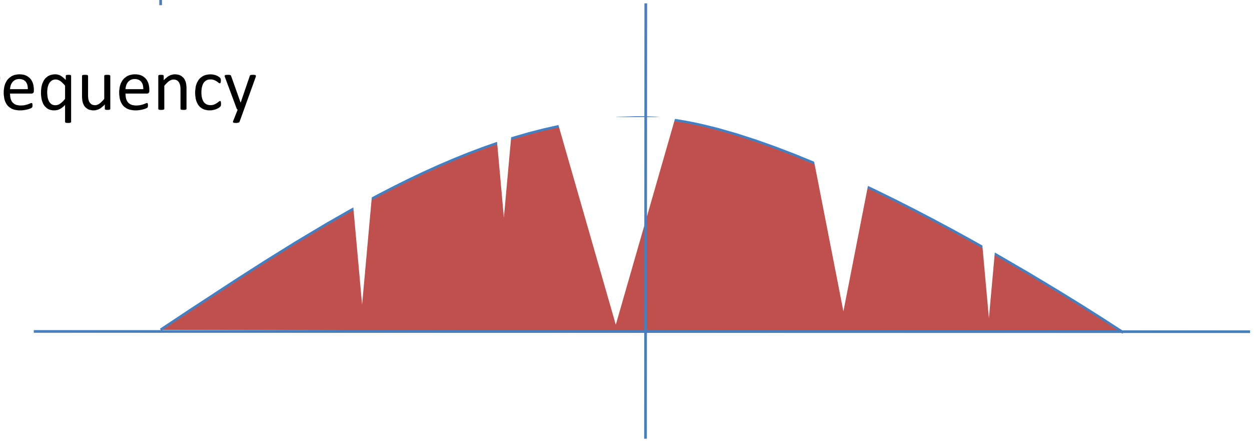


Background

Time



Frequency

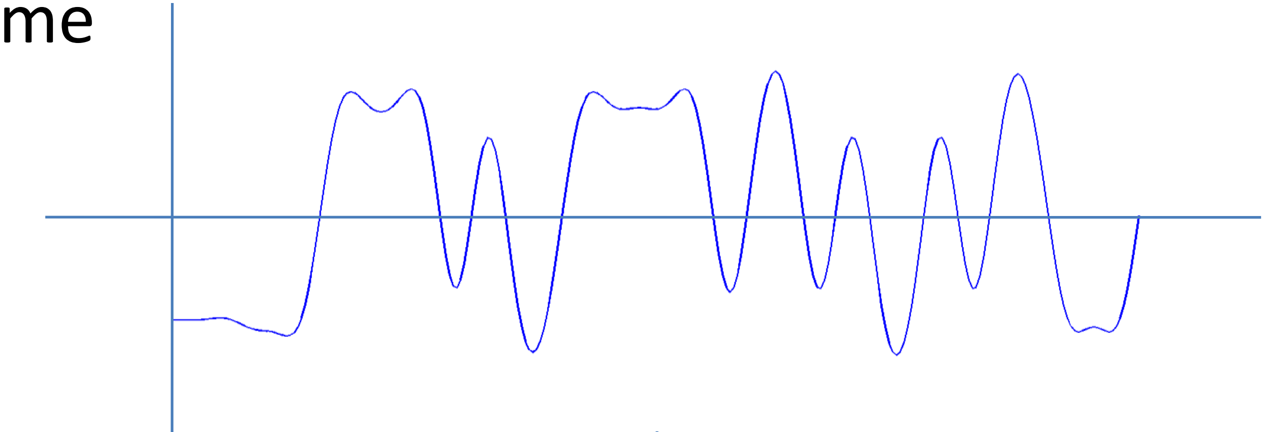


Background

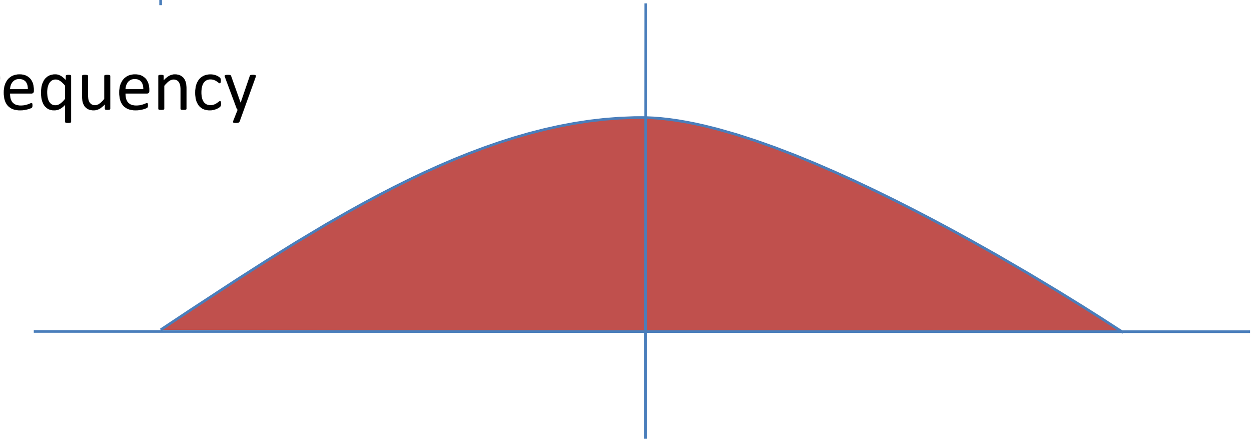
Narrowband Interference

Background

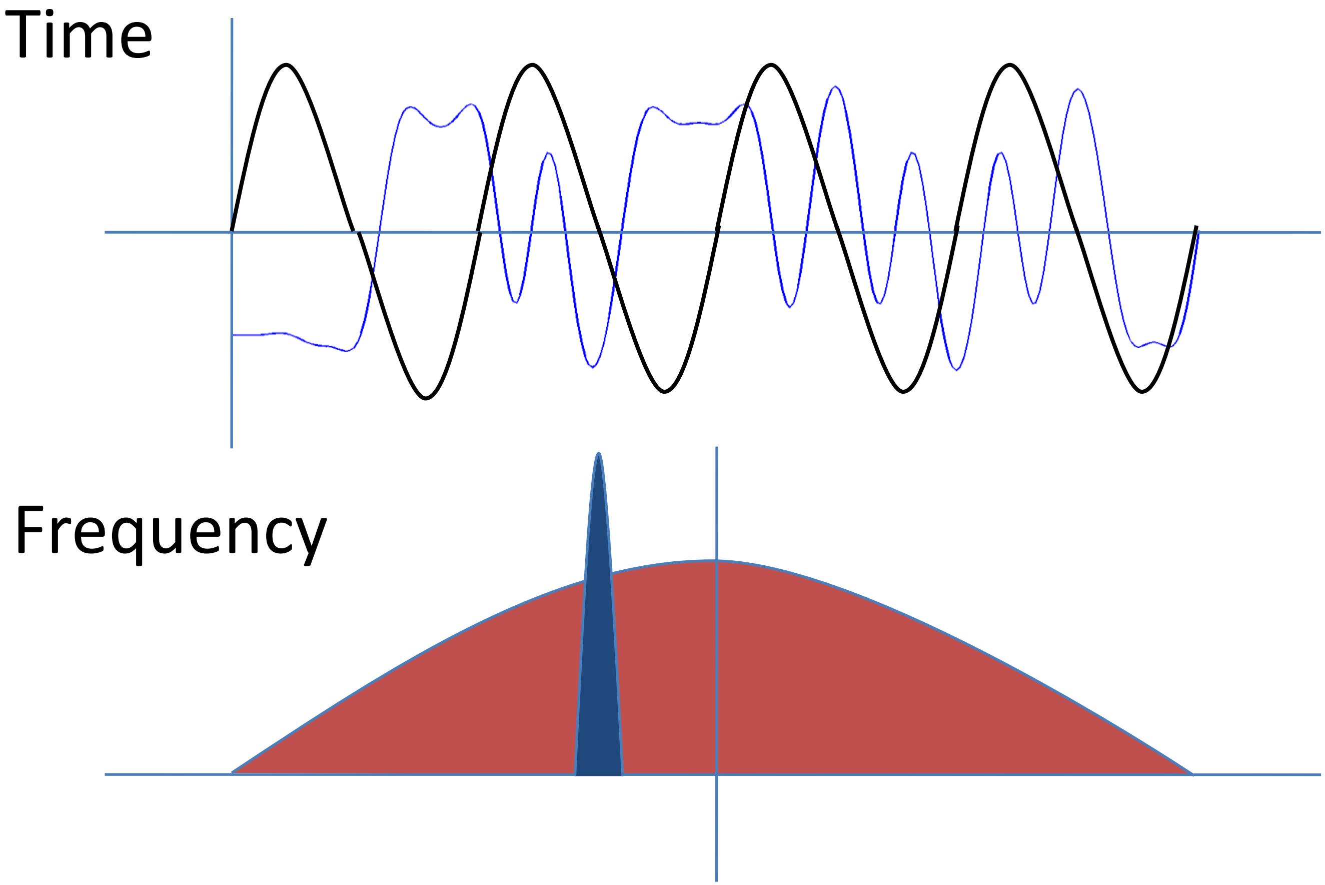
Time



Frequency

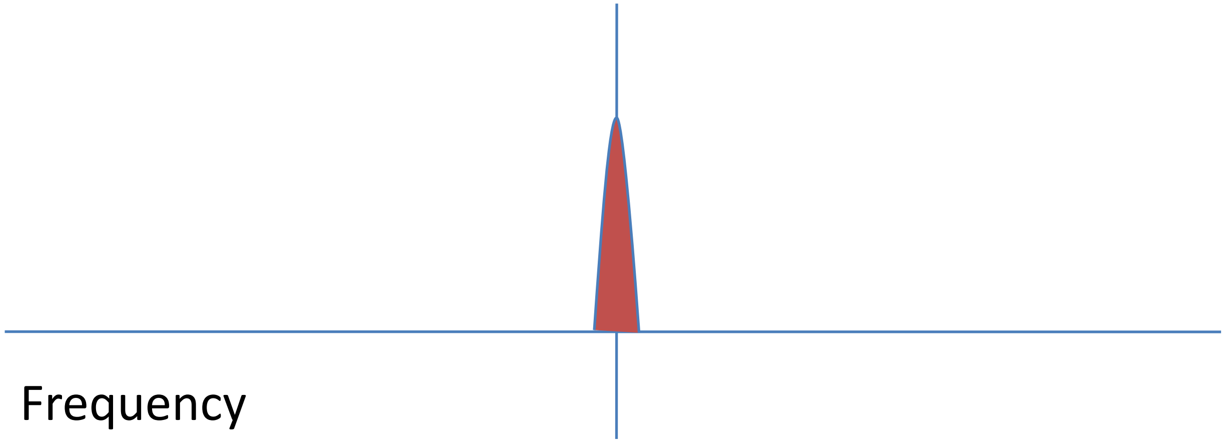


Background



Background

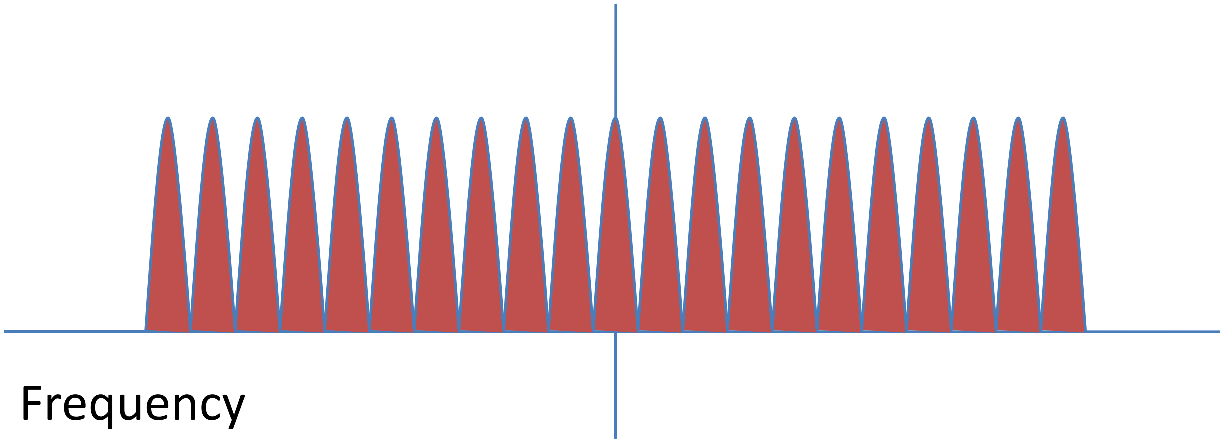
Single-Carrier System



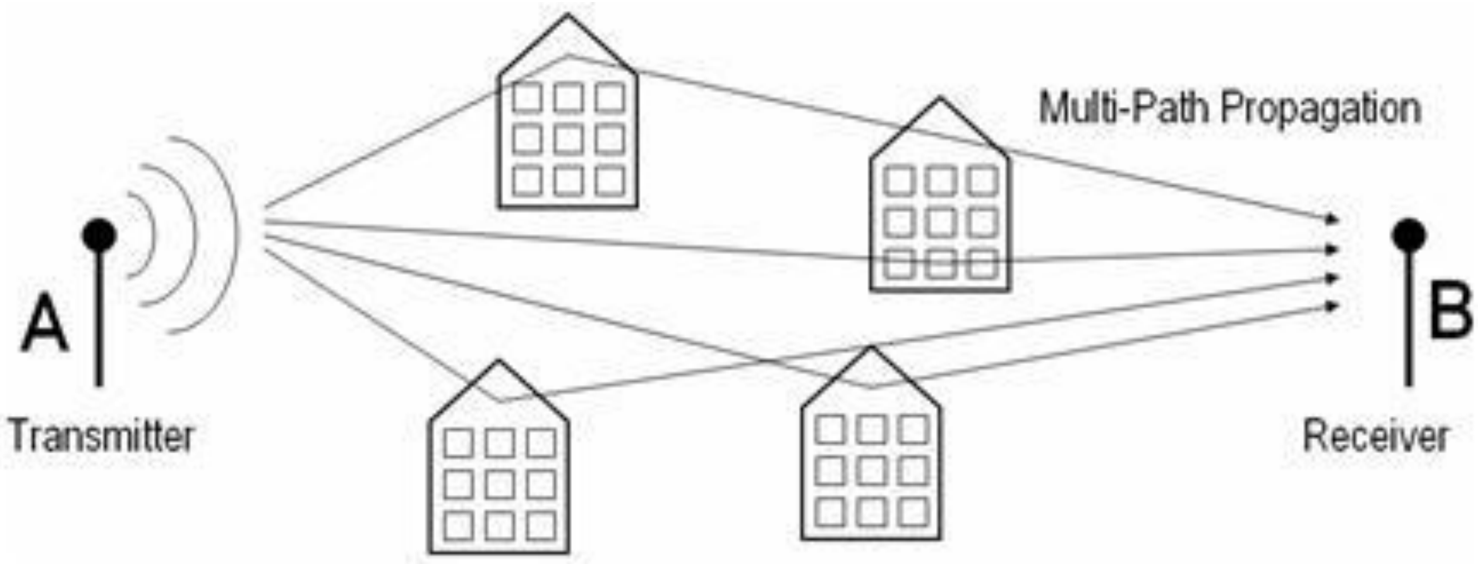
Background

Multi-Carrier System

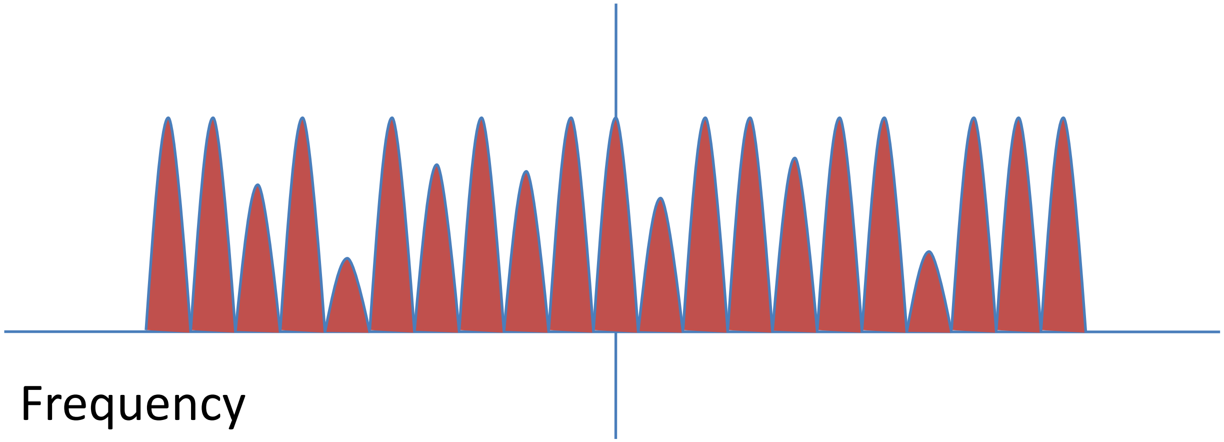
Frequency Division Multiplexing (FDM)



Background

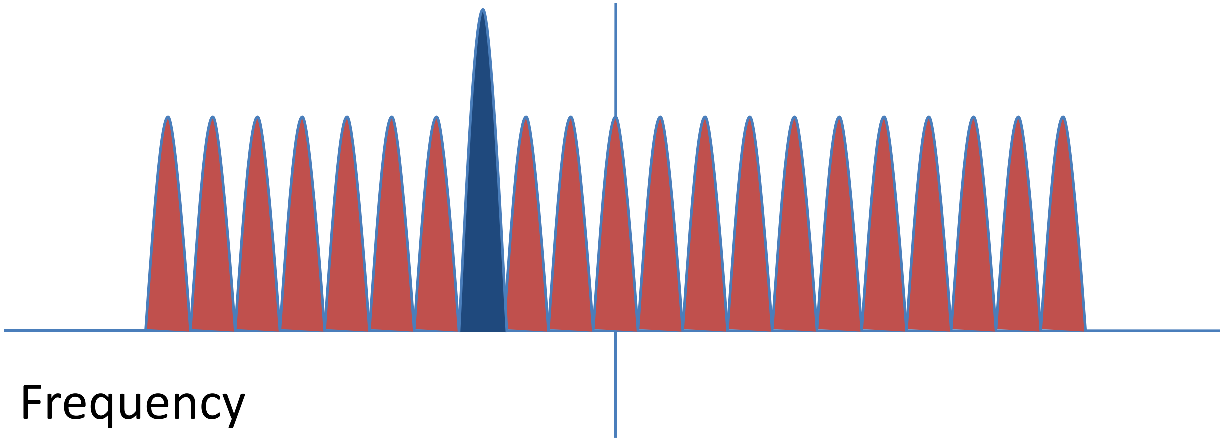


Background

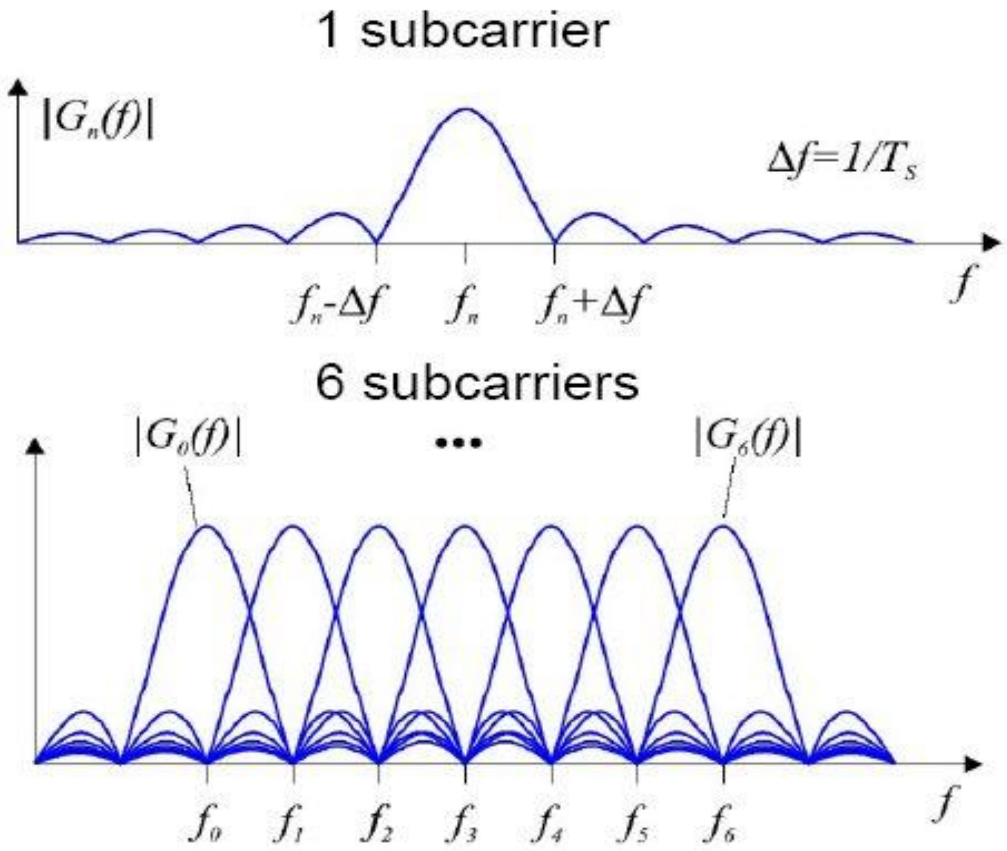


Background

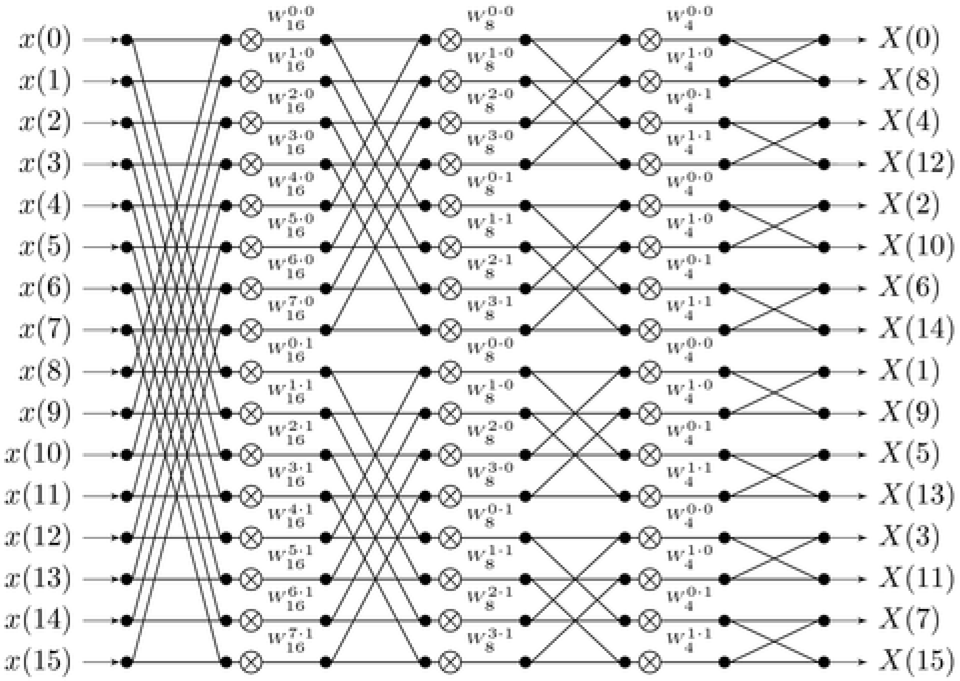
Background



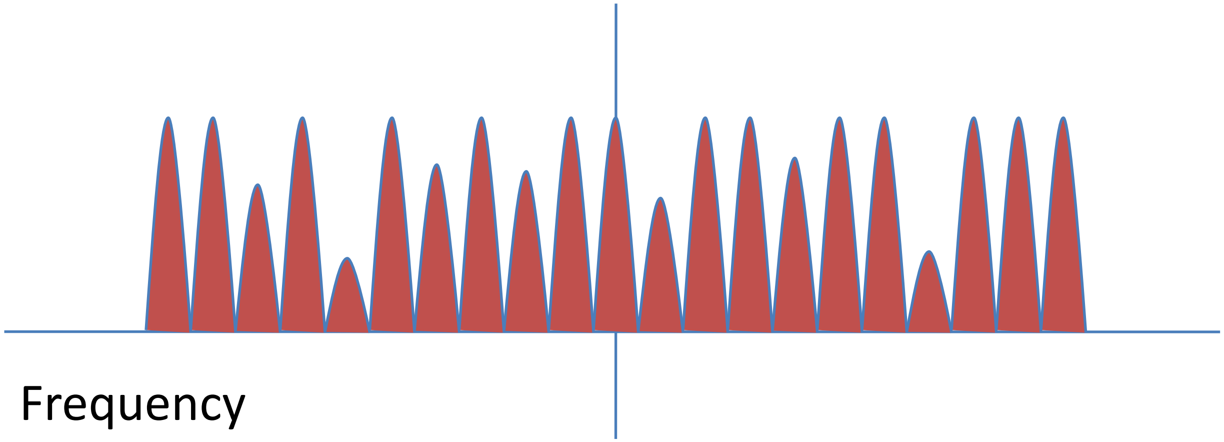
Orthogonal Frequency Division Multiplexing



