SUN

Tutorial on USRP

Xiuzhen Guo

What is SDR?

Software Defined Radio

Joe Mitola of MITRE explicitly

Puts forward the concept of

software radio for the first time

in IEEE National Telesystems

Conference.

Using software to realize the radio functions.

Modulation, demodulation

Spread spectrum, despreading

Synchronization

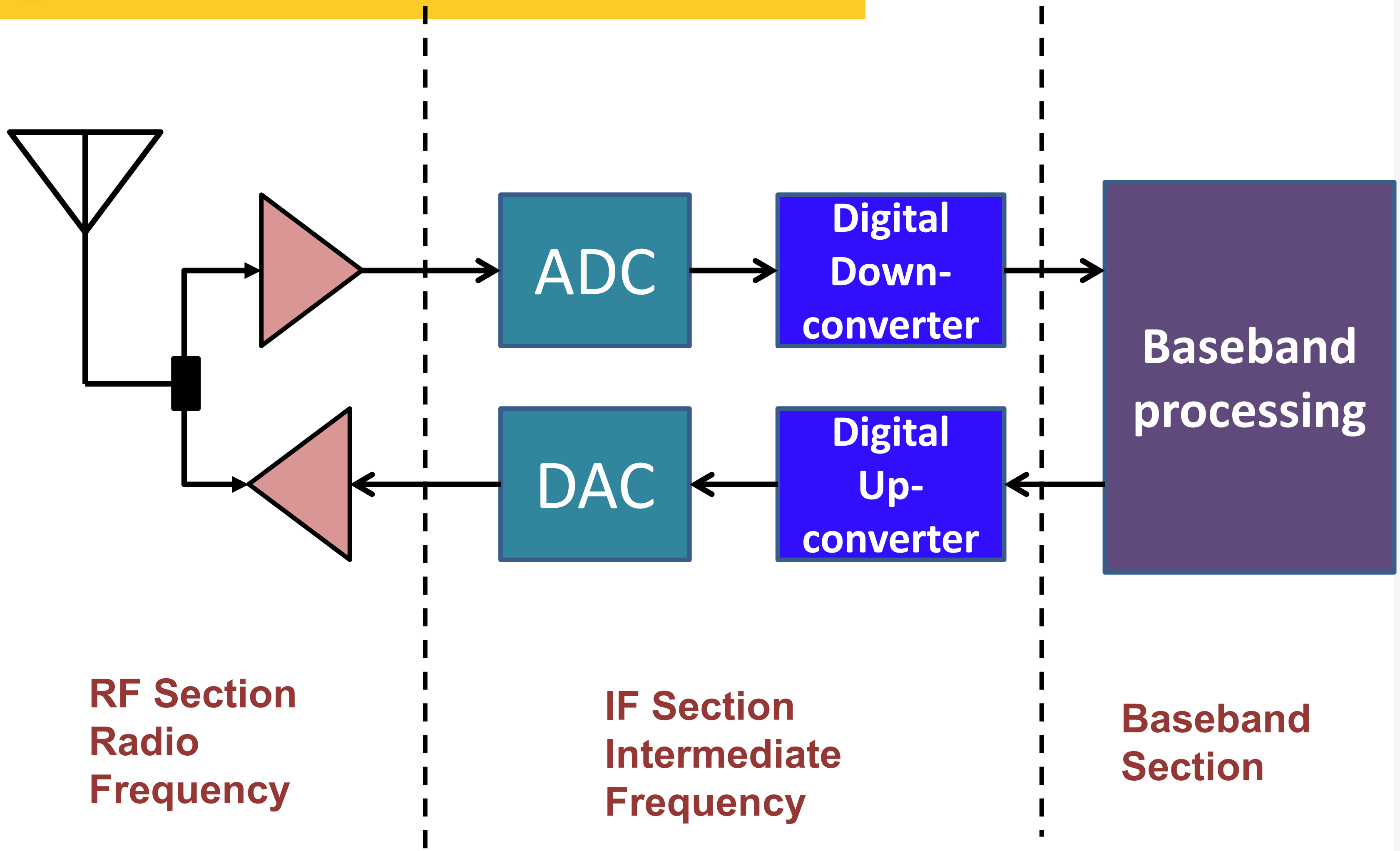
Correlation operation

Filtering

Channel coding and decoding

etc.

Structure of SDR



Advantages of SDR

Function

- Re-configurability
  - Strengthen the possible potential applications
  - Evolve to a new standard conveniently
- Multifunctional devices
  - Multi-mode
  - Multi-band
- Shorter time-to-market
  - compatibility

Performance

- Lower hardware cost
  - hardware module
- widespread applications
  - Different hardware
  - Different demand
- Improve development efficiency
  - open architecture

Definition of USRP

USRP: Universal Software Radio Peripheral

PC ⟶ **RF**

- ◆Used in the communication system in the digital baseband and intermediate frequency section
- ◆All operations related with the waveform are implemented in the CPU
- ◆All of the high speed processes（e.g. digital up and down convert， sample） are implemented in FPGA

What is USRP？

Ettus Research™
A National Instruments Company

History of USRP

Performance

120MHz

bandwidth

16MHz

accuracy

speed

duplexing
40MHz
最高支持
120MHz
Lower latency

USRP RIO

USRP N210

Simplex
20M
Millisecond

16MHz

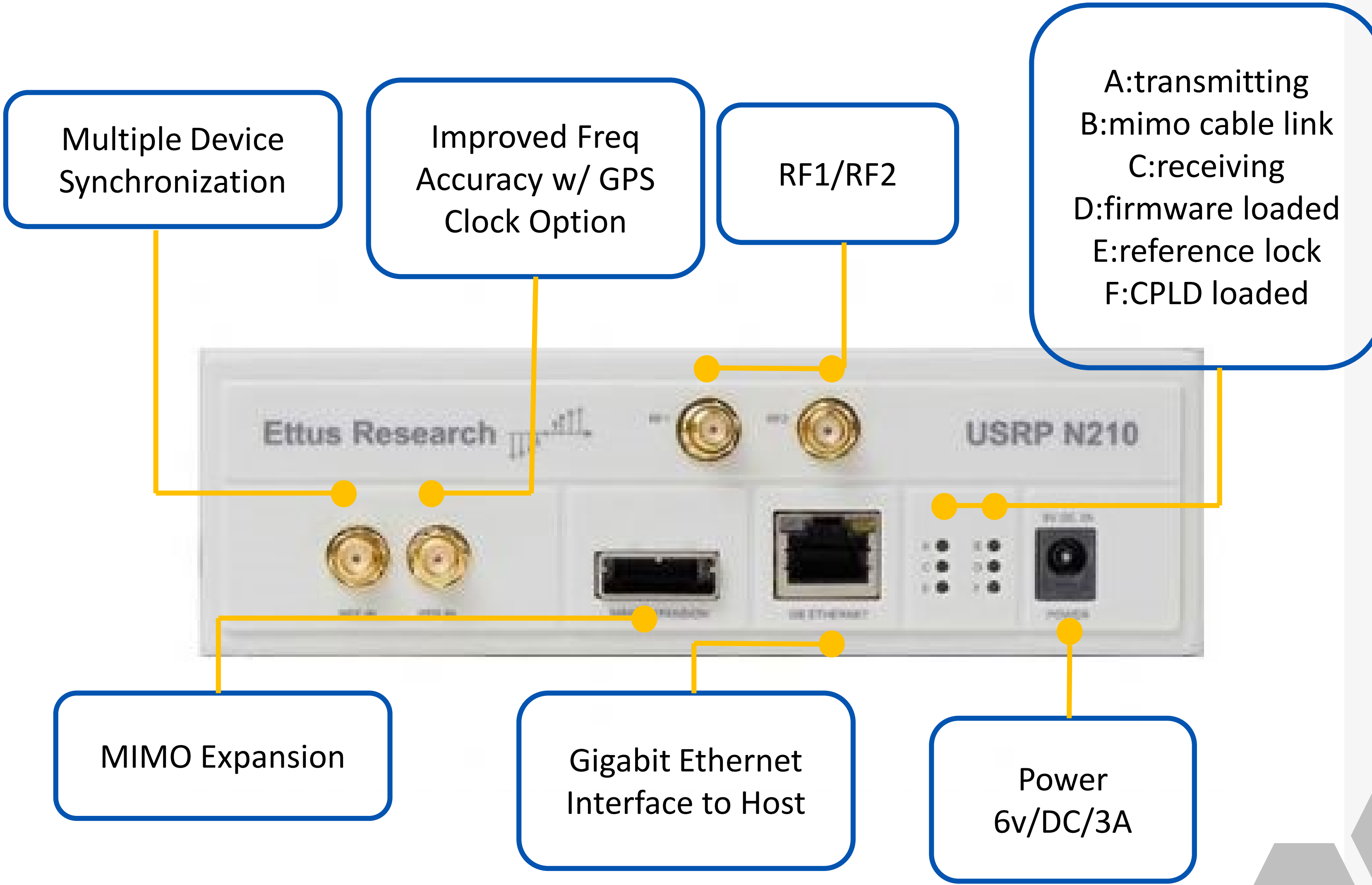
USRP 2

USRP 1

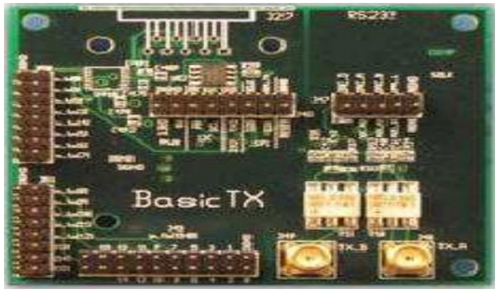
频率范围	应用
50MHz-2.2GHz	FM、雷达、GSM、ISM
70MHz-6GHz	FM、蓝牙、GSM、ISM
…	…

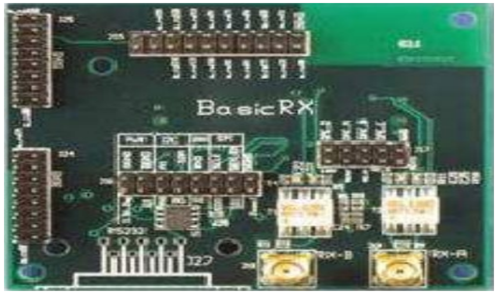
Price

USRP N210

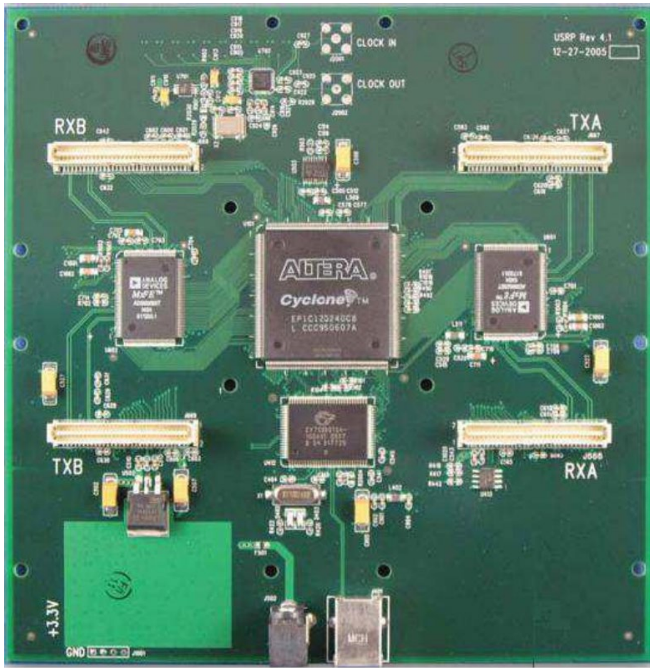

Multiple Device Synchronization
Improved Freq Accuracy w/ GPS Clock Option
RF1/RF2
A:transmitting
B:mimo cable link
C:receiving
D:firmware loaded
E:reference lock
F:CPLD loaded
Ettus Research
USRP N210
MIMO Expansion
Gigabit Ethernet Interface to Host
Power 6v/DC/3A

