직접 변환 SDR의 DC 오프셋 제거 최적화: AD936x 및 ADRV9002 아키텍처 비교 분석 및 구현 가이드

제 1장: 직접 변환 아키텍처에서의 DC 오프셋 발생 원리

1.1. 제로-IF 패러다임: 집적화의 혁신

직접 변환 수신기(Direct-Conversion Receiver, DCR)는 호모다인(Homodyne) 또는 제로-IF(Zero-IF) 수신기로도 알려져 있으며, RF(Radio Frequency) 신호를 단일 믹싱 단계를 통해 기저대역(Baseband)으로 직접 복조하는 혁신적인 설계 방식입니다.[1] 이 접근 방식은 중간 주파수(Intermediate Frequency, IF) 단계를 사용하는 전통적인 슈퍼헤테로다인(Superheterodyne) 아키텍처와 대조됩니다. DCR의 핵심 장점은 이미지 제거 필터(Image-Reject Filter)의 불필요, 부품 수 감소, 비용 절감, 그리고 단일 칩 집적화에 대한 높은 용이성에 있습니다.[1] 이러한 장점 덕분에 현대의 수많은 무선 통신 시스템에서 DCR 아키텍처가 널리 채택되고 있습니다.

하지만 이러한 구조적 단순성에도 불구하고, 제로-IF 설계는 DC 오프셋(DC Offset)과 I/Q 불균형(I/Q Imbalance)이라는 두 가지 중대한 기술적 과제를 내포하고 있습니다.[3] 이 보고서에서는 DC 오프셋을 단순한 설계 결함이 아닌, 제로-IF 아키텍처의 본질적인 부산물로 간주하고, 이를 효과적으로 완화하기 위한 정교한 전략을 탐구하고자 합니다.

1.2. 근본 원인 분석: DC 오프셋의 다양한 출처

DC 오프셋은 단일 원인이 아닌 여러 요인이 복합적으로 작용하여 발생합니다. 주요 원인은 다음과 같습니다.

- **LO** 누설 및 자체 혼합 **(LO Leakage and Self-Mixing):** 이는 DC 오프셋의 가장 주된 원인입니다. 믹서 내부의 불완전한 절연(Isolation)으로 인해 국부 발진기(Local Oscillator, LO) 신호가 RF 입력 포트나 저잡음 증폭기(Low Noise Amplifier, LNA) 출력으로 누설됩니다.[4] 이렇게 누설된 LO 신호는 다시 주 LO 신호와 혼합되는 "자체 혼합" 현상을 겪게 되며, 이 과정에서 믹서 출력에 원치 않는 DC 성분이 생성됩니다.[2] 이 현상은 송신기에서 발생하는 LO 누설(LOL) 문제와 원리적으로 유사합니다.[7]
- **2**차 비선형성 **(Second-Order Non-Linearity, IP2):** 수신 대역 내에 강력한 간섭 신호가 존재할 경우, LNA와 믹서의 2차 비선형성으로 인해 이 간섭 신호가 누설되어 자기 자신과 혼합될 수 있습니다. 이 과정 역시 시간에 따라 변동할 수 있는 추가적인 DC 오프셋 성분을 생성합니다.[5] 이는 DC 오프셋 완화에 있어 수신기의 선형성(높은 IP2 값)이 왜 중요한지를 명확히 보여줍니다.
- 소자 부정합 **(Device Mismatches):** 믹서에서부터 기저대역 증폭기 및 필터에 이르는 차동 I(In-phase) 및 Q(Quadrature) 신호 경로 간의 미세한 물리적 부정합 또한 정적인 DC 오프셋을 유발하는 원인이 됩니다.[3]
- 동적 **DC** 오프셋 **(Dynamic DC Offset):** 특히 해결하기 까다로운 문제는 동적 또는 시변(Time-Varying) DC 오프셋입니다. 이는 누설된 LO 신호가 안테나를 통해 방사된 후, 주변의 움직이는 물체에 반사되어 다시 수신 안테나로 유입될 때 발생합니다.[2] 이렇게 생성된 오프셋은 시간에 따라 변동하기 때문에, 단순한 정적 보상 방법으로는 효과적으로 제거할 수 없습니다.

이러한 원인들을 종합해 볼 때, DC 오프셋 문제는 단일 현상이 아니라 정적 요인(소자 부정합, 직접적인 LO 누설)과 동적 요인(환경적 반사, 강력한 간섭 신호)이 결합된 복합적인 문제입니다. 이러한 구분은 해결책의 복잡성을 결정하는 데 매우 중요합니다. 정적 성분은 단 한 번의 초기 보정(Calibration)으로 해결할 수 있지만, 동적 성분에 효과적으로 대응하고 견고한 성능을 확보하기 위해서는 지속적인 추적(Tracking) 알고리즘이 필수적입니다. 이는 AD936x와 같은 초기 아키텍처와 ADRV9002와 같은 최신 아키텍처 간의 설계 철학 차이를 이해하는 핵심적인 단서가 됩니다.

1.3. 시스템 성능에 미치는 영향**:** 단순한 **DC** 스파이크를 넘어서

보정되지 않은 DC 오프셋은 시스템 성능에 심각한 악영향을 미칩니다. 생성된 DC 전압은 원하는 기저대역 신호의 크기보다 수십, 수백 배 더 클 수 있습니다.[2] 이처럼 큰 DC 성분은 후단의 기저대역 증폭기와 아날로그-디지털 변환기(ADC)를 포화(Saturation)시켜 수신기의 동적 범위(Dynamic Range)를 급격히 감소시키고, 미약한 신호의 탐지를 불가능하게 만듭니다.[4]

이러한 영향은 수신된 스펙트럼의 중심 주파수(0 Hz)에서 크고 지속적인 스파이크(Spike) 형태로 가장 흔하게 관찰됩니다. 이 스파이크는 DC 근처에 위치한 목표 신호를 가리거나 간섭을 일으켜 통신 품질을 저하시키는 주된 요인이 됩니다.[9]

특히 DC 오프셋 문제의 심각성은 수신기 이득(Gain)과 직접적으로 연관됩니다. 미약 신호를 수신하기 위해 필요한 높은 이득 설정에서는 아주 작은 LO 누설 신호조차 기저대역에서 크게 증폭되어 포화 문제를 더욱 악화시킵니다.[12] 이는 감도(Sensitivity)와 DC 오프셋 내성(Tolerance) 사이에 근본적인 상충 관계(Trade-off)를 만들어내며, 이는 직접 변환 수신기 시스템 설계의 핵심 과제 중 하나입니다.

제 2장: AD9361/AD9363 기반 SDR(ADALM-PLUTO)의 DC 오프셋 완화

이 장에서는 널리 사용되는 AD936x 플랫폼에서 DC 오프셋 제거를 최적화하기 위한 실질적인 가이드를 제공합니다. 최상의 결과를 얻기 위해 하드웨어 기능과 소프트웨어 기법을 결합한 하이브리드 접근 방식을 제안합니다.

2.1. AD9361/AD9363 수신기 경로의 아키텍처 개요

AD9361 및 AD9363은 고도로 집적된 직접 변환 방식의 RF 트랜시버입니다.[13] 이 칩의 수신기 체인은 내부 DC 오프셋 보정, 직교 오차 보정(Quadrature Correction), 그리고 프로그래밍 가능한 디지털 FIR 필터와 같은 핵심 기능들을 포함하고 있습니다.[13] 이러한 높은 수준의 집적도는 ADALM-PLUTO와 같은 플랫폼이 강력한 성능과 높은 접근성을 동시에 제공할 수 있는 기반이 됩니다.[15]

2.2. 온칩 보정 메커니즘

AD936x 제품군은 DC 오프셋을 완화하기 위한 내부 보정 루틴을 내장하고 있습니다. 이 기능들은 주로 기저대역 DC 추적(Baseband DC Tracking)에 초점을 맞추고 있습니다.[12] Analog Devices가 제공하는 Linux(IIO) 및 No-OS 환경용 디바이스 드라이버는 이러한 보정 기능을 관리하는 데 필요한 저수준 레지스터 제어를 추상화하여 제공합니다.[16] 하드웨어가 상당한 수준의 보정을 제공하지만, 사용자 보고 및 일반적인 사용 사례에 따르면 약간의 잔여 DC 성분이 남는 경우가 많습니다.[10]

2.3. 실제 구현(POC): 최적 성능을 위한 하이브리드 접근 방식

AD936x 기반 장치에 대한 가장 효과적인 전략은 다층적 접근 방식을 채택하는 것입니다. 즉, 온칩 하드웨어 보정을 1차 방어선으로 사용하고, 소프트웨어 기법을 적용하여 잔여 아티팩트를 제거하거나 하드웨어의 한계를 우회하는 것입니다.

2.3.1. pyadi-iio를 통한 온칩 보정 활용

pyadi-iio 파이썬 라이브러리는 Linux IIO 프레임워크를 통해 Analog Devices 하드웨어를 제어하는 고수준 인터페이스를 제공합니다.[15] AD9363을 사용하는 ADALM-PLUTO의 경우, 이것이 표준 소프트웨어 제어 방법입니다.[13]

아래의 표는 ADALM-PLUTO에서 DC 오프셋 관리를 위해 사용하는 핵심 pyadi-iio 제어 속성을 요약한 것입니다.

표 **1: AD936x (ADALM-PLUTO) DC** 오프셋 관리 속성

속성	설명	타입
sdr.rx_bbdc_tracking_en	온칩 기저대역 DC 추적 보정 기능을 활성화/비활성화합니다.	bool

POC 구현: 다음 파이썬 스크립트는 adi.Pluto 객체를 생성하고 기저대역 DC(BBDC) 추적 보정을 제어하는 방법을 보여줍니다. 핵심 속성은 sdr.rx_bbdc_tracking_en입니다. 이 코드는 해당 기능을 활성화하고 I/Q 데이터를 수집하는 과정을 보여줍니다.

Python

```
import adi
import matplotlib.pyplot as plt
import numpy as np

# ADALM-PLUTO SDR 초기화
```

```
sdr = adi.Pluto("ip:192.168.2.1")
sdr.sample_rate = int(2e6)
sdr.rx_rf_bandwidth = int(2e6)
sdr.rx_lo = int(2400e6)
sdr.rx_buffer_size = 2**12
sdr.gain_control_mode_chan0 = "manual"
sdr.rx_hardwaregain_chan0 = 70.0

# 1. BBDC 추적 비활성화 상태에서 데이터 수집
sdr.rx_bbdc_tracking_en = False
data_bbdc_off = sdr.rx()

# 2. BBDC 추적 활성화 상태에서 데이터 수집
sdr.rx_bbdc_tracking_en = True
data_bbdc_on = sdr.rx()

# FFT 결과 시각화
plt.figure(figsize=(12, 6))
plt.psd(data_bbdc_off, NFFT=1024, Fs=sdr.sample_rate, label='BBDC Tracking OFF')
plt.psd(data_bbdc_on, NFFT=1024, Fs=sdr.sample_rate, label='BBDC Tracking ON')
plt.title("ADALM-PLUTO On-Chip BBDC Tracking Effect")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Power Spectral Density (dB/Hz)")
plt.legend()
plt.grid(True)
plt.show()

del sdr
```

검증: 위 코드를 실행하면 생성되는 FFT 플롯은 하드웨어 추적 기능이 활성화되었을 때 중심 주파수(0 Hz)의 DC 스파이크가 현저히 감소하는 것을 시각적으로 보여줍니다.

2.3.2. 소프트웨어 증강 - 오프셋 튜닝 (디지털 IF)

오프셋 튜닝은 DC 오프셋 문제를 효과적으로 우회하는 데 널리 사용되는 강력한 소프트웨어 기법입니다.[9] 이 기법의 원리는 SDR의 하드웨어 LO를 원하는 신호의 중심 주파수에서 약간 벗어난 주파수로 설정하는 것입니다. 이렇게 하면 문제가 되는 DC 아티팩트가 목표 신호 대역 밖인 오프셋 주파수로 이동하게 됩니다. 이후 수신된 신호는 소프트웨어에서 디지털 방식으로 주파수를 이동시켜 다시 기저대역으로 중심을 맞춥니다.

POC 구현: 이전 POC 코드를 확장하여 다음을 수행합니다.

1. 주파수 오프셋(예: 100 kHz)을 정의합니다.
2. Pluto의 LO 주파수를 target_frequency + offset으로 설정합니다.
3. I/Q 데이터를 수신합니다.
4. NumPy를 사용하여 복소 I/Q 샘플에 $e^{-j2\pi \frac{f_{offset}}{f_s} n}$을 곱하여 디지털 주파수 이동을 적용합니다.

Python

```
import adi
import matplotlib.pyplot as plt
import numpy as np

# SDR 초기화
sdr = adi.Pluto("ip:192.168.2.1")
sdr.sample_rate = int(2e6)
sdr.rx_rf_bandwidth = int(2e6)
sdr.rx_buffer_size = 2**12
sdr.gain_control_mode_chan0 = "manual"
sdr.rx_hardwaregain_chan0 = 70.0

# BBDC 추적은 활성화된 상태로 유지
sdr.rx_bbdc_tracking_en = True

# 오프셋 튜닝 파라미터
target_freq = int(2400e6)
freq_offset = int(100e3) # 100 kHz 오프셋

# 하드웨어 LO를 오프셋 주파수로 설정
sdr.rx_lo = target_freq + freq_offset

# 데이터 수집
rx_samples = sdr.rx()

# 디지털 주파수 이동 (소프트웨어에서 보상)
num_samples = len(rx_samples)
n = np.arange(num_samples)
correction_signal = np.exp(-1j * 2 * np.pi * freq_offset / sdr.sample_rate * n)
corrected_samples = rx_samples * correction_signal
```

```
# FFT 결과 시각화
plt.figure(figsize=(12, 6))
# 오프셋 튜닝 전 스펙트럼 (DC 스파이크가 0 Hz에 위치)
plt.psd(rx_samples, NFFT=1024, Fs=sdr.sample_rate, Fc=sdr.rx_lo, label='Before Digital Shift (DC at -100kHz)')
# 오프셋 튜닝 후 스펙트럼 (DC 스파이크가 -100kHz로 이동)
plt.psd(corrected_samples, NFFT=1024, Fs=sdr.sample_rate, Fc=target_freq, label='After Digital Shift (Signal at 0Hz)')
plt.title("Software Offset Tuning Effect")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Power Spectral Density (dB/Hz)")
plt.legend()
plt.grid(True)
plt.show()

del sdr
```

검증**:** 생성된 FFT 플롯은 목표 신호가 0 Hz에 중심을 맞추고, DC 아티팩트는 -offset Hz(-100 kHz)로 이동하여 채널 필터에 의해 쉽게 제거될 수 있음을 보여줍니다.

2.3.3. 소프트웨어 증강 **-** 디지털 **DC** 블로킹 필터

오프셋 튜닝이 바람직하지 않은 경우(예: 관심 신호가 실제로 DC에 있는 경우), 디지털 고역 통과 필터(High-Pass Filter)를 사용하여 I/Q 데이터 스트림에서 DC 성분을 직접 제거할 수 있습니다.[11] 간단하고 계산 효율적인 방법은 1차 IIR 필터를 사용하는 것입니다.[24]

POC 구현**:** 다음 파이썬 함수는 차분 방정식 $y[n] = x[n] - x[n-1] + R \cdot y[n-1]$을 구현합니다. 여기서 R은 1에 가까운 계수(예: 0.995)입니다. 이 필터는 수신된 데이터의 I 및 Q 스트림 각각에 적용됩니다.

Python

```
import adi
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import lfilter
```

```
def dc_blocker(samples, R=0.995):
    """ 1차 IIR DC 블로킹 필터 """
    b = [1, -1]  # Numerator coefficients (x[n] - x[n-1])
    a =   # Denominator coefficients (y[n] - R*y[n-1])

    # I와 Q 채널에 각각 필터 적용
    filtered_i = lfilter(b, a, np.real(samples))
    filtered_q = lfilter(b, a, np.imag(samples))

    return filtered_i + 1j * filtered_q

# SDR 초기화 (오프셋 튜닝 없이 중앙 주파수 설정)
sdr = adi.Pluto("ip:192.168.2.1")
sdr.sample_rate = int(2e6)
sdr.rx_rf_bandwidth = int(2e6)
sdr.rx_lo = int(2400e6)
sdr.rx_buffer_size = 2**12
sdr.gain_control_mode_chan0 = "manual"
sdr.rx_hardwaregain_chan0 = 70.0
sdr.rx_bbdc_tracking_en = True # 하드웨어 추적은 켜두는 것이 좋음

# 데이터 수집
raw_samples = sdr.rx()

# 디지털 DC 블로킹 필터 적용
filtered_samples = dc_blocker(raw_samples)

# FFT 결과 시각화
plt.figure(figsize=(12, 6))
plt.psd(raw_samples, NFFT=1024, Fs=sdr.sample_rate, label='Before DC Blocker')
plt.psd(filtered_samples, NFFT=1024, Fs=sdr.sample_rate, label='After DC Blocker')
plt.title("Software DC Blocking Filter Effect")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Power Spectral Density (dB/Hz)")
plt.legend()
plt.grid(True)
plt.show()

del sdr
```

검증: FFT 플롯은 필터링 전후의 스펙트럼을 비교하여 필터가 DC 스파이크를 감쇠시키는 효과를 보여줍니다. 그러나 이 방법은 DC에서 널(Null)을 생성하므로, 상당한 저주파수 성분을

가진 신호에 왜곡을 유발할 수 있다는 단점이 있습니다.[11]

AD936x 플랫폼에 대한 소프트웨어 기반 해결책, 특히 오프셋 튜닝의 보편성과 효과는 중요한 설계 철학을 시사합니다. 하드웨어는 "충분히 좋은" 수준의 기본 보정을 제공하지만, 최종적인 성능은 소프트웨어/FPGA 영역의 유연성에 달려 있다는 것입니다. 이는 플랫폼의 다용성을 높이는 동시에, 디지털 기저대역 설계자에게 더 큰 책임을 부여합니다. 즉, PlutoSDR과 같은 AD936x 기반 SDR을 사용하는 중요한 애플리케이션의 경우, 소프트웨어 기반 DC 제거 전략은 사후 처리 방식이 아닌 초기 시스템 설계의 일부로 고려되어야 합니다.

제 **3**장**: ADRV9002** 트랜시버의 고급 **DC** 오프셋 보정

이 장에서는 ADRV9002의 차세대 DC 오프셋 보정 아키텍처를 탐구하며, 세분화된 제어 방식과 프로파일 기반 구성의 특징을 중점적으로 다룹니다.

3.1. 아키텍처의 진화**:** 보정 패러다임의 전환

ADRV9002는 AD936x에서 한 단계 더 발전한 트랜시버로, 미션 크리티컬 통신과 같이 더욱 까다로운 애플리케이션을 위해 설계되었습니다.[26] 이 아키텍처는 더 진보되고 포괄적인 디지털 신호 처리 기능을 포함하며 [26], 그 핵심적인 발전 중 하나는 더욱 강력하고 구성 가능한 보정 프레임워크입니다.

3.2. ADRV9002 보정 프레임워크**:** 프로파일과 추적

AD936x와 달리 ADRV9002의 구성은 트랜시버 평가 소프트웨어(Transceiver Evaluation Software, TES)에서 생성된 "프로파일"에 크게 의존합니다.[30] 이 프로파일은 샘플링 속도, 대역폭, 그리고 실행할 보정 알고리즘을 포함한 전체 신호 경로를 정의합니다. 보정은 크게 두 가지 유형으로 나뉩니다 [31]:

- 초기 보정 **(Initial Calibrations):** 장치가 초기화되거나 새 프로파일이 로드될 때 한 번 실행되는 포괄적인 보정 세트(RF DC 오프셋, QEC, LO 누설 등 포함)입니다. 실행할 항목은 TES 프로파일에 의해 결정됩니다.
- 추적 보정 **(Tracking Calibrations):** 온도 및 기타 동적 효과로 인한 변화를 추적하고 보정하기 위해 백그라운드에서 지속적으로 실행될 수 있는 보정의 하위 집합입니다. 이

기능들은 런타임에 IIO 드라이버를 통해 활성화하거나 비활성화할 수 있습니다.

3.3. 세분화된 제어: RFDC 대 BBDC 추적 보정

ADRV9002에서 DC 오프셋 보정 기능의 가장 중요한 아키텍처 발전은 보정 루프를 두 개의 도메인으로 분리한 것입니다. pyadi-iio 라이브러리는 이러한 제어 기능을 직접 노출합니다.[32]

- 기저대역 **DC (BBDC)** 제거: 이는 bbdc_rejection_tracking_en_chanX 속성에 해당합니다. 이 루프는 ADC 이후의 디지털 기저대역에서 작동하며, 원리적으로는 AD936x의 추적 루프와 유사하지만 더욱 발전된 알고리즘을 사용합니다.
- **RF DC (RFDC)** 보정: 이는 rfdc_tracking_en_chanX 속성에 해당하며, ADRV9002의 핵심적인 추가 기능입니다. 이 루프는 아날로그/RF 도메인에서 작동하며, ADC 이전에 미세한 DC 전류나 전압을 주입하여 오프셋을 소스에서 상쇄하는 방식으로 동작할 가능성이 높습니다.[7] 이 방식은 큰 DC 성분이 ADC에 도달하여 잠재적으로 포화시키는 것을 원천적으로 방지하여 변환기의 전체 동적 범위를 보존합니다.

이러한 RFDC와 BBDC 루프의 분리는 1장에서 확인된 이득과 DC 오프셋 내성 간의 근본적인 상충 관계에 대한 직접적인 아키텍처적 해답입니다. ADC 이전에 RF 도메인에서 오프셋을 제거함으로써, ADRV9002는 ADC의 동적 범위를 큰 DC 성분에 희생시키지 않으면서도 높은 이득(즉, 높은 감도)을 유지할 수 있습니다. 기저대역 보정에만 의존하는 아키텍처와 비교할 때 이는 상당한 성능상의 이점입니다. ADC 포화 문제는 기저대역 전용 보정이 ADC가 이미 큰 DC 오프셋을 포함한 신호를 디지털화한 이후에 발생한다는 점에서 근본적인 한계를 가집니다. 만약 오프셋이 ADC를 클리핑할 만큼 크다면, 원하는 신호에 대한 정보는 손실되어 복구할 수 없습니다. ADRV9002의 RFDC 루프는 이러한 문제를 예방하는 조치로 작용하여, ADC에 입력되는 신호가 0을 중심으로 분포하도록 보장합니다. 이는 ADC의 전체 비트 깊이가 큰 DC 바이어스가 아닌 실제 관심 신호를 표현하는 데 사용됨을 의미하며, 근본적으로 더 견고한 접근 방식입니다.

3.4. 실제 구현(POC): pyadi-iio를 통한 ADRV9002 미세 조정

ADRV9002의 제어는 pyadi-iio의 adi.adrv9002 클래스를 통해 이루어집니다.[34] 다음 POC는 프로파일을 로드한 후 런타임 추적 보정을 조작하는 방법을 보여줍니다.

아래 표는 ADRV9002에서 DC 오프셋 관리를 위해 사용하는 핵심 pyadi-iio 제어 속성을 요약한 것입니다.

표 **2: ADRV9002 DC** 오프셋 관리 속성

속성	설명	타입
sdr.profile	JSON 파일로부터 장치 구성 프로파일을 로드합니다.	str (경로)
sdr.bbdc_rejection_tracking_en_chanX	RX 채널 X에 대한 기저대역 DC 제거 추적을 활성화/비활성화합니다.	bool
sdr.rfdc_tracking_en_chanX	RX 채널 X에 대한 RF DC 오프셋 추적을 활성화/비활성화합니다.	bool

POC 구현: 다음 파이썬 스크립트는 다음을 수행합니다.

1. adi.adrv9002 객체를 생성합니다.
2. TES에서 내보낸 사전 생성된 JSON 프로파일 및 스트림 파일을 로드합니다.
3. 초기에는 BBDC 및 RFDC 추적 보정을 모두 비활성화합니다.
4. 데이터를 수집하고 FFT를 플롯하여 보정되지 않은 DC 스파이크를 보여줍니다.
5. BBDC 추적만 활성화하고 개선 사항을 보여줍니다.
6. BBDC와 RFDC 추적을 모두 활성화하여 최고 수준의 보정 효과를 보여줍니다.

Python

```
# 참고: 이 코드는 ADRV9002 평가 보드와 적절한 Linux 환경이 설정되어 있다고 가정합니다.
# TES에서 생성된 'profile.json'과 'stream.bin' 파일이 필요합니다.
import adi
import matplotlib.pyplot as plt
import numpy as np

# ADRV9002 SDR 초기화 (URI는 시스템에 맞게 수정 필요)
sdr = adi.adrv9002(uri="ip:192.168.1.233")

# TES에서 생성한 프로파일 로드
# 프로파일은 샘플링 속도, 대역폭, 초기 보정 등을 정의합니다.
try:
    sdr.profile = "path/to/your/profile.json"
```

```
except Exception as e:
    print(f"Error loading profile: {e}")
    print("Please ensure profile.json and stream.bin are in the correct path.")
    del sdr
    exit()

# 수신기 설정
sdr.rx_ensm_mode_chan0 = "rf_enabled"
sdr.gain_control_mode_chan0 = "automatic"
fs = int(sdr.rx0_sample_rate)
sdr.rx_buffer_size = 2**14

# 1. 모든 추적 보정 비활성화
sdr.bbdc_rejection_tracking_en_chan0 = False
sdr.rfdc_tracking_en_chan0 = False
data_all_off = sdr.rx()

# 2. BBDC 추적만 활성화
sdr.bbdc_rejection_tracking_en_chan0 = True
sdr.rfdc_tracking_en_chan0 = False
data_bbdc_on = sdr.rx()

# 3. BBDC 및 RFDC 추적 모두 활성화
sdr.bbdc_rejection_tracking_en_chan0 = True
sdr.rfdc_tracking_en_chan0 = True
data_all_on = sdr.rx()

# FFT 결과 시각화
plt.figure(figsize=(15, 8))
plt.psd(data_all_off, NFFT=2048, Fs=fs, label='All Tracking OFF')
plt.psd(data_bbdc_on, NFFT=2048, Fs=fs, label='BBDC Tracking ON')
plt.psd(data_all_on, NFFT=2048, Fs=fs, label='BBDC & RFDC Tracking ON')

plt.title("ADRV9002 DC Offset Tracking Calibration Effects")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Power Spectral Density (dB/Hz)")
plt.legend()
plt.grid(True)
plt.show()

del sdr
```

분석: 결과 분석을 통해 높은 이득 설정에서 ADC 포화를 방지하기 위해서는 RFDC 추적을

활성화하는 것이 중요하며, BBDC 추적은 디지털 도메인에서 남은 잔여 오프셋을 정리하는 역할을 한다는 점을 설명할 수 있습니다. 대부분의 애플리케이션에서는 두 가지 추적 기능을 모두 활성화하는 것이 최상의 성능을 제공합니다.

제 **4**장**:** 비교 분석 및 모범 사례

이 마지막 장에서는 분석 결과를 종합하여 두 플랫폼을 직접 비교하고, 엔지니어를 위한 시나리오 기반 권장 사항을 제공합니다.

4.1. 정면 비교**:** 알고리즘 및 성능 차이

두 플랫폼 간의 DC 오프셋 보정 기능의 주요 차이점은 다음 표에 요약되어 있습니다. 이 표는 시스템 설계자가 각 플랫폼의 장단점과 기능을 한눈에 파악하여 애플리케이션에 적합한 하드웨어를 선택하는 데 도움을 줍니다.

표 **3: DC** 오프셋 보정 기능 비교 **(AD936x** 대 **ADRV9002)**

기능	AD9361/AD9363 (예: ADALM-PLUTO)	ADRV9002
주요 메커니즘	단일 기저대역 DC(BBDC) 추적 루프	이중 루프: RF 도메인(RFDC) 및 기저대역 도메인(BBDC)
제어 세분성	BBDC 추적을 위한 단일 On/Off 제어	채널별 RFDC 및 BBDC 추적에 대한 독립적인 On/Off 제어
구성 패러다임	API/드라이버 속성을 통한 직접적인 런타임 제어	초기 설정은 프로파일 기반(TES 사용), 추적 기능은 런타임 제어
소프트웨어 의존성	높음. 최적의 결과를 위해 소프트웨어 오프셋 튜닝 또는 DC 블로킹이 필요한	낮음. 온칩 하드웨어가 매우 효과적이며, 소프트웨어

	경우가 많음.	제어는 미세 조정용.
핵심 장점	제어의 높은 유연성과 단순성	높은 이득 시나리오에서 우수한 동적 범위 보존

ADRV9002가 TES 생성 프로파일에 의존하는 것은 엔지니어링 워크플로우의 변화를 의미합니다. 이는 시스템 구성의 상당 부분을 런타임 코드에서 GUI 도구를 사용하는 "설계 시간" 단계로 이동시킵니다. 이 접근 방식은 샘플링 속도, 필터 설정, 보정 파라미터 간의 복잡한 상호 의존성이 검증된 도구에 의해 올바르게 처리되도록 보장하여 잘못된 구성의 위험을 줄입니다. 하지만 동시에 새 프로파일을 로드하지 않고는 전체 신호 경로를 즉석에서 재구성하는 유연성은 감소합니다. 이는 성능/견고성과 런타임 유연성 사이의 절충을 나타냅니다.

4.2. 시나리오 기반 권장 사항

- 범용 실험실**,** 교육 및 신속한 프로토타이핑 **(AD936x):** 하이브리드 접근 방식을 권장합니다. 내장된 BBDC 추적을 사용하고, 견고성을 위해 소프트웨어 오프셋 튜닝을 기본 전략으로 구현합니다. 디지털 DC 블로커는 관심 신호가 DC에 있고 약간의 신호 왜곡 가능성을 수용할 수 있는 경우에만 사용합니다.
- 고성능 통신 시스템 **(**예**:** 셀룰러**,** 공공 안전**) (ADRV9002):** 전체 하드웨어 기능을 활용할 것을 권장합니다. TES를 사용하여 모든 관련 초기 보정이 활성화된 최적화된 프로파일을 생성합니다. 런타임 시에는 RFDC 및 BBDC 추적 보정을 모두 활성화하여 동적 채널 조건 및 간섭에 대한 성능과 견고성을 극대화합니다.
- **DC** 근처의 저대역폭 신호**:** AD936x의 경우, 이는 가장 어려운 시나리오이며 디지털 DC 블로커가 유일한 옵션이므로 신호에 미치는 영향을 신중하게 특성화해야 합니다. ADRV9002의 경우, 매우 효과적인 하드웨어 제거 기능만으로도 왜곡을 유발하는 소프트웨어 필터링 없이 DC 채널을 확보하기에 충분할 수 있습니다.

4.3. 검증 및 측정 방법론

DC 오프셋 제거 기술의 효과를 검증하기 위한 간단하고 반복 가능한 프로세스는 다음과 같습니다.

1. 설정**:** 외부 신호 유입이 없도록 SDR의 수신기 입력을 50옴 부하로 종단합니다. 수신기 이득을 적당히 높은 수준으로 설정합니다.
2. 수집**:** I/Q 샘플 블록을 수집합니다.

3. 분석: FFT를 사용하여 전력 스펙트럼 밀도(PSD)를 계산하고 플롯합니다.
4. 정량화: 0 Hz에서의 스파이크 크기를 잡음 플로어 대비 측정합니다. 성공적인 제거는 이 스파이크를 잡음 플로어 수준까지 현저히 감소시킵니다.
5. 반복: 보정 기법(하드웨어 또는 소프트웨어)을 적용하고 2-4단계를 반복하여 개선 정도를 정량화합니다. 이는 성능을 평가하는 명확하고 데이터 기반의 방법을 제공합니다.

결론

본 보고서는 AD936x 및 ADRV9002 아키텍처를 기반으로 하는 직접 변환 SDR에서 DC 오프셋을 최적화하기 위한 포괄적인 분석과 실질적인 구현 가이드를 제공했습니다. 분석 결과, DC 오프셋은 직접 변환 아키텍처의 고유한 특성이며, 그 완화 전략은 SDR의 아키텍처와 목표 애플리케이션의 요구 사항에 따라 크게 달라진다는 점이 명확해졌습니다.

AD936x 기반 플랫폼(예: ADALM-PLUTO)은 온칩 BBDC 추적 기능과 소프트웨어 기법(오프셋 튜닝, DC 블로킹)을 결합한 하이브리드 접근 방식을 통해 높은 유연성과 비용 효율성을 제공합니다. 이 방식은 소프트웨어 개발자에게 더 많은 제어 권한과 책임을 부여하며, 다양한 시나리오에 맞게 성능을 조정할 수 있게 합니다.

반면, ADRV9002는 RFDC와 BBDC 추적을 결합한 이중 루프 보정 아키텍처를 통해 한 단계 더 진보된 솔루션을 제공합니다. 이 구조는 ADC 포화를 원천적으로 방지하여 높은 이득이 요구되는 까다로운 통신 환경에서도 탁월한 동적 범위를 유지합니다. 프로파일 기반 구성은 복잡한 시스템 설정을 단순화하고 안정성을 보장하지만, 런타임 유연성 측면에서는 일부 제약이 따릅니다.

결론적으로, SDR 및 DC 오프셋 완화 전략의 선택은 애플리케이션의 특정 성능 요구 사항에 따라 신중하게 이루어져야 하는 중요한 시스템 설계 결정입니다. ADRV9002는 고성능이 필수적인 시스템에 우수한 즉시 사용 가능한 성능을 제공하는 반면, AD936x는 소프트웨어를 활용하여 높은 성능을 달성할 수 있는 유연하고 접근성 높은 플랫폼으로 자리매김하고 있습니다. 엔지니어는 각 아키텍처의 근본적인 차이점을 이해함으로써 주어진 요구 사항에 가장 적합한 최적의 솔루션을 구현할 수 있을 것입니다.

참고 자료

1. Direct-conversion receiver - Wikipedia, 9월 21, 2025에 액세스, https://en.wikipedia.org/wiki/Direct-conversion_receiver
2. An adaptive offset cancellation mixer for direct conversion receivers in 2.4GHZ CMOS - College of Engineering | Oregon State University, 9월 21, 2025에 액세스, https://web.engr.oregonstate.edu/~moon/research/files/iscas00_mixer.pdf
3. IQ imbalance - Wikipedia, 9월 21, 2025에 액세스, https://en.wikipedia.org/wiki/IQ_imbalance
4. Direct Conversion (Zero-IF) Receiver | Wireless Pi, 9월 21, 2025에 액세스,

https://wirelesspi.com/direct-conversion-zero-if-receiver/
5. DC offsets in direct-conversion receivers: Characterization and implications - ResearchGate, 9월 21, 2025에 액세스, https://www.researchgate.net/publication/3427295_DC_offsets_in_direct-conversion_receivers_Characterization_and_implications
6. Illustration of the DC offset problem in direct-conversion receiver. - ResearchGate, 9월 21, 2025에 액세스, https://www.researchgate.net/figure/Illustration-of-the-DC-offset-problem-in-direct-conversion-receiver_fig1_2646342
7. Transmit LO Leakage (LOL)—An Issue of Zero-IF That Isn't Making People Laugh Out Loud, 9월 21, 2025에 액세스, https://www.analog.com/en/resources/analog-dialogue/articles/transmit-lo-leakage-lol-an-issue-of-zero-if-that-isn-t-making-people-laugh-out-loud.html
8. Zero IF demodulator DC offset compensation circuit - EEVblog, 9월 21, 2025에 액세스, https://www.eevblog.com/forum/rf-microwave/zero-if-demodulator-dc-offset-compensation-circuit/
9. Correcting Frequency Offset on ADALM PlutoSDR - Software - Libre Space Community, 9월 21, 2025에 액세스, https://community.libre.space/t/correcting-frequency-offset-on-adalm-plutosdr/10717
10. Bought a plutosdr and I have a static line in the waterfall in sdr# and some other issues / questions... : r/RTLSDR - Reddit, 9월 21, 2025에 액세스, https://www.reddit.com/r/RTLSDR/comments/93rewt/bought_a_plutosdr_and_i_have_a_static_line_in_the/
11. Removing that Center Frequency DC Spike in Gnuradio the Easy Way - RTL-SDR.com, 9월 21, 2025에 액세스, https://www.rtl-sdr.com/removing-that-center-frequency-dc-spike-in-gnuradio-the-easy-way/
12. AD9361 DC offset calibration - Q&A - EngineerZone - Analog Devices, 9월 21, 2025에 액세스, https://ez.analog.com/wide-band-rf-transceivers/design-support/f/q-a/80508/ad9361-dc-offset-calibration
13. Lab 2 - Getting Started on PlutoSDR - Codinghub, 9월 21, 2025에 액세스, https://codinghub.sellfy.store/p/lab-2-getting-started-on-plutosdr/
14. FMCOMMS2/3/4 HDL Project, 9월 21, 2025에 액세스, https://analogdevicesinc.github.io/hdl/projects/fmcomms2/index.html
15. PlutoSDR in Python | PySDR: A Guide to SDR and DSP using Python, 9월 21, 2025에 액세스, https://pysdr.org/content/pluto.html
16. AD9361 Reference Manual (Rev. A) - Farnell, 9월 21, 2025에 액세스, https://www.farnell.com/datasheets/2007082.pdf
17. AD9361 Reference Manual | UG-570 - Analog Devices, 9월 21, 2025에 액세스, https://ez.analog.com/cfs-file/__key/telligent-evolution-components-attachments/00-441-00-00-00-07-91-97/AD9361_5F00_Reference_5F00_Manual_5F00_UG_2D00_570.pdf

18. analogdevicesinc/pyadi-iio: Python interfaces for ADI ... - GitHub, 9월 21, 2025에 액세스, https://github.com/analogdevicesinc/pyadi-iio
19. pyadi-iio - PyPI, 9월 21, 2025에 액세스, https://pypi.org/project/pyadi-iio/
20. Examples — Analog Devices Hardware Python Interfaces 0.0.19 documentation, 9월 21, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/guides/examples.html
21. how to eliminate the center peak of HAckRF in SDRSHARP - RadioReference.com Forums, 9월 21, 2025에 액세스, https://forums.radioreference.com/threads/how-to-eliminate-the-center-peak-of-hackrf-in-sdrsharp.441554/
22. SDRSharp Users Guide - RTL-SDR.com, 9월 21, 2025에 액세스, https://www.rtl-sdr.com/sdrsharp-users-guide/
23. UHD / USRP Support DC removal · Issue #1190 · AlexandreRouma/SDRPlusPlus - GitHub, 9월 21, 2025에 액세스, https://github.com/AlexandreRouma/SDRPlusPlus/issues/1190
24. Digital Music Programming 2: DC Blocking Filter, 9월 21, 2025에 액세스, https://peabody.sapp.org/class/dmp2/lab/dcblock/
25. DC Blocker | Introduction to Digital Filters - DSPRelated.com, 9월 21, 2025에 액세스, https://www.dsprelated.com/freebooks/filters/DC_Blocker.html
26. ADRV9002 Dual Narrow/Wideband RF Transceiver - ADI - Mouser Electronics, 9월 21, 2025에 액세스, https://www.mouser.com/new/analog-devices/adi-adrv9002-rf-transciever/
27. Data Sheet - ADRV9002 - Analog Devices, 9월 21, 2025에 액세스, https://www.analog.com/media/en/technical-documentation/data-sheets/adrv9002.pdf
28. Data Sheet - ADRV9002 - Mouser Electronics, 9월 21, 2025에 액세스, https://www.mouser.com/datasheet/2/609/adrv9002-1894174.pdf
29. adrv9002 Device Driver - ADI Developer, 9월 21, 2025에 액세스, https://developer.analog.com/software/drivers/linux/adrv9002
30. ADRV9002 + ZCU102 + Pyadi iio + RX Testing - Q&A - Software Interface Tools, 9월 21, 2025에 액세스, https://ez.analog.com/sw-interface-tools/f/q-a/539809/adrv9002-zcu102-pyadi-iio-rx-testing
31. How to configurate ADRV9002 initial calibrations by IIO drivers - EngineerZone, 9월 21, 2025에 액세스, https://ez.analog.com/sw-interface-tools/f/q-a/591793/how-to-configurate-adrv9002-initial-calibrations-by-iio-drivers
32. adrv9002 — Analog Devices Hardware Python Interfaces 0.0.19 ..., 9월 21, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/devices/adi.adrv9002.html
33. LO leakage for zero-IF transmitters : r/rfelectronics - Reddit, 9월 21, 2025에 액세스, https://www.reddit.com/r/rfelectronics/comments/109atxe/lo_leakage_for_zeroif_transmitters/
34. Analog Devices Hardware Python Interfaces 0.0.19 documentation, 9월 21, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/

35. Analog Devices Hardware Python Interfaces — Analog Devices Hardware Python Interfaces 0.0.3 documentation, 9월 21, 2025에 액세스, https://pyadi-iio.readthedocs.io/en/latest/