SR2020-34　　2020年11月19日 スマート無線研究会

※本資料は，下記での報告資料であり，検討内容に関する報告時点でのスナップショットに相当します．

中浜智也, 山田洋士, 亀田 卓, “GNU Radioを用いたUSRPおよびドータボードでの位相同期の実装例," 電子情報通信学会技術研究報告 スマート無線 SR2020-34, vol.120, no.238, pp.74-81, 2020年11月19日.

GNU Radioを用いたUSRP およびドーターボードでの位相同期の実装例

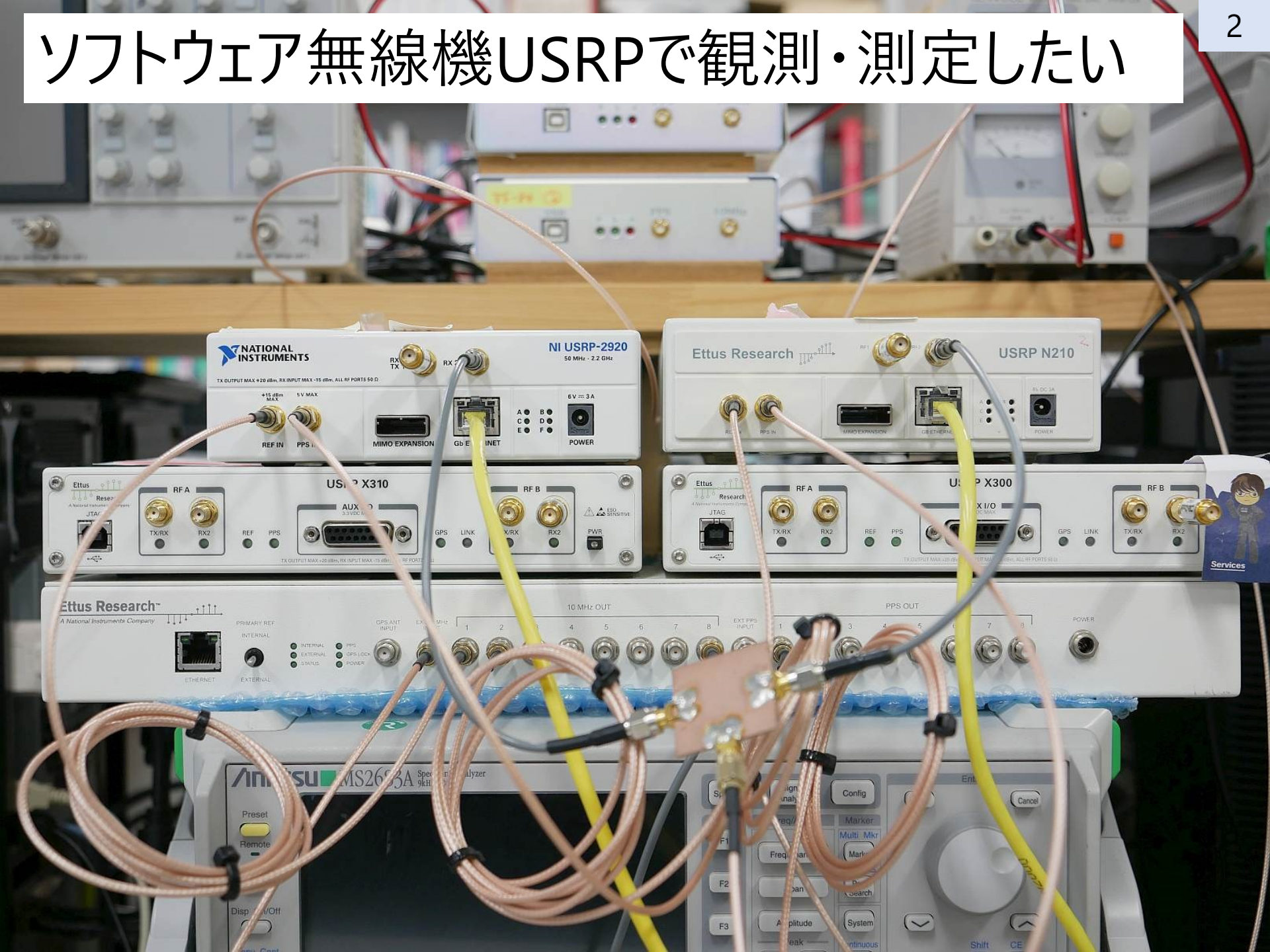
○中浜 智也
(石川高専 専攻科)

山田 洋士
(石川高専 電子情報工学科)

亀田　卓
(東北大学 電気通信研究所)

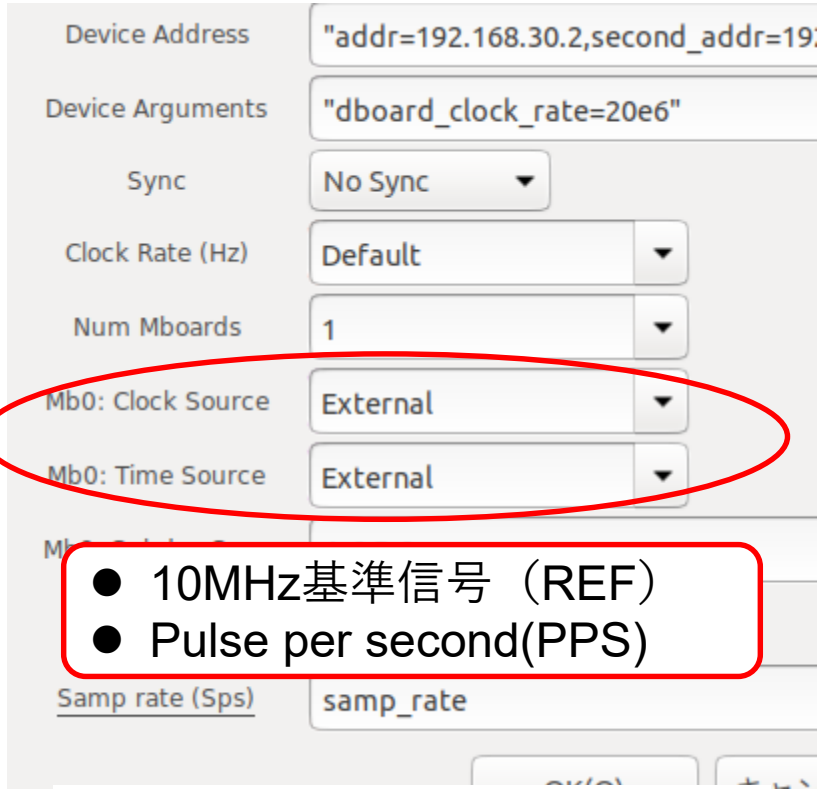
予稿の正誤表
p.78 左段 下から7行目（5.2節）
（正）位相差の変動が約0.8度以内
（誤）位相差が約0.8度以内

ソフトウェア無線機USRPで観測・測定したい



USRP / GNU Radioと同期 (はじめに)

GPS disciplined oscillator(GPSDO)からの基準信号の供給例



- 10MHz基準信号（REF）
- Pulse per second(PPS)

GNU Radio Companion(GRC)での指定例

- USRPで多チャネル送受信を行い，観測・測定を行う際には，位相・タイミング同期が実験の成否を分ける場合が多い
- USRPの能力を生かした使い方をするサンプル実装を紹介

本報告の内容

- USRPおよびドータボード間でのタイミング同期・LO位相同期を確立するPython付加コードの実装例を紹介
  - UHD timed commands使用によりタイミング同期を実現
- 3種類の典型的な利用例に対応
  - 1台のUSRPで送信・受信を同時に実行
  - 2台のUSRPで位相の揃った同時受信
  - 2台のUSRPで送信・受信を同時に実行
- USRP N200およびX300に対応
  - BasicTX / RXまたはSBX-40/120ドータボードを対象(UBXも)

- 同期が必要となる無線信号の観測・測定を初めて行いたい技術者を対象に，シンプルで有用なサンプルコードを提供
- GNU Radio Companion(GRC) で生成されたPythonコードに本付加コードを追記して使用

本報告での同期の定義・種類

試行ごとの復調信号を重ね書きした際に，ぴったりと重なるように

- 以下の条件を仮定
  - 全てのチャネルに同一の基準信号が供給
- 本稿での同期の定義
  - 送信・受信(Full duplex)の際は，試行間での初期位相が一定値となること.
  - 2チャネル受信の際は，チャネル間の位相差が試行ごとに変動しないこと.
- 同期の種類
  - LO位相同期（LO:local oscillator）
  - タイミング同期

LO位相同期（キャリア位相同期）

LO:local oscillator

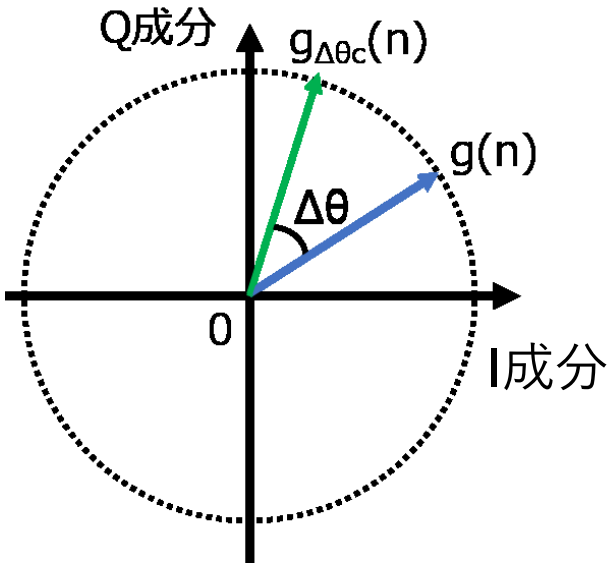
- LO位相ずれはLOの位相が試行ごとに異なることで発生
- LOを有するドーターボード利用時に発生(SBXやUBX)
- $\Delta\theta_c$が試行ごとに変動しないようにする

I成分とQ成分からなる
ある信号

$$g(n) = A(nT_s)e^{j\varphi(nT_s)}$$

LOの位相が$\Delta\theta_c$
変動した場合

$$g_{\Delta\theta_c}(n) = A(nT_s)e^{j\varphi(nT_s)}e^{j\Delta\theta_c}$$



コンスタレーションでのイメージ

タイミング同期

- サンプリングを開始するタイミングが試行ごとに異なることでタイミングずれが発生
- USRPのFPGA上で周波数変換が実施される場合にタイミングずれが発生すると，LO位相ずれと類似した位相ずれ（キャリア位相ずれ）も生じる.

I成分とQ成分からなるある信号

$$g(n) = A(nT_s)e^{j\varphi(nT_s)}$$

サンプリング開始タイミングが
Δt変動した場合

$$g_{\Delta t}(n) = A(nT_s + \Delta t)e^{j\varphi(nT_s+\Delta t)}e^{j2\pi F_p \Delta t}$$

