ADALM-PLUTO 펌웨어 빌드**:** 전체 프로세스에 대한 심층 기술 보고서

I. PlutoSDR 펌웨어의 기초 아키텍처

ADALM-PLUTO (이하 PlutoSDR) 펌웨어를 성공적으로 빌드하고 커스터마이징하기 위해서는, 단순히 일련의 명령어를 실행하는 것을 넘어 그 기반이 되는 하드웨어 및 소프트웨어 아키텍처에 대한 깊이 있는 이해가 선행되어야 한다. PlutoSDR의 핵심은 하드웨어(FPGA)와 소프트웨어(임베디드 리눅스)가 유기적으로 결합된 시스템이며, 펌웨어 빌드 프로세스는 이 두 세계를 통합하는 정교한 과정이다.

1.1 Zynq-7000 SoC: 이기종 컴퓨팅 코어

PlutoSDR의 심장부에는 Xilinx Zynq 7Z010 SoC(System-on-Chip)가 자리 잡고 있다.[1] Zynq는 일반적인 마이크로프로세서와는 근본적으로 다른 구조를 가진다. 이는 단일 칩 내에 두 개의 독립적인 컴퓨팅 도메인을 통합한 이기종(heterogeneous) 아키텍처를 특징으로 한다.

- 프로세싱 시스템 **(Processing System, PS):** 듀얼 코어 ARM Cortex-A9 프로세서로 구성되며, 임베디드 리눅스 운영체제, 부트로더, 사용자 애플리케이션 등 소프트웨어 스택 전체를 실행하는 역할을 담당한다. 이는 전통적인 임베디드 시스템의 두뇌에 해당한다.
- 프로그래머블 로직 **(Programmable Logic, PL):** FPGA(Field-Programmable Gate Array) 패브릭으로, 하드웨어 로직을 직접 설계하고 구현할 수 있는 유연한 영역이다.[2] PlutoSDR에서는 AD9363 RF 트랜시버와의 고속 데이터 인터페이스, 디지털 신호 처리 블록, 타이밍 제어 로직 등이 이 PL 영역에 구현된다.

이러한 PS-PL 분리 구조는 펌웨어 빌드 프로세스의 본질을 규정한다. 빌드 과정은 단순히 ARM용 C 코드를 컴파일하는 데 그치지 않는다. 이는 하드웨어와 소프트웨어의 공동 설계(Co-design) 워크플로우를 자동화하는 과정이다. 빌드 시스템은 먼저 Xilinx Vivado 도구를 호출하여 HDL(Hardware Description Language)로 작성된 PL 설계를 합성(synthesis)하고, 그 결과물로 FPGA 비트스트림(system_top.bit)과 하드웨어 정의 파일(.hdf 또는 .xsa)을 생성한다.[3] 이후, 이 하드웨어 정의 파일은 소프트웨어 빌드 단계의 입력으로 사용되어, PS에서 실행될

부트로더와 리눅스 커널이 PL에 어떤 하드웨어가 구성되었는지 정확히 인지하고 상호작용할 수 있도록 한다. 이처럼 하드웨어 설계의 결과물이 소프트웨어 빌드의 전제 조건이 되기 때문에, 특정 버전의 Xilinx 도구에 대한 의존성이 매우 높게 나타난다.

1.2 IIO 프레임워크: 커널과 물리적 세계의 가교

PlutoSDR의 소프트웨어 스택은 리눅스 커널의 핵심 기능 중 하나인 IIO(Industrial I/O) 프레임워크를 사용하여 RF 하드웨어와 사용자 공간 애플리케이션 간의 통신을 중재한다.[4] IIO는 센서, ADC, DAC 등 다양한 데이터 수집 및 생성 장치를 위한 표준화된 커널 내 인터페이스를 제공한다.

IIO 프레임워크는 PlutoSDR의 핵심 RF 칩인 AD9363 트랜시버를 추상화하는 역할을 한다. 이를 통해 GNU Radio [6], MATLAB [7], 또는

pyadi-iio를 사용한 파이썬 스크립트 [8]와 같은 고수준 애플리케이션이 복잡한 커널 드라이버 코드를 직접 작성하지 않고도 하드웨어의 기능을 제어(예: 중심 주파수, 샘플링 속도, 이득 설정)하고 I/Q 데이터를 송수신할 수 있다. 데이터 전송의 효율성은 DMA(Direct Memory Access) 버퍼를 통해 보장되며, 이는 고속 샘플링 속도에서 CPU의 개입을 최소화하여 데이터 손실 없는 연속적인 스트리밍을 가능하게 하는 핵심 기술이다.[5]

펌웨어 커스터마이징의 관점에서 IIO 프레임워크의 중요성은 더욱 부각된다. 만약 개발자가 FPGA(PL) 설계를 수정하여 새로운 신호 처리 블록을 추가했다면, 이 새로운 하드웨어 기능을 소프트웨어에서 사용하기 위해서는 해당 기능을 IIO 서브시스템에 등록하는 커널 드라이버를 수정하거나 새로 작성해야 한다. 즉, 고급 펌웨어 개발은 단순히 HDL 설계나 사용자 공간 애플리케이션 개발에 국한되지 않으며, IIO 프레임워크 내에서의 커널 드라이버 개발을 포함하는 포괄적인 작업이 된다.

1.3 plutosdr-fw Git 저장소 해부

PlutoSDR의 공식 펌웨어 소스 코드는 plutosdr-fw라는 이름의 Git 저장소를 통해 관리된다. 이 저장소는 단일 프로젝트가 아니라, 여러 독립적인 핵심 프로젝트들을 Git 서브모듈(submodule)로 포함하는 슈퍼 프로젝트(super-project) 구조를 가진다.[3] 이 구조를 이해하는 것은 빌드 프로세스 전체를 파악하는 데 필수적이다.

- **hdl:** Analog Devices가 제공하는 IP 코어(예: AXI_AD9361)와 FPGA 비트스트림을 생성하기 위한 Vivado 프로젝트 파일들을 포함한다.[10]
- **u-boot-xlnx:** U-Boot 부트로더의 포크 버전으로, 시스템의 초기 하드웨어 초기화, 리눅스

커널 로딩 등의 부팅 시퀀스를 담당한다.[4]
- **linux:** 리눅스 커널의 포크 버전으로, AD9363 트랜시버 및 Zynq SoC 내부의 다양한 주변 장치들을 위한 ADI 전용 드라이버가 포함되어 있다.[4]
- **buildroot:** 임베디드 리눅스 시스템을 구축하기 위한 강력한 도구로, C 라이브러리, 셸(Busybox), 그리고 선택된 모든 사용자 공간 애플리케이션을 소스 코드로부터 컴파일하여 최종 루트 파일 시스템(rootfs.cpio.gz)을 생성한다.[2]

이 서브모듈 구조에서 가장 중요한 점은, plutosdr-fw 저장소를 클론할 때 각 서브모듈은 최신 master 브랜치가 아닌, **특정 커밋 해시(commit hash)**로 고정되어 체크아웃된다는 것이다.[3] 이 커밋 해시들의 조합은 ADI 엔지니어들에 의해 상호 호환성이 검증된 "골든 구성(golden configuration)"을 의미한다. 이는 HDL 설계, U-Boot, 커널 드라이버, 사용자 공간 라이브러리 간의 미묘한 의존성 문제를 해결하고 빌드의 재현성(reproducibility)을 보장하기 위한 의도적인 설계 결정이다. 따라서 개발자가 임의로 서브모듈을 최신 버전으로 업데이트하는 행위(

git submodule update --remote)는 예기치 않은 빌드 실패를 초래할 가능성이 매우 높다. 펌웨어 수정은 이러한 복잡한 생태계 내의 상호 의존성을 명확히 이해한 상태에서 신중하게 이루어져야 한다.

II. 빌드 환경 구축: 정밀 가이드

PlutoSDR 펌웨어 빌드 환경은 매우 민감하며, 사소한 버전 불일치나 설정 오류가 전체 빌드 실패로 이어질 수 있다. 이 섹션에서는 안정적이고 정확한 개발 환경을 구축하기 위한 구체적이고 규범적인 절차를 제시한다.

2.1 호스트 시스템 구성: 리눅스는 필수 사항

PlutoSDR 펌웨어 빌드 시스템은 셸 스크립트, Makefile, 그리고 리눅스 네이티브 도구들에 깊이 의존하므로, 리눅스 개발 환경은 선택이 아닌 필수 사항이다. Windows 환경에서의 직접적인 크로스 컴파일은 공식적으로 지원되지 않으며, 호환되는 ARM 컴파일러를 구하기 어렵기 때문에 권장되지 않는다.[11]

최상의 호환성을 위해 **Ubuntu 22.04 LTS** 사용이 강력히 권장된다.[8] 개발 환경을 구축하는 방법은 다음과 같다.

- 가상 머신 **(VirtualBox):** 가장 일반적이고 격리된 환경을 제공한다. VirtualBox 설치 후 Ubuntu 22.04 LTS 데스크톱 이미지를 사용하여 새 가상 머신을 생성한다. 빌드 프로세스는 상당한 시스템 자원을 요구하므로, 가상 머신 설정 시 호스트 시스템 RAM의 50%와 최소

3개 이상의 CPU 코어를 할당하는 것이 좋다.[8]

- **Windows Subsystem for Linux (WSL):** Windows 11 사용자의 경우, WSL2를 통해 거의 네이티브에 가까운 리눅스 환경을 구축할 수 있어 편리한 대안이 된다.[8]

2.2 Xilinx Vivado Design Suite: 가장 중요한 의존성

펌웨어 빌드 과정에서 가장 빈번하게 실패를 유발하는 요인은 바로 Xilinx Vivado Design Suite의 버전 불일치이다. 최신 버전의 Vivado가 현재 빌드하려는 펌웨어 소스 코드와 호환될 것이라고 가정해서는 안 된다. 각 펌웨어 릴리스는 특정 Vivado 버전을 사용하여 개발 및 테스트되었으며, 해당 버전을 정확히 맞춰주는 것이 성공적인 빌드의 첫걸음이다.

펌웨어 버전과 그에 맞는 Vivado 버전 정보는 plutosdr-fw GitHub 저장소의 릴리스 노트에서 확인할 수 있다.[12] 다음 표는 주요 펌웨어 릴리스와 요구되는 Vivado 버전을 정리한 것이다. 빌드를 시작하기 전, 반드시 이 표를 참조하여 올바른 버전의 Vivado를 설치해야 한다.

표 **1: PlutoSDR** 펌웨어 및 툴체인 버전 매트릭스

펌웨어 릴리스 (Git 태그)	요구되는 Vivado 버전	하드웨어 정의 파일	비고
v0.39	2023.2	.xsa	최신 릴리스 (작성 시점 기준) [12]
v0.38	2022.2	.xsa	Linux 커널 5.15로 업데이트 [12]
v0.37	2021.2	.xsa	Linux 커널 5.10으로 업데이트 [12]
v0.35	2021.1	.xsa	[12]
v0.31	2019.1	.hdf	[14]
이전 버전	2016.4 등	.hdf	[4]

Vivado 설치 과정은 AMD-Xilinx 웹사이트에서 계정을 생성하고 해당 버전의 설치 파일을 다운로드해야 한다.[15] 설치 시 다음 옵션을 반드시 포함해야 한다.

- **Devices:** Zynq-7000 시리즈 SoC 지원을 반드시 선택해야 한다.[4]
- **Design Tools:** Software Development Kit (SDK)를 포함해야 한다.[15]

또한, 2019.2 버전 이후의 Vivado는 하드웨어 정의 파일을 기존의 .hdf 형식 대신 .xsa (Xilinx Shell Archive) 형식으로 내보낸다.[16] 펌웨어 빌드 스크립트는 이러한 변화에 맞춰 업데이트되었지만, 오래된 빌드 가이드를 최신 도구와 함께 사용하려는 경우 이 차이점을 인지하고 있어야 한다.

2.3 시스템 의존성 및 크로스 컴파일러 툴체인 설치

Vivado 외에도 빌드 프로세스는 다양한 시스템 라이브러리와 도구들을 필요로 한다. 새로 설치한 Ubuntu 22.04 시스템에서 다음 명령어를 실행하여 모든 필수 패키지를 한 번에 설치할 수 있다.

Bash

```
sudo apt-get install git build-essential fakeroot libncurses5-dev libssl-dev ccache \
dfu-util u-boot-tools device-tree-compiler libssl1.0-dev mtools bc python cpio \
zip unzip rsync file wget
```

3

PlutoSDR 펌웨어 빌드는 Vivado에 포함된 ARM GCC 툴체인 대신, 외부 Linaro ARM 크로스 컴파일러를 사용한다. 이는 Buildroot와의 호환성 문제 때문이며, 빌드 시스템의 중요한 아키텍처적 특징이다.[3] 특정 펌웨어 버전에 맞는 Linaro 툴체인(예: 7.x 버전)을 다운로드하여 설치하고, 해당 컴파일러의 경로를 시스템의

PATH 환경 변수에 추가해야 한다.[17]

2.4 펌웨어 소스 코드 획득

모든 환경 설정이 완료되면, 다음 명령어를 사용하여 공식 plutosdr-fw 저장소에서 소스 코드를 클론한다.

Bash

```
git clone --recursive https://github.com/analogdevicesinc/plutosdr-fw.git
cd plutosdr-fw
```

3

여기서 --recursive 플래그는 절대적으로 중요하다. 이 플래그는 메인 저장소뿐만 아니라 hdl, linux, buildroot, u-boot-xlnx와 같은 모든 서브모듈을 올바른 버전(커밋 해시)으로 함께 클론하도록 지시한다. 만약 이 플래그 없이 클론했다면, 서브모듈 디렉토리가 비어있게 되며 빌드는 즉시 실패한다. 이 경우, git submodule update --init --recursive 명령을 실행하여 서브모듈을 초기화하고 클론할 수 있다.

III. 펌웨어 컴파일 프로세스: 단계별 실행

성공적으로 구축된 개발 환경을 기반으로, 이제 실제 펌웨어 컴파일을 진행할 차례이다. 이 과정은 몇 가지 핵심 환경 변수를 설정하고 단일 make 명령을 실행하는 것으로 요약되지만, 그 내부에서는 복잡하고 순차적인 작업들이 수행된다.

3.1 필수 환경 변수 설정

make 명령을 실행하기 전에, 빌드 스크립트가 툴체인의 위치와 설정을 알 수 있도록 세 가지 중요한 환경 변수를 설정해야 한다.[4]

- CROSS_COMPILE: make가 네이티브 gcc 대신 ARM 크로스 컴파일러를 사용하도록 지시한다. 일반적으로 arm-linux-gnueabihf-로 설정된다.
- PATH: 시스템이 Linaro 크로스 컴파일러 바이너리를 찾을 수 있도록 경로를 포함해야 한다.
- VIVADO_SETTINGS: Vivado 설치 디렉토리 내의 settings64.sh 스크립트 경로를 지정한다. 이 스크립트는 Vivado가 GUI 없이 배치 모드로 실행될 수 있도록 필요한 환경을 설정한다.

다음은 일반적인 환경 변수 설정 예시이다. 사용자의 실제 설치 경로에 맞게 수정해야 한다.

Bash

```
# Unset any conflicting library paths
unset LD_LIBRARY_PATH

# Set the cross-compiler prefix
export CROSS_COMPILE=arm-linux-gnueabihf-

# Add the Linaro toolchain to the PATH (update with your actual path)
export PATH=$PATH:/path/to/your/gcc-linaro-7.x.x/bin

# Set the path to the Vivado settings script (update with your actual version and path)
export VIVADO_SETTINGS=/opt/Xilinx/Vivado/2022.2/settings64.sh
```

이 설정들을 매번 터미널을 열 때마다 입력하는 것은 번거로우므로, ~/.bashrc 파일에 추가하거나 별도의 환경 설정 스크립트 파일로 만들어 사용하는 것이 효율적이다.

3.2 빌드 시작 및 시퀀스 분석

환경 변수 설정이 완료되면, plutosdr-fw 저장소의 루트 디렉토리에서 다음의 단일 명령어로 전체 빌드 프로세스를 시작한다.

Bash

```
make
```

[4]

이 간단한 명령어 뒤에서는 최상위 Makefile에 의해 다음과 같은 정교한 작업들이 순차적으로 진행된다.

1. **HDL 합성:** make는 먼저 hdl 서브모듈 디렉토리로 이동하여, Tcl 스크립트를 통해 Vivado를 실행한다. 이 과정에서 FPGA 설계가 합성되고, 그 결과물로 하드웨어 정의 파일(system_top.hdf 또는 .xsa)과 FPGA에 직접 프로그래밍될 비트스트림 파일(system_top.bit)이 생성된다.[2]
2. 부트로더 빌드: 이전 단계에서 생성된 하드웨어 정의 파일을 입력으로 사용하여, 1단계

부트로더(FSBL)와 2단계 부트로더(U-Boot)를 컴파일한다.

3. 커널 컴파일**:** linux 서브모듈 디렉토리에서 리눅스 커널(zImage)과 PlutoSDR 하드웨어 리비전에 맞는 디바이스 트리 블롭(.dtb)을 컴파일한다.[3] 디바이스 트리는 커널에게 시스템의 하드웨어 구성을 알려주는 중요한 데이터 구조이다.
4. 루트 파일 시스템 생성**:** buildroot 서브모듈을 사용하여 전체 루트 파일 시스템을 처음부터 빌드한다. 이 과정에는 C 라이브러리, 기본 유틸리티(Busybox), 그리고 menuconfig에서 선택된 모든 추가 패키지들이 포함되며, 최종적으로 압축된 아카이브 파일(rootfs.cpio.gz)로 패키징된다.[3]
5. 최종 이미지 패키징**:** 마지막으로, 커널, 디바이스 트리, 루트 파일 시스템을 U-Boot가 로드할 수 있는 FIT(Flattened Image Tree) 이미지(pluto.itb)로 결합한다. 그리고 이 FIT 이미지와 부트로더들을 함께 묶어, 최종적으로 배포 가능한 펌웨어 파일인 pluto.frm과 pluto.dfu를 생성한다.[3]

3.3 최종 빌드 결과물 확인 및 식별

빌드가 성공적으로 완료되면, plutosdr-fw 프로젝트 루트 디렉토리 내에 build라는 이름의 디렉토리가 생성되며, 모든 최종 및 중간 결과물들이 이곳에 위치하게 된다.[3] 사용자가 주로 관심을 가져야 할 최종 결과물은 다음과 같다.

- pluto.frm: PlutoSDR을 USB 저장 장치로 인식시켜 펌웨어를 업데이트하는, 가장 일반적이고 간편한 방식에 사용되는 펌웨어 이미지 파일이다.[3]
- pluto.dfu: DFU(Device Firmware Update) 모드에서 사용되는 펌웨어 이미지 파일이다. 이는 장치가 정상적으로 부팅되지 않는 "벽돌" 상태가 되었을 때 복구하거나, 특정 저수준 업데이트를 진행할 때 사용된다.[3]

build 디렉토리는 단순한 결과물 저장소 이상의 의미를 가진다. 빌드 실패 시, 이 디렉토리의 상태는 훌륭한 진단 도구가 된다. 예를 들어, system_top.bit 파일이 없다면 HDL 합성 단계에서 문제가 발생한 것이고, zImage는 있지만 rootfs.cpio.gz가 없다면 Buildroot 단계에서 실패한 것이다. 따라서 빌드 실패 시 무조건 디렉토리를 삭제하기보다는, 내부를 검토하여 어느 단계에서 문제가 발생했는지 파악하는 것이 원인 해결에 큰 도움이 된다.

IV. 펌웨어 커스터마이징 및 기능 향상

단순히 공식 펌웨어를 재컴파일하는 것을 넘어, PlutoSDR의 동작을 변경하고 새로운 기능을 추가하는 것이 커스텀 펌웨어 빌드의 주된 목적이다. 빌드 시스템은 커널, 사용자 공간, 하드웨어 설정 등 다양한 계층에서 커스터마이징을 수행할 수 있는 강력한 인터페이스를

제공한다.

4.1 make linux-menuconfig를 통한 커널 및 드라이버 설정

리눅스 커널의 동작 방식을 변경하거나 특정 드라이버를 추가/제거하고 싶을 때, 다음 명령어를 사용한다.

Bash

```
make linux-menuconfig
```

[14]

이 명령어는 커널 소스 디렉토리에서 텍스트 기반의 설정 인터페이스(menuconfig)를 실행한다. 이를 통해 개발자는 실시간(real-time) 애플리케이션을 위해 커널 선점 모델(Preemption Model)을 변경하거나, 커스텀 네트워킹 애플리케이션에 필요한 TUN/TAP 드라이버와 같은 특정 커널 모듈을 활성화할 수 있다.[14] 설정 변경 후에는

.config 파일을 저장해야 하며, 이 파일은 다음 make 실행 시 커널을 컴파일하는 데 사용된다.

4.2 make buildroot-menuconfig를 통한 사용자 공간 애플리케이션 관리

PlutoSDR의 루트 파일 시스템에 포함될 소프트웨어 패키지를 관리하기 위해서는 다음 명령어를 사용한다.

Bash

```
make buildroot-menuconfig
```

[14]

이 명령어는 Buildroot의 설정 인터페이스를 실행하며, 이를 통해 수천 개의 사용 가능한 패키지 중에서 원하는 것을 선택하여 펌웨어에 포함시킬 수 있다. 예를 들어, 네트워크 대역폭 테스트를 위한 iperf3, 패킷 분석을 위한 tcpdump와 같은 시스템 유틸리티나, Python 인터프리터 자체를 추가할 수 있다.[14]

4.2.1 사례 연구**:** 루트 파일 시스템에 커스텀 패키지 추가하기

Buildroot의 강력한 기능 중 하나는 BR2_EXTERNAL 메커니즘을 통해 외부의 커스텀 패키지를 손쉽게 통합하는 것이다.[19] 다음은 간단한 "hello world" C 애플리케이션을 새로운 패키지로 추가하는 과정을 요약한 것이다.

1. 외부 디렉토리 생성**:** plutosdr-fw 외부에 커스텀 패키지를 위한 디렉토리(예: pluto_external)를 생성한다.
2. 패키지 파일 작성**:** pluto_external/package/myapp/ 디렉토리 내에 다음 두 파일을 생성한다.
   - Config.in: menuconfig에 표시될 옵션을 정의한다.[20]

     ```
     config BR2_PACKAGE_MYAPP
       bool "myapp"
       help
         My custom hello world application.
     ```

   - myapp.mk: 패키지의 버전, 소스 위치, 빌드 및 설치 방법을 정의하는 Makefile이다.[22]

     ```
     Makefile
     MYAPP_VERSION = 1.0
     MYAPP_SITE = /path/to/myapp/source/code
     MYAPP_SITE_METHOD = local

     define MYAPP_BUILD_CMDS
       $(MAKE) $(TARGET_CONFIGURE_OPTS) -C $(@D)
     endef

     define MYAPP_INSTALL_TARGET_CMDS
       $(INSTALL) -D -m 0755 $(@D)/myapp $(TARGET_DIR)/usr/bin/myapp
     endef

     $(eval $(generic-package))
     ```

3. **Buildroot**에 등록**:** make BR2_EXTERNAL=/path/to/pluto_external menuconfig 명령으로 Buildroot 설정을 실행하고, "External options" 메뉴에서 "myapp"을 활성화한다.
4. 빌드 및 확인**:** 전체 펌웨어를 다시 빌드(make)한 후, 생성된 루트 파일 시스템(build/target/usr/bin/) 내에 myapp 실행 파일이 포함되었는지 확인한다.

이러한 과정을 통해 어떤 커스텀 애플리케이션이든 체계적으로 펌웨어에 통합할 수 있다.

4.3 잠재력 해제: 트랜시버 프로파일 변경 (AD9363에서 AD9364로)

PlutoSDR의 가장 인기 있는 수정 사항은 내장된 AD9363 트랜시버를 상위 모델인 AD9364처럼 동작하도록 변경하는 것이다. AD9363은 공식적으로 325 MHz ~ 3.8 GHz의 주파수 범위를 지원하지만 [7], 하드웨어적으로 매우 유사한 AD9364는 70 MHz ~ 6 GHz의 훨씬 넓은 범위를 지원한다.[25]

간단한 소프트웨어 명령을 통해 드라이버가 칩을 AD9364로 인식하게 만들 수 있으며, 이를 통해 공식 사양을 넘어서는 주파수 범위와 최대 56 MHz의 대역폭을 사용할 수 있게 된다.[8] 이 작업은 PlutoSDR에 SSH로 접속한 후, U-Boot 환경 변수를 설정하는

fw_setenv 명령어를 통해 수행된다.

- 펌웨어 **v0.31** 이하:
  Bash
  ```
  fw_setenv attr_name compatible
  fw_setenv attr_val ad9364
  reboot
  ```
  [8]
- 펌웨어 **v0.32** 이상:
  Bash
  ```
  fw_setenv compatible ad9364
  reboot
  ```
  [8]

이 변경의 기저에는 소프트웨어를 통한 제품 등급화 전략이 있다. ADI는 저렴한 PlutoSDR에 AD9363을 사용하면서도, 소프트웨어적으로 AD9364의 기능을 활성화할 수 있는 경로를 남겨둠으로써 교육 및 취미 시장에 강력한 도구를 제공하는 동시에 공식 제품 라인업을 유지했다. 이 "핵"은 우연한 버그가 아니며, 이러한 소프트웨어 정의 기능(software-defined capability)을 활용하는 것이다. 단, AD9363의 공식 사양을 벗어나는 범위에서의 성능은 ADI가 보증하지 않는다는 점을 명확히 인지해야 한다.[25]

4.4 네트워크 설정: PlutoSDR의 IP 주소 영구 변경

PlutoSDR은 기본적으로 192.168.2.1이라는 고정 IP 주소를 사용한다.[8] 이는 단일 장치를 사용할 때는 문제가 없지만, 여러 대의 PlutoSDR을 동일 네트워크에 연결하거나 기존의 192.168.2.0/24 서브넷과 충돌이 발생할 경우 변경이 필요하다.[27]

IP 주소는 PlutoSDR을 USB로 연결했을 때 나타나는 대용량 저장 장치 내의 config.txt 파일을 편집하여 변경할 수 있다.[8]

Ini, TOML

```
hostname = pluto
ipaddr = 192.168.3.1
ipaddr_host = 192.168.3.10
netmask = 255.255.255.0
```

여기서 ipaddr는 PlutoSDR 장치 자체의 IP 주소이고, ipaddr_host는 USB 이더넷 링크의 호스트 측에 할당될 IP 주소이다.[27] 파일을 수정한 후에는 장치를 물리적으로 분리하지 말고, 운영체제의 "꺼내기(eject)" 기능을 사용하여 논리적으로만 연결을 해제해야 한다. 그러면 PlutoSDR이 설정을 적용하고 재부팅한다.[8]

V. 배포 및 검증

커스텀 펌웨어를 성공적으로 빌드했다면, 마지막 단계는 이를 실제 장치에 설치하고 변경 사항을 확인하는 것이다. PlutoSDR은 두 가지 주요 펌웨어 업데이트 방법을 제공한다.

5.1 펌웨어 업데이트 방법론

5.1.1 대용량 저장 장치 방식 (권장)

이것은 가장 간단하고 안전한 표준 업데이트 방법이다.

1. PlutoSDR을 USB 케이블로 호스트 컴퓨터에 연결한다. 장치가 USB 드라이브와 같은 대용량 저장 장치로 인식된다.[24]
2. 빌드 결과물인 pluto.frm 파일을 이 드라이브에 복사한다.[28]
3. 파일 복사가 완료되면, 운영체제의 "안전하게 제거" 또는 "꺼내기" 기능을 사용하여 드라이브를 논리적으로 분리한다. 절대로 **USB** 케이블을 바로 뽑아서는 안 된다**.** [30]
4. 꺼내기 작업이 완료되면 PlutoSDR의 LED가 몇 분간 빠르게 깜박이기 시작한다. 이는 내부 QSPI 플래시 메모리에 새로운 펌웨어를 기록하는 과정이다.[24]
5. LED 깜박임이 멈추고 장치가 자동으로 재부팅되면 업데이트가 완료된 것이다.

5.1.2 DFU(Device Firmware Update) 모드 (복구용)

DFU는 펌웨어가 손상되어 장치가 정상적으로 부팅되지 않는 "벽돌" 상태일 때 사용하는 저수준 복구 모드이다.

1. DFU 모드로 진입한다. 이는 장치의 전원을 연결하는 동안 리셋 버튼을 누르고 있거나, 정상 부팅된 상태에서 SSH로 접속하여 pluto_reboot ram 또는 pluto_reboot sf 명령을 실행하여 진입할 수 있다.[2]
2. 호스트 컴퓨터에 dfu-util 명령줄 도구를 설치한다.[3]
3. 다음 명령어를 사용하여 빌드 결과물인 pluto.dfu 파일을 장치에 플래싱한다.
   Bash
   dfu-util -d 0456:b673 -a firmware.dfu -D pluto.dfu

   2

5.2 배포 후 검증

펌웨어 업데이트 및 재부팅이 완료된 후, 변경 사항이 올바르게 적용되었는지 확인해야 한다.

1. 네트워크 연결 확인**:** PlutoSDR이 호스트에 USB 이더넷 어댑터로 인식되었는지 확인한다. ping 명령을 사용하여 장치의 IP 주소(기본값 192.168.2.1)로 기본 네트워크 연결을 테스트한다.[8]
2. **SSH** 접속**:** ping이 성공하면, SSH 클라이언트를 사용하여 장치에 접속한다. 기본 사용자 이름은 root, 비밀번호는 analog이다.[8]
   Bash
   ssh root@192.168.2.1

3. 변경 사항 검증**:** SSH로 접속한 후, 다양한 명령어를 통해 커스텀 빌드의 결과를 확인할 수 있다.
    - uname -a: 업데이트된 커널 버전을 확인한다.
    - ls /usr/bin: 새로 추가한 커스텀 애플리케이션이 존재하는지 확인한다.
    - iio_attr -d ad9361-phy | grep compatible: 트랜시버가 ad9364로 인식되는지 확인하여 주파수 확장 "핵"이 적용되었는지 검증한다.

VI. 일반적인 빌드 및 배포 문제 해결

PlutoSDR 펌웨어 빌드는 복잡한 과정인 만큼 다양한 문제에 직면할 수 있다. 이 섹션에서는 가장 흔하게 발생하는 문제들의 원인과 해결책을 체계적으로 정리한다.

6.1 툴체인 및 의존성 버전 충돌 해결

- 문제**:** make 실행 시 Vivado, GCC, 또는 특정 라이브러리와 관련된 오류 메시지와 함께 빌드가 실패한다.
- 근본 원인**:** 이 문제는 빌드 실패의 가장 흔한 원인이다. 사용 중인 Vivado 또는 Linaro 툴체인의 버전이 현재 빌드하려는 plutosdr-fw 소스 코드 버전과 호환되지 않기 때문이다.[12]
- 해결책**:**
    1. 본 보고서의 **표 1 (펌웨어 및 툴체인 버전 매트릭스)**을 참조하여, 빌드하려는 펌웨어 태그에 맞는 정확한 버전의 Vivado를 설치한다.
    2. plutosdr-fw 저장소의 문서를 확인하여 권장되는 Linaro GCC 버전을 확인하고 설치한다.
    3. 툴체인 버전을 수정한 후에는, 이전 빌드의 잔여 파일과의 충돌을 막기 위해 빌드 디렉토리를 완전히 정리하는 것이 좋다. make clean 또는 더 확실한 make distclean 명령을 사용한다.[33]

6.2 make 실패 진단 및 빌드 로그 해석

- 문제**:** make 명령이 중간에 알 수 없는 오류와 함께 중단된다.
- 근본 원인**:** 잘못된 환경 변수 설정, 소스 코드 수정 중 발생한 버그, 시스템 라이브러리 부족 등 원인은 매우 다양하다.
- 해결책**:** 체계적인 진단이 필요하다.

1. 실패 단계 식별**:** 빌드 로그의 마지막 수십 줄을 확인하여 어떤 서브모듈(예: Entering directory '.../buildroot')을 빌드하던 중 실패했는지 파악한다.
2. 빌드 디렉토리 검사**:** build/ 디렉토리를 검사하여 어떤 중간 결과물까지 성공적으로 생성되었는지 확인한다. 이는 실패 단계를 재확인하는 데 도움이 된다 (예: system_top.bit는 있으나 zImage가 없는 경우, HDL 합성은 성공했으나 커널 컴파일에서 실패).
3. 문제 분리 및 재실행**:** 실패가 발생한 서브모듈 디렉토리로 직접 이동하여 make를 실행하면(예: cd buildroot; make), 더 구체적이고 집중된 오류 메시지를 얻을 수 있어 문제 해결에 도움이 된다.

6.3 펌웨어 업데이트 실패 처리

- 문제**:** pluto.frm 파일을 복사하고 꺼내기를 실행했으나 LED가 깜박이지 않거나, 체크섬(checksum) 오류가 발생했다는 메시지가 나타난다.
- 근본 원인**:** 다운로드 또는 빌드된 pluto.frm 파일 자체의 손상, 부적절한 "꺼내기" 절차, 또는 초기 PlutoSDR 하드웨어 리비전의 알려진 버그일 수 있다.[34]
- 해결책**:** 다음 사항들을 순서대로 점검한다.
  1. 빌드 시 함께 생성된 pluto.frm.md5 파일의 내용과 실제 pluto.frm 파일의 MD5 체크섬을 비교하여 파일이 손상되지 않았는지 확인한다.[3]
  2. 호스트 운영체제(특히 Windows)는 파일 쓰기를 캐싱할 수 있으므로, 파일 복사 직후 USB 케이블을 분리하지 말고 반드시 "안전하게 제거" 또는 "꺼내기" 절차를 따른다.[30]
  3. 전원 부족 문제일 가능성을 배제하기 위해 다른 USB 포트에 연결하거나, 별도의 전원이 공급되는 USB 허브를 사용해 본다.[37]
  4. 위의 방법으로도 해결되지 않으면, 더 저수준이고 강력한 업데이트 방식인 DFU 모드를 사용해 본다.

6.4 호스트**-**장치 간 연결 문제 해결

- 문제**:** 펌웨어 업데이트는 성공적으로 완료된 것으로 보이나, 장치의 IP 주소로 ping 또는 SSH 접속이 불가능하다.
- 근본 원인**:** 대부분 호스트 측의 네트워크 설정 문제이다. 호스트 운영체제가 PlutoSDR의 USB 이더넷 어댑터를 올바르게 인식하지 못했거나, 필요한 드라이버(예: macOS의 HoRNDIS)가 설치되지 않았을 수 있다.[8] 또한 호스트의 방화벽이 연결을 차단하거나, IP 주소 충돌이 발생했을 수도 있다.[29]
- 해결책**:** 체계적인 네트워크 디버깅을 수행한다.
  1. 물리 계층 확인**:** 리눅스에서 lsusb 또는 Windows의 장치 관리자에서 PlutoSDR 장치가 올바르게 인식되는지 확인한다.

2. 네트워크 인터페이스 확인**:** ifconfig (구형) 또는 ip addr (신형) 명령을 사용하여 새로운 네트워크 인터페이스가 생성되었는지 확인한다.
3. **IP** 설정 확인**:** 호스트 측의 해당 인터페이스가 PlutoSDR과 동일한 서브넷에 속하도록 IP 주소가 올바르게 설정되었는지 확인한다(예: PlutoSDR이 192.168.2.1일 경우, 호스트 인터페이스는 192.168.2.10으로 설정). 필요한 경우 수동으로 IP를 할당한다.[29]
4. **Ping** 테스트**:** ping 192.168.2.1을 실행하여 기본적인 L3 연결성을 테스트한다.[8] Ping이 실패하면 SSH는 당연히 실패하므로, 네트워크 계층의 문제를 먼저 해결해야 한다.

VII. 결론

Analog Devices의 ADALM-PLUTO 펌웨어를 빌드하는 과정은 단순한 컴파일 작업을 넘어, Zynq SoC의 이기종 아키텍처 위에서 하드웨어(FPGA)와 소프트웨어(임베디드 리눅스)를 통합하는 정교한 시스템 엔지니어링 활동이다. 성공적인 빌드와 커스터마이징을 위해서는 plutosdr-fw 저장소의 서브모듈 구조, Buildroot를 통한 루트 파일 시스템 구축, 그리고 Xilinx Vivado 툴체인과의 엄격한 버전 의존성에 대한 명확한 이해가 필수적이다.

본 보고서는 빌드 환경 구축부터 소스 코드 획득, 컴파일, 커스터마이징, 배포 및 문제 해결에 이르는 전 과정을 체계적으로 안내하였다. 특히, 펌웨어 버전과 Vivado 버전 간의 호환성 매트릭스를 제시하고, make menuconfig를 통한 커널 및 사용자 공간 커스터마이징 방법을 상세히 설명함으로써, 개발자가 단순한 재빌드를 넘어 능동적으로 펌웨어의 기능을 확장하고 최적화할 수 있는 기반을 제공하고자 했다. AD9364 프로파일 활성화와 같은 핵심적인 기능 향상 기법과 일반적인 문제 해결 절차를 포함함으로써, 본 문서는 PlutoSDR을 활용하는 연구자, 엔지니어, 그리고 고급 사용자들이 겪을 수 있는 잠재적인 어려움을 최소화하고 장치의 모든 잠재력을 끌어내는 데 기여할 수 있을 것이다. 결국, PlutoSDR의 진정한 가치는 이처럼 개방된 소스 코드와 강력한 빌드 시스템을 통해 사용자가 자신의 필요에 맞게 시스템을 재구성하고 실험할 수 있는 무한한 가능성에서 비롯된다.

참고 자료

1. How to Program the Xilinx Zynq XC 7Z010 inside Adalm Pluto SDR - Adaptive Support, 9월 22, 2025에 액세스, https://adaptivesupport.amd.com/s/question/0D54U000067pKz8SAE/how-to-program-the-xilinx-zynq-xc-7z010-inside-adalm-pluto-sdr?language=en_US
2. Private LTE with Analog ADALM-PLUTO - Quantulum Ltd, 9월 22, 2025에 액세스, https://www.quantulum.co.uk/blog/private-lte-with-analog-adalm-pluto/
3. analogdevicesinc/plutosdr-fw: PlutoSDR Firmware - GitHub, 9월 22, 2025에 액세스, https://github.com/analogdevicesinc/plutosdr-fw
4. ADALM-PLUTO INTRODUCTION - Previous FOSDEM Editions, 9월 22, 2025에 액세스, https://archive.fosdem.org/2018/schedule/event/plutosdr/attachments/slides/2503

/export/events/attachments/plutosdr/slides/2503/pluto_stupid_tricks.pdf
5. ARM PlutoSDR With Custom Applications GNU Radio Conference 2018, 9월 22, 2025에 액세스, https://www.gnuradio.org/grcon/grcon18/presentations/PlutoSDR/8-Michael_Henerich.pdf
6. Lab 2 - Getting Started on PlutoSDR - Codinghub, 9월 22, 2025에 액세스, https://codinghub.sellfy.store/p/lab-2-getting-started-on-plutosdr/
7. ADALM-PLUTO Evaluation Board - Analog Devices, 9월 22, 2025에 액세스, https://www.analog.com/en/resources/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html
8. PlutoSDR in Python | PySDR: A Guide to SDR and DSP using Python, 9월 22, 2025에 액세스, https://pysdr.org/content/pluto.html
9. Examples — Analog Devices Hardware Python Interfaces 0.0.19 documentation, 9월 22, 2025에 액세스, https://analogdevicesinc.github.io/pyadi-iio/guides/examples.html
10. PLUTO HDL Project — HDL documentation - Repositories, 9월 22, 2025에 액세스, https://analogdevicesinc.github.io/hdl/projects/pluto/index.html
11. how to cross-compile from windows 11? - Q&A - Virtual Classroom for ADI University Program - EngineerZone, 9월 22, 2025에 액세스, https://ez.analog.com/adieducation/university-program/f/q-a/565301/how-to-cross-compile-from-windows-11
12. Releases · analogdevicesinc/plutosdr-fw - GitHub, 9월 22, 2025에 액세스, https://github.com/analogdevicesinc/plutosdr-fw/releases
13. Private LTE with Analog ADALM-PLUTO v0.38 Update - Quantulum Ltd, 9월 22, 2025에 액세스, https://www.quantulum.co.uk/blog/private-lte-with-analog-adalm-pluto-v0-38-update/
14. Building the Firmware - manual.hnap.de, 9월 22, 2025에 액세스, https://manual.hnap.de/building_the_fw/
15. Installing Vivado, Xilinx SDK, and Digilent Board Files, 9월 22, 2025에 액세스, https://digilent.com/reference/programmable-logic/guides/installing-vivado-and-sdk
16. How to make custom Firmware for PLUTOSDR for my Vhdl Design - EngineerZone, 9월 22, 2025에 액세스, https://ez.analog.com/adieducation/university-program/f/q-a/598343/how-to-make-custom-firmware-for-plutosdr-for-my-vhdl-design
17. Build apps without firmware - manual.hnap.de, 9월 22, 2025에 액세스, https://manual.hnap.de/build_apps/
18. Error during the building of the Pluto firmware - Q&A - Virtual Classroom for ADI University Program - EngineerZone, 9월 22, 2025에 액세스, https://ez.analog.com/adieducation/university-program/f/q-a/550251/error-during-the-building-of-the-pluto-firmware
19. BR2_EXTERNAL framework for Analog Device's PlutoSDR Zynq - GitHub, 9월 22, 2025에 액세스, https://github.com/oscimp/PlutoSDR
20. The First Steps With Buildroot - ejaaskel, 9월 22, 2025에 액세스,

https://ejaaskel.dev/the-first-steps-with-buildroot/
21. 4. Building Buildroot Packages - http - Texas Instruments, 9월 22, 2025에 액세스, https://software-dl.ti.com/processor-sdk-linux/esd/AM62X/11_00_09_04_Buildroot/exports/docs/buildroot/Building_Buildroot_Packages.html
22. Adding new packages to Buildroot - Bootlin, 9월 22, 2025에 액세스, https://bootlin.com/~thomas/site/buildroot/adding-packages.html
23. adding-packages-autotools.txt - Buildroot, 9월 22, 2025에 액세스, https://buildroot.org/downloads/manual/adding-packages-autotools.txt
24. SDR Adalm-Pluto setup - F1ATB, 9월 22, 2025에 액세스, https://f1atb.fr/sdr-adalm-pluto-setup/
25. ADALM-PLUTO SDR Hack: Tune 70 MHz to 6 GHz and GQRX Install, 9월 22, 2025에 액세스, https://www.rtl-sdr.com/adalm-pluto-sdr-hack-tune-70-mhz-to-6-ghz-and-gqrx-install/
26. configurePlutoRadio - Configure ADALM-PLUTO radio firmware - MATLAB - MathWorks, 9월 22, 2025에 액세스, https://www.mathworks.com/help/comm/plutoradio/ref/configureplutoradio.html
27. Customizing the Pluto configuration [Analog Devices Wiki], 9월 22, 2025에 액세스, https://www.gods69.com/download/Pluto_SDR/Customizing%20the%20Pluto%20configuration%20%5BAnalog%20Devices%20Wiki%5D.pdf
28. OpenCPI Plutosdr Getting Started Guide, 9월 22, 2025에 액세스, https://opencpi.gitlab.io/releases/v2.3.4/docs/osp_plutosdr/Plutosdr_Getting_Started_Guide.pdf
29. cannot get ADALM-PLUTO on the network - Q&A - Virtual Classroom for ADI University Program - EngineerZone, 9월 22, 2025에 액세스, https://ez.analog.com/adieducation/university-program/f/q-a/589803/cannot-get-adalm-pluto-on-the-network
30. Lost Config.txt Settings - ADALM-PLUTO - AMSAT-DL Forum, 9월 22, 2025에 액세스, https://forum.amsat-dl.org/index.php?thread/3704-lost-config-txt-settings/
31. Installation - Maia SDR, 9월 22, 2025에 액세스, https://maia-sdr.org/installation/
32. How to Update the Firmware of Adalm Pluto - gpspatron, 9월 22, 2025에 액세스, https://support.gpspatron.com/support/solutions/articles/101000536910-how-to-update-the-firmware-of-adalm-pluto
33. ADALM-Pluto: removing package installed using buildroot - EngineerZone - Analog Devices, 9월 22, 2025에 액세스, https://ez.analog.com/adieducation/university-program/f/q-a/104397/adalm-pluto-removing-package-installed-using-buildroot
34. New Custom Firmware for the PlutoSDR with Several Linux SDR Programs Pre-Installed, 9월 22, 2025에 액세스, https://www.rtl-sdr.com/new-custom-firmware-for-the-plutosdr-with-several-linux-sdr-programs-pre-installed/
35. ADALM-PLUTO firmware update fails - Q&A - Virtual Classroom for ADI University Program, 9월 22, 2025에 액세스, https://ez.analog.com/adieducation/university-program/f/q-a/91484/adalm-pluto-f

irmware-update-fails
36. actionchen/plutosdr-fw - Gitee, 9월 22, 2025에 액세스, https://gitee.com/actionchen/plutosdr-fw
37. Common Problems and Fixes - MATLAB & Simulink - MathWorks, 9월 22, 2025에 액세스, https://www.mathworks.com/help/comm/plutoradio/ug/common-problems-and-fixes.html