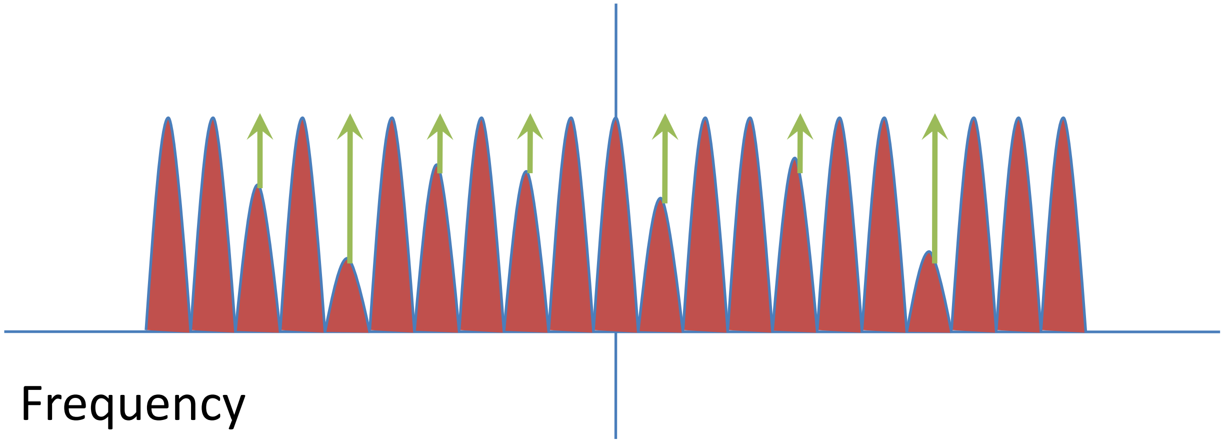
Fast Fourier Transform



Background



Background



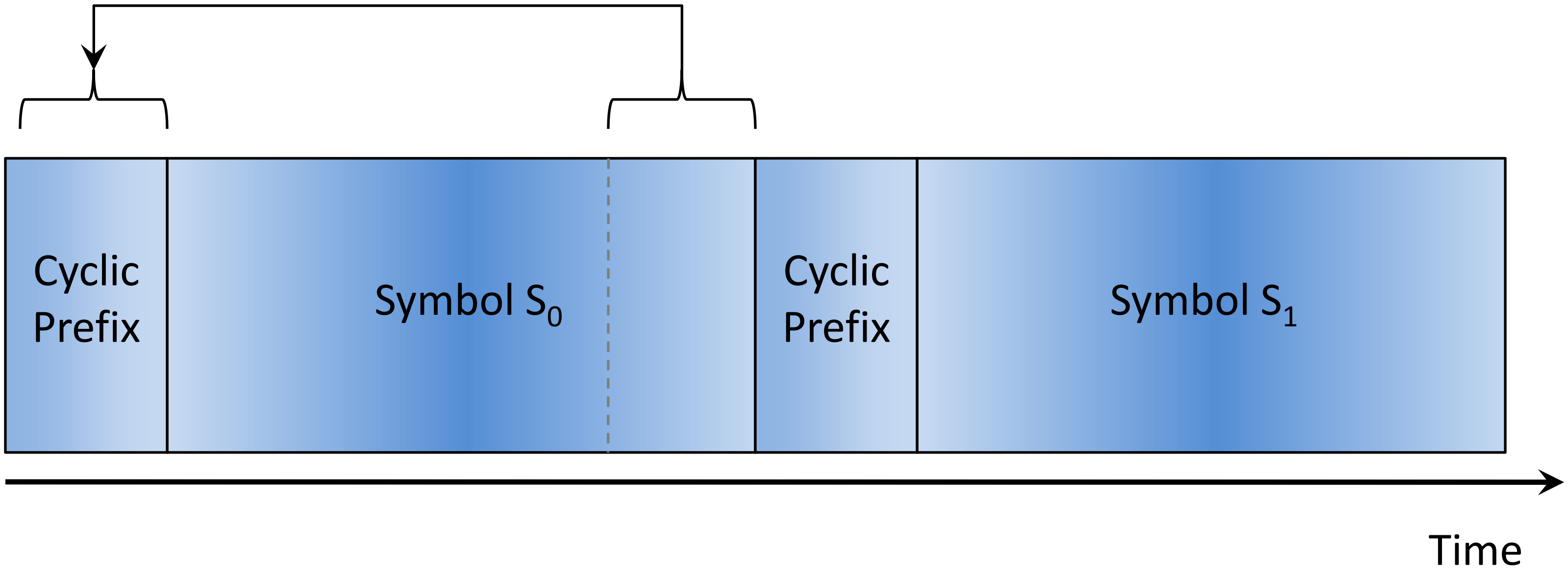
Background

- We want to use a simple multiplication in the frequency domain to equalize the channel.
  - CTFT: convolution in time domain corresponds to multiplication in frequency domain.
- This fact does not hold for DFT.
  - DFT: circular convolution in (discrete) time domain corresponds to multiplication in (discrete) frequency domain.

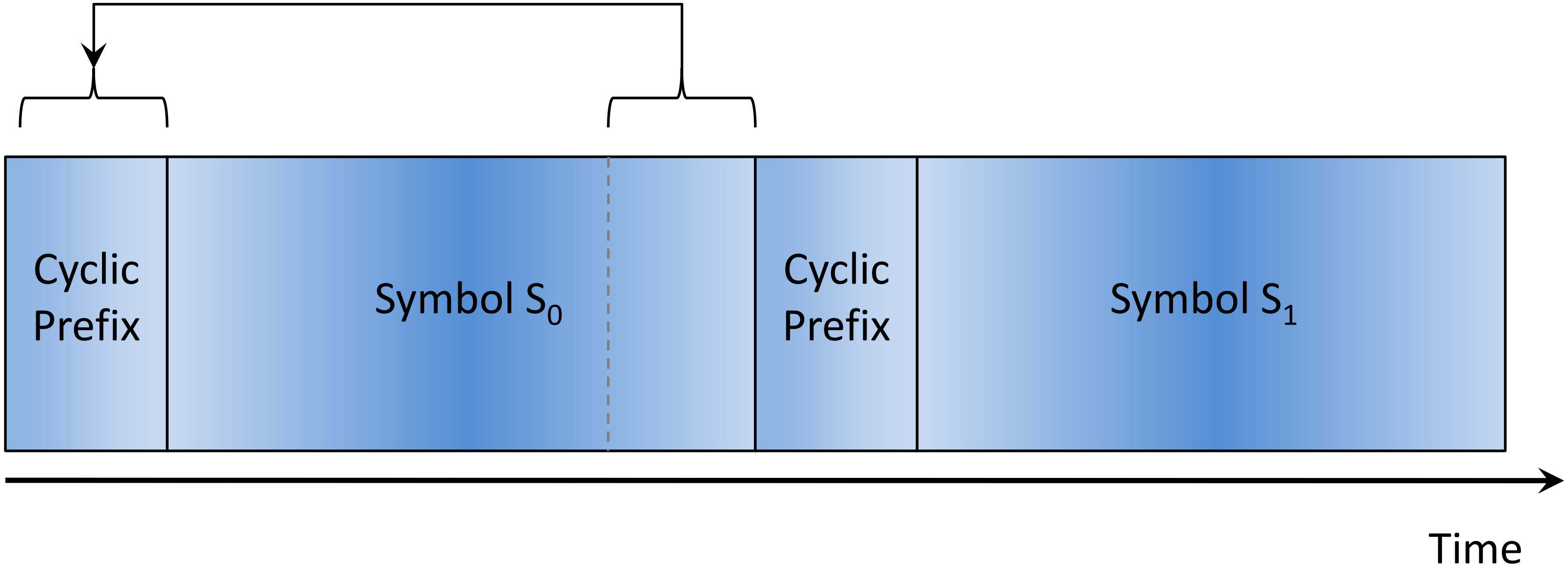
Background

- We want to use a simple multiplication in the frequency domain to equalize the channel.
  - CTFT: convolution in time domain corresponds to multiplication in frequency domain.
- This fact does not hold for DFT.
  - DFT: circular convolution in (discrete) time domain corresponds to multiplication in (discrete) frequency domain.
- So, we want circular convolution and not the regular convolution.
  - Problem: Real channel does regular convolution.
  - Solution: With cyclic prefix, regular convolution can be used to create circular convolution.

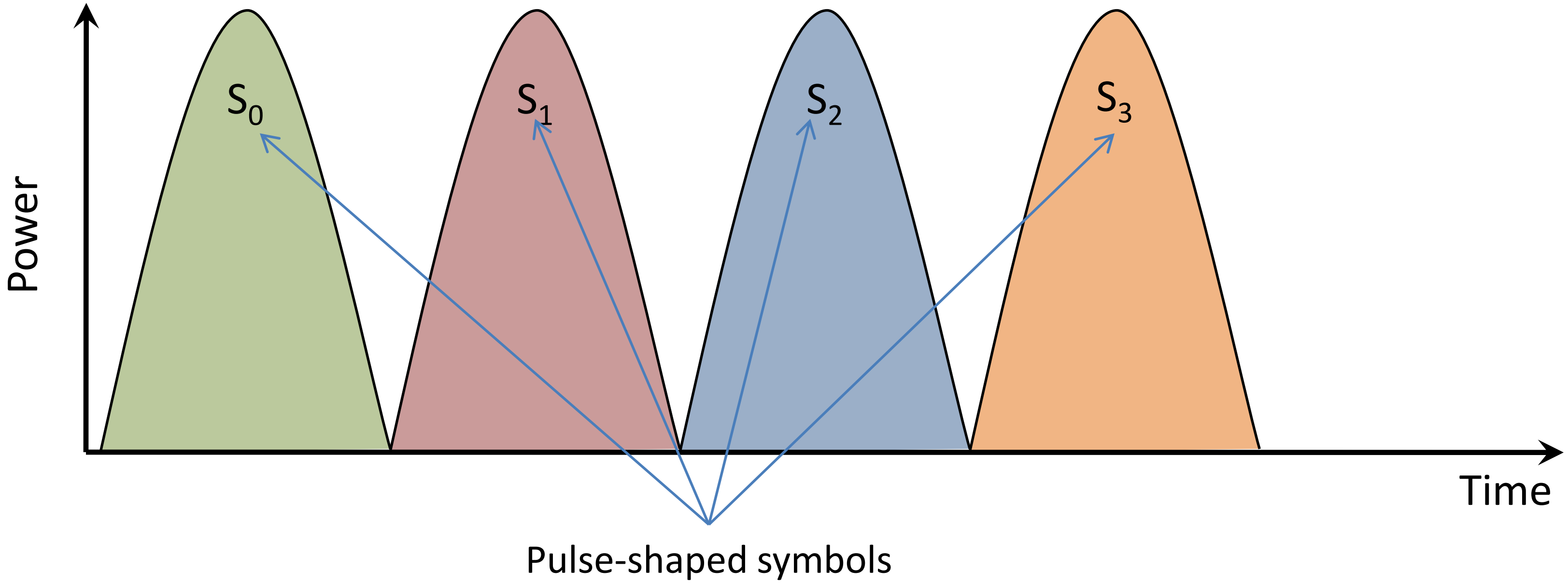
Background

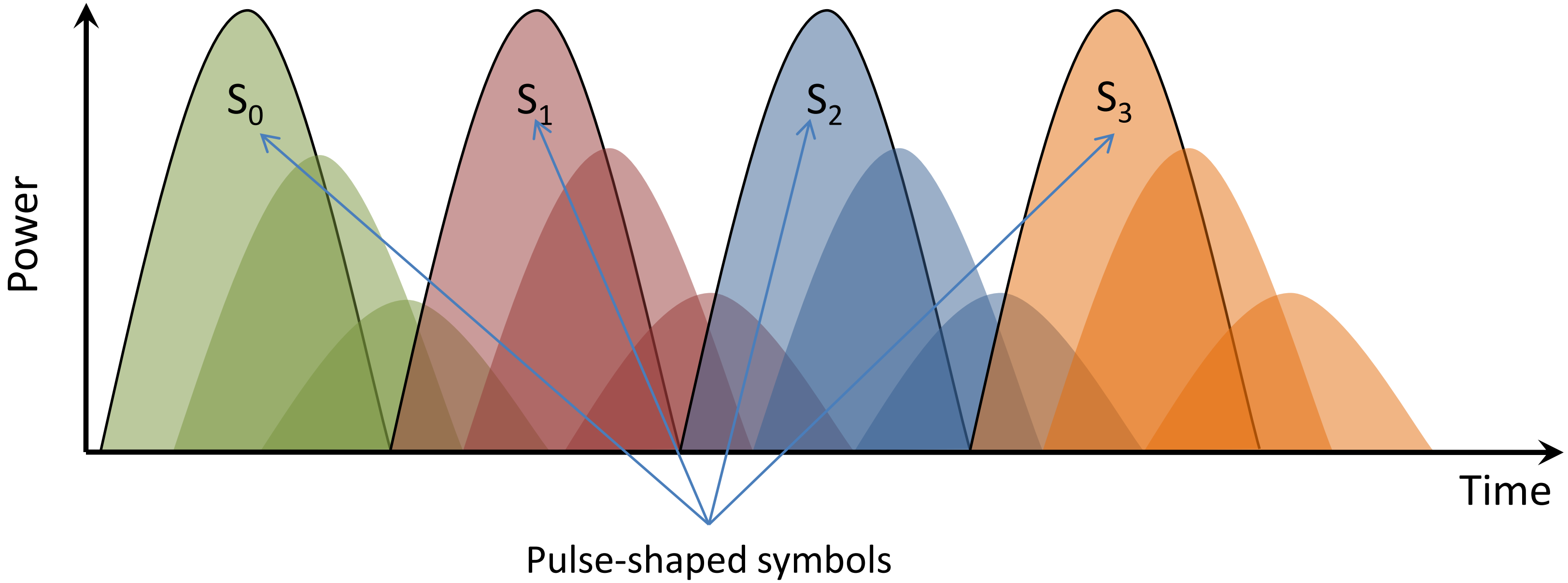


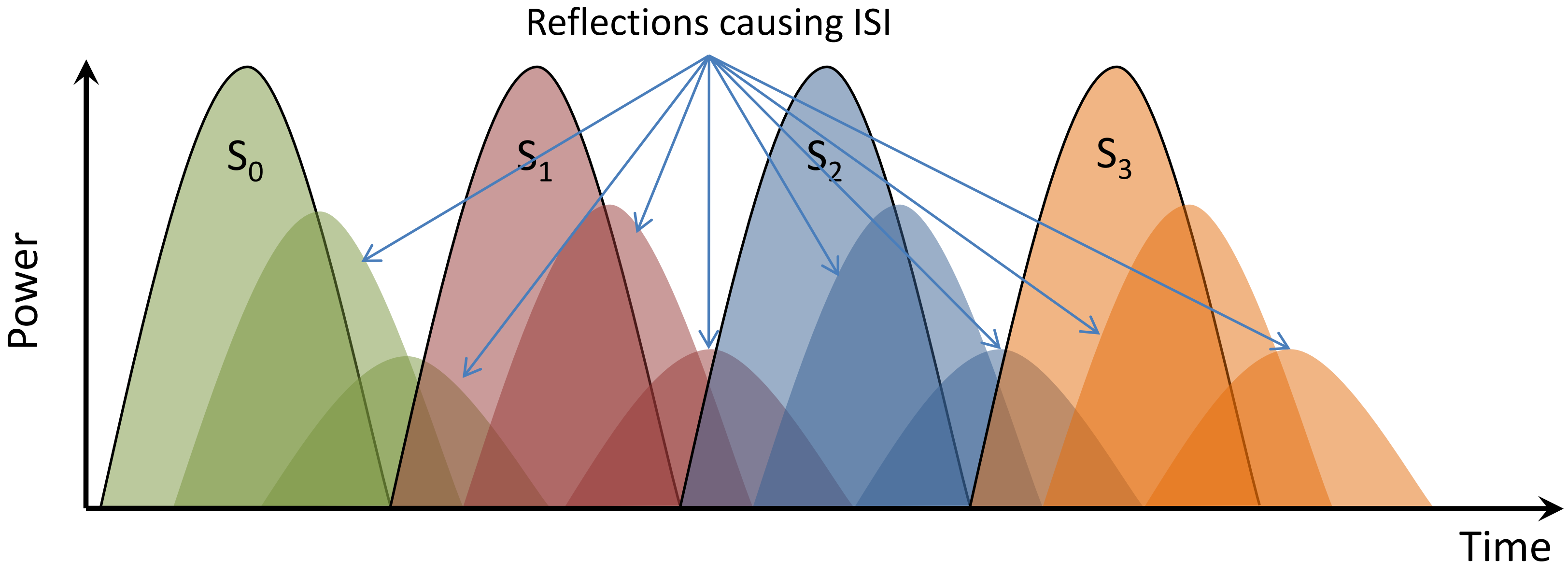
Background

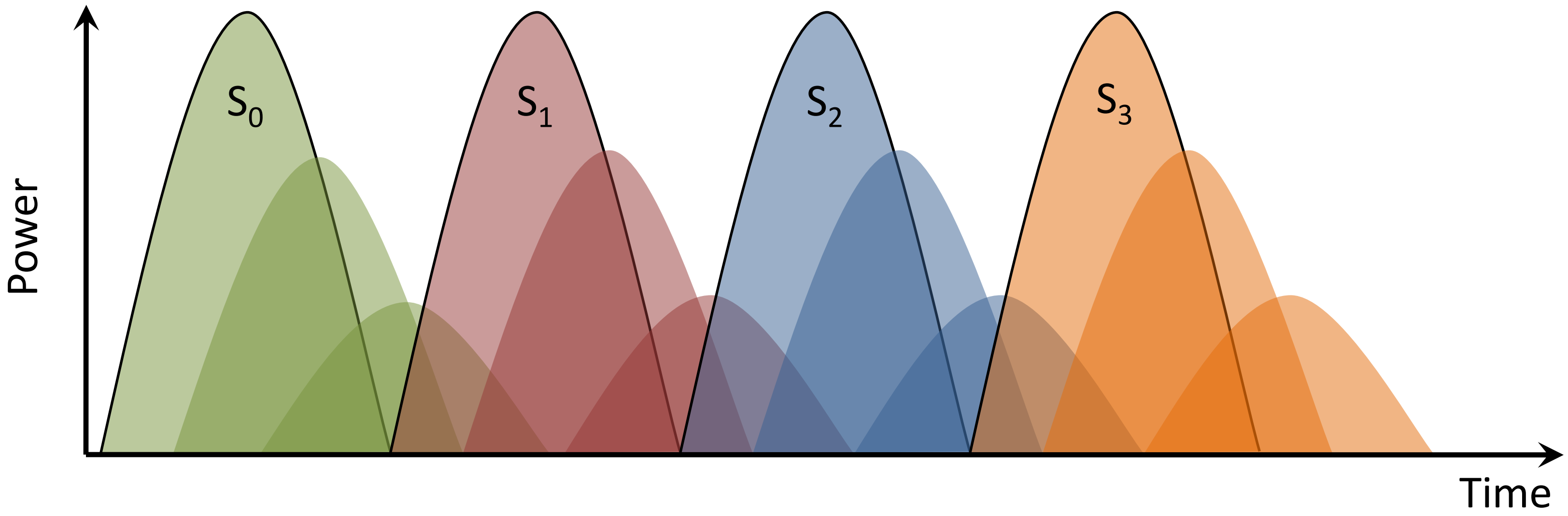


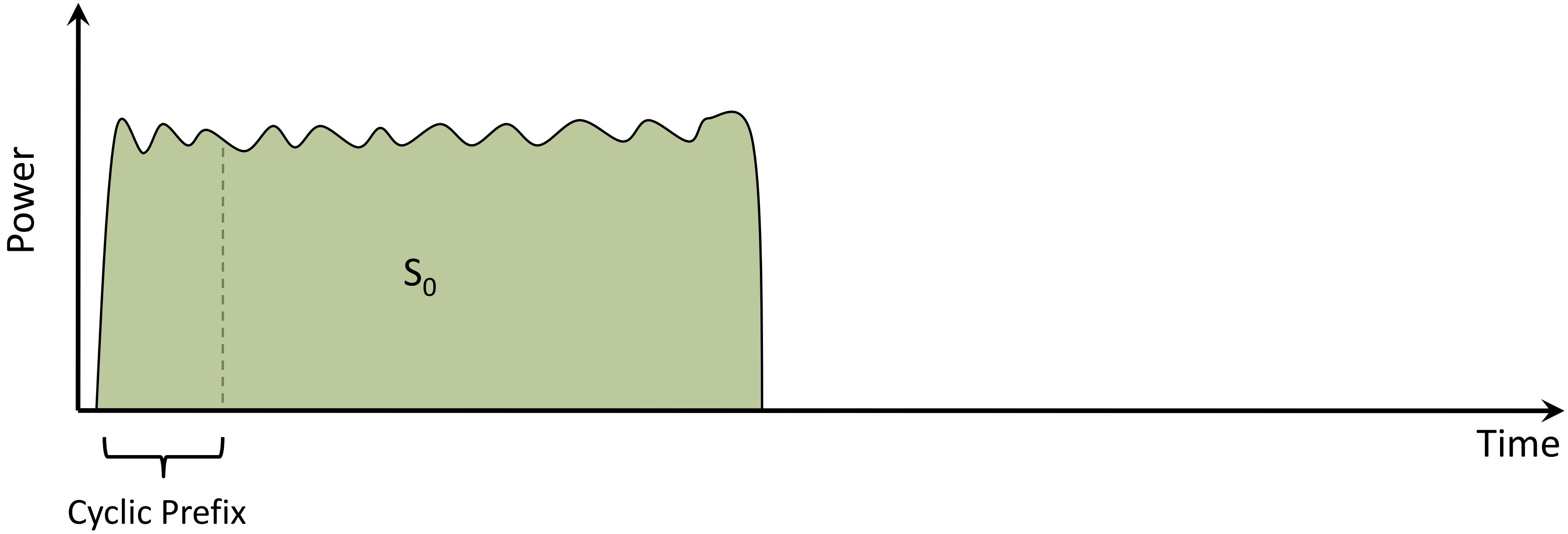
- Eliminate ISI
- Turn linear convolution into circular convolution