dsp_freq in Hz

USRPでの復調の機構

– two stage tuning processと呼ばれている –

一例

927MHz

925MHz
rf_freq

2MHz
dsp_freq

$$F_c = F_{LO} + F_p$$

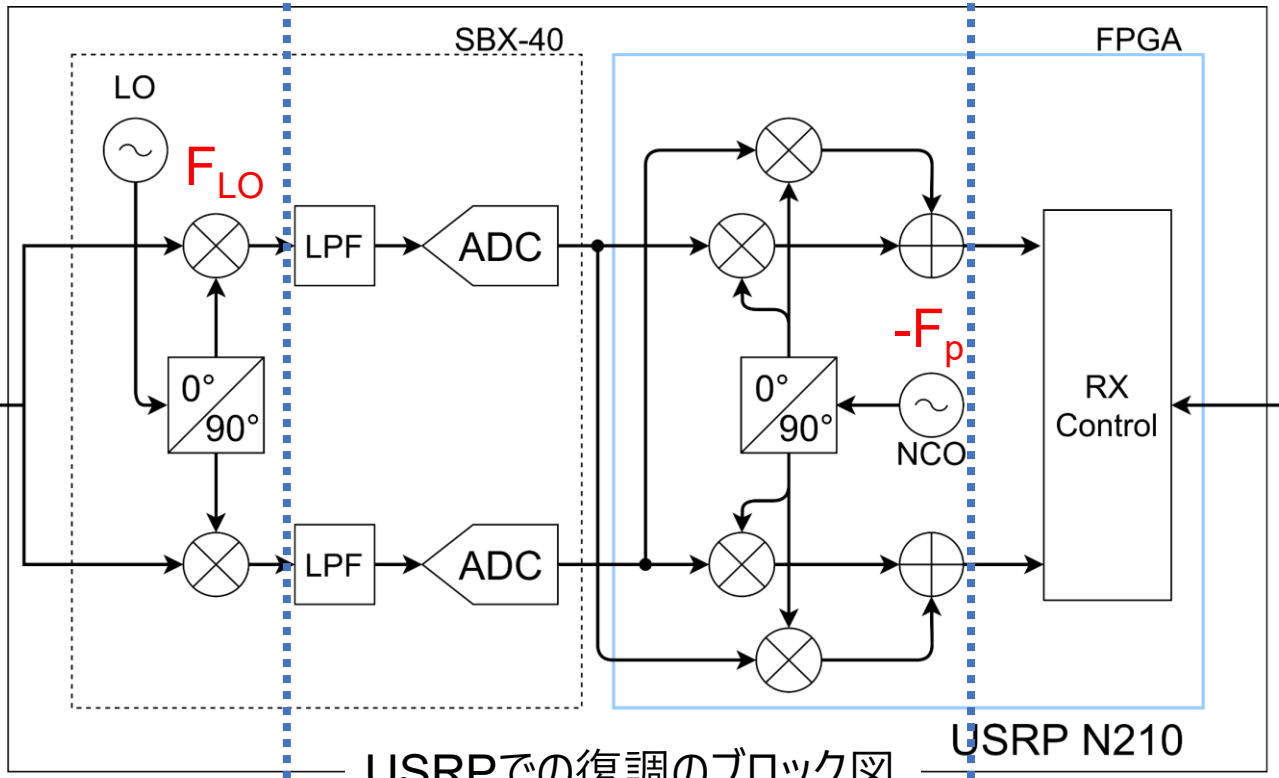
ドーターボードの
直交復調器

FPGA上での
周波数変換

RF tuning

DSP tuning

Baseband

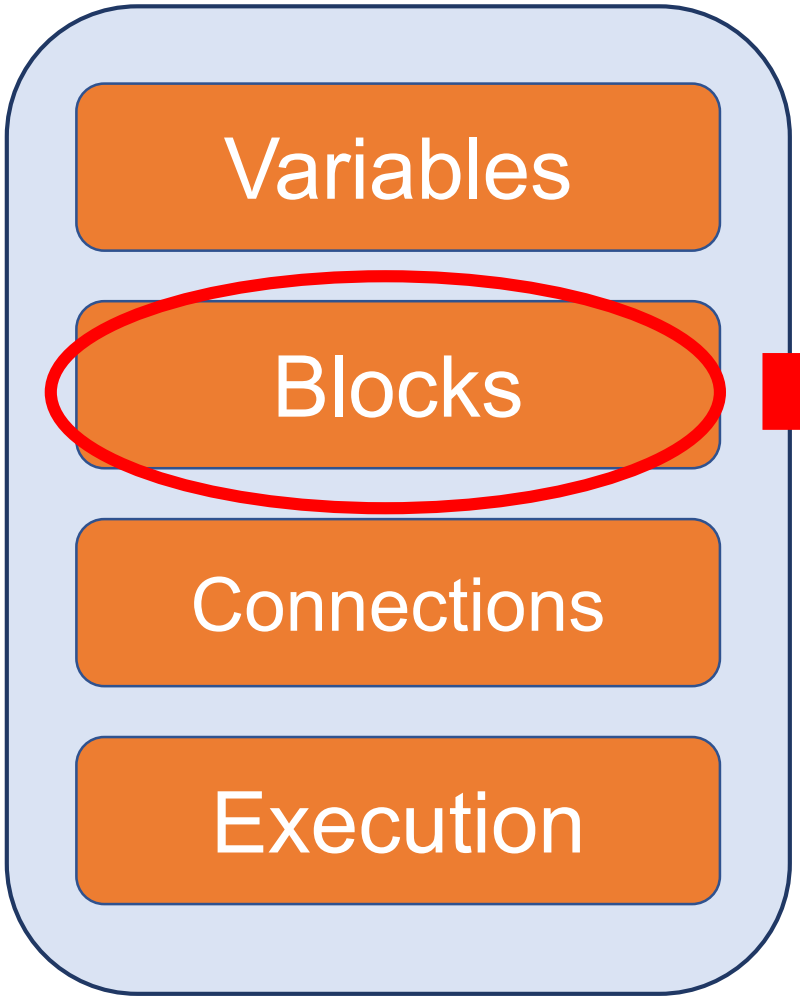


USRPでの復調のブロック図

numerically-controlled oscillator (NCO)

□ キャリア位相ずれは2か所で発生し得る

GRCが生成するPythonコード



ソースコードの基本的な構成

```
####################################
# Blocks
####################################
self.uhd_usrp_source_0 = uhd.usrp_source(
        ",".join(("addr=192.168.30.2,second_addr=192.168.40.2",
"")),
        uhd.stream_args(
                cpu_format="fc32",
                channels=range(1),
        ),
)
self.uhd_usrp_source_0.set_subdev_spec('B:0', 0)
self.uhd_usrp_source_0.set_samp_rate(samp_rate)
self.uhd_usrp_source_0.set_center_freq(freq, 0)
self.uhd_usrp_source_0.set_gain(0, 0)
self.uhd_usrp_source_0.set_antenna('RX2', 0)
```

```
####################################
# Connections
####################################
```

ここにコードの追加を行う

本報告で用いるLO位相同期確立手法

- LOの位相はset_center_freq()が実行されるタイミングで決まる(実験的に理解)
- 以下のコードを記述することで，set_center_freq()が実行される時刻を指定し，
- 各チャネルのset_center_freq()を同時に実行させることでLOの位相同期が確立する.

set_command_time(ある時刻)

set_center_freq(キャリア周波数)

clear_command_time()

- 普通に書いたら，上の行から順に実行される.
- 明示的に同時に実行するように記述する.

本稿で用いるタイミング同期確立手法

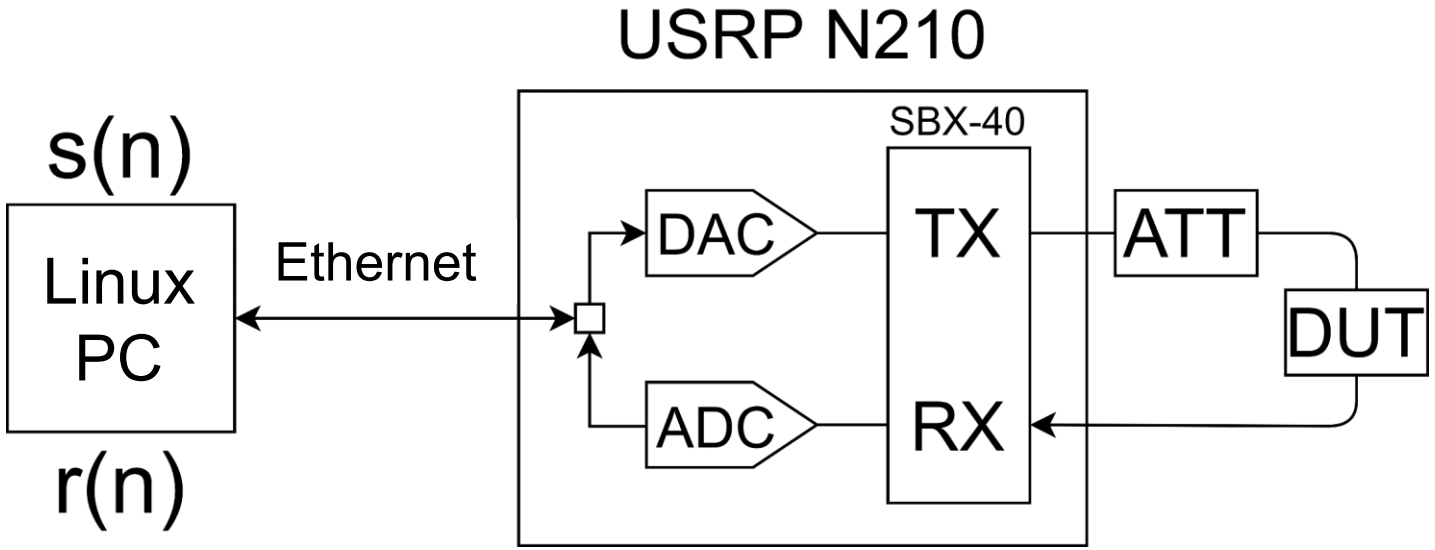
- 以下のコードで，サンプリングの開始時刻を指定することができる.
- 複数チャネルのサンプリング開始タイミングを固定化することでタイミング同期を確立させる.

set_start_time(ある時刻)

ベースバンド信号s(n)・復調信号r(n)

- s(n)とr(n)の時刻は固定化するが，処理遅延は存在
- 復調信号が定常状態になるまでに，時間を要することは変わらない(数10～数100ms)

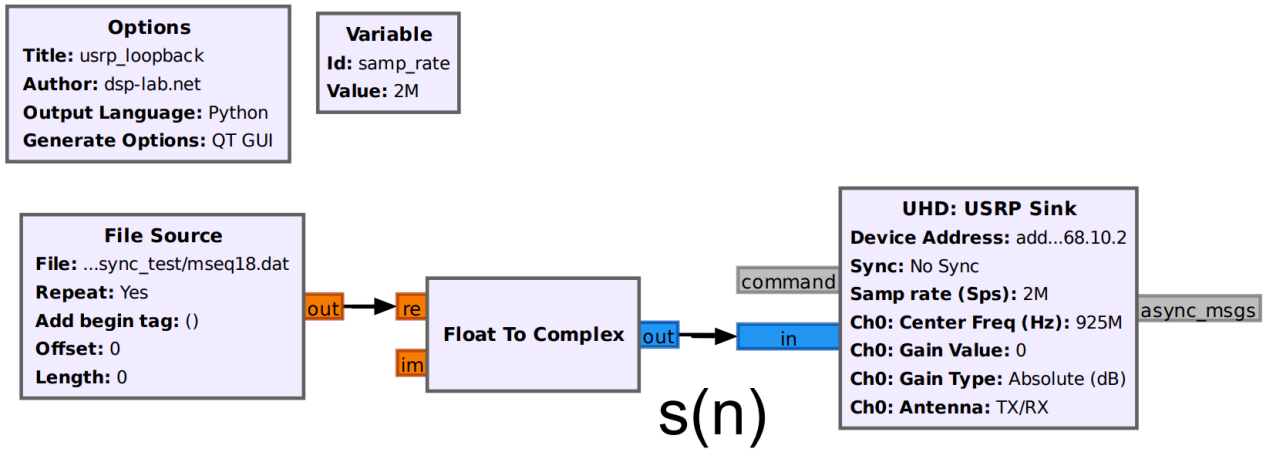
USRP1台で送受信を行う構成(予稿5.1節)



- s(n)で変調した信号を減衰器(ATT)を介して測定対象(DUT)に加え，応答信号を復調した信号r(n)を得る想定（本稿の実行例では，DUTなしで直結）
- s(n)に対して，r(n)が時間的に同期していてほしい
- DUTの有無によるr(n)の違いが測定上必要になる場合があることから，試行ごとにr(n)の時間軸が変動しないことが望まれる

- DUTのインパルス応答測定[6]など，DUTに関わる測定を行う場合に，この構成を基本として発展させられる

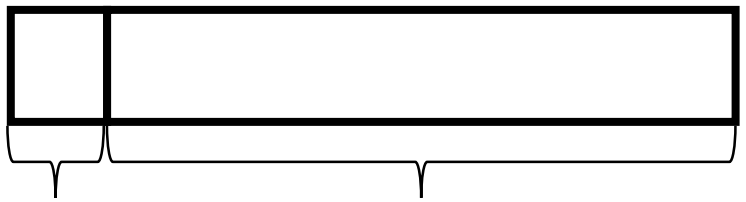
GRCのフローグラフ(USRP1台で送受信)



Ubuntu 18.04.5
GNU Radio 3.8.2.0
UHD 3.15.0.0

検証条件

中心周波数Fc	925MHz
ベースバンド信号のサンプリング周波数Fsb	2MHz
変調方式	DSB-SC

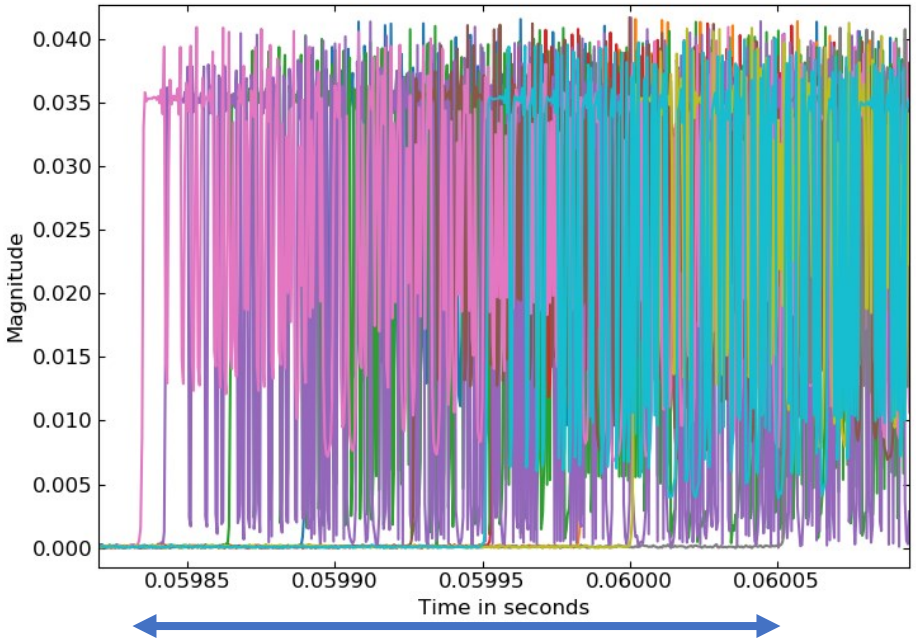


0値データ
1×10^5 点

18次のM系列

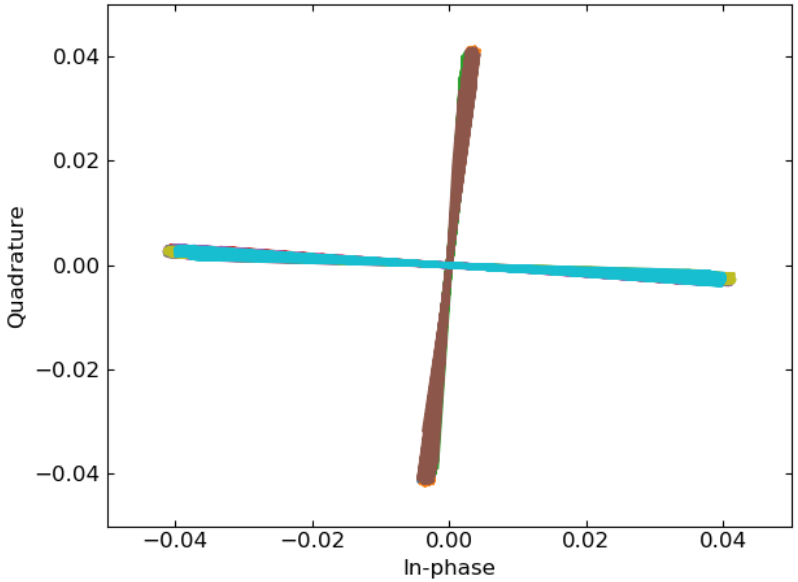
検証に使用するs(n)

GRC上でそのまま実行した場合



約220μs(試行ごとに異なる)

10回の試行の復調信号r(n)の絶対値



10回の試行の復調信号r(n)の
コンスタレーション

- 試行ごとのサンプリング開始時刻には数100μsの時間差が存在(送信側+受信側)
- 試行ごとに位相が変動する(全部で4パターン)

同期確立のための付加コード(1台で送受信)

LO位相同期コード

0.01秒は文献[11]記載のAPI呼び出し例に従った

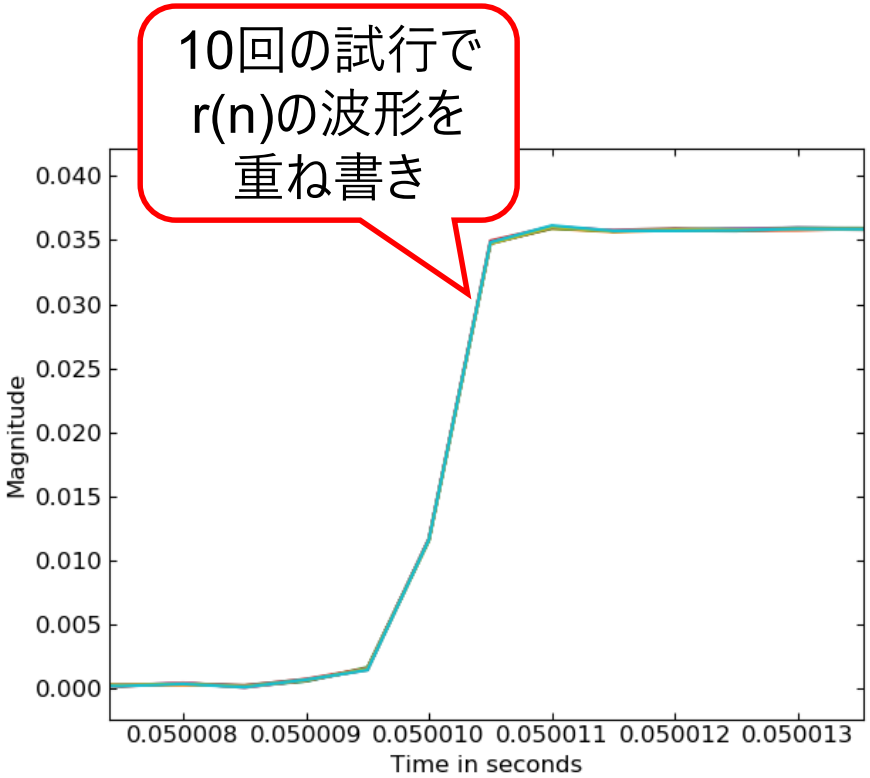
```
time_now = self.uhd_usrp_source_0.get_time_now() + uhd.time_spec(0.01)
self.uhd_usrp_source_0.set_command_time(time_now)
self.uhd_usrp_source_0.set_center_freq(925e6)
self.uhd_usrp_source_0.clear_command_time()
self.uhd_usrp_sink_0.set_command_time(time_now)
self.uhd_usrp_sink_0.set_center_freq(925e6)
self.uhd_usrp_sink_0.clear_command_time()
while self.uhd_usrp_source_0.get_sensor("lo_locked").to_bool() != True:
    pass
while self.uhd_usrp_sink_0.get_sensor("lo_locked").to_bool() != True:
    pass
```

```
time_now = self.uhd_usrp_source_0.get_time_now() + uhd.time_spec(0.1)
self.uhd_usrp_source_0.set_start_time(time_now)
self.uhd_usrp_sink_0.set_start_time(time_now)
```

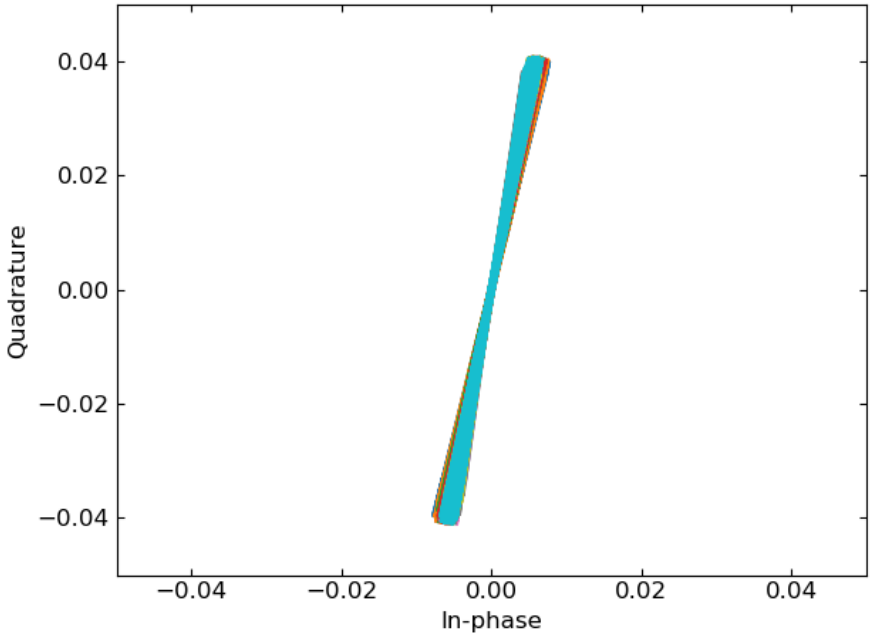
タイミング同期コード

0.1秒は動作開始命令に対するシステム応答時間よりも十分に長い時間という意図

コード追記後の復調信号(1台で送受信)



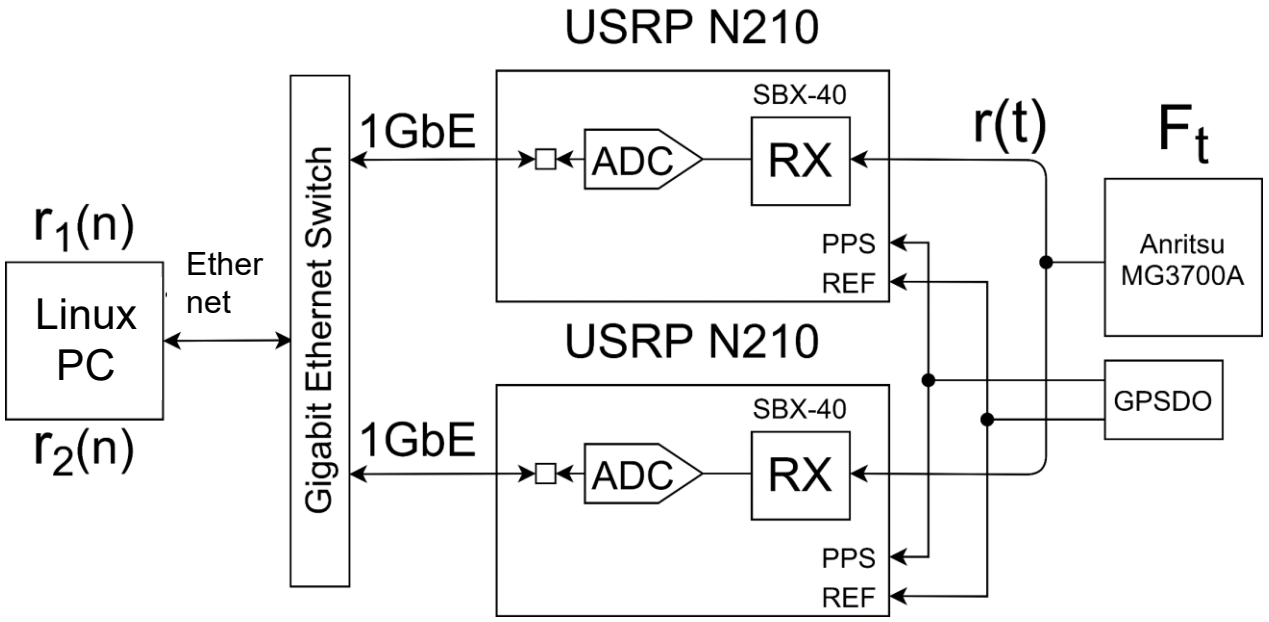
10回の試行の復調信号r(n)の絶対値



10回試行時の復調信号r(n)の
コンスタレーション

- 試行ごとのサンプリング開始タイミングが1サンプル単位で固定化
- 試行ごとのキャリア位相変動が数度程度になった

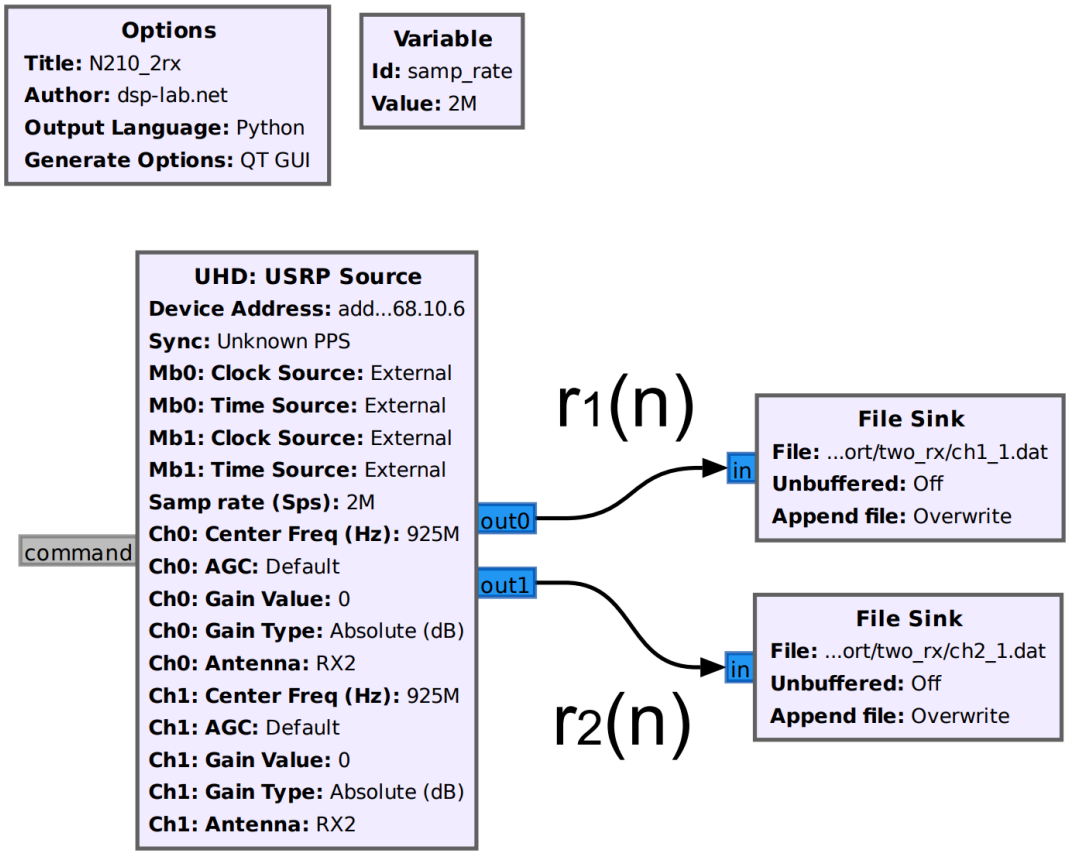
USRP2台で同時受信を行う構成(予稿5.2節)



- $r_1(n)$と$r_2(n)$が同期している
- 異なる試行において，$r_1(n)$と$r_2(n)$の初期位相がいつも同じであったら，コヒーレント受信機としては便利（予稿[11]のEttus社のアプリケーションノートはこれができていない）

- 受信波の到来方向の推定やビームフォーミング
- 電波のセンシング等への応用などに，この構成を基本として発展させられる

GRCのフロー図(2台のUSRPで同時受信)



検証条件

中心周波数 Fc	925MHz
ベースバンド信号の サンプリング周波数 Fsb	2MHz

Ubuntu 18.04.5
GNU Radio 3.8.2.0
UHD 3.15.0.0

同期確立の付加コード(2チャネル同時受信)

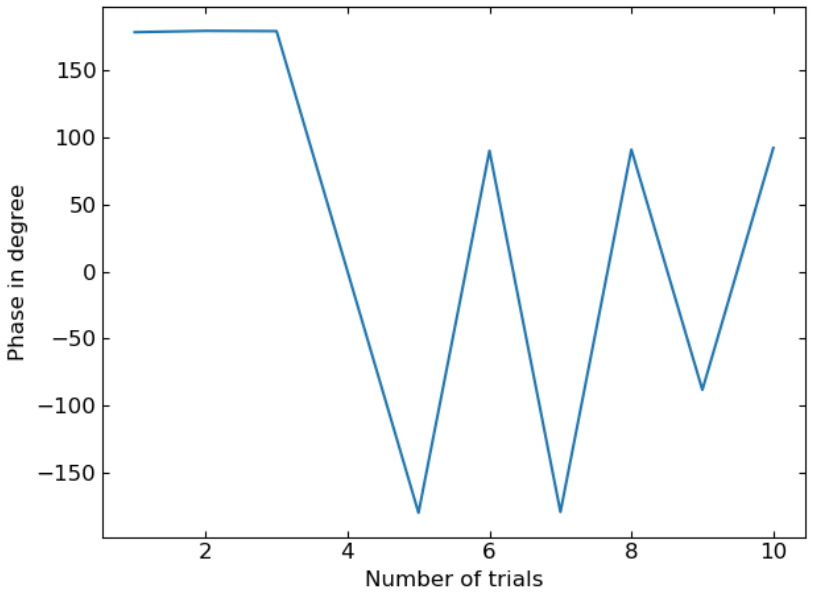
外部から供給するPPSを基準に2台のUSRPの時刻を同期
GRCのSyncでUnknown PPSを指定している場合はすでに記述されている

```
self.uhd_usrp_source_0.set_time_unknown_pps(uhd.time_spec())
time_now = self.uhd_usrp_source_0.get_time_now() + uhd.time_spec(0.01)
self.uhd_usrp_source_0.set_command_time(time_now ,uhd.ALL_MBOARDS)
for i in range(2):
    self.uhd_usrp_source_0.set_center_freq(925e6,i)
self.uhd_usrp_source_0.clear_command_time(uhd.ALL_MBOARDS)
for i in range(2):
    while self.uhd_usrp_source_0.get_sensor("lo_locked",i).to_bool() != True:
        pass
```

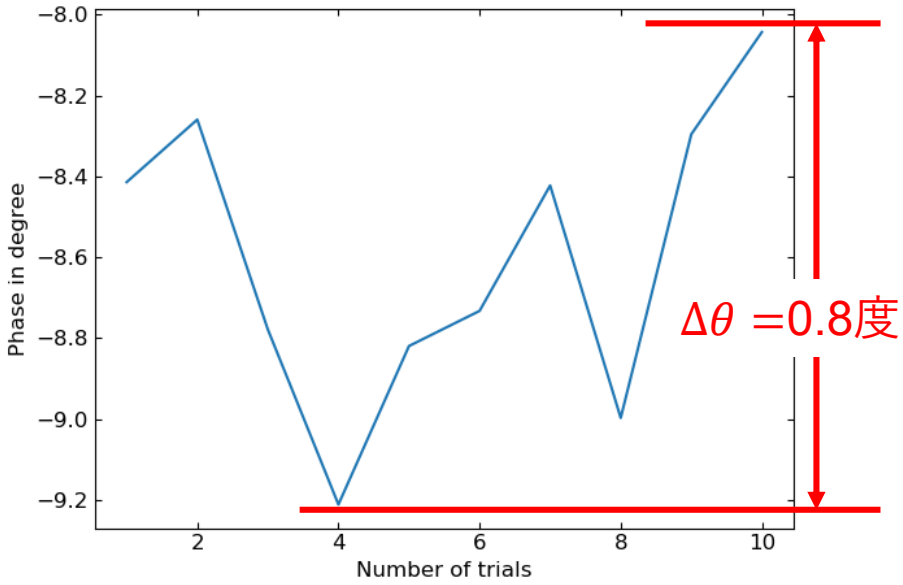
LO位相同期コード

追記：ブロック図1つで複数チャネルを記述した場合はタイミング同期はデフォルトで確立されるためLO位相同期だけ行う.

同期確立コードの有無によるチャネル間位相差



コード適用前



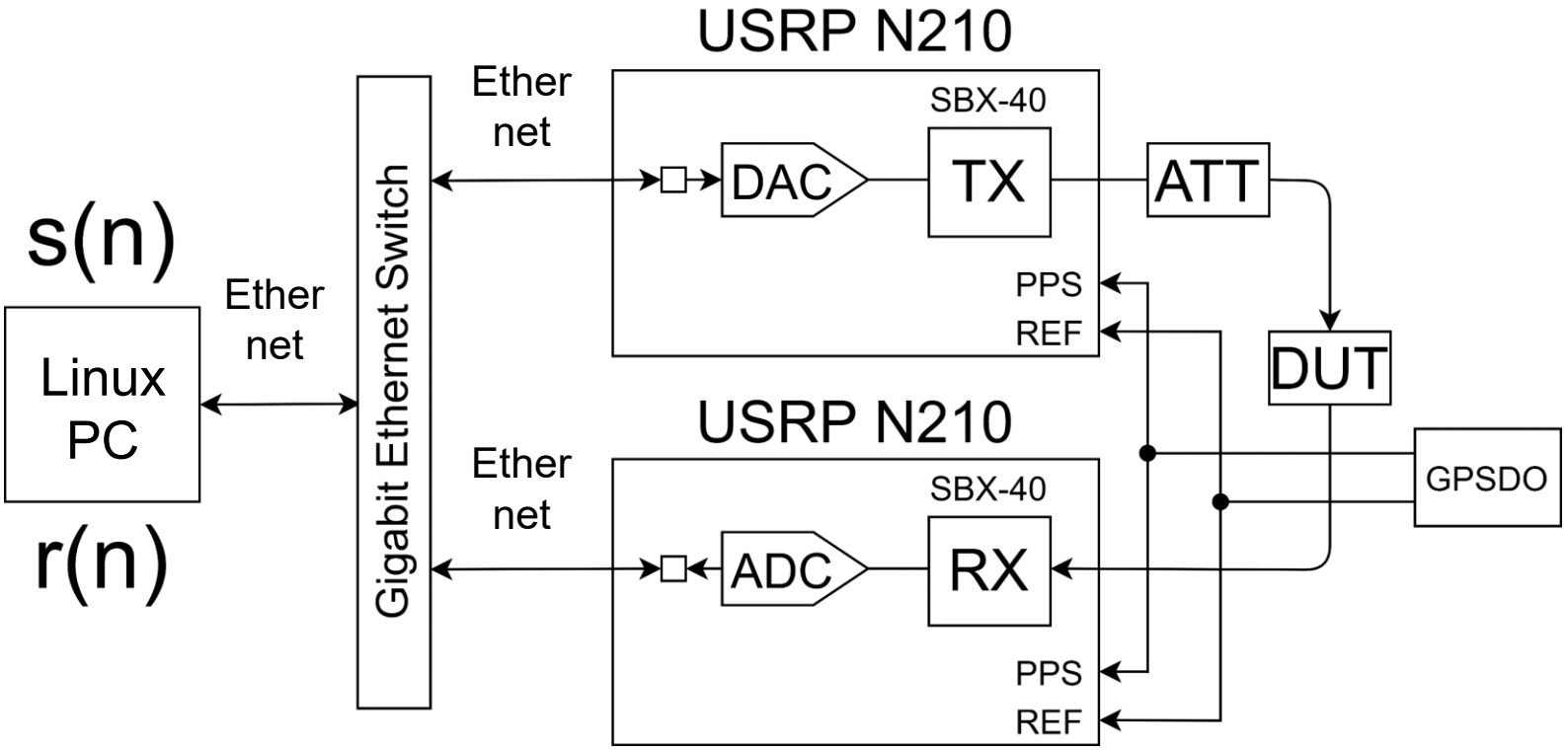
コード適用後

位相算出式　$\Delta\theta = \frac{180}{\pi}\tan^{-1}\frac{Im[R]}{Re[R]}$　$R = \sum_{n=D}^{N+D} r_2(n)r_1^*(n)$

N：位相差算出に使用する信号点数(N=262144)
D：USRPが定常状態になった後の信号を切り出すためのパラメタ

- 10回試行時の$r_1(n)$と$r_2(n)$の位相差の変動が約0.8度(一例)
- ある1回の試行での動作中の位相変動は，別に評価が必要

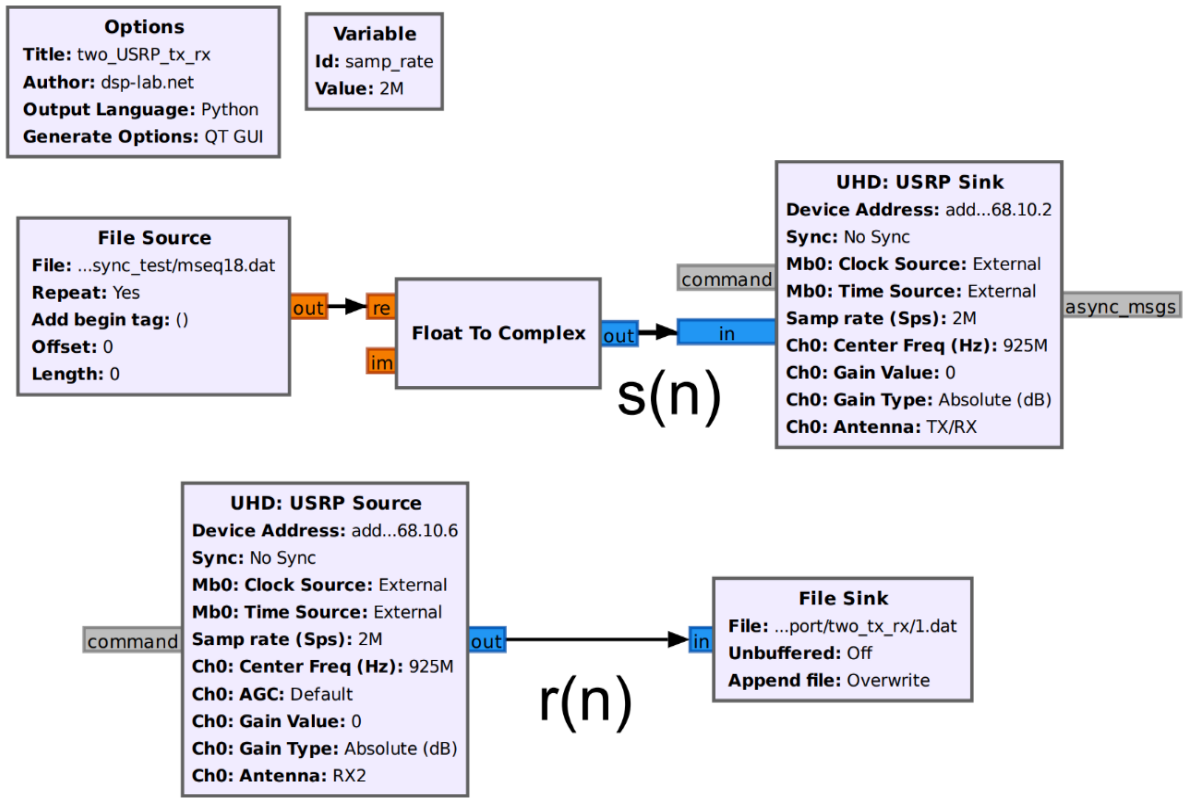
USRP2台で送受信を行う構成(予稿5.3節)



- USRP 1台で同時に送信・受信する構成(5.1節)と同一の要求条件
- s(n)に対して， r(n)が時間的に同期していること
- 試行ごとにr(n)の時間軸が変動しないこと

- 5.1節の構成の拡張版(汎用性が向上)
- 信号の回り込みの低減・帯域幅の増加などの利点

GRCのフロー図(2台のUSRPで送受信)



検証条件

中心周波数 Fc	925MHz
ベースバンド信号の サンプリング周波数 Fsb	2MHz
変調方式	DSB-SC

Ubuntu 18.04.5
GNU Radio 3.8.2.0
UHD 3.15.0.0

送受信間での同期(2台のUSRP)

以下のコードを追加することで同期を確立

```
self.uhd_usrp_source_0.set_time_next_pps(uhd.time_spec())
self.uhd_usrp_sink_0.set_time_next_pps(uhd.time_spec())
time.sleep(1)
```

時刻同期

```
time_now = self.uhd_usrp_source_0.get_time_now() + uhd.time_spec(0.01)
self.uhd_usrp_source_0.set_command_time(time_now)
self.uhd_usrp_source_0.set_center_freq(925e6)
self.uhd_usrp_source_0.clear_command_time()
self.uhd_usrp_sink_0.set_command_time(time_now)
self.uhd_usrp_sink_0.set_center_freq(925e6)
self.uhd_usrp_sink_0.clear_command_time()
while self.uhd_usrp_source_0.get_sensor("lo_locked").to_bool() != True:
    pass
while self.uhd_usrp_sink_0.get_sensor("lo_locked").to_bool() != True:
    pass
```

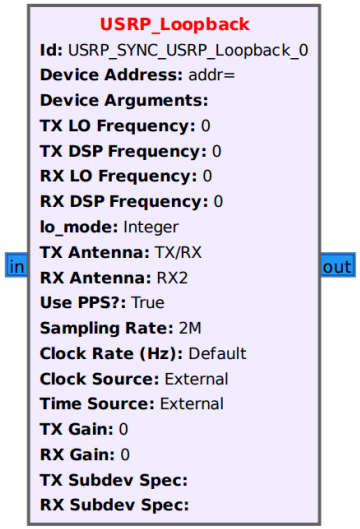
LO位相同期

```
time_now = self.uhd_usrp_source_0.get_time_now() + uhd.time_spec(0.1)
self.uhd_usrp_source_0.set_start_time(time_now)
self.uhd_usrp_sink_0.set_start_time(time_now)
```

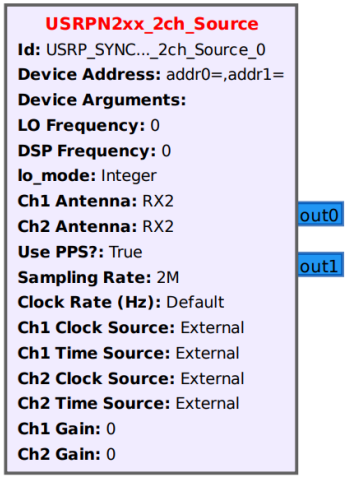
タイミング同期

独自ブロックの作成

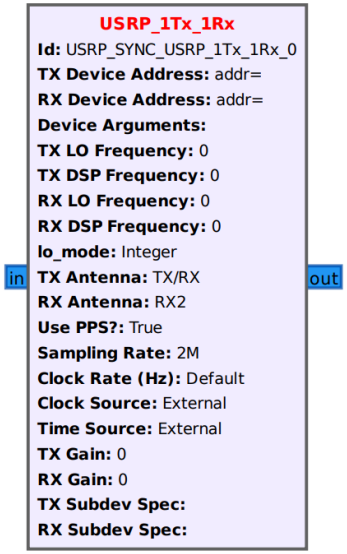
- コードの追加不要でGRC上で同期が確立される
- gr-modtoolを用いて作成
  - 複数の処理やブロックを一つのブロックにまとめるタイプのHierブロックで作成



5.1節の構成用



5.2節の構成用



5.3節の構成用

まとめ

- 3種類の典型的な利用例に対応したタイミング同期・LO位相同期を確立するPython付加コードの実装例を紹介
  - 無線信号観測・測定への利用を想定
- 付加コードの動作確認条件の一例(同一の付加コードが利用可能)
  - USRP N210 + SBX-40 Fc=925MHz, Fsb=2MHz
  - USRP X300 + SBX-120 Fc=920MHz, Fsb=20MHz
  - BasicTX/RXでの利用も可能
- 提案した同期確立機能を備えた，GRC上で使用可能な3種類の独自ブロックを作成
  - 付加コードを手作業で追記する必要がなく，開発効率が向上

紹介した付加コードは，下記からダウンロード可能
https://dsp-lab.net/release

紹介したサンプル実装の位置づけ

- USRP Hardware Driver and USRP Manual[8]のC++ API(application programming interface)記述を参照
  - GNU Radio環境で実行可能なPythonコード実装例を紹介
  - C++でフルスクラッチでUSRPを制御するコードを記述する際には，UHD APIの深い知識が必要
- 現状のGRCでは，UHD APIのプロパティ値を一部しか設定できない
  - Pythonコード上でUHD時刻等の細かいプロパティ値を指定
- 具体的な動くコードを提供する
  - USRPをGNU Radioで使用する際に「必須」と言ってもよい機能なのに，実際的な条件下で動作するコード例が，あまり公開されていない