Compositions



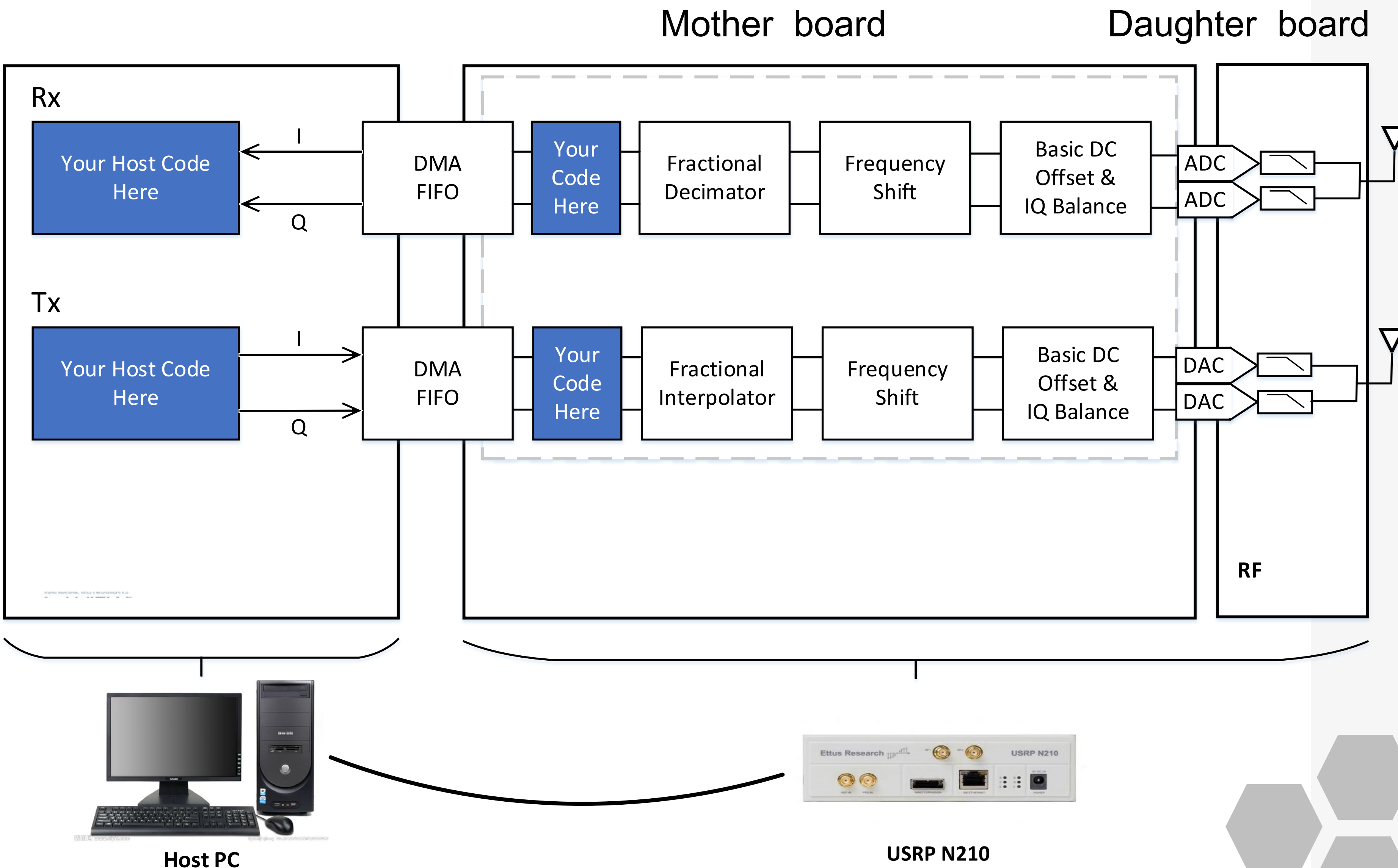


Daughter Board:
RF front end
Cover different frequency range
Convertible



Mother Board:
high-speed signal processing
ADC/DAC/FPGA/I/O

Overall architecture



Features of USRP N210

- Use with GNU Radio, UHD
- Modular Architecture: DC-6 GHz
- Dual 100 MS/s, 14-bit ADC
- Dual 400 MS/s, 16-bit DAC
- DDC/DUC with 25 mHz Resolution
- Up to 50 MS/s Gigabit Ethernet Streaming
- Fully-Coherent MIMO Capability
- Gigabit Ethernet Interface to Host
- 2 Gbps Expansion
- Spartan 3A-DSP 3400 FPGA (N210)
- 1 MB High-Speed SRAM
- Auxiliary Analog and Digital I/O
- 2.5 ppm TCXO Frequency Reference
- 0.01 ppm w/ GPSDO Option

What is GNU Radio?

GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement software radios.

http://gnuradio.org/redmine/projects/gnuradio/wiki

The Installation of GNURadio

```
gxz@ubuntu: ~
gxz@ubuntu:~$ wget http://www.sbrac.org/files/build-gnuradio && chmod a+x ./buil
d-gnuradio && ./build-gnuradio
```

Attention:

1> Updata Ubuntu

2>UHD failed——Network problems

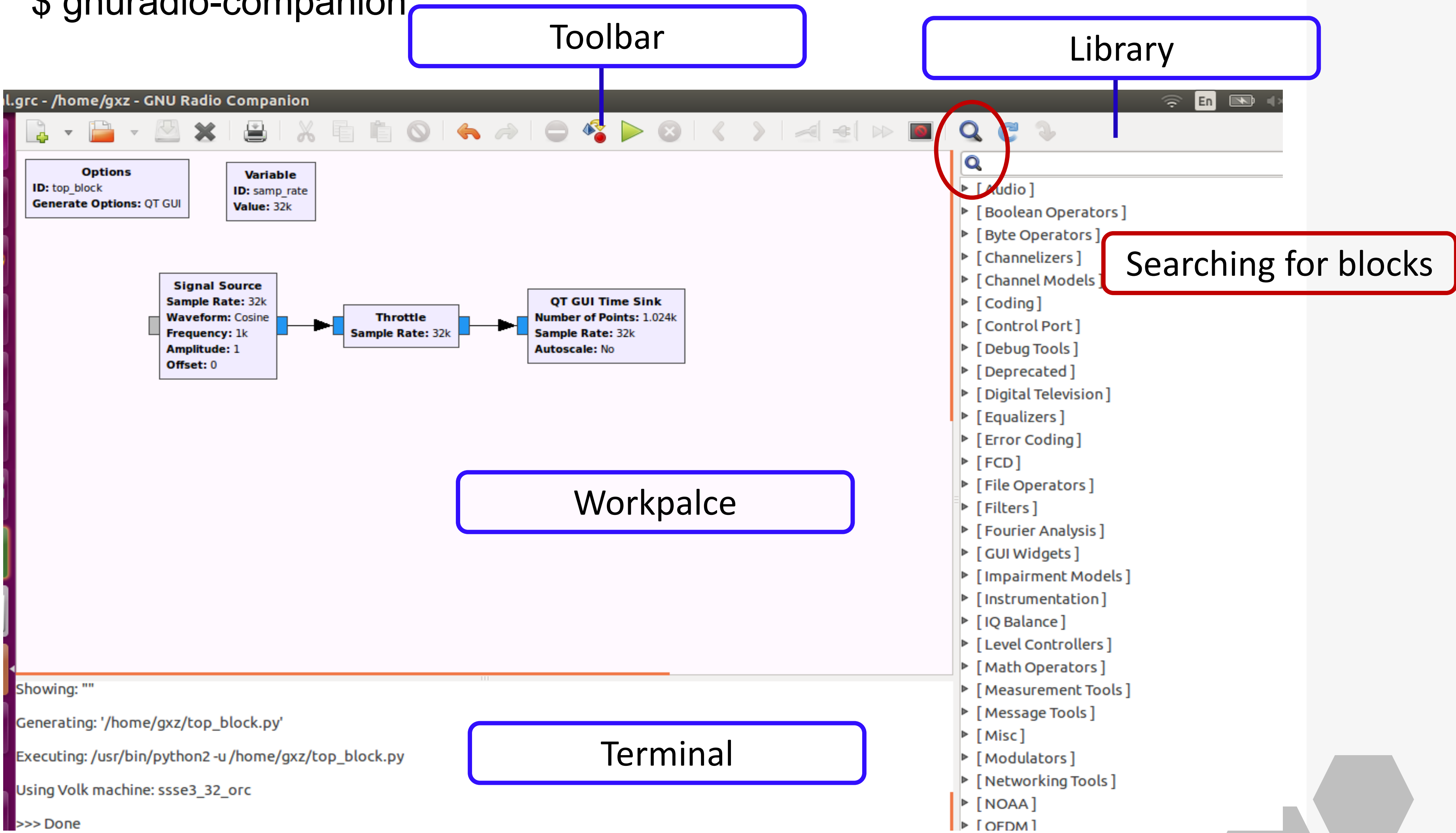
3>Wait quite a while (15:10~18:05)

