ADALM-PLUTO 리눅스 펌웨어의 ANTSDR E310 플랫폼 포팅에 대한 기술 보고서

1.0 개요: PlutoSDR에서 고성능 하드웨어 플랫폼으로의 확장

Analog Devices(ADI)의 ADALM-PLUTO는 소프트웨어 정의 라디오(SDR) 기술의 접근성을 획기적으로 개선한 교육 및 개발용 플랫폼으로 평가받는다.[1] 이 장치는 AD9363 RF 트랜시버와 Xilinx Zynq Z-7010 시스템 온 칩(SoC)을 기반으로 하며, USB 전원만으로 동작하는 독립적인 임베디드 리눅스 시스템을 내장하고 있다.[3] 합리적인 가격과 방대한 소프트웨어 생태계(MATLAB, Simulink, GNU Radio 등) 덕분에 ADALM-PLUTO는 SDR 입문자부터 전문가까지 폭넓은 사용자층을 확보하며 RF 실험의 대중화를 이끌었다.[1]

이러한 PlutoSDR의 성공적인 생태계를 기반으로, 하드웨어 성능을 대폭 향상시킨 호환 기기들이 등장하기 시작했다. 그중 ANTSDR E310은 PlutoSDR과의 호환성을 유지하면서도 핵심 부품을 업그레이드하여 더 높은 성능과 확장성을 제공하는 플랫폼이다.[7] 본 보고서의 목적은 ADALM-PLUTO의 오픈소스 리눅스 펌웨어를 ANTSDR E310 하드웨어에 성공적으로 이식(porting)하는 전 과정을 상세히 기술하는 종합 엔지니어링 가이드를 제공하는 것이다. 이를 통해 사용자는 기존 PlutoSDR의 익숙한 소프트웨어 환경을 유지하면서 ANTSDR E310의 우수한 하드웨어 성능을 최대한 활용할 수 있게 될 것이다.

포팅 과정은 단순히 펌웨어를 복사하는 수준을 넘어, 두 플랫폼 간의 하드웨어 차이를 정확히 분석하고 이를 소프트웨어에 반영하는 체계적인 접근을 요구한다. 주요 단계는 다음과 같다: (1) 하드웨어 사양 비교 분석, (2) 크로스 컴파일 개발 환경 구축, (3) 소스 코드 수정(FPGA HDL, U-Boot 부트로더, 리눅스 커널 및 디바이스 트리), (4) 펌웨어 전체 빌드, (5) 최종 펌웨어 배포 및 시스템 검증. 이 과정은 제조사에서 제공하는 공식 포팅용 리소스를 기반으로 하므로, 무에서 유를 창조하는 것이 아닌 검증된 절차를 체계적으로 이해하고 적용하는 과정에 가깝다.[10]

2.0 기반 하드웨어 분석: 두 Zynq 플랫폼의 비교

성공적인 펌웨어 포팅의 첫걸음은 원본(ADALM-PLUTO)과 대상(ANTSDR E310) 플랫폼 간의 하드웨어 차이점을 명확히 이해하는 것이다. 모든 소프트웨어 수정 작업은 이러한 하드웨어의 차이에서 비롯되기 때문이다. 두 장치는 모두 Xilinx Zynq-7000 SoC와 ADI의 RF 트랜시버를 기반으로 하지만, 세부 사양에서는 포팅 작업의 방향을 결정하는 중요한 차이점들이 존재한다.

2.1 시스템 온 칩(SoC): Zynq Z-7010 대 Z-7020

ADALM-PLUTO는 Xilinx Zynq Z-7010 SoC를 탑재하고 있다. 이 칩은 단일 코어 ARM Cortex-A9 프로세서와 28k개의 로직 셀(Logic Cell), 80개의 DSP 슬라이스를 포함하는 프로그래머블 로직(PL)으로 구성된다.[3] 반면, ANTSDR E310은 상위 모델인 Zynq Z-7020을 채택했다. Z-7020은 듀얼 코어 ARM Cortex-A9 프로세서와 85k개의 로직 셀, 220개의 DSP 슬라이스를 제공한다.[7]

이러한 업그레이드는 단순한 성능 향상 이상의 의미를 갖는다. 프로그래머블 로직 리소스가 약 3배 증가함에 따라 [11], 사용자는 더 복잡하고 정교한 신호 처리 알고리즘(예: 맞춤형 필터, 변복조기, FFT 엔진 등)을 FPGA 패브릭에 직접 구현할 수 있는 여유를 확보하게 된다. 펌웨어 포팅 관점에서 이는 Vivado 프로젝트의 타겟 디바이스를 Z-7010에서 Z-7020으로 변경하는 작업이 필요함을 의미한다.

2.2 메모리 서브시스템: DDR3L 및 QSPI 플래시

메모리 용량은 임베디드 시스템의 성능을 좌우하는 핵심 요소 중 하나다. ADALM-PLUTO는 512MB의 DDR3L RAM과 32MB(256Mbit)의 QSPI 플래시 메모리를 장착하고 있다.[3] ANTSDR E310은 이보다 두 배 많은 1GB의 DDR3L RAM을 탑재했으며, QSPI 플래시 용량은 32MB로 동일하다.[7]

RAM 용량의 증가는 U-Boot 부트로더의 메모리 맵 설정과 리눅스 커널의 디바이스 트리(Device Tree) 수정을 요구한다. 수정 없이는 운영체제가 추가된 512MB의 메모리 공간을 인식하고 활용할 수 없다. 따라서 포팅 과정에서 부트로더와 커널이 전체 1GB의 물리 메모리 주소 공간을 올바르게 인지하도록 설정하는 작업이 필수적이다.

2.3 RF 프론트엔드: 1T1R 대 2T2R MIMO

두 장치 간의 가장 중요한 기능적 차이는 RF 채널 구성에 있다. ADALM-PLUTO는 1개의 송신(TX) 채널과 1개의 수신(RX) 채널(1T1R)을 제공하며, 반이중(half-duplex) 또는 전이중(full-duplex) 모드로 동작한다.[1]

이에 반해 ANTSDR E310은 2개의 송신 채널과 2개의 수신 채널(2T2R)을 지원하여 진정한 MIMO(Multiple-Input Multiple-Output) 시스템 구현이 가능하다.[7] 두 장치 모두 AD9363 RF 트랜시버를 사용할 수 있지만, ANTSDR E310은 보드 설계를 통해 AD9363의 두 번째 RF 채널 쌍을 물리적으로 외부와 연결했다. 이 핵심적인 하드웨어 개선 사항을 활용하기 위해서는 펌웨어 수정이 반드시 필요하다. 구체적으로는 U-Boot 환경 변수와 리눅스 디바이스 트리 속성을 통해 AD936x 드라이버가 2R2T 모드로 동작하도록 설정해야 한다.[10]

2.4 I/O 및 주변장치: USB 전용 대 확장된 연결성

연결성 측면에서 두 플랫폼은 큰 차이를 보인다. ADALM-PLUTO는 전원 공급, 데이터 통신(USB 이더넷 장치), 펌웨어 업데이트(USB 대용량 저장 장치) 등 모든 기능을 단일 USB 2.0 OTG 인터페이스에 의존한다.[3]

ANTSDR E310은 여기에 더해 기가비트 이더넷 포트, 부팅 및 저장 공간으로 활용 가능한 마이크로 SD 카드 슬롯, 그리고 통합된 USB-JTAG 디버거 등 훨씬 풍부한 인터페이스를 제공한다.[7] 이러한 추가 하드웨어는 포팅 과정에서 가장 복잡한 수정을 요구하는 부분이다. 리눅스 커널에서 새로운 드라이버(예: 이더넷을 위한 macb, SD 카드를 위한 sdhci)를 활성화해야 하며, 디바이스 트리에서 해당 하드웨어의 물리적 연결 정보(레지스터 주소, 인터럽트 등)를 정확히 기술해야 한다. 특히 이더넷의 경우, 외부 PHY 칩과의 연결 정보까지 포함되어야 한다. 또한, U-Boot 부트로더에는 SD 카드나 네트워크 TFTP 서버로부터 커널 이미지를 로드할 수 있는 새로운 부팅 명령어가 추가되어야 한다.[10]

이러한 하드웨어 차이점과 그에 따른 포팅 요구사항을 요약하면 다음 표와 같다. 이 표는 전체 포팅 과정의 로드맵 역할을 하며, 5장에서 기술될 모든 소프트웨어 수정 작업의 기술적 근거를 제공한다.

표 **2.1: ADALM-PLUTO** 대 **ANTSDR E310** 하드웨어 사양 비교

기능	ADALM-PLUTO	ANTSDR E310	포팅 시 고려사항
SoC	Xilinx Zynq Z-7010	Xilinx Zynq Z-7020	Vivado 프로젝트 타겟 디바이스 변경, 듀얼 코어 활용

ARM 코어	Single-core Cortex-A9 @ 667MHz	Dual-core Cortex-A9 @ 766MHz	커널 SMP(Symmetric Multi-Processing) 설정 확인
FPGA 로직 셀	28k	85k	더 복잡한 FPGA 로직 구현 가능
DSP 슬라이스	80	220	고성능 신호 처리 가속기 구현 용이
DDR3L RAM	512MB	1GB	U-Boot 메모리 맵 및 커널 디바이스 트리 수정
QSPI 플래시	32MB	32MB	변경 없음
RF 트랜시버	AD9363	AD9363/AD9361	드라이버 설정으로 2T2R 모드 활성화
RF 채널	1T1R	2T2R (MIMO)	디바이스 트리 속성 추가, U-Boot 환경 변수 설정
연결성	USB 2.0 OTG	USB 2.0 OTG, 기가비트 이더넷, SD 카드	커널 드라이버 활성화, 디바이스 트리 노드 추가
부팅 미디어	QSPI 플래시, USB (DFU)	QSPI 플래시, SD 카드, USB (DFU)	U-Boot에 SD 카드 부팅 로직 추가
클럭 소스	±25ppm TCXO	0.5ppm TCXO	디바이스 트리 클럭 소스 정보 확인

ANTSDR E310이 Pluto 펌웨어 및 Openwifi와 같은 오픈소스 프로젝트를 지원한다고 명시적으로 홍보하고 [7], 제조사인 MicroPhase가 Pluto 펌웨어 개조를 위한 전용 GitHub 리포지토리(antsdr-fw-patch)를 제공한다는 점은 [10] 중요한 사실을 시사한다. 이는 ANTSDR E310의 Pluto 호환성이 우연한 결과가 아니라, 성숙하고 방대한 PlutoSDR의 소프트웨어 생태계(MATLAB, GNU Radio, libiio 등)를 활용하기 위한 의도적인 설계 결정임을 보여준다.[1] 따라서 이 포팅

작업은 미지의 하드웨어를 리버스 엔지니어링하는 탐험이 아니라, 제조사가 제시한 검증된 경로를 따라가며 그 기술적 배경을 이해하는 체계적인 과정이다.

3.0 PlutoSDR 펌웨어 스택의 구조

ANTSDR E310으로의 포팅 작업을 수행하기 전에, 원본이 되는 ADALM-PLUTO 펌웨어의 구조와 빌드 시스템을 이해하는 것이 필수적이다. PlutoSDR 펌웨어는 임베디드 리눅스 시스템의 표준적인 구성 요소들을 포함하며, 고도로 통합된 빌드 환경을 통해 관리된다.

3.1 Zynq 부팅 시퀀스

Xilinx Zynq SoC의 부팅 과정은 여러 단계로 이루어진다.

1. **BootROM:** 전원이 인가되면, Zynq 칩 내부에 고정된 BootROM 코드가 가장 먼저 실행된다. 이 코드는 부팅 모드 핀 설정을 읽어 QSPI 플래시, SD 카드, JTAG 등 지정된 부팅 장치에서 1단계 부트로더(FSBL)를 찾아서 로드한다.[4]
2. **FSBL (First-Stage Bootloader):** FSBL은 Vivado/Vitis SDK에서 생성되며, 하드웨어 플랫폼에 특화된 초기화 코드를 담고 있다. 주된 역할은 Zynq의 처리 시스템(PS) 부분을 초기화하는 것인데, 특히 DDR 메모리 컨트롤러를 설정하여 외부 RAM을 사용할 수 있도록 만드는 것이 핵심이다. 초기화가 완료되면 FSBL은 2단계 부트로더인 U-Boot를 DDR 메모리에 로드하고 실행 제어권을 넘긴다.[5]
3. **U-Boot (Universal Boot Loader):** U-Boot는 더 범용적이고 강력한 기능을 갖춘 부트로더다. USB 컨트롤러, 이더넷 등 추가적인 하드웨어를 초기화하고, 부팅 스크립트에 따라 리눅스 커널 이미지(zImage), 디바이스 트리 블롭(.dtb), 그리고 루트 파일 시스템 이미지를 메모리에 로드한다. PlutoSDR의 경우, U-Boot는 USB를 통해 PC에 대용량 저장 장치로 인식되게 하여 펌웨어 업데이트를 용이하게 하는 기능도 수행한다. 모든 준비가 끝나면 U-Boot는 커널의 시작 주소로 점프하여 운영체제를 부팅시킨다.[4]

3.2 plutosdr-fw 리포지토리 구조

Analog Devices의 공식 펌웨어 소스 코드는 analogdevicesinc/plutosdr-fw GitHub 리포지토리에서 관리된다.[17] 이 리포지토리는 단일 코드를 담고 있는 것이 아니라, 펌웨어를 구성하는 여러 핵심 컴포넌트들의 소스 코드 리포지토리를 Git 서브모듈(submodule)로

포함하는 메타-리포지토리 형태를 띤다.

- u-boot-xlnx: Xilinx용으로 포팅된 U-Boot 소스 코드
- linux: ADI에서 유지보수하는 리눅스 커널 소스 코드
- hdl: Zynq PL 부분을 위한 하드웨어 기술 언어(HDL) 소스 코드 및 Vivado 프로젝트
- buildroot: 루트 파일 시스템을 생성하기 위한 Buildroot 소스 코드

리포지토리의 최상위 디렉토리에 위치한 Makefile은 이 모든 컴포넌트의 빌드 과정을 총괄하는 역할을 한다. make 명령을 실행하면, 이 Makefile은 각 서브모듈 디렉토리로 이동하여 컴포넌트를 순서대로 컴파일하고, 최종적으로 빌드된 결과물들을 조합하여 pluto.frm(USB 저장 장치용)과 pluto.dfu(DFU 모드용)라는 최종 펌웨어 파일을 생성한다.[17]

3.3 Buildroot와 루트 파일 시스템

PlutoSDR의 임베디드 리눅스 시스템은 경량화를 위해 Buildroot 프레임워크를 사용하여 구축된다.[4] Buildroot는 임베디드 리눅스 시스템을 처음부터 생성하기 위한 강력하고 유연한 도구다. 사용자가 선택한 설정에 따라 크로스 컴파일 툴체인, C 라이브러리, 커널, 부트로더 및 필요한 모든 사용자 공간 애플리케이션을 자동으로 다운로드하고 컴파일하여 최종 루트 파일 시스템 이미지를 생성한다.

PlutoSDR의 경우, Buildroot는 다음과 같은 핵심 요소들을 포함하는 루트 파일 시스템을 만든다.

- **BusyBox:** ls, cat, ifconfig 등 표준 리눅스 유틸리티들을 하나의 실행 파일로 통합한 경량화된 셸 환경을 제공한다.
- **C 라이브러리:** 임베디드 시스템에 적합한 uClibc 또는 musl과 같은 경량 C 라이브러리를 사용한다.
- **IIO 데몬 (iiod):** PlutoSDR의 핵심 기능인 Industrial I/O(IIO) 프레임워크를 네트워크를 통해 원격으로 제어할 수 있도록 해주는 서버 애플리케이션이다.
- **기타 유틸리티:** SSH 서버(Dropbear), 네트워크 설정 도구 등 임베디드 환경에 필요한 최소한의 프로그램들을 포함한다.

이렇게 생성된 파일 시스템은 rootfs.cpio.gz 형태의 압축 아카이브로 만들어지며, 부팅 시 리눅스 커널에 의해 메모리에 로드되어 루트(/) 디렉토리로 마운트된다.[17]

4.0 크로스-개발 환경 구축

ANTSDR E310용 펌웨어를 성공적으로 빌드하기 위해서는 특정 버전의 도구와 라이브러리로

구성된 정교한 크로스-개발 환경을 호스트 PC에 구축해야 한다. 이 과정에서의 작은 실수나 버전 불일치는 빌드 실패로 이어질 수 있으므로 정확한 절차를 따르는 것이 매우 중요하다.

4.1 호스트 시스템 사전 요구사항

펌웨어 빌드 시스템은 Debian 또는 Ubuntu 기반의 리눅스 배포판에서 가장 원활하게 동작한다. 빌드를 시작하기 전에, 다음 패키지들이 호스트 시스템에 설치되어 있어야 한다. 이들은 각각 소스 코드 컴파일, 펌웨어 이미지 생성, 디바이스 트리 컴파일 등 빌드 과정의 다양한 단계에서 필수적으로 사용된다.[17]

Bash

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

4.2 Xilinx 툴체인 설치

Zynq SoC의 FPGA 로직(PL)과 1단계 부트로더(FSBL)를 생성하기 위해서는 AMD-Xilinx의 Vivado Design Suite가 필요하다. 포팅 작업의 기반이 되는 antsdr-fw-patch 리포지토리의 최신 버전은 Vivado 2023.2 버전을 기준으로 작성되었다.[10] 따라서 AMD-Xilinx 공식 웹사이트에서 Vivado ML Edition 2023.2 버전을 다운로드하여 설치해야 한다. Vivado 설치 과정에서 Vitis Unified Software Platform도 함께 설치되며, 이는 FSBL 생성과 임베디드 소프트웨어 개발에 사용된다.

4.3 ARM 크로스-컴파일러

과거 PlutoSDR 빌드 시스템은 Vivado에 포함된 ARM 크로스-컴파일러를 사용했으나, Buildroot와의 호환성 문제로 인해 현재는 외부의 독립적인 툴체인을 사용하도록 변경되었다.[10]

antsdr-fw-patch v0.39 릴리즈는 Linaro에서 배포하는 gcc-linaro-7.3.1-2018.05-i686_arm-linux-gnueabihf 툴체인을 요구한다.

이 툴체인을 다운로드하여 적절한 위치(예: /opt/linaro)에 압축을 해제한 후, 셸 환경 변수를 설정하여 빌드 시스템이 툴체인을 찾을 수 있도록 해야 한다. .bashrc 파일에 다음 라인들을 추가하는 것이 일반적이다.

Bash

```
# Linaro ARM Toolchain
export CROSS_COMPILE=arm-linux-gnueabihf-
export PATH=$PATH:/opt/linaro/gcc-linaro-7.3.1-2018.05-i686_arm-linux-gnueabihf/bin

# Xilinx Vivado/Vitis
export VIVADO_SETTINGS=/opt/Xilinx/Vivado/2023.2/settings64.sh
source $VIVADO_SETTINGS
```

환경 변수 설정 후에는 터미널을 재시작하거나 source ~/.bashrc 명령을 실행하여 변경 사항을 적용해야 한다.

4.4 소스 코드 확보

모든 개발 도구가 준비되었다면, 포팅에 필요한 소스 코드를 GitHub에서 복제(clone)해야 한다. antsdr-fw-patch 리포지토리는 내부에 plutosdr-fw를 서브모듈로 포함하고, plutosdr-fw는 다시 linux, u-boot-xlnx 등의 서브모듈을 포함하는 다중 계층 구조를 가지고 있다. 따라서 모든 서브모듈을 한 번에 올바르게 가져오기 위해 --recursive 옵션을 사용하는 것이 매우 중요하다.

다음 명령어를 사용하여 특정 버전(v0.39)의 소스 코드를 재귀적으로 복제한다.[10]

Bash

```
git clone -b v0.39 --recursive https://github.com/MicroPhase/antsdr-fw-patch.git
```

이 명령이 성공적으로 완료되면, antsdr-fw-patch 디렉토리 내부에 모든 관련 소스 코드가

준비되어 포팅 및 빌드 작업을 시작할 수 있는 상태가 된다.

5.0 핵심 포팅 작업: 소스 코드 및 로직 수정

개발 환경이 구축되면, ANTSDR E310의 하드웨어 특성에 맞게 펌웨어의 각 구성 요소를 수정하는 핵심 포팅 작업에 들어간다. 이 과정은 antsdr-fw-patch 리포지토리에 포함된 patch.sh 스크립트를 통해 자동화될 수 있지만, 본 보고서에서는 스크립트가 수행하는 각 수정 작업의 기술적 내용과 그 의미를 심층적으로 분석한다.

5.1 FPGA 하드웨어 정의(HDL) 수정

FPGA 하드웨어 정의는 Zynq SoC의 프로그래머블 로직(PL)과 처리 시스템(PS) 간의 인터페이스를 결정하는 청사진이다.

1. **SoC** 타겟 변경: 가장 먼저 Vivado 프로젝트의 타겟 디바이스를 ADALM-PLUTO의 Z-7010에서 ANTSDR E310의 Z-7020으로 변경해야 한다. 이는 더 많은 로직 셀과 DSP 자원을 활용할 수 있게 해준다.
2. 블록 디자인 수정: hdl 소스 코드에 적용되는 패치는 Vivado의 블록 디자인을 생성하는 TCL 스크립트를 수정한다. 이 수정의 핵심은 ANTSDR E310에 추가된 주변장치들을 Zynq PS에 연결하는 것이다. 구체적으로, Zynq PS의 GEM1(기가비트 이더넷 MAC)과 SDHCI(SD 카드 컨트롤러)를 활성화하고, 이들의 신호를 외부 핀으로 연결(EMIO를 통해)하도록 블록 디자인을 변경한다.
3. 하드웨어 정의 파일 생성: 수정된 블록 디자인으로부터 새로운 하드웨어 정의 파일(.xsa)을 생성(export)한다. 이 파일에는 PS 설정, 클럭 정보, 주변장치 연결 정보 등 시스템의 모든 하드웨어 구성이 담겨 있다. Vitis와 PetaLinux 같은 후속 툴들은 이 .xsa 파일을 입력으로 받아 FSBL과 기본 디바이스 트리를 자동으로 생성한다.[20]

5.2 U-Boot 부트로더 개조

U-Boot는 시스템 부팅의 두 번째 단계를 책임지며, 하드웨어 초기화와 커널 로딩을 담당한다.

1. **DDR** 메모리 설정: U-Boot의 초기 단계(SPL)는 Vivado에서 생성된 ps7_init.c/h 파일을 사용하여 DDR 컨트롤러를 초기화한다. ANTSDR E310의 1GB RAM을 올바르게 인식하고 사용하기 위해, 새로운 .xsa 파일로부터 생성된 ps7_init 관련 파일들로 교체하는 패치가

적용된다. 이는 부트로더가 전체 메모리 공간에 접근할 수 있도록 하는 필수적인 과정이다.[22]

2. 새로운 부팅 명령어 추가**:** ANTSDR E310은 SD 카드와 네트워크 부팅을 지원한다. 이를 위해 u-boot-xlnx 소스 코드에 mmcinfo, load mmc (SD 카드용) 및 dhcp, tftpboot (네트워크용)와 같은 새로운 부팅 명령어를 지원하는 코드가 패치를 통해 추가된다. 또한, 부팅 시 실행될 기본 부팅 스크립트(bootcmd) 환경 변수도 SD 카드나 이더넷을 우선적으로 확인하도록 수정된다.[10]
3. **2T2R** 모드 지원 환경 변수**:** U-Boot는 부팅 시 특정 환경 변수(예: mode=2r2t, compatible=ad9361)를 읽어 리눅스 커널에 전달하는 기능을 수행한다. 패치를 통해 이러한 변수들을 처리하고, 커널 부팅 시 디바이스 트리나 커맨드 라인을 통해 AD936x 드라이버에 전달하는 로직이 추가된다. 이는 사용자가 부팅 시점에 SDR의 동작 모드를 유연하게 설정할 수 있게 해준다.[10]

5.3 리눅스 커널 및 디바이스 트리 맞춤화

리눅스 커널과 디바이스 트리는 운영체제가 하드웨어를 인식하고 제어하는 데 가장 중요한 역할을 한다.

1. 커널 설정**(defconfig)** 변경**:** 리눅스 커널 소스에 적용되는 패치는 Zynq용 기본 커널 설정 파일(defconfig)을 수정한다. ANTSDR E310에 추가된 기가비트 이더넷과 SD 카드 컨트롤러를 지원하기 위해, Cadence MACB 이더넷 컨트롤러 드라이버(CONFIG_MACB)와 Arasan SDHCI 컨트롤러 드라이버(CONFIG_MMC_SDHCI_ARASAN)를 활성화(=y 또는 =m)하는 내용이 추가된다.[24]
2. 디바이스 트리 소스**(.dts)** 수정**:** 디바이스 트리는 리눅스 커널에 하드웨어 구성을 알려주는 데이터 구조다. 포팅 과정에서 가장 핵심적인 수정이 이루어지는 부분이다. zynq-ant-sdr.dts라는 새로운 디바이스 트리 파일이 생성되며, 여기에는 다음과 같은 ANTSDR E310 고유의 하드웨어 정보가 기술된다.
    - 메모리 노드**:** memory 노드의 reg 속성을 <0x0 0x40000000>으로 수정하여 전체 1GB RAM 크기를 커널에 알린다.
    - 이더넷 노드**:** &gem1 노드의 status를 "okay"로 설정하여 이더넷 MAC을 활성화한다. 또한, 이 MAC에 연결된 외부 이더넷 PHY 칩의 주소와 통신 방식(phy-mode = "rgmii-id";)을 명시한다.[26]
    - **SD** 카드 노드**:** sdhci 컨트롤러에 대한 노드를 추가하여, 레지스터 주소, 인터럽트 번호, 그리고 SD 카드 슬롯의 특성(예: non-removable)을 기술한다.
    - **AD936x** 트랜시버 노드**:** SPI 버스에 연결된 AD9363 트랜시버 노드를 수정하는 것이 핵심이다. 2T2R MIMO 기능을 활성화하기 위해 adi,phy-2rx-2tx-mode-enable; 속성을 추가한다. 이 속성이 존재하면, ADI의 AD9361/3 드라이버는 초기화 과정에서 칩을 2T2R 모드로 설정한다. 이는 ANTSDR E310의 가장 큰 하드웨어적 장점을 소프트웨어적으로 구현하는 결정적인 단계다.

이러한 포팅 방식은 "포크(fork)에 패치(patch)를 적용하는" 유지보수 모델을 따른다. 즉, ANTSDR 펌웨어는 ADI의 공식 plutosdr-fw 리포지토리를 직접 수정하는 대신, 특정 버전의 공식 펌웨어 위에 antsdr-fw-patch 리포지토리의 패치들을 덧씌우는 방식으로 관리된다.[10] 이 방식은 초기 포팅을 빠르고 체계적으로 수행할 수 있다는 장점이 있다. 하지만 단점도 명확하다. ADI에서 보안 업데이트나 새로운 기능이 포함된

plutosdr-fw의 새 버전을 출시하더라도, ANTSDR 사용자는 MicroPhase가 antsdr-fw-patch를 새 버전에 맞게 업데이트해 줄 때까지 기다려야 한다. 이는 상위 프로젝트의 최신 변경 사항을 즉시 반영하기 어렵게 만들며, 장기적인 유지보수 관점에서 제3자(MicroPhase)에 대한 의존성을 발생시키는 잠재적 위험 요소로 작용할 수 있다.

6.0 펌웨어 컴파일, 배포 및 시스템 검증

모든 소스 코드 수정 사항이 준비되면, 전체 펌웨어 이미지를 빌드하고 이를 ANTSDR E310에 배포하여 정상적으로 동작하는지 검증하는 단계로 넘어간다. 이 과정은 일련의 명확한 명령어로 수행되며, 각 단계의 결과물을 확인하여 문제를 조기에 발견하는 것이 중요하다.

6.1 빌드 실행

수정된 소스 트리로부터 최종 펌웨어를 생성하는 과정은 antsdr-fw-patch 디렉토리의 최상위에서 시작된다. 다음 명령어 시퀀스는 ANTSDR E310(ant)을 타겟으로 설정하고, 필요한 패치를 적용한 후, 전체 빌드 프로세스를 실행한다.

1. 타겟 설정: 빌드 시스템에 ANTSDR E310용으로 빌드할 것을 알린다.
   Bash
   ```
   export TARGET=ant
   ```
   10
2. 패치 적용: patch.sh 스크립트를 실행하여 U-Boot, 리눅스 커널 등 각 컴포넌트의 소스 코드에 ANTSDR E310용 수정 사항을 적용한다.
   Bash
   ```
   sh patch.sh ant
   ```
   10
3. 빌드 시작: plutosdr-fw 디렉토리로 이동하여 make 명령을 실행한다. 이 명령은 최상위 Makefile을 호출하여 FSBL, U-Boot, 리눅스 커널, 디바이스 트리, 루트 파일 시스템을

순차적으로 컴파일하고, 최종 펌웨어 이미지(ant.dfu, ant.frm)를 생성한다.

```
Bash
cd plutosdr-fw
make
```

10

6.2 SD 카드 부팅을 위한 아티팩트 패키징

ANTSDR E310의 주요 부팅 방식 중 하나인 SD 카드 부팅을 위해서는 빌드된 결과물들을 특정 구조에 맞게 SD 카드에 복사해야 한다. antsdr-fw-patch 빌드 시스템은 이 과정을 단순화하기 위해 make sdimg라는 추가 목표(target)를 제공한다.[10] 이 명령을 실행하면 SD 카드의 부트 파티션에 복사해야 할 파일들이

build_sdimg 디렉토리에 생성된다.

SD 카드 부트 파티션의 주요 구성 요소는 다음과 같다.

- BOOT.BIN: Zynq의 BootROM이 가장 먼저 로드하는 복합 부트 이미지다. Xilinx의 bootgen 유틸리티를 통해 생성되며, 내부에는 FSBL(fsbl.elf), FPGA 비트스트림(system_top.bit), 그리고 U-Boot 부트로더(u-boot.elf)가 순서대로 포함되어 있다.[16]
- uImage: 리눅스 커널 실행 이미지.
- zynq-ant-sdr.dtb: 컴파일된 디바이스 트리 블롭.
- uramdisk.image.gz: 압축된 루트 파일 시스템 이미지.

6.3 배포

1. **SD** 카드 준비**:** 4GB 이상의 마이크로 SD 카드를 FAT32 파일 시스템으로 포맷한다.[29]
2. 파일 복사**:** 위에서 생성된 BOOT.BIN, uImage, zynq-ant-sdr.dtb, uramdisk.image.gz 파일들을 포맷된 SD 카드의 루트 디렉토리로 복사한다.
3. 부트 모드 설정**:** ANTSDR E310 보드의 부트 모드 점퍼를 SD 카드 부팅 모드로 설정한다. 정확한 점퍼 설정은 장치 매뉴얼을 참조해야 한다.[19]
4. 부팅**:** 파일이 복사된 SD 카드를 ANTSDR E310에 삽입하고 전원을 인가한다.

6.4 시스템 검증 프로토콜

펌웨어가 성공적으로 배포되었는지 확인하고 시스템이 의도대로 동작하는지 검증하기 위해 다음 절차를 수행한다.

1. 시리얼 콘솔 모니터링**:** ANTSDR E310의 UART 핀에 USB-to-Serial 어댑터를 연결하고, PC에서 PuTTY나 minicom과 같은 터미널 에뮬레이터를 사용하여 115200 baud rate로 시리얼 포트에 연결한다.[30] 전원을 켜면 FSBL, U-Boot, 리눅스 커널의 부팅 메시지가 순서대로 출력되는 것을 확인할 수 있다. 이 과정에서 에러 메시지가 출력되지 않는지 면밀히 관찰하는 것은 디버깅에 매우 중요하다.
2. 네트워크 연결 검증**:** 리눅스 부팅이 완료된 후, ANTSDR E310을 이더넷 케이블로 로컬 네트워크에 연결한다. 시리얼 콘솔이나 SSH를 통해 장치에 로그인한 후, ifconfig 또는 ip addr 명령으로 eth0 인터페이스가 IP 주소를 할당받았는지 확인한다. 동일 네트워크 상의 다른 PC에서 해당 IP 주소로 ping을 보내 응답이 오는지 확인하여 네트워크 스택이 정상적으로 동작하는지 검증한다.[14]
3. **SDR** 서브시스템 검증**:** 포팅의 가장 중요한 목표인 SDR 기능의 정상 동작을 확인한다. 먼저, SSH를 통해 장치에 접속한다 (기본 사용자: root, 비밀번호: analog).[4] 그 다음, 호스트 PC에
   libiio 라이브러리를 설치하고, 다음 명령어를 실행하여 ANTSDR E310의 IIO 장치 정보를 확인한다.
   Bash
   `iio_info -n <ANTSDR_E310의_IP주소>`

   명령 실행 결과, iio:device 목록에 ad9361-phy가 표시되어야 한다. 더 중요한 것은, 이 장치 아래에 4개의 IIO 채널이 나타나는지 확인하는 것이다.
   - voltage0 (input): RX1
   - voltage1 (input): RX2
   - voltage0 (output): TX1
   - voltage1 (output): TX2
     이 4개의 채널이 모두 정상적으로 인식된다면, 2T2R MIMO 모드가 성공적으로 활성화되었음을 의미하며, 펌웨어 포팅이 성공적으로 완료되었음을 최종적으로 확인할 수 있다.

7.0 결론 및 고급 주제

본 보고서는 ADALM-PLUTO의 오픈소스 리눅스 펌웨어를 하드웨어적으로 강화된 ANTSDR E310 플랫폼으로 이식하는 전 과정을 체계적으로 기술했다. 이 포팅 작업의 핵심은 두 플랫폼 간의 명확한 하드웨어 차이점—SoC 업그레이드, 메모리 확장, 2T2R MIMO 지원, 이더넷 및 SD 카드 인터페이스 추가—을 분석하고, 이를 FPGA 하드웨어 정의, U-Boot 부트로더, 리눅스 커널 및 디바이스 트리에 정확하게 반영하는 것이었다. "포크에 패치를 적용하는" 방식의 포팅

모델은 초기 개발을 용이하게 하지만, 상위 프로젝트의 업데이트에 대한 의존성을 발생시켜 장기적인 유지보수 측면에서 고려가 필요하다는 점도 확인했다.

성공적으로 포팅된 ANTSDR E310은 단순한 PlutoSDR의 복제품을 넘어, 훨씬 더 넓은 가능성을 지닌 강력한 SDR 플랫폼으로 거듭난다. 사용자는 이제 다음과 같은 고급 주제들을 탐구할 수 있다.

7.1 ANTSDR의 고급 기능 활용

- **MIMO** 애플리케이션**:** 활성화된 2T2R 기능을 통해 빔포밍(beamforming), 공간 다이버시티(spatial diversity), 또는 완전한 전이중 통신 시스템과 같은 고급 MIMO 기술을 GNU Radio와 같은 프레임워크 내에서 실험하고 구현할 수 있다. 이는 통신 시스템의 신뢰성과 전송률을 획기적으로 향상시킬 수 있는 잠재력을 제공한다.
- 온보드 프로세싱**:** 듀얼 코어 ARM Cortex-A9 프로세서와 1GB의 넉넉한 RAM은 호스트 PC의 부담을 줄이고 장치 자체에서 상당한 수준의 신호 처리를 수행할 수 있게 한다. 예를 들어, 독립형 스펙트럼 모니터링 장치를 구축하거나, srsRAN과 같은 소프트웨어 기반 무선 통신 스택 전체를 ANTSDR E310 내부에서 직접 구동하는 것도 가능하다.[8]
- **FPGA** 가속**:** Z-7020 SoC의 확장된 프로그래머블 로직은 맞춤형 DSP 가속기를 구현할 수 있는 충분한 공간을 제공한다. 복잡한 FIR 필터, 고속 푸리에 변환(FFT) 엔진, 또는 특정 통신 표준을 위한 변복조기 등을 FPGA에 하드웨어로 구현하여 ARM 코어의 부하를 줄이고, 실시간으로 더 높은 데이터 처리량을 달성할 수 있다.

7.2 향후 연구 및 개발 방향

이 포팅 작업은 끝이 아니라 새로운 시작이다. 안정적인 임베디드 리눅스 환경이 구축된 ANTSDR E310을 기반으로, 사용자는 표준 Zynq 개발 워크플로우를 따라 시스템을 더욱 깊이 있게 맞춤화할 수 있다. Buildroot를 사용하여 필요한 라이브러리나 애플리케이션을 포함하는 자신만의 루트 파일 시스템을 구성하거나 [31], 특정 하드웨어를 제어하기 위한 맞춤형 리눅스 커널 모듈을 개발하는 등의 고급 개발이 가능하다. 궁극적으로, ANTSDR E310은 PlutoSDR의 방대한 소프트웨어 생태계와 고성능 하드웨어가 결합된 이상적인 플랫폼으로서, 차세대 무선 통신 기술의 연구와 프로토타이핑을 위한 무한한 가능성을 제공한다.

참고 자료

1. 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

2. Analog Devices ADALM-PLUTO - Reference - SuperPacket, 9월 22, 2025에 액세스, https://www.superpacket.org/analog-devices-adalm-pluto-reference.html
3. Analog Devices Inc. ADALM-PLUTO Active Learning Module - Mouser Electronics, 9월 22, 2025에 액세스, https://www.mouser.com/new/analog-devices/adi-adalm-pluto/
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. Analog Devices ADALM-PLUTO - DigiKey, 9월 22, 2025에 액세스, https://www.digikey.com/en/blog/analog-devices-adalm-pluto
6. ADALM-PLUTO SDR: Unboxing and Initial Testing, 9월 22, 2025에 액세스, https://www.rtl-sdr.com/adalm-pluto-sdr-unboxing-and-initial-testing/
7. MicroPhase ANTSDR E310 AD9363 SDR Development Board for ADI Pluto Communication- | eBay, 9월 22, 2025에 액세스, https://www.ebay.com/itm/126254547389
8. MicroPhase ANTSDR E310 Software Defined Radio Demo Board transceiver ZYNQ 7000 SoC ADI AD9361 SDR transmitter and receiver - AliExpress, 9월 22, 2025에 액세스, https://www.aliexpress.com/item/1005003181244737.html
9. ANTSDR: Low-cost SDR Alternatives for Ettus Research and Lime Microsystems | Hacker News, 9월 22, 2025에 액세스, https://news.ycombinator.com/item?id=41213514
10. MicroPhase/antsdr-fw-patch: Repository of antsdr firmware make - GitHub, 9월 22, 2025에 액세스, https://github.com/MicroPhase/antsdr-fw-patch
11. Is there a noticeable difference between Xilinx ZYNQ7010 and ZYNQ7020? : r/FPGA - Reddit, 9월 22, 2025에 액세스, https://www.reddit.com/r/FPGA/comments/157xr7h/is_there_a_noticeable_difference_between_xilinx/
12. Software radio E310-9363 ZYNQ7020 ADI Pluto Experimental Platform Antsdr, 9월 22, 2025에 액세스, https://www.sdrstore.eu/software-defined-radio/instruments/plutosdr/software-radio-e310-9363-zynq7020-adi-pluto-experimental-platform-antsdr-en/
13. Beginner's guide to SDR (using ADALM-Pluto) - element14 Community, 9월 22, 2025에 액세스, https://community.element14.com/products/roadtest/rv/roadtest_reviews/1698/beginners_guide_to_adalm-pluto
14. ANTSDR Unpacking and Test Rev. 1.1, 9월 22, 2025에 액세스, https://down.hgeek.com/download/93159/93159-E310-AD9361-Rev-1-1-English-Manual.pdf
15. MicroPhase - GitHub, 9월 22, 2025에 액세스, https://github.com/microphase
16. Prepare Boot Image - Xilinx Wiki - Confluence, 9월 22, 2025에 액세스, https://xilinx-wiki.atlassian.net/wiki/display/A/prepare+boot+image
17. analogdevicesinc/plutosdr-fw - GitHub, 9월 22, 2025에 액세스, https://github.com/analogdevicesinc/plutosdr-fw

18. BR2_EXTERNAL framework for Analog Device's PlutoSDR Zynq - GitHub, 9월 22, 2025에 액세스, https://github.com/oscimp/PlutoSDR
19. MicroPhase/antsdr-fw: ANTSDR Firmware - GitHub, 9월 22, 2025에 액세스, https://github.com/MicroPhase/antsdr-fw
20. ZynqLinux - HERO Documentation, 9월 22, 2025에 액세스, https://pulp-platform.org/hero/doc/software/host/zynqlinux/
21. BondMachineHQ/bondmachine_zedboard_buildroot_example: Buildroot on Zedboard. How to create from scratch a complete BondMachine accelerated buildroot image for the Zedboard - GitHub, 9월 22, 2025에 액세스, https://github.com/BondMachineHQ/bondmachine_zedboard_buildroot_example
22. [Buildroot] [PATCH v2 1/1] boot/uboot: add support for custom zynq ps7_init_gpl.c, 9월 22, 2025에 액세스, https://lists.buildroot.org/pipermail/buildroot/2025-June/781427.html
23. U-Boot Commands - Xilinx Wiki - Confluence, 9월 22, 2025에 액세스, https://xilinx-wiki.atlassian.net/wiki/display/A/U-boot
24. Mainline Linux on Zynq - Xilinx Wiki - Confluence, 9월 22, 2025에 액세스, https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842076/Mainline+Linux+on+Zynq
25. Linux Kernel defconfig for ADRV9364-Z7020 + ADVR1CC-BoB - Q&A - EngineerZone, 9월 22, 2025에 액세스, https://ez.analog.com/linux-software-drivers/f/q-a/556535/linux-kernel-defconfig-for-adrv9364-z7020-advr1cc-bob
26. TE0720 custom carrier / 2nd Ethernet (TE0706 based) - Trenz Electronic GmbH Support Forum, 9월 22, 2025에 액세스, https://forum.trenz-electronic.de/index.php?topic=1720.0
27. Example design for Zynq-7000 with Ethernet routed through the MIO (ETH0) including device tree - Adaptive Support, 9월 22, 2025에 액세스, https://adaptivesupport.amd.com/s/question/0D52E00006hpWFwSAM/example-design-for-zynq7000-with-ethernet-routed-through-the-mio-eth0-including-device-tree?language=en_US
28. Creating a Baremetal Boot Image for Zynq-7000 Devices - Digilent Reference, 9월 22, 2025에 액세스, https://digilent.com/reference/programmable-logic/guides/zynq-baremetal-boot
29. Prepare Boot Medium - Xilinx Wiki - Confluence, 9월 22, 2025에 액세스, https://xilinx-wiki.atlassian.net/wiki/display/A/prepare+boot+medium
30. ADALM-PLUTO Revision D | Radio Club PZK LAB-EL HF5L, 9월 22, 2025에 액세스, https://hf5l.pl/en/adalm-pluto-revision-d/
31. How to add another target to make for buildroot (i.e. custom FSBL, or bistream from Vivado project)? - Stack Overflow, 9월 22, 2025에 액세스, https://stackoverflow.com/questions/46682305/how-to-add-another-target-to-make-for-buildroot-i-e-custom-fsbl-or-bistream-f
32. Buildroot & Yocto: Which Platform to Choose for BSP Development - Promwad, 9월 22, 2025에 액세스, https://promwad.com/news/bsp-linux-buildroot-comparison-peta-linux