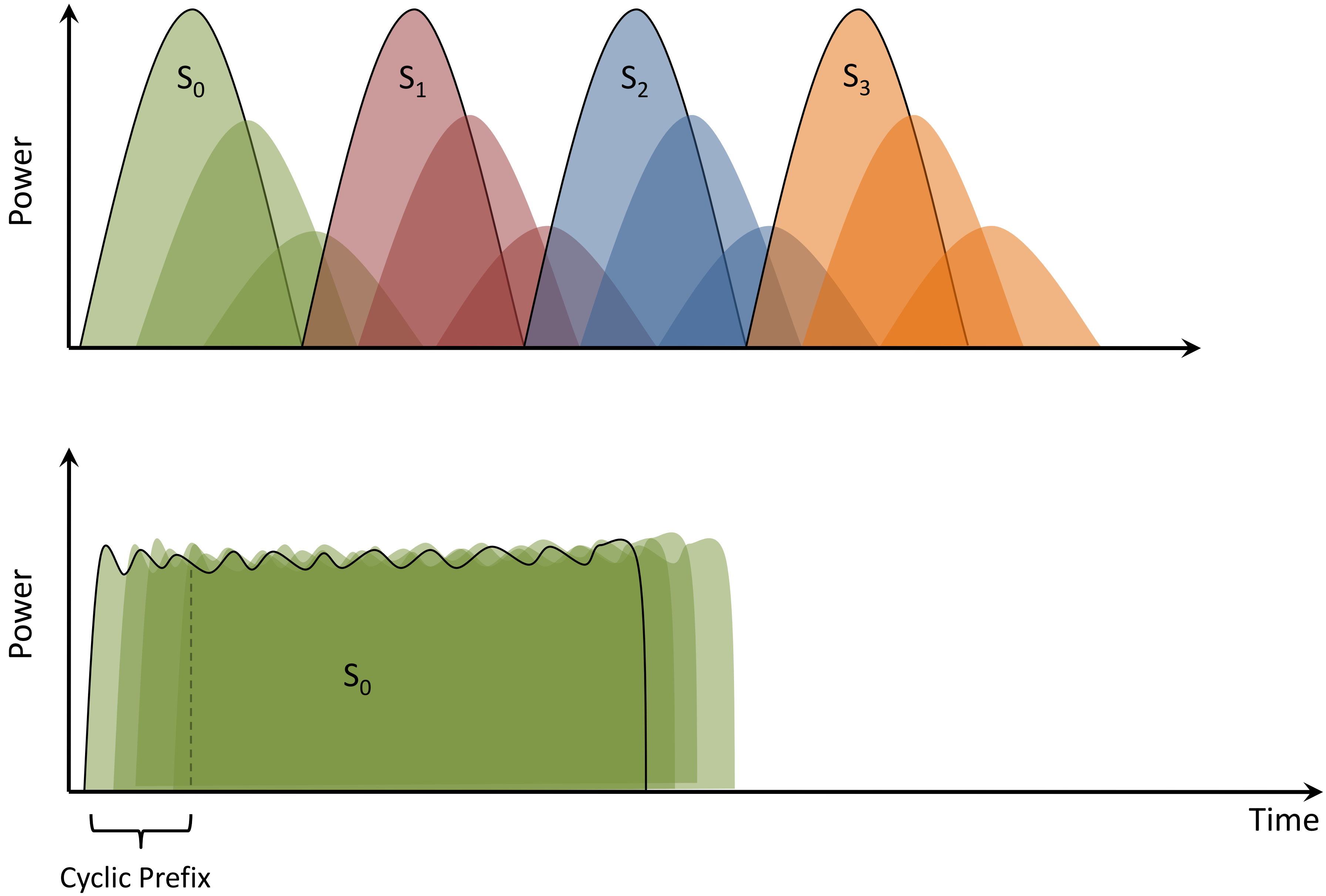

Power
S_0
S_1
S_2
S_3
Time
Pulse-shaped symbols


Power
S_0
S_1
S_2
S_3
Time
Pulse-shaped symbols


Reflections causing ISI
S_0
S_1
S_2
S_3
Power
Time

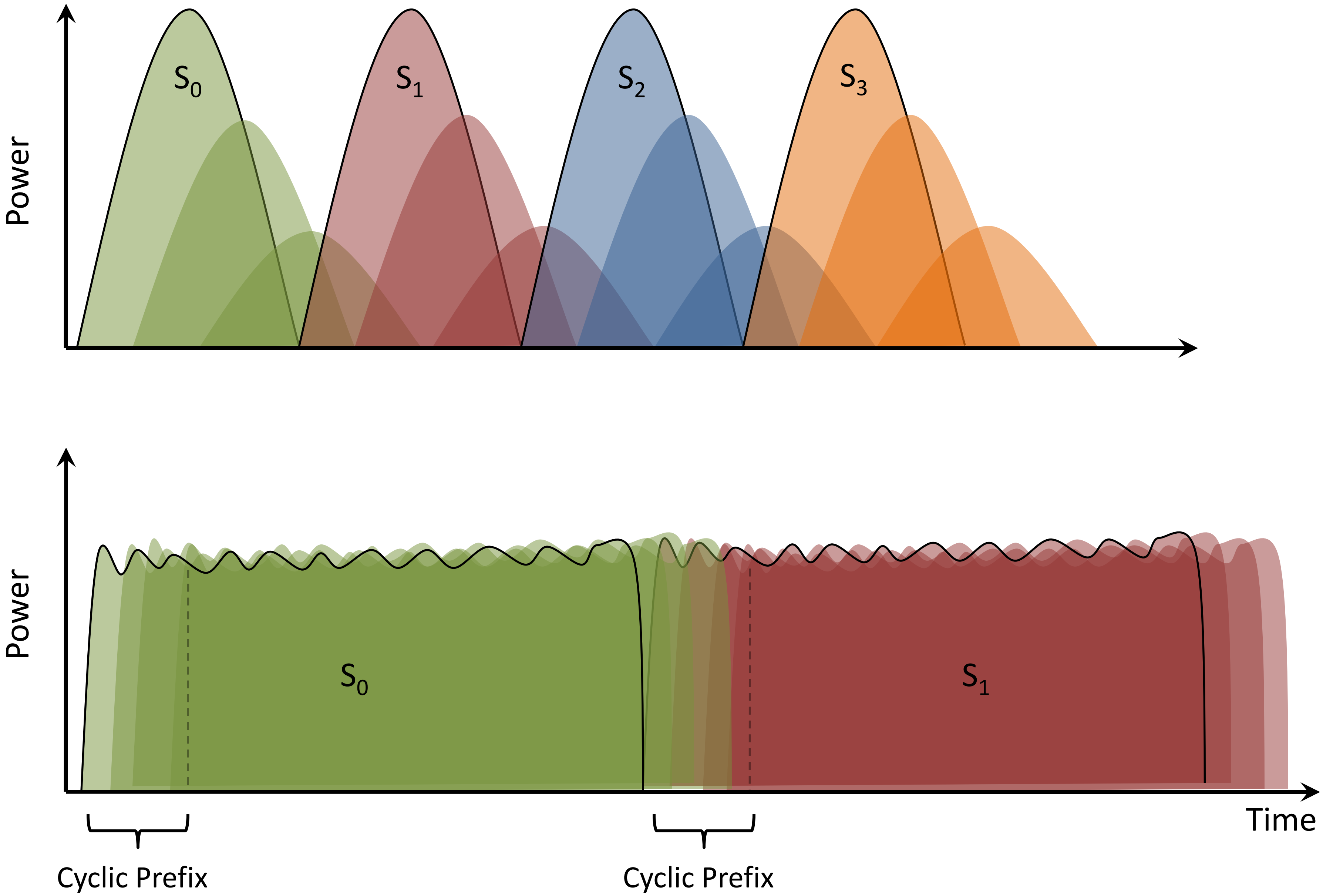

Power
S_0
S_1
S_2
S_3
Time


Power
S_0
Time
Cyclic Prefix



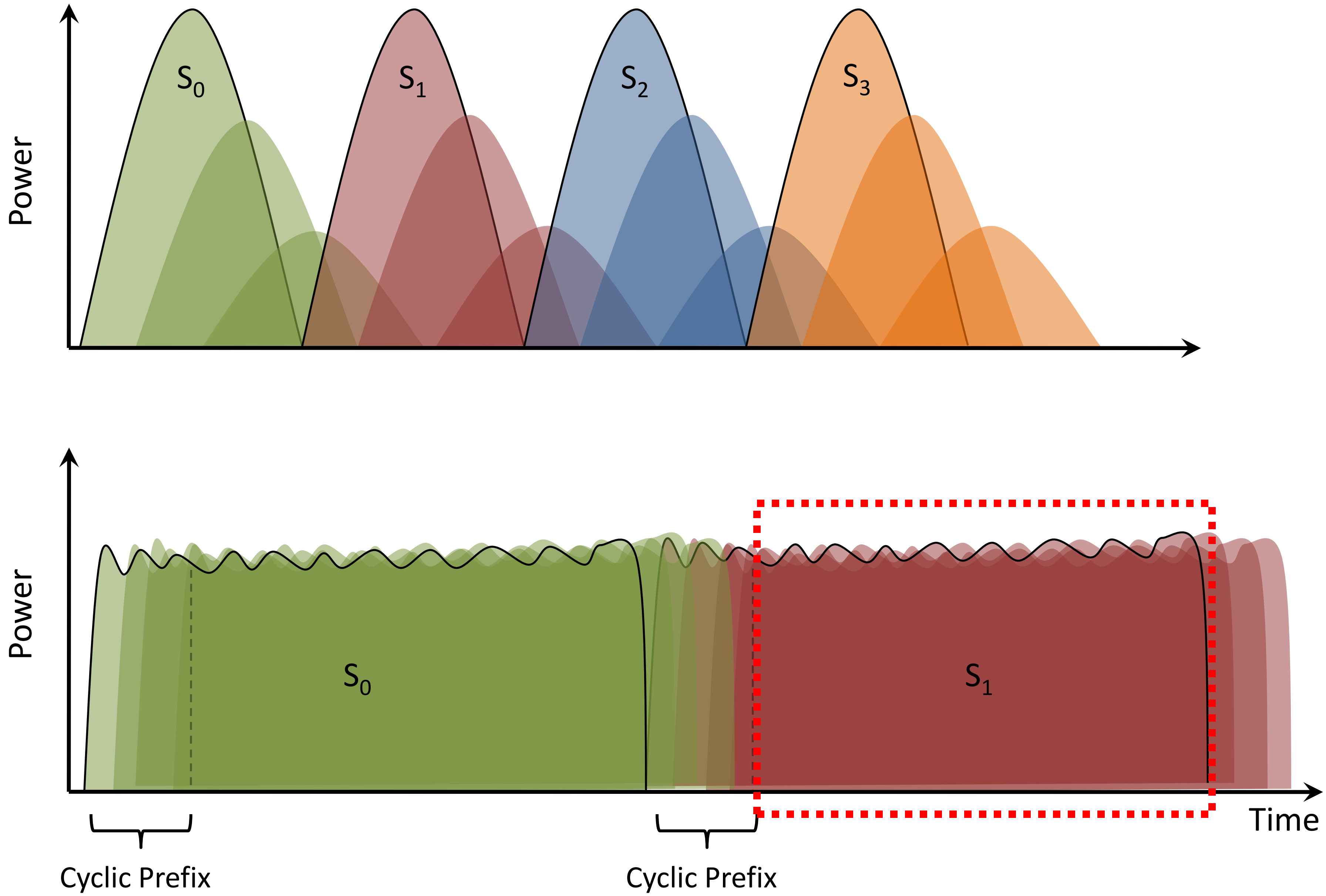
Software Radio Systems

CTVR the telecommunications research centre



Software Radio Systems

CTVR the telecommunications research centre

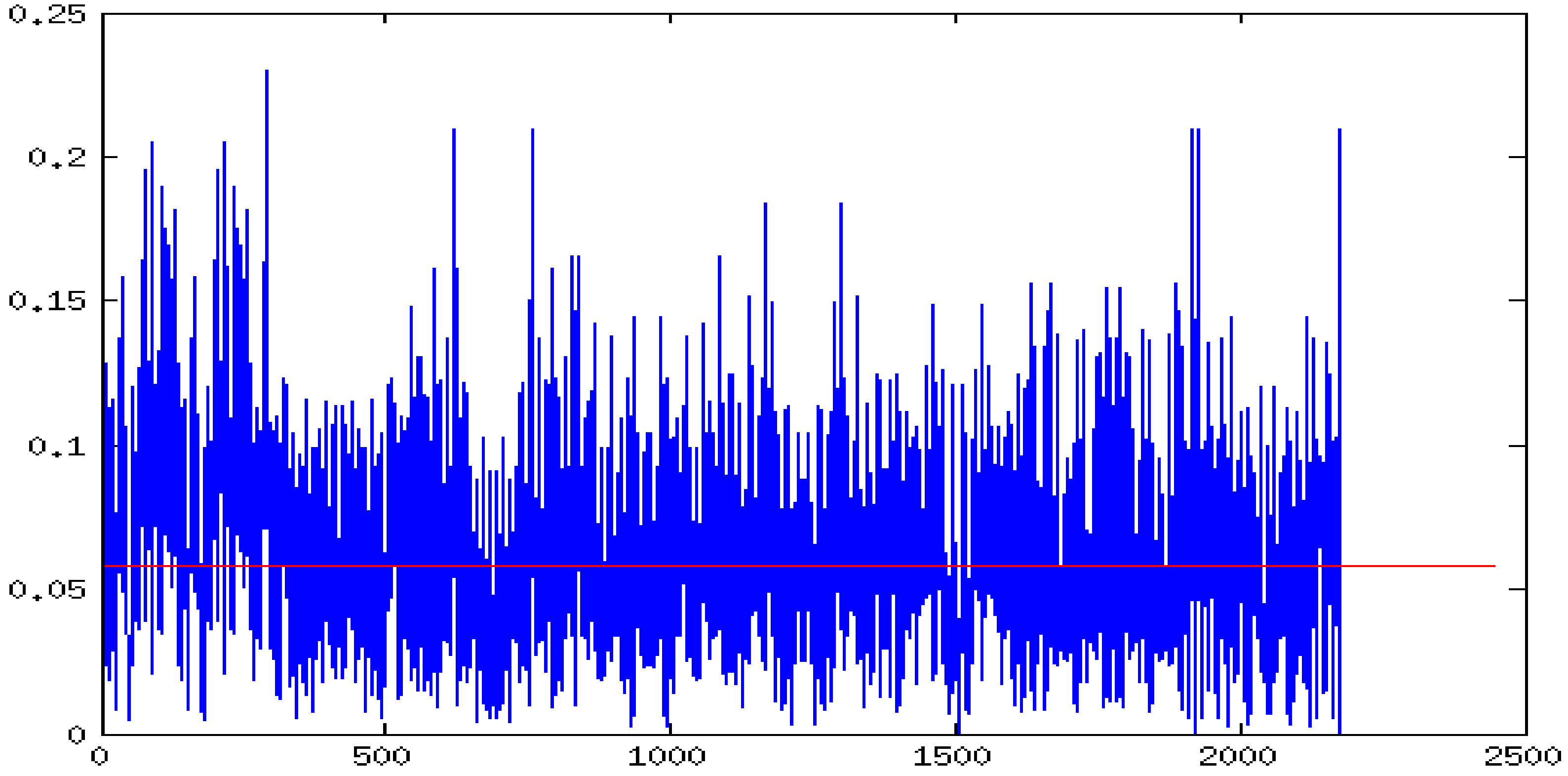

Power
S_0
S_1
S_2
S_3
Power
S_0
S_1
Time
Cyclic Prefix
Cyclic Prefix

Background

- Drawbacks:

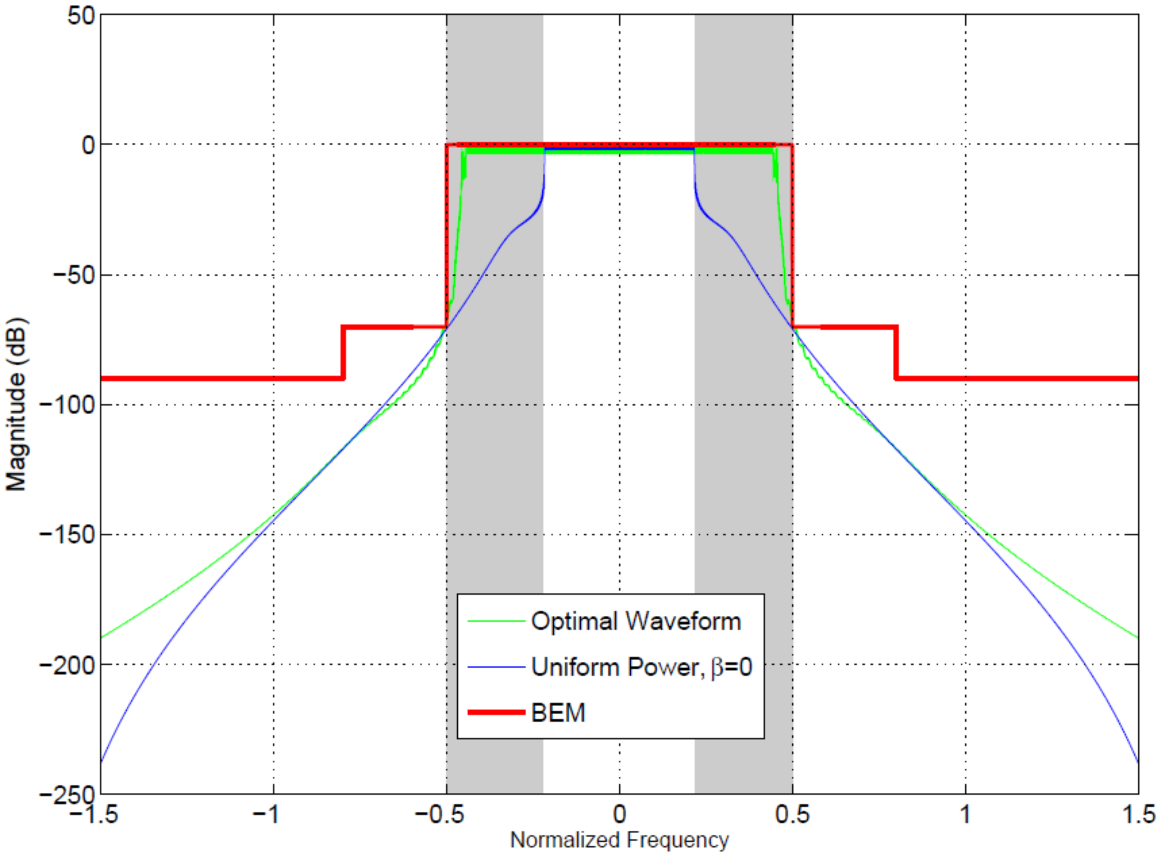
Background

- Drawbacks:
  - – High PAPR



Background

- Drawbacks:
  - – Slow roll-off in out-of-band frequencies

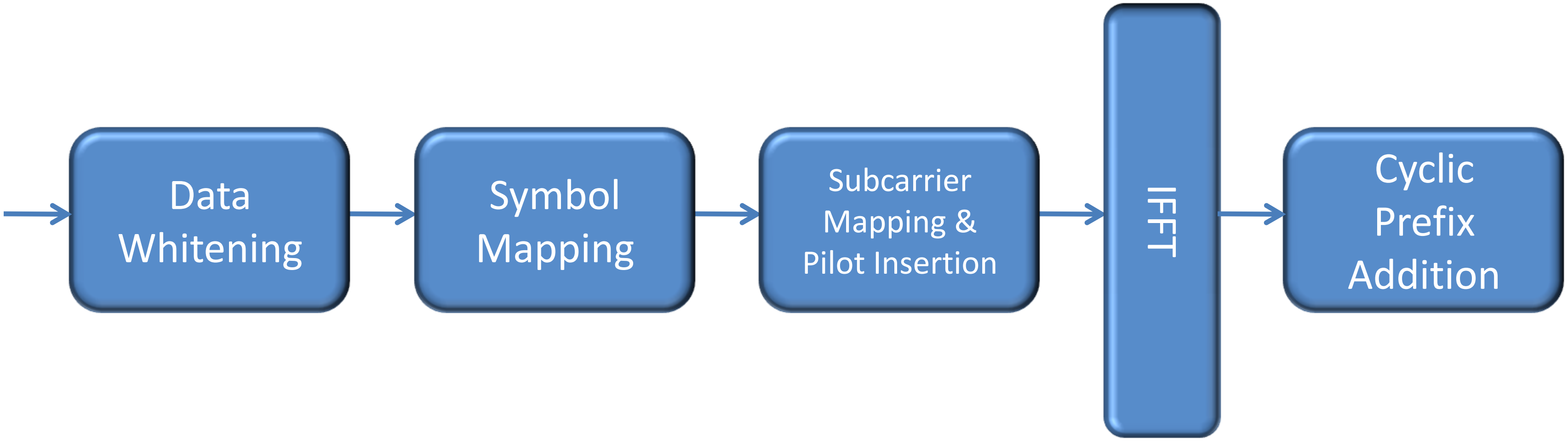


Improving OFDM

- PAPR reduction
  - Clipping
  - Signal scrambling
  - Block coding
  - Selected mapping
  - Etc.
- Spectral shaping
  - Symbol shaping
  - Cancellation carriers
  - Etc.

