[依頼講演]時間・位相同期に着目したGNU Radio / UHDによるUSRPの利用事例

－USRP利用時に知っておくと便利なこと－

○山田 洋士
(石川高専 電子情報工学科)

中浜 智也
(金沢大学大学院 M1)

本報告には，第二著者の高専在学中の活動成果を含みます．本報告は，報告時点での第一著者の理解内容に基づいており，責任は第一著者にあります．ソフトウェア，ファームウェア等のバージョンや実行環境により状況が異なる可能性があります．

著者らの興味・関心

- 中浜智也, 山田洋士, 亀田 卓, 本良瑞樹, 末松憲治, “ソフトウェア無線機USRP X300を用いた非因果的成分を考慮したチャネルインパルス応答測定," 電子情報通信学会論文誌, vol.J104-B, no.3, pp.199-209, Mar. 2021. 早期公開日 2020年11月9日. DOI: 10.14923/transcomj.2020GWP0019

- 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, Nov. 2020.

- T. Nakahama, Y. Yamada, S. Kameda, M. Motoyoshi and N. Suematsu, “Implementation and Evaluation of Phase Synchronization of USRP devices in GNU Radio / GRC Environment for Rapid Prototyping,” 2019 RIEC Annual Meeting on Cooperative Research Projects, Feb. 21 2020.

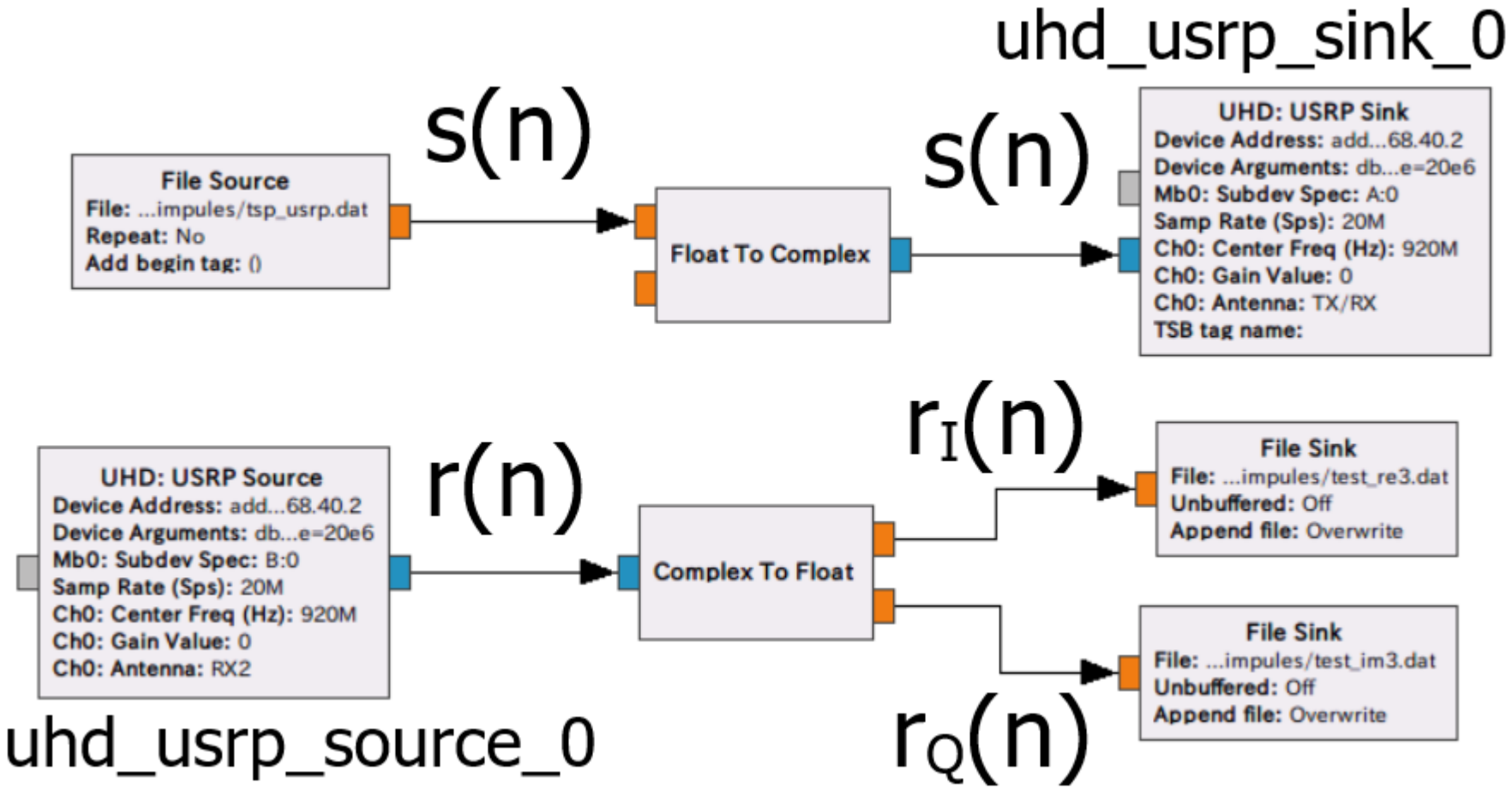
- 無線伝搬路のチャネルインパルス応答を正確に測定するなど， 無線信号計測に興味を持っている
- 近距離での離散的なインパルス応答に非因果成分が観測されることを示した

