



**THE IMAGINATION UNIVERSITY PROGRAMME**

# **RVfpga-SoC Lab 5**

**SweRVolf 에서  
Tensorflow Lite 실행**

표 1. RVfpga 용어

Name	Description
<b>Courses</b>	
<b>RVfpga</b>	RVfpgaNexys 및 RVfpgaSim, RISC-V SoC(System-on-Chip)를 사용하여 프로그램을 실행하고 주변 장치를 추가하여 시스템을 확장하고(RVfpga Labs 1-10), 시뮬레이션, 성능 측정, 명령 추가 및 메모리 시스템 수정(RVfpga Labs 11-20)을 실행하여 코어 및 메모리 시스템을 탐색하는 방법을 보여주는 과정입니다. 과정 전반에 걸쳐 사용자는 RISC-V 툴체인(컴파일러 및 디버거) 및 시뮬레이터, Verilator HDL 시뮬레이터 및 Western Digital 의 Whisper 명령어 세트 시뮬레이터(ISS)를 사용하는 방법도 보여줍니다.
<b>RVfpga-SoC</b>	SweRV 코어, 메모리 및 주변 장치와 같은 빌딩 블록을 사용하여 처음부터 SweRVolfX SoC 의 하위 집합을 구축하는 방법을 보여주는 과정입니다. 이 과정은 또한 Zephyr 실시간 운영 체제(RTOS)를 SweRVolf 에 로드하고 운영 체제 상단에서 Tensorflow Lite 의 hello world 예제를 포함한 프로그램을 실행하는 방법을 보여줍니다.
<b>Cores and SoCs</b>	
<b>SweRV EH1 Core</b>	Western Digital 에서 개발한 오픈 소스 상용 RISC-V 코어 ( <a href="https://github.com/chipsalliance/Cores-SweRV">https://github.com/chipsalliance/Cores-SweRV</a> ).
<b>SweRV EH1 Core Complex</b>	추가된 메모리(ICCM, DCCM 및 명령 캐시), 프로그래밍 가능한 인터럽트 컨트롤러(PIC), 버스 인터페이스 및 디버그 장치가 있는 SweRV EH1 코어 ( <a href="https://github.com/chipsalliance/Cores-SweRV">https://github.com/chipsalliance/Cores-SweRV</a> ).
<b>SweRVolfX</b>	RVfpga 과정에서 사용하는 System on Chip 으로 SweRVolf 의 확장입니다.  <b>SweRVolf</b> ( <a href="https://github.com/chipsalliance/Cores-SweRVolf">https://github.com/chipsalliance/Cores-SweRVolf</a> ): SweRV EH1 Core Complex 를 기반으로 구축된 오픈 소스 SoC 입니다. boot ROM, UART 인터페이스, 시스템 컨트롤러, 상호 연결(AXI Interconnect, Wishbone Interconnect 및 AXI-to-Wishbone 브리지) 및 SPI 컨트롤러를 추가합니다.  <b>SweRVolfX</b> : SweRVolf 에 GPIO, PTC, 추가 SPI 및 8 자리 7-세그먼트 디스플레이 컨트롤러, 4 가지 새로운 주변 장치를 추가합니다.
<b>RVfpgaNexys</b>	Nexys A7 보드 및 주변 장치를 대상으로 하는 SweRVolfX SoC 입니다. DDR2 인터페이스, CDC(클럭 도메인 교차) 장치, BSCAN 로직(JTAG 인터페이스용) 및 클럭 생성기를 추가합니다.  RVfpgaNexys 는 SweRVolf Nexys( <a href="https://github.com/chipsalliance/Cores-SweRVolf">https://github.com/chipsalliance/Cores-SweRVolf</a> )와 동일하지만 SweRVolf Nexys 는 SweRVolf 를 기반으로 합니다.
<b>RVfpgaSim</b>	시뮬레이션을 위한 테스트 벤치 래퍼(wrapper) 및 AXI 메모리가 있는 SweRVolfX SoC 입니다.