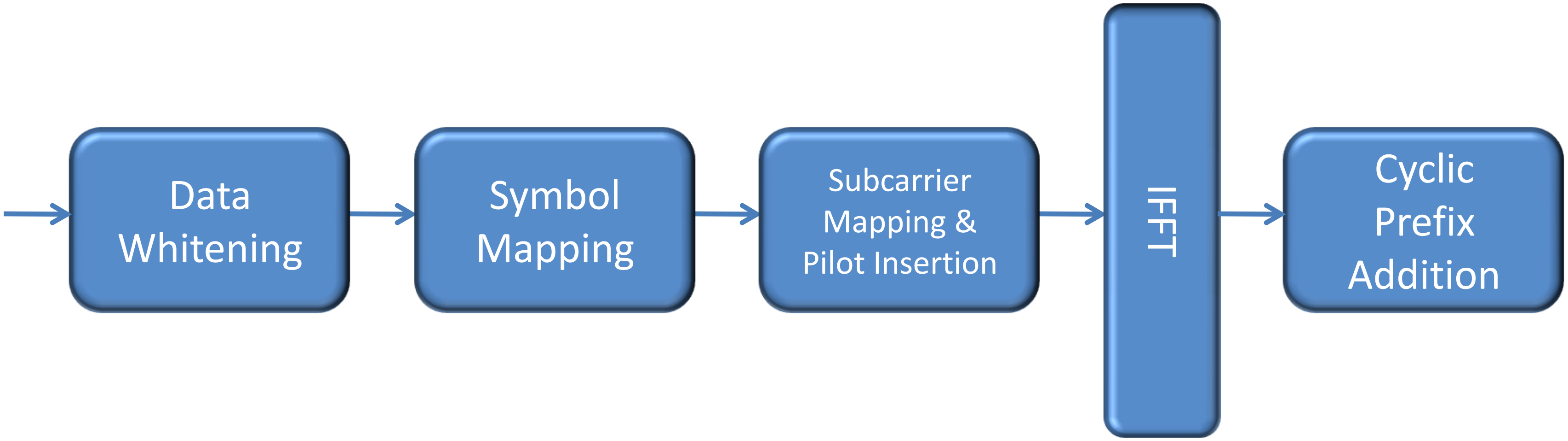
Modulation

- OFDM Modulation



Modulation

- OFDM Modulation



Data Whitening

Symbol Mapping

Subcarrier Mapping & Pilot Insertion

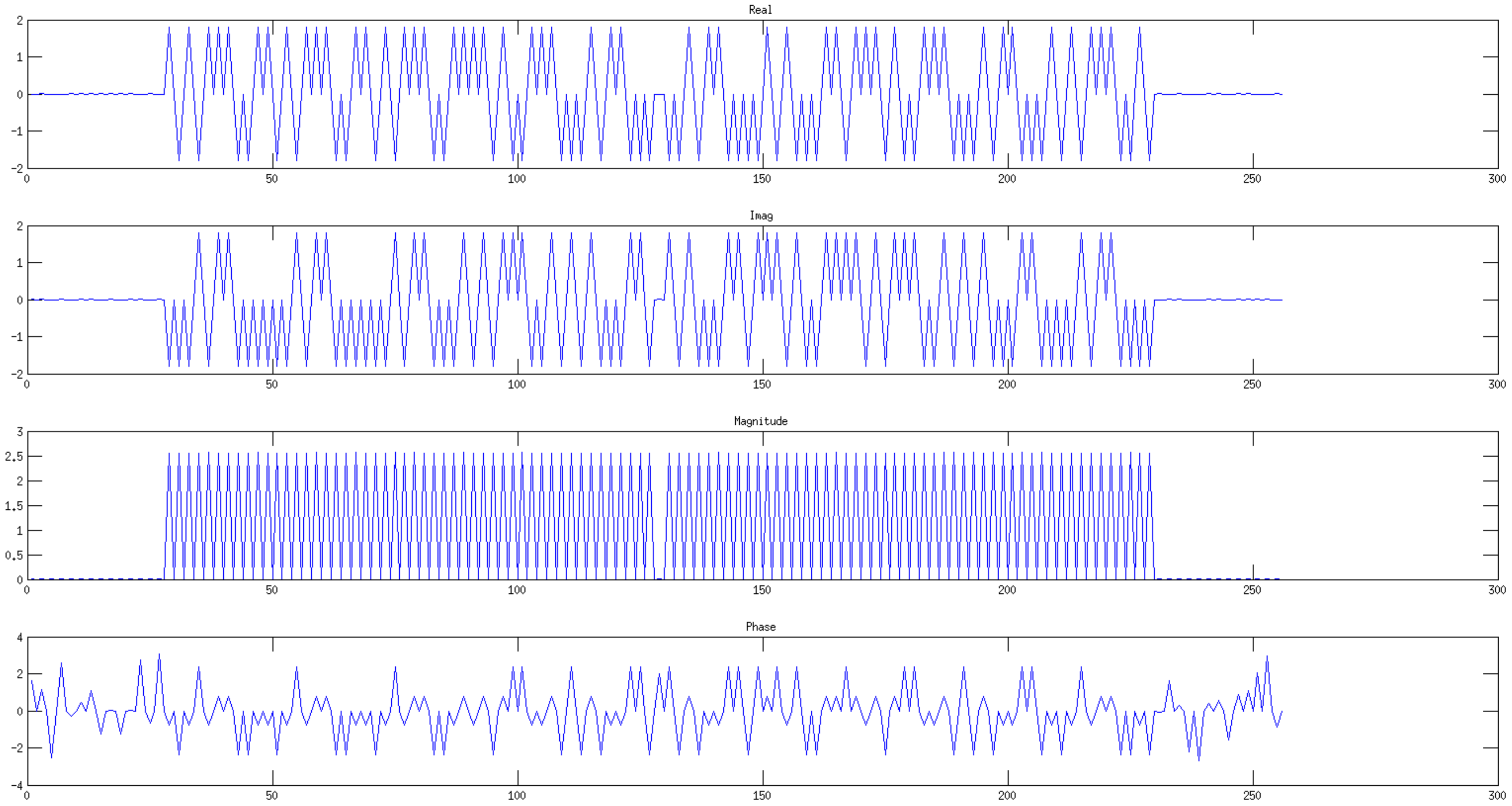
IFFT

Cyclic Prefix Addition

Preamble Symbol(s)	Header Symbol(s)	Data Symbols	Frame Guard

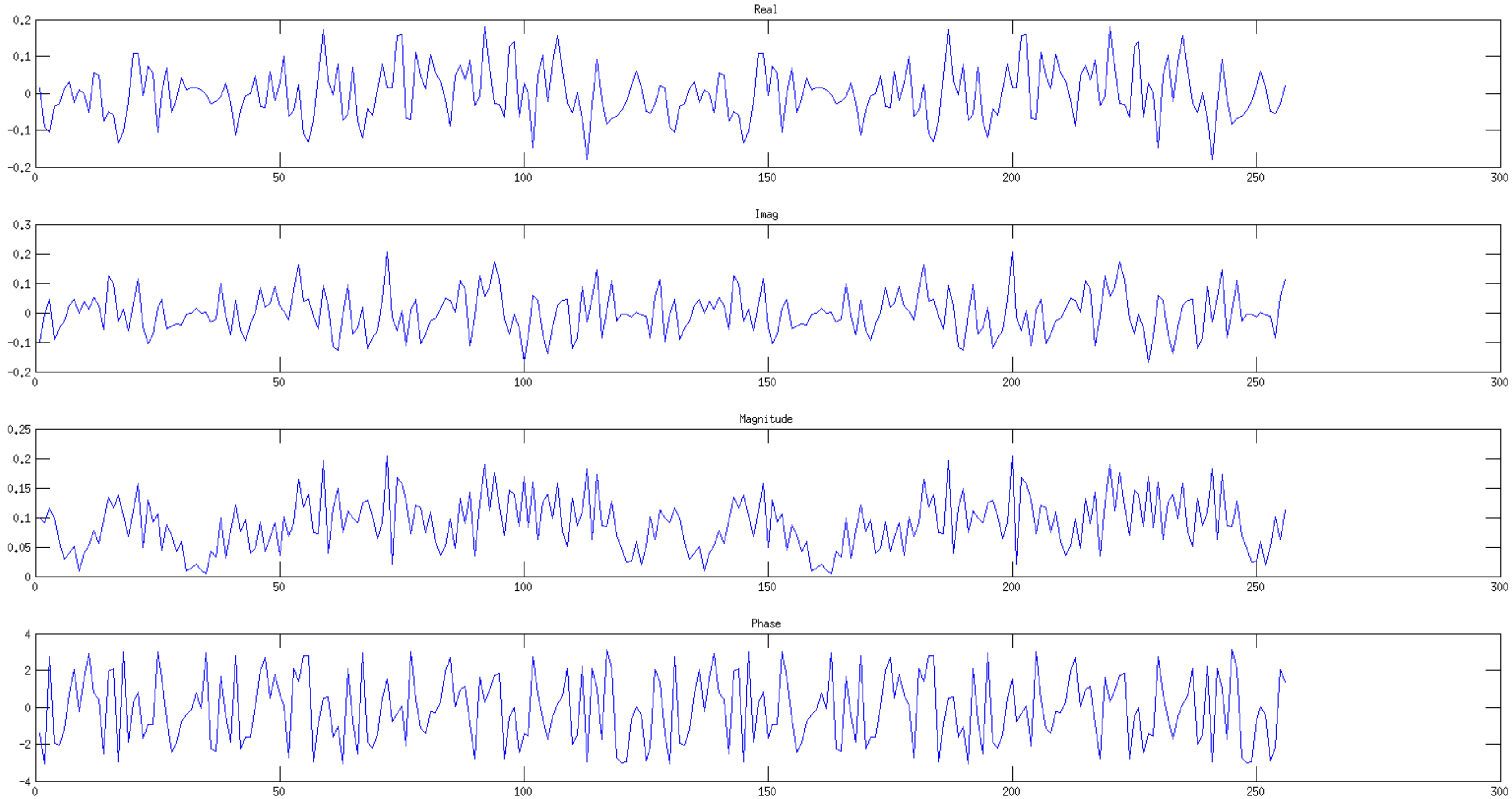
Modulation

Transmit Preamble (Frequency)



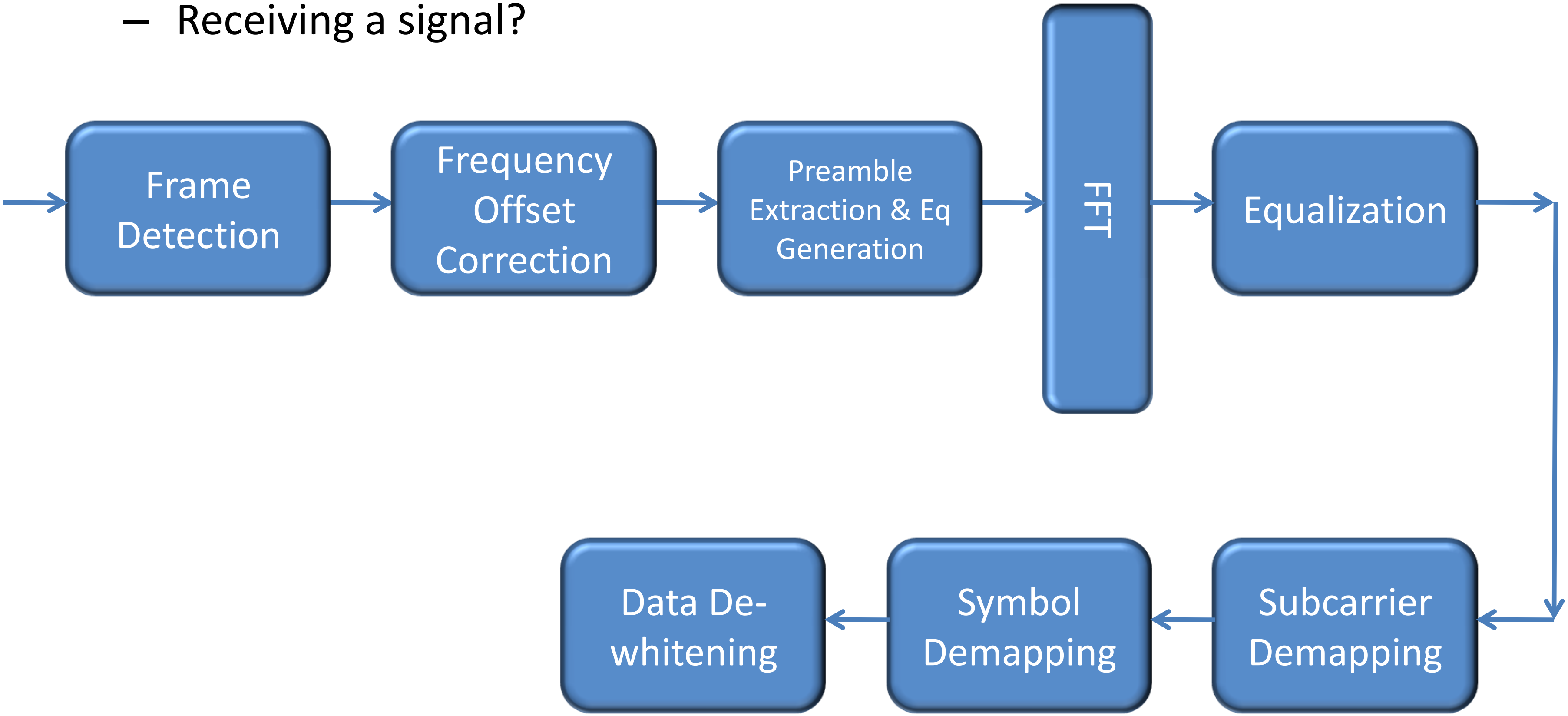
Modulation

Transmit Preamble (Time)



Demodulation

- OFDM Demodulation
  - Receiving a signal?

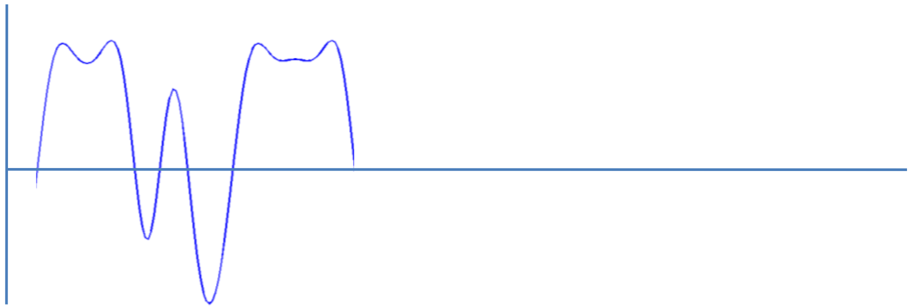


Demodulation

- Time and frequency synchronization
  - Moose/ Schmidl & Cox

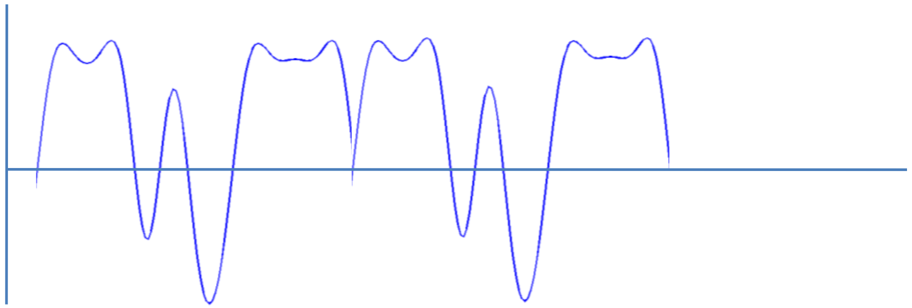
Demodulation

- Time and frequency synchronization
  - Moose/ Schmidl & Cox



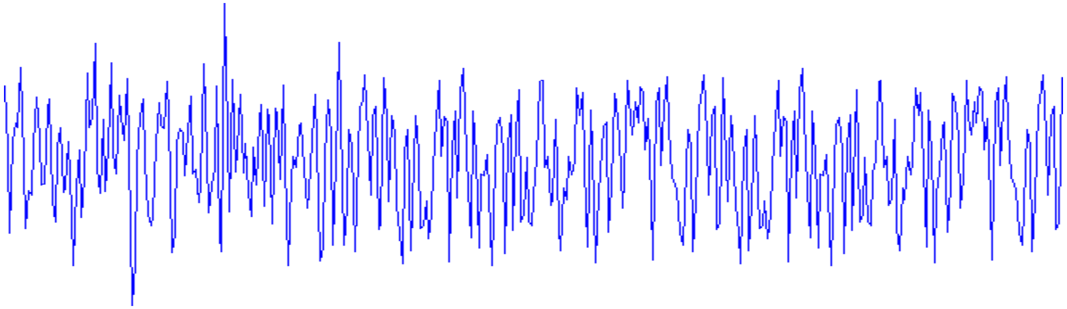
Demodulation

- Time and frequency synchronization
  - Moose/ Schmidl & Cox



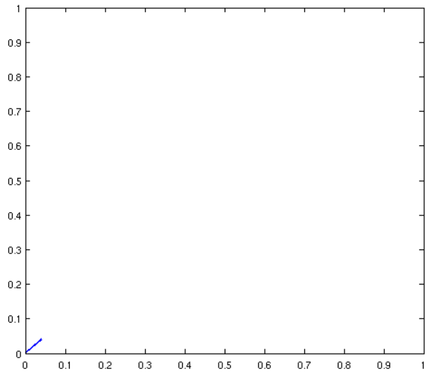
Demodulation

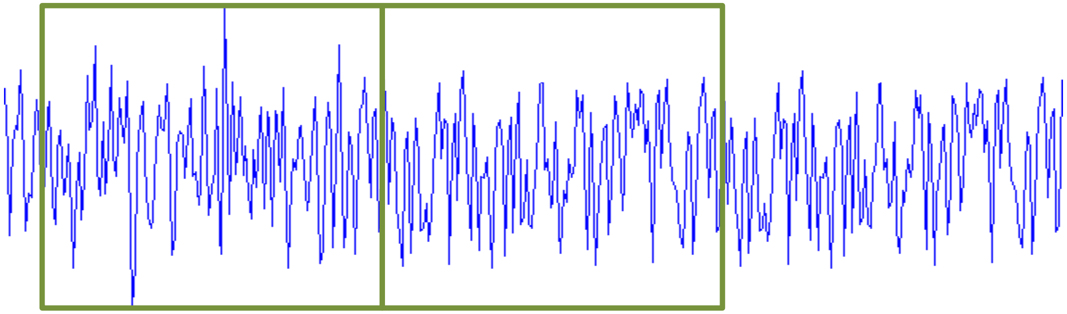
- Time and frequency synchronization
  - Moose/ Schmidl & Cox

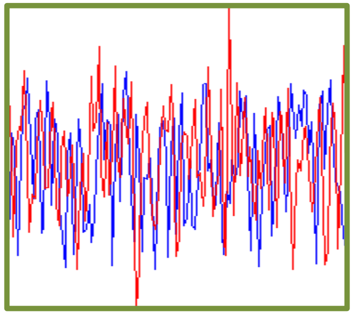


Demodulation

- Time and frequency synchronization
  - Moose/ Schmidl & Cox

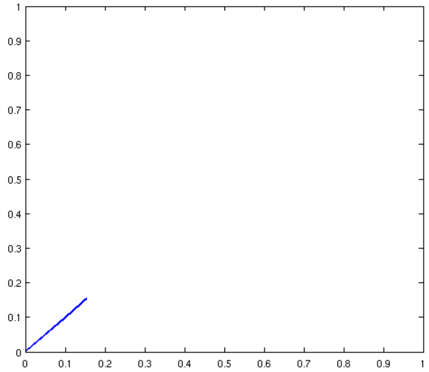


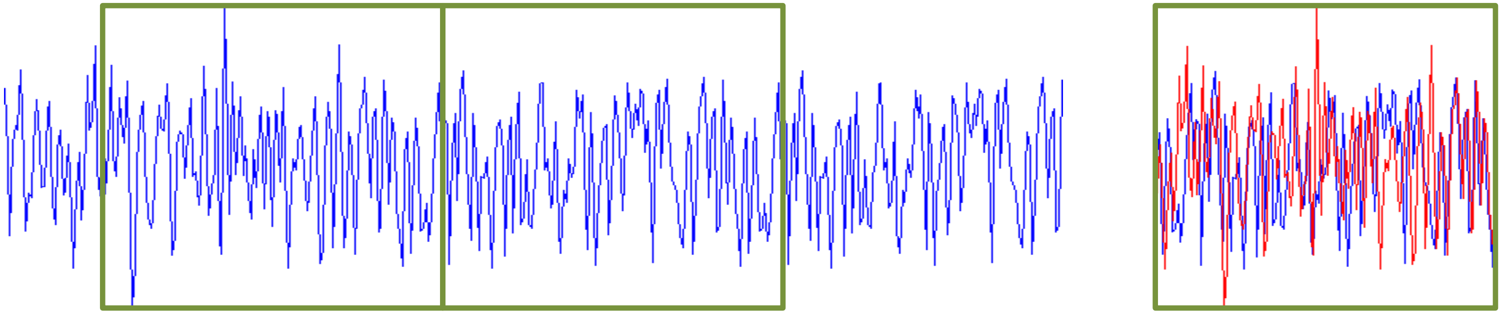




Demodulation

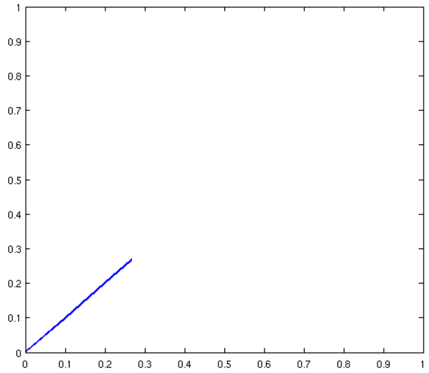
- Time and frequency synchronization
  - Moose/ Schmidl & Cox

