



A MIMO Modeling Framework Using a Software Defined Radio Paradigm

Angel Camelo Vasquez
Advisor: Dr. Domingo Rodriguez
University of Puerto Rico
July 2012

Edited by D. Rodriguez – Sept. 2012

Outline

- Motivation
- Objectives
- Background
- Methodology
- Design
- Implementation
- Testing
- Conclusions

Motivation



Motivation

- Multiple Input Multiple Output (MIMO): use of multiple antennas, both, at transmitter and receiver.
 - MIMO is a trending technology into the signal communications realm.
 - Actual implementations with 3GPP, WiMAX, WiFi.
- Software Defined Radio
 - Radio implemented by means of software.
- Time-frequency Representations

Objectives

- Design, implement, and test a framework for MIMO channel monitoring and simulation
 - Redesign the existent **S**ignal **R**epresentation **LAB**oratory application software named **SIRLAB**.
 - Implement time-frequency Representations.
 - Provide a MIMO simulation.

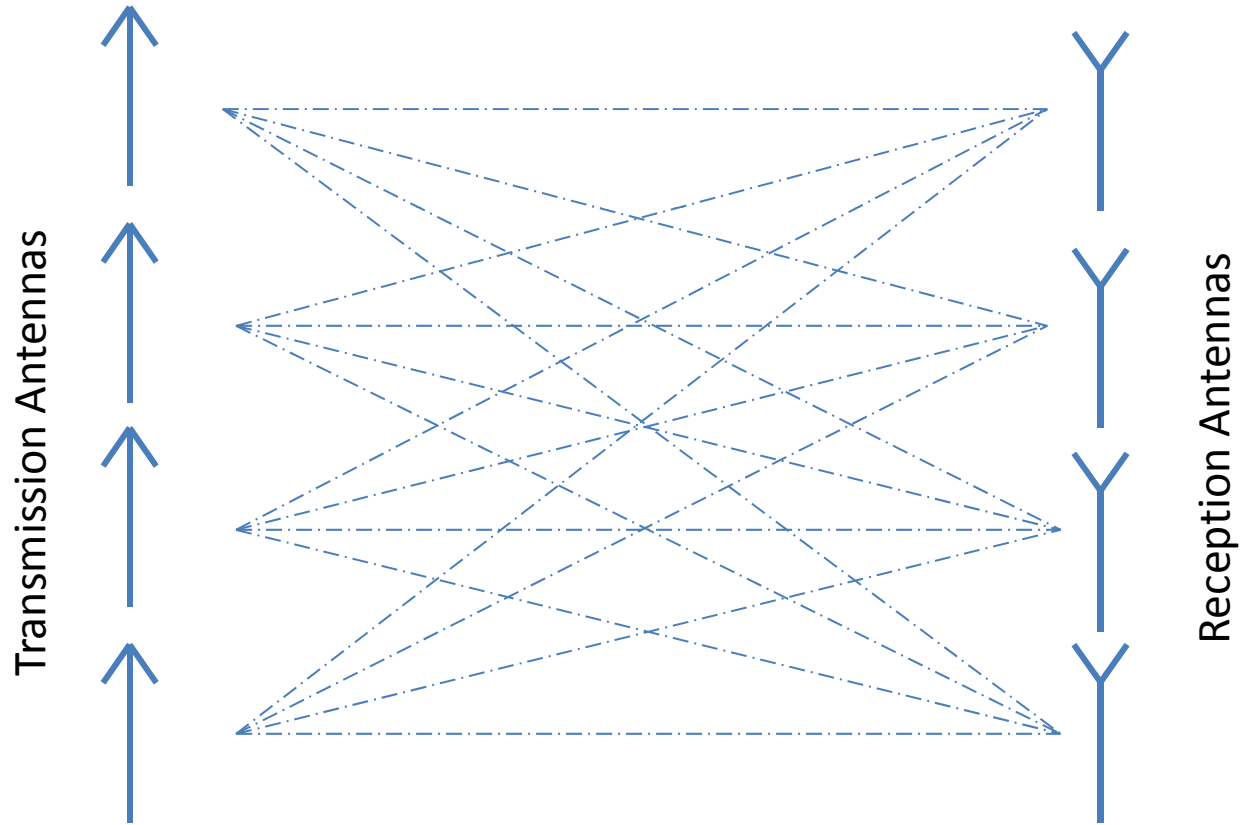
Background

- Communications
- Software Defined Radio
- Time-Frequency Representations

Communications

- Communications
 - Send multiple symbols with different antennas.
 - The signal is modified by the channel and possibly reflected by several scatterers.
 - Multiple copies of the signal may be received and processed.

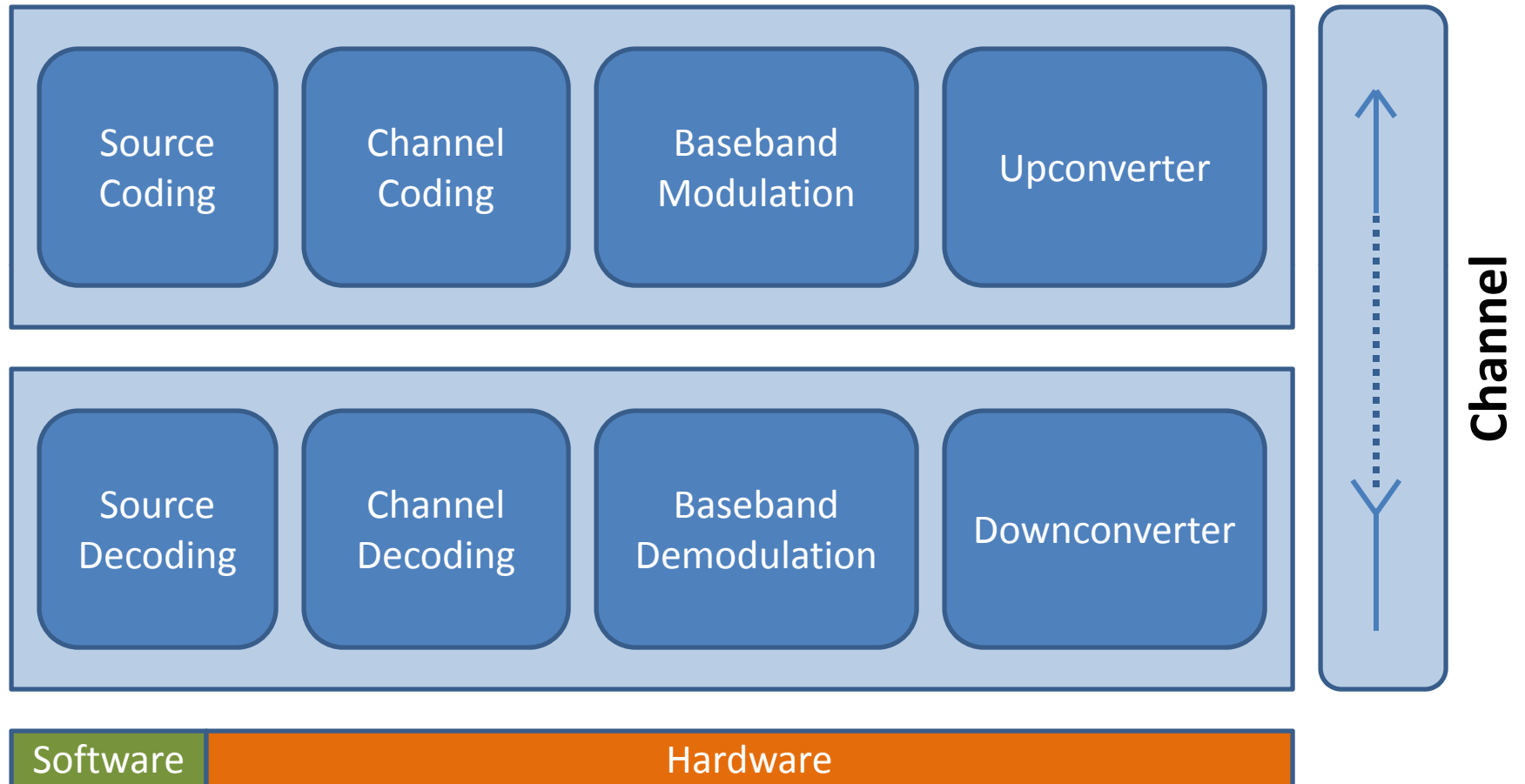
Communications Channel



Software Defined Radio

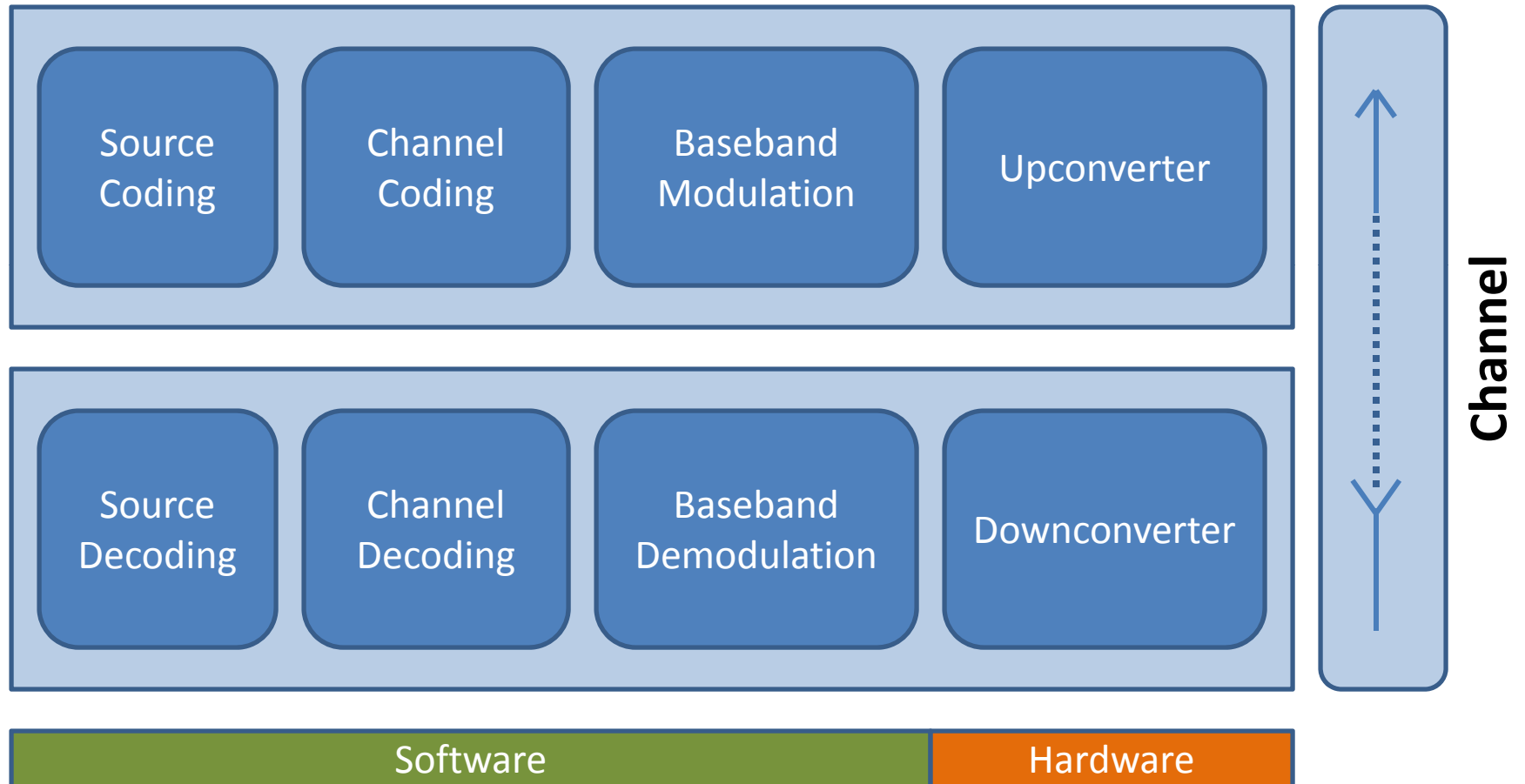
- Like traditional radio
 - It does what a typical radio does.
- Unlike traditional radio
 - Several components have been implemented computationally.
- Software can perform jobs that before were done by hardware.
- Extending the software towards the antenna

Traditional Radio



Traditional Radio Paradigm

Software Defined Radio



SDR Paradigm

Software Defined Radio Tools

- SoRa, by Microsoft
- Simulink, by MATLAB
- GNUradio, as an Open Source

Software Defined Radio Implementation

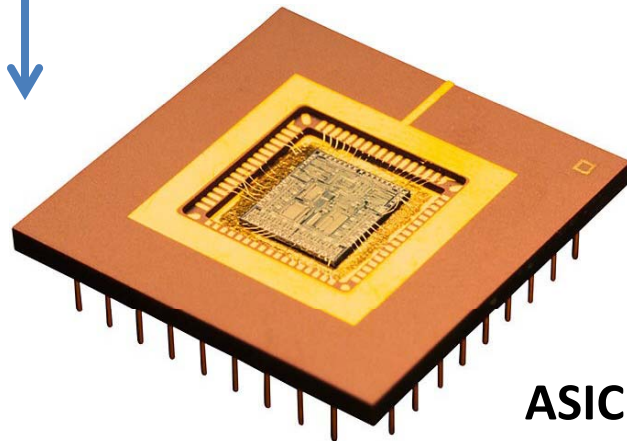
```
#include "civclient.h"

/* this is used in strange places,
   needed (hence, it is not 'extern')
bool is_server = FALSE;

char *logfile = NULL;
char *scriptfile = NULL;
static char tileset_name[512] = "\0";
char sound_plugin_name[512] = "\0";
char sound_set_name[512] = "\0";
char server_host[512] = "\0";
char user_name[512] = "\0";
char password[MAX_LEN_PASSWORD] = "\0";
char metaserver[512] = "\0";
int server_port = -1;
bool auto_connect = FALSE; /* TRUE = skip "C
bool in_ggz = FALSE;
```



FPGA



ASIC



Personal Computer

Software Defined Radio Implementation



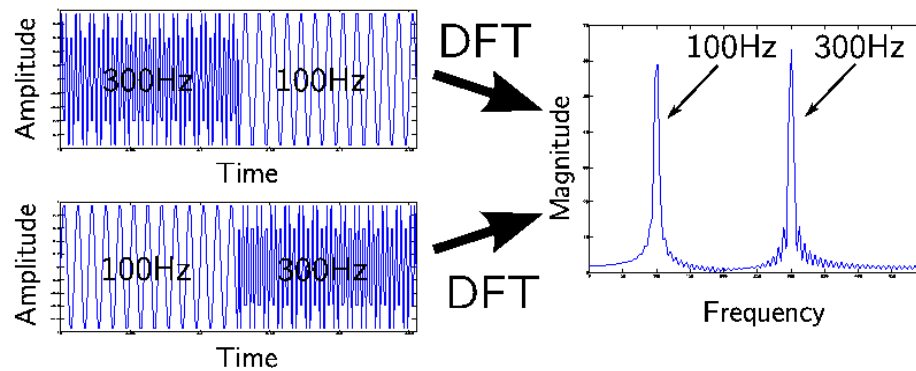
Personal Computer

Time-Frequency Representations

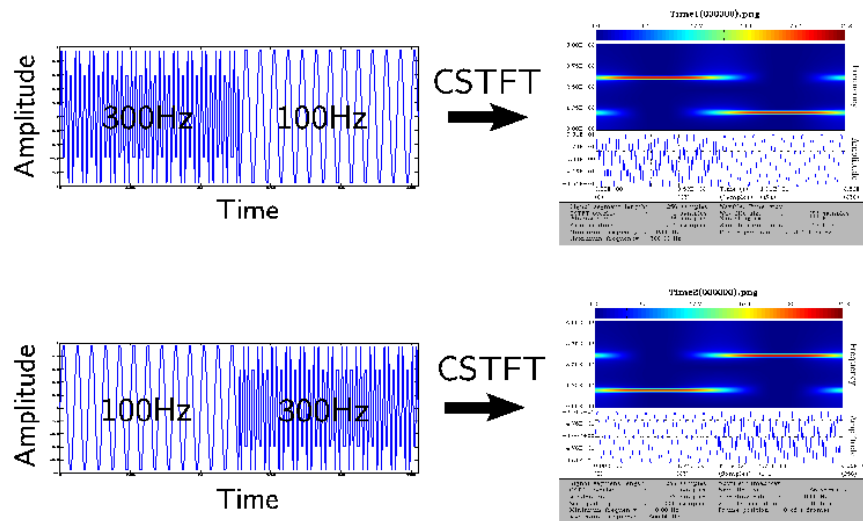
- Techniques and methods
- Signals whose characteristics are changing with time
- Allow to extract additional information
- High computational requirements