本報告で想定するUSRP使用環境

- Linuxマシン（Ubuntu20.04 or 18.04）を前提
  - GNU Radio / UHD（USRP Hardware Driver）で使用
  - Pybombsでソースからビルドした環境が便利(GitHUB記載の手順で大丈夫)
- USRP N200シリーズ
  - 1Gigabit Ethernet
- USRP X300シリーズ
  - 10Gigabit Ethernet×2（XGドライバ使用）
  - Intel X710-DA2ボード（X520-DA2も使用中）
  - 10GBASE-CU Twinaxケーブル（Intel XDACBL1M互換 10G SFP+Cable）
  - PCIe-8371ボード（NI社）のLinuxドライバは事実上使えない（2019年9月に調査した古い情報）ように思われる

- PCIe-8371では，NI USRP RIO driver stackをLinuxマシンにインストールする必要がある.
- Currently, the latest supported kernel version is 4.2.x.（2019年当時でもカーネル古すぎ）

GNU Radio Companion(GRC)の使用例



GNU Radio Companion(GRC)でのブロック図の接続例

GRC使用時によく起きる挙動

- PCとUSRP間のデータ伝送に伴う遅延（相対的に充分小さい？）
  - Gigabit Ethernet (GbE or 10GbE)
- PC上の信号処理に伴う遅延（大きい）
  - GNU Radio処理モジュール間でのバッファの存在
  - FMラジオを実装すると，数秒間以上のバッファリングが自動的に行われていることを体感できる
  - アンテナを外しても，放送が聞こえる？
- 連続的にストリームデータが流れている際には，準リアルタイム的に動作しているようだ

- 菅沼久浩, 長縄潤一, 鈴木誠, 猿渡俊介ほか，“ソフトウェア無線機開発支援プロファイラの設計とバッファリングによる最大遅延時間の測定,” 信学ソサイエティ大会, B-17-15, Sept. 2011.
- M. Müller, Behind the Veil: A Peek at GNU Radio’s Buffer Architecture, https://www.gnuradio.org/blog/buffers/

GNU Radio / USRP の構成

USRPX300 + ドータボード

PCへのトランスポート層

データストリーム

レート変換
(FPGA)

ADC14bit, 200Ms/sec
DAC16bit, 800Ms/sec

ドータボード

高周波入出力

10GbE接続

GNU Radio + Linuxマシン

アプリケーション
(Python)

モジュール群
(C++)

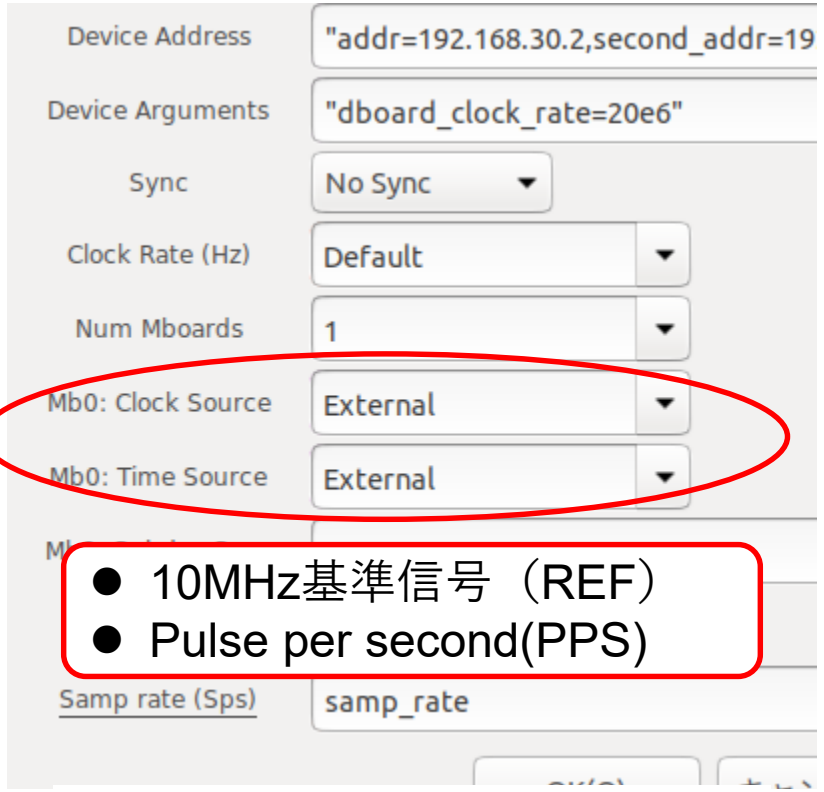
I成分, Q成分

USRPへのトランスポート層

森川博之, 猿渡俊介, GNU Radioで作る無線送受信機, 第7期YRP情報通信技術研修, 2010年11月18日より著者が改変して再作成

GRC利用時によく起こる"同期問題"

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
- GRCそのままでは細かい指定ができない

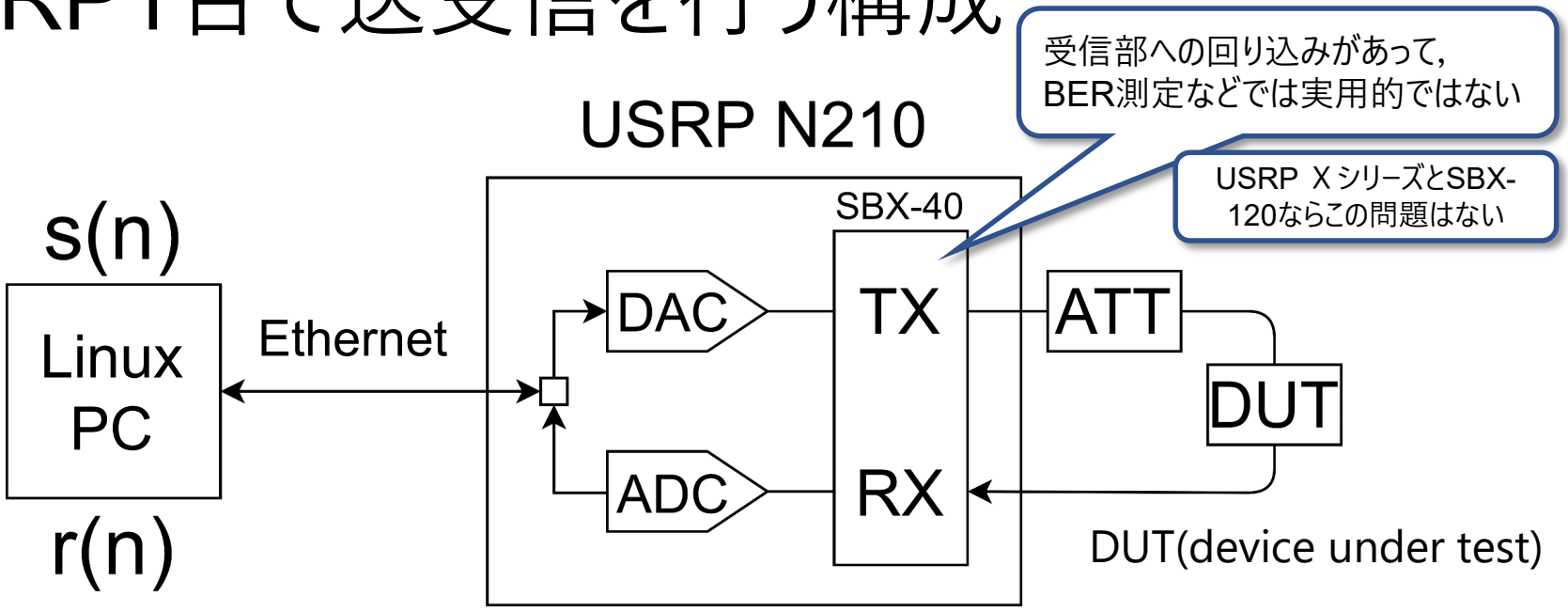
USRPでの測定に適した同期の実現

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, Nov. 2020

- 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

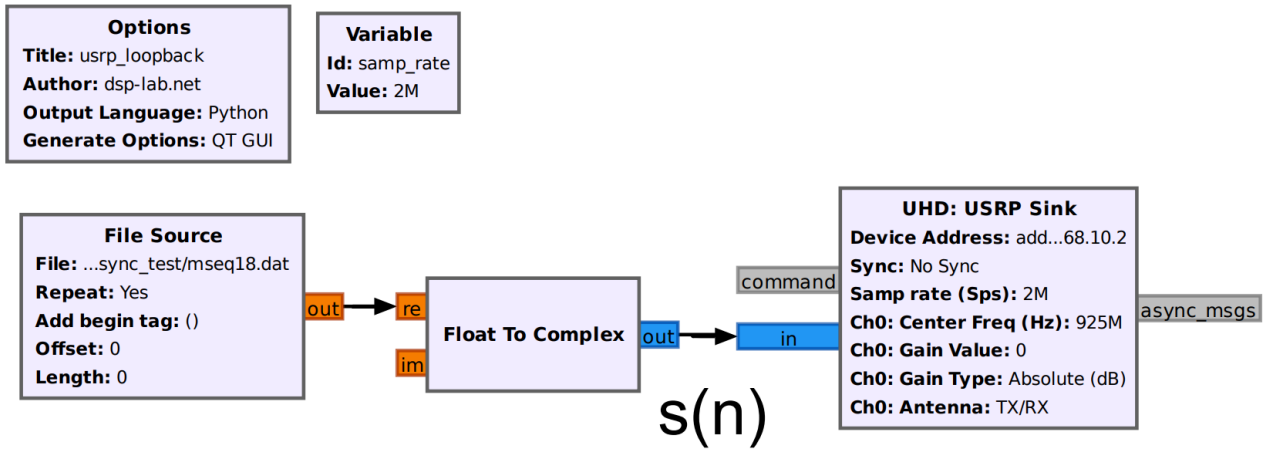
USRP1台で送受信を行う構成

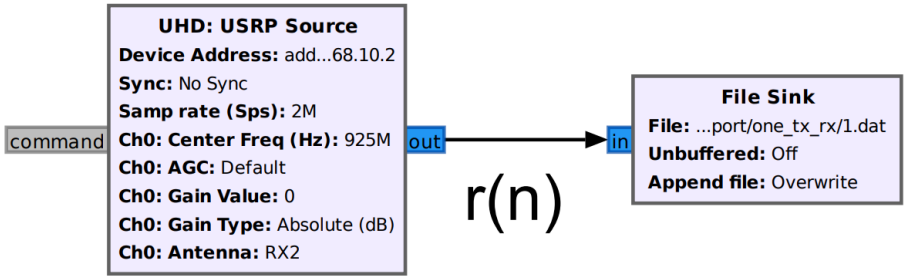


- s(n)で変調した信号を減衰器(ATT)を介して測定対象(DUT)に加え，応答信号を復調した信号r(n)を得る想定（本稿の実行例では，DUTなしで直結）
- s(n)に対して，r(n)が時間的に同期していてほしい
- DUTの有無によるr(n)の違いが測定上必要になる場合があることから，試行ごとにr(n)の時間軸が変動しないことが望まれる

- 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)

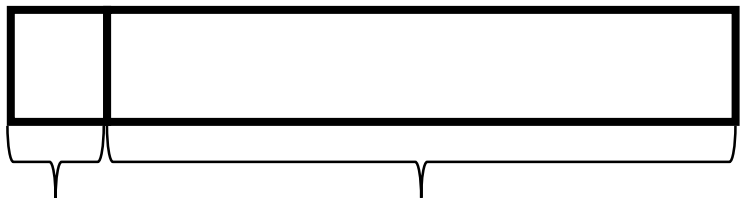




Ubuntu 18.04.5
GNU Radio 3.8.2.0
UHD 3.15.0.0

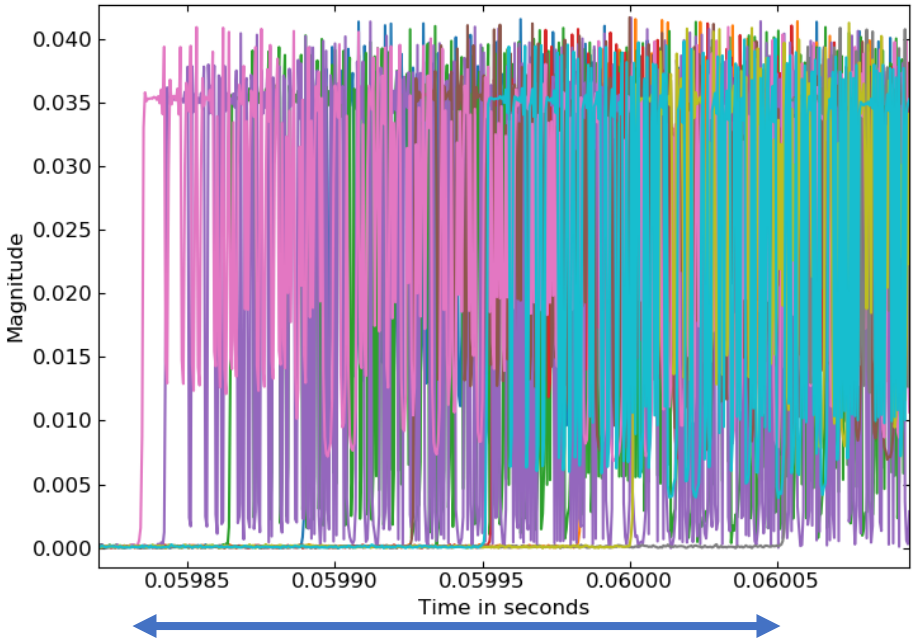
検証条件

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



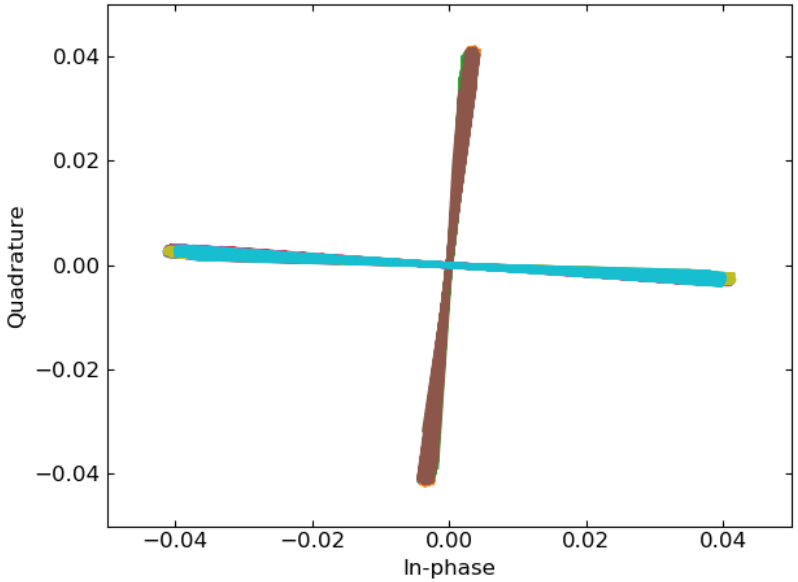
検証に使用するs(n)