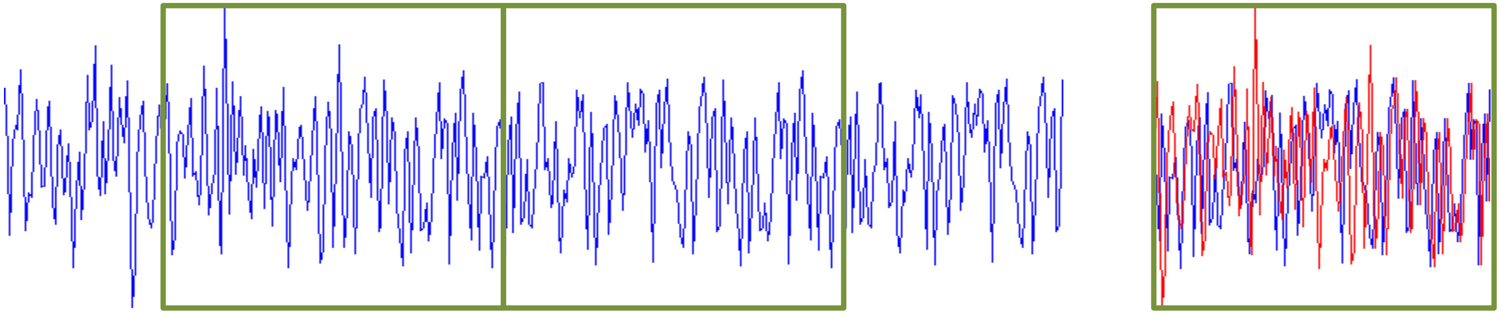




Demodulation

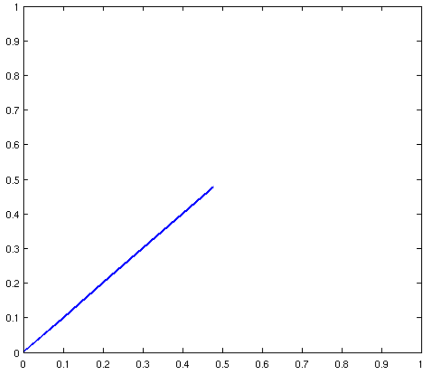
- Time and frequency synchronization
  - Moose/ Schmidl & Cox

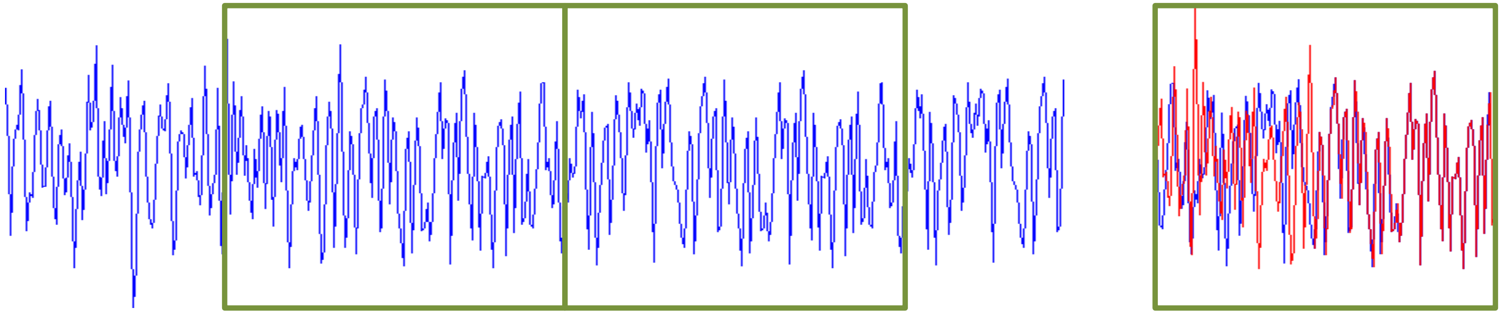




Demodulation

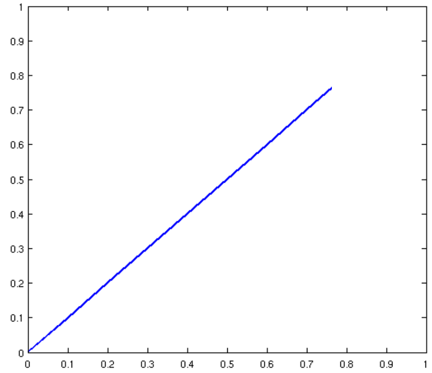
- Time and frequency synchronization
  - Moose/ Schmidl & Cox

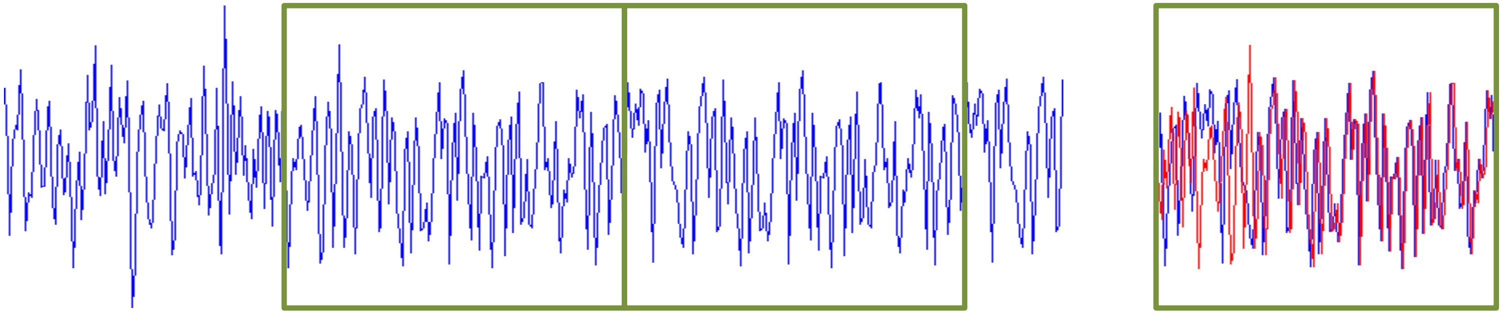




Demodulation

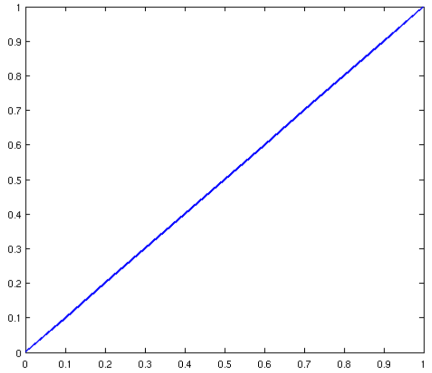
- Time and frequency synchronization
  - Moose/ Schmidl & Cox

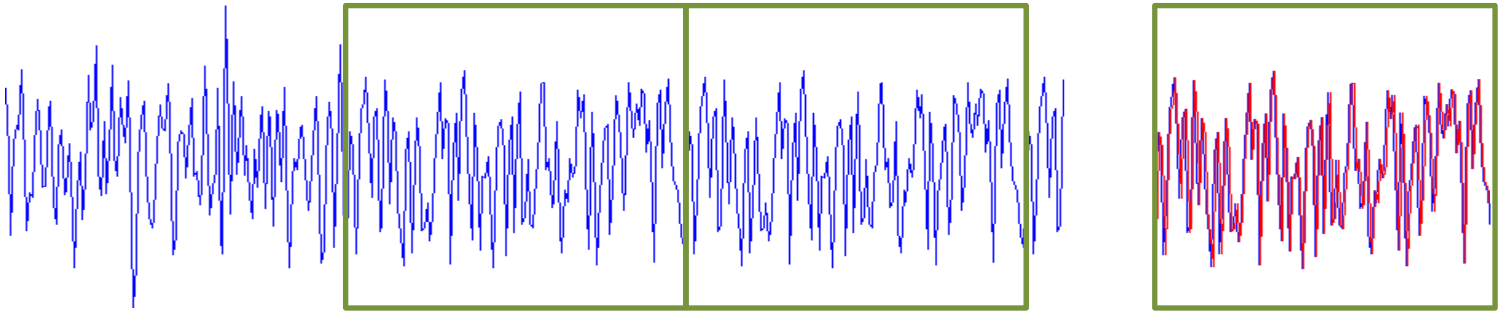




Demodulation

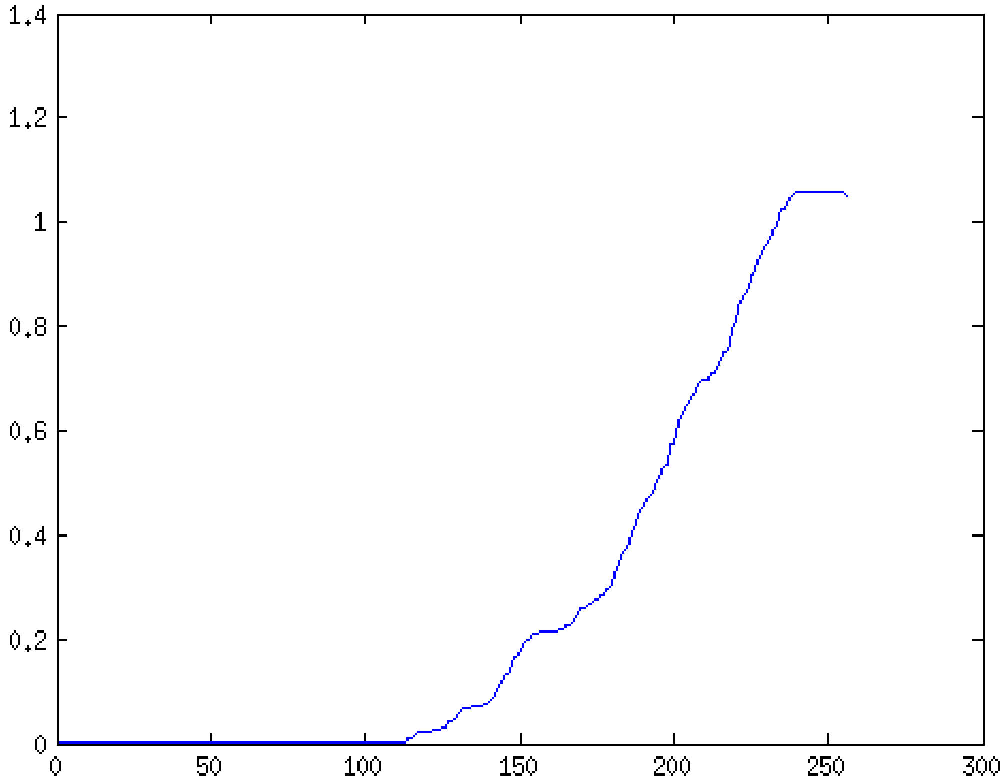
- Time and frequency synchronization
  - Moose/ Schmidl & Cox





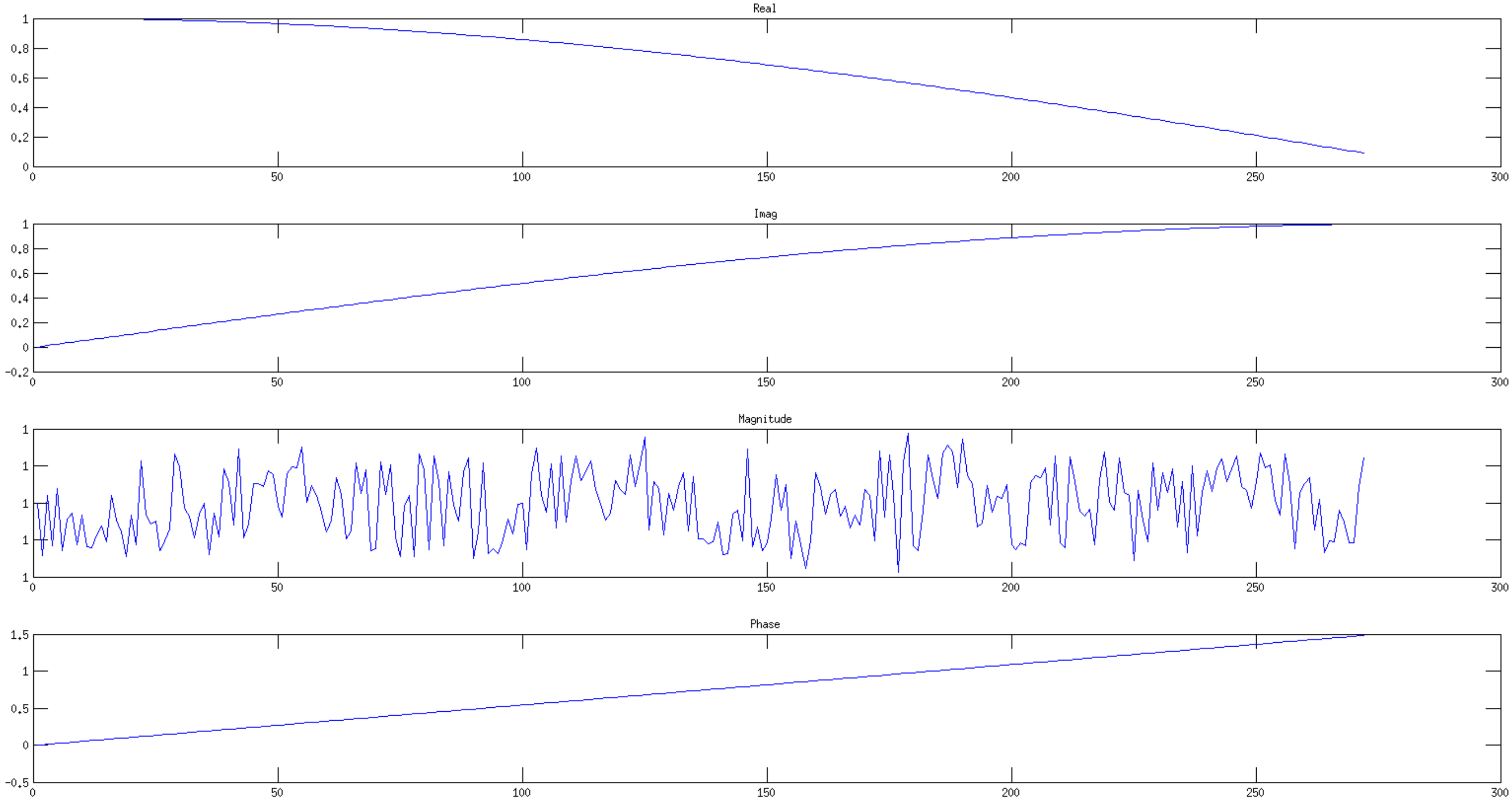
Demodulation

Frame Detector(Time)



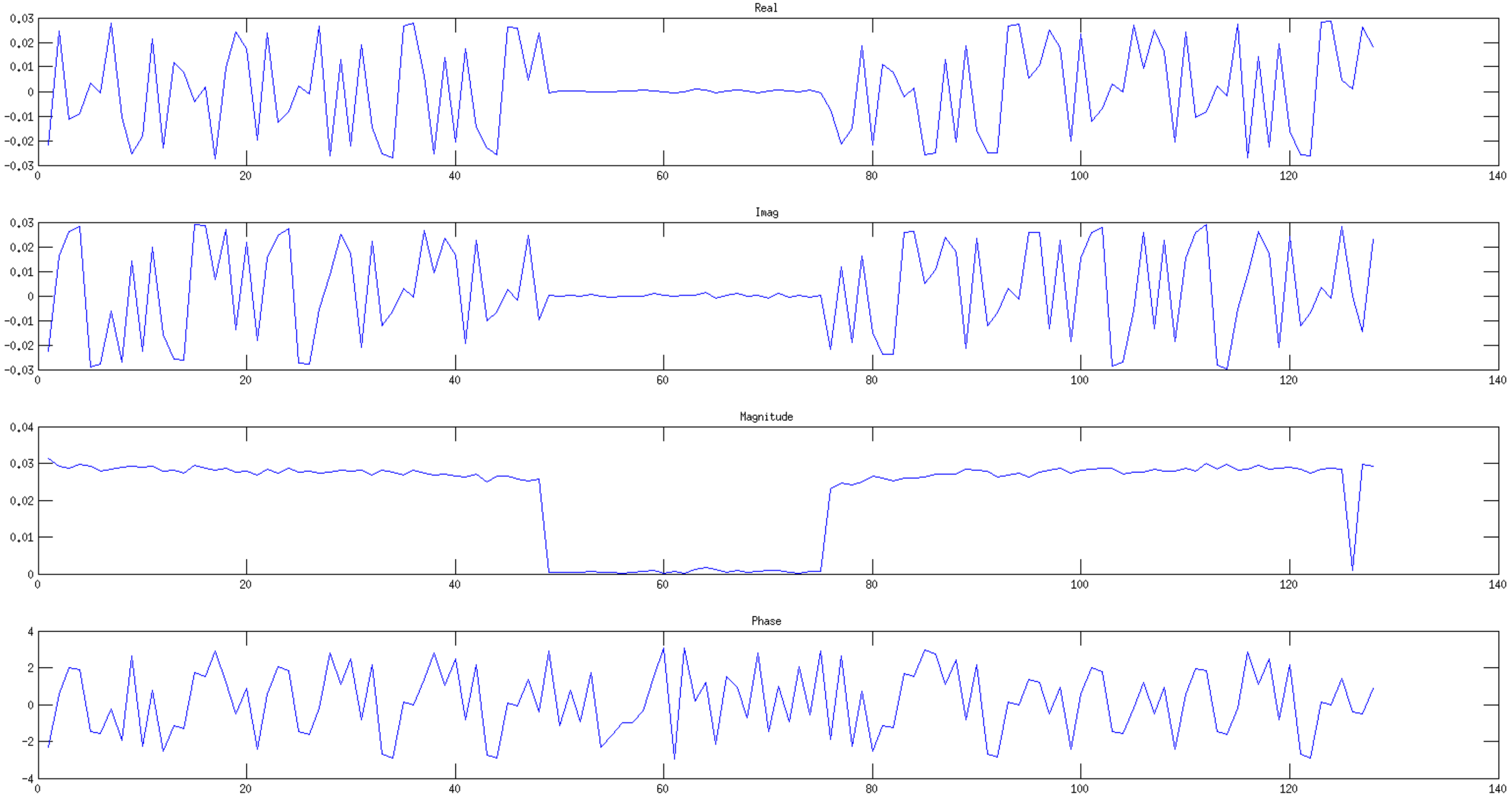
Demodulation

Fine Frequency Offset Correction (Time)



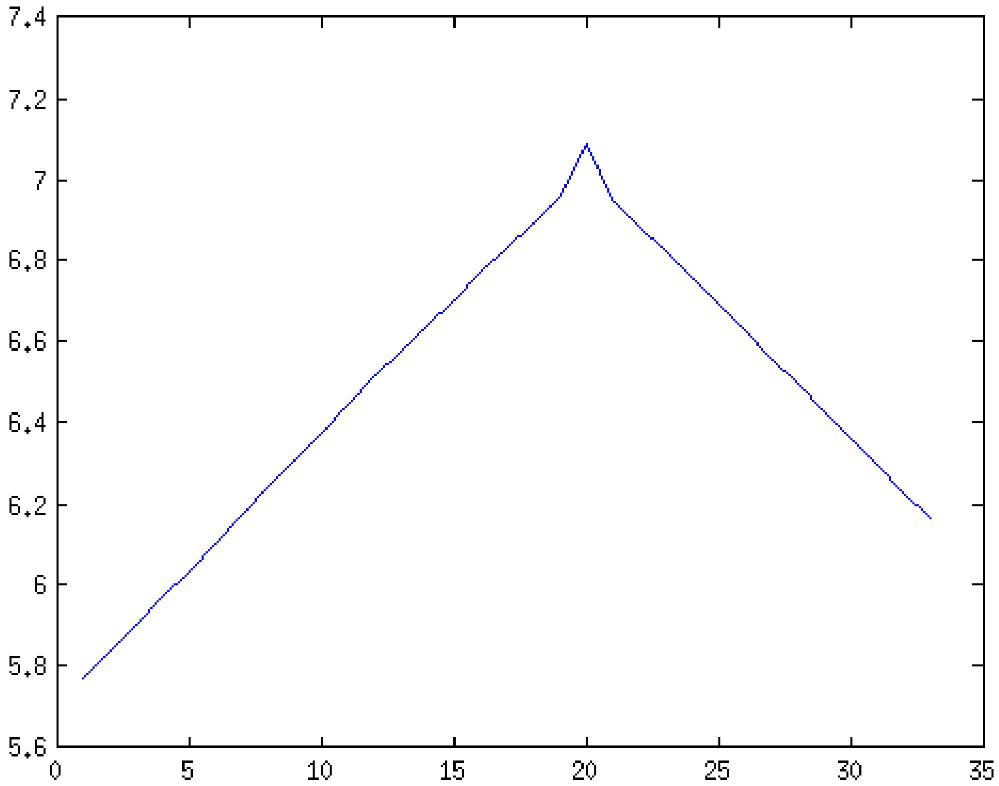
Demodulation

Detected Preamble (Frequency)