Time-Frequency Representations

**NO
DISCRIMINATION**



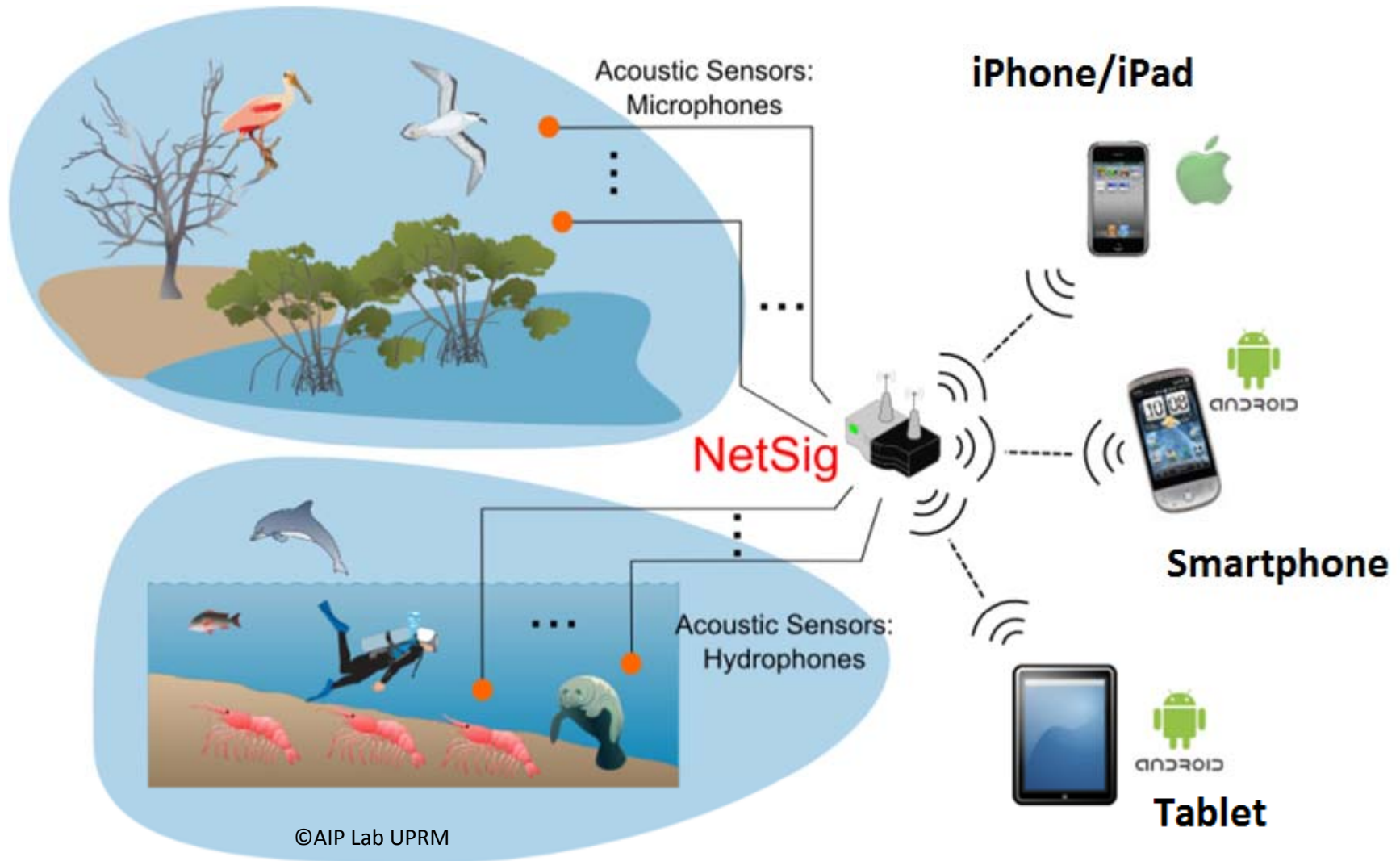
DISCRIMINATION



Methodology

- Available tools
 - SIRLAB, by the AIP Lab
 - GNUradio, as an Open Source
- Preliminary Results
 - SIRLAB Parallelization
 - User Integration: webSIRLAB, SIRDroid
- Framework for MIMO channel monitoring and simulation
 - Design
 - Implementation
 - Testing

SIRLAB on a NetSig Node Processor



SIRLAB

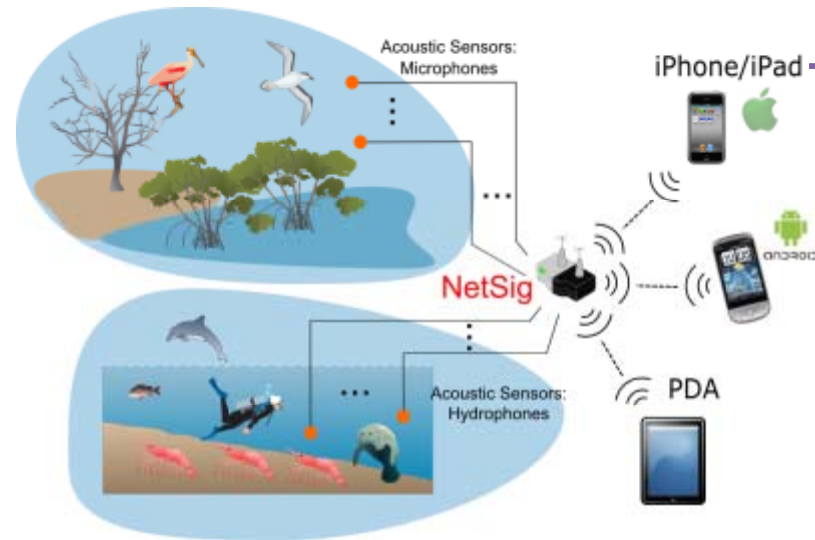
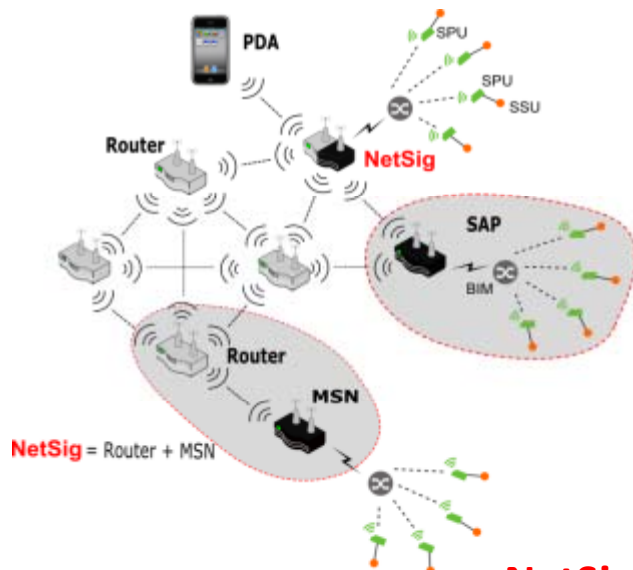
SIRLAB: A Tool for Research and Education

- **SIRLAB: Signal Representation LABoratory**
- Open-Source Framework Developed at the [AIP Lab](#)
- Research Tool for [Time-Frequency Analysis](#)
- Educational Tool for [Signal Representation](#)
- Based on [OpenCV](#) and [FFTW](#) Software Libraries
- Uses [R. Tolimieri's Theory on Time-Freq. Repres.](#)
- **Input Format: [WAV File](#)**
- **Language Written: [C/C++](#)**
- **Speed Up Time: 20+ Times Faster than Most Known MATLAB-based Time-Frequency Tools**

- Known Time-Frequency (T-F) Toolboxes
 - <http://tftb.nongnu.org> (TFTB)
 - Developed by **Rice Univ.:** <http://dsp.rice.edu/> & **CN Recherche Scientifique:** <http://www.cnrs.fr/>
 - <http://espace.library.uq.edu.au/view/UQ:211321>
 - **T-F Signal Analysis (TFSA) Matlab Toolbox v.5.5**
 - Developed by **Univ. of Queensland:** uq.edu.au/
 - <http://www.mathworks.com/matlabcentral>
 - **DiscreteTFDs – T-F Signal Analysis Software**
 - Developed by **Jeff O'Neill:** jco8@cornell.edu

SIRLAB: Running on a NetSig Node

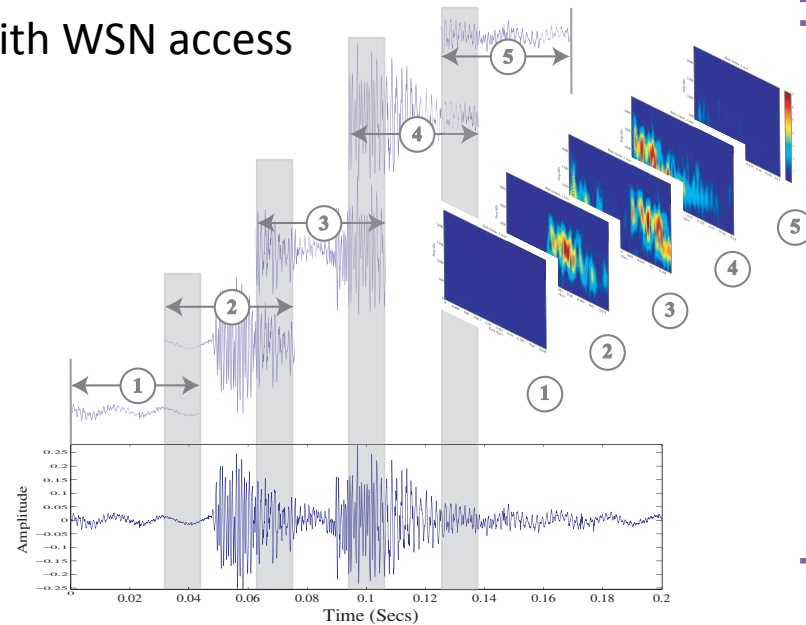
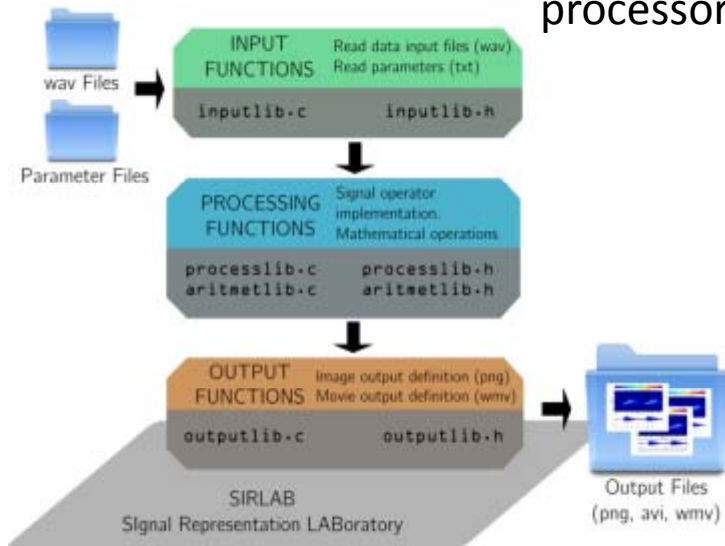
Monitoring Network



IWITDroid

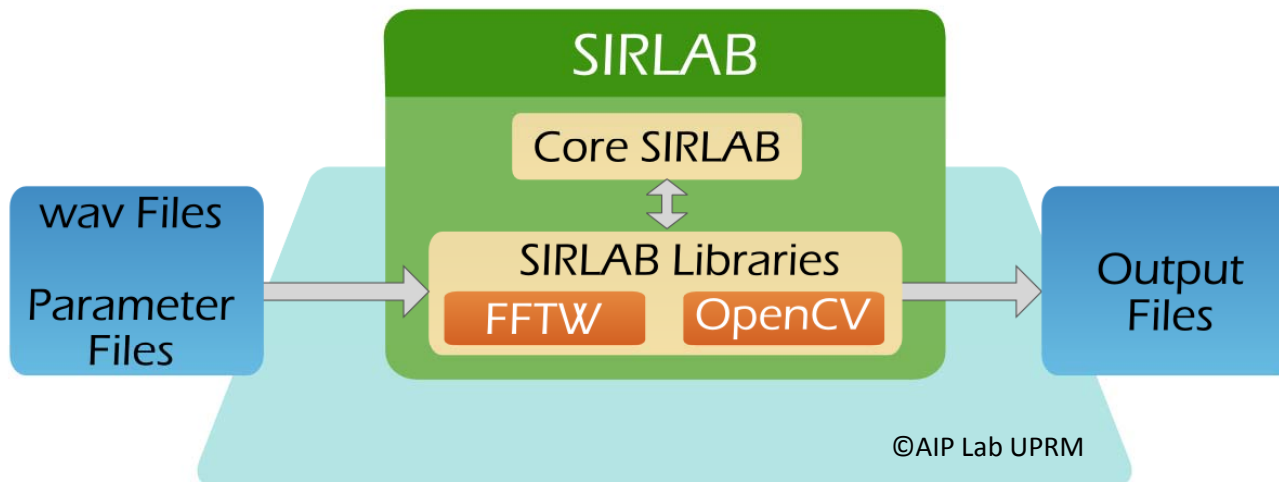
NetSig: on a Linux-based node processor with WSN access

SIRLAB Framework

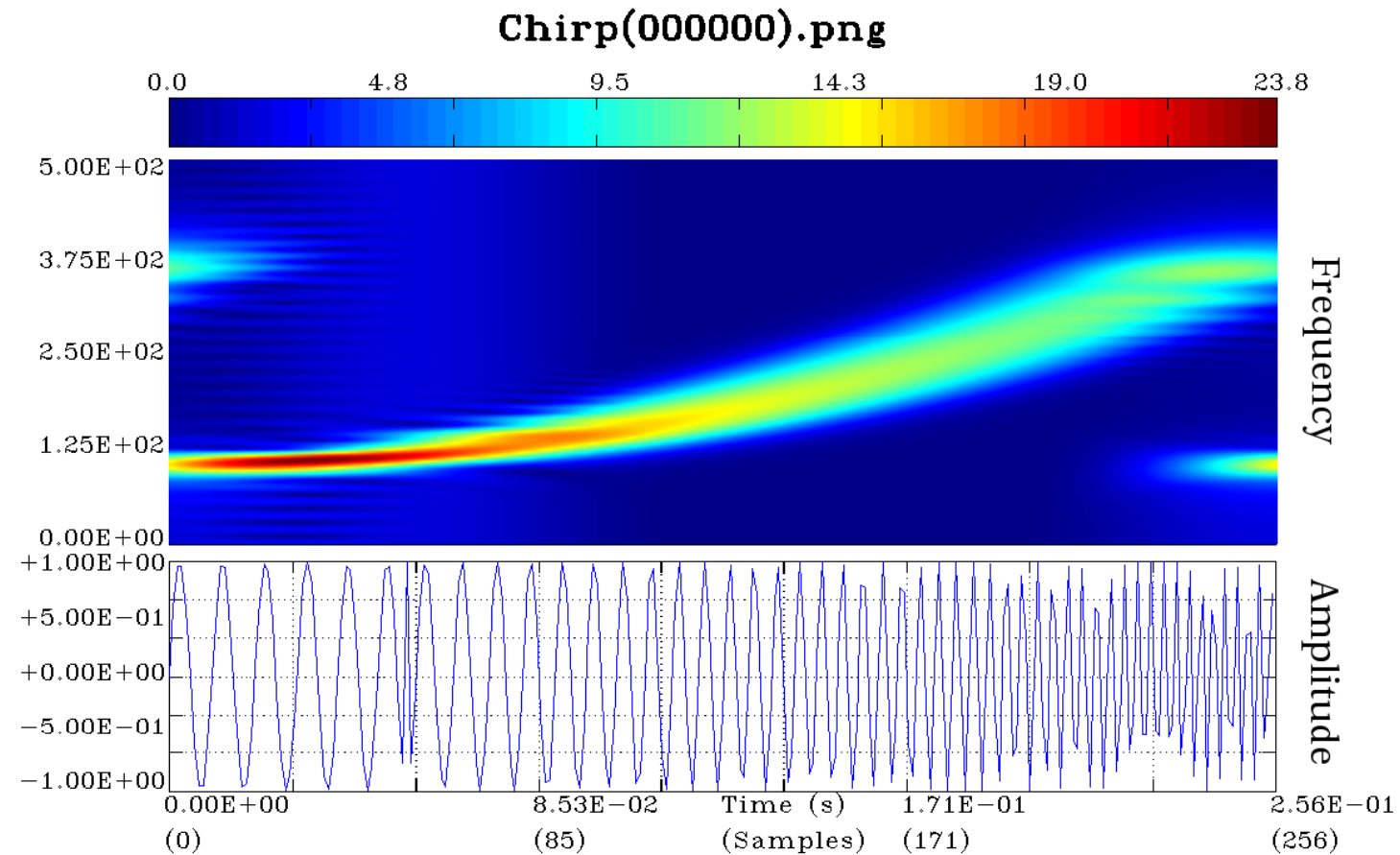


Spectrogram Displays

SIRLAB Architecture

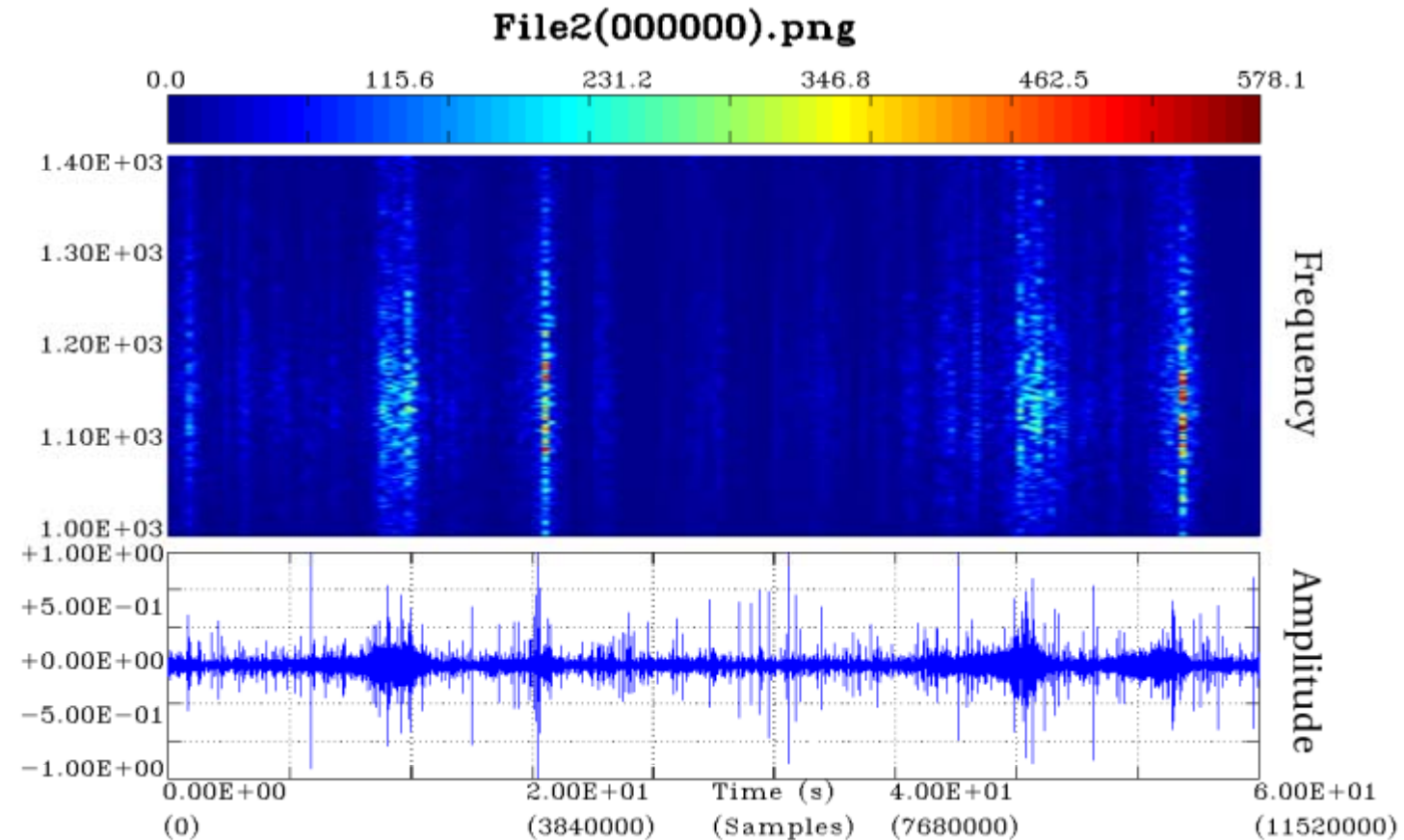


SIRLAB Frame Example



| | | | |
|------------------------|-------------|--------------------|---------------|
| Signal segment length: | 256 samples | Wavfile: | Chirp.wav |
| CSTFT overlay : | 1 samples | Wav file size : | 256 samples |
| Window size : | 128 samples | Sampling rate : | 1000 Hz |
| Zero padding : | 256 samples | Sample resolution: | 16 bits |
| Minimum frequency : | 0.00 Hz | Frame position : | 0 of 1 frames |
| Maximum frequency : | 500.00 Hz | | |

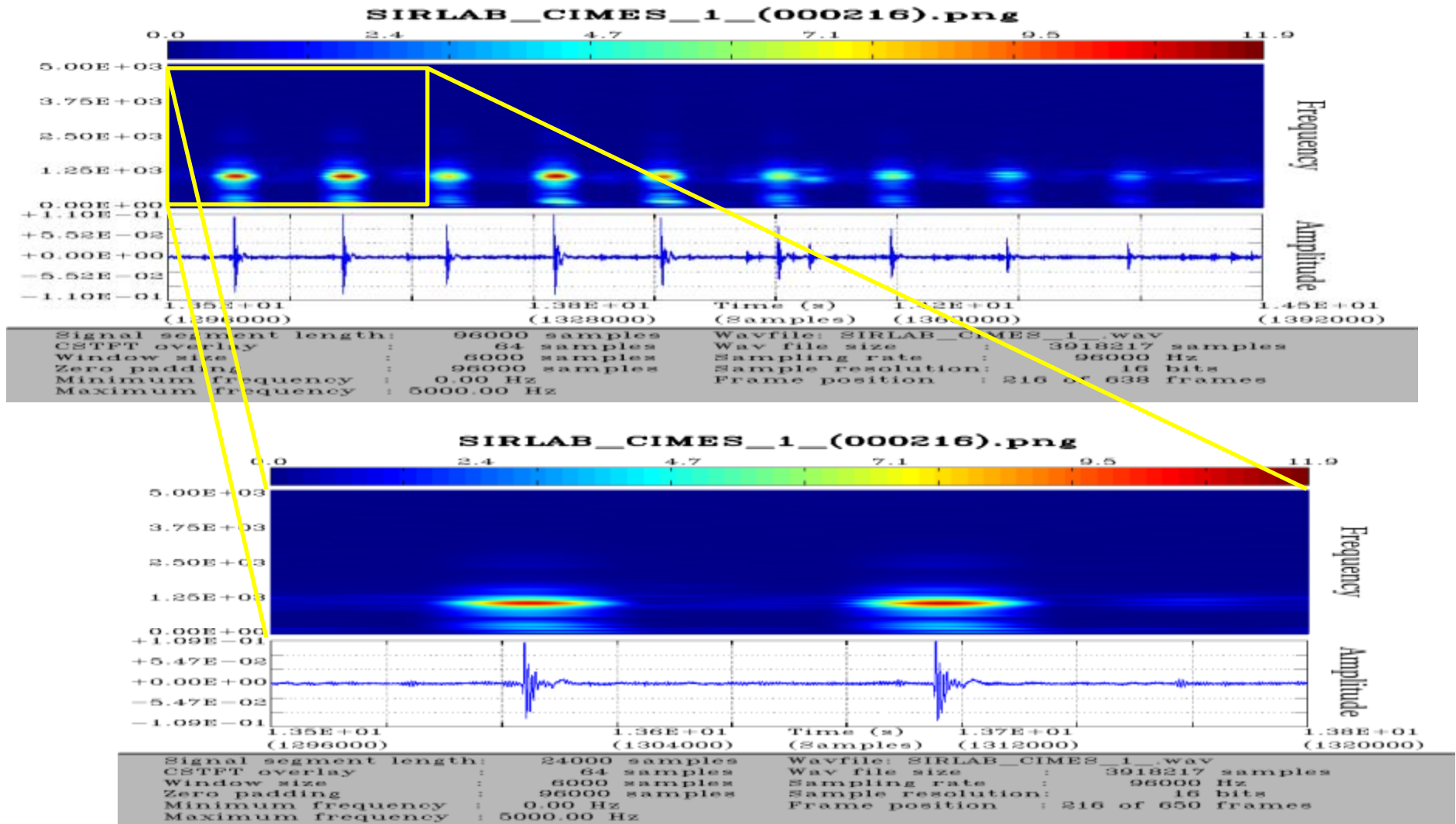
SIRLAB's Standard Output



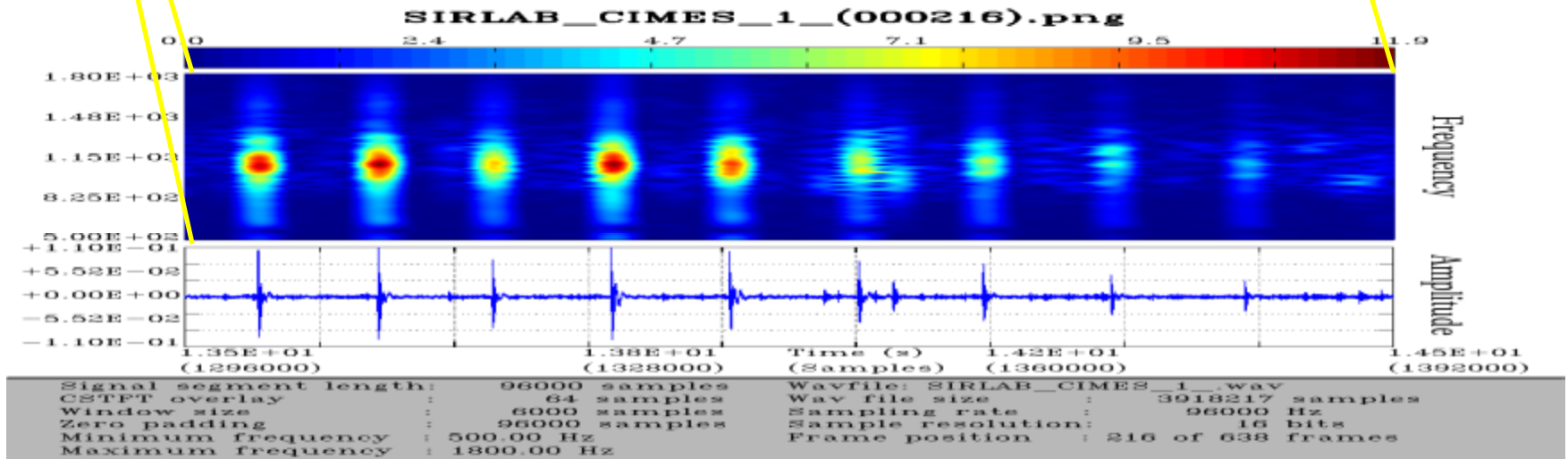
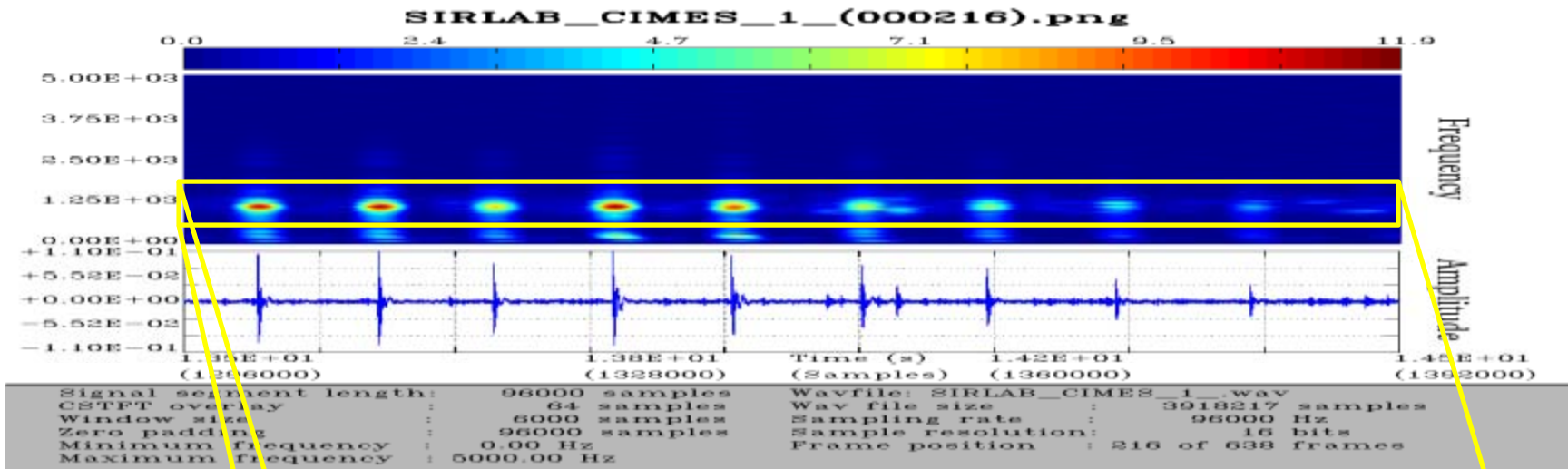
| | |
|---|-----------------------------------|
| Signal segment length: 11520000 samples | Wavfile: File2.wav |
| CSTFT overlay : 65536 samples | Wav file size : 15998282 samples |
| Window size : 100000 samples | Sampling rate : 192000 Hz |
| Zero padding : 11520000 samples | Sample resolution: 24 bits |
| Minimum frequency : 1000.00 Hz | Frame position : 0 of 4374 frames |
| Maximum frequency : 1400.00 Hz | |

Sea waves...

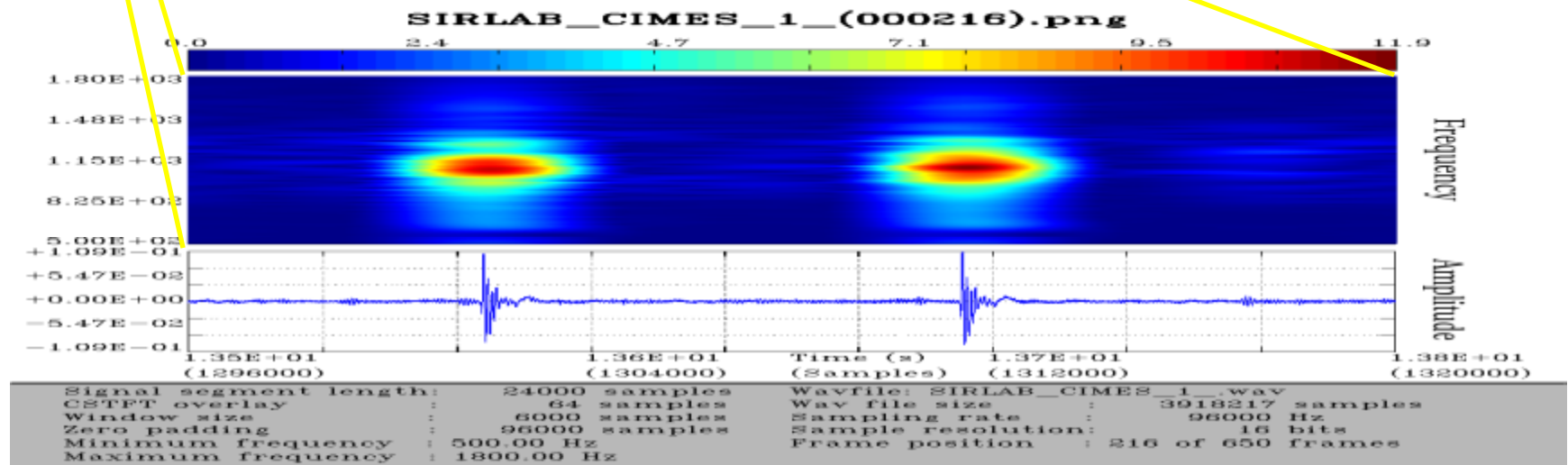
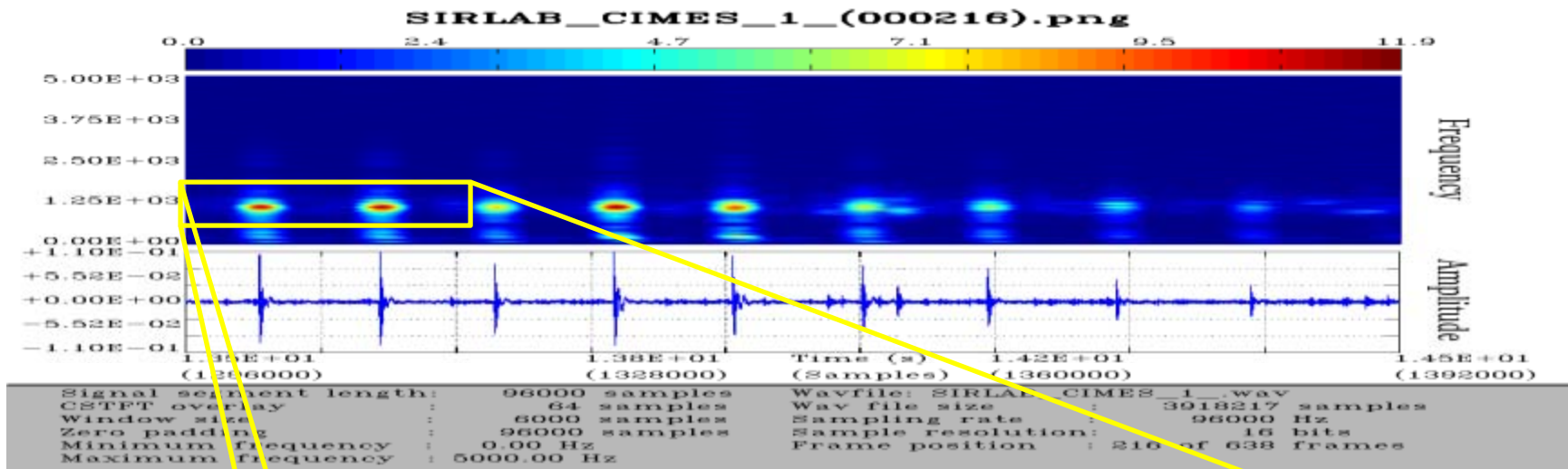
Frame Magnification: Time



Frame Magnification: Frequency



Frame Magnification: Time & Frequency



SIRLAB Challenges

- Parallelization Into Multiple Cores
- Improve the User Experience
 - Minimum Installation Effort
 - User Friendly Interaction
- Client – Server Architecture
- Integration with Mobile Devices

webSIRLAB

- Web Interface for SIRLAB
- Client Server Architecture
- Flexible Parameter Selection
- Easy Signal Submission for Processing at Server
- Friendly Display for Results
- Minimum Installation Effort
 - Modern **html5** web browser is required.

Research contribution

webSIRLAB's User Interface



SIRLAB

Num Samples frame:

Jump Samples:

Window Width:

Zero Padding:

Min Freq:

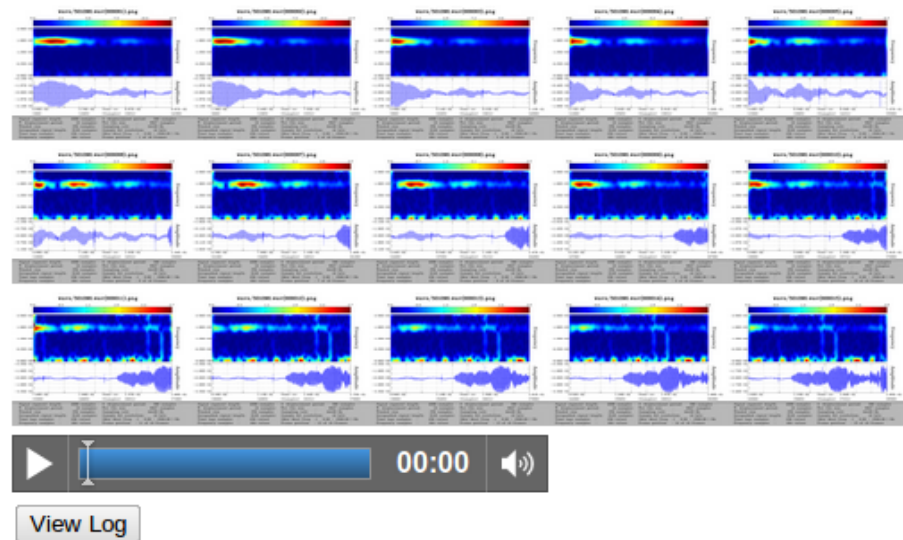
Max Freq:

Frame Overlay:

Start Percent:

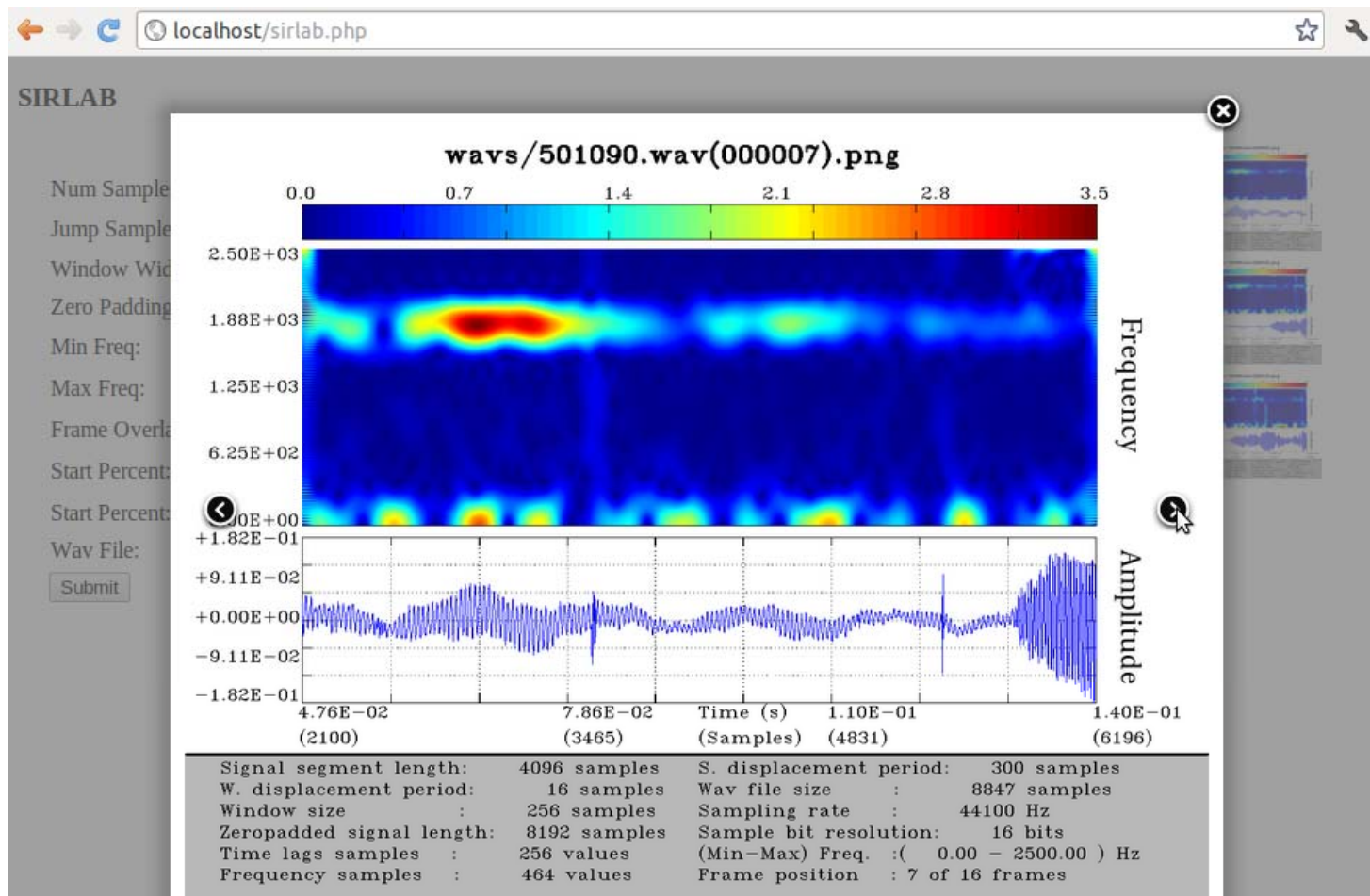
Start Percent:

Wav File:



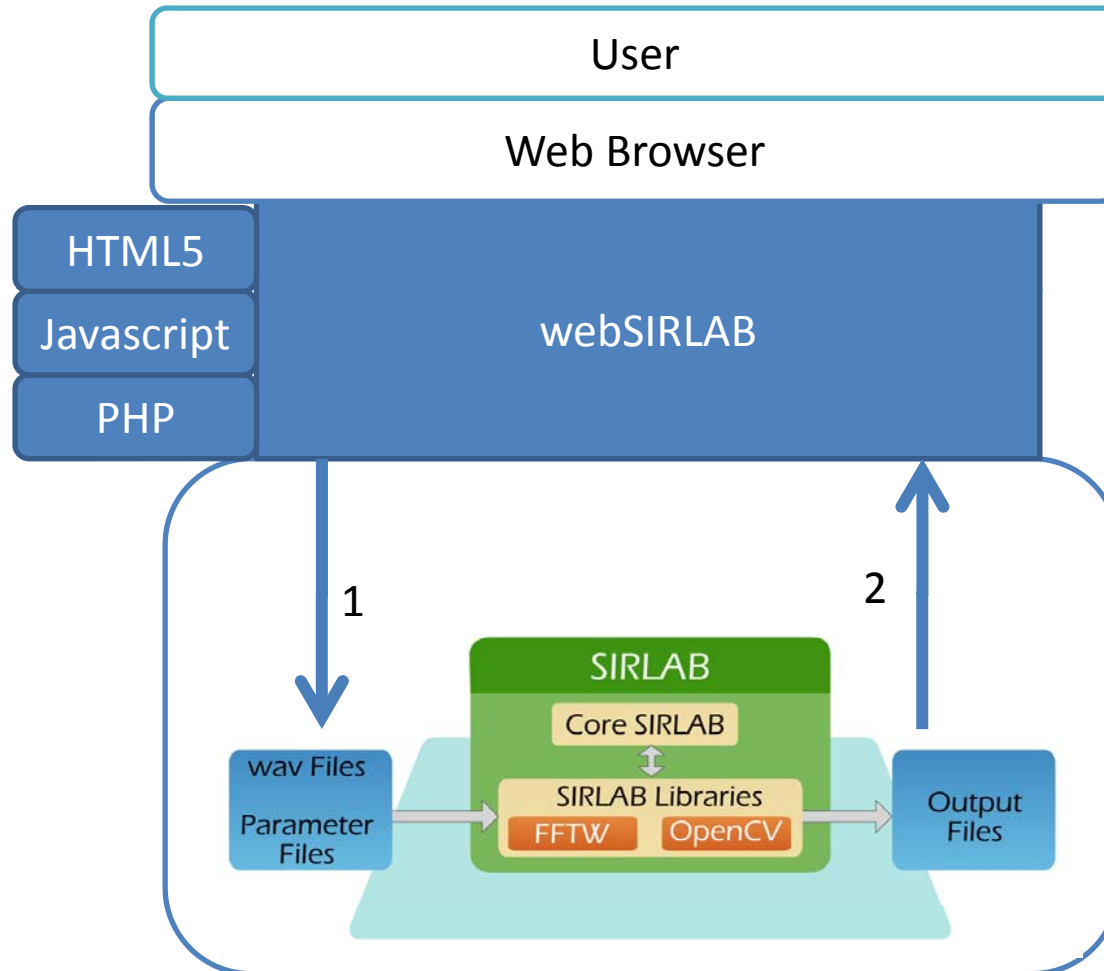
Research contribution

webSIRLAB



Research contribution

webSIRLAB Architecture



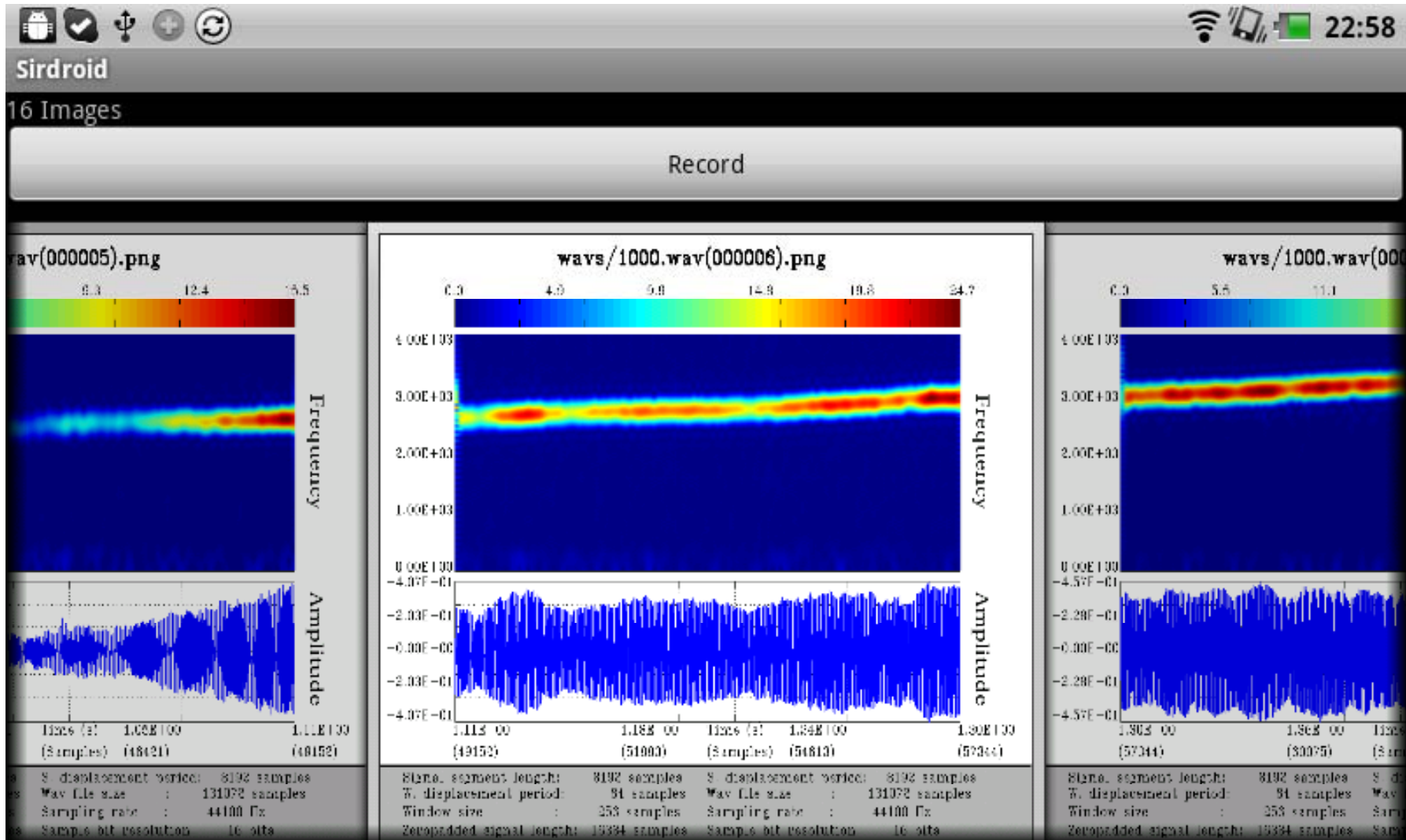
Research contribution

SIRDroid

- Android interface for SIRLAB
 - Android is the leading mobile operating system
- Client Server Architecture
- Can record data on real environment and get results in a few seconds

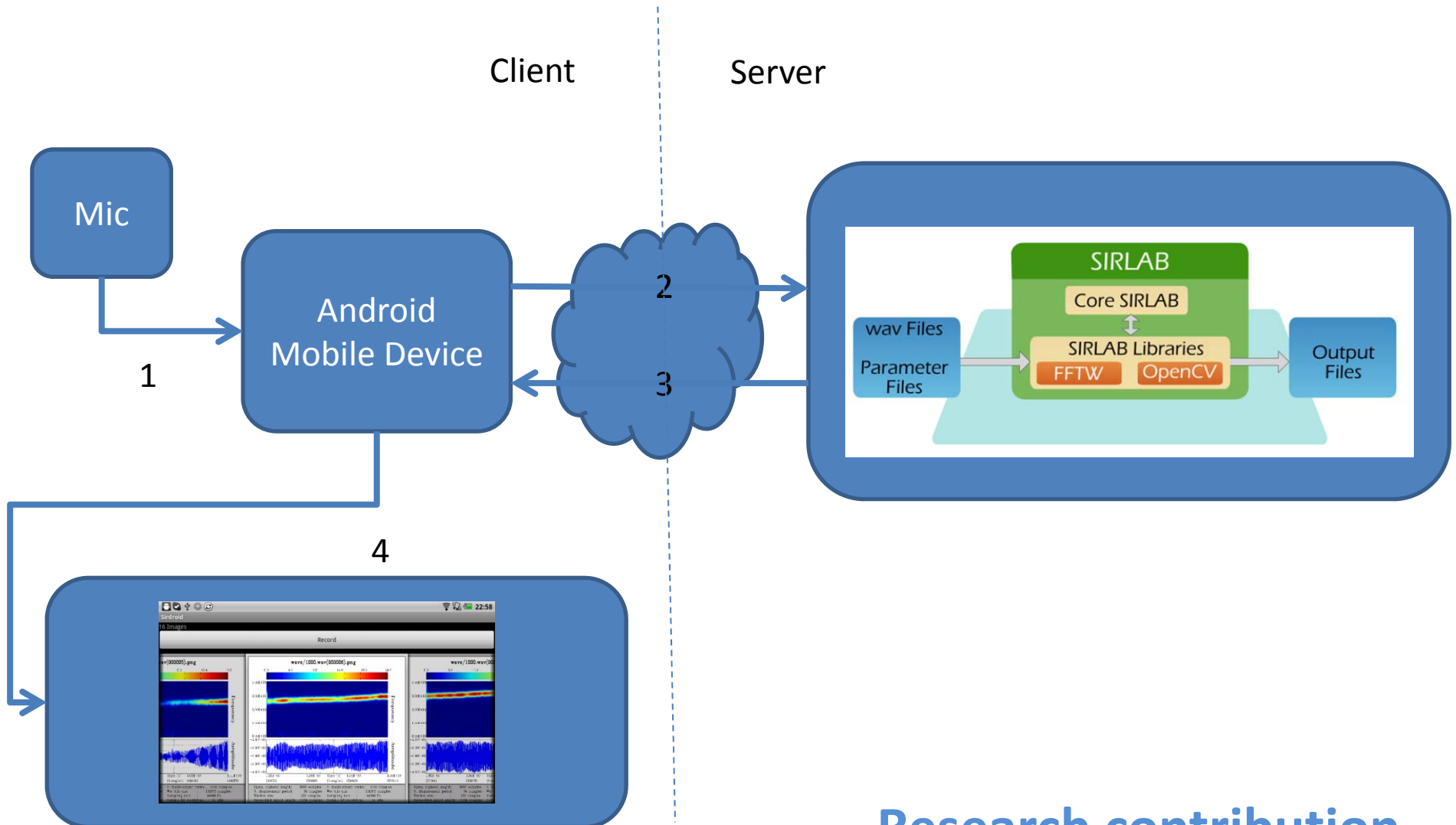
Research contribution

SIRDroid



Research contribution

SIRDroid Architecture

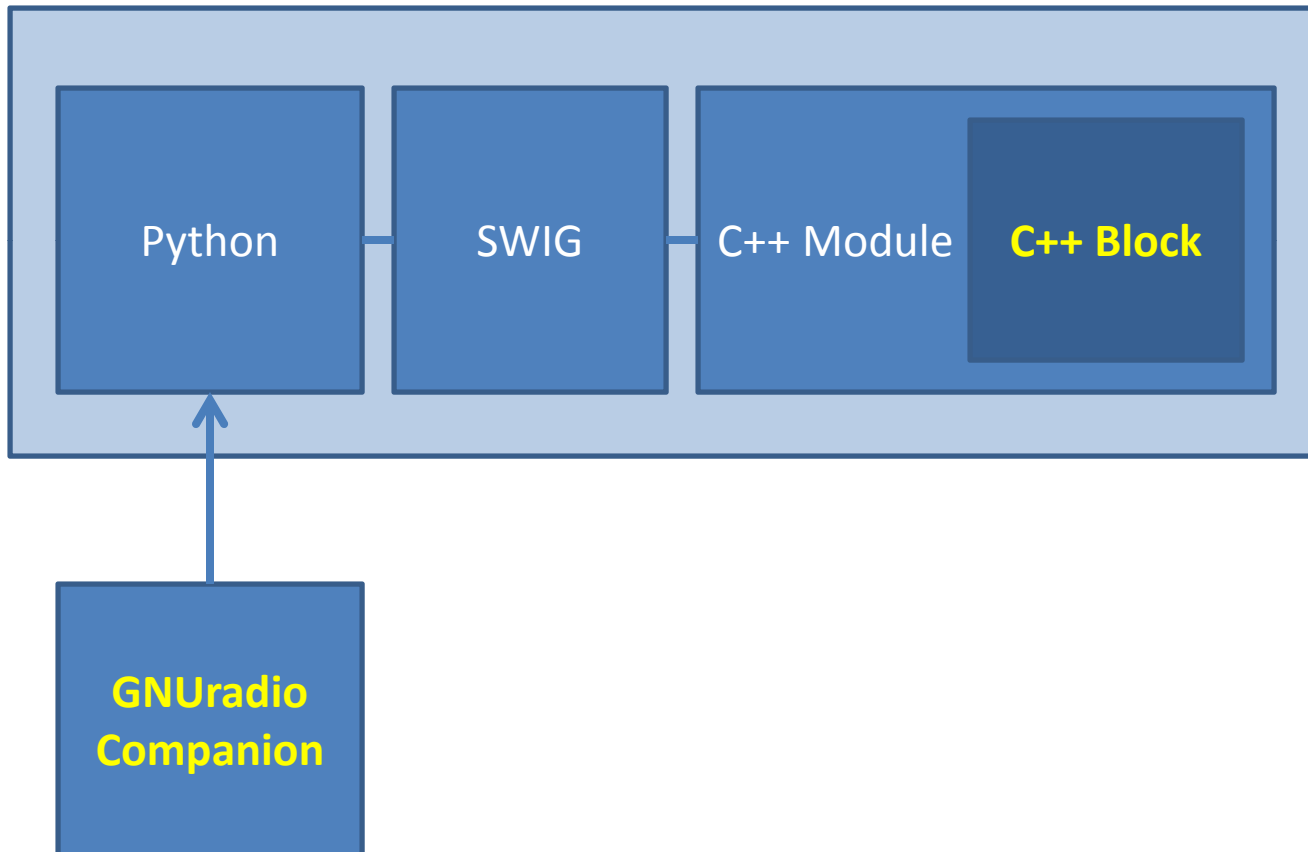


Research contribution

GNUradio

- Open Source Software
- Provides a library of signal processing blocks
- Sends and receives signals
- Audio, USRP, Network (TCP, UDP), File, Wav File
- USRP makes GNUradio wireless capable
- Software Defined Radio

GNUradio Architecture



*SWIG: Simplified Wrapper Interface Generator

GNUradio Companion

- Graphical Interface to GNUradio
- Iconic Programming Language
- Powerful Editor
 - Place the Required Blocks
 - Configure Each Block
 - Connect them Together
 - Build and Run

GNUradio Companion

The screenshot displays the GNUradio Companion interface for a file named 'uhd_test.grc'. The main workspace contains a flow graph with the following components and connections:

- UHD: USRP Source** (top center): Device Addr: addr=...168.20.2, Mb0: Ref Source: Internal, Mb0: Subdev Spec: A:0, Samp Rate (Sps): 10M, Ch0: Center Freq (Hz): 5.5G, Ch0: Gain (dB): 90, Ch0: Antenna: J2. Its output is connected to two sinks.
- QT GUI Sink** (top right): Name: QT GUI Plot, FFT Size: 1.024k, Center Frequency (Hz): 0, Bandwidth (Hz): 10M.
- QT GUI Time Sink** (middle right): Name: QT GUI Plot, Number of Points: 1.024k, Bandwidth (Hz): 10M.
- Signal Source** (middle left): Sample Rate: 10M, Waveform: Cosine, Frequency: 100k, Amplitude: 1, Offset: 0. Its output is connected to the 'Float To Complex' block.
- Constant Source** (bottom left): Constant: 0. Its output is connected to the 'Float To Complex' block.
- Float To Complex** (center): Receives inputs from the Signal Source and Constant Source. Its output is connected to two sinks.
- UHD: USRP Sink** (middle right): Device Addr: addr=...168.10.2, Mb0: Ref Source: Internal, Mb0: Subdev Spec: A:0, Samp Rate (Sps): 10M, Ch0: Center Freq (Hz): 5.5G, Ch0: Gain (dB): 30, Ch0: Antenna: J2.
- QT GUI Time Sink** (bottom right): Name: QT GUI Plot, Number of Points: 1.024k, Bandwidth (Hz): 10M.

On the left side, there are control panels:

- Options:** ID: top_block, Author: Burak Kelleci, Generate Options: QT GUI.
- Variable:** ID: samp_rate, Value: 10M.
- Variable:** ID: center_freq, Value: 5.5G.

At the bottom, a terminal window shows the following output:

```
- Current recv frame size: 1472 bytes
- Current send frame size: 1472 bytes
- Opening a USRP2/N-Series device...
- Current recv frame size: 1472 bytes
- Current send frame size: 1472 bytes
U
>>> Done
```

On the right side, a 'Blocks' panel lists various components:

- [Sources]
- [Sinks]
- [Operators]
- [Type Conversions]
- [Stream Conversions]
- [Misc Conversions]
- [Synchronizers]
- [Level Controls]
- [Filters]
- [Modulators]
- [Error Correction]
- [Line Coding]
- [Vocoders]
- [Probes]
- [Variables]
- [Misc]
- [UHD]
 - UHD: USRP Source
 - UHD: USRP Sink
- [NOAA]
- [WX GUI Widgets]
- [Pager]
- [QT GUI Widgets]
- [USRP]

An 'Add' button is located at the bottom right of the blocks panel.

GNUradio Challenges

- Build Time-frequency Representations
 - Short Time Fourier Transform
 - Ambiguity Function
 - Wigner Distribution
 - Choi – Williams Distribution

The theory behind the solution

DESIGN

Design Outline

- Time-Frequency Representations
 - Algorithms, complexity and parallelization
- MIMO Channel
 - MIMO System Modeling
 - Proposed Channel Modeling
 - Modulation-Convolution-Delay
 - Delay-Convolution-Modulation

Time-Frequency Representations

Short Time Fourier Transform

$$S_{x,w}[k, m] = \sum_{n \in \mathbb{Z}_N} x[n] w[n - m] e^{-j2\pi \frac{kn}{N}}$$

Ambiguity Function

$$A_{f,g}[m, k] = \sum_{n \in \mathbb{Z}_N} f[n] g^*[\langle n + m \rangle_N] e^{-j2\pi \frac{kn}{N}}$$

Wigner Distribution

$$W_x[n, k] = \frac{1}{N} \sum_{\tau=0}^{N-1} \sum_{v=0}^{N-1} \sum_{l=0}^{N-1} \rho_N e^{j2\pi vl} x[\langle l + \tau \rangle_N] x^*[l] e^{-j\frac{2\pi}{N}(nv + k\tau)}$$

Choi-Williams Distribution

$$C_x[t, f] = \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} A_x[m, k] \Phi[m, k] e^{j\frac{2\pi}{N}(kt - mf)},$$

Short Time Fourier Transform

Algorithm 3 Short Time Fourier Transform

Input: Signal $x[n] \in l^2(\mathbb{Z}_N)$, Window width $w > 0 \in \mathbb{Z}$, Zero-padding $P > 0 \in \mathbb{Z}$.

Output: Matrix $s[M][P]$

```
1:  $M \leftarrow \lfloor \frac{N}{w} \rfloor$ 
2: for  $i = 0 \rightarrow M - 1$  do
3:    $wn \in \mathbb{Z}^N \leftarrow 0$ 
4:    $y \in \mathbb{Z}^P \leftarrow 0$ 
5:   for  $j = 0 \rightarrow w - 1$  do
6:      $wn[j + w \cdot i] \leftarrow 1$ 
7:   end for
8:   for  $j = 0 \rightarrow N - 1$  do
9:      $y[j] \leftarrow x[j] \cdot wn[j]$ 
10:  end for
11:   $s[i] \leftarrow \text{dft}(y)$ 
12: end for
```

Computational Time Complexity: $O(n^2 \log(n))$

Research contribution

Ambiguity Function

Algorithm 4 Ambiguity Function

Input: Signal $x[n] \in l^2(\mathbb{Z}_N)$, $y[n] \in l^2(\mathbb{Z}_N)$

Output: Matrix $A[N][N]$

```
1: for  $i = 0 \rightarrow N - 1$  do
2:   for  $j = 0 \rightarrow N - 1$  do
3:      $y1[j] \leftarrow \text{conj}(y[\text{mod}(j + i, N)])$ 
4:   end for
5:   for  $j = 0 \rightarrow N - 1$  do
6:      $z[j] \leftarrow x[j] \cdot y1[j]$ 
7:   end for
8:    $A[i] \leftarrow \text{dft}(z)$ 
9: end for
```

Computational Time Complexity: $O(n^2 \log(n))$

Research contribution

Wigner Distribution

Algorithm 5 Wigner

Input: Signal $x[n] \in l^2(\mathbb{Z}_N)$, $y[n] \in l^2(\mathbb{Z}_N)$

Output: Matrix $W[N][N]$

```
1: for  $i = 0 \rightarrow N - 1$  do
2:   for  $j = 0 \rightarrow N - 1$  do
3:      $y1[j] \leftarrow \text{conj}(y[\text{mod}(j + i, N)])$ 
4:   end for
5:   for  $j = 0 \rightarrow N - 1$  do
6:      $z[j] \leftarrow x[j] \cdot y1[j]$ 
7:   end for
8:    $A[i] \leftarrow \text{dft}(z)$ 
9: end for
10:  $W \leftarrow \text{dft2}(A)$ 
```

Computational Time Complexity: $O(n^2 \log(n))$

Research contribution

Choi-Williams Distribution

Algorithm 6 Choi-Williams

Input: Signal $x[n] \in l^2(\mathbb{Z}_N)$, $y[n] \in l^2(\mathbb{Z}_N)$, α

Output: Matrix $W[N][N]$

```
1: for  $i = 0 \rightarrow N - 1$  do
2:   for  $j = 0 \rightarrow N - 1$  do
3:      $y1[j] \leftarrow y[\text{mod}(j + i, N)]$ 
4:   end for
5:   for  $j = 0 \rightarrow N - 1$  do
6:      $z[j] \leftarrow x[j] \cdot y1[j]$ 
7:   end for
8:    $A[i] \leftarrow \text{dft}(z)$ 
9:   for  $j = 0 \rightarrow N - 1$  do
10:     $A[i][j] \leftarrow A[i][j] \cdot e^{-\alpha(i \cdot j)^2}$ 
11:  end for
12: end for
13:  $W1 \leftarrow \text{dft2}(A)$ 
14:  $W \leftarrow \text{conj}(W1)$ 
```

Computational Time Complexity: $O(n^2 \log(n))$ **Research contribution**

MIMO System

The channel is seen as a linear system

$$y = Hx + N.$$

H then is decomposed

$$H = U\Sigma V^*$$

Finally

$$U^{-1}y = \Sigma(V^*x) + N$$

The symbols are preprocessed at the transmitter and post-processed at the receiver

MIMO System

- The channel must be estimated
 - Several mechanisms have been proposed and implemented
 - Subject to errors
- The Shannon capacity increases linearly with every pair of antennas at the transmitter and receiver.

Formulated MIMO Channel

- A “MxN” MIMO channel can be seen as a collection of $C=M \times N$ Single Input Single Output (SISO) Channels.
- Every SISO Channel is then modeled like a channel with 3 operators:
 - **Delay:** The signal has a latency while is propagating
 - **Convolution:** The medium acts like a FIR Filter
 - **Modulation:** The medium inserts a Doppler shift
- The order is important, this yields two approximations:
 - Delay-Convolution-Modulation (DCM)
 - Modulation-Convolution-Delay (MCD)

Research contribution

DCM

Operators

$$y_{a,b} = g_{a,b} \odot_D X_{k_{a,b}},$$

where $g_{a,b} \in l^2(\mathbb{Z}_D)$

$$g_{a,b} = T_{h_{a,b}}\{f_{a,b}\},$$

where $f_{a,b} \in l^2(\mathbb{Z}_D)$

$$f_{a,b} = x_b \otimes_D \delta_{m_{a,b}}.$$

Composition

$$y_{a,b} = (T_{h_{a,b}}\{x_b \otimes_D \delta_{m_{a,b}}\}) \odot_D X_{k_{a,b}}.$$

MIMO

$$y_a = \sum_{b=0}^{M-1} (T_{h_{a,b}}\{x_b \otimes_D \delta_{m_{a,b}}\}) \odot_D X_{k_{a,b}}.$$

Research contribution

MCD

Operators

$$y_{a,b} = g_{a,b} \circledast_D \delta_{m_{a,b}},$$

where $g_{a,b} \in l^2(\mathbb{Z}_D)$

$$g_{a,b} = T_{h_{a,b}}\{f_{a,b}\},$$

where $f_{a,b} \in l^2(\mathbb{Z}_D)$

$$f_{a,b} = x_b \odot_D X_{k_{a,b}}.$$

Composition

$$y_{a,b} = (T_{h_{a,b}}\{x_b \odot_D X_{k_{a,b}}\}) \circledast_D \delta_{m_{a,b}}.$$

MIMO

$$y_a = \sum_{b=0}^{M-1} (T_{h_{a,b}}\{x_b \odot_D X_{k_{a,b}}\}) \circledast_D \delta_{m_{a,b}},$$

Research contribution

The code behind the solution

IMPLEMENTATION

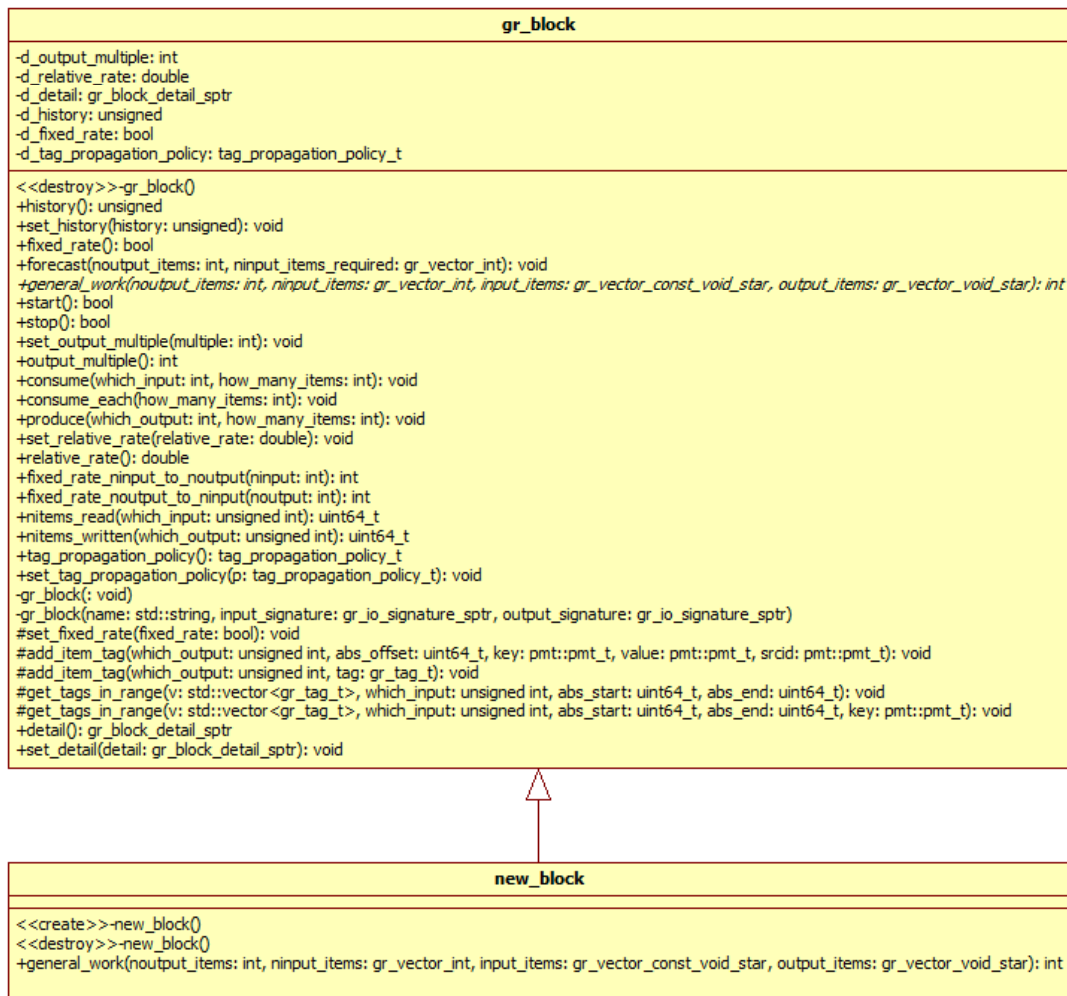
Time Frequency Representations

- The main implementation is done in C++
- Object oriented programming
- Each time-frequency representation is a class
 - Implements from *gr_block* interface
 - The method *general_work* is implemented
 - A constructor sets the initial properties of each time frequency representation

Time-Frequency Representations

Block
Interface

Constructor,
destructor and
general_work are
implemented



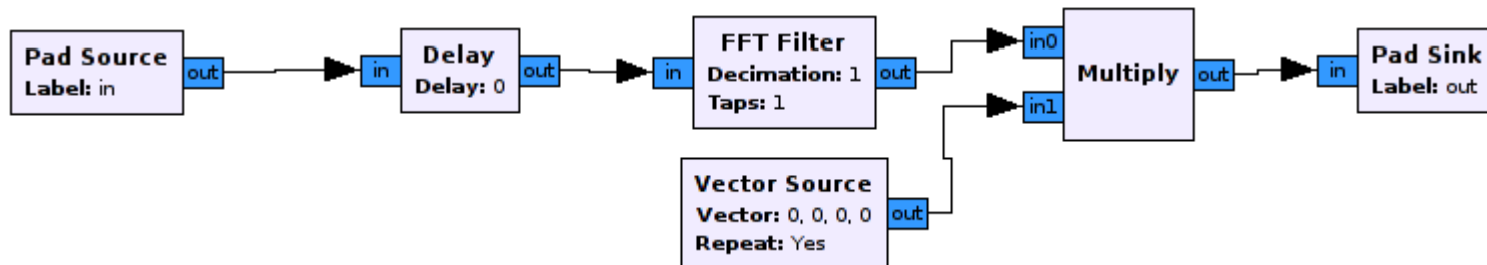
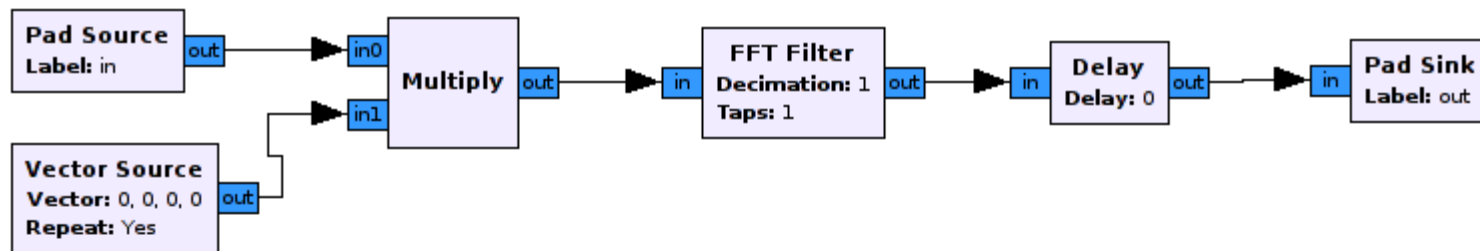
Parallelization

- Each running block is a thread into the whole system
- Our time-frequency representation blocks are executed in parallel as an option
 - Data parallelism
 - OpenMP was selected for parallelization
 - API for automatic parallelization, include compiler directives, libraries and functions
 - Cross compatibility, great support and meets the project requirement

MIMO Channel

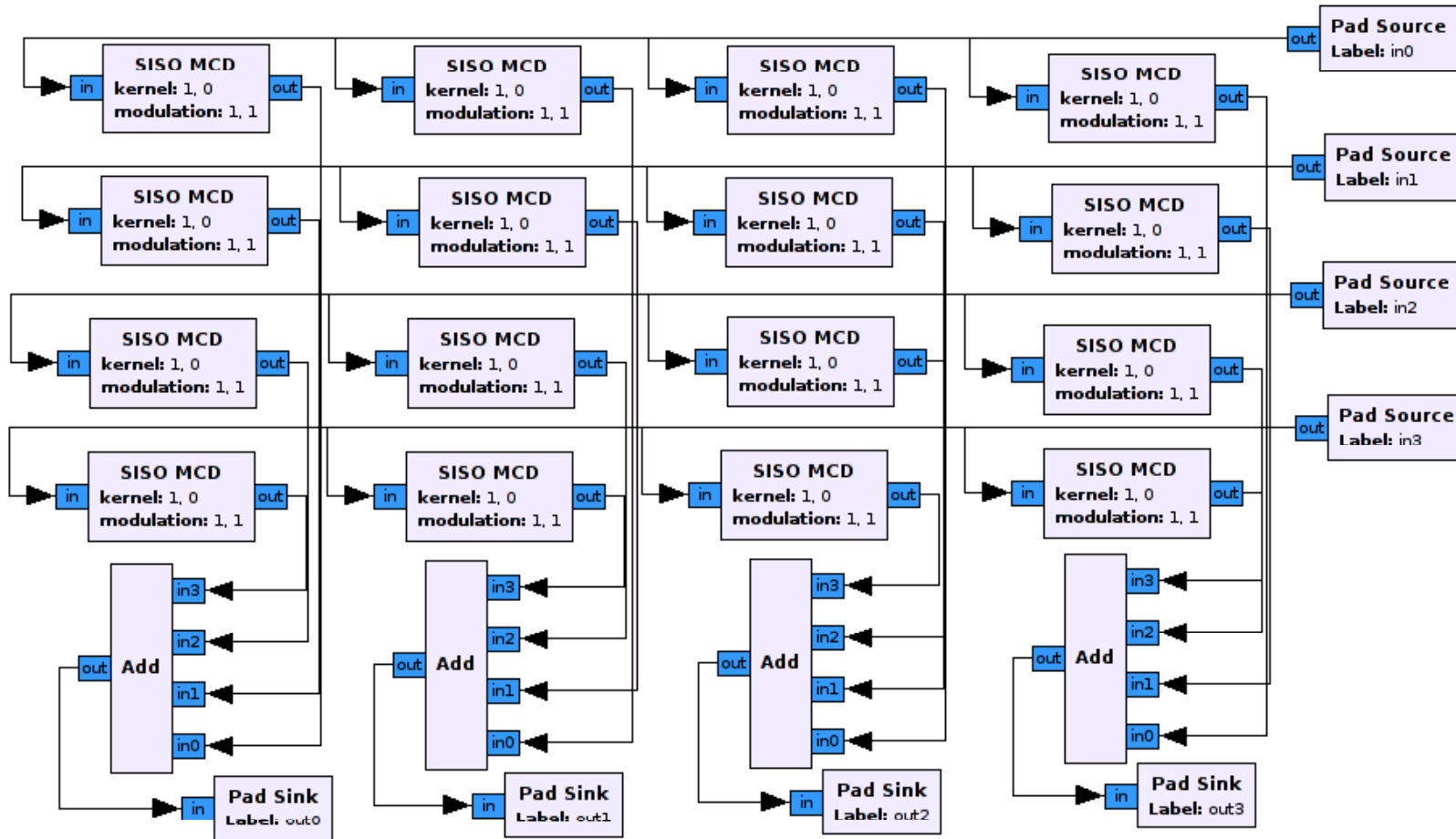
- Development of each block, MCD and DCM internally with GNUradio Companion
- Iconic Programming

MCD and DCM Blocks



Research contribution

4x4 MIMO



Research contribution

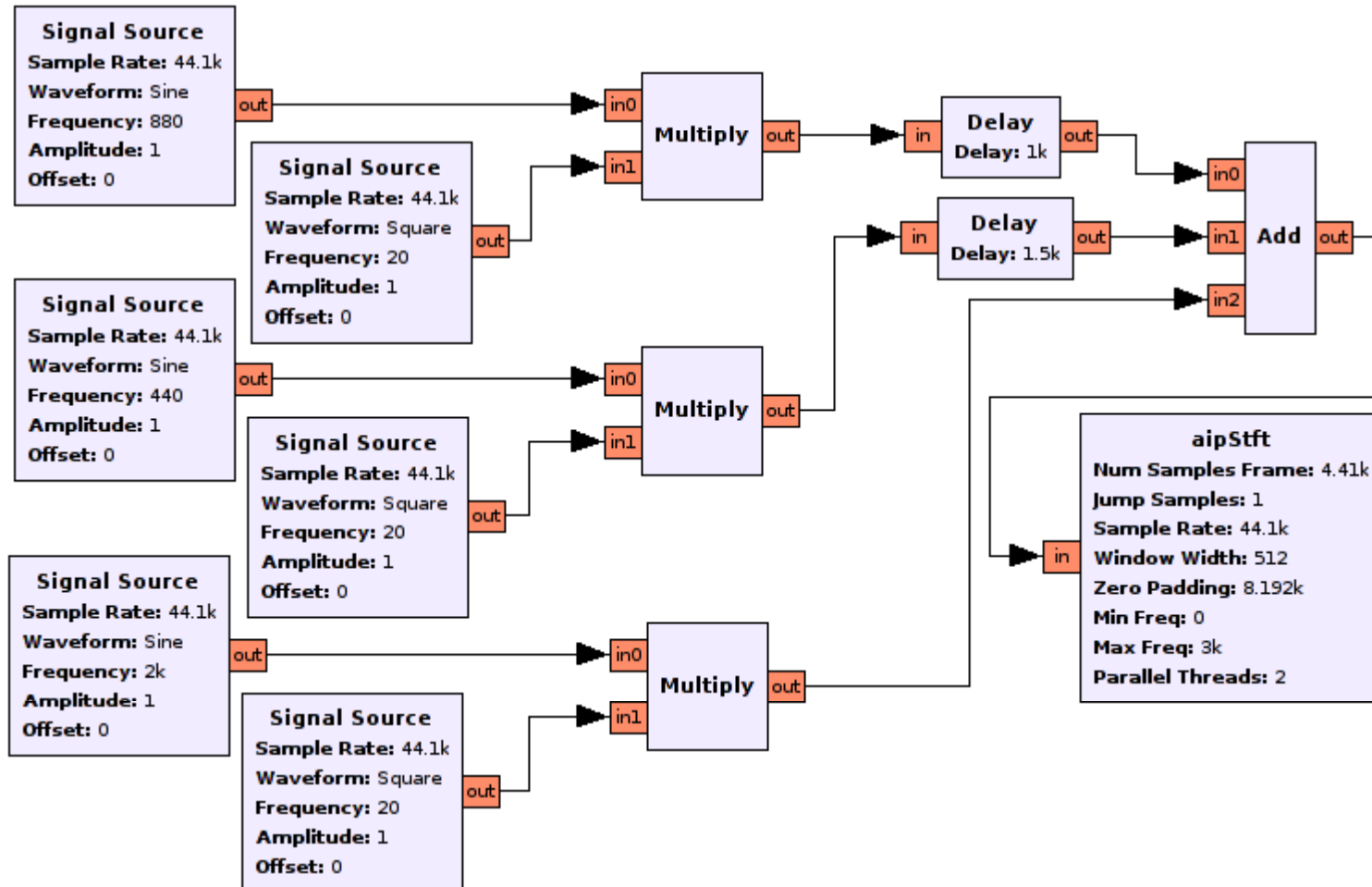
The results behind the solution

TESTING

Acoustic Channel Surveillance

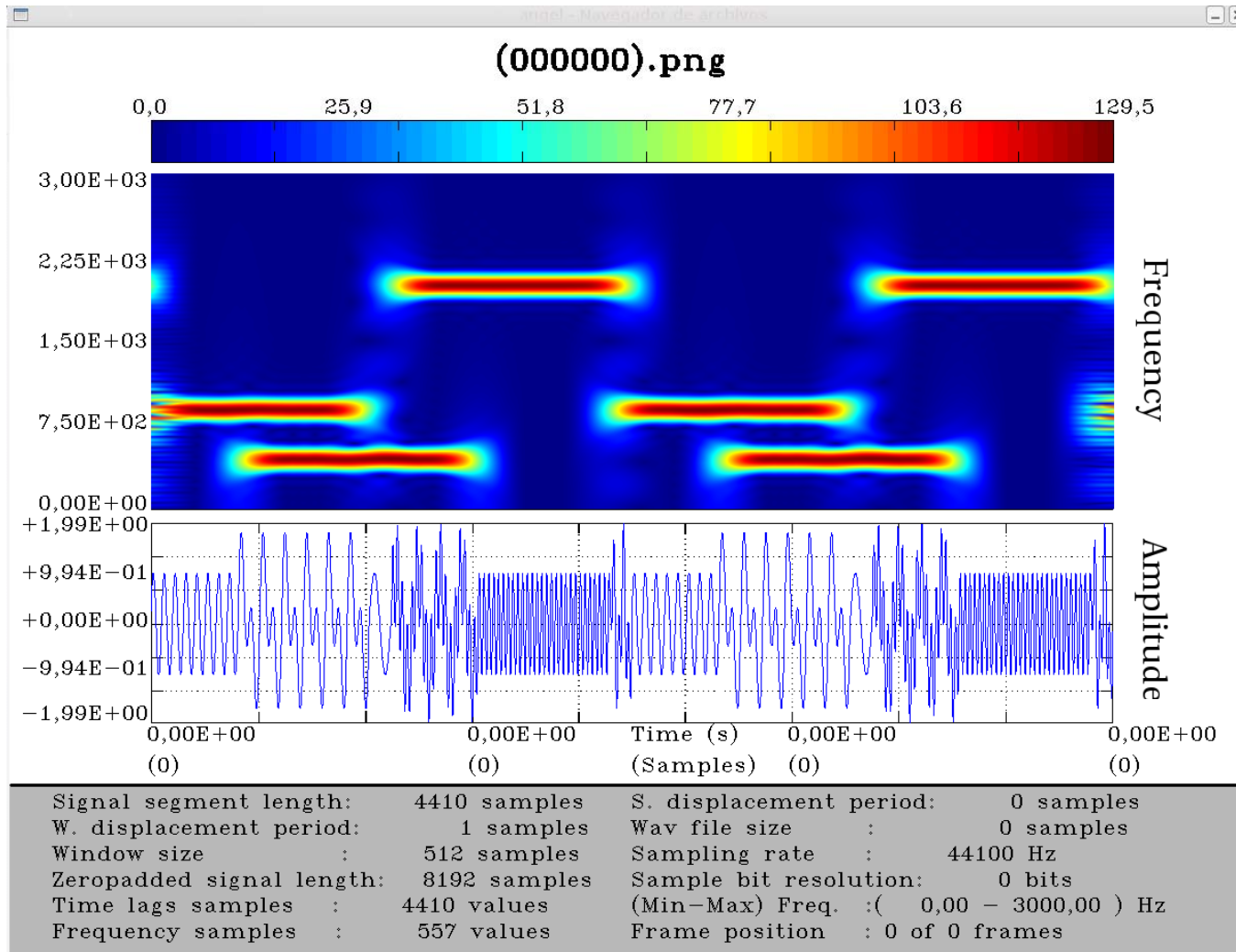
- The STFT can be used to monitor the environment
 - Species characterization in bioacoustics
 - Sonar

Acoustic Channel Surveillance



Research contribution

Channel Surveillance

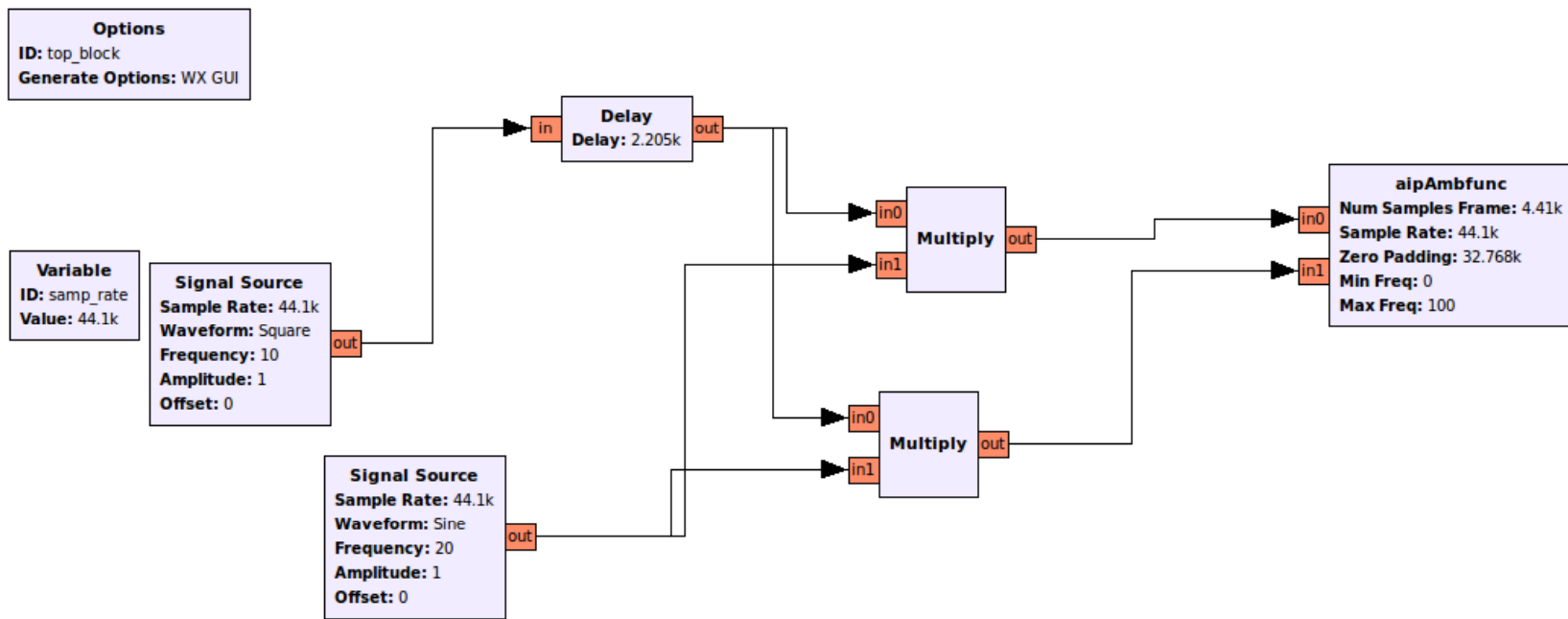


Research contribution

Ambiguity Function

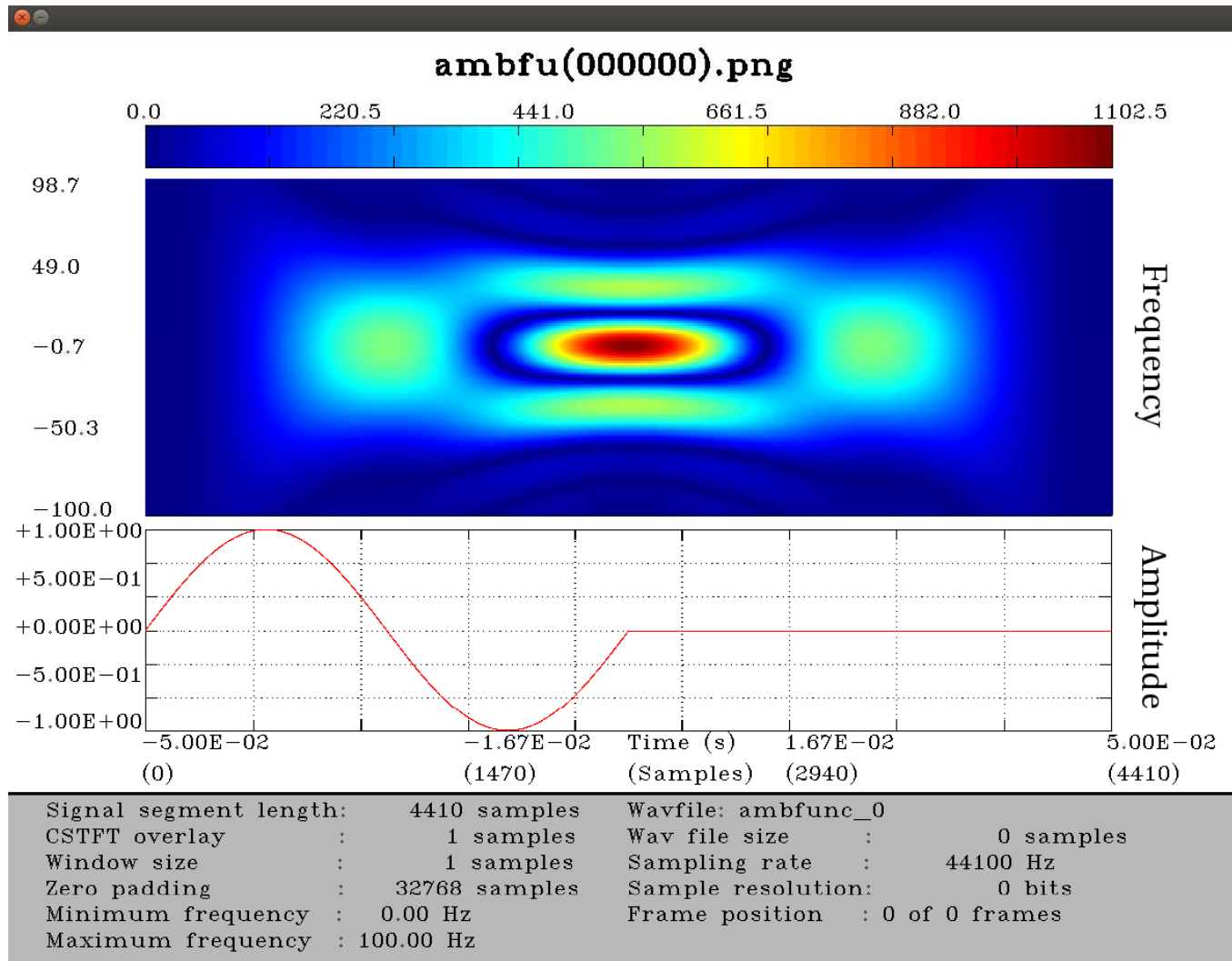
- Used for Sonars
- Design of Signals
- Detection of Delay and Doppler Shifts
- Testing With Well Known Signals
 - Sine
 - Square

Sine



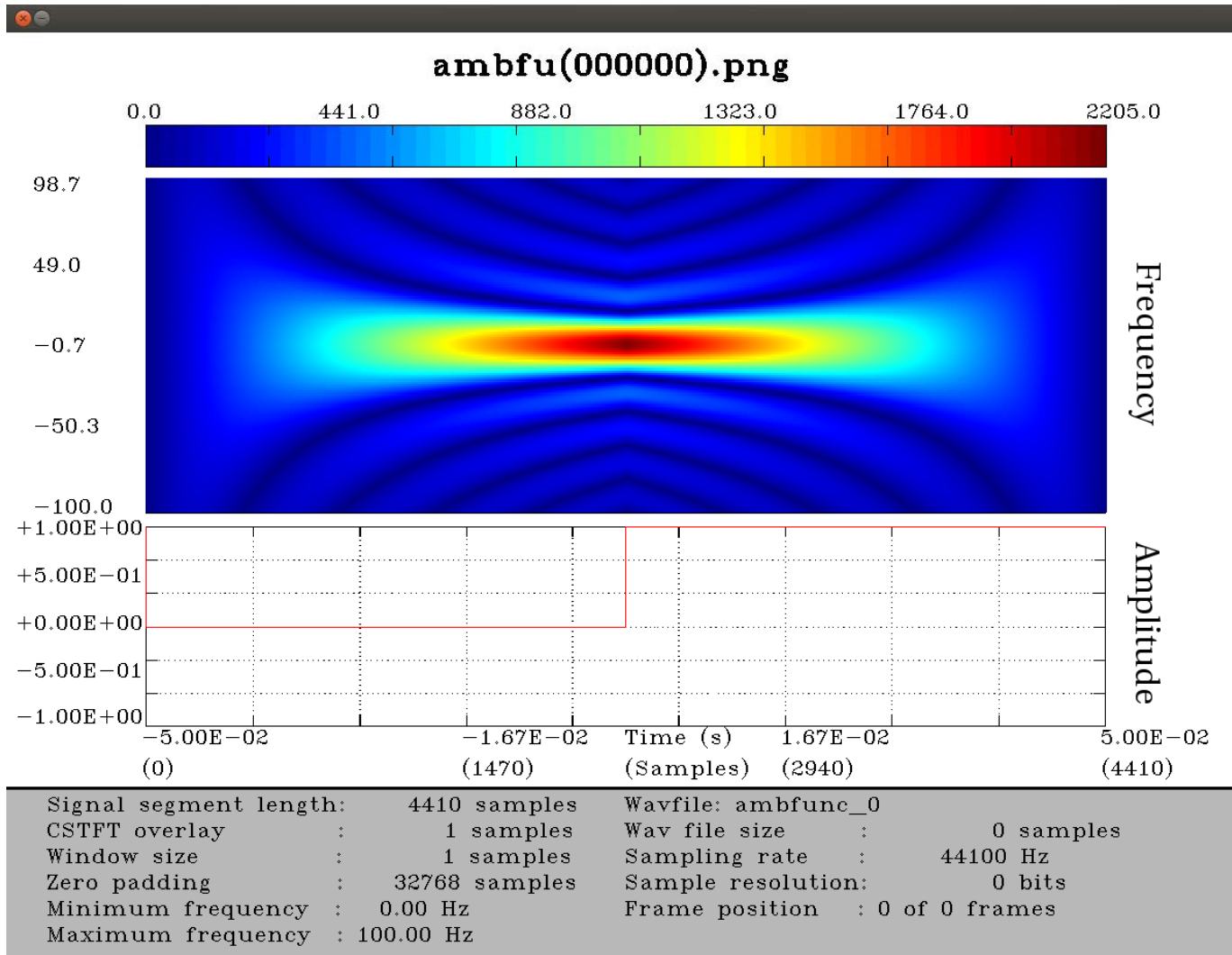
Research contribution

Sine



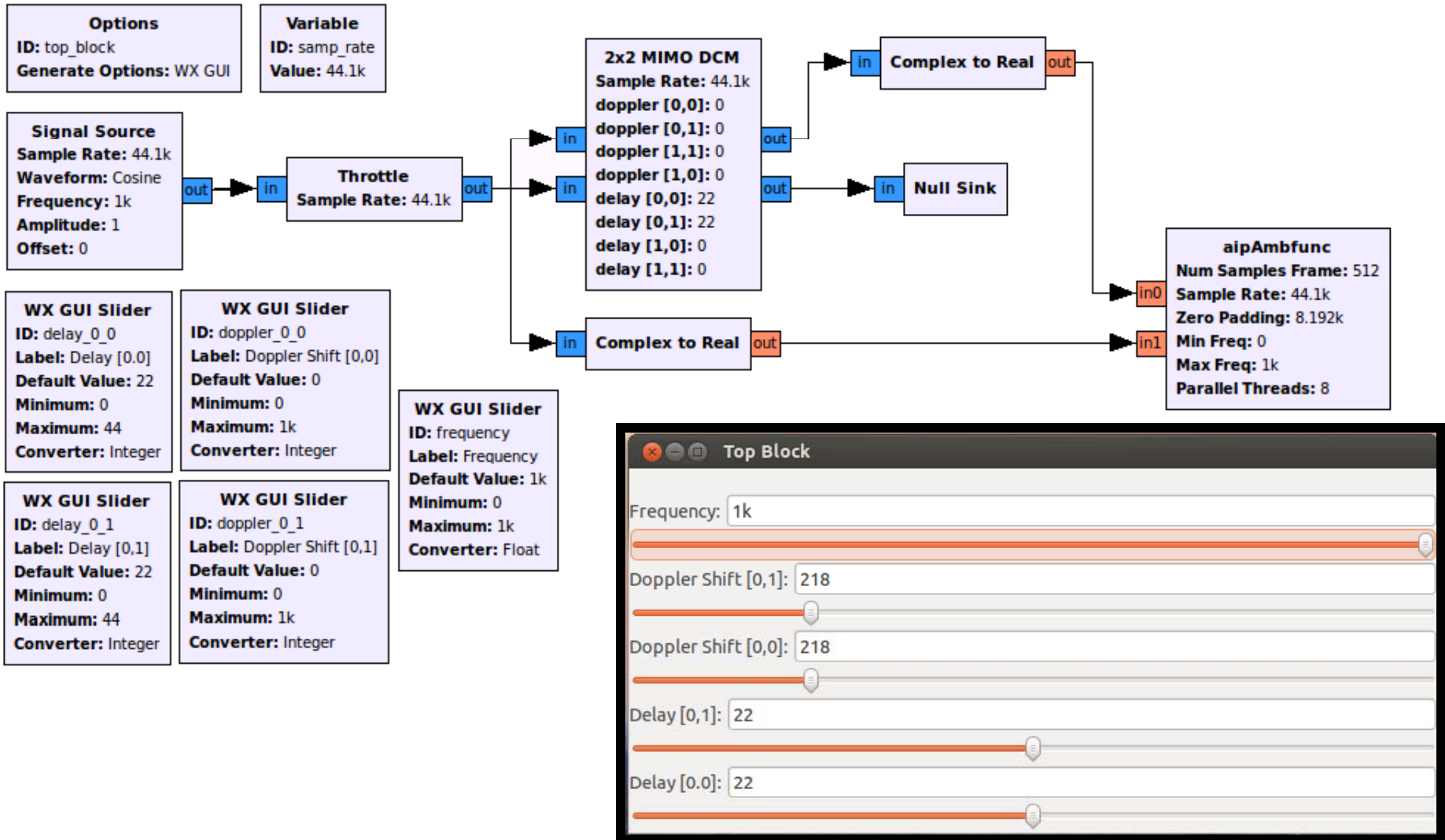
Research contribution

Square



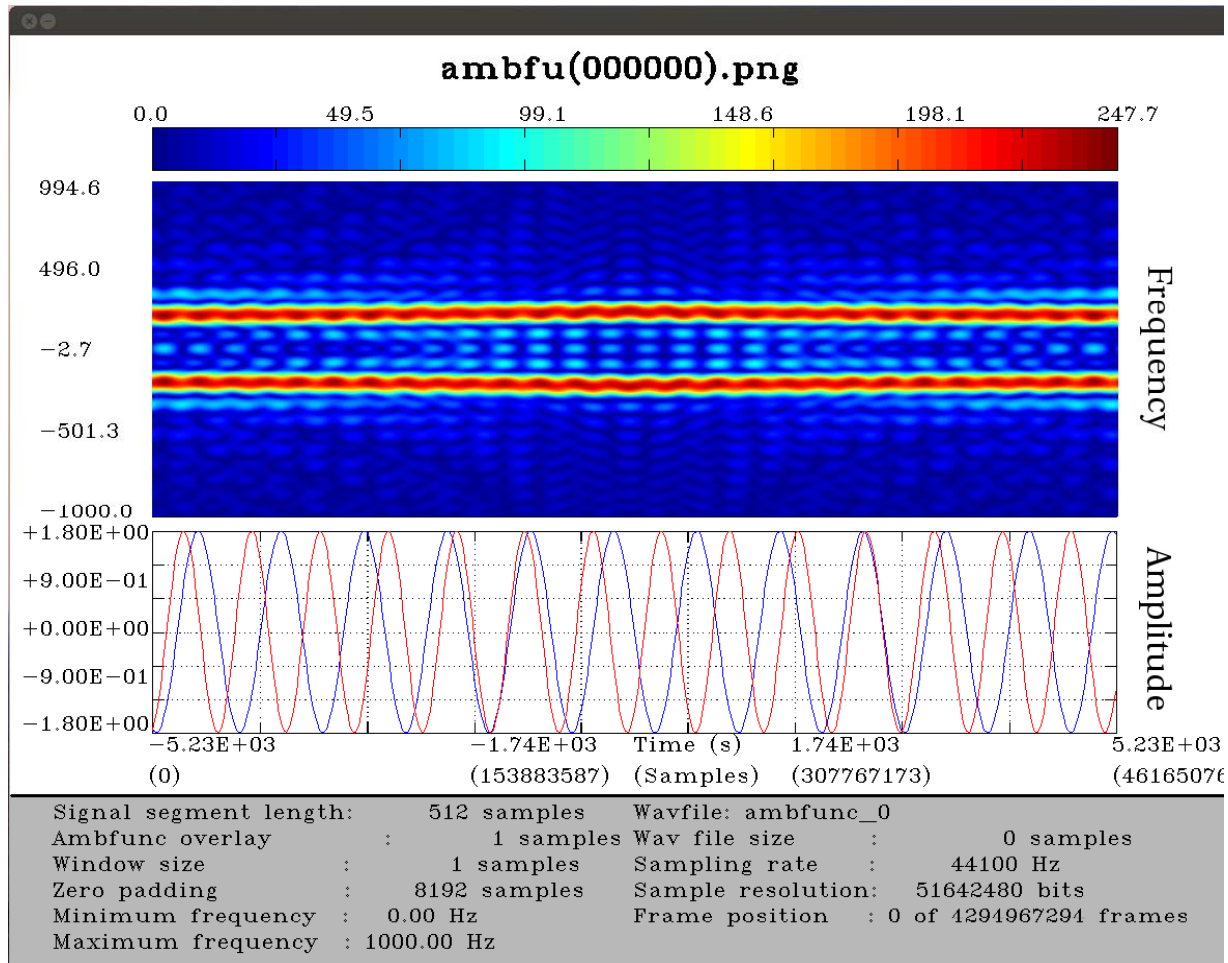
Research contribution

2x2 MIMO



Research contribution

2x2 MIMO

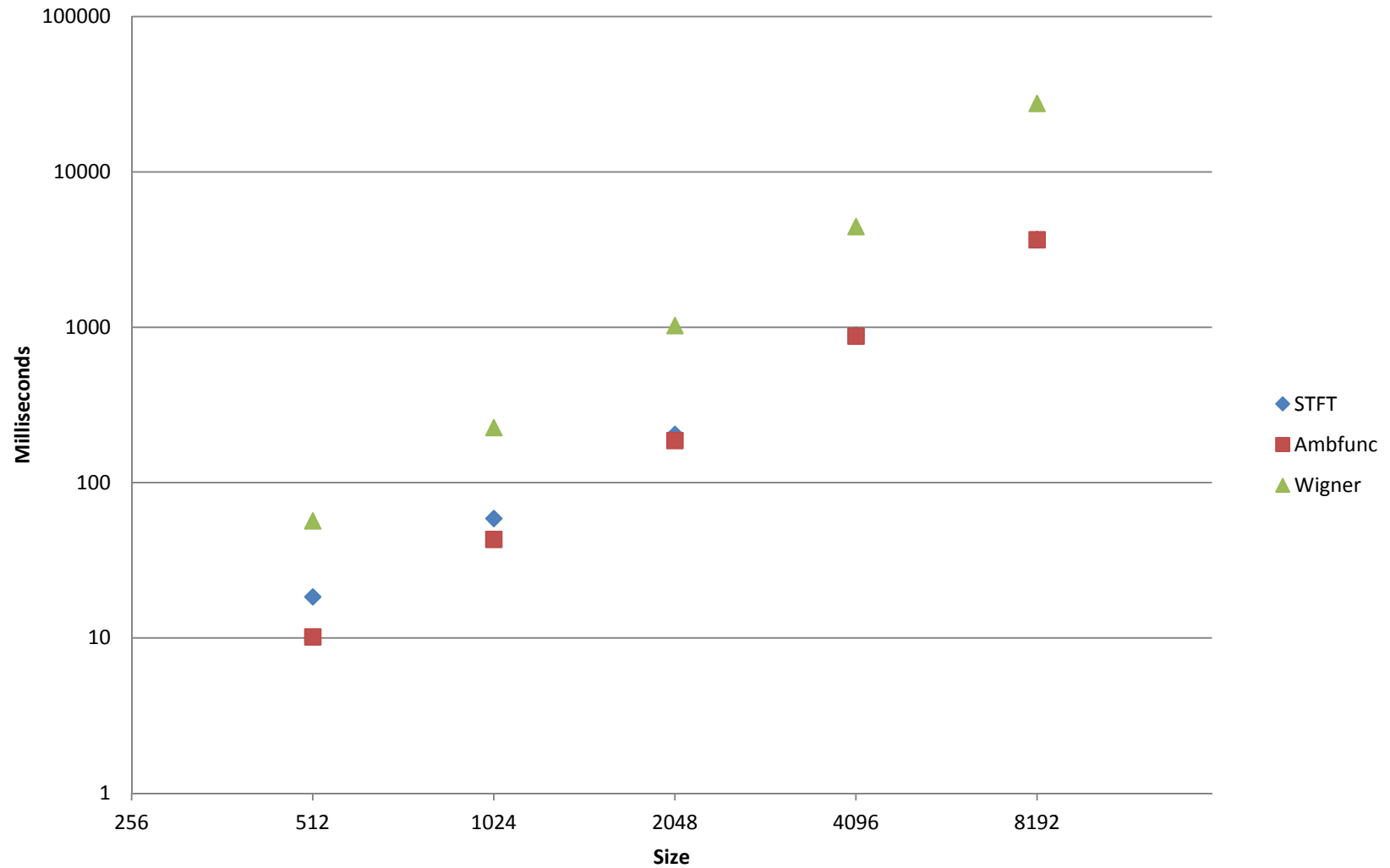


Research contribution

Parallelization Metrics

- Dell 1520
- Intel Core2 T5850
 - 2.16 GHz
 - 4MB Cache
 - 2 Cores, 2 Threads
- 4GB@667MHz
- Linux 2.32
- Samsung RC512
- Intel Core i7 2630QM
 - 2.0 -> 2.9 GHz
 - 6MB Cache
 - 4Cores, 8 Threads
- 6GB@1333MHz
- Linux 3.2