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

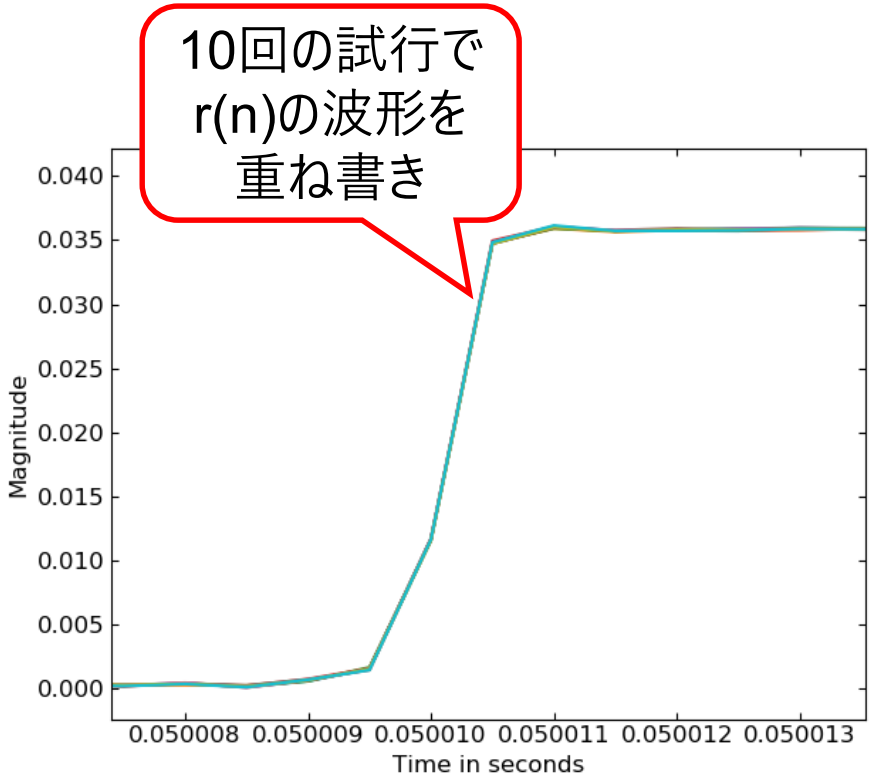


10回の試行の復調信号r(n)の
コンスタレーション

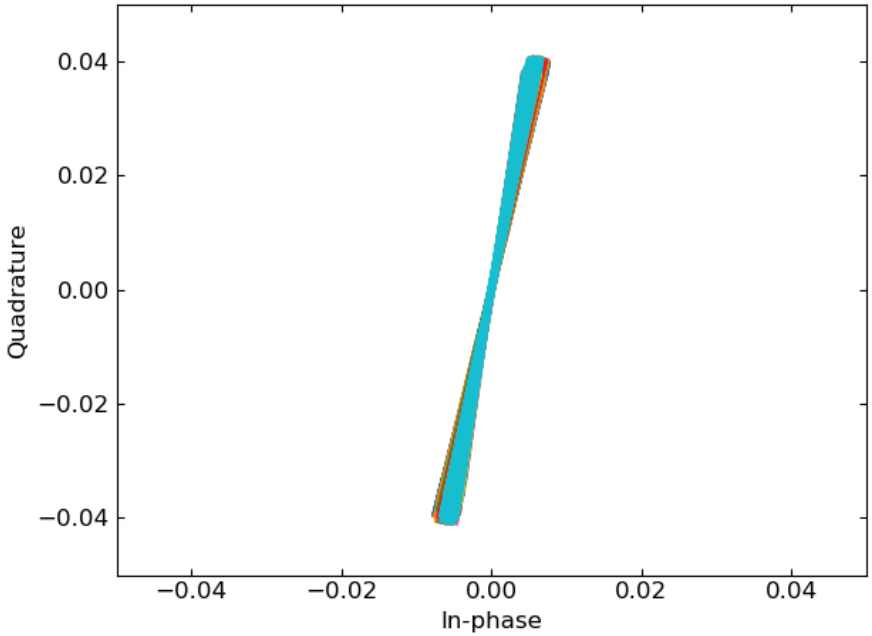
- 試行ごとのサンプリング開始時刻には数100μsの時間差が存在(送信側+受信側)
- 試行ごとに位相が変動する(2パターン)

Fc=927MHz
なら10パターン

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

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Ettus社の文献記載のAPI呼び出し例に従った

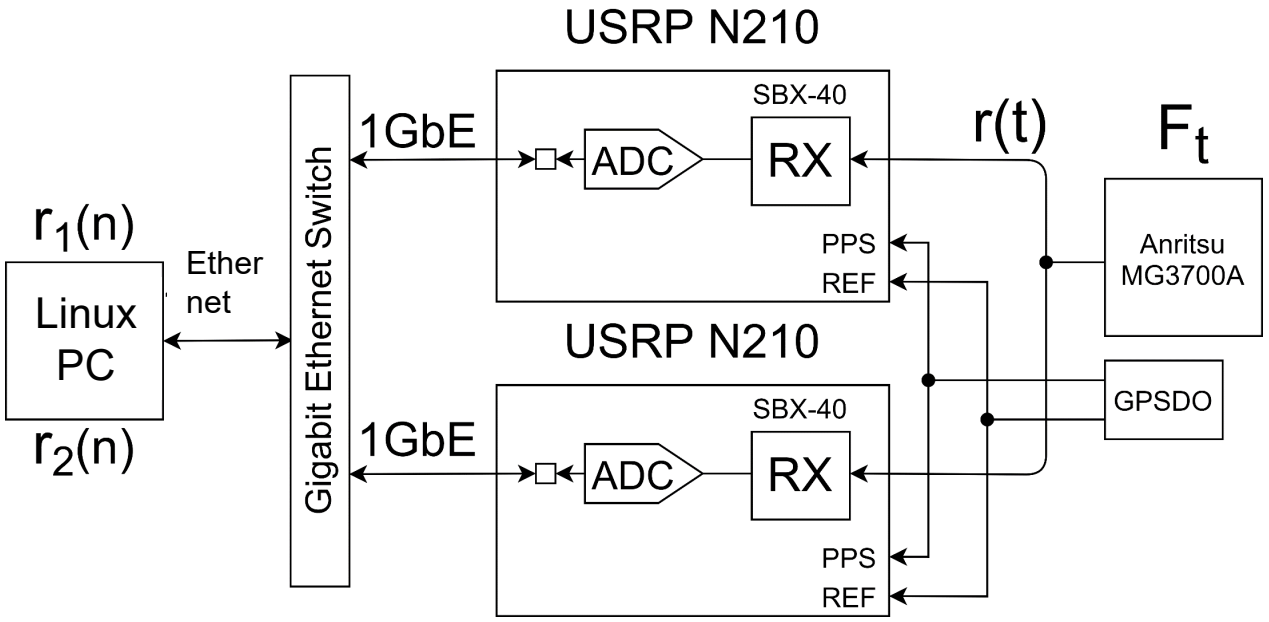
```
time_now = self.uhd_usrp_source_0.get_time_now() + uhd.time_spec(0.01)
self.uhd_usrp_source_0.set_command_time(time_now)
self.uhd_usrp_source_0.set_center_freq(925e6)
self.uhd_usrp_source_0.clear_command_time()
self.uhd_usrp_sink_0.set_command_time(time_now)
self.uhd_usrp_sink_0.set_center_freq(925e6)
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

USRP2台で同時受信を行う構成

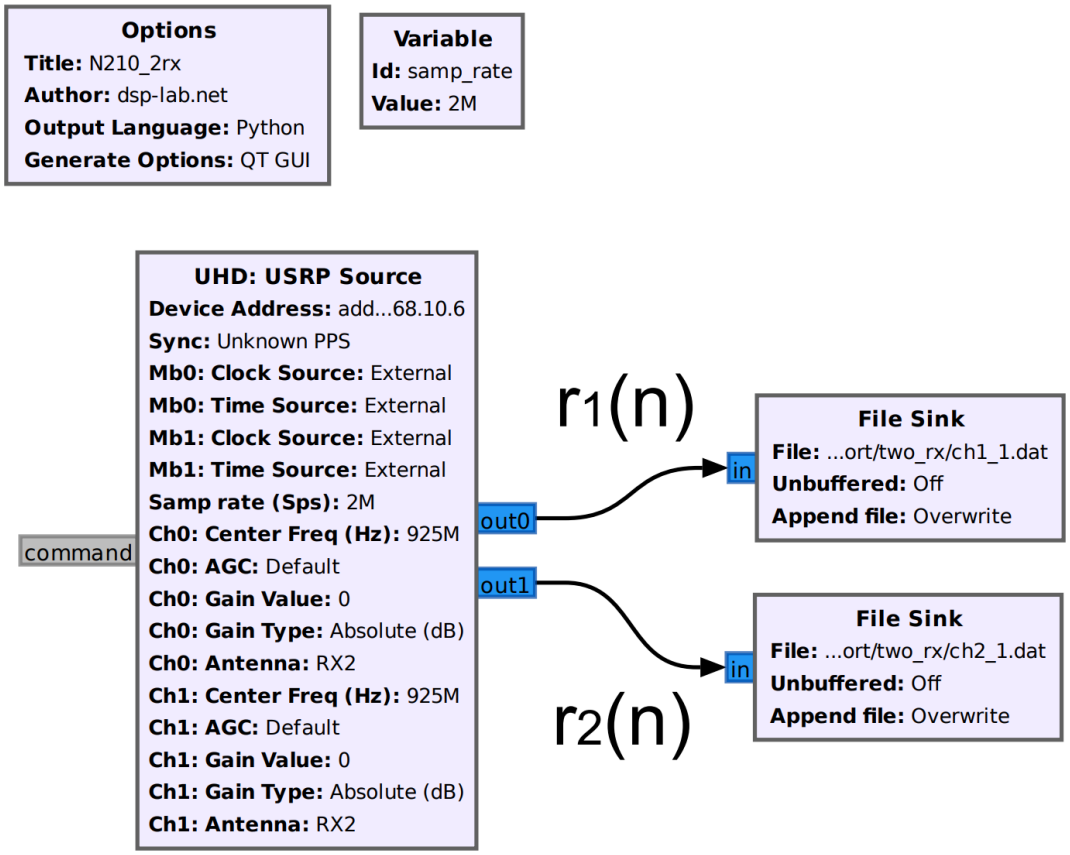


- $r_1(n)$と$r_2(n)$が同期している
- 異なる試行において，$r_1(n)$と$r_2(n)$の初期位相がいつも同じであったら，コヒーレント受信機としては便利（Ettus社のアプリケーションノートはこれができていない）

AN-882:Synchronization and MIMO Capability with USRP Devices

- 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)

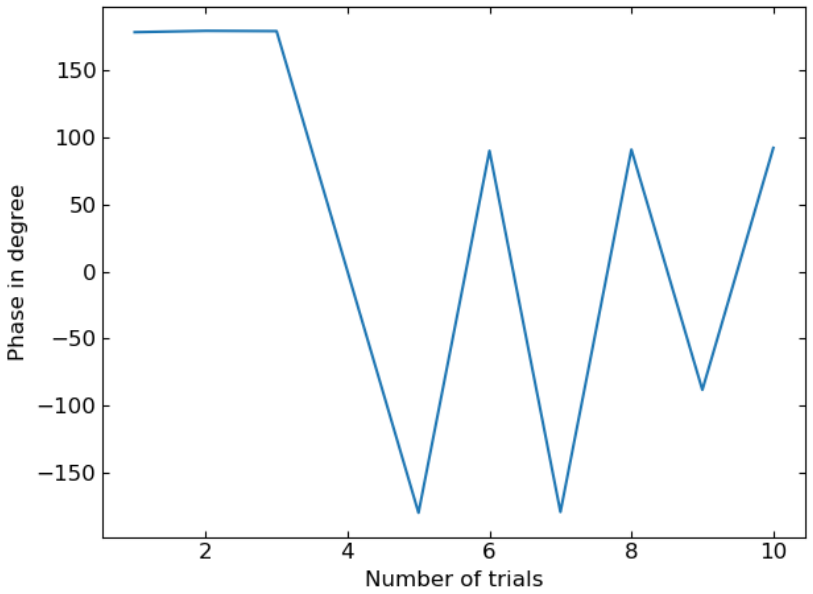
外部から供給するPPSを基準に2台のUSRPの時刻を同期
GRCのSyncでUnknown PPSを指定している場合はすでに記述されている