Demodulation

Integer Frequency Offset Estimation (Frequency)

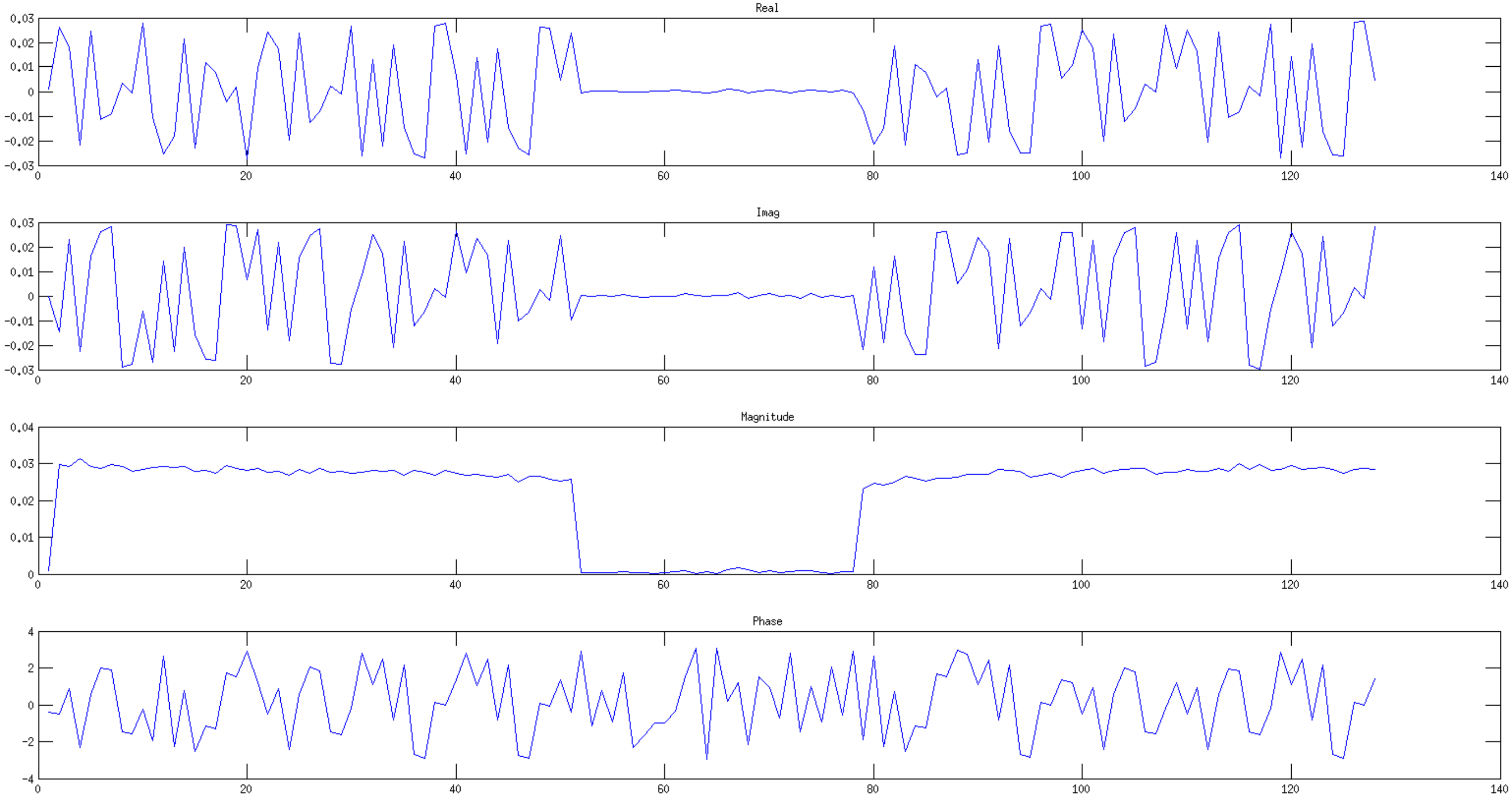


Software Radio Systems

the telecommunications research centre

Demodulation

Frequency-offset Corrected Preamble (Frequency)

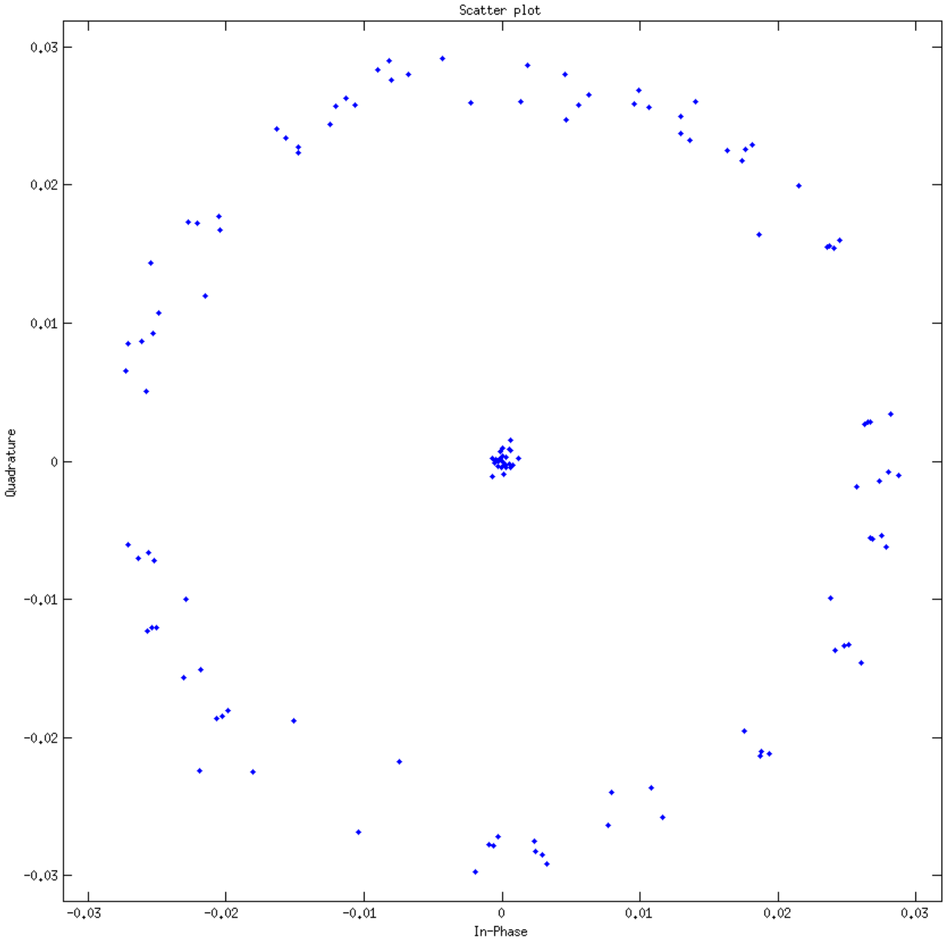


Software Radio Systems

the telecommunications research centre

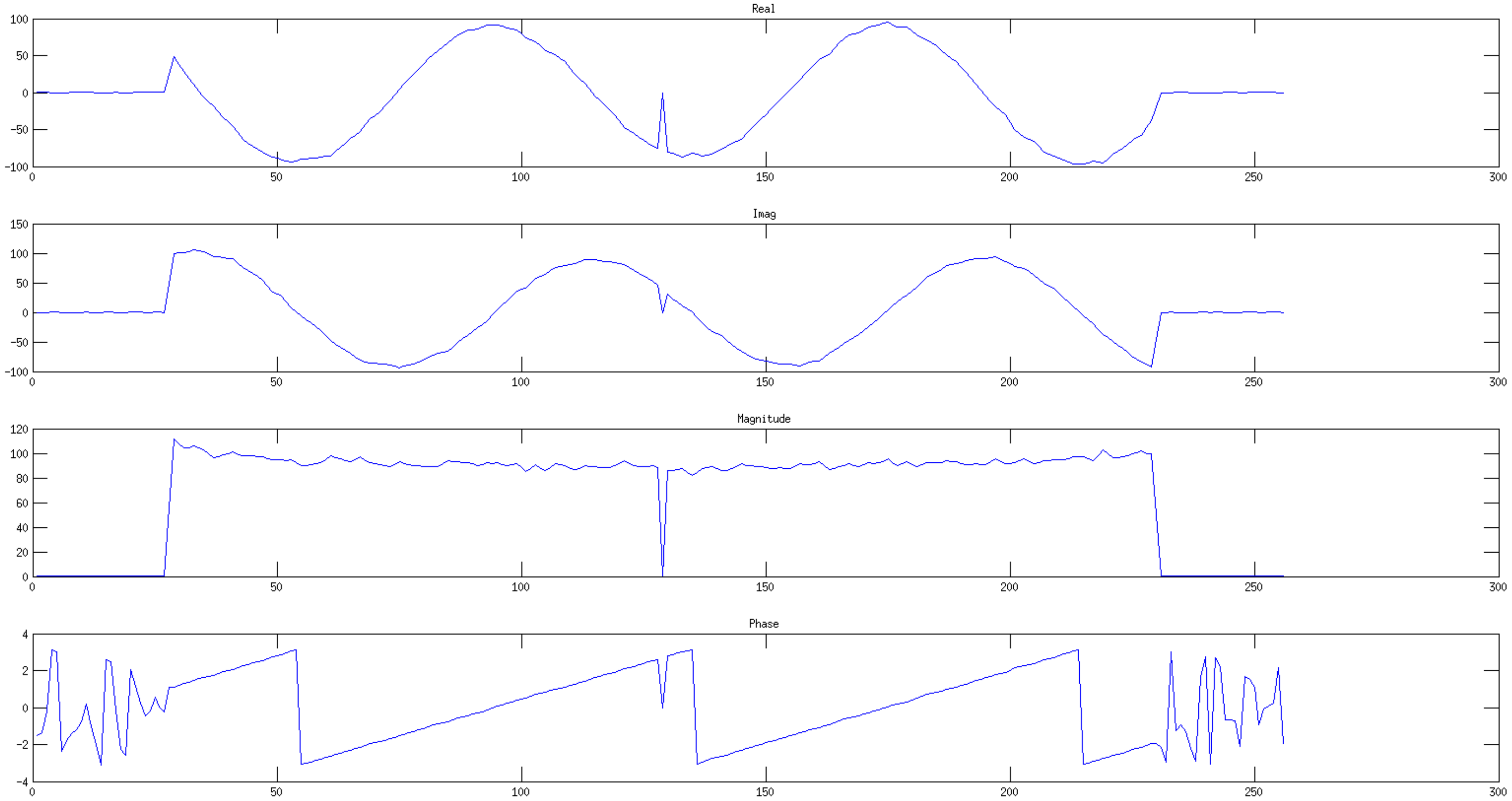
Demodulation

Frequency-offset Corrected Preamble (Frequency)



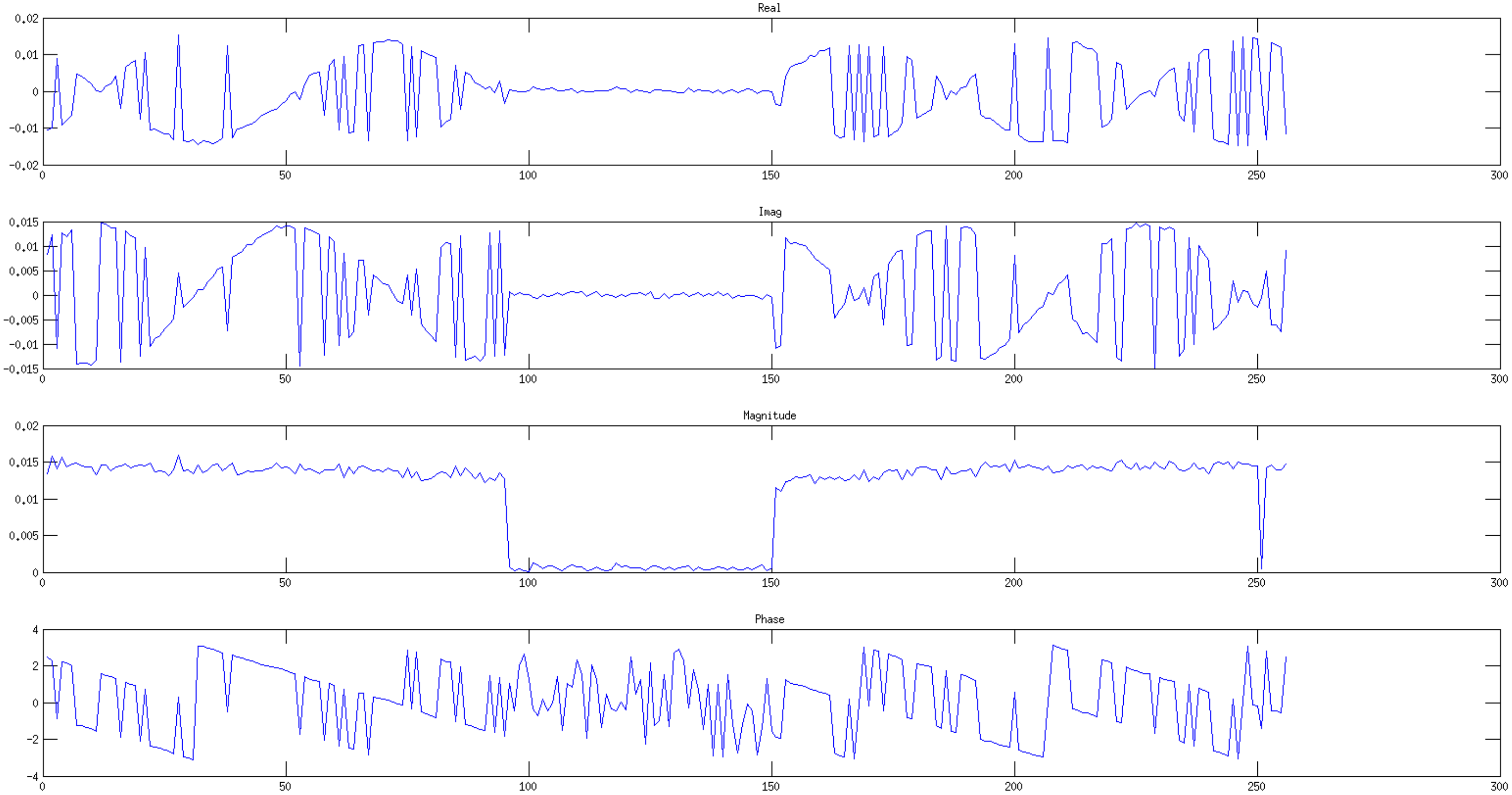
Demodulation

Equalizer (Frequency)



Demodulation

Received Data Symbol (Frequency)

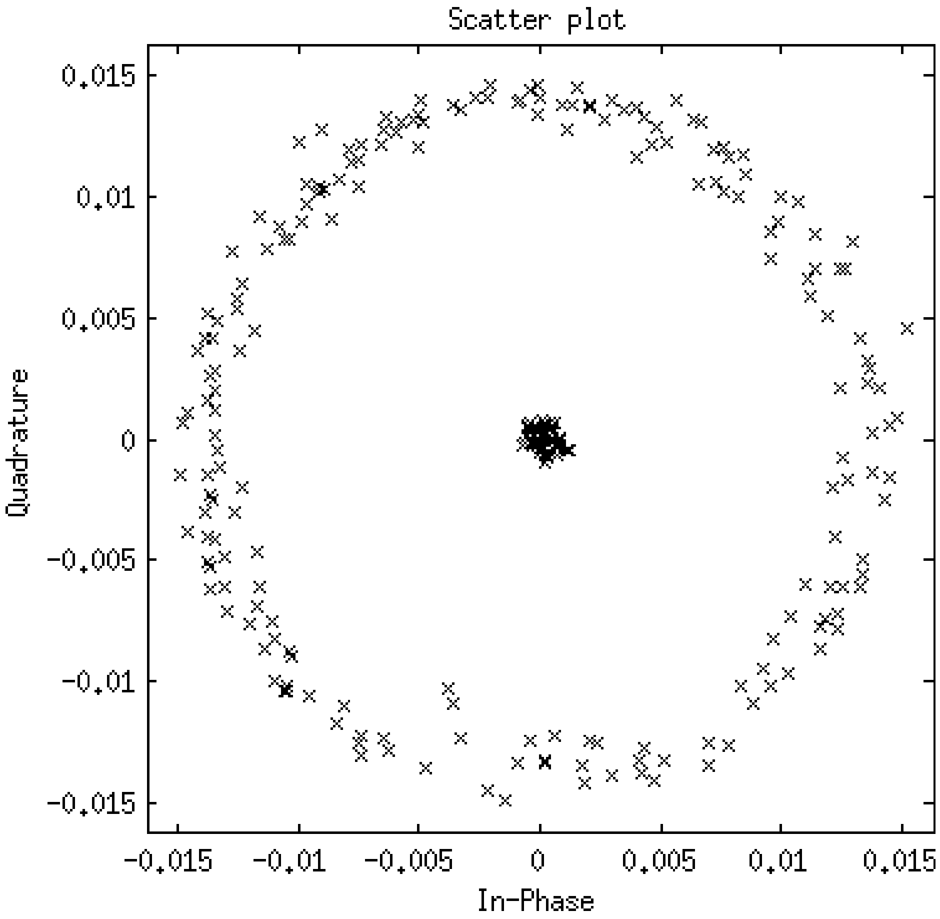


Software Radio Systems

the telecommunications research centre

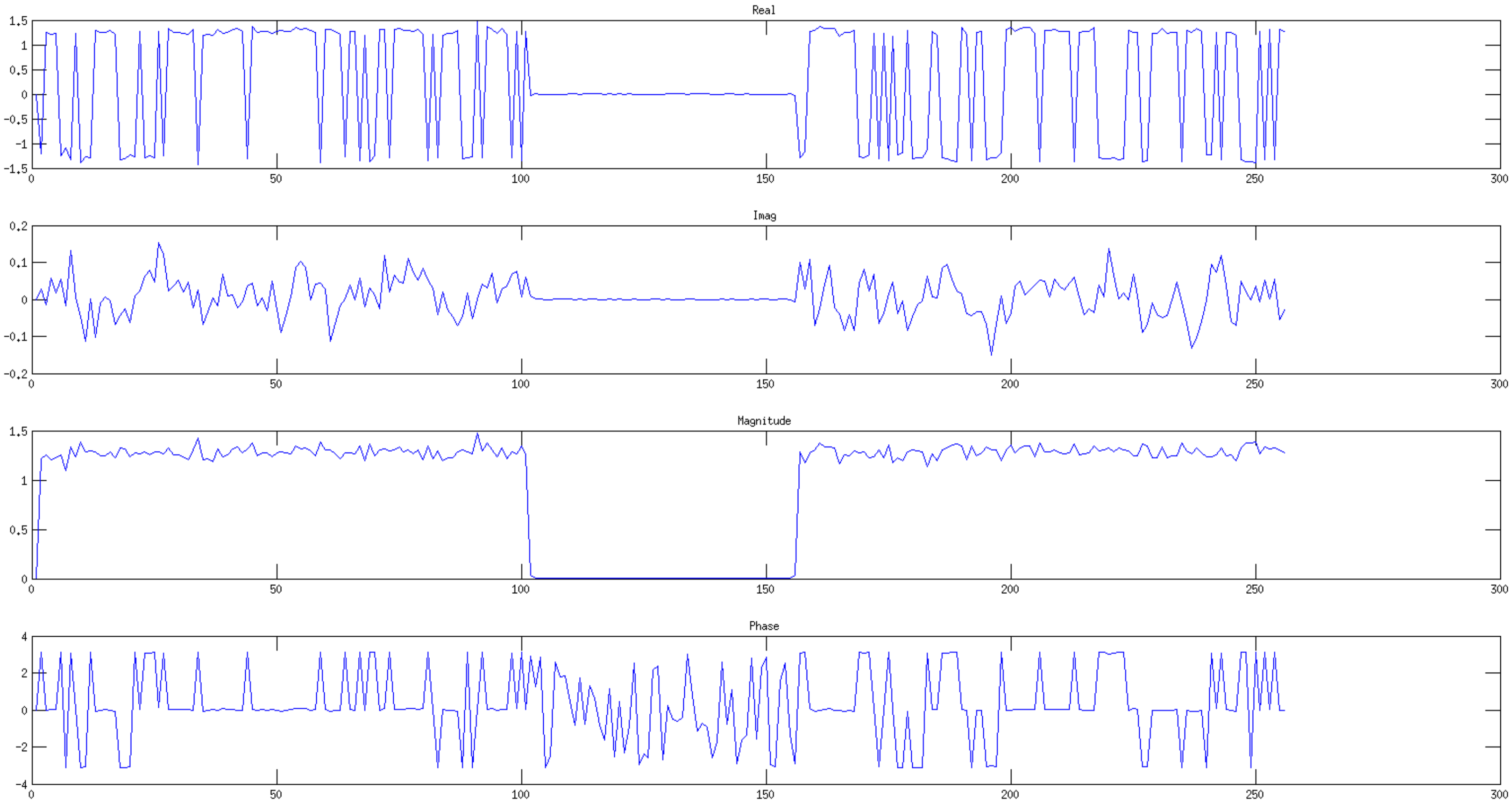
Demodulation

Received Data Symbol (Frequency)



Demodulation

Equalized Data Symbol (Frequency)