4>_uhd_find_devices

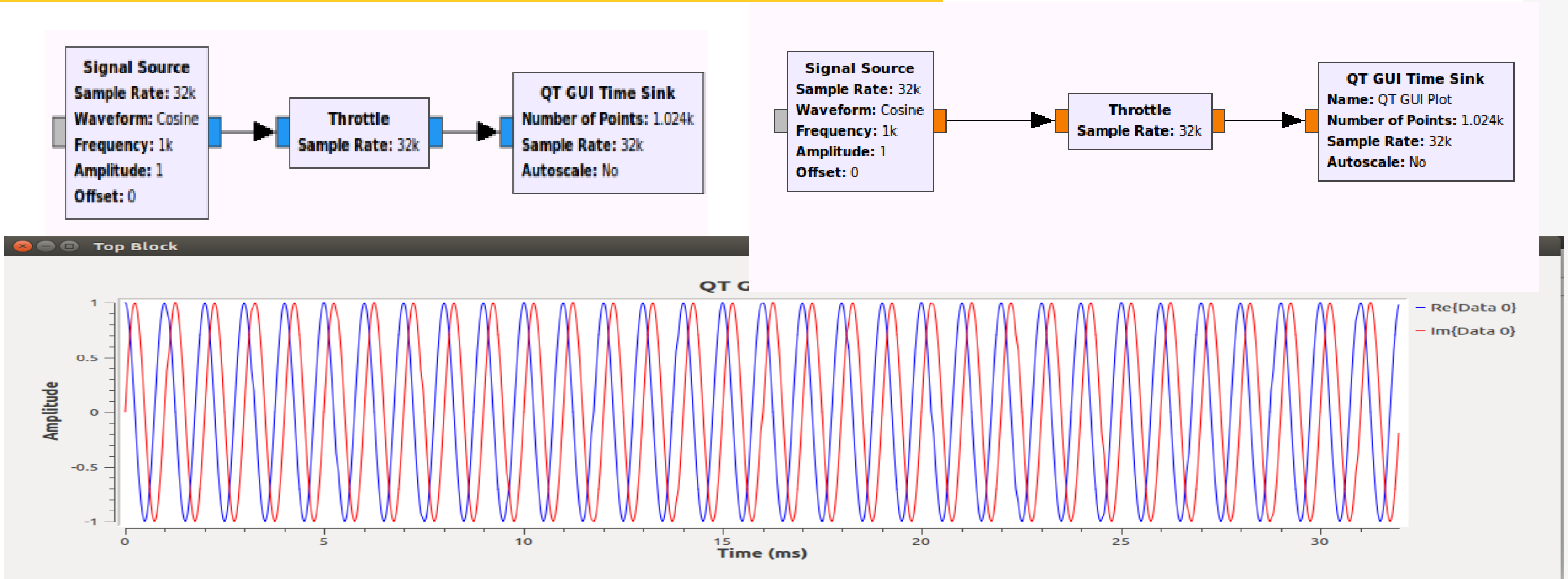
```
gxz@ubuntu:~$ uhd_find_devices
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.git-156-g2d68f228

--------------------------------------------------
-- UHD Device 0
--------------------------------------------------
Device Address:
    type: usrp2
    addr: 192.168.10.2
    name:
    serial: F412B7
```

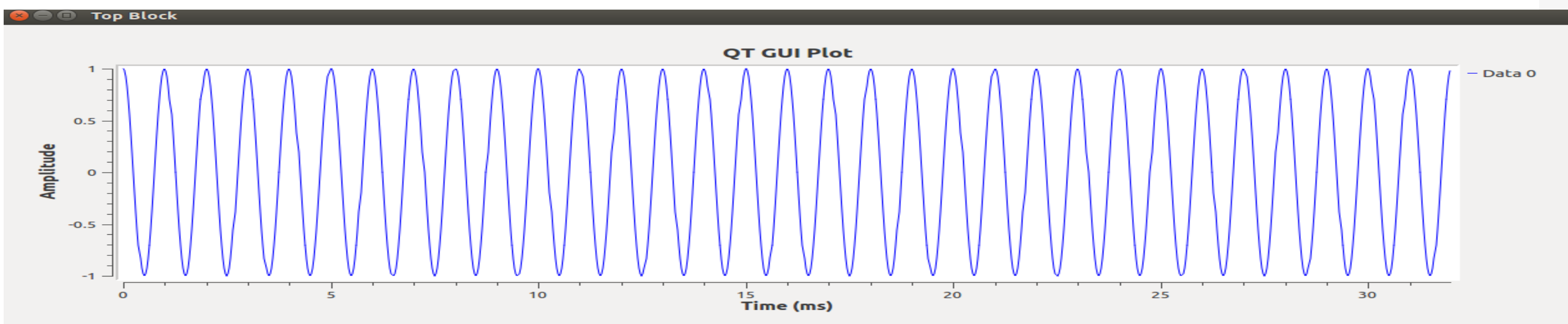
GRC

$ gnuradio-companion

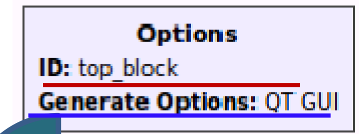


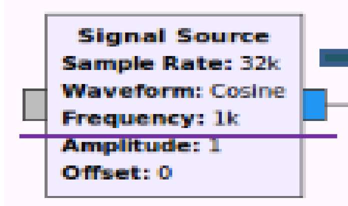
Example

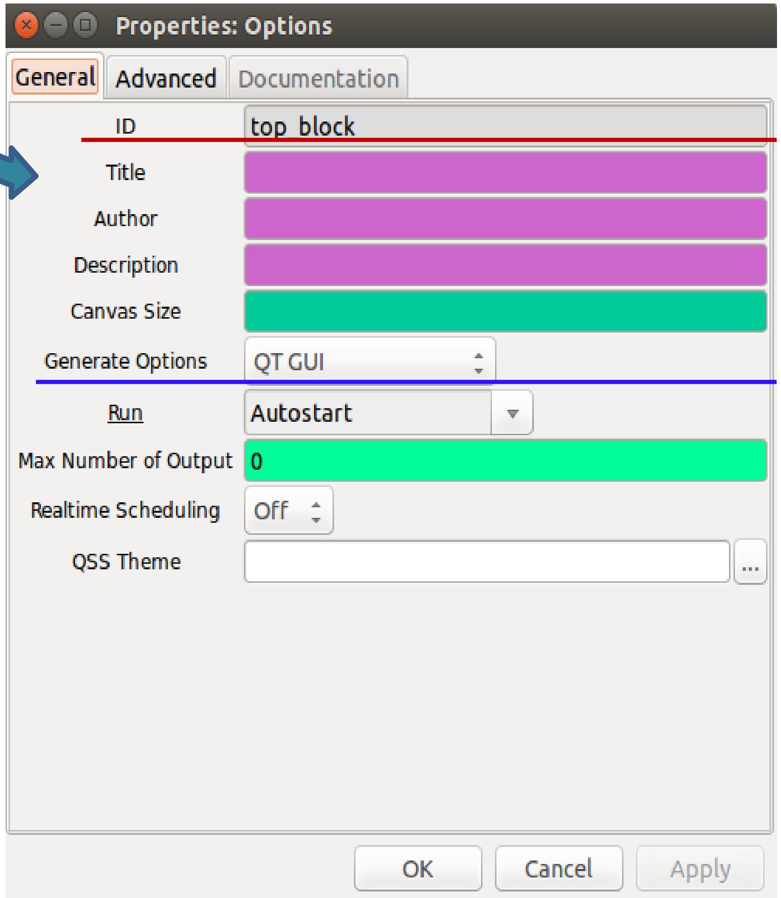

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0
Throttle
Sample Rate: 32k
QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 32k
Autoscale: No
Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0
Throttle
Sample Rate: 32k
QT GUI Time Sink
Name: QT GUI Plot
Number of Points: 1.024k
Sample Rate: 32k
Autoscale: No
Top Block
Amplitude
Time (ms)
Re{Data 0}
Im{Data 0}

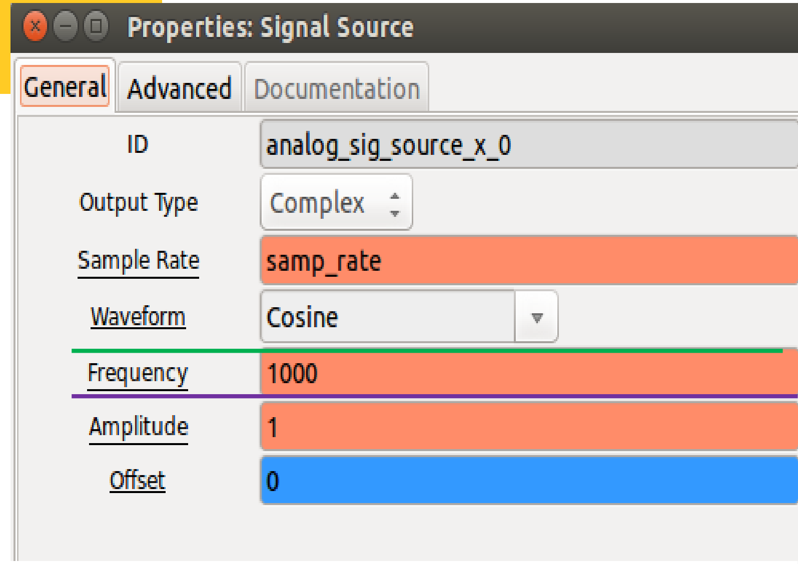

Top Block
QT GUI Plot
Amplitude
Time (ms)
Data 0

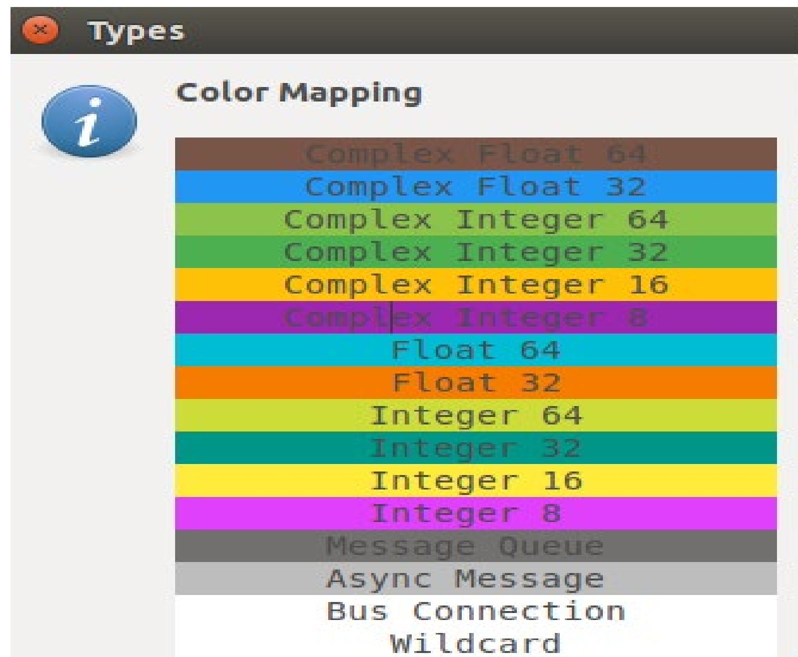
Modifying block properties











Blocks

Singnal sources
gr.sig_source_X
gr.noise_source_X
gr.null_source
gr.vector_source_X
gr.file_source
gr.audio_source
usrp.source_c

Type conversions
gr.complex_to _float
Gr.float_to_short

Signal sinks
gr.null_sink
gr.vector_sink_X
gr.file_sink
gr.audio_sink
usrp.source_c

Filters
gr.firdes::low_pass
gr.firdes::hilbert
gr.firdes::root_raised_cosine
gr.firdes::gaussian

Simple operators
gr.add_const_XX
gr.add_XX
gr.sub_XX
gr.multiply_const_XX
gr.multply_XX
gr.divide_XX
gr.nlog10_ff

FFT
gr.fft_vcc
gr.fft_vfc

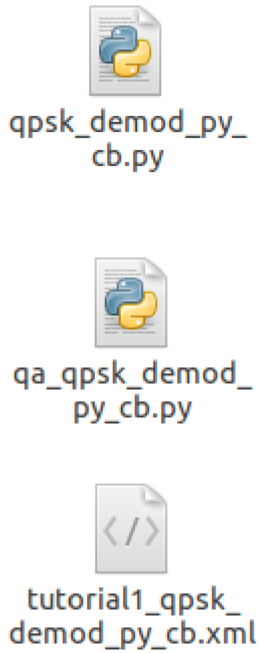
How to make our own blocks?

