Wireless Communication Systems
@CS.NCTU

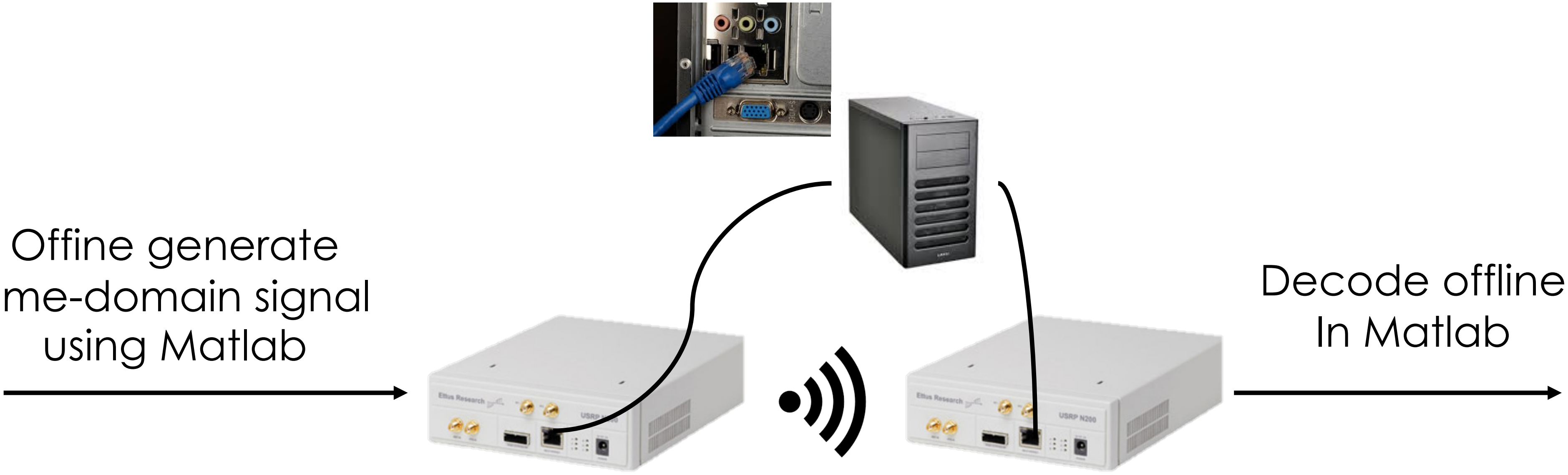
USRP Lab 2

Yu-Lin Wei

2016.10.20

Outline

- Intro
  - Environment
- Tasks
  - Generate time-domain signal in UHD
- Grading Criteria



Offine generate
time-domain signal
using Matlab

Decode offline
In Matlab

Run single_tx
at USRP_TX

Run single_rx
at USRP_RX

- USRP Testbed in LAB / office
- Access through ssh (test your single_tx_f / single_rx)
- Run Matlab in your own machine

How to Compile

- File (Source) Directory
  - Use built in Makefile
  - Put your files in ~/uhd/host/examples/
  - Add your files to the Cmakelist.txt in ~/uhd/host/examples

- Compile (Binary) Directory
  - cd ~/uhd/host/build/examples
  - make

USRP Server

- ssh wcs-g#@140.113.203.6
  ssh wcs-g#@140.113.207.100
  - e.g., wcs-g1@140.113.203.6 default password:
- HW code put in ~/uhd/host/example
  - single_tx_f.cpp/ single_tx_f.h
  - single_rx.cpp/ single_rx.h
- cd ~/uhd/host/build
- cmake .. (only the first time)
- make
- cd example (now in ~/uhd/host/build/example)

USRP Server

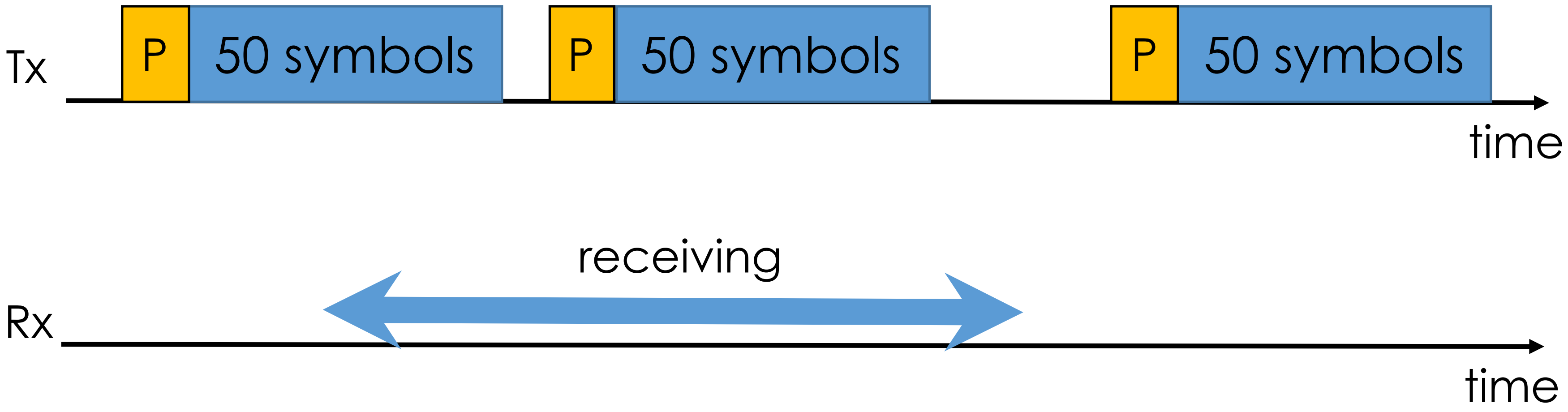
- mkdir wcs_trace
- Transimitter :
  - ./single_tx_f --f=2.49 --i=128
- Receiver:
  - ./single_rx --f=2.49 --i=128
- Received data in ./wcs_trace/recv_signal.bin

Outline

- Intro
  - Environment
- Tasks
  - Generate time-domain signal in UHD
- Grading Criteria

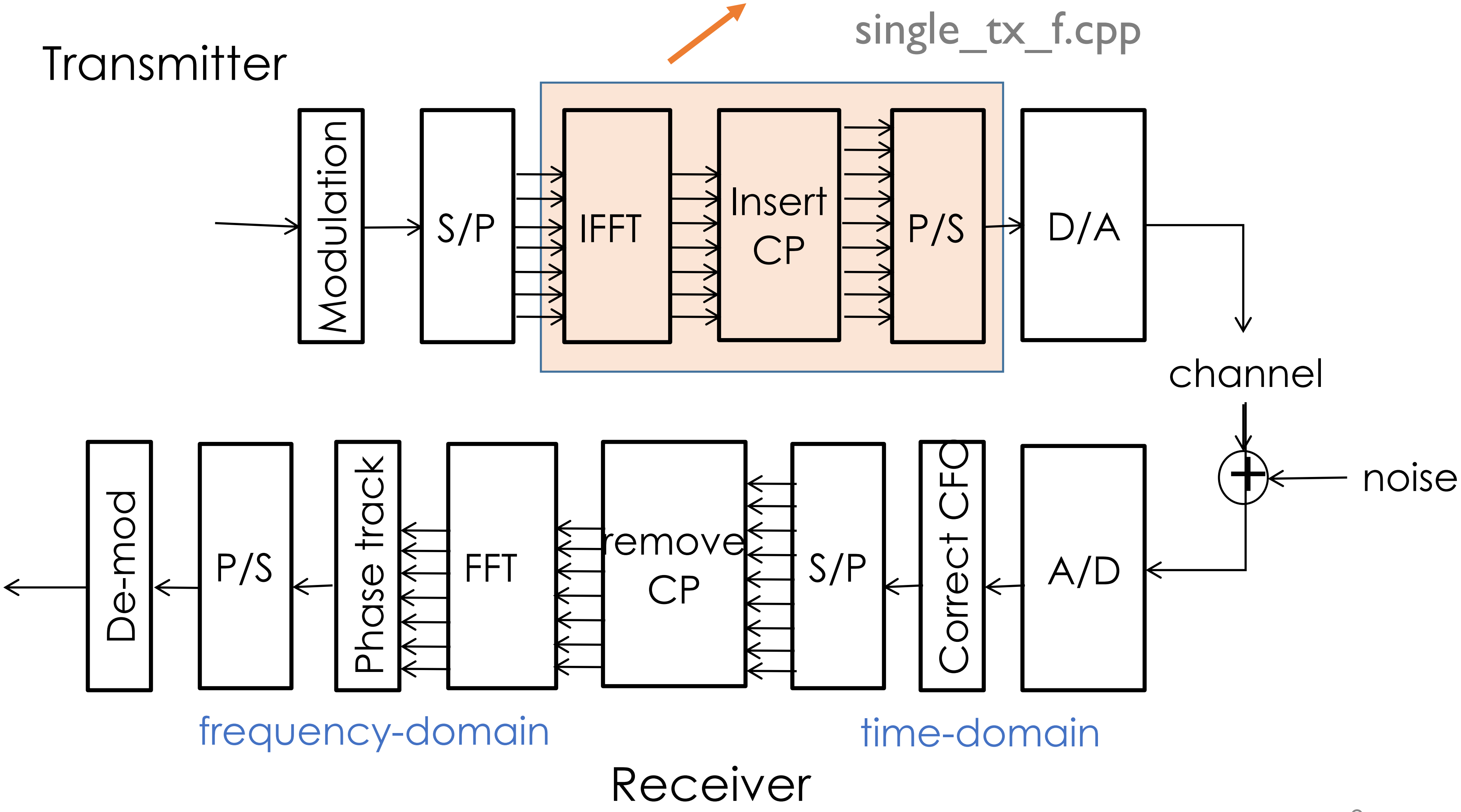
Task

- Tx repetitively sends 50 symbols
  - **Generate time-domain samples in UHD**
- Rx receives at least one batch of 50 symbols
- Matlab offline decoding



Task

Implement this block in single_tx_f.cpp



Preliminary – FFT library Installation

- Library : FFTW3
  - http://www.fftw.org/
  - Document
    - http://www.fftw.org/fftw3_doc/Complex-One_002dDimensional-DFTs.html
- Installation already done by TA
- TODO: Add linking library to Cmake file (important)
  - Edit /uhd/host/example/CMakeLists.txt
  - Line 54: add fftw3 and m to link library
    - TARGET_LINK_LIBRARIES(${example_name} uhd fftw3 m ${Boost_LIBRARIES})
  - Clean all things in build directory, and cmake again according to lab1 slide (page 12)

Task 1: Freq. Symbol Generator

- TODO (signal_f_gen.m)
  - Cut (re-write) the part for generating frequency-domain samples (including pilot and unused subcarriers)
  - Save as signal_f_gen.m
  - signal_f_gen.m outputs the signal to be transmitted as src_time_preamble.bin, src_time_1.bin, src_freq_1.bin and src_data_1.mat
  - src_time_preamble.bin – will be fed into single_tx_f.c for USRP transmission, so you don't need to do IFFT for the preamble
  - src_freq_1.bin – will be fed into single_tx_f.c for USRP transmission
  - src_time_1.bin – just for debugging
  - src_data_1.mat – ground truth for decoding / plotting

Task 2: Freq. to Time Conversion

- Read frequency-domain symbols and convert them to time-domain
- Both ways work, but we implement the right one

```
for (i=1; i<NUM_SYM; i++) {
        // read freq. symbols from file
        // do IFFT here
}

While(1) {
    for (i=1; i<NUM_SYM; i++) {
        // send time samples
    }
}
```

✔

```
for (i=1; i<NUM_SYM; i++) {
        // read freq. symbols from file
}

While(1) {
    for (i=1; i<NUM_SYM; i++) {
        // do IFFT here
        // send time samples
    }
}
```

Task 2: Freq. to Time Conversion

- Example code for IFFT: fft_sample.cpp
  - Sample code of FFT using fftw3
  - Performing fft in a complex array of size 100
  - (0+0i, 1+1i, 2+2i…)
- Compile
  - g++ fft_sample.cpp –o fft_sample -lfftw3 –lm
- Implement IFFT in your single_tx_f.cpp based on the example code in fft_sample.cpp

- NOTE: please compare the fft result w/ Matlab to check the difference
  - especially careful about fft_shift in Matlab

Task 3: Add Cyclic Prefix (CP)

- For each symbol, copy the last 16 time-domain sample, and insert them in the beginning of the symbol

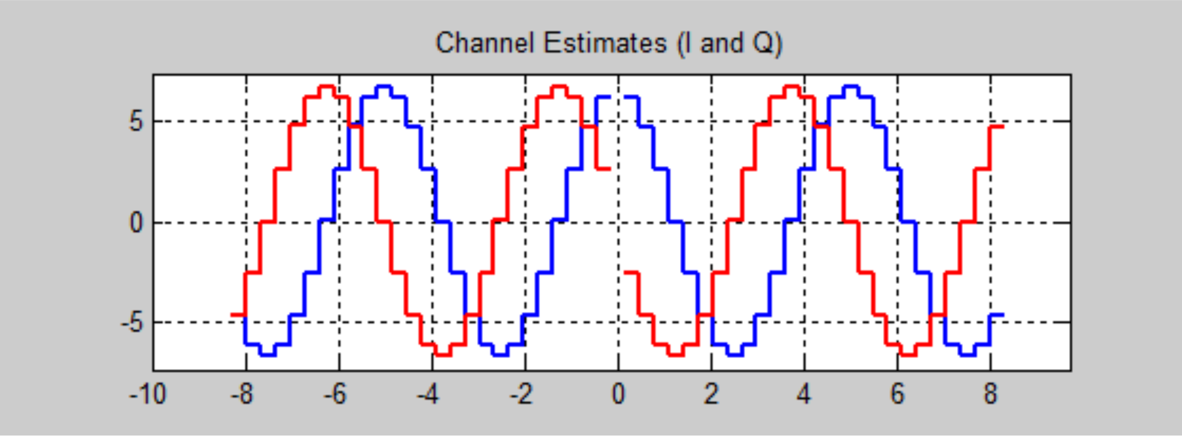
Task 3: Gain Control

- You don’t know what is the maximal amplitude of the time-domain sample as doing real-time F-to-T conversion
- Measure the maximal amplitude in the first batch of 50 symbols, and use it to normalize the time-sample of all the remaining batches

```
scale = 1;
while(1) {
    if (batch > 1)  { scale = max_ampl;  }
    for (i=1; i<NUM_SYM; i++) {
        // do IFFT here
        if (batch == 1) {
            // update max_ampl
        }
        // send scaled time samples: sample/scale
    }
}
```

Required figures

- Figure 1: Channel Estimation H[k] (WARP figure 4-1)



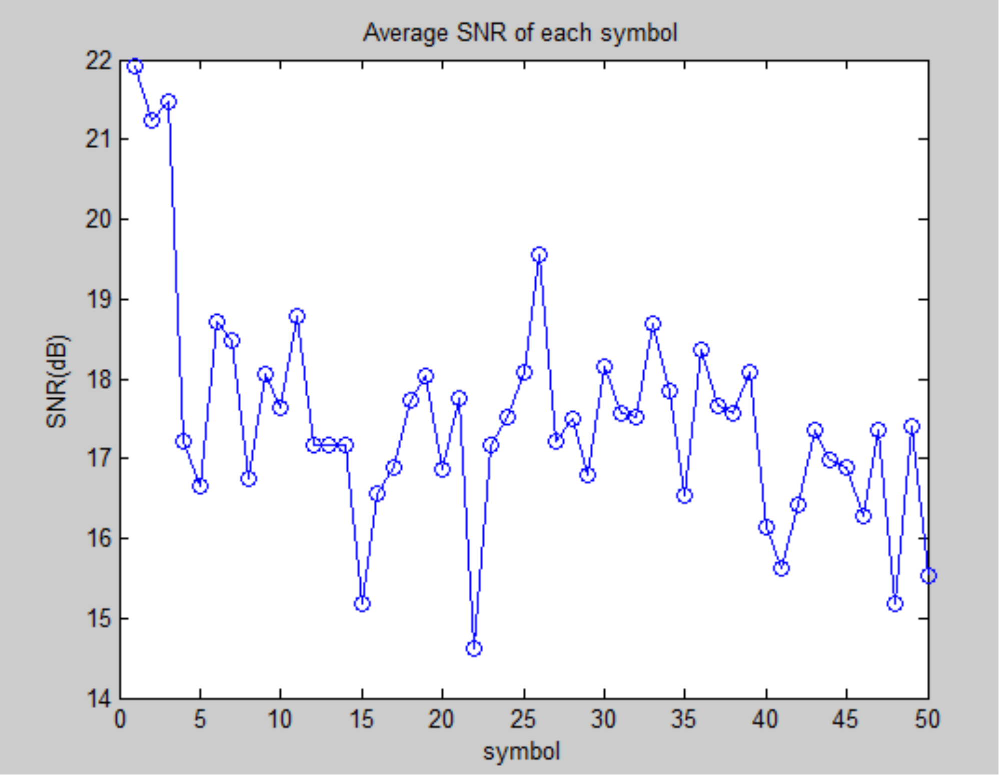
Required figures

- Figure 2: subcarrier SNR
  - average SNR of each data subcarrier among all symbols (bar graph)
  - With and without phase track

- Observation
  - Check if there exists deep fading

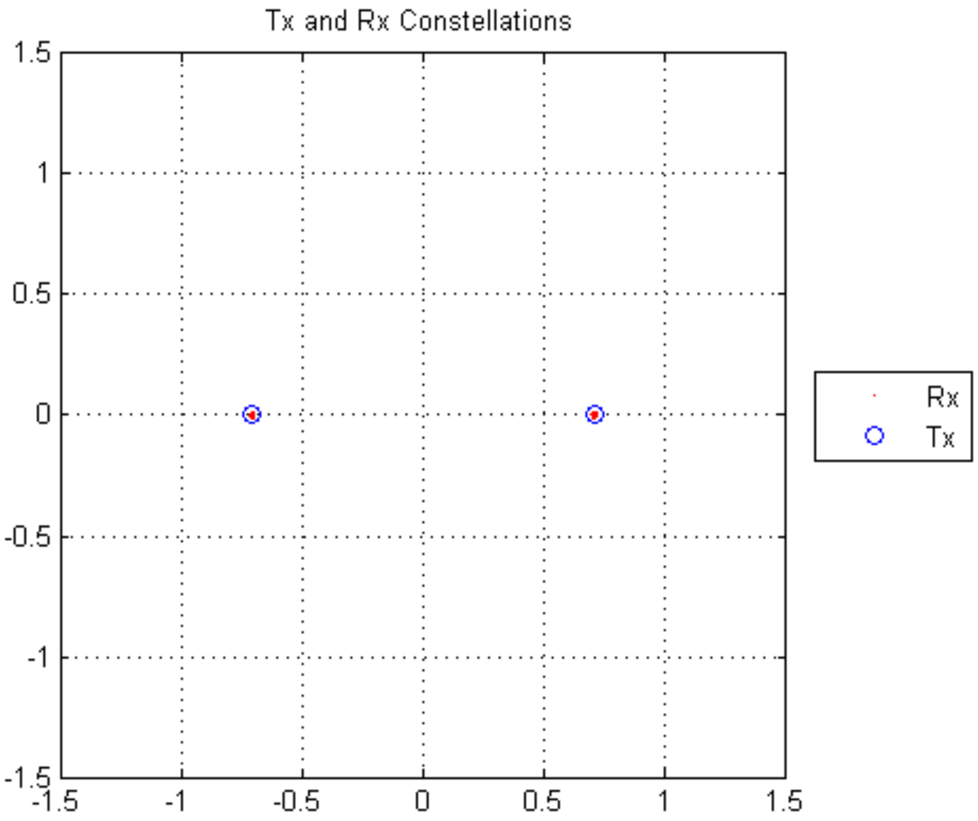
Required figures

- Figure 3: symbol SNR
  - average SNR of all subcarriers for symbols over time (line graph or scatter plot)
  - With and without phase track
- Observation
  - Check if SNR drops over time if phase track is disabled



Required figures

- Figure 4: constellation points (WARP figure 6)



Demo

- Time/Location
  - Nov. 4(Fri.) 10:00~12:00 in EC-538
  - Or by appointment
  - Contact with TX (張威竣) to sign up the time slot
- Flow
  - Run signal_f_gen.m
  - Get src_freq_preamble.bin, src_freq_1.bin, src_time_1.bin, src_data_1.mat
  - Upload src_freq_preamble.bin and src_data_1.bin to wcs-g[#] account
  - Run ./single_tx_f and ./single_rx under ~/uhd/host/build/example
  - Download example/wcs_trace/recv_signal.bin
  - Put recv_signal.bin in program/trace
  - Run decode.m to get the figures

Grading

- signal_f_gen.m: 10%
- signal_tx_f.cpp : 60%
- decode.m: 10%
- Code readability: 10%

90%

- Peer review: ±15%

Peer Review

- 15% group member peer review
  - Anonymous
  - Range from -15 ~ 15
  - Grade for each peer, excluding yourself
  - Zero mean

	Alice	Bob	Chris	David
Alice	N/A	-10	-5	+15

- Total score: up to 105

Code Submission

- Deadline: Nov. 4 (Fri.) 23:59
- Email to
  - msn.nctu@gmail.com
  - Email subject: [WCS] lab2_gX
  - WCS_lab2_gX.zip
    - source code (single_tx_f.cpp/ single_rx.cpp/ decode.m/ signal_f_gen.m)

Q&A