ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Δίκτυα Καθοριζόμενα από Λογισμικό

Exercise Session 4:
Introduction to GNU Radio

Τμήμα Επιστήμης Υπολογιστών

Introduction into GNU Radio

Manolis Surligas
surligas@csd.uoc.gr

Computer Science Department, University of Crete

November 16, 2014

Outline

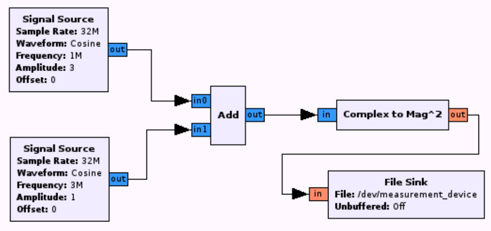
1. GNU Radio
2. The GNU Radio Companion
3. Block Types
4. IO Signatures
5. Creating a GNU Radio block
6. Compile GNU Radio from source

GNU Radio

GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

- GNU Radio is the most widely used SDR software
- Open source, active development community
- The platform provides basic signal and other processing units
- Easy to expand and develop custom modules-plugins
- Critical operations → C/C++
- Components bindings → Python
- Python wrappers for C/C++ functions/methods using SWIG

GNU Radio



Signal Source
Sample Rate: 32M
Waveform: Cosine
Frequency: 1M
Amplitude: 3
Offset: 0

Signal Source
Sample Rate: 32M
Waveform: Cosine
Frequency: 3M
Amplitude: 1
Offset: 0

Add

Complex to Mag^2

File Sink
File: /dev/measurement_device
Unbuffered: Off

- **Block**: processing unit performing a specific task
  - arbitrary number of I/O ports
  - each port: specific data type (float, integer, . . .)
  - One thread per block
- **Flowgraph**: program scenario with the connected blocks

The GNU Radio Companion

- GNU Radio ships together with a set of commonly used Signal Processing blocks
- It provides a GUI IDE called GNU Radio Companion (GRC)
- GRC offers:
  - Drag and drop functionality for inserting blocks
  - Point and click block connections
  - On the fly, data type and size validation for the connected blocks
  - Block parameters can easily configured via a pop-up window
- GRC greatly reduces time and code lines required to build up an SDR application

Block Types

- Blocks are classified depending their input - output ratio
- Synchronous blocks (1:1)
  - Equal number of input and output items are consumed and produced respectively
- Decimation Blocks (N:1)
  - Fixed ratio of input-output items
  - Input items are a fixed multiple of output items
- Interpolation Blocks (1:N) is the exact opposite of decimation blocks
- Block
  - Most generic block type
  - Arbitrary number of input and output items
  - Developers should inherit this block type only if the above are not suitable

IO Signatures

- Each block may have and arbitrary number of input and output ports
- Each port has a size and data type
- Only blocks with ports of equal size and data type can be connected
- Blocks with only output ports are called *sources*
- *Sinks* are blocks with input ports only

IO Signatures

- Each block may have and arbitrary number of input and output ports
- Each port has a size and data type
- Only blocks with ports of equal size and data type can be connected
- Blocks with only output ports are called *sources*
- *Sinks* are blocks with input ports only

Note!

Internally, input and output ports are implemented as buffers, where items are read and stored respectively

IO Signatures

- Input and output ports are declared at the block constructor using the *gr::io_signature* API

Prototypes

```
gr::io_signature::make(min, max, size)
gr::io_signature::make2(min, max, size1, size2)
gr::io_signature::make3(min, max, size1, size2, size3)
```

- *min* specifies the minimum number of ports
- *max* specifies the maximum number of ports
- *size* is the size of the port

Creating a GNU Radio block

- Lets create our first GNU Radio block

```
hello_world_block::hello_world_block(const size_t param1,
                                     const double param2,
                                     int *other_params)
/* Block type and name */
:sync_block("hello_world",
/* Input: exactly one input of complex numbers */
            io_signature::make(1,1,sizeof(gr_complex)),
/* Outputs: exactly 2 output ports of float numbers */
            io_signature::make(2,2,sizeof(float))),
/* Normal C++ constructor initialization */
d_param1(param1),
d_param2(param2 + param1),
d_other_params(other_params)
{
    //Other C++ stuff
}
```

Implement software processing

- After creating and initializing the block, the actual data processing should be implemented
- Depending the block type, the appropriate inherited method should be implemented
  - *work()* for synchronous, decimation and interpolation blocks
  - *general_work()* for basic blocks

Implement software processing

- *work()* and *general_work()* are automatically called by the GNU Radio scheduler when enough input and output items are available
- Developers should perform the appropriate processing with the available input items and produce the corresponding output
- If the method state should be saved for a future invocation, the class (private) members should be used

The work() method

Prototype

```
int
work(int noutput_items,
     gr_vector_const_void_star &input_items,
     gr_vector_void_star &output_items);
```

- *noutput_items* indicates the number of items that are available at the output port(s).

- Due to the fixed relation between the available input and output ports of synchronous (1:1), decimation (N:1) and interpolation (1:N) blocks, the available input items are derived from the *noutput_items* parameter

The work() method

Prototype

```
int
work(int noutput_items,
     gr_vector_const_void_star &input_items,
     gr_vector_void_star &output_items);
```

- *input_items* is a vector of pointers to buffers that contain the data of each input port
- The minimum and maximum size of the vector is specified by the *min*, *max* value of the I/O input signature of the constructor
- The actual size depends on the number of connected input ports

The work() method

Prototype

```
int
work(int noutput_items,
     gr_vector_const_void_star &input_items,
     gr_vector_void_star &output_items);
```

- *output_items* is a vector of pointers to buffers where the output items can be stored
- The minimum and maximum size of the vector is specified by the *min*, *max* value of the I/O output signature of the constructor
- The actual size depends on the number of connected output ports

The work() method

Prototype

```
int
work(int noutput_items,
     gr_vector_const_void_star &input_items,
     gr_vector_void_star &output_items);
```

- As ‘item’ the entity with the specified size at the I/O signature is considered
- The return value of this method, informs the scheduler about the items produced and consequently, consumed during the method invocation

The general_work() method

Prototype

```
int
general_work(int noutput_items, gr_vector_int &ninput_items,
       gr_vector_const_void_star &input_items,
       gr_vector_void_star &output_items);
```

- *general_work()* method should be implemented when a basic block type is used
- The *noutput_items*, *input_items* and *output_items* parameters have the same purpose with those of *work()* method

The general_work() method

Prototype

```
int
general_work(int noutput_items, gr_vector_int &ninput_items,
         gr_vector_const_void_star &input_items,
         gr_vector_void_star &output_items);
```

- *ninput_items* is a vector of integers, with each element to contain the number of the available input items of the corresponding input port
- Produced items are reported to the scheduler with the return value
- Consumed input items are reported using the *consume()* or *consume_each()* methods

The general_work() method

Prototype

```
int
general_work(int noutput_items, gr_vector_int &ninput_items,
         gr_vector_const_void_star &input_items,
         gr_vector_void_star &output_items);

/**
 * @how_many_items items consumed from input port
 * @which_input
 */
void
consume(int which_input, int how_many_items);

/**
 * @how_many_items items consumed from all the input
 * ports of the current block
 */
void
consume(int how_many_items);
```

Retrieve input and output buffers

- Input and output buffers of the corresponding input and output ports are retrieved from *input_items* and *noutput_items* respectively
- Both *input_items* and *noutput_items* have *void ** data type
- They can be cast to the data type used at the constructor of the block
- For our example:

```
int
work(int noutput_items,
     gr_vector_const_void_star &input_items,
     gr_vector_void_star &output_items)
{
const gr_complex *in = (const gr_complex *)input_items[0];
float *out0 = (float *)output_items[0];
float *out1 = (float *)output_items[1];

/* Rest of the code */
}
```

Compile GNU Radio from source

- Before trying to compile the GNU Radio several packets should be installed

- cmake
- cppuint
- boost and dev files
- fftw3 and dev files
- python
- swig
- numpy
- Doxygen
- Cheetah and python-Cheetah

- gsl and dev files
- qt4
- qwt and dev files
- pyqt
- pyqwt
- wxpython
- python-lxml
- WxWidgets
- audio-oss and dev files
- portaudio and dev files

Compile GNU Radio from source

- Grab the source code from `https://www.gnuradio.org`
- For the purpose of this course, a custom GNU Radio version will be provided with a set of examples
- Before compiling, the appropriate Makefiles should be produced using the *CMake* tool

```
cd gnuradio_dir
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Debug ..
```

Compile GNU Radio from source

```
  Gnuradio enabled components
##############################
   * python-support
   * testing-support
   * volk
   * doxygen
   * sphinx
   * gnuradio-runtime
   * gr-ctrlport
   * gr-blocks
   * gnuradio-companion
   * gr-fec
   * gr-fft
   * gr-filter
   * gr-analog
   * gr-digital
   * gr-dtv
   * gr-atsc
   * gr-channels
   * gr-noaa
   * gr-pager
   * gr-qtgui
   * gr-trellis
   * gr-utils
   * gr-video-sdl
   * gr-vocoder
   * gr-fcd
   * gr-wavelet
   * gr-wxgui
```

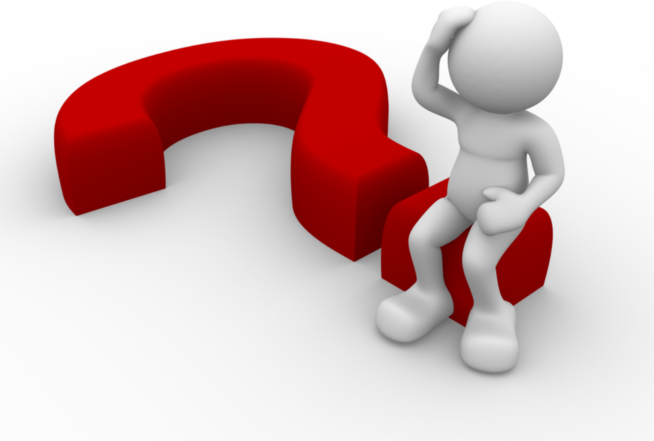
- After cmake, in order to get a fully functional GNU Radio instance, components of the left column should be enabled
- To compile and install GNU Radio execute inside the *build/* directory:

```
make
make install
```

Resources

- Check the sample code of the gr-hy436 GNU Radio module
- `http://www.gnuradio.org`
- `http://gnuradio.org/doc/doxygen/index.html`

Questions



Τέλος Ενότητας

cc BY NC ND

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ
2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

- Ως **Μη Εμπορική** ορίζεται η χρήση:

–που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο

–που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο

–που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

.

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Ξενοφώντας Δημητρόπουλος. «**Δίκτυα Καθοριζόμενα από Λογισμικό. Exercise Session 4: Intro to GNU Radio**». Έκδοση: 1.0. Ηράκλειο/Ρέθυμνο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση: http://www.csd.uoc.gr/~hy436/