	RVfpgaSim 은 SweRVolf Sim( <a href="https://github.com/chipsalliance/Cores-SweRVolf">https://github.com/chipsalliance/Cores-SweRVolf</a> )과 동일하지만 SweRVolf Sim 은 SweRVolf 를 기반으로 합니다.
--	--

## 1. 소개

이 실습에서는 Zephyr(실시간 운영 체제)용 Tensorflow Lite 프로젝트를 빌드한 다음 SweRVolf 에서 해당 Zephyr 프로그램을 실행하는 방법을 보여줍니다. 이전 실습에서 본 것과 유사하게 기본 C 또는 어셈블리 언어 프로그램 대신 Zephyr 에서 Tensorflow 프로그램을 실행할 것입니다.

### 1. TensorFlow Lite 의 간략한 배경

TensorFlow Lite 는 개발자가 모바일, 임베디드 및 IoT 기기에서 모델을 실행할 수 있도록 지원하여 기기 내 머신 러닝을 지원하는 툴 세트입니다.

TensorFlow 모델을 바이너리 크기가 작은 .tflite 모델로 압축합니다. 이를 통해 기기 내 기계 학습이 가능하고 하드웨어 가속을 사용하여 성능이 향상됩니다.

주요 기능은 다음과 같습니다:

- 5 가지 주요 constraint(제약 사항)을 해결하여 on-device machine learning 최적화: 대기 시간(서버로 왕복하지 않음), 개인 정보 보호(기기에서 나가는 개인 데이터 없음), 연결성(인터넷 연결이 필요하지 않음), 크기(축소된 모델 및 바이너리 크기) 및 전력 소비(효율적인 추론 및 네트워크 연결 부족).
- 다중 플랫폼 지원: Android 및 iOS 장치, 임베디드 Linux 및 마이크로컨트롤러를 다룹니다.
- 다양한 언어 지원: Java, Swift, Objective-C, C++, Python 등등.
- 고성능: 하드웨어 가속 및 모델 최적화를 이용
- 여러 플랫폼에서 이미지 분류, 물체 감지, 포즈 추정, 질문 답변, 텍스트 분류 등과 같은 일반적인 기계 학습 작업에 대한 예제.

자세한 내용은 <https://www.tensorflow.org/lite/microcontrollers> 를 방문 하십시오.

## SweRVolf 및 Tensorflow Lite

그림 1 은 이 실습에서 구현할 Nexys A7 보드 상단의 계층적 레이어를 보여줍니다..

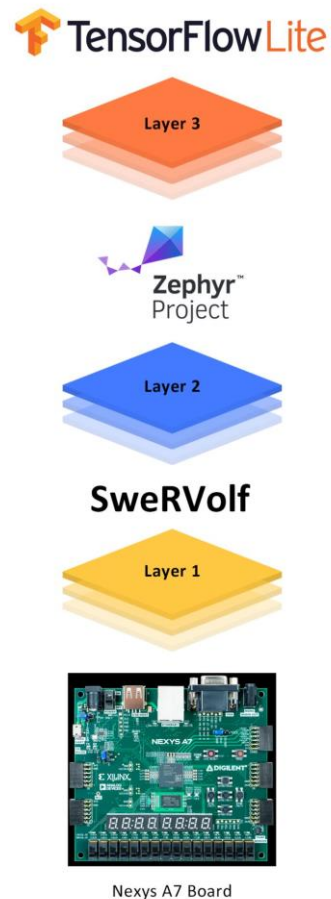


그림 1. FPGA 보드 상단의 레이어

Nexys A7 보드에서 TensorFlow Lite 프로그램을 실행하는 단계는 Lab 4 의 단계와 미묘하게 다릅니다.

### 1 단계. SweRVolf 를 FPGA 보드에 다운로드

먼저 FPGA 를 대상으로 하는 RISC-V 시스템인 Nexys A7 FPGA 보드에 SweRVolf 를 다운로드합니다. PlatformIO 를 사용하여 보드에 비트스트림을 업로드하거나, 연결되어 있는 경우 보드에 생성된 비트스트림을 업로드하는 FuseSoC 실행 명령을 사용하여 SweRVolf 를 보드에 다운로드합니다.

### 2 단계. Tensorflow 프로그램 빌드

이 단계에서는 Zephyr 용 Tensorflow Lