Microphase ANTSDR E300/E310 플랫폼을 위한 PYNQ 포팅 최종 가이드

제 1장: ANTSDR E310 플랫폼 아키텍처 심층 분석

PYNQ(Python Productivity for Zynq) 프레임워크를 새로운 하드웨어 플랫폼에 성공적으로 이식하는 과정은 대상 하드웨어에 대한 정확하고 깊이 있는 이해에서 시작됩니다. 본 장에서는 포팅 프로세스의 기반이 되는 ANTSDR E310의 하드웨어 아키텍처를 면밀히 분석합니다. 이 분석은 Vivado에서의 하드웨어 정의 생성과 Petalinux에서의 정확한 디바이스 트리 구성을 위한 필수적인 선행 작업입니다.

1.1. 서론: 대상 하드웨어의 재구성

ANTSDR E310 모델은 광범위한 문서와 판매 정보를 통해 그 사양이 상세히 알려져 있으며, 기능적으로 사용자의 요구에 가장 부합하는 플랫폼으로 판단됩니다. 따라서 본 가이드에서는 ANTSDR E310을 포팅 대상의 기준 모델로 가정하고 분석을 진행합니다.

ANTSDR E310의 핵심 아키텍처는 두 가지 주요 구성 요소로 이루어집니다. 첫째는 처리 시스템(Processing System, PS)과 프로그래머블 로직(Programmable Logic, PL)을 통합한 AMD/Xilinx Zynq-7000 SoC이며, 둘째는 SDR(Software Defined Radio)의 아날로그 프론트엔드 역할을 수행하는 Analog Devices사의 RF 트랜시버입니다. 성공적인 PYNQ 포팅은 이 두 시스템을 정확하게 설정하고 유기적으로 연동시키는 것에 달려 있습니다.

1.2. 처리 시스템 (PS): Xilinx Zynq XC7Z020

ANTSDR E310의 심장부에는 Xilinx Zynq-7000 시리즈의 **XC7Z020CLG400** SoC가 탑재되어 있습니다.[4] 이 SoC는 PYNQ 환경이 의존하는 핵심적인 연산 및 제어 기능을 제공합니다.

주요 사양은 다음과 같습니다.

- 프로세서**:** 듀얼 코어 ARM Cortex-A9 MPCore™ 프로세서가 탑재되어 리눅스 운영체제와 PYNQ의 파이썬 환경을 구동하는 주체로 작동합니다.[4]
- 프로그래머블 로직**:** 85k 개의 로직 셀(Logic Cells)을 포함하여, SDR 신호 처리 가속기와 같은 사용자 정의 하드웨어 로직을 구현할 수 있는 충분한 공간을 제공합니다.[4]
- 메모리**:** 1 GByte 용량의 DDR3 RAM이 장착되어 있어, 리눅스 커널과 사용자 애플리케이션이 원활하게 동작할 수 있는 메모리 공간을 확보합니다.[4] 이 메모리 사양은 Vivado에서 Zynq의 메모리 컨트롤러를 설정할 때 정확히 반영되어야 합니다.
- 주변장치**:** PYNQ의 운영에 필수적인 기가비트 이더넷, USB 2.0 OTG, UART, 그리고 부팅 이미지를 저장하는 Micro SD 카드 슬롯과 같은 핵심 주변장치 인터페이스를 내장하고 있습니다.[6]

1.3. 프로그래머블 로직 (PL) 및 RF 프론트엔드: Analog Devices AD9361/AD9363

SDR 기능의 핵심은 RF 신호를 송수신하고 디지털 신호로 변환하는 RF 프론트엔드입니다. ANTSDR 플랫폼은 두 가지 버전의 Analog Devices RF 트랜시버를 선택적으로 탑재합니다: AD9361 또는 AD9363.[4]

두 칩셋의 주요 차이점은 다음과 같습니다.

- **AD9361:** 더 넓은 주파수 범위(70 MHz ~ 6 GHz)와 신호 대역폭(최대 56 MHz)을 지원하여, 광범위한 RF 애플리케이션에 적용할 수 있는 유연성을 제공합니다.[8]
- **AD9363:** 상대적으로 좁은 주파수 범위(325 MHz ~ 3.8 GHz)와 신호 대역폭(최대 20 MHz)을 가지지만, 특정 애플리케이션에 비용 효율적인 솔루션을 제공합니다.[8]

포팅 과정에서 사용자가 보유한 하드웨어의 정확한 트랜시버 모델을 파악하는 것은 매우 중요합니다. Zynq SoC와 AD936x 트랜시버 간의 인터페이스는 고속 데이터 전송을 위한 LVDS(Low-Voltage Differential Signaling) 병렬 데이터 버스와 칩 제어를 위한 SPI(Serial Peripheral Interface) 버스로 구성됩니다. 이 연결 정보는 4장에서 다룰 Petalinux 디바이스 트리 구성의 핵심 요소가 됩니다.

1.4. 온보드 주변장치 및 연결성

PYNQ 환경의 완전한 기능을 활용하기 위해 ANTSDR E310은 다음과 같은 필수적인 I/O 및 주변장치를 갖추고 있습니다.

- 기가비트 이더넷**:** RTL8211F PHY 칩을 통해 구현되며, Jupyter Notebook 접속 및 원격 제어를 위한 네트워크 연결을 제공합니다.[5]
- **USB 2.0 OTG:** 다양한 USB 장치 연결을 지원합니다.[6]
- **Micro SD** 카드 슬롯**:** PYNQ 부팅 이미지를 저장하고 시스템을 부팅하는 주 저장소 역할을 합니다.[6]
- **USB-UART:** 시스템 부팅 과정을 모니터링하고 디버깅하기 위한 시리얼 콘솔 접근을 제공합니다.[6]

Zynq 기반 보드에서 PYNQ를 부팅하기 위한 표준 절차와 마찬가지로, ANTSDR E310 역시 부팅 모드 점퍼를 SD 카드 모드로 설정해야 합니다. 이 물리적인 설정은 부트롬(BootROM)이 SD 카드에서 부트로더를 로드하도록 지시하는 첫 단계입니다.[9]

아래 표는 ANTSDR E310의 핵심 하드웨어 사양을 요약한 것입니다. 이 정보는 이어지는 장에서 하드웨어 정의 파일(.xsa)과 디바이스 트리를 생성하는 데 직접적으로 사용될 중요한 기초 자료입니다.