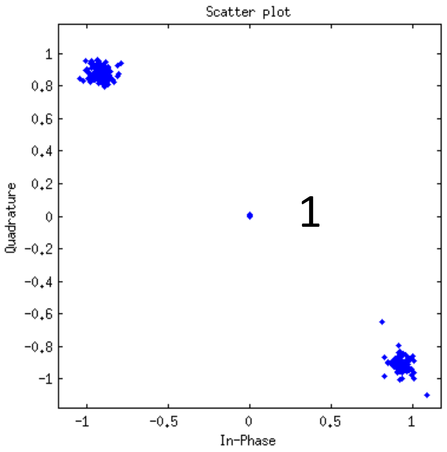


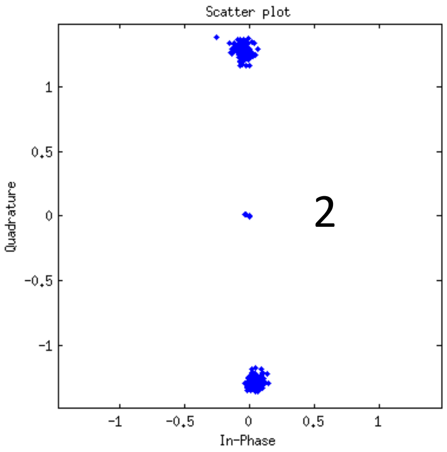
Software Radio Systems

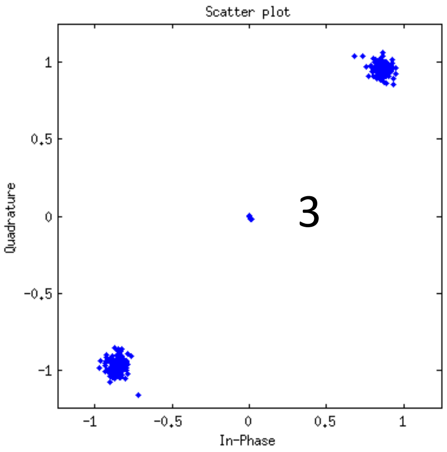
CTVR the telecommunications research centre

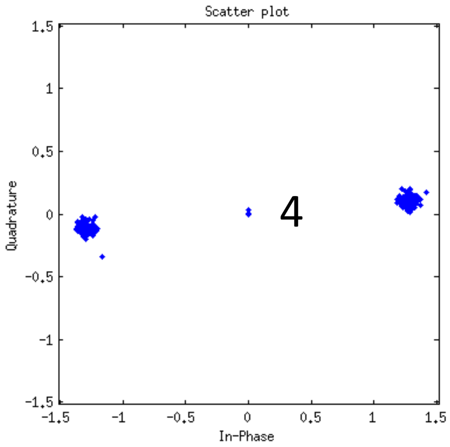
Demodulation

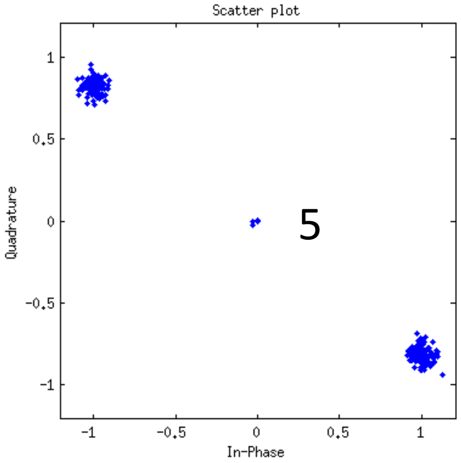
Equalized Data Symbols (Frequency)

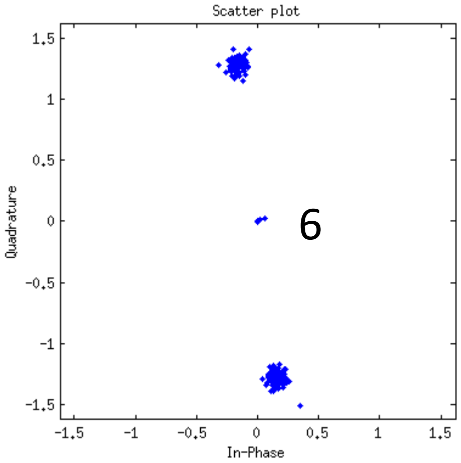


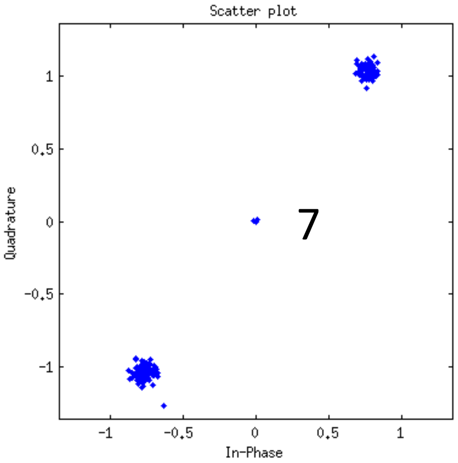






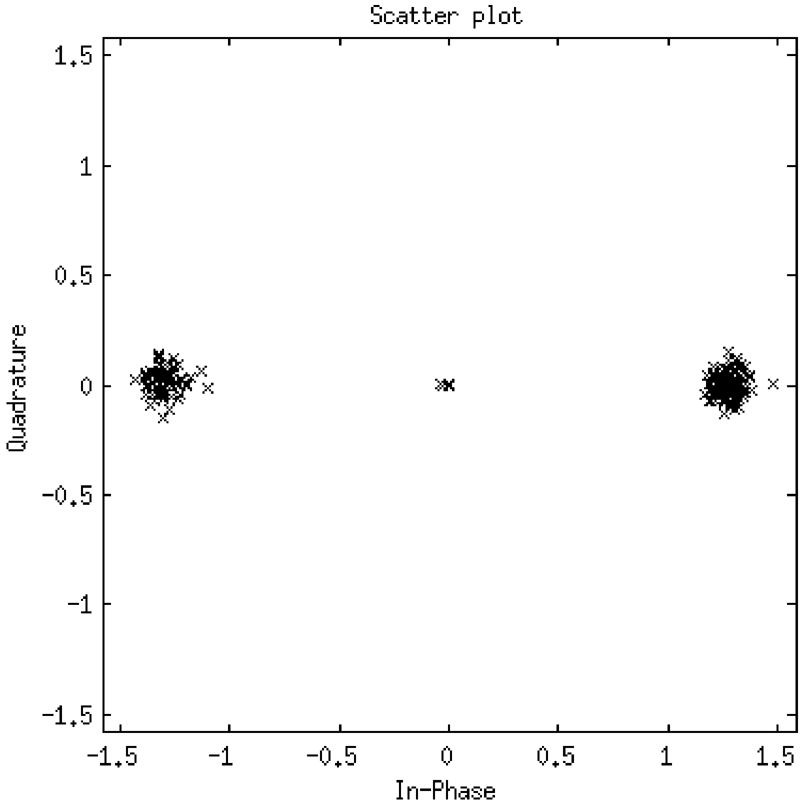






Demodulation

Pilot-Rotated Data Symbols



Interesting Applications

- Iris Overview
- Iris Architecture
- Getting Started
- Controllers
- Case Study - OFDM
- Interesting Applications

Interesting Applications

Interesting Applications

DARPA Spectrum Challenge

“A competition to demonstrate a radio protocol that can best use a given communication channel in the presence of other dynamic users and interfering signals”

Interesting Applications

DARPA Spectrum Challenge

"A competition to demonstrate a radio protocol that can best use a given communication channel in the presence of other dynamic users and interfering signals"

- Use a standardized radio hardware platform (USRP N210).

- Head-to-head competitions between your radio protocol and an opponent's in a structured testbed environment.

- The best strategies for guaranteeing successful communication in the presence of other competing radios will win.

Interesting Applications

DARPA Spectrum Challenge

Multiple Phases:

- Qualification
- Wildcard selection
- Tournament
  - Competitive
  - Cooperative

Interesting Applications

DARPA Spectrum Challenge

Qualification:

- Single radio pair (Transmitter – Receiver)
- Take input data from a source, packetize, transmit and receive.
- 2.5MHz band to operate in.
- Total number of correct packets received in 5 minutes.
- 3 types of possible interference (random time sequence):
  - N0 = one second period of no interference
  - N1 = one second period of short-term 1.25MHz band-limited white noise interference signal that resides in the lower half of the 2.5MHz band
  - N2 = one second of short-term 1.25MHz band-limited white noise interference signal that resides in the upper half of the 2.5MHz band.

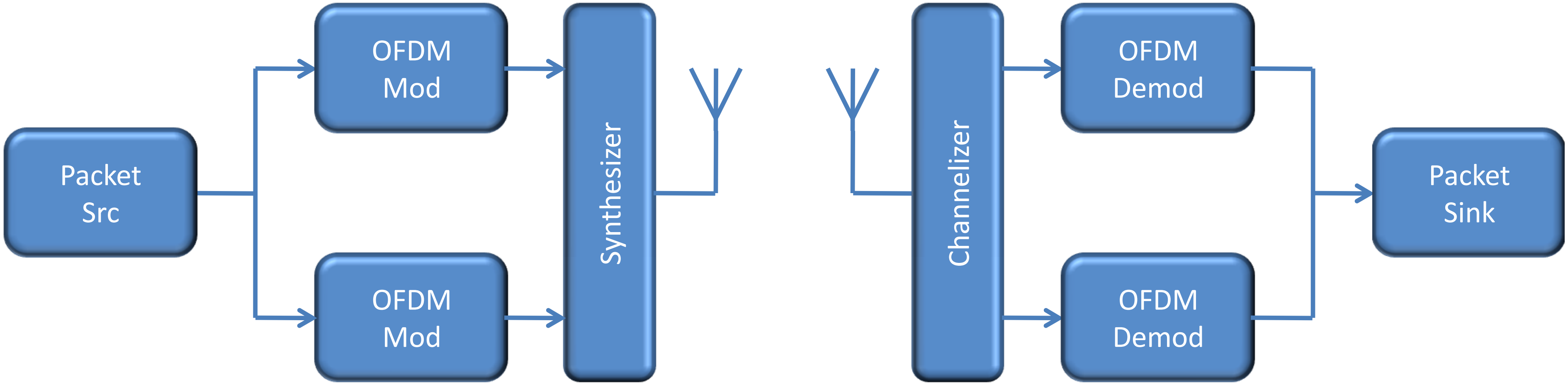
Interesting Applications

System Design?

- Simplex/Duplex?
- Robust waveform or detect & reconfigure?
- Single/Multi-carrier?
- Channelization?

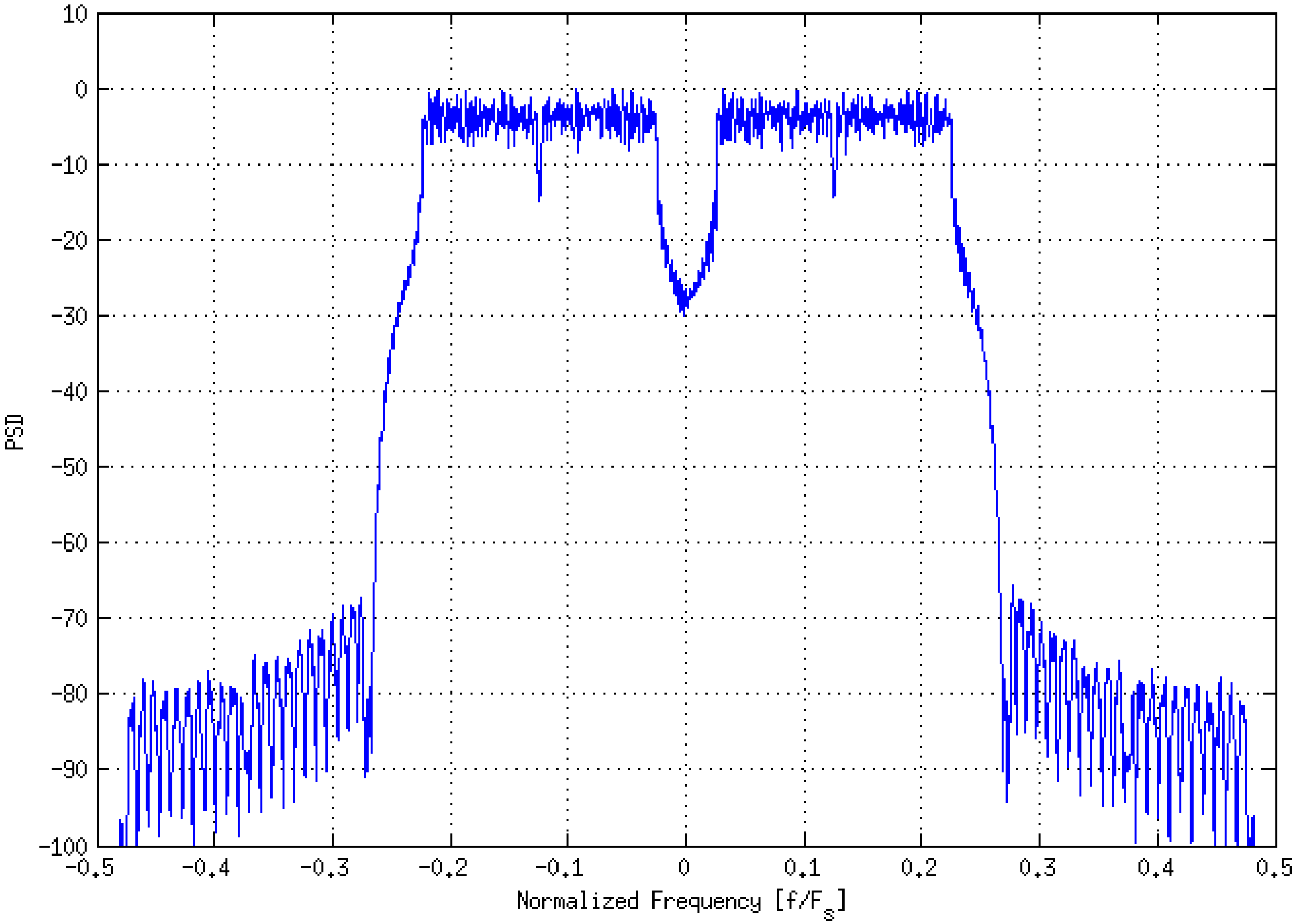
Interesting Applications

DARPA Spectrum Challenge



Interesting Applications

DARPA *Spectrum* Challenge



Interesting Applications

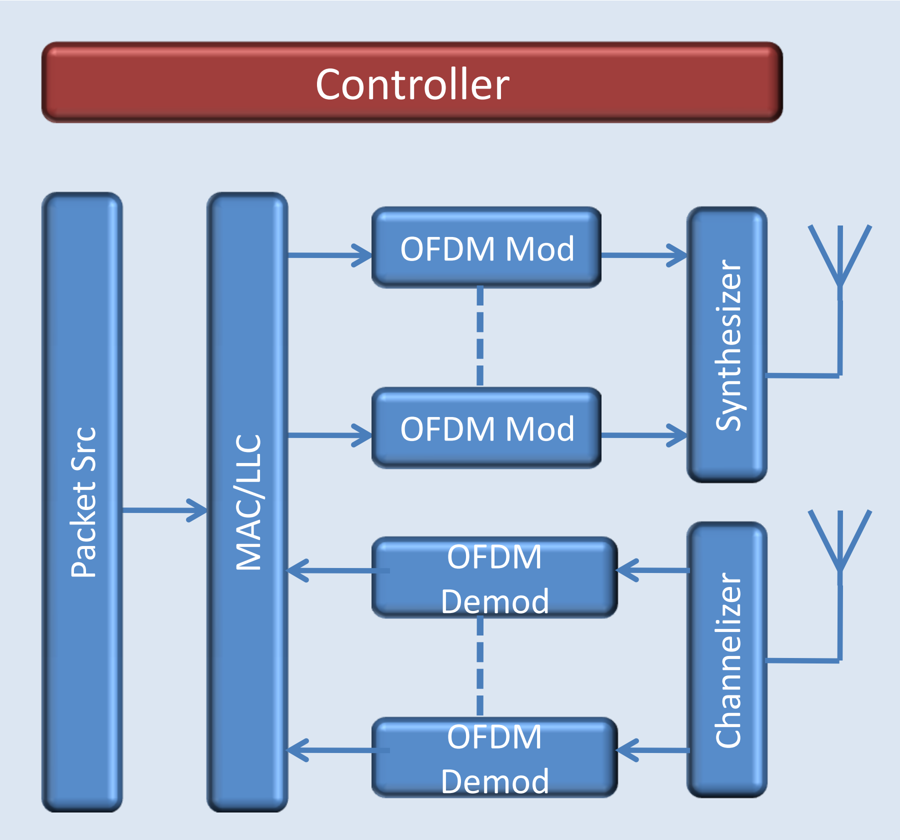
DARPA Spectrum Challenge

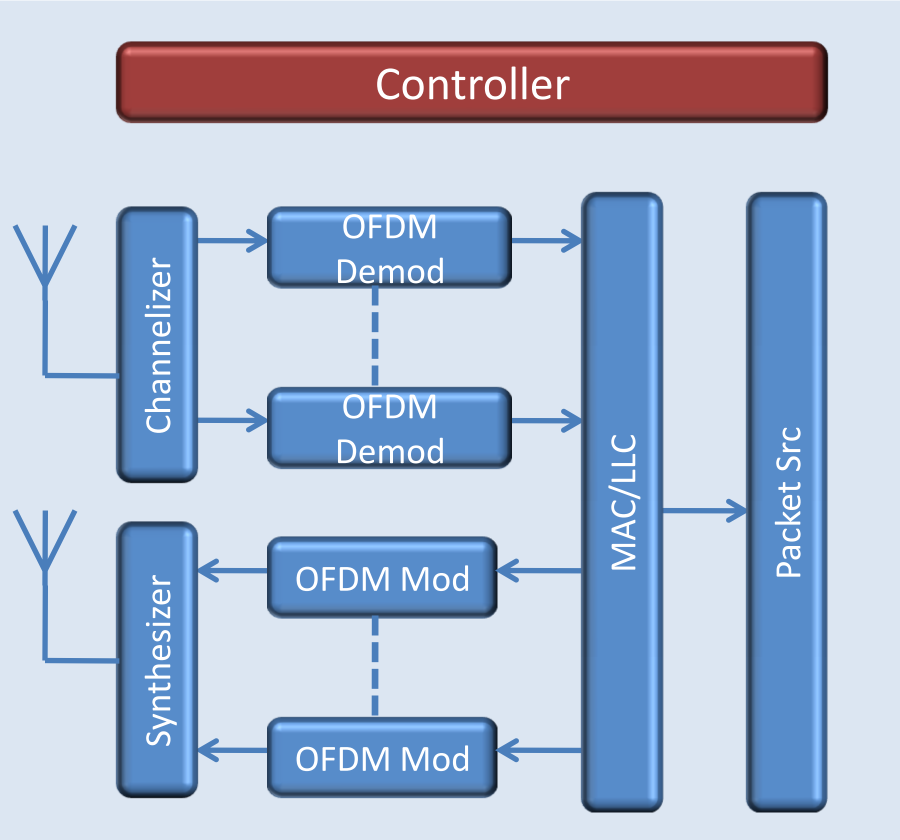
Wildcard Selection:

- Single radio pair (Transmitter – Receiver)
- Tested against "house radios" and other possible interferers.
- Transfer a data file *without errors* as fast as possible.
- Competitive match
  - Tested against single house radio pair.
  - Fastest team wins.
- Cooperative match
  - Tested with two house radio pairs.
  - Weighted average of time taken and the number of error-free packets received by *each radio pair*.

Interesting Applications

DARPA Spectrum Challenge





Interesting Applications

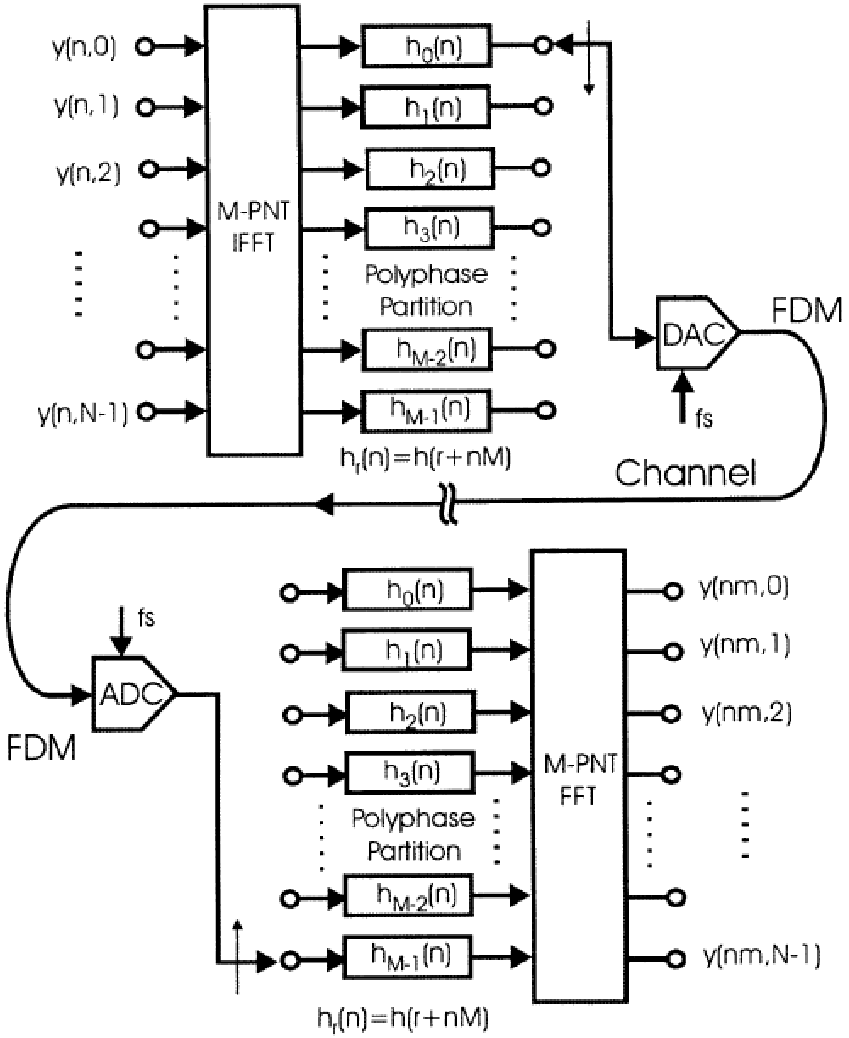
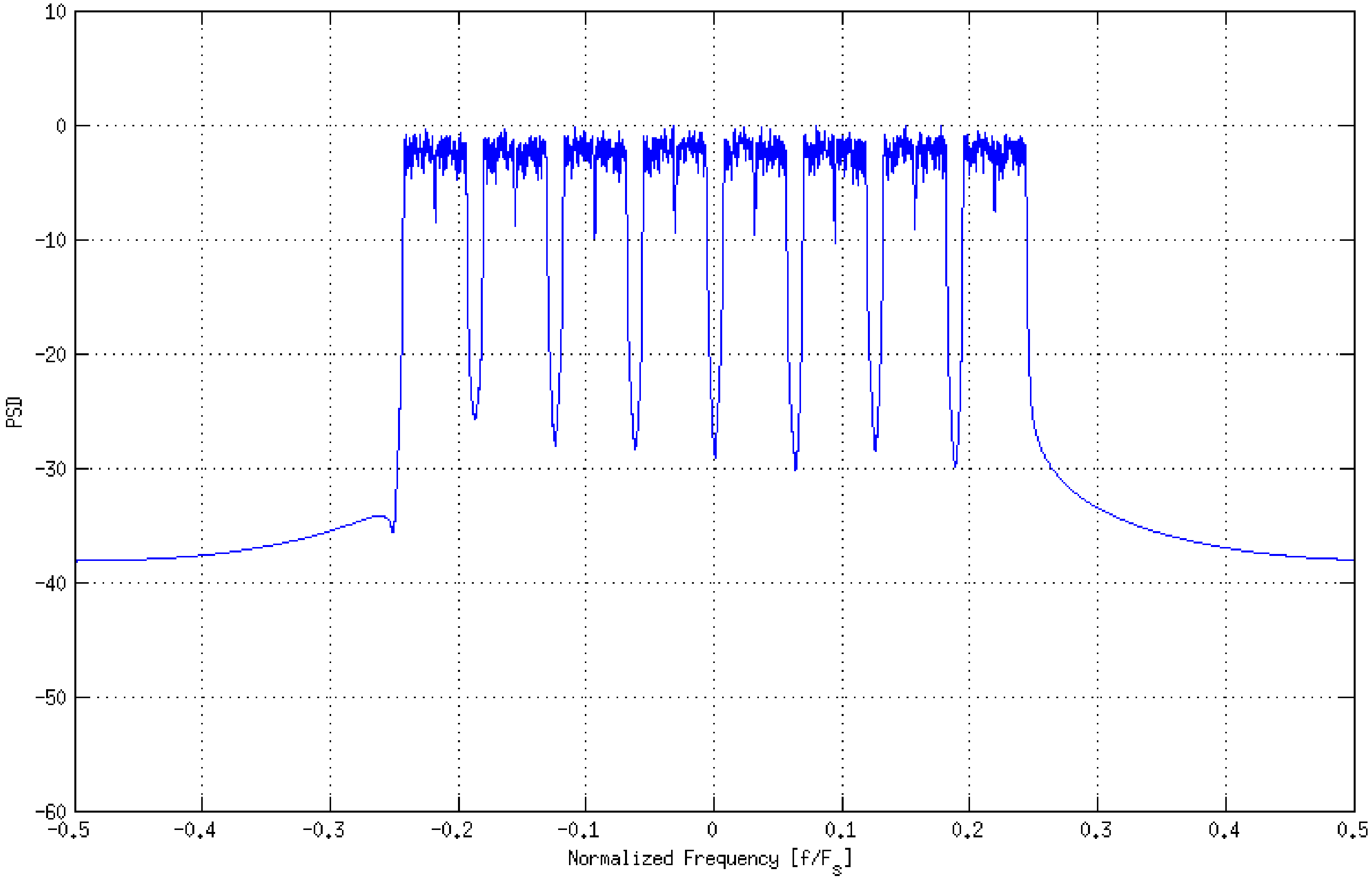


Fig. 27. N-channel transmitter and N-channel receiver. Dual circuits formed with polyphase filters, FFT, and commutator.