1.Using gr_modtool

```
gxz@ubuntu:~$ gr_modtool help
Usage:
gr_modtool <command> [options] -- Run <command> with the given options.
gr_modtool help -- Show a list of commands.
gr_modtool help <command> -- Shows the help for a given command.

List of possible commands:

Name      Aliases          Description
=====================================================================
disable   dis              Disable block (comments out CMake entries for files)
info      getinfo,inf      Return information about a given module
remove    rm,del           Remove block (delete files and remove Makefile entrie
s)
makexml   mx               Make XML file for GRC block bindings
add       insert           Add block to the out-of-tree module.
newmod    nm,create        Create a new out-of-tree module
rename    insert           Add block to the out-of-tree module.
gxz@ubuntu:~$
```

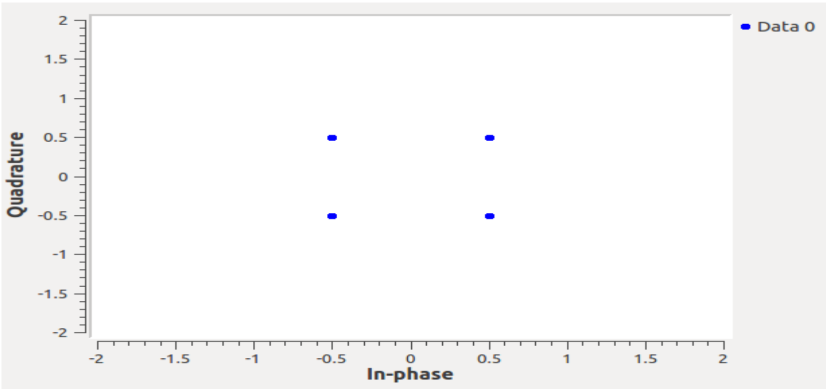
Block:qpsk_demod

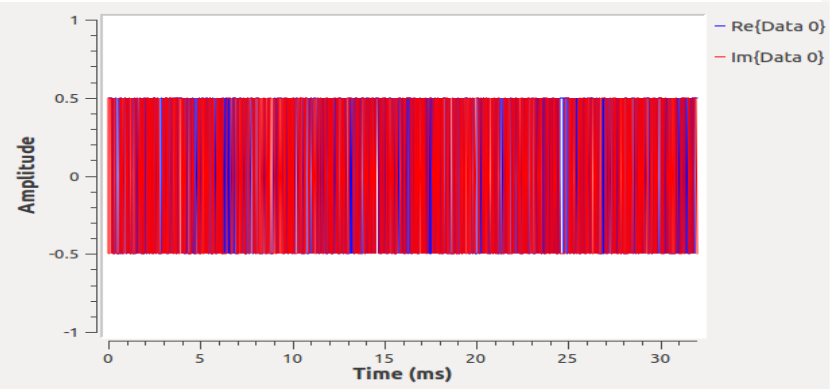
```
gxz@gxz:~$ gr_modtool newmod tutorial1
Creating out-of-tree module in ./gr-tutorial1... Done.
Use 'gr_modtool add' to add a new block to this currently empty module.
gxz@gxz:~$ cd gr-tutorial1
gxz@gxz:~/gr-tutorial1$ gr_modtool add -t sync -l python
GNU Radio module name identified: tutorial1
Language: Python
Enter name of block/code (without module name prefix): qpsk_demod_py_cb
Block/code identifier: qpsk_demod_py_cb
Enter valid argument list, including default arguments: gray_code
Add Python QA code? [Y/n] y
Adding file 'python/qpsk_demod_py_cb.py'...
```

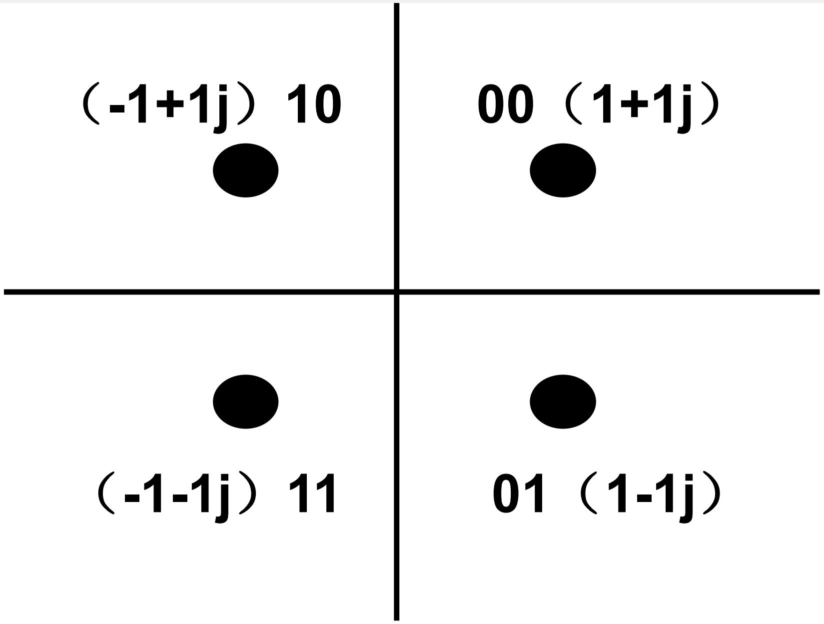


- Synchronous (1:1) - Number of items at input port equals the number of items at output port
- Decimation (N:1) - Number of input items is a fixed multiple of the number of output items
- Interpolation (1:M) - Number of output items is a fixed multiple of the number of input items.
- General/Basic (N:M) - Provides no relation between the number of input items and the number of output items.

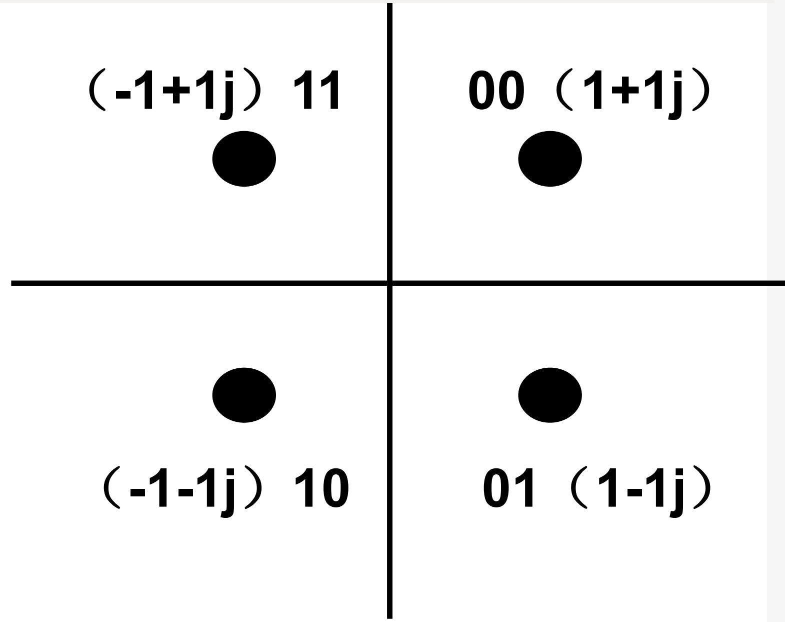
QPSK(4QAM)







With Gray coding



Without Gray coding

Modifying the python block file

```python
import numpy
from gnuradio import gr

class qpsk_demod_py_cb(gr.sync_block):
    """
    docstring for block qpsk_demod_py_cb
    """
    def __init__(self, gray_code):
        self.gray_code=gray_code
        gr.sync_block.__init__(self,
            name="qpsk_demod_py_cb",
            in_sig=[numpy.complex64],
            out_sig=[numpy.uint8])

    def get_minimum_distances(self,sample):
        if self.gray_code==1:
            if(sample.imag>=0 and sample.real>=0):
                return 0 #1+1j
            elif(sample.imag>=0 and sample.real<0):
                return 2 #-1+1j
            elif(sample.imag<0 and sample.real<0):
                return 3 #-1-1j
            elif(sample.imag<0 and sample.real>=0):
                return 2 #1-1j
        else:
            if(sample.imag>=0 and sample.real>=0):
                return 0 #1+1j
            elif(sample.imag>=0 and sample.real<0):
                return 3 #-1+1j
            elif(sample.imag<0 and sample.real<0):
                return 2 #-1-1j
            elif(sample.imag<0 and sample.real>=0):
                return 1 #1-1j
```

```python
def work(self, input_items, output_items):
    in0 = input_items[0]
    out = output_items[0]
    # <+signal processing here+>
    for i in range(0,len(in0)):
        sample=in0[i]
        out[i]=self.get_minimum_distances(sample)
    return len(output_items[0])
```

QA tests

```
from gnuradio import gr, gr_unittest
from gnuradio import blocks
from qpsk_demod_py_cb import qpsk_demod_py_cb

class qa_qpsk_demod_py_cb (gr_unittest.TestCase):

    def setUp (self):
        self.tb = gr.top_block ()

    def tearDown (self):
        self.tb = None

    def test_001_t (self):
        # set up fg
        gray_code=False
        src_data=((-1-1j), (-1+1j), (1+1j), (1-1j))
        expected_result=(2, 3, 0, 1)
        src=blocks.vector_source_c(src_data)
        qpsk=qpsk_demod_py_cb(gray_code)
        snk=blocks.vector_sink_b()
        self.tb.connect(src,qpsk)
        self.tb.connect(qpsk,snk)
        self.tb.run ()
        result_data=snk.data()
        self.assertFloatTuplesAlmostEqual (expected_result,result_data,6)
        # check data
```

QA tests

```python
    def test_002_t (self):
        # set up fg
        gray_code=True
        src_data=((-1-1j), (-1+1j), (1+1j), (1-1j))
        expected_result=(3, 2, 0, 1)
        src=blocks.vector_source_c(src_data)
        qpsk=qpsk_demod_py_cb(gray_code)
        snk=blocks.vector_sink_b()
        self.tb.connect(src,qpsk)
        self.tb.connect(qpsk,snk)
        self.tb.run ()
        result_data=snk.data()
        self.assertFloatTuplesAlmostEqual (expected_result,result_data,6)
        # check data

if __name__ == '__main__':
    gr_unittest.run(qa_qpsk_demod_py_cb, "qa_qpsk_demod_py_cb.xml")
```