표 **1.1: ANTSDR E310** 핵심 구성 요소 사양

구성 요소	사양	출처
SoC	AMD/Xilinx Zynq XC7Z020CLG400 (듀얼 코어 ARM Cortex-A9, 85k 로직 셀)	[4]
RF 트랜시버 (옵션 1)	Analog Devices AD9361 (70 MHz - 6 GHz, 56 MHz 대역폭, 2x2 MIMO)	[8]
RF 트랜시버 (옵션 2)	Analog Devices AD9363 (325 MHz - 3.8 GHz, 20 MHz 대역폭, 2x2 MIMO)	[8]
DDR3 RAM	1 GByte	[4]
QSPI Flash	32 MByte	[4]
이더넷 PHY	Realtek RTL8211F (기가비트 이더넷)	[5]

연결 포트	기가비트 이더넷, USB 2.0 OTG, Micro SD 카드, USB-UART, USB-JTAG	[6]

제 2장: PYNQ v3.0 빌드 환경 구축

성공적인 PYNQ 이미지 빌드를 위해서는 개발 환경을 엄격한 요구사항에 맞춰 구성하는 것이 무엇보다 중요합니다. PYNQ 프레임워크와 Xilinx 개발 도구(Vivado, Petalinux) 간의 버전 불일치는 빌드 실패의 가장 흔한 원인입니다. 본 장에서는 PYNQ v3.0 포팅을 위한 개발 호스트 머신 준비 과정을 단계별로 안내합니다.

2.1. PYNQ-툴체인 의존성 매트릭스

PYNQ의 각 릴리스는 특정 버전의 Vivado 및 Petalinux와 긴밀하게 연결되어 있습니다. 이는 단순한 권장사항이 아닌, 반드시 지켜야 할 기술적 요구사항입니다. PYNQ의 빌드 스크립트와 사전 컴파일된 구성 요소들은 특정 버전의 리눅스 커널 및 라이브러리를 기반으로 개발되고 테스트되기 때문입니다.

본 가이드의 목표인 **PYNQ v3.0.x** 버전을 빌드하기 위해서는 반드시 **Vivado, Vitis, Petalinux 2022.1** 버전을 사용해야 합니다.[11] 이 버전의 Petalinux는 리눅스 커널 5.15 버전을 기반으로 하며, PYNQ v3.0의 소프트웨어 스택은 이 커널 버전에 맞춰져 있습니다.[12] 또한, 하드웨어 오버레이는 Vivado 2022.1 버전으로 합성되어야 완벽한 호환성을 보장합니다. 다른 버전을 사용할 경우, 빌드 스크립트의 버전 확인 단계에서부터 오류가 발생하거나, 성공적으로 빌드되더라도 런타임에서 예기치 않은 문제를 일으킬 수 있습니다.

2.2. 호스트 운영체제 요구사항

Xilinx 2022.1 개발 도구는 특정 리눅스 배포판 및 버전을 공식적으로 지원합니다. 가장 안정적인 빌드 환경을 위해 **Ubuntu 20.04 LTS**를 사용하는 것이 권장됩니다.[15] Ubuntu 22.04와 같은 미지원 운영체제에서는 도구 설치나 빌드 스크립트 실행 과정에서 호환성 문제로 인해 실패할 수 있습니다.[16]

개발 환경의 독립성과 청결성을 유지하기 위해, VirtualBox나 VMWare와 같은 가상 머신 또는 Docker 컨테이너를 사용하는 것이 가장 좋은 방법입니다. 이는 호스트 시스템의 다른 라이브러리와의 충돌을 방지하고, 빌드에 필요한 모든 종속성을 격리된 공간에 설치할 수 있게 해줍니다. PYNQ 프로젝트는 공식적으로 Docker를 이용한 빌드 흐름을 제공하여 이 과정을 용이하게 합니다.[11]

2.3. 단계별 설치 가이드

아래는 PYNQ v3.0 이미지 빌드에 필요한 모든 구성 요소를 설치하는 상세한 절차입니다.

2.3.1. Xilinx Vivado/Vitis 2022.1 설치

1. AMD/Xilinx 공식 다운로드 페이지에 접속하여 Vivado ML Edition 2022.1 버전의 통합 설치 프로그램을 다운로드합니다.
2. 다운로드한 설치 프로그램을 실행하고, 안내에 따라 설치를 진행합니다. Vitis 통합 소프트웨어 플랫폼을 포함하여 설치하는 것이 좋습니다.

2.3.2. Petalinux 2022.1 설치

1. Xilinx 다운로드 페이지에서 Petalinux 2022.1 설치 프로그램(.run 파일)을 다운로드합니다.
2. Petalinux 설치에 필요한 종속 패키지를 설치합니다. 터미널에서 다음 명령어를 실행합니다.
   Bash
   ```
   sudo apt-get install -y tofrodos iproute2 gawk gcc g++ make net-tools libncurses5-dev tftpd zlib1g-dev libssl-dev flex bison libselinux1 xterm autoconf libtool texinfo zlib1g:i386 libstdc++6:i386 libglib2.0-0:i386
   ```

3. 다운로드한 설치 프로그램을 실행하여 Petalinux를 설치합니다. 예를 들어, /opt/pkg/petalinux/2022.1 경로에 설치할 수 있습니다.

2.3.3. PYNQ 저장소 복제

1. Git을 사용하여 공식 PYNQ GitHub 저장소를 복제합니다.
   Bash
   git clone https://github.com/Xilinx/PYNQ.git

2. v3.0 릴리스에 해당하는 브랜치나 태그로 체크아웃합니다.
   Bash
   cd PYNQ
   git checkout v3.0.1 # 예시 태그, 최신 v3.0.x 태그 확인 필요

2.3.4. 환경 설정

빌드 명령을 실행하기 전에, Vivado, Vitis, Petalinux 도구를 사용하기 위한 환경 변수를 설정해야 합니다. 터미널에서 다음 명령어를 실행하여 각 도구의 settings.sh 스크립트를 소싱(source)합니다. 이 명령어들은 .bashrc 파일에 추가하여 터미널 세션이 시작될 때마다 자동으로 실행되도록 설정하는 것이 편리합니다.

Bash

```
# Vivado/Vitis 환경 설정
source /opt/Xilinx/Vivado/2022.1/settings64.sh
source /opt/Xilinx/Vitis/2022.1/settings64.sh

# Petalinux 환경 설정
source /opt/pkg/petalinux/2022.1/settings.sh
```

이처럼 빌드 환경은 제안이 아닌 엄격한 명세입니다. PYNQ 생태계 전체가 버전 잠금 방식으로 운영된다는 점을 이해하는 것이 중요합니다. 이 단계를 정확히 따름으로써, 잘못된 환경 구성으로 인해 발생할 수 있는 수많은 디버깅 시간을 절약하고 안정적인 빌드 프로세스를 보장할 수 있습니다.

제 3장: Vivado를 이용한 기본 하드웨어 정의 설계

본 장에서는 Vivado를 사용하여 ANTSDR E310을 위한 기본 하드웨어 설계를 생성하는 과정을 안내합니다. 이 설계의 결과물인 .xsa 파일은 Zynq 프로세서의 구성과 외부 세계와의 연결 방식을 정의하며, 전체 소프트웨어 빌드 프로세스의 기초가 됩니다. PYNQ 포팅의 초기 단계에서는 복잡한 PL 로직 구현보다 PS 측의 주변장치를 정확하게 설정하는 것이 더 중요합니다.

3.1. 신규 **Vivado** 프로젝트 생성

1. Vivado 2022.1을 실행하고 'Create Project'를 선택하여 새 프로젝트를 시작합니다.
2. 프로젝트 유형으로 'RTL Project'를 선택하고, 'Do not specify sources at this time' 옵션을 체크합니다.
3. 'Default Part' 설정 단계에서 ANTSDR E310의 사양에 맞는 Zynq SoC 부품을 선택합니다. 검색 필터를 사용하여 **XC7Z020CLG400-1**을 찾아 선택합니다.[4] 여기서 '-1'은 속도 등급을 의미하며, 일반적인 기본값입니다.
4. 프로젝트 생성을 완료합니다.

3.2. **Zynq** 블록 디자인 구축

1. 'Flow Navigator' 패널에서 'IP Integrator' 아래의 'Create Block Design'을 클릭합니다. 기본 이름(예: design_1)으로 블록 디자인을 생성합니다.
2. 다이어그램 캔버스에서 '+' 버튼을 클릭하거나 Ctrl+I를 눌러 IP를 추가합니다. 검색창에 'ZYNQ7'을 입력하고 'ZYNQ7 Processing System' IP 코어를 추가합니다.
3. Vivado가 자동으로 블록 자동화(Block Automation)를 제안할 것입니다. 'Run Block Automation'을 클릭하고, 나타나는 대화상자에서 'Apply Board Preset' 옵션을 체크 해제한 후 'OK'를 클릭합니다. ANTSDR은 공식 지원 보드가 아니므로, 모든 설정을 수동으로 진행해야 합니다.

3.3. **ZYNQ7** 처리 시스템 **(PS)** 구성

이 단계는 포팅 과정에서 가장 중요한 하드웨어 설정 부분입니다. Zynq PS IP 코어를 더블 클릭하여 'Re-customize IP' 대화상자를 엽니다. 1장에서 정리한 하드웨어 사양(표 1.1)을 바탕으로 다음 항목들을 정확하게 설정합니다.

- **DDR Configuration:** 'DDR Controller' 페이지로 이동하여 메모리 설정을 진행합니다. ANTSDR E310은 1 GByte의 DDR3 RAM을 사용하므로, 'Memory Part'를 보드에 장착된

DDR3 칩 사양에 맞게 설정합니다.

- **MIO Configuration:** 'MIO Configuration' 페이지에서 PYNQ 운영에 필수적인 주변장치들을 활성화하고 MIO 핀에 매핑합니다.
  - **UART:** 콘솔 통신을 위해 UART 1을 활성화합니다.
  - **Gigabit Ethernet:** 네트워크 연결을 위해 ENET 0 (GEM0)을 활성화하고, 인터페이스를 MIO로 설정합니다.
  - **USB:** USB 0을 활성화합니다.
  - **SD Card:** 부팅을 위해 SD 0 (SDIO)를 활성화합니다.
- **Clock Configuration:** 'Clock Configuration' 페이지에서 PS의 입력 클럭 주파수를 설정합니다.
- **SPI Configuration:** AD9361 트랜시버와의 제어 통신을 위해 SPI 인터페이스를 활성화해야 합니다. 'MIO Configuration' -> 'I/O Peripherals'에서 SPI 0 또는 SPI 1을 활성화합니다. 이때, SPI 핀들을 MIO가 아닌 **EMIO(Extended MIO)**로 라우팅하도록 설정합니다. 이는 SPI 신호가 PS에서 나와 PL을 통해 외부 핀으로 연결되도록 하기 위함입니다.
- **GPIO Configuration (선택 사항):** 부팅 상태 확인용 LED 제어를 위해 AXI GPIO IP를 추가하는 것이 일반적입니다. 블록 디자인에 'AXI GPIO' IP를 추가하고, 'Run Connection Automation'을 통해 Zynq PS의 AXI GP 포트에 연결합니다.[18]

3.4. 검증, 합성 및 하드웨어 익스포트

1. 모든 PS 설정이 완료되면, 블록 디자인 창 상단의 'Validate Design' (체크 표시 아이콘)을 실행하여 설계 오류가 없는지 확인합니다.
2. 'Sources' 패널에서 블록 디자인 파일(.bd)을 마우스 오른쪽 버튼으로 클릭하고 'Create HDL Wrapper'를 선택하여 Verilog 또는 VHDL 래퍼 파일을 생성합니다.
3. 'Flow Navigator'에서 'Run Synthesis'와 'Run Implementation'을 차례로 실행합니다.
4. 구현이 완료되면 'Generate Bitstream'을 실행하여 비트스트림 파일(.bit)을 생성합니다. 이 단계의 비트스트림은 PL 로직이 거의 비어있지만, 전체 설계의 유효성을 검증하는 역할을 합니다.
5. 가장 중요한 마지막 단계는 하드웨어 정의를 내보내는 것입니다. Vivado 메뉴에서 **File -> Export -> Export Hardware**를 선택합니다. 대화상자에서 'Include bitstream' 옵션을 선택하고, .xsa (Xilinx Support Archive) 파일을 생성합니다.[18]

이 .xsa 파일은 PS의 모든 구성 정보, PL의 인터페이스 정보, 그리고 비트스트림을 포함하는 압축 아카이브입니다. 이 파일은 다음 장에서 Petalinux가 리눅스 커널과 디바이스 트리를 생성하는 데 사용하는 유일하고 가장 중요한 입력물이 됩니다. Vivado에서 주변장치를 활성화하지 않으면, Petalinux는 해당 하드웨어의 존재를 전혀 알 수 없으며, 결과적으로 리눅스 커널에서도 해당 장치를 사용할 수 없게 됩니다. 따라서 이 단계의 정확성은 전체 포팅 성공의 전제 조건입니다.