Digital Receivers and Transmitters Using Polyphase Filter Banks for Wireless Communications
Fredric J. Harris, Chris Dick and Michael Rice
IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES, VOL. 51, NO. 4, APRIL 2003

Interesting Applications

DARPA Spectrum Challenge



Why use Iris?

Why use Iris?

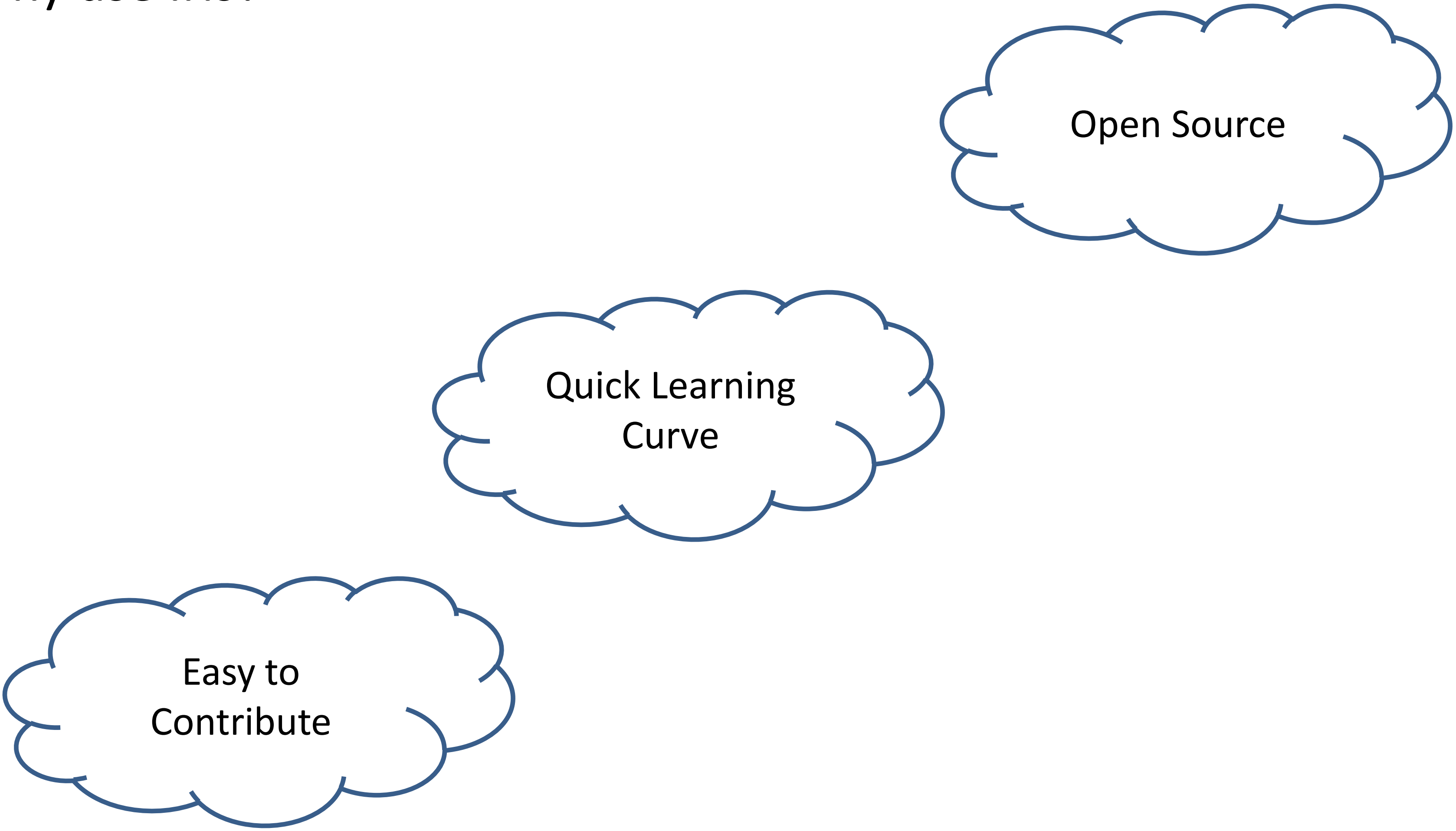
Quick Learning Curve

Why use Iris?

Open Source

Quick Learning Curve

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

Open Source

Quick Learning Curve

Easy to Contribute

Small Project

Try it out

https://github.com/softwareradiosystems

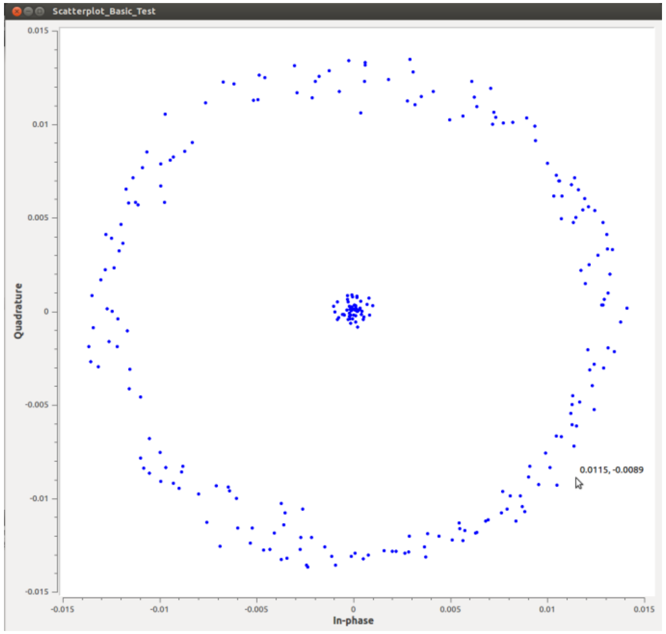
Thank you

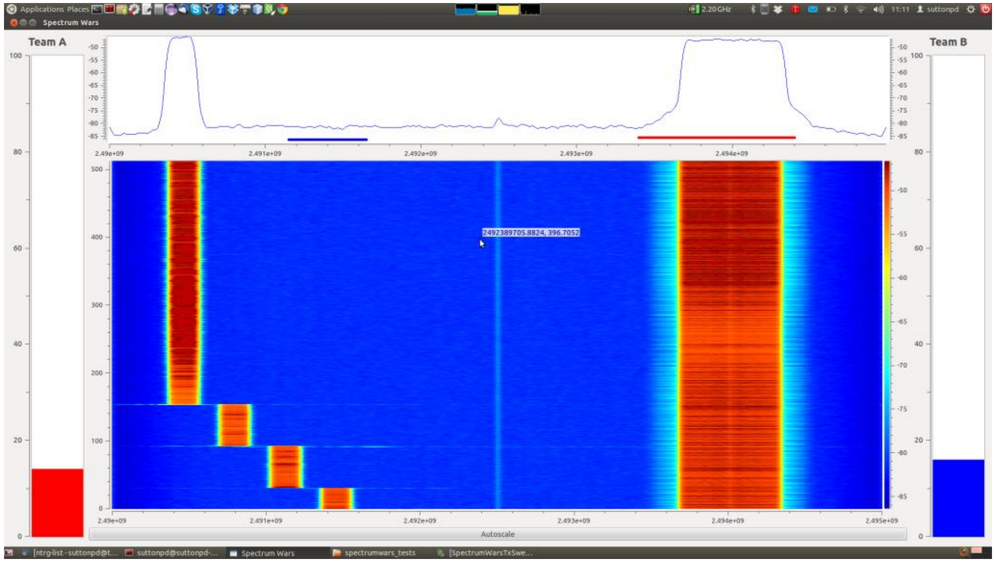
suttonpd@tcd.ie

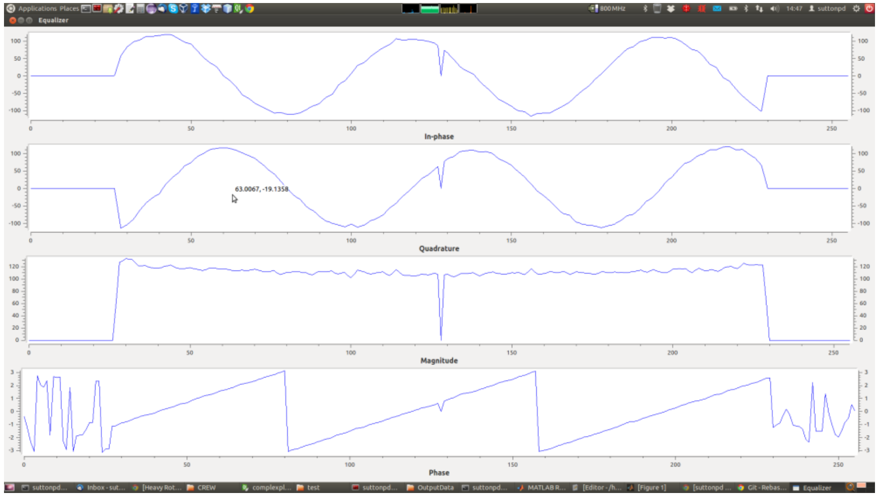
paul@softwareradiosystems.com

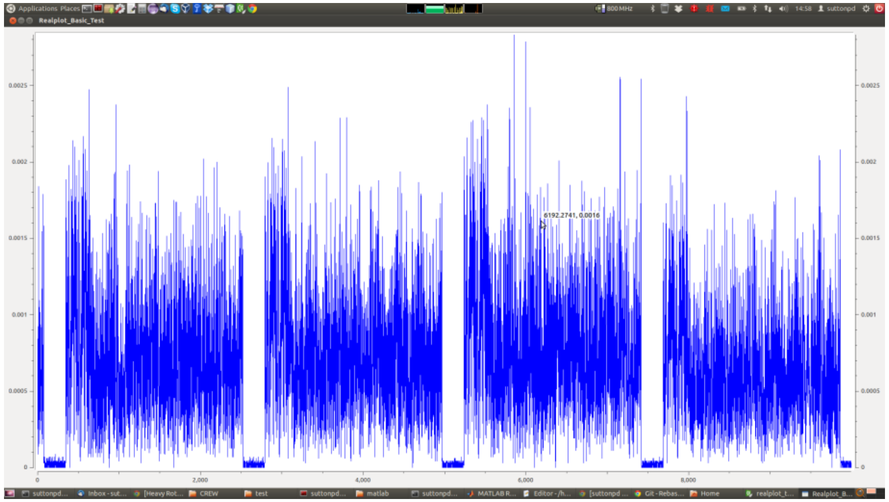
Additional Material

Release 1.1.0







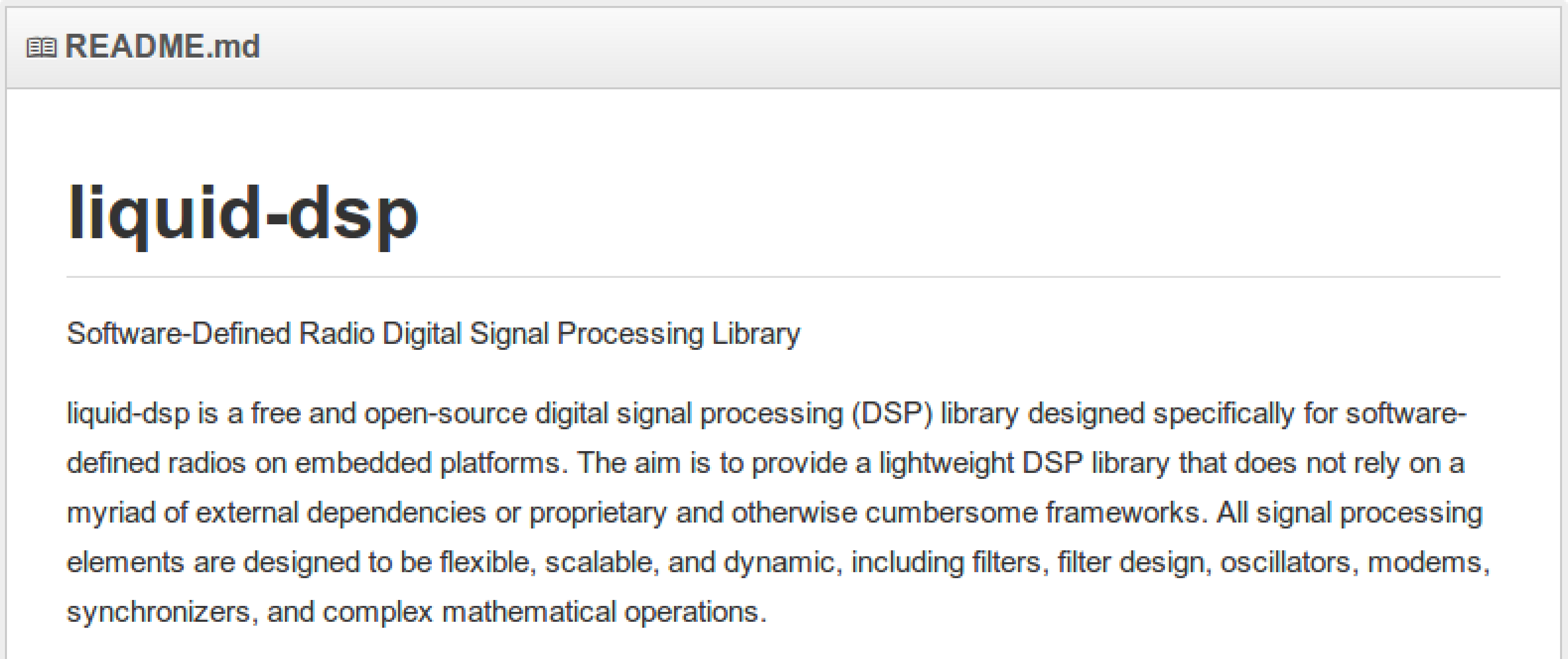


Software Radio Systems

the telecommunications research centre

Release 1.1.0

- Liquid-DSP Components



README.md

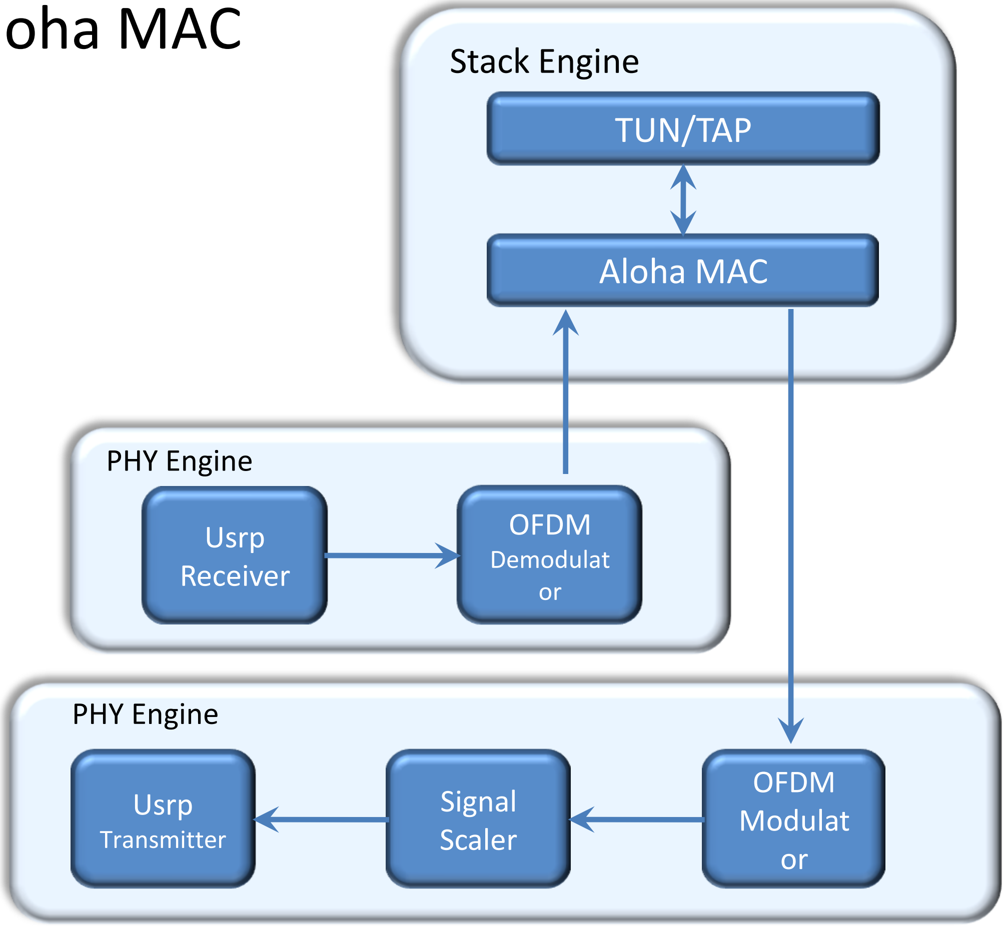
liquid-dsp

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.

MAC Development with Iris

- Simple Aloha MAC



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

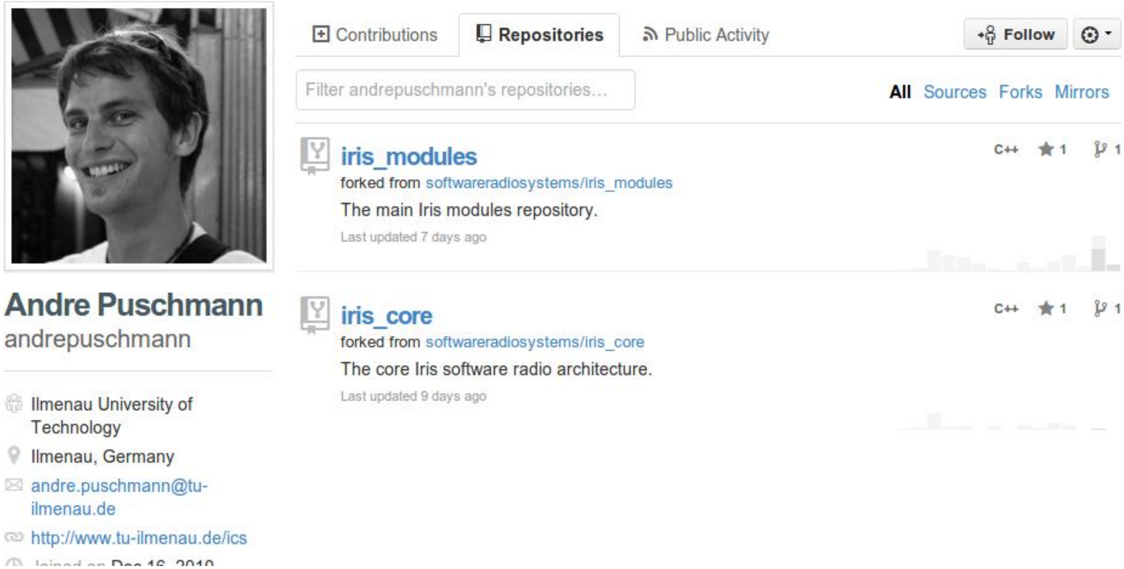
MAC Development with Iris

- 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
Joined on Dec 16, 2010

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