```
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
```

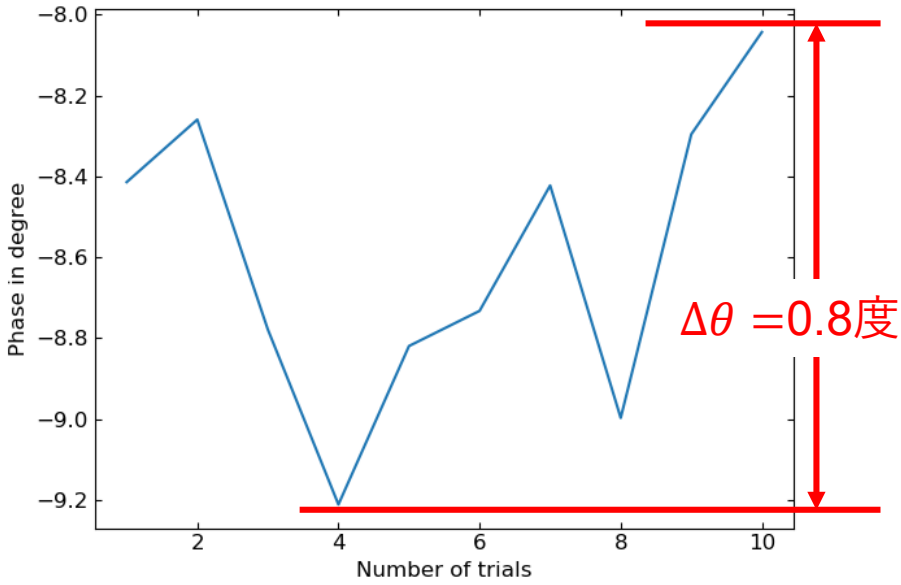
LO位相同期コード

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

ドータボードは
アナログ回路

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

本報告での同期の定義・種類

- 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

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

望ましい要件が異なる

- 通信における同期
- 無線信号計測における同期

LO位相同期（キャリア位相同期）

LO:local oscillator

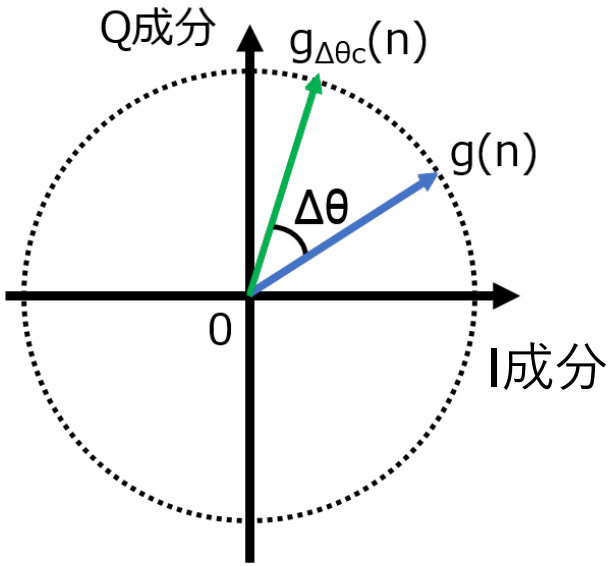
- LO位相ずれはLOの位相が試行ごとに異なることで発生
- LOを有するドーターボード利用時に発生(SBXやUBX)
- $\Delta\theta_c$が試行ごとに変動しないようにする

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

920MHz

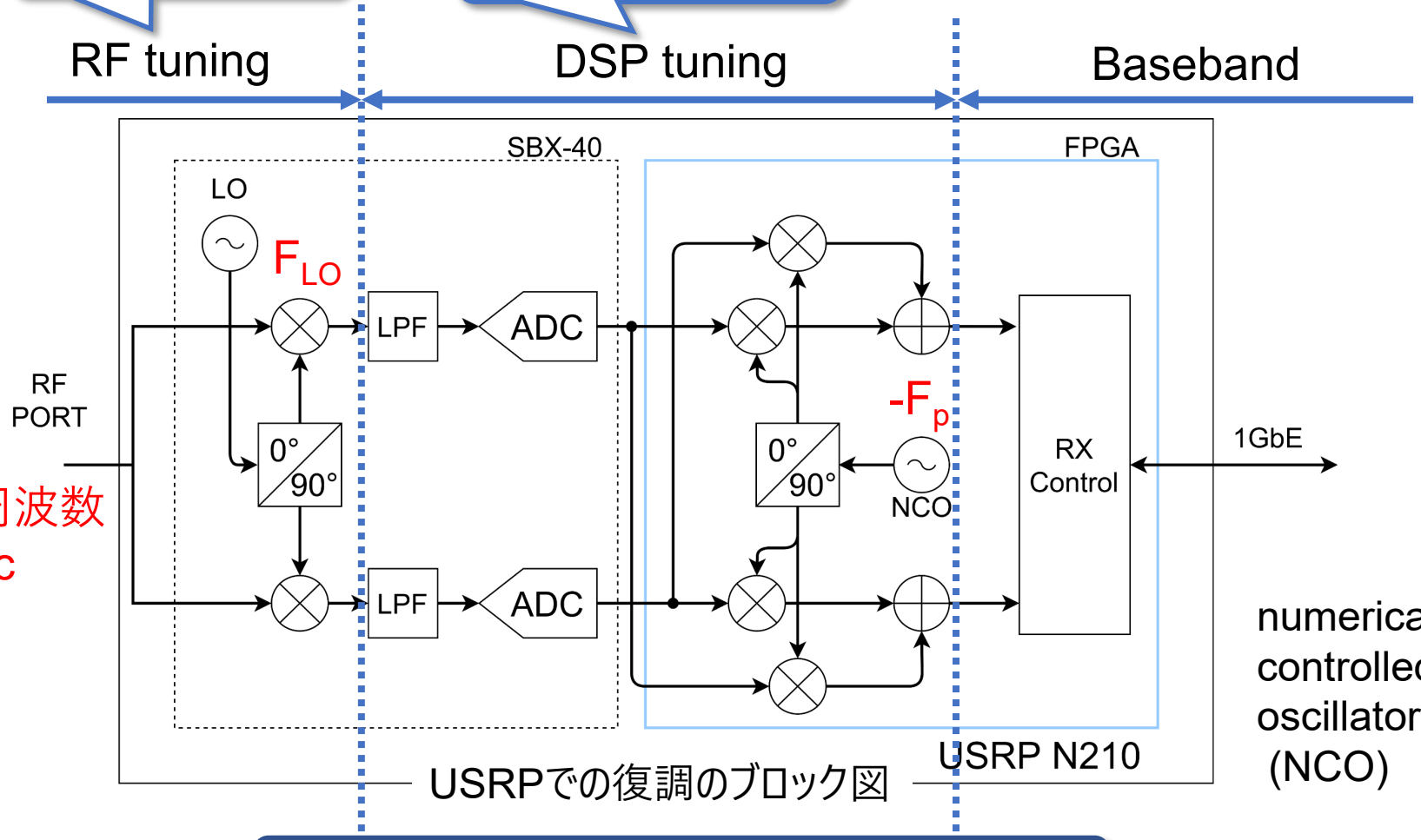
920MHz
rf_freq

0MHz
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



USRPでの復調のブロック図

□ キャリア位相ずれは2か所で発生し得る

USRPのキャリア周波数は要注意

- USRP2の時代は，内部の基準信号発振器の精度に課題
  - 外部基準信号の供給で解決できた
  - 例:Fc=920MHz 偏差7kHz
- Integer tuningとFractional tuningというモードが存在
- IntegerモードでFLOをセットし，dsp_freqで調整するモードが高精度のようだ
  - 920MHz
  - 920MHz rf_freq
  - 0MHz dsp_freq

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

  - 許容すべき挙動と考えられる。USRPは標準信号発生器ではない。
  - two stage tuning processと呼ばれている
  - 設定するキャリア周波数によって，Integer tuningができない場合もある
- USRP N200とUSRP X300とで挙動が異なる
  - USRP X300+SBX-120で，送信・受信周波数の実際の値が異なる挙動
    - 受信がずれる

- 通信では，受信機での同期によって吸収される場合もあるかもしれない
- 高精度測定の用途には支障がある

USRPでキャリア周波数が合わない例

－ tune requestに対する応答 －

920MHzなら設定値は，ずれない