제 4장: Petalinux를 이용한 리눅스 커널 맞춤화

본 장은 소프트웨어 맞춤화 과정의 핵심입니다. 3장에서 생성한 하드웨어 정의(.xsa 파일)를 기반으로, ANTSDR E310에 최적화된 리눅스 배포판을 빌드합니다. 이 과정의 가장 중요한 목표는 SDR 기능의 핵심인 AD9361 트랜시버 드라이버를 리눅스 커널에 성공적으로 통합하는 것입니다.

4.1. Petalinux 프로젝트 생성

1. 작업 디렉터리를 생성하고, 터미널에서 다음 명령어를 실행하여 Zynq 템플릿을 기반으로 새 Petalinux 프로젝트를 생성합니다.
   Bash
   ```
   petalinux-create --type project --template zynq --name antsdr_pynq
   ```

2. 생성된 프로젝트 디렉터리(antsdr_pynq)로 이동합니다.
3. 3장에서 생성한 .xsa 파일의 경로를 지정하여 Petalinux 프로젝트가 하드웨어 구성을 인식하도록 설정합니다.
   Bash
   ```
   petalinux-config --get-hw-description=<path_to_xsa_directory>
   ```

   이 명령을 실행하면 텍스트 기반의 시스템 설정 메뉴가 나타납니다. 특별히 변경할 사항이 없다면, 저장하고 종료합니다.

4.2. Analog Devices meta-adi 레이어 통합

Xilinx의 기본 리눅스 커널에는 AD9361 드라이버가 포함되어 있지 않습니다. 이 드라이버들은 Analog Devices(ADI)에서 별도로 유지 관리하는 Yocto 레이어인 meta-adi에 포함되어 있습니다.[20] 따라서 이 레이어들을 Petalinux 프로젝트에 추가해야 합니다.

1. Petalinux 프로젝트 외부의 적절한 위치에 meta-adi 저장소를 복제합니다.
   Bash
   ```
   git clone https://github.com/analogdevicesinc/meta-adi.git
   ```

2. Petalinux 프로젝트의 project-spec/meta-user/conf/bblayers.conf 파일을 텍스트 편집기로

엽니다.

3. 파일의 BBLAYERS 변수에 meta-adi-core와 meta-adi-xilinx 레이어의 절대 경로를 추가합니다. 예를 들어, 다음과 같이 수정합니다.[23]
   코드 스니펫
   ```
   BBLAYERS?= " \
    <path-to-project>/project-spec/meta-user \
    <path-to-project>/components/yocto/layers/meta-xilinx/meta-xilinx-bsp \
    <path-to-project>/components/yocto/layers/meta-openembedded/meta-oe \
    <path-to-project>/components/yocto/layers/meta-openembedded/meta-python \
    <path-to-project>/components/yocto/layers/meta-openembedded/meta-networking \
    <path-to-project>/components/yocto/layers/meta-petalinux \
    <path-to-project>/components/yocto/layers/meta-xilinx-tools \
    /path/to/meta-adi/meta-adi-core \
    /path/to/meta-adi/meta-adi-xilinx \
    "
   ```

   이 설정은 Petalinux 빌드 시스템이 ADI에서 제공하는 레시피와 드라이버를 찾아서 빌드에 포함하도록 지시합니다.

4.3. 커널 구성 (petalinux-config -c kernel)

이제 리눅스 커널 설정 메뉴로 진입하여 AD9361 드라이버를 명시적으로 활성화해야 합니다.

1. 프로젝트 루트 디렉터리에서 다음 명령어를 실행합니다.
   Bash
   ```
   petalinux-config -c kernel
   ```

2. 텍스트 기반의 menuconfig 인터페이스가 나타나면, 다음 경로로 이동하여 관련 드라이버를 활성화(*로 선택)합니다.
   - Device Drivers --->
     - Industrial I/O support --->
       - Analog to digital converters --->
         - <*> Analog Devices AD9361/AD9364 transceiver [20]
       - Transceivers ---> (위 옵션을 선택하면 자동으로 활성화될 수 있음)
   - 디버깅을 위해 SPI 장치 드라이버도 활성화하는 것이 좋습니다.
   - Device Drivers --->
     - SPI support --->
       - <*> User mode SPI device driver support [24]
3. 설정을 저장하고 menuconfig를 종료합니다.

4.4. 디바이스 트리 오버레이 작성

임베디드 리눅스에서 하드웨어를 활성화하는 것은 두 부분으로 구성됩니다: 커널 드라이버(기능, "what")와 디바이스 트리(연결 정보, "where"). 커널에 드라이버를 포함시켰더라도, 디바이스 트리를 통해 해당 하드웨어가 시스템의 어디에 어떻게 연결되어 있는지 알려주지 않으면 드라이버는 로드되지 않습니다.

project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi 파일을 열고, AD9361 트랜시버를 기술하는 노드를 추가해야 합니다. 아래는 ANTSDR E310에 적용할 수 있는 디바이스 트리 오버레이 예시입니다.[20]

DTS

```
/include/ "system-conf.dtsi"
/ {
  /* AD9361에 40MHz 레퍼런스 클럭을 제공하기 위한 고정 클럭 노드 */
  clocks {
    ad9361_clkin: clock@0 {
      compatible = "fixed-clock";
      clock-frequency = ; /* 40 MHz */
      clock-output-names = "ad9361_ext_refclk";
      #clock-cells = ;
    };
  };
};

/* Vivado에서 EMIO로 라우팅한 SPI 컨트롤러를 참조 */
&spi0 {
  status = "okay";
  /* AD9361 트랜시버 디바이스 노드 */
  ad9361-phy@0 {
    compatible = "adi,ad9361"; /* 이 문자열을 보고 커널이 ad9361 드라이버를 로드함 */
    reg = ; /* SPI Chip Select 0 */
    spi-cpha;
    spi-max-frequency = ; /* 10 MHz SPI 클럭 */

     /* 클럭 설정: 위에서 정의한 ad9361_clkin을 레퍼런스 클럭으로 사용 */
    clocks = <&ad9361_clkin>;
```

```
        clock-names = "ad9361_ext_refclk";

        /* ANTSDR E310의 기본 모드 설정 */
        adi,2rx-2tx-mode-enable; /* 2x2 MIMO 모드 활성화 */
        adi,frequency-division-duplex-mode-enable; /* FDD 모드 */

        /* RF 파라미터 예시 (실제 사용 시 애플리케이션에 맞게 조정 필요) */
        adi,rf-rx-bandwidth-hz = ; /* 18 MHz RX 대역폭 */
        adi,rf-tx-bandwidth-hz = ; /* 18 MHz TX 대역폭 */
        adi,rx-synthesizer-frequency-hz = /bits/ 64 ; /* 2.4 GHz RX LO 주파수 */
        adi,tx-synthesizer-frequency-hz = /bits/ 64 ; /* 2.45 GHz TX LO 주파수 */

        /* 디지털 인터페이스 설정 */
        adi,lvds-mode-enable;
    };
};
```

이 .dtsi 파일은 커널에게 "SPI 0 버스의 칩 셀렉트 0번에 ad9361과 호환되는 장치가 연결되어 있으며, 이 장치는 ad9361_clkin이라는 40 MHz 클럭을 사용한다"고 알려주는 역할을 합니다. 이 정보가 없으면 커널은 부팅되더라도 AD9361의 존재를 인지하지 못해 SDR 기능이 활성화되지 않습니다.

4.5. Petalinux 이미지 빌드 (petalinux-build)

모든 설정이 완료되면, 프로젝트 루트 디렉터리에서 다음 명령어를 실행하여 커널, 디바이스 트리, 루트 파일 시스템, U-Boot 등 모든 구성 요소를 컴파일합니다.

Bash

```
petalinux-build
```

빌드가 성공적으로 완료되면, images/linux 디렉터리에 부팅에 필요한 파일들(BOOT.BIN, image.ub, rootfs.tar.gz 등)이 생성됩니다. 이 파일들은 다음 장에서 PYNQ 이미지로 통합될 준비를 마친 것입니다.

제 5장: PYNQ sdbuild 통합 및 이미지 컴파일

본 장에서는 4장에서 생성한 Petalinux 부트 구성 요소들을 완전한 PYNQ 환경과 결합하는 과정을 다룹니다. 이 작업은 Petalinux를 대체하는 것이 아니라, PYNQ 프로젝트의 고유한 빌드 시스템인 sdbuild를 사용하여 최종 부팅 가능한 SD 카드 이미지를 조립하는 과정입니다. sdbuild는 이식성과 재사용성을 극대화하기 위해 설계된 오케스트레이션 계층입니다.

5.1. PYNQ sdbuild 흐름의 이해

PYNQ sdbuild는 모듈식 접근 방식을 채택합니다. 이 시스템은 보드에 특화된 부트 구성 요소(BSP, Board Support Package)와 아키텍처는 같지만 보드와는 무관한 Ubuntu 기반의 루트 파일 시스템을 결합합니다. 마지막으로 PYNQ 관련 파이썬 라이브러리와 패키지를 설치하여 최종 이미지를 완성합니다.[17] 이 방식은 모든 보드에 대해 전체 Ubuntu 배포판을 처음부터 빌드하는 데 드는 엄청난 시간을 절약해 줍니다. 즉, Petalinux는 보드별 부팅에 필요한 최소한의 요소(FSBL, U-Boot, 커널, 디바이스 트리)를 만드는 데 사용되고,

sdbuild는 이 요소들을 PYNQ의 표준 루트 파일 시스템과 통합하는 역할을 담당합니다.

5.2. 사용자 정의 보드 저장소 생성

PYNQ sdbuild는 보드별 설정을 저장소(repository) 단위로 관리합니다. ANTSDR E310을 위한 새 저장소를 생성해야 합니다.

1. 2장에서 복제한 PYNQ 저장소의 boards 디렉터리로 이동합니다.
2. ANTSDR E310을 위한 새 디렉터리를 생성합니다.
   Bash
   ```
   cd <PYNQ_repository_path>/boards
   mkdir ANTSDR_E310
   ```

   이 디렉터리는 ANTSDR E310 빌드에 필요한 모든 파일을 담는 공간이 됩니다.[19]

5.3. 보드 명세 (.spec) 파일 작성

sdbuild 메이크파일은 .spec 파일을 읽어 각 보드에 대한 빌드 방법을 결정합니다. ANTSDR_E310 디렉터리 내에 ANTSDR_E310.spec 파일을 생성하고 다음 내용을 작성합니다. 이 내용은 다른 Zynq-7000 기반 보드의 예시를 참고하여 작성되었습니다.[19]

RPM spec files

```
# 아키텍처 지정: Zynq-7000 시리즈는 'arm'
ARCH_ANTSDR_E310 := arm

# Petalinux BSP 파일 경로 지정. 이 파일은 다음 단계에서 생성하여 위치시킬 것임
BSP_ANTSDR_E310 := petalinux_bsp/antsdr_e310.bsp

# 최종 이미지에 설치할 PYNQ 패키지 목록
# 'pynq'는 필수이며, 'ethernet'은 네트워크 자동 설정을 위해 추가
STAGE4_PACKAGES_ANTSDR_E310 := pynq ethernet
```

이 파일은 make 명령에게 ANTSDR_E310 보드가 ARM 아키텍처이며, 지정된 BSP를 사용하여 부트 파티션을 구성하고, 최종적으로 pynq와 ethernet 패키지를 루트 파일 시스템에 설치해야 함을 알려줍니다.

5.4. 보드 저장소 채우기

이제 4장에서 수행한 Petalinux 작업을 PYNQ 빌드 흐름에 통합할 차례입니다.

1. **BSP** 패키징**:** 4장에서 작업한 Petalinux 프로젝트(antsdr_pynq)의 루트 디렉터리로 이동합니다.
2. 다음 명령어를 실행하여 부트 파일을 포함하는 BSP를 패키징합니다.
   Bash
   ```
   petalinux-package --bsp -o antsdr_e310.bsp --with-bootfiles
   ```
   이 명령은 antsdr_e310.bsp라는 단일 아카이브 파일을 생성합니다.
3. **BSP** 배치**:** 생성된 antsdr_e310.bsp 파일을 PYNQ의 사용자 정의 보드 저장소 내 적절한 위치로 복사합니다.
   Bash
   ```
   # ANTSDR_E310 보드 저장소 내에 petalinux_bsp 디렉터리 생성
   mkdir -p <PYNQ_repository_path>/boards/ANTSDR_E310/petalinux_bsp
   ```

```
# 생성된 BSP 파일을 해당 디렉터리로 복사
cp antsdr_e310.bsp <PYNQ_repository_path>/boards/ANTSDR_E310/petalinux_bsp/
```

5.5. 최종 make 명령 실행

모든 준비가 완료되었습니다. 이제 PYNQ sdbuild 디렉터리에서 최종 이미지 빌드를 시작합니다.

1. PYNQ 저장소의 sdbuild 디렉터리로 이동합니다.
   ```
   Bash
   cd <PYNQ_repository_path>/sdbuild
   ```

2. make 명령을 실행하여 ANTSDR E310 보드 이미지를 빌드합니다.
   ```
   Bash
   make BOARDS=ANTSDR_E310
   ```

   이 명령은 다음과 같은 작업을 순차적으로 수행합니다.
   - ANTSDR_E310.spec 파일을 읽어 빌드 설정을 확인합니다.
   - 지정된 경로에서 antsdr_e310.bsp 파일을 찾아 압축을 해제합니다.
   - BSP에 포함된 부트 파일(FSBL, U-Boot, 커널 등)을 사용하여 부트 파티션을 구성합니다.
   - 사전 빌드된 PYNQ의 Ubuntu 루트 파일 시스템 이미지를 마운트합니다.
   - .spec 파일에 명시된 pynq, ethernet 등의 패키지를 마운트된 루트 파일 시스템에 설치(chroot 환경)합니다.
   - 모든 작업을 완료한 후, 부트 파티션과 수정된 루트 파일 시스템을 결합하여 최종 SD 카드 이미지 파일(.img)을 생성합니다.

빌드가 성공적으로 끝나면, sdbuild/output 디렉터리에서 ANTSDR_E310-v3.0.1.img와 같은 이름의 최종 이미지 파일을 찾을 수 있습니다. 이 파일이 바로 ANTSDR E310에서 PYNQ를 실행하기 위한 모든 것을 담고 있는 결과물입니다.

제 6장: 배포, 검증 및 초기 시스템 테스트

성공적으로 빌드된 PYNQ 이미지를 실제 하드웨어에서 실행하고 포팅의 성공 여부를 확인하는 마지막 단계입니다. 본 장에서는 생성된 이미지를 SD 카드에 기록하고, ANTSDR E310 보드에서

부팅한 후, 다층적 검증 절차를 통해 시스템이 의도대로 작동하는지 체계적으로 테스트하는 방법을 안내합니다.

6.1. SD 카드에 이미지 기록

1. sdbuild/output 디렉터리에서 생성된 .img 파일을 찾습니다.
2. BalenaEtcher, Raspberry Pi Imager 또는 Linux의 dd 명령어와 같은 도구를 사용하여 이 이미지 파일을 Micro SD 카드(최소 8GB 권장)에 기록합니다.[10] 이 과정에서 SD 카드의 기존 내용은 모두 삭제되므로 주의해야 합니다.

dd 명령어 예시 **(Linux):**

Bash

```
# 먼저 'lsblk' 명령으로 SD 카드의 장치 이름(예: /dev/sdX)을 정확히 확인
sudo dd if=ANTSDR_E310-v3.0.1.img of=/dev/sdX bs=4M status=progress conv=fsync
```

6.2. 첫 부팅 및 콘솔 연결

1. 부팅 모드 설정**:** ANTSDR E310 보드의 부팅 모드 점퍼를 'SD 카드' 부팅 모드로 설정합니다. 이는 부트로더가 SD 카드에서 운영체제를 로드하도록 지시하는 필수적인 물리적 설정입니다.[9]
2. **SD** 카드 삽입**:** 이미지가 기록된 Micro SD 카드를 보드의 슬롯에 삽입합니다.
3. 콘솔 연결**:** Micro USB 케이블을 사용하여 컴퓨터와 ANTSDR E310의 UART 포트를 연결합니다.
4. 시리얼 터미널 실행**:** 컴퓨터에서 MobaXterm, PuTTY, screen과 같은 시리얼 터미널 프로그램을 실행하고, ANTSDR의 시리얼 포트에 다음 설정으로 연결합니다: **115200 baud, 8 data bits, no parity, 1 stop bit (8N1)**.[30]
5. 전원 인가**:** 보드에 전원을 공급합니다. 시리얼 터미널에 U-Boot와 리눅스 커널의 부팅 메시지가 출력되기 시작할 것입니다. 이 메시지들을 통해 부팅 과정을 실시간으로 모니터링하고, 문제가 발생할 경우 원인을 파악할 수 있습니다.

6.3. 명령줄을 통한 시스템 검증

시스템이 부팅되어 로그인 프롬프트가 나타나면, 기본 PYNQ 자격 증명(사용자명: xilinx, 비밀번호: xilinx)으로 로그인합니다. 그 후, 다음 명령어를 통해 포팅의 핵심 요소들이 올바르게 작동하는지 확인합니다. 이 다층적 검증은 문제 발생 시 원인 범위를 좁히는 데 매우 효과적입니다.

1. 커널 드라이버 확인 **(**가장 중요**):** dmesg 명령어를 사용하여 커널 로그를 확인하고, AD9361 드라이버가 성공적으로 로드되었는지 검사합니다.
   Bash
   dmesg | grep ad9361

   성공적인 경우, 다음과 유사한 출력이 나타나야 합니다. 이는 디바이스 트리 설정이 올바르고 커널이 하드웨어를 성공적으로 인식했음을 의미합니다.
   ad9361 spi0.0: ad9361_probe : enter

...
ad9361 spi0.0: ad9361 Rev 2, Firmware Rev 3, Pitch 15 successfully initialized
```
이 메시지가 보이지 않는다면, 문제는 커널 설정 또는 디바이스 트리 단계에 있을 가능성이 높습니다.

2. 네트워크 연결 확인**:** 이더넷 인터페이스가 IP 주소를 할당받았는지 확인합니다.
   Bash
   ip a

   eth0 인터페이스에 유효한 IP 주소가 표시되는지 확인합니다.
3. **IIO** 디바이스 노드 확인**:** Industrial I/O(IIO) 프레임워크는 사용자 공간 애플리케이션이 AD9361과 통신하는 표준 인터페이스입니다. 관련 디바이스 파일이 생성되었는지 확인합니다.
   Bash
   ls /sys/bus/iio/devices/

   iio:device0와 같은 디렉터리가 존재해야 합니다.

## 6.4. PYNQ 환경 접속

1. ANTSDR E310의 이더넷 포트를 로컬 네트워크(라우터 등)에 연결합니다.
2. 보드의 IP 주소를 확인한 후, 컴퓨터의 웹 브라우저에서 다음 주소로 접속합니다. Jupyter
```

Notebook의 기본 포트는 9090입니다.
http://<board_ip_address>:9090 31

3. PYNQ 로그인 화면이 나타나면, 동일한 자격 증명(xilinx/xilinx)으로 로그인합니다.

6.5. Jupyter를 이용한 "Hello, SDR" 테스트

마지막 검증 단계는 PYNQ의 Python 환경에서 실제로 SDR 하드웨어와 통신해보는 것입니다. 이는 커널 드라이버부터 최상위 Python 라이브러리까지 전체 소프트웨어 스택이 올바르게 작동함을 증명합니다.

1. Jupyter 대시보드에서 새 노트북을 생성합니다.
2. 다음과 같은 간단한 Python 코드를 셀에 입력하고 실행합니다. 이 코드는 libiio의 Python 바인딩을 사용하여 AD9361 디바이스 컨텍스트에 연결하고, 현재 설정된 LO(Local Oscillator) 주파수와 같은 속성을 읽어옵니다.

Python

```
import iio

try:
    # 로컬 IIO 컨텍스트 찾기
    ctx = iio.LocalContext()

    # ad9361-phy 디바이스 찾기
    phy = ctx.find_device("ad9361-phy")

    if phy:
        print("AD9361-phy 디바이스를 찾았습니다.")

        # 송신 LO 주파수 속성 읽기
        tx_lo_freq = phy.attrs.value
        print(f"현재 TX LO 주파수: {int(tx_lo_freq) / 1e6} MHz")

        # 수신 LO 주파수 속성 읽기
        rx_lo_freq = phy.attrs.value
        print(f"현재 RX LO 주파수: {int(rx_lo_freq) / 1e6} MHz")

        print("\nPYNQ에서 AD9361 통신 검증 성공!")
    else:
```

```
    print("AD9361-phy 디바이스를 찾을 수 없습니다.")

except Exception as e:
    print(f"오류 발생: {e}")
```

이 코드가 오류 없이 실행되고 현재 LO 주파수 값을 성공적으로 출력한다면, ANTSDR E310으로의 PYNQ 포팅이 완벽하게 성공했음을 최종적으로 확인한 것입니다.

결론: **PYNQ**를 이용한 **SDR** 개발의 다음 단계

본 가이드는 Microphase ANTSDR E310이라는 강력한 SDR 하드웨어 플랫폼에 최신 고생산성 파이썬 프레임워크인 PYNQ를 성공적으로 이식하는 전 과정을 체계적으로 안내했습니다. 하드웨어 아키텍처 분석부터 시작하여, 정밀한 빌드 환경 구축, Vivado를 통한 하드웨어 정의, Petalinux를 이용한 커널 및 디바이스 트리 맞춤화, 그리고 PYNQ sdbuild를 통한 최종 이미지 통합에 이르기까지, 복잡한 포팅 프로세스를 단계별로 수행했습니다.

이 성공적인 포팅은 단순한 운영체제 설치를 넘어, ANTSDR E310의 잠재력을 최대한 활용할 수 있는 새로운 개발 패러다임을 열어줍니다. 이제 개발자들은 저수준의 C/C++나 HDL 코딩 없이도, 파이썬의 풍부한 라이브러리와 Jupyter Notebook의 상호작용 환경을 통해 SDR 시스템을 신속하게 프로토타이핑하고 개발할 수 있습니다.

향후 개발 방향은 다음과 같습니다.

- **PYNQ** 오버레이를 이용한 하드웨어 가속**:** PYNQ의 핵심 기능인 Overlay 클래스를 사용하여, FIR 필터, FFT, 변복조기(modulator/demodulator)와 같은 계산 집약적인 SDR 신호 처리 알고리즘을 Zynq SoC의 프로그래머블 로직(PL)에 하드웨어로 구현할 수 있습니다. 이를 통해 ARM 프로세서의 부담을 줄이고 실시간 고성능 신호 처리를 달성할 수 있습니다.
- **RF** 프레임워크 통합**:** iio-oscilloscope, GNURadio, srsRAN과 같은 기존 SDR 애플리케이션 및 프레임워크를 PYNQ 환경에 통합하여, ANTSDR E310을 더욱 전문적인 무선 통신 시스템 연구 및 개발 플랫폼으로 확장할 수 있습니다.
- 커뮤니티 기여**:** 본 포팅 과정에서 생성된 보드 저장소와 빌드 스크립트를 공개하고 공유함으로써, 다른 ANTSDR 사용자들이 PYNQ를 더 쉽게 시작할 수 있도록 기여하고, 커뮤니티 기반의 지속적인 개선을 이끌어낼 수 있습니다.

결론적으로, ANTSDR E310과 PYNQ의 결합은 하드웨어의 성능과 소프트웨어의 생산성이라는 두 마리 토끼를 모두 잡는 강력한 시너지 효과를 창출합니다. 이제 이 플랫폼을 기반으로 무선 통신, 스펙트럼 감지, 레이더 시스템 등 다양한 분야에서 혁신적인 아이디어를 현실로 구현할 수 있는 무한한 가능성이 열렸습니다.

참고 자료

1. ANTSDR Usage Guidelines — antsdr_doc_en v0.1 documentation, 9월 20, 2025에 액세스, https://antsdr-docs.microphase.cn/
2. Data Sheet - E300 - Magnum Electronics, Inc., 9월 20, 2025에 액세스, https://blog.magnumelectronics.com/wp-content/uploads/2020/04/e300-data-sheet.pdf
3. E300 and E302 Technical Specifications, 9월 20, 2025에 액세스, https://dox.packetpower.com/support/current/e300-and-e302-technical-specifications
4. MicroPhase ANTSDR E310 AD9363 SDR Development Board for ADI Pluto Communication- | eBay, 9월 20, 2025에 액세스, https://www.ebay.com/itm/126254547389
5. ANTSDR-E200 - Hackaday.io, 9월 20, 2025에 액세스, https://hackaday.io/project/188635-antsdr-e200
6. MicroPhase ANTSDR E310 Software Defined Radio transceiver ZYNQ 7000 SoC ADI AD9361 AD9363 MIMO SDR transmitter and receiver - AliExpress, 9월 20, 2025에 액세스, https://www.aliexpress.com/i/1005003181244737.html
7. AntSDR Series RF Modules - Mouser Electronics, 9월 20, 2025에 액세스, https://www.mouser.com/c/embedded-solutions/wireless-rf-modules/?series=AntSDR
8. AntSDR E200 - Crowd Supply, 9월 20, 2025에 액세스, https://www.crowdsupply.com/microphase-technology/antsdr-e200
9. PYNQ-Z1 Reference Manual - Digilent, 9월 20, 2025에 액세스, https://digilent.com/reference/programmable-logic/pynq-z1/reference-manual
10. Getting Started — NengoPYNQ 0.3.3.dev0 docs, 9월 20, 2025에 액세스, https://www.nengo.ai/nengo-pynq/getting-started.html
11. Change Log — Python productivity for Zynq (Pynq) - Read the Docs, 9월 20, 2025에 액세스, https://pynq.readthedocs.io/en/latest/changelog.html
12. Port PYNQ 3.01 to new board - Support, 9월 20, 2025에 액세스, https://discuss.pynq.io/t/port-pynq-3-01-to-new-board/5110
13. Releases · Xilinx/PYNQ - GitHub, 9월 20, 2025에 액세스, https://github.com/Xilinx/PYNQ/releases
14. PYNQ 3.0 kernel support ground (backward compile using PetaLinux 2021.1 or 2021.2), 9월 20, 2025에 액세스, https://discuss.pynq.io/t/pynq-3-0-kernel-support-ground-backward-compile-using-petalinux-2021-1-or-2021-2/4856
15. PYNQ 3.0.1 VirtualBox install and SD build - Learn, 9월 20, 2025에 액세스, https://discuss.pynq.io/t/pynq-3-0-1-virtualbox-install-and-sd-build/4859
16. PYNQv3.0.1 image for Custom Board using Petalinux, and host OS Ubuntu 22.04.5 LTS, 9월 20, 2025에 액세스, https://discuss.pynq.io/t/pynqv3-0-1-image-for-custom-board-using-petalinux-and-host-os-ubuntu-22-04-5-lts/8369
17. PYNQ SD Card image — Python productivity for Zynq (Pynq) - Read the Docs, 9월 20, 2025에 액세스, https://pynq.readthedocs.io/en/latest/pynq_sd_card.html

18. Pynq 2.6 custom board image build method that works - Learn, 9월 20, 2025에 액세스, https://discuss.pynq.io/t/pynq-2-6-custom-board-image-build-method-that-works/2894
19. Creating Pynq images for custom boards. - controlpaths.com, 9월 20, 2025에 액세스, https://www.controlpaths.com/2021/07/26/creating-pynq-images-for-custom-boards/
20. how to add ad9361 to petalinux2018.2 - Adaptive Support - AMD, 9월 20, 2025에 액세스, https://adaptivesupport.amd.com/s/question/0D52E00006hpROkSAM/how-to-add-ad9361-to-petalinux20182?language=en_US
21. Custom board device tree doubts : r/FPGA - Reddit, 9월 20, 2025에 액세스, https://www.reddit.com/r/FPGA/comments/1dv1taa/custom_board_device_tree_doubts/
22. petalinux 2021.2 driver for ad9361 - Q&A - Linux Software Drivers - EngineerZone, 9월 20, 2025에 액세스, https://ez.analog.com/linux-software-drivers/f/q-a/594694/petalinux-2021-2-driver-for-ad9361
23. ZC702 FMCOMMS3 PetaLinux Starting Guide - BijoKH, 9월 20, 2025에 액세스, https://bijan.binaee.com/2019/07/zc702-fmcomms3-petalinux-starting-guide/
24. ZYNQ Ultrascale+ and PetaLinux (part 17): Customizing Linux kernel and devicetree, 9월 20, 2025에 액세스, https://www.youtube.com/watch?v=WyVLb0ykaF4
25. Quick Porting of PYNQ using Pre-built Images - Learn, 9월 20, 2025에 액세스, https://discuss.pynq.io/t/quick-porting-of-pynq-using-pre-built-images/1075
26. PYNQ SD Card — Python productivity for Zynq (Pynq) v1.0 - Read the Docs, 9월 20, 2025에 액세스, https://pynq.readthedocs.io/en/v2.3/pynq_sd_card.html
27. PYNQ image for custom board make error - Support, 9월 20, 2025에 액세스, https://discuss.pynq.io/t/pynq-image-for-custom-board-make-error/1504
28. Getting Started with PYNQ using EDGE ZYNQ SoC FPGA kit - Invent Logics, 9월 20, 2025에 액세스, https://allaboutfpga.com/getting-started-with-pynq-using-edge-zynq-soc-fpga-kit/
29. bkerler/antsdr_new: Latest firmware for antsdr E310 based on PlutoSDR - GitHub, 9월 20, 2025에 액세스, https://github.com/bkerler/antsdr_new/
30. ANTSDR Unpacking and Test Rev. 1.1, 9월 20, 2025에 액세스, https://down.hgeek.com/download/93159/93159-E310-AD9361-Rev-1-1-English-Manual.pdf
31. PYNQ-Z1 Setup Guide - Read the Docs, 9월 20, 2025에 액세스, https://pynq.readthedocs.io/en/v2.3/getting_started/pynq_z1_setup.html
32. Python productivity for Zynq (Pynq) Documentation - Read the Docs, 9월 20, 2025에 액세스, https://buildmedia.readthedocs.org/media/pdf/pynq/v2.4/pynq.pdf