ZCU208 RFSoC 기반 SDR 데모 구현을 위한 5단계 기술 가이드

서론

개요

AMD Zynq UltraScale+ RFSoC는 RF(Radio Frequency) 신호 체인, 디지털 처리 로직, 애플리케이션 처리 장치를 단일 실리콘 다이에 통합하여 기술 패러다임의 전환을 이룬 혁신적인 솔루션이다.[1] 이 통합 아키텍처는 기존의 다중 칩 솔루션에서 필수적이었던 복잡하고 전력 소모가 큰 JESD204 인터페이스를 제거하여 시스템 전체의 전력 효율을 높이고 설계 복잡성을 획기적으로 낮춘다.[3] 본 가이드에서 다루는 ZCU208 평가 키트는 플래그십 모델인 ZU48DR 디바이스를 탑재하고 있으며, 5G 무선 통신, 위상 배열 레이더, 첨단 테스트 및 계측 장비와 같은 차세대 SDR(Software Defined Radio) 애플리케이션의 신속한 프로토타이핑 및 개발을 위한 이상적인 플랫폼이다.[1]

보고서의 목적 및 구조

본 보고서의 목적은 ZCU208 평가 보드를 사용하여 점진적으로 복잡성이 증가하는 5단계의 SDR 데모를 성공적으로 구현하기 위한 포괄적인 기술 지침을 제공하는 것이다. 이 가이드는 사용자가 기초적인 기술을 먼저 습득하고 이를 바탕으로 더 복잡한 단계로 나아갈 수 있도록 체계적으로 구성되었다. 이를 통해 개발 과정에서 발생할 수 있는 위험을 최소화하고 각 단계의 성공 가능성을 극대화한다. 본 가이드에서는 개발 프레임워크로 PYNQ를 채택하였다. PYNQ는 프로그래머블 로직(PL)의 실시간 성능과 Python 및 Jupyter 노트북의 신속하고 반복적인 개발 사이클을 결합하여, 복잡한 SDR 시스템의 프로토타이핑 및 시각화에 최적화된 환경을 제공한다.[6]

5단계 데모 계획 요약

아래 표는 본 가이드에서 다룰 5단계 데모의 전체 로드맵을 요약한 것이다. 각 단계의 목표, 사용 도구, 그리고 핵심 학습 성과를 명확히 제시하여 사용자가 전체 학습 과정을 조망할 수 있도록 돕는다.

단계	제목	주요 도구/프레임워크	핵심 학습 성과
1	초기 RF 무결성 검증	RF Data Converter Evaluation Tool	코드 작성 없이 RF-ADC/DAC 신호 경로의 물리적 무결성 검증
2	PYNQ 환경 및 프로그래밍 기반 RF I/O	PYNQ 및 xrfdc 라이브러리	Python을 사용하여 RF 데이터 컨버터를 프로그래밍 방식으로 제어
3	실시간 스펙트럼 분석기	PYNQ 및 rfsoc_sam 오버레이	RF 환경 분석을 위한 수신 전용 SDR 애플리케이션 구현
4	완전한 디지털 통신 시스템: QPSK	PYNQ 및 rfsoc_radio 오버레이	디지털 변조, 복조, 시각화를 포함한 완전한 송수신기 구축
5	고급 파형: OFDM 송수신기	PYNQ 및 rfsoc_ofdm 오버레이	5G 및 Wi-Fi에 사용되는 현대적이고 스펙트럼 효율적인 파형 구현

섹션 **1:** 플랫폼 기초 및 초기 검증

1.1. ZCU208 하드웨어 아키텍처 및 성능

ZCU208 평가 보드는 ZU48DR RFSoC 디바이스를 중심으로 설계된 고성능 플랫폼이다.[1] 이 디바이스의 가장 큰 특징은 단일 칩 아키텍처로, 기존의 분리된 데이터 컨버터 솔루션에서 요구되던 복잡하고 전력 소모가 많은 JESD204 인터페이스의 필요성을 제거한다. 이를 통해 시스템 전력 소모를 줄이고 PCB 설계를 단순화하여 전체 시스템의 효율성과 안정성을 높인다.[3]

RF 데이터 컨버터

ZCU208에 탑재된 ZU48DR은 8개의 14비트, 5 GSPS RF-ADC와 8개의 14비트, 10 GSPS RF-DAC를 내장하고 있다. 이러한 직접 RF 샘플링(Direct RF Sampling) 기능은 아날로그 신호를 RF 대역에서 직접 디지털로 변환하거나 그 반대의 과정을 가능하게 하여, 신호 처리가 디지털 영역에서 이루어지도록 한다. 이는 기존의 아날로그 믹서나 필터 없이도 소프트웨어적으로 주파수 변환 및 필터링을 수행할 수 있게 하여 시스템에 전례 없는 유연성을 제공한다.[1]

처리 서브시스템

RFSoC는 다양한 연산 자원을 통합한 이기종 컴퓨팅 플랫폼이다.

- 애플리케이션 처리 장치 **(APU):** 쿼드코어 Arm Cortex-A53 프로세서는 Linux와 같은 고수준 운영체제를 실행하며, PYNQ 프레임워크를 포함한 복잡한 애플리케이션을 처리한다.
- 실시간 처리 장치 **(RPU):** 듀얼코어 Arm Cortex-R5 프로세서는 시스템 제어와 같이 결정성(deterministic)이 요구되는 실시간 작업을 담당한다.
- 프로그래머블 로직 **(PL):** 930K의 시스템 로직 셀과 4,272개의 DSP 슬라이스를 갖춘 PL은 대규모 병렬 신호 처리에 최적화되어 있다. FFT, 필터링, 변복조와 같은 연산 집약적인 알고리즘을 하드웨어 가속하여 실시간 성능을 보장한다.[4]

ZCU208 주요 하드웨어 사양

아래 표는 ZCU208 보드의 핵심 하드웨어 사양을 정리한 것이다. 이 정보는 이후 진행될 SDR 데모의 성능을 이해하고, 향후 맞춤형 설계를 위한 중요한 기준이 된다. 예를 들어, 10 GSPS의 최대 DAC 샘플링 속도는 생성 가능한 신호의 최대 대역폭을 결정하며, 4,272개의 DSP 슬라이스는 PL에서 구현할 수 있는 신호 처리 알고리즘의 복잡도를 제한한다.[1]

구성 요소	사양
RFSoC 디바이스	Zynq UltraScale+ RFSoC Gen 3 ZU48DR
RF-ADC	8 채널, 14비트 해상도, 최대 5 GSPS
RF-DAC	8 채널, 14비트 해상도, 최대 10 GSPS
SD-FEC 코어	8개 내장 코어
시스템 로직 셀 (K)	930
DSP 슬라이스	4,272
PS 메모리 (DDR4 SODIMM)	4 GB, 64비트
PL 메모리 (DDR4 Component)	8 GB, 64비트

1.2. 개발 환경 개요

ZCU208을 위한 개발 방법론은 크게 두 가지로 나뉜다.

- **Vivado/Vitis** 흐름**:** 이는 전통적인 하드웨어 중심 접근 방식으로, HDL, C/C++, Tcl 스크립팅을 사용하여 최대의 성능과 제어 기능을 구현한다. 이 방법은 주로 최종 상용 제품 개발에 사용된다.[4]
- **PYNQ** 프레임워크**:** PYNQ는 Linux 기반의 Python 중심 접근 방식으로, Jupyter 노트북을 통해 신속한 프로토타이핑을 가능하게 한다. "오버레이(Overlay)"라 불리는 사전 컴파일된 비트스트림과 Python API를 통해 하드웨어의 복잡성을 추상화한다.[6] 본 가이드는 빠른 데모 구현과 상호작용적인 시각화라는 목표에 부합하기 때문에 PYNQ 프레임워크를 채택하였다.

1.3. 사전 준비 사항 및 초기 설정

데모를 진행하기 위해 필요한 하드웨어 및 소프트웨어 목록은 다음과 같다.

- 하드웨어**:** ZCU208 평가 보드, XM655 브레이크아웃 카드, CLK104 RF 클록 카드, 전원 공급 장치, Micro USB 케이블, 이더넷 케이블, SMA 케이블, 그리고 3단계에서 사용할 광대역 안테나.[1]
- 소프트웨어**:** 호스트 컴퓨터(Windows/Linux), 최소 16GB 용량의 MicroSD 카드, SD 카드 포맷 소프트웨어, 터미널 에뮬레이터(예: Tera Term, PuTTY), 그리고 각 단계별로 필요한 소프트웨어 패키지(RFDC 평가 도구 ZIP 파일, PYNQ 이미지).[4]

섹션 **2: 1**단계 **-** 초기 **RF** 무결성 검증

이 첫 번째 단계는 단순한 기능 테스트를 넘어, 전체 개발 과정의 위험을 관리하는 핵심적인 전략이다. 개발 과정에는 두 가지 주요 소프트웨어 생태계가 존재한다. 하나는 AMD/Xilinx에서 공식적으로 제공하는 RFDC Evaluation Tool이고, 다른 하나는 오픈 소스 기반의 PYNQ 프레임워크이다.[6] 초심자는 이 둘을 경쟁적인 선택지로 볼 수 있지만, 실제로는 상호 보완적인 관계에 있다. RFDC 평가 도구는 하드웨어 검증이라는 단일 목적을 위해 설계된, 사전 컴파일된 폐쇄형 시스템이다. 반면 PYNQ는 유연하고 개방적인 개발 환경이다.

따라서 공식 RFDC 평가 도구를 먼저 사용하여 "정상 작동이 확인된 기준(known good baseline)"을 설정하는 것이 매우 중요하다. 만약 보드 제조사가 특별히 설계한 이 공식 도구를 사용한 단순 루프백 테스트가 실패한다면, 이는 케이블 불량, 커넥터 손상, 보드 자체의 결함과 같은 하드웨어 문제를 강력하게 시사한다. 이 접근법은 PYNQ와 같은 복잡한 개발 환경을 도입하기 전에 하드웨어 문제를 미리 분리해낼 수 있게 해준다. 이러한 체계적인 접근은 PYNQ 환경에서 발생할 수 있는 디버깅의 막다른 길을 단순한 하드웨어 점검으로 전환시켜, 개발자의 시간과 노력을 크게 절약해준다. 결과적으로, 이 단계를 성공적으로 완료하면 사용자는 하드웨어가 정상적으로 작동한다는 확신을 가질 수 있으며, 이후 PYNQ에서 발생하는 모든 문제는 소프트웨어나 구성(configuration)과 관련된 것으로 간주하고 접근할 수 있다.

2.1. 평가를 위한 하드웨어 조립

정확한 하드웨어 조립은 성공적인 테스트의 첫걸음이다. 아래의 지침에 따라 각 부품을

신중하게 연결한다.

- ZCU208 메인 보드에 CLK104 클록 모듈과 XM655 브레이크아웃 카드를 부착한다.[11]
- 부팅 모드 DIP 스위치(SW2)를 SD 부팅 모드로 설정한다. 이는 매우 중요하며 종종 간과되는 단계이다.[13]
- 물리적 루프백을 연결한다. 제공된 SMA 케이블을 사용하여 XM655 카드의 특정 DAC 출력(예: DAC Tile 0, Ch 0)을 특정 ADC 입력(예: ADC Tile 0, Ch 0)에 연결한다.[13]

2.2. 평가 이미지로 부팅

- AMD/Xilinx 웹사이트에서 "RF Evaluation Tool and Board Setup" 패키지를 다운로드하여 압축을 해제한다.[13]
- MicroSD 카드를 FAT32로 포맷한 후, 압축 해제된 폴더 내의 image/208 디렉터리의 모든 내용을 카드에 복사한다.[13]
- 전원, 이더넷, 그리고 시리얼 콘솔용 Micro USB 케이블을 연결한다. 보드 전원을 켜고 시리얼 터미널(전송 속도 115200)을 통해 부팅 순서를 모니터링하여 Linux가 성공적으로 부팅되었는지 확인한다. 부팅 과정은 약 60초 정도 소요될 수 있다.[11]

2.3. DAC-ADC 루프백 테스트 실행

- 호스트 **PC** 네트워크 설정**:** 이는 매우 중요한 단계이다. 호스트 PC의 이더넷 어댑터에 보드의 기본 IP 주소와 동일한 서브넷의 고정 IP 주소를 설정해야 한다. 예를 들어, 보드 IP가 169.254.10.2라면 호스트 PC의 IP는 169.254.10.1로 설정한다.[13]
- **GUI** 실행**:** 호스트 PC에서 RF_DC_Evaluation_UI.exe를 실행한다. 애플리케이션은 자동으로 보드에 연결을 시도한다.[13]
- 신호 생성**:** GUI에서 DAC Tile 0을 탐색하여 DAC 0을 선택하고, 단일 톤 사인파(예: 100 MHz)를 설정한 후 "Generate" 버튼을 클릭한다.[13]
- 신호 수집 및 검증**:** ADC Tile 0으로 이동하여 ADC 0을 선택하고 "Acquire" 버튼을 클릭한다. GUI는 수집된 신호의 FFT(고속 푸리에 변환) 결과를 표시한다. 예상되는 결과는 100 MHz에서 뚜렷한 단일 피크이며, 이는 DAC에서 ADC까지의 전체 아날로그 신호 경로가 정상적으로 작동함을 확인시켜 준다.[13]

섹션 **3: 2**단계 **- PYNQ** 환경 및 프로그래밍 기반 **RF I/O**

PYNQ는 단순한 Python 라이브러리를 넘어, RFSoC의 복잡한 하드웨어를 추상화하는 정교한 하드웨어 추상화 계층(HAL, Hardware Abstraction Layer)으로 기능한다. PYNQ는 xrfdc와 같은 Python 라이브러리를 통해 하드웨어와 상호작용하며, 이 라이브러리는 저수준 드라이버를 감싼 Python 래퍼(wrapper)이다.[18] Jupyter 노트북 환경은 이 라이브러리에 대한 상호작용적인 인터페이스를 제공한다.[12] 이는 하드웨어(RFDC) → 드라이버 → Python API (

xrfdc) → 사용자 애플리케이션(Jupyter 노트북)으로 이어지는 계층적 아키텍처를 형성한다.

이 단계의 "Hello, World" 데모는 이러한 계층적 개념을 명시적으로 학습시키기 위해 설계되었다. 사용자는 xrfdc 객체를 직접 조작함으로써 단순히 스크립트를 실행하는 것을 넘어, RF 하드웨어를 위한 잘 정의된 HAL과 상호작용하는 경험을 하게 된다. 이 계층적 추상화에 대한 이해는 RFSoC에서 PYNQ를 능숙하게 사용하기 위한 기본이다. 이는 하드웨어 기능이 Python과 같은 고수준 언어에 어떻게 노출되는지를 보여준다. 이 지식은 사용자가 나중에 xrfdc 라이브러리의 소스 코드를 탐색하거나 프로그래머블 로직에 자신만의 맞춤형 IP를 위한 Python 드라이버를 생성하는 데까지 확장될 수 있는 기반이 된다. 따라서 이 단계는 단순한 사용법 안내를 넘어, 임베디드 시스템 아키텍처에 대한 교육적 도구로서의 가치를 지닌다.

3.1. PYNQ 이미지 준비 및 보드 부팅

- PYNQ.io 웹사이트에서 ZCU208용 공식 PYNQ 이미지(예: v3.0.1 이상)를 다운로드한다. 특정 데모는 특정 PYNQ 버전을 요구할 수 있음에 유의해야 한다.[6]
- BalenaEtcher나 Raspberry Pi Imager와 같은 도구를 사용하여 다운로드한 .iso 이미지를 SD 카드에 기록한다. 이는 1단계의 파일 복사 방식과는 다르다.
- SW2 부팅 스위치가 SD 부팅 모드(ON, OFF, OFF, OFF)로 올바르게 설정되었는지 확인한다.[16]

3.2. 연결 설정

- 방법 **1 (**라우터**/DHCP):** ZCU208과 호스트 PC를 동일한 네트워크 라우터에 연결한다. 보드는 DHCP를 통해 자동으로 IP 주소를 할당받는다. 이 IP 주소는 라우터의 관리 페이지나, 시리얼 콘솔에 연결하여 ifconfig 명령을 실행함으로써 확인할 수 있다.[12]
- 방법 **2 (**직접 연결**):** ZCU208을 호스트 PC의 이더넷 포트에 직접 연결한다. 이 경우, PYNQ의 기본 고정 IP(192.168.2.99)와 통신하기 위해 호스트 PC에 고정 IP(예: 192.168.2.1)를 설정해야 한다.[12]
- **Jupyter Lab** 접속**:** 웹 브라우저를 열고 http://<board_ip_address>:9090 또는 http://pynq:9090으로 접속한다. 기본 비밀번호는 'xilinx'이다.[12]

3.3. Jupyter 및 xrfdc 라이브러리 첫걸음

- Jupyter Lab 인터페이스를 간략히 둘러본다: 새 노트북 생성, 셀 실행, 터미널 접속 등.
- 노트북에서 기본 오버레이를 로드하고 RF 데이터 컨버터 객체를 인스턴스화한다: from pynq.overlays.base import BaseOverlay; import xrfdc; base = BaseOverlay('base.bit'); rfdc = base.rfdc.[22]
- rfdc 객체를 사용하여 ADC 및 DAC 타일의 상태를 프로그래밍 방식으로 조회함으로써 하드웨어와의 실시간 상호작용을 확인한다.

3.4. RFSoC를 위한 "Hello, World"

아래의 단계에 따라 처음부터 Jupyter 노트북을 작성하여 RF 루프백을 프로그래밍 방식으로 검증한다.

- 필수 라이브러리 임포트: numpy, matplotlib.pyplot, pynq를 임포트한다.
- **DAC** 설정: xrfdc 메서드를 사용하여 DAC 타일과 채널을 선택하고, 샘플링 속도를 설정한다. 내장된 NCO(Numerically Controlled Oscillator)를 구성하여 단일 톤 사인파(예: 200 MHz)를 생성한다. 이는 프로그래밍 방식의 직접적인 신호 생성을 보여준다.[22]
- **ADC** 설정: xrfdc를 사용하여 ADC 타일과 채널을 선택하고 샘플링 속도를 설정한다.
- 데이터 캡처: 기본 오버레이에 포함된 PYNQ DMA 블록을 사용하여 ADC로부터 일정량의 샘플 블록을 ZCU208의 DDR4 메모리로 캡처한다.[24]
- 결과 시각화: 캡처된 데이터(numpy 배열 형태)를 PYNQ 버퍼에서 가져와 matplotlib을 사용하여 시간 영역 파형을 그린다. 플롯에 뚜렷한 사인파가 나타나면 RF 루프백이 프로그래밍 방식으로 성공적으로 검증된 것이다.

섹션 4: 3단계 - 실시간 스펙트럼 분석기 구현

이 단계는 단순한 데모를 넘어, 고가의 전문 계측 장비 기능을 재구성 가능한 플랫폼에서 구현하는 패러다임의 전환을 보여준다. 전통적인 벤치탑 스펙트럼 분석기는 매우 비싸고 특화된 장비이다. 하지만 ZCU208의 최대 5 GSPS에 달하는 직접 RF 샘플링 능력과 rfsoc_sam이라는 오픈 소스 PYNQ 오버레이를 결합하면, 이 전문 장비의 핵심 기능을 저렴한 개발 보드 위에서 재현할 수 있다.[4] 이는 RFSoC가 단순한 SDR 부품이 아니라, 그 자체로 완결된 "칩 위의 RF 계측기(RF instrument on a chip)"임을 의미한다. PYNQ 프레임워크는

HDL(하드웨어 기술 언어) 전문가가 아니더라도 이러한 강력한 기능을 쉽게 활용할 수 있도록 진입 장벽을 낮춘다.

이러한 능력은 연구소나 기업이 기존 상용 장비보다 훨씬 저렴한 비용으로 맞춤형 고성능 테스트 및 계측 시스템을 구축할 수 있게 한다. 또한, 라디오가 실시간으로 주변 환경을 감지하고 통신 방식을 조절해야 하는 인지 라디오(cognitive radio)와 같은 새로운 애플리케이션의 개발을 촉진한다.[27] 휴대폰의 셀룰러 신호를 실시간으로 탐지하는 영상 [29]은 이러한 가능성을 보여주는 강력한 실제 사례이다. 이 단계를 통해 사용자는 단순히 개발 보드 사용법을 배우는 것을 넘어, 자신만의 RF 계측기를 구축할 수 있는 기술을 습득하게 된다.

4.1. 스펙트럼 분석기 오버레이 설치

- 전제 조건**:** ZCU208 보드가 인터넷에 연결되어 있어야 한다. 이는 라우터를 통하거나 호스트 PC의 인터넷 연결을 공유하여 설정할 수 있으며, 이 단계에서 가장 흔한 실패 원인이므로 반드시 확인해야 한다.[16]
- Jupyter Lab 내에서 터미널을 열고 pip 설치 명령을 실행한다: pip3 install git+https://github.com/strath-sdr/rfsoc_sam.[19]
- 이 명령은 Python 라이브러리, 하드웨어 오버레이 파일(.bit, .hwh), 그리고 예제 Jupyter 노트북을 PYNQ 환경으로 다운로드 및 설치한다.

4.2. RF 수신을 위한 하드웨어 구성

- 2단계에서 사용했던 루프백 케이블을 분리한다.
- XM655 카드의 RF-ADC 입력 중 하나에 광대역 안테나를 연결한다. 어떤 ADC 입력과 발룬(balun)을 선택하느냐에 따라 수신 가능한 주파수 범위가 결정된다. 예를 들어, FM 라디오 방송 대역(88-108 MHz)을 수신하려면 저주파수 발룬을 사용해야 한다.[13]

4.3. RF 스펙트럼 시각화

- Jupyter 작업 공간에 새로 생성된 spectrum-analyzer 폴더로 이동하여 메인 노트북을 연다.[19]
- 노트북의 셀을 순서대로 실행한다. 코드는 rfsoc_sam 오버레이를 로드하고 노트북 내에 상호작용적인 GUI를 표시한다.
- GUI 위젯을 사용하여 중심 주파수와 스팬(span)을 설정한다. 가이드에 따라 지역 FM 방송

대역(88-108 MHz) [31]이나 셀룰러 대역 [29]으로 튜닝하여 공기 중의 실제 신호를 확인한다. 출력 결과는 주파수에 따른 신호의 세기를 실시간으로 보여주는 플롯으로, 완전한 SDR 수신기가 성공적으로 구현되었음을 증명한다.

섹션 **5: 4**단계 **-** 완전한 디지털 통신 시스템**: QPSK** 송수신기

디지털 송수신기는 다양한 작업을 포함하며, 이를 효율적으로 처리하기 위해서는 하드웨어와 소프트웨어 간의 역할 분담이 중요하다. 일부 작업은 펄스 쉐이핑이나 심볼 동기화처럼 고속의 반복적인 연산이 필요하고, 다른 작업은 변조 차수 설정이나 성상도(constellation) 플롯처럼 저속의 제어 및 시각화가 필요하다. rfsoc_radio 프로젝트 [20]와 관련 문헌 [32]은 RFSoC 플랫폼에서 이러한 문제를 해결하기 위한 최적의 분할 방식을 보여준다. 즉, 고속의 물리 계층(PHY) 처리는 실시간 성능을 보장하기 위해 프로그래머블 로직(PL)에 구현하고, 상위 계층인 MAC(Media Access Control) 및 애플리케이션 계층, 그리고 사용자 제어 및 시각화는 처리 시스템(PS)의 Python에서 실행한다.

이 단계에서는 이러한 분할 원칙을 명시적으로 분석한다. GitHub 저장소에 있는 Vivado 블록 다이어그램을 통해 하드웨어 IP 코어들을 확인하고, Jupyter 노트북의 Python 코드를 통해 제어 로직을 살펴볼 것이다. 이를 통해 사용자는 Zynq 플랫폼의 핵심적인 하드웨어/소프트웨어 공동 설계 원칙을 학습하게 된다. 즉, PL은 고속 병렬 처리에, PS는 유연한 제어, 복잡한 의사 결정, 사용자 인터페이스에 각각의 장점을 활용하는 것이다. 이는 모든 고급 Zynq 및 RFSoC 개발자에게 필수적인 핵심 개념이다.

5.1. QPSK 변조의 원리

- 직교 위상 편이 변조(QPSK, Quadrature Phase-Shift Keying)는 반송파의 위상을 4가지 상태로 변경하여 심볼당 2비트의 데이터를 인코딩하는 디지털 변조 방식이다.
- 성상도(Constellation Diagram)는 디지털 변조 신호의 품질을 시각화하는 핵심 도구이다. 이상적인 QPSK 성상도는 4개의 뚜렷하고 밀집된 점으로 나타난다.

5.2. rfsoc_radio 오버레이 설치

- 3단계와 동일한 절차를 따른다. Jupyter 터미널을 열고 Strathclyde SDR GitHub

저장소에서 rfsoc_radio 패키지를 pip을 사용하여 설치한다.[20]

5.3. 하드웨어-소프트웨어 분할 분석

- 하드웨어 **(PL):** 프로젝트의 GitHub 페이지에서 블록 다이어그램을 확인한다. 송신단의 펄스 쉐이핑 필터, 수신단의 정합 필터, 그리고 반송파/타이밍 동기화 루프와 같은 핵심 IP 블록들을 식별한다. 이러한 블록들은 높은 처리량과 낮은 지연 시간을 위해 HDL로 구현되어 있다.[20]
- 소프트웨어 **(PS):** 예제 노트북의 Python 코드 일부를 분석한다. Python이 어떻게 임의의 데이터 페이로드를 생성하고, 하드웨어 IP를 설정하며(예: 루프백 모드 설정), DMA를 트리거하여 데이터를 송수신하고, matplotlib이나 plotly를 사용하여 수신된 성상도를 렌더링하는지 보여준다.[20]

5.4. QPSK 루프백 데모 실행

- 2단계에서와 같이 DAC-ADC 간 물리적 루프백 케이블을 연결한다.
- rfsoc_radio 폴더로 이동하여 QPSK 예제 노트북을 실행한다.
- 노트북은 데이터 생성, 변조, DAC를 통한 전송, ADC를 통한 수신, 복조, 그리고 결과 표시까지 완전한 송수신 과정을 수행한다.
- 핵심 출력은 성상도 플롯이다. 4개의 뚜렷한 점 군집이 나타나는지 확인하여 성공적인 송수신을 검증한다.[18]

섹션 6: 5단계 - 고급 파형: OFDM 송수신기

전통적인 HDL 기반 개발 방식에서 물리 계층(PHY)의 파라미터(예: 부반송파의 QAM 차수)를 변경하려면 HDL 코드를 수정하고, 전체 디자인을 다시 합성 및 구현해야 했다. 이 과정은 수 시간이 소요될 수 있다. 그러나 rfsoc_ofdm PYNQ 데모[26]는 혁신적인 워크플로우를 제시한다. 핵심적인 IFFT/FFT 및 데이터 경로는 PL에 구현되어 하드웨어 가속을 받지만, 주요 제어 레지스터들은 PS에 노출된다. Jupyter 노트북은 슬라이더나 드롭다운 메뉴와 같은 상호작용적인 위젯을 사용하여 이 레지스터들을 실시간으로 수정한다.

이러한 접근 방식은 RFSoC를 정적인 하드웨어 구현체에서 동적이고 상호작용적인 "물리 계층 실험실"로 탈바꿈시킨다. 연구자는 이제 다양한 변조 방식, 파일럿 톤, 또는 순환 전치(Cyclic Prefix) 길이를 실험하고 그 효과를 수신된 성상도나 스펙트럼에서 즉시 확인할 수 있다. 이는

무선 통신 분야의 연구 개발 속도를 극적으로 가속화한다. 이 마지막 단계는 PYNQ와 RFSoC 조합의 궁극적인 힘, 즉 하드웨어 가속과 소프트웨어 유연성을 결합하여 OFDM과 같은 복잡한 시스템의 전례 없는 신속한 프로토타이핑 및 탐구를 가능하게 함을 보여준다.[34]

6.1. OFDM의 기초

- 직교 주파수 분할 다중화(OFDM, Orthogonal Frequency Division Multiplexing)는 고속 데이터 스트림을 다수의 저속 스트림으로 나누어, 서로 직교성을 갖는 좁은 간격의 부반송파들을 통해 동시에 전송하는 기술이다.
- 송신단의 IFFT(역고속 푸리에 변환)와 수신단의 FFT는 OFDM 변복조의 핵심적인 역할을 수행한다.
- 순환 전치(CP, Cyclic Prefix)는 다중 경로 간섭(multipath interference)을 완화하기 위해 사용되는 기술이다.

6.2. OFDM 데모 배포

- 이전 단계에서 익숙해진 pip 설치 과정을 Jupyter 터미널에서 사용하여 rfsoc_ofdm 패키지를 설치한다.[34]

6.3. 물리 계층의 상호작용적 제어

- OFDM 예제 노트북을 실행하면 다수의 플롯과 상호작용 위젯을 포함한 정교한 GUI가 나타난다.
- 위젯을 사용하여 실험을 진행한다. 예를 들어, 부반송파의 변조 방식을 QPSK에서 16-QAM이나 64-QAM으로 변경한다.
- 사용자가 설정을 변경하면, 수신된 부반송파의 성상도 다이어그램이 실시간으로 변하는 것을 관찰할 수 있다. 이는 하드웨어로 가속되는 물리 계층을 실시간으로, 상호작용적으로 제어하고 있음을 보여준다.

6.4. 성능 분석

- OFDM 루프백을 사용하여 전체 데이터 전송을 실행한다.

- 노트북이 제공하는 다양한 시각화 플롯을 분석한다. 여기에는 수신된 스펙트럼, 개별 부반송파의 성상도, 그리고 가능한 경우 비트 오류율(BER, Bit Error Rate) 측정이 포함될 수 있다. 이는 ZCU208에서 실행되는 복잡하고 현대적인 무선 통신 시스템을 시연하는 것으로, 본 가이드의 정점을 이룬다.

결론 및 다음 단계

성과 요약

본 가이드는 1단계의 기본적인 하드웨어 검증부터 시작하여 5단계의 정교한 OFDM 송수신기 구현에 이르기까지 체계적인 여정을 안내했다. 이 과정을 통해 사용자는 하드웨어 설정, PYNQ 환경 탐색, 프로그래밍 방식의 RF I/O, 그리고 실제 SDR 문제에 대한 하드웨어/소프트웨어 공동 설계 원칙 적용과 같은 핵심 기술들을 습득했다.

추가 탐구를 위한 제언

지속적인 학습을 장려하기 위해 다음과 같은 다음 단계들을 제안한다.

- 맞춤형 오버레이 개발: 사전 구축된 데모를 사용하는 단계를 넘어, Vivado를 사용하여 자신만의 하드웨어 디자인을 만들고 이를 위한 PYNQ 오버레이를 개발하는 방법을 학습한다. 관련 자료를 통해 심화 학습을 진행할 수 있다.[8]
- 다른 프레임워크와의 통합: RFSoC를 GNU Radio와 같은 다른 인기 있는 SDR 프레임워크와 통합하는 가능성을 탐구한다. PYNQ와 GNU Radio를 공동 설계하여 고속 데이터 오프로드 시스템을 구축할 수 있다.[36]
- 고급 애플리케이션: 5G 물리 계층 구현, 위상 배열을 위한 디지털 빔포밍, 또는 RFSoC 로직 상에서 직접 변조 방식을 분류하는 실시간 머신러닝과 같은 더 진보된 주제를 탐구한다. 관련 문서 및 튜토리얼을 참고하여 지식을 확장할 수 있다.[17]

참고 자료

1. Zynq UltraScale+ RFSoC ZCU208 Evaluation Kit - AMD, 9월 19, 2025에 액세스, https://www.xilinx.com/publications/product-briefs/xilinx-zcu208-product-brief.pdf
2. IMPLEMENTING A MODERN SOFTWARE DEFINED RADIO USING RFSOC-FPGA -

Trepo, 9월 19, 2025에 액세스, https://trepo.tuni.fi/bitstream/10024/187946/2/LaurilaSanteri.pdf
3. An Adaptable Direct RF Sampling Solution (WP489) - AMD, 9월 19, 2025에 액세스, https://www.amd.com/content/dam/amd/en/documents/solutions/direct-rf-sampling-solution-white-paper.pdf
4. AMD Zynq™ UltraScale+™ RFSoC ZCU208 Evaluation Kit, 9월 19, 2025에 액세스, https://www.amd.com/en/products/adaptive-socs-and-fpgas/evaluation-boards/zcu208.html
5. Zynq UltraScale+ RFSoC ZCU208 Evaluation Kit Part Number - Farnell, 9월 19, 2025에 액세스, https://www.farnell.com/datasheets/3295904.pdf
6. Getting started with RFSoC-PYNQ | RFSoC-PYNQ, 9월 19, 2025에 액세스, http://www.rfsoc-pynq.io/getting_started.html
7. New free eBook: SDR with Zynq Ultrascale+ RFSoC | RFSoC-PYNQ, 9월 19, 2025에 액세스, http://www.rfsoc-pynq.io/
8. Using the RFSoC Data Converter Tool - Xilinx Wiki, 9월 19, 2025에 액세스, https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/57770205/Zynq+UltraScale+RFSoC+Data+Converter+Evalution+Tool
9. PYNQ supported boards and PYNQ pre-built images | PYNQ, 9월 19, 2025에 액세스, http://www.pynq.io/boards.html
10. Xilinx Zcu208 Product Brief | PDF - Scribd, 9월 19, 2025에 액세스, https://www.scribd.com/document/705681241/Xilinx-Zcu208-Product-Brief
11. XTP607 - ZCU208 Software Install and Board Setup, 9월 19, 2025에 액세스, https://www.xilinx.com/support/documents/boards_and_kits/zcu208/2020_1/xtp607-zcu208-setup-c-2020-1.pdf
12. PYNQ-Z2 Setup Guide — Python productivity for Zynq (Pynq) - Read the Docs, 9월 19, 2025에 액세스, https://pynq.readthedocs.io/en/v2.7.0/getting_started/pynq_z2_setup.html
13. Board setup - Xilinx Wiki, 9월 19, 2025에 액세스, https://xilinx-wiki.atlassian.net/wiki/plugins/viewsource/viewpagesrc.action?pageId=569017820
14. RF DC Evaluation Tool for ZCU208 board - Quick Start - Xilinx Wiki, 9월 19, 2025에 액세스, https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/569017820/RF+DC+Evaluation+Tool+for+ZCU208+board+-+Quick+Start
15. xtp600 Zcu208 Bit C 2020 1 | PDF | User (Computing) - Scribd, 9월 19, 2025에 액세스, https://www.scribd.com/document/708414221/xtp600-zcu208-bit-c-2020-1
16. The RFSoC Book and Design Examples for the ZCU208 & ZCU216 - Support - PYNQ, 9월 19, 2025에 액세스, https://discuss.pynq.io/t/the-rfsoc-book-and-design-examples-for-the-zcu208-zcu216/5630
17. Zynq UltraScale+ RFSoC - Xilinx Wiki - Confluence, 9월 19, 2025에 액세스, https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/189530203
18. MicroZed Chronicles: Pynq on the RFSoC! | by Adam Taylor | Medium, 9월 19, 2025에 액세스,

https://medium.com/@aptaylorceng/microzed-chronicles-pynq-on-the-rfsoc-eca22fc857fc
19. strath-sdr/rfsoc_sam: RFSoC Spectrum Analyser Module on PYNQ. - GitHub, 9월 19, 2025에 액세스, https://github.com/strath-sdr/rfsoc_sam
20. strath-sdr/rfsoc_radio: PYNQ example of using the RFSoC as a QPSK/BPSK radio transceiver. - GitHub, 9월 19, 2025에 액세스, https://github.com/strath-sdr/rfsoc_radio
21. PYNQ Radio – Final Report, 9월 19, 2025에 액세스, https://kastner.ucsd.edu/ryan/wp-content/uploads/sites/5/2014/03/admin/pynq_radio.pdf
22. RFSoC 2x2 Phase Measurement - Support - PYNQ, 9월 19, 2025에 액세스, https://discuss.pynq.io/t/rfsoc-2x2-phase-measurement/3892
23. RFSoC-PYNQ design flow - Support, 9월 19, 2025에 액세스, https://discuss.pynq.io/t/rfsoc-pynq-design-flow/7829
24. RFSoC Single DAC - Support - PYNQ, 9월 19, 2025에 액세스, https://discuss.pynq.io/t/rfsoc-single-dac/6553
25. Passing a signal into RFADC, reading out from RFDAC - Support - PYNQ, 9월 19, 2025에 액세스, https://discuss.pynq.io/t/passing-a-signal-into-rfadc-reading-out-from-rfdac/5754
26. RFSoC-PYNQ overlays, 9월 19, 2025에 액세스, http://www.rfsoc-pynq.io/overlays.html
27. Real-time Automatic Modulation Classification using RFSoC - Kastner Research Group, 9월 19, 2025에 액세스, https://kastner.ucsd.edu/wp-content/uploads/2020/05/admin/raw2020-real_time_modulation_classification.pdf
28. Software Defined Radio with the open source RFSoC-PYNQ framework, 9월 19, 2025에 액세스, https://pureportal.strath.ac.uk/en/activities/software-defined-radio-with-the-open-source-rfsoc-pynq-framework
29. RFSoC-PYNQ shared spectrum demo - YouTube, 9월 19, 2025에 액세스, https://www.youtube.com/watch?v=laofKUc7Rzw
30. ZCU208 Evaluation Board User Guide, 9월 19, 2025에 액세스, https://static.chipdip.ru/lib/114/DOC030114706.pdf
31. RTL-SDR FM Receiver - GNU Radio, 9월 19, 2025에 액세스, https://wiki.gnuradio.org/index.php?title=RTL-SDR_FM_Receiver
32. Overview of our SDR system on the RFSoC with PYNQ. (Note, this is a functional block diagram, and does not show the IQ split in the RF-DAC). - ResearchGate, 9월 19, 2025에 액세스, https://www.researchgate.net/figure/Overview-of-our-SDR-system-on-the-RFSoC-with-PYNQ-Note-this-is-a-functional-block_fig1_342909337
33. strath-sdr/rfsoc_qpsk: PYNQ example of using the RFSoC as a QPSK transceiver. - GitHub, 9월 19, 2025에 액세스, https://github.com/strath-sdr/rfsoc_qpsk
34. strath-sdr/rfsoc_ofdm: PYNQ example of an OFDM Transmitter and Receiver on RFSoC., 9월 19, 2025에 액세스, https://github.com/strath-sdr/rfsoc_ofdm

35. Xilinx/RFSoC-PYNQ: Python productivity for RFSoC platforms - GitHub, 9월 19, 2025에 액세스, https://github.com/Xilinx/RFSoC-PYNQ
36. 100GBit/s RF sample offload for RFSoC using GNU Radio and PYNQ - University of Strathclyde, 9월 19, 2025에 액세스, https://pureportal.strath.ac.uk/files/175905824/Siauciulis_etal_NEWCAS2023_100GBit_s_RF_sample_offload_for_RFSoC.pdf
37. Ultra-wideband SDR architecture for AMD RFSoCs “using” PYNQ based GNU Radio blocks, 9월 19, 2025에 액세스, https://events.gnuradio.org/event/21/contributions/418/attachments/138/317/gr_with_rfsoc.pdf
38. RFSoC Implementation - MATLAB & Simulink - MathWorks, 9월 19, 2025에 액세스, https://www.mathworks.com/help/phased/rf-soc.html
39. CASPER RFSoC README — CASPER Tutorials 0.1 documentation, 9월 19, 2025에 액세스, https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/tutorials/rfsoc/readme.html
40. ZCU208 Evaluation Board User Guide - Mouser Electronics, 9월 19, 2025에 액세스, https://www.mouser.com/pdfDocs/ug1410-zcu208-eval-bd.pdf