（例）USRP X300でFc=927MHz指定(Fractional tuning指定)

- RX/TX FLO=926.9987MHz, dsp_freq=0 表示（同期コードあり）
- 設定周波数値とずれがあるが，本当に送受信で同じずれ方なら，問題は少ない

N200のほうが問題が少ない

独自にGRCモジュールを作成して
やや極端な条件で強制指定

（例）USRP X300+SBX-120でIntegerモード指定でFc=927MHz指定

- TX FLO=920MHz, dsp_freq = 6.999MHz 表示　実際：約927MHz
- RX FLO=920MHz, dsp_freq = −6.999MHz 表示　実際：約920MHz

送信と受信で「ずれ」が異なる

X300+BasicRX(LOなしのボード)で，BasicRXの使用周波
範囲ではこの種の問題はない(X300のマザーボードは問題ない)

N200+SBXでは
上記の挙動はない

- USRP X300+SBX-120で送信・受信を行う場合に，キャリア周波数の値によっては，周波数設定値が同じでも，送信側・受信側で15Hz程度周波数が合わないことがある
- 使用するキャリア周波数で検証してから実験！
- LabVIEWでもUHDを用いているので，同様の性質を有する可能性が高い

一例:Fc=927MHz
のとき

USRPを使用した論文でお願いしたいこと

- 使用したUSRPの種類，ドータボード，PCとの接続方法
- ソフトウェア（GNU Radio, UHD, LabVIEW）のバージョン
- ソフトウェア開発環境を明示してほしい
- GNU Radio
  - C++またはPythonフルスクラッチ（UHD使用）
  - GRC使用（UHD使用）
  - GRC+Timed commands使用（UHD使用）
- LabVIEW
  - 通常のLabVIEWで開発（UHD使用）
  - LabVIEW FPGAで開発（多くの場合は，UHDを使用しないはず）

有効性・信頼性の観点から

- USRPを用いた，とだけ書いてあっても，判断できないのでは？
- コードの明示が不可だとしても，環境は記述してほしい

まとめ

- GNU Radio / GRC使用時に，疑問に思われる点について紹介
  - キャリア周波数設定については，独自解釈になっているかもしれない
  - キャリア周波数の設定値と実際の値がずれるのは当然としても，ずれ方が設定値によって異なることと，送信・受信で異なるずれ方をする場合が存在することを意識して，計測・観測分野での応用では回避する必要がある
  - USRP X300とドータボードの組み合わせに注意
- GNU Radioはソースコードが公開されており，USRPは回路図が公開されている
  - 利用者がソフトウェアを開発し，検証して利用することが求められている
- 今後のGNU Radio
  - UHD 4.xとRFNoc(RF Network on Chip)
  - https://www.ettus.com/sdr-software/rfnoc/

- GNU RadioとUSRPの今後に期待
- PCIeバスやSSDの高速化が，汎用PCを使用するソフトウェア無線機の発展につながる