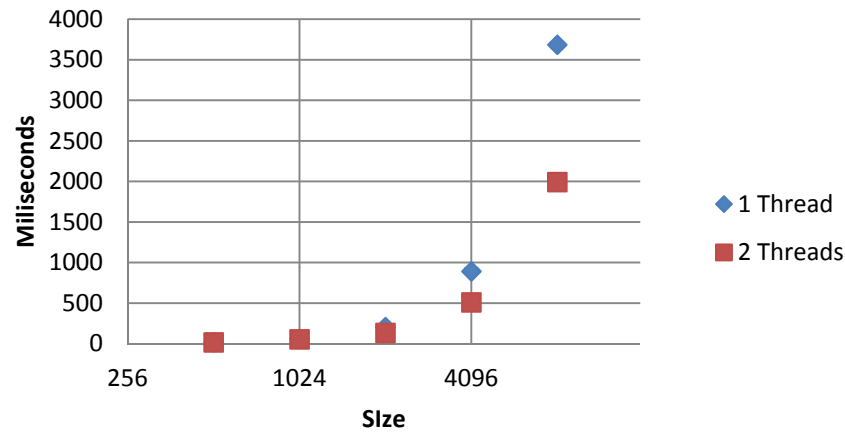
Dell 1520



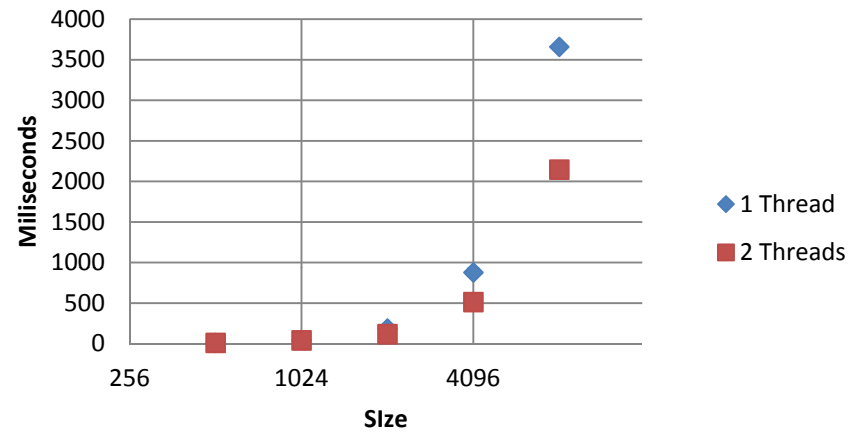
Research contribution

Dell 1520

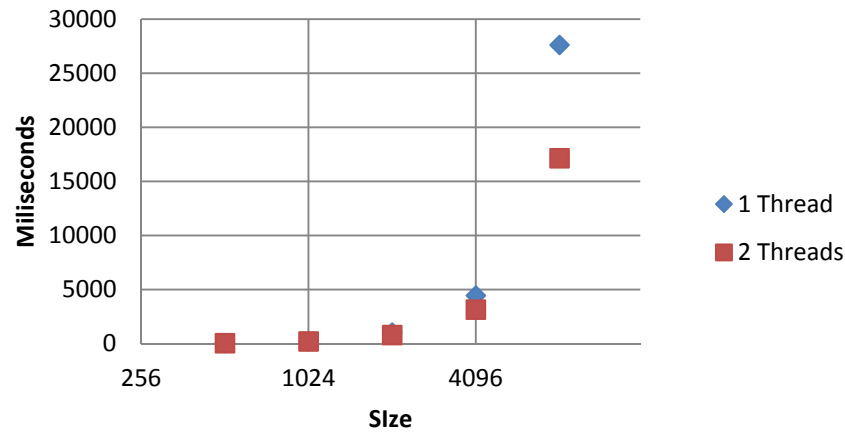
STFT



AmbFunc



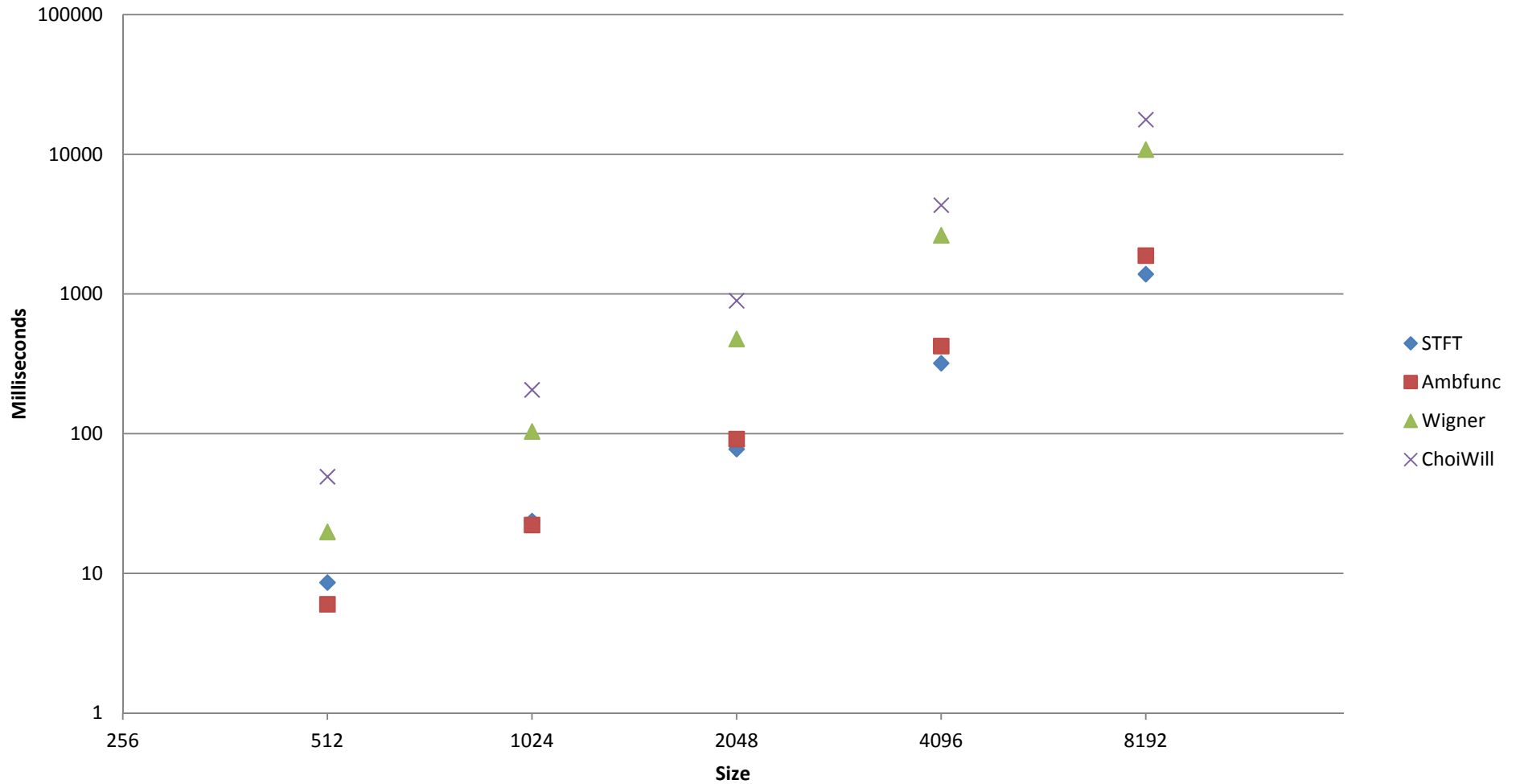
Wigner



Research contribution

Samsung RC512

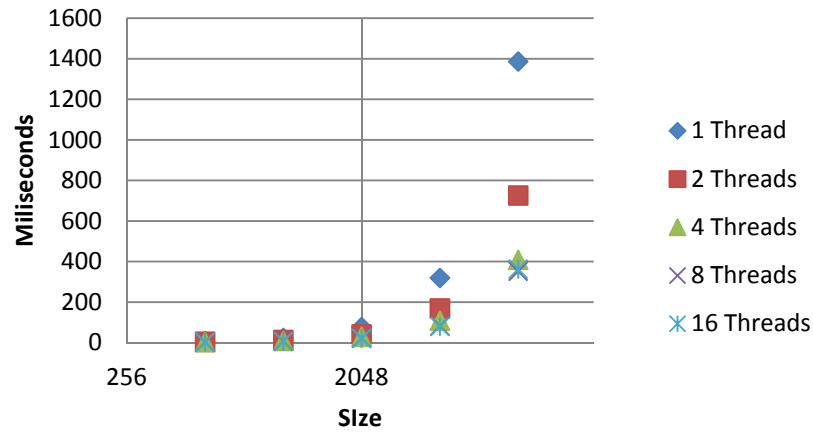
Time-Frequency Representations



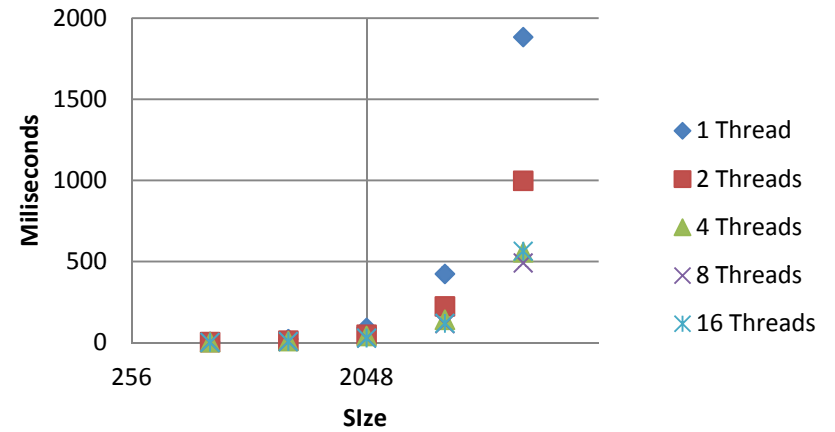
Research contribution

Samsung RC512

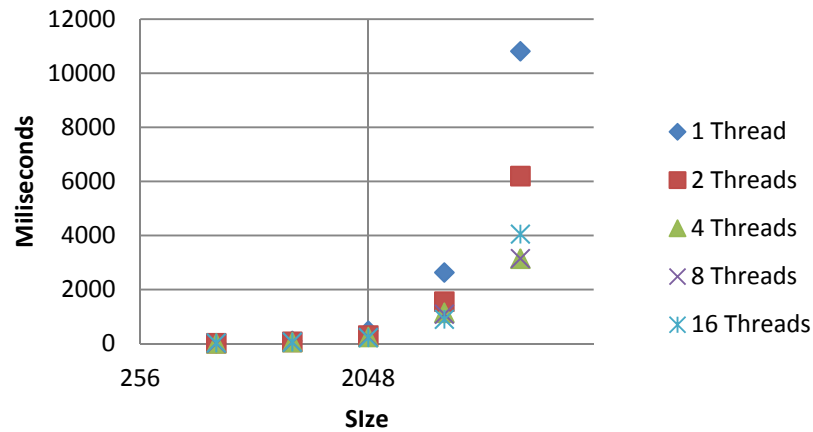
STFT



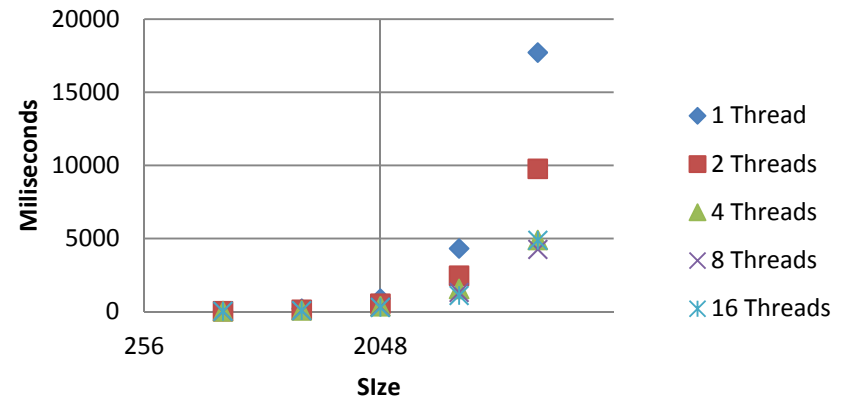
AmbFunc



Wigner



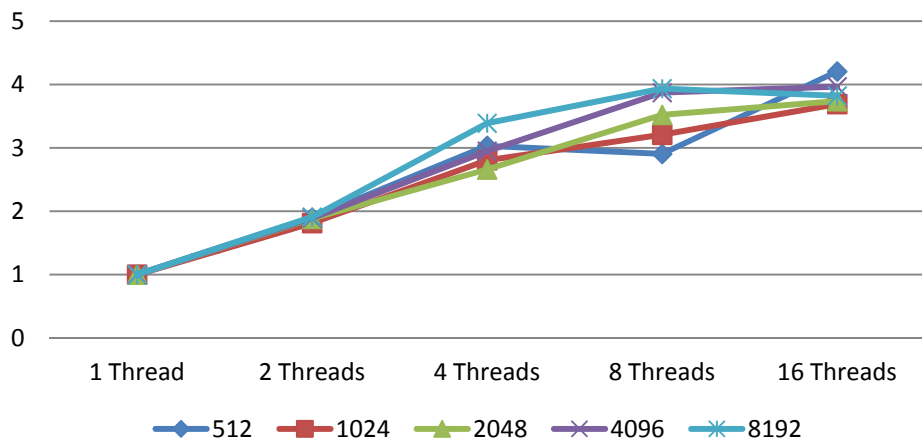
ChoiWill



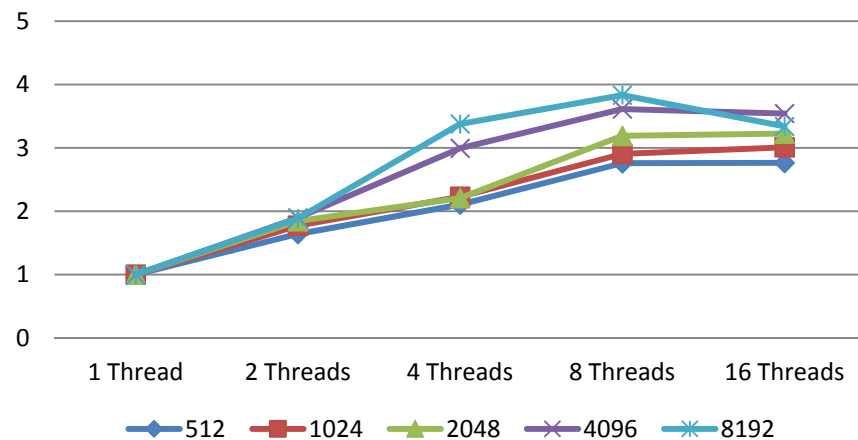
Research contribution

Speed Ups

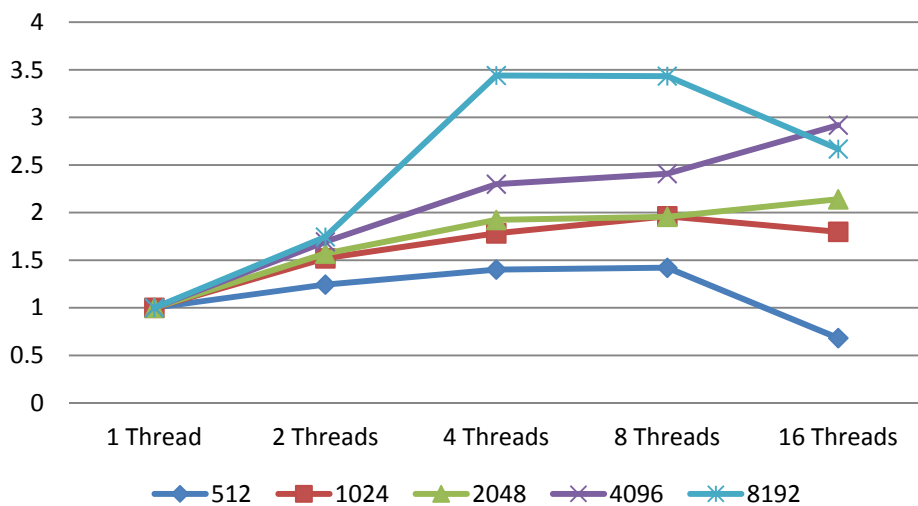
Speed Up STFT



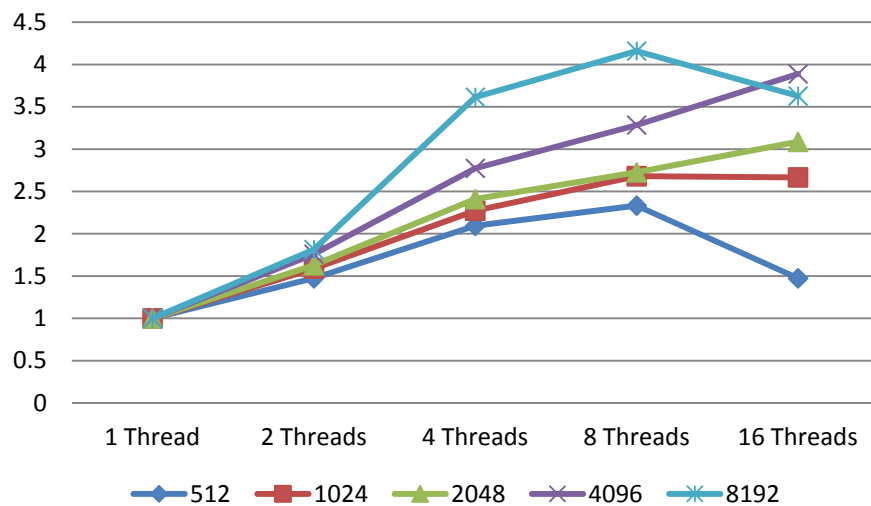
Speed Up AmbFunc



Speed Up Wigner



Speed Up ChoiWill



Conclusions

- A MIMO Modeling framework was presented using a software defined radio paradigm into GNUradio.
- Several time-frequency blocks were designed into GNUradio using the proposed framework.
- A testing workflow was defined using the models and algorithms developed.
- Several works requiring a time-frequency representation can take advantage of the developed framework.

Contributions

- MIMO Channel Modeling
- SIRLAB Web Integration
- SIRLAB Android Integration
- STFT, AmbFunc, Wigner, Choi-Williams SDR Implementation
- MIMO Channel SDR Implementation
- Applications: Channel Surveillance
- Publications:
 - IEEE LASCAS
 - IBERSENSOR

Future work

- Channel Estimation
- Develop Input Signals With the Framework
- DSP, FPGA Integration
- ***NetSig Integration***