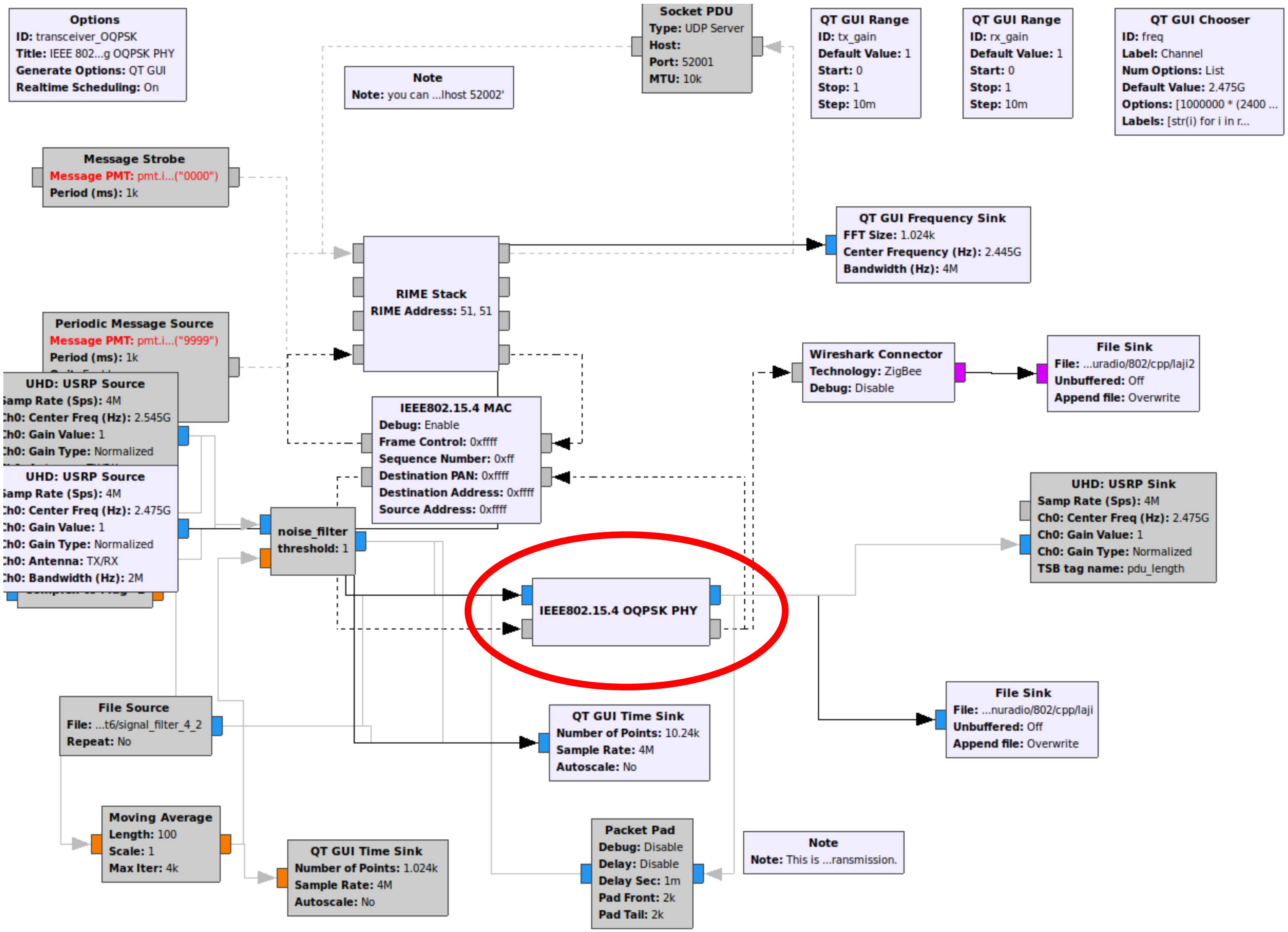
gxz@ubuntu: ~/gr-tutorial/python

```
gxz@ubuntu:~$ cd gr-tutorial/python
gxz@ubuntu:~/gr-tutorial/python$ python qa_qpsk_demod_py_cb.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.007s

OK
gxz@ubuntu:~/gr-tutorial/python$
```

XML files

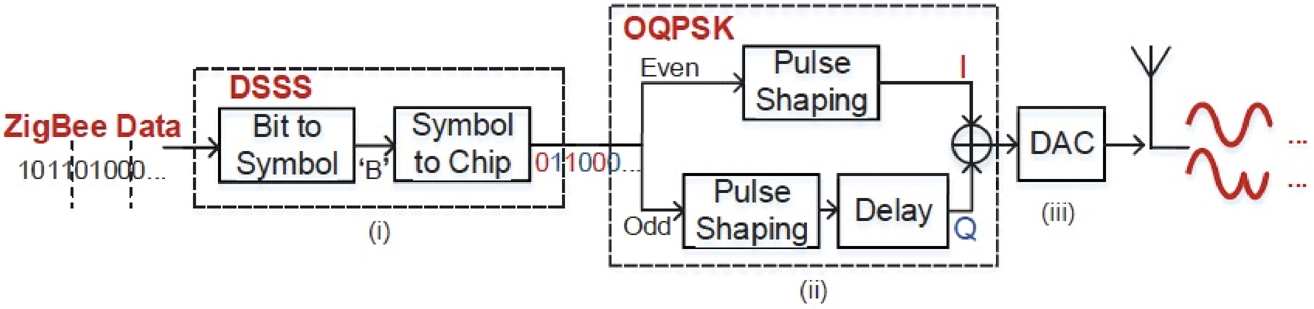
```xml
<?xml version="1.0"?>
<block>
  <name>qpsk_demod_py_cb</name>
  <key>tutorial1_qpsk_demod_py_cb</key>
  <category>tutorial1</category>
  <import>import tutorial1</import>
  <make>tutorial1.qpsk_demod_py_cb($gray_code)</make>
  <!-- Make one 'param' node for every Parameter you wan
       Sub-nodes:
       * name
       * key (makes the value accessible as $keyname, e.
       * type -->
  <param>
    <name>Gray Code</name>
    <key>gray_code</key>
    <type>int</type>
  </param>

  <!-- Make one 'sink' node per input. Sub-nodes:
       * name (an identifier for the GUI)
       * type
       * vlen
       * optional (set to 1 for optional inputs) -->
  <sink>
    <name>in</name>
    <type>complex</type>
  </sink>

  <source>
    <name>out</name>
    <type>byte</type>
  </source>
</block>
```

Application: ZigBee

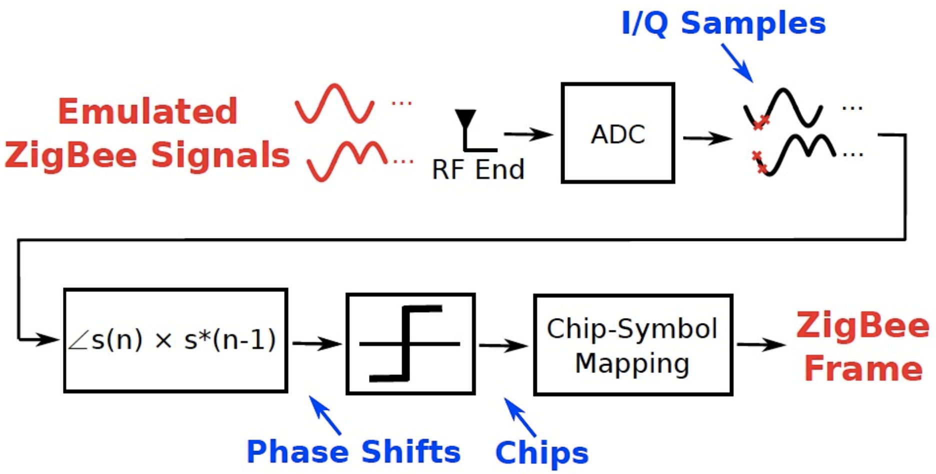


Application: ZigBee

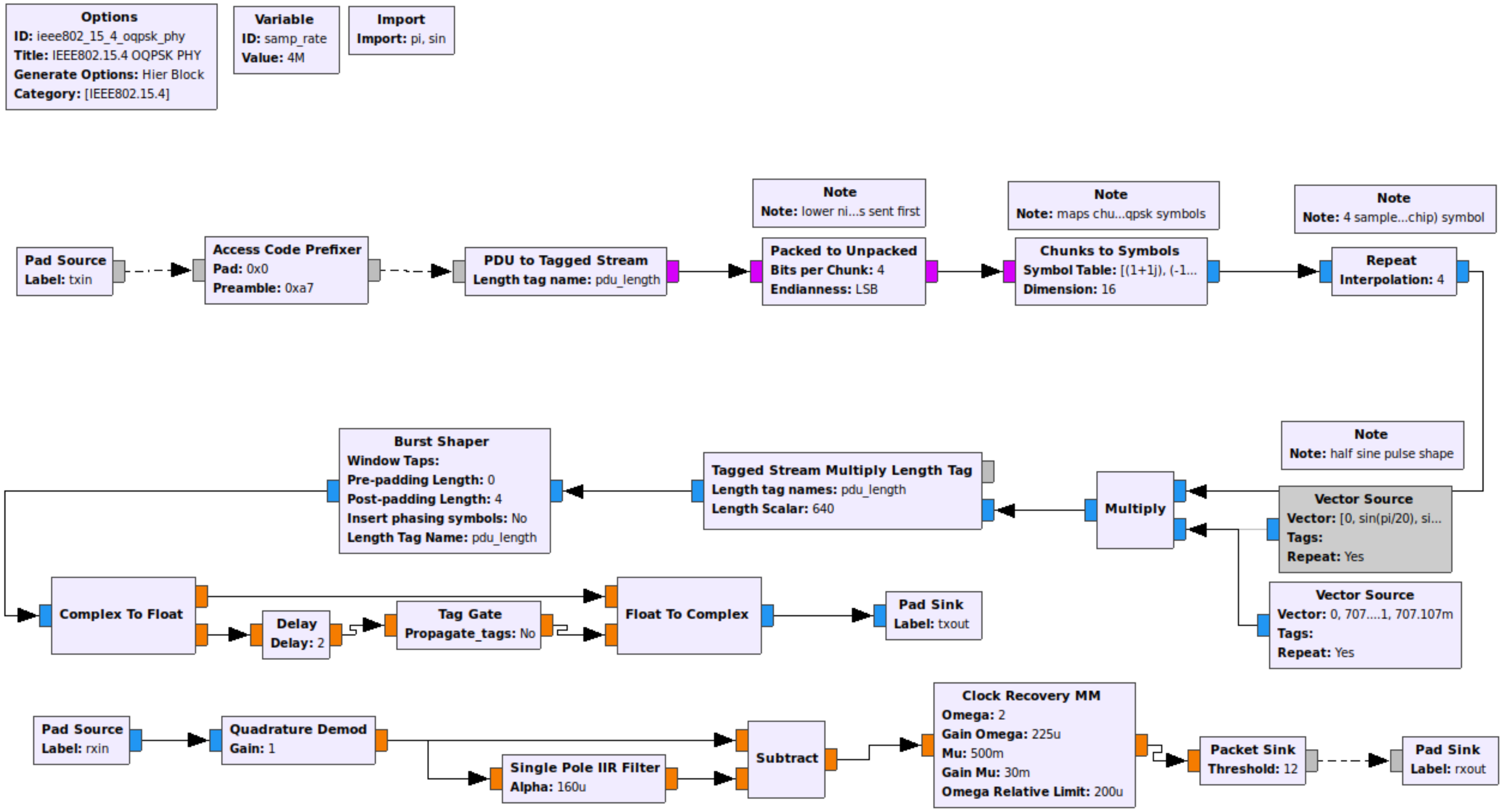
Tx



Rx



Application

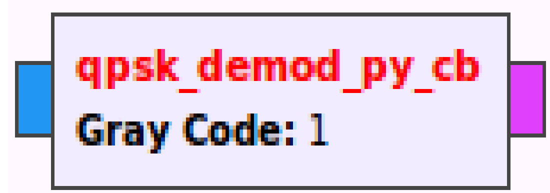


Backup

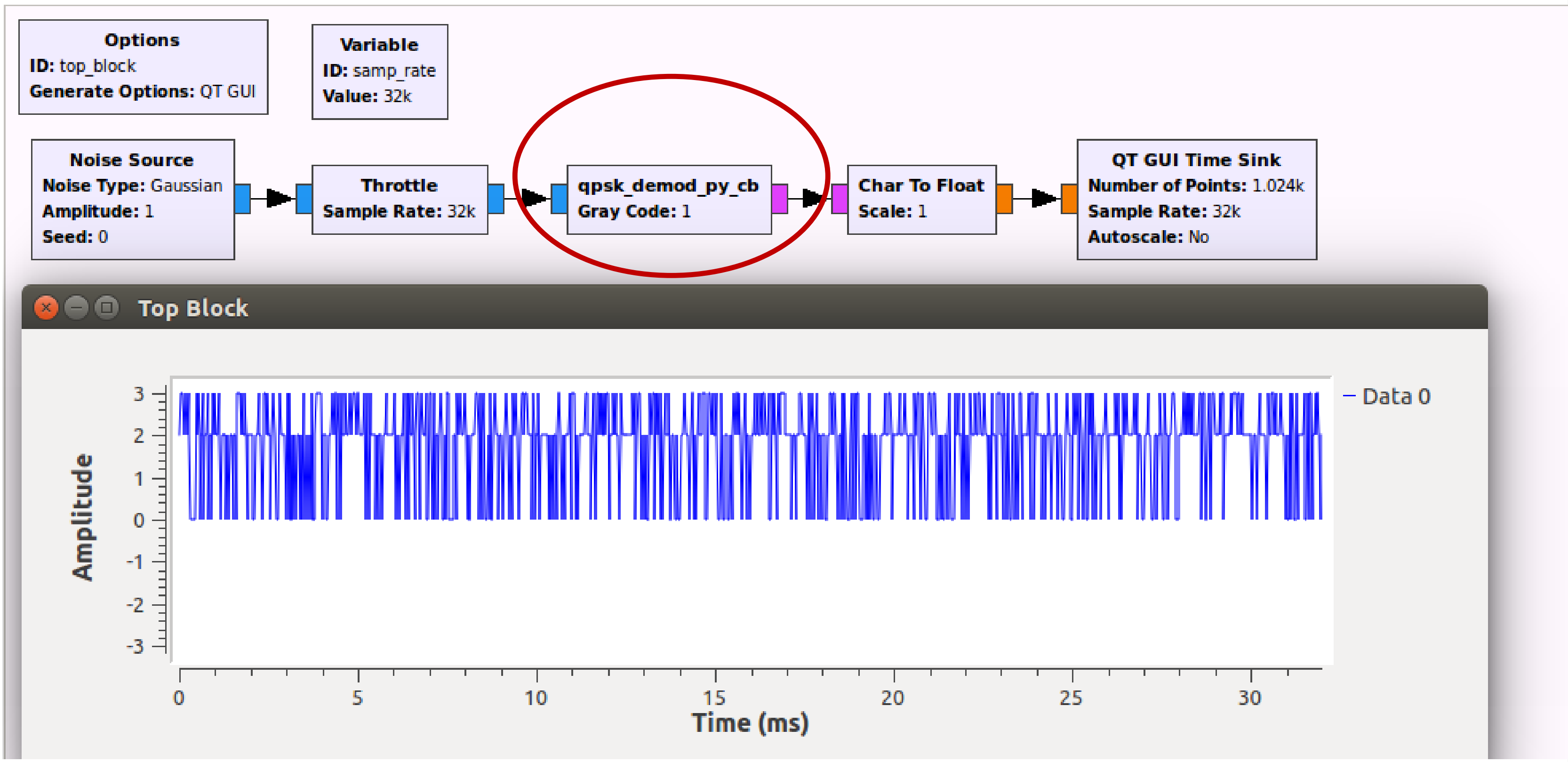
Installation

```
gxz@gxz:~/gr-tutorial1/build$ make
Scanning dependencies of target pygen_python_b2675
[ 33%] Generating __init__.pyc, qpsk_demod_py_cb.pyc
[ 66%] Generating __init__.pyo, qpsk_demod_py_cb.pyo
[ 66%] Built target pygen_python_b2675
Scanning dependencies of target pygen_apps_9a6dd
[ 66%] Built target pygen_apps_9a6dd
Scanning dependencies of target doxygen_target
[100%] Generating documentation with doxygen
Warning: Tag `XML_SCHEMA' at line 1510 of file `/home/gxz/gr-tutorial1/bui
         To avoid this warning please remove this line from your configura
Warning: Tag `XML_DTD' at line 1516 of file `/home/gxz/gr-tutorial1/build/
         To avoid this warning please remove this line from your configura
[100%] Built target doxygen_target
gxz@gxz:~/gr-tutorial1/build$ sudo make install
[sudo] password for gxz:
[ 66%] Built target pygen_python_b2675
[ 66%] Built target pygen_apps_9a6dd
[100%] Built target doxygen_target
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/local/lib/cmake/tutorial1/tutorial1Config.cmake
-- Installing: /usr/local/include/tutorial1/api.h
-- Installing: /usr/local/lib/python2.7/dist-packages/tutorial1/__init__.p
```

- [Stream Tag Tools]
- [Symbol Coding]
- [Synchronizers]
- [Trellis Coding]
- [tutorial]
- [tutorial1]
  - qpsk_demod_py_cb
- [Type Converters]
- [UHD]
- [Variables]
- [Video]
- [Waveform Generators]

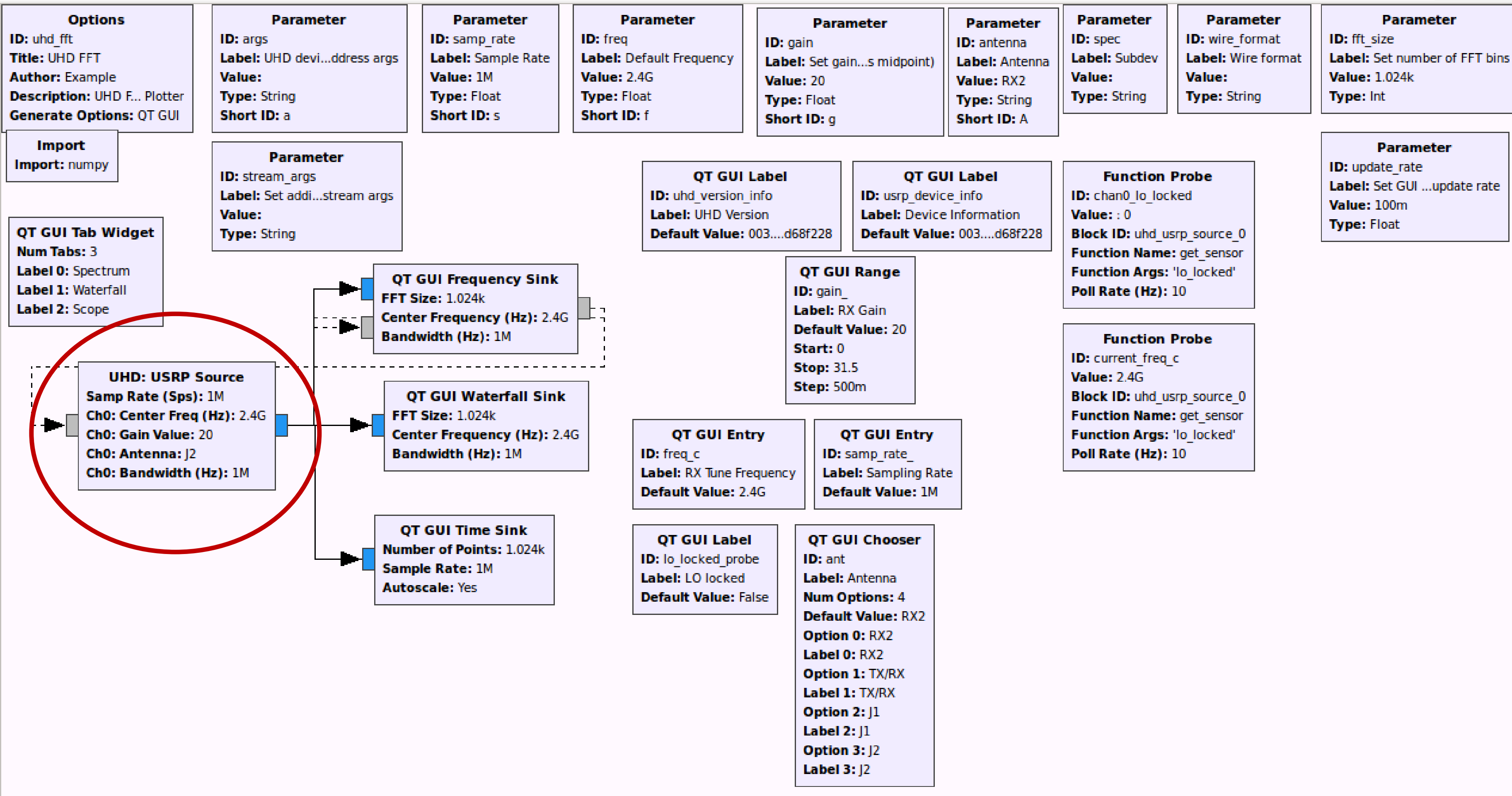


Running

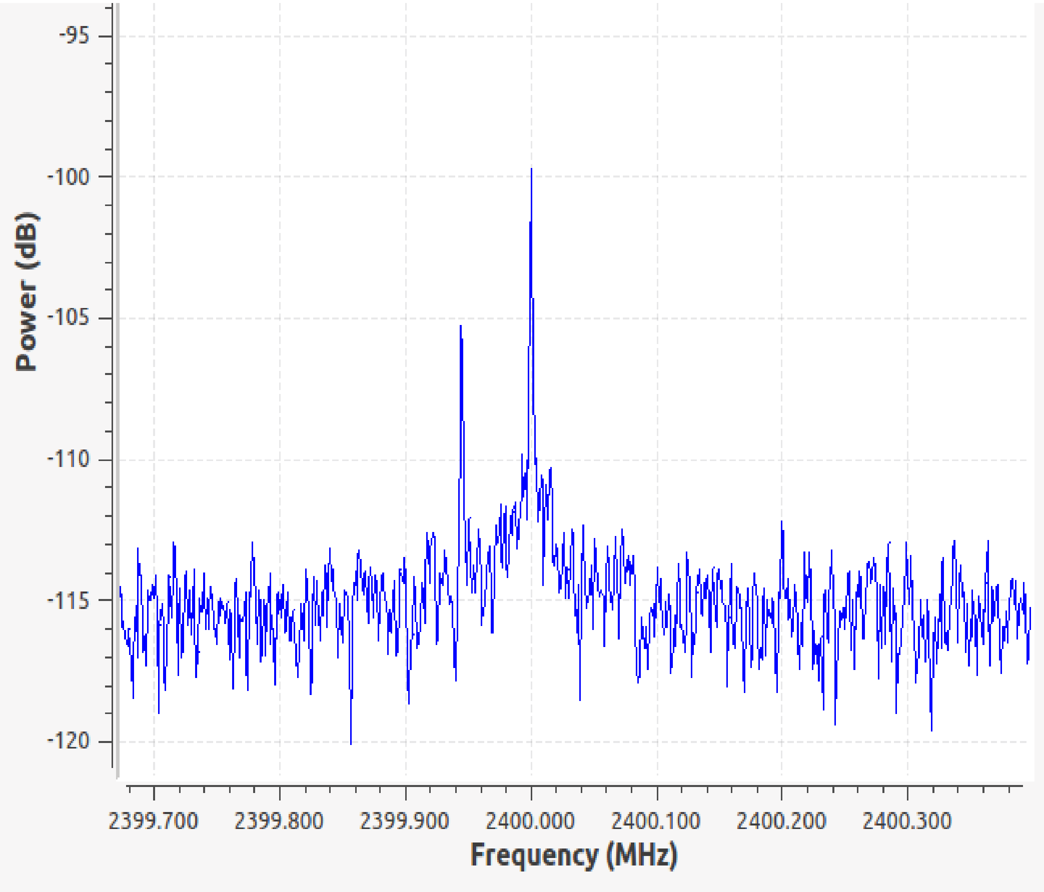


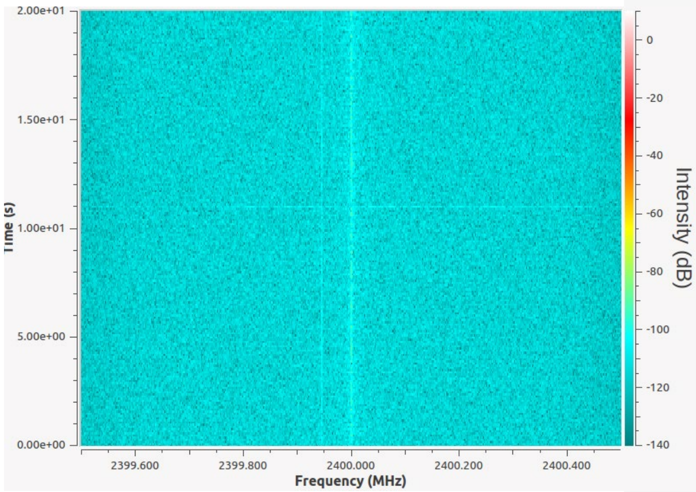
Options
ID: top_block
Generate Options: QT GUI
Variable
ID: samp_rate
Value: 32k
Noise Source
Noise Type: Gaussian
Amplitude: 1
Seed: 0
Throttle
Sample Rate: 32k
qpsk_demod_py_cb
Gray Code: 1
Char To Float
Scale: 1
QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 32k
Autoscale: No
Top Block
Data 0
Amplitude
Time (ms)

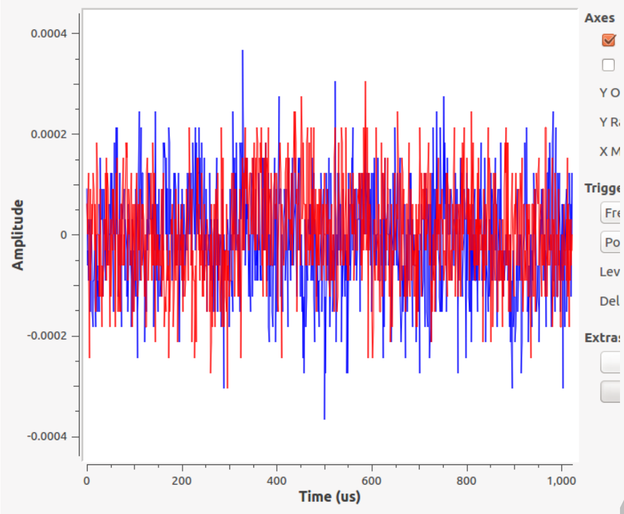
Uhd_fft(2.4GHz)



Uhd_fft(2.4GHz)

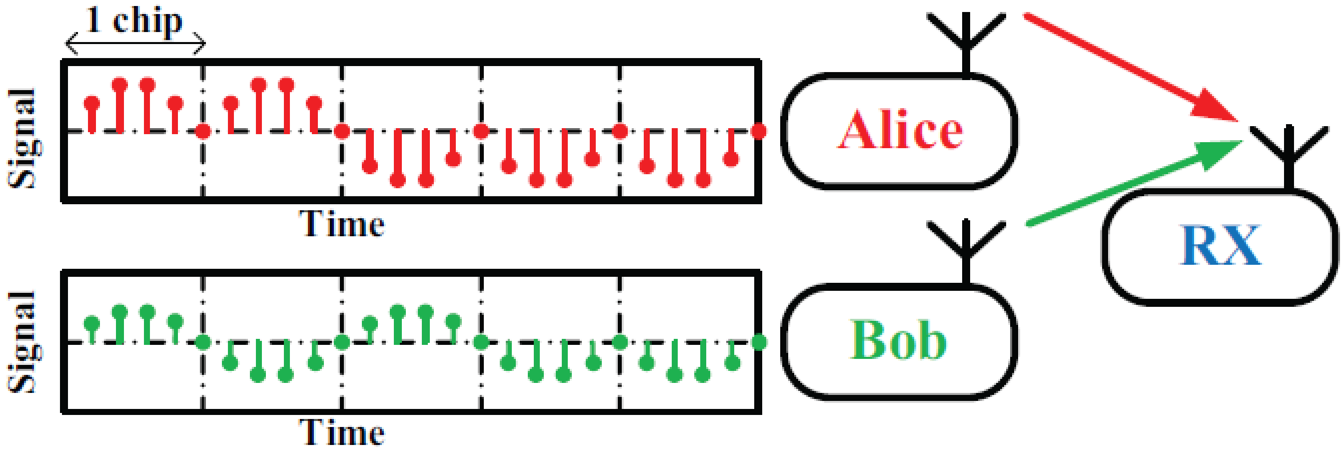






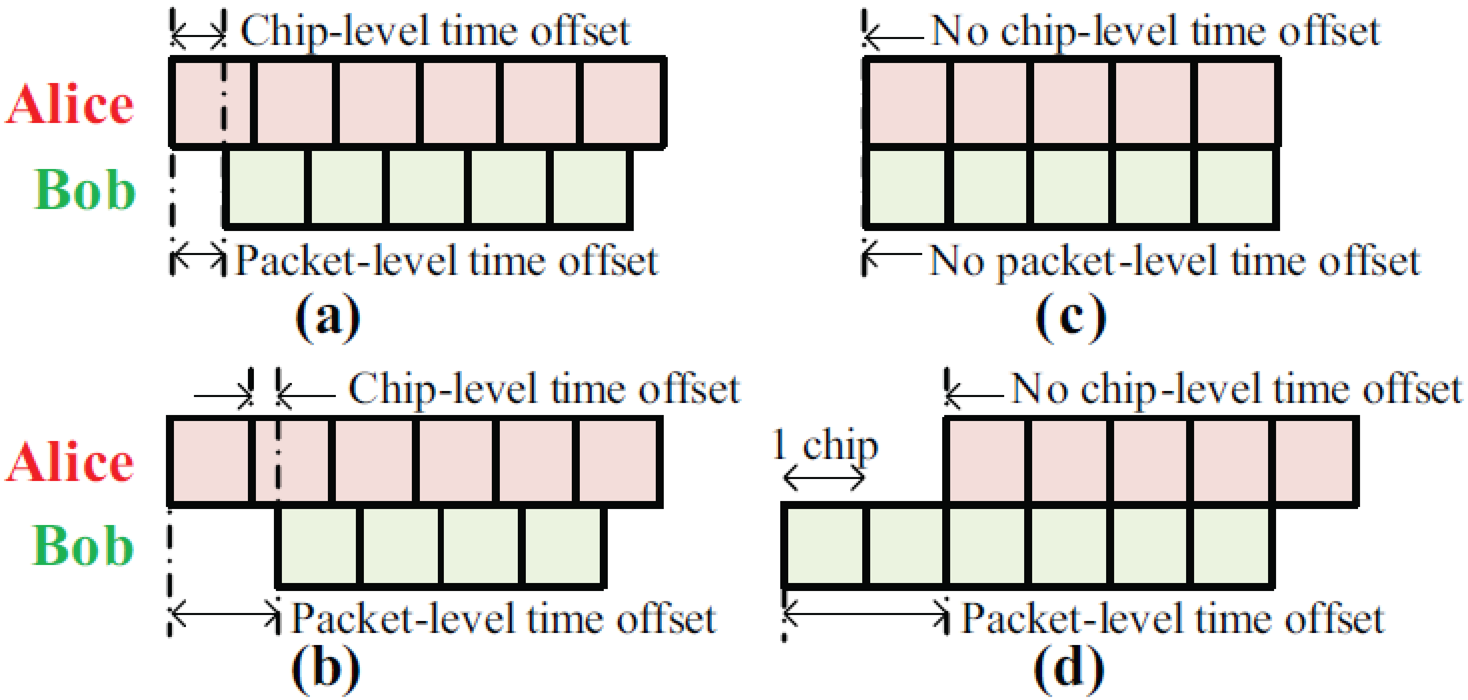
Application

mZig: Enabling Multi-Packet Reception in ZigBee（MobiCom 2015）



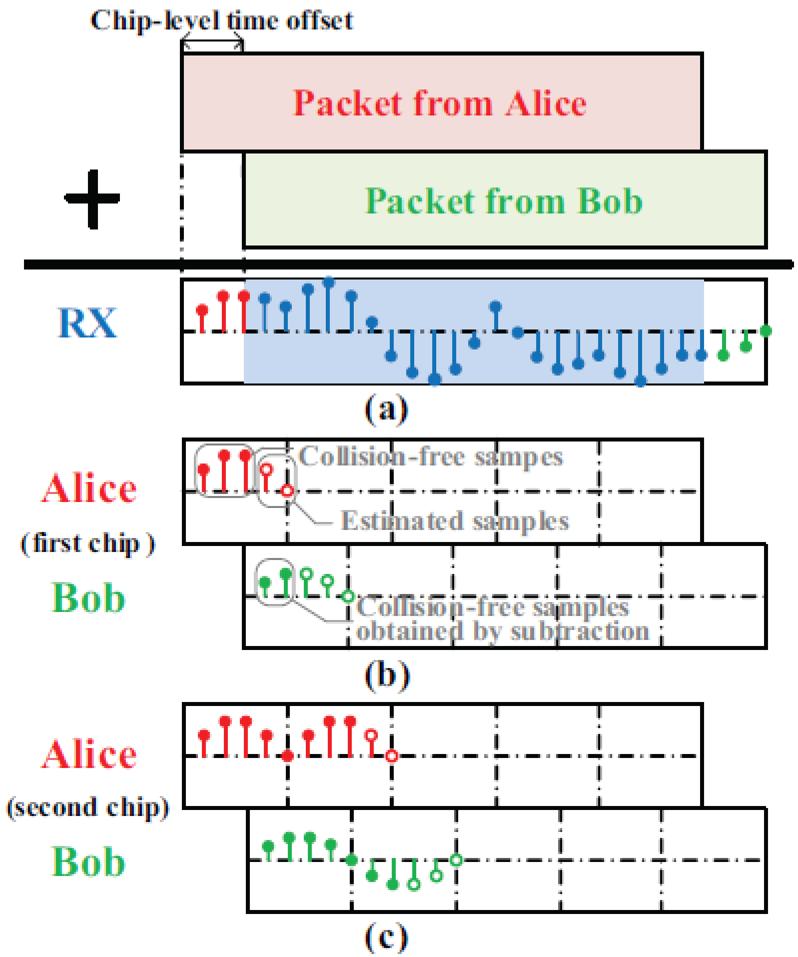
A convergecast scenario.

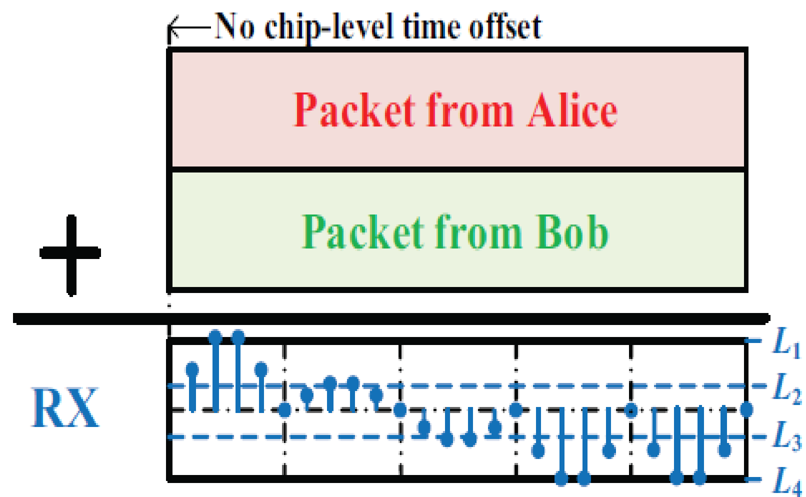
collisions



Chip-level time offset
Alice
Bob
Packet-level time offset
(a)
No chip-level time offset
No packet-level time offset
(c)
Chip-level time offset
Alice
Bob
Packet-level time offset
(b)
No chip-level time offset
1 chip
Packet-level time offset
(d)

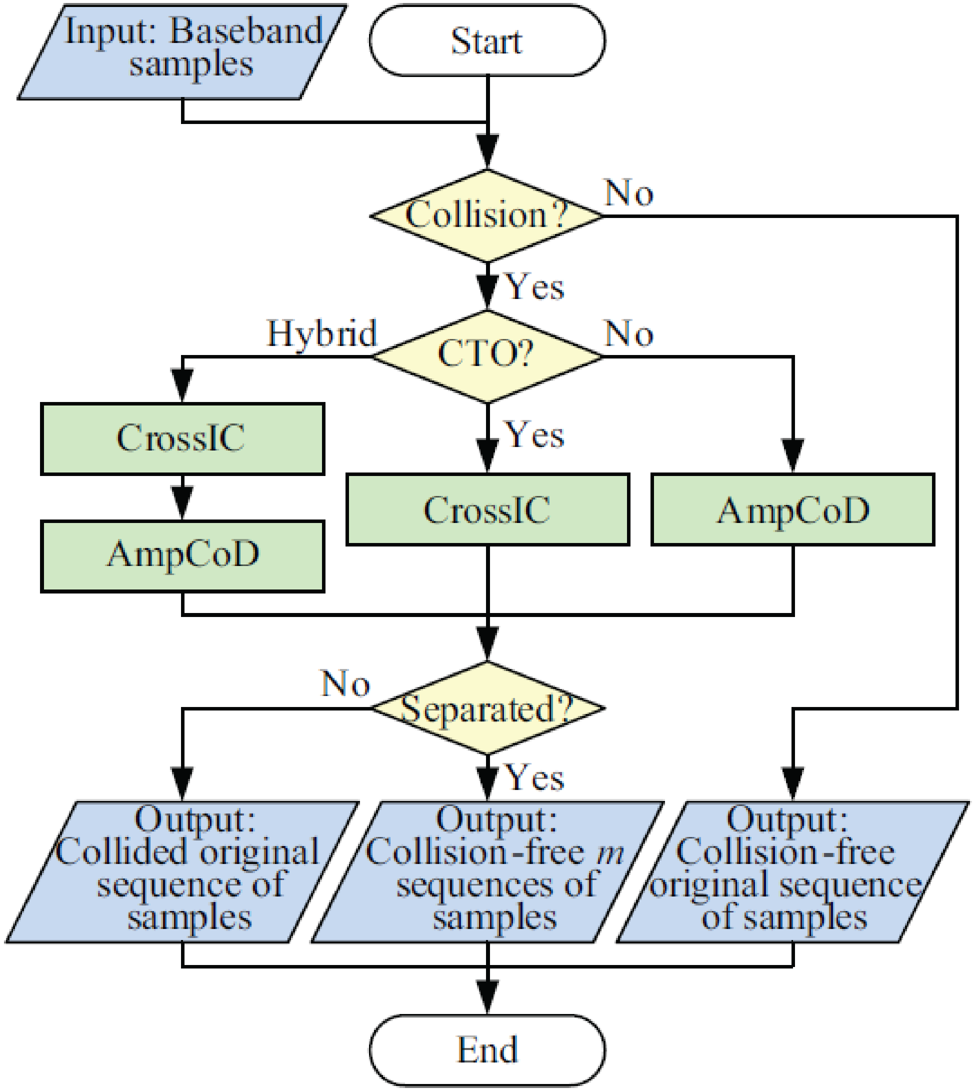
Decomposing





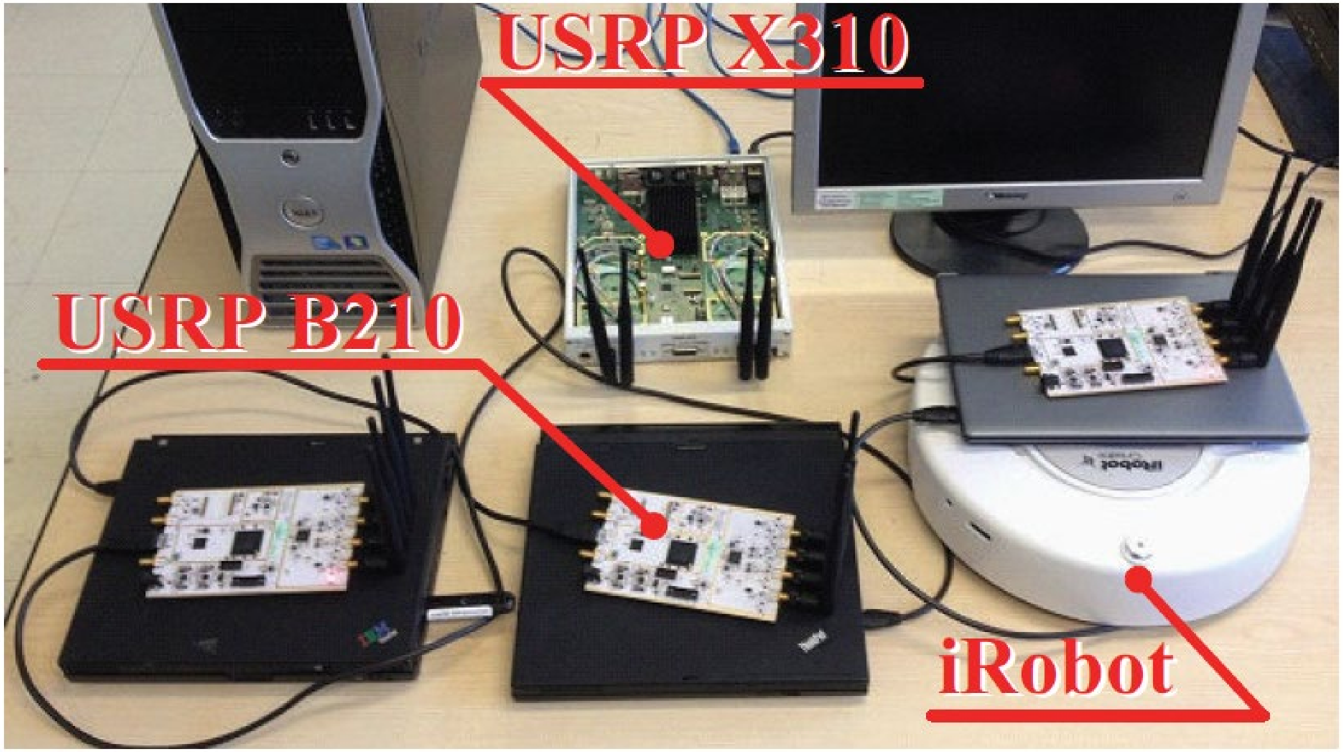
Alice	chip='1'	'1'	'0'	'0'
Bob	chip='1'	'0'	'1'	'0'
Amplitude	$\alpha + \beta$	$\alpha - \beta$	$-\alpha + \beta$	$-\alpha - \beta$

Flow chart


Input: Baseband samples
Start
Collision?
No
Yes
Hybrid
CTO?
No
CrossIC
Yes
CrossIC
AmpCoD
AmpCoD
No
Separated?
Yes
Output: Collided original sequence of samples
Output: Collision-free m sequences of samples
Output: Collision-free original sequence of samples
End

Testbed

RX: USRP X310 + PC
TX: USRP B210*6 + Laptop*6 + iRobots*6



Application

- DVB - T real-time communication system(the University of Pisa, Italy）
http://www.mail-archive.com/discussgnuradio@gnu.org/msg11639.html
- SmartRadio, a cognitive radio program (Virginia Tech University , USA)
http://www.cognitiveradio.wireless.vt.edu/dokuwiki/doku.php?id=home
- Tests of MIMO and multi-hop network (University of Texas, USA)
http://hydra.ece.utexas.edu/testbed/

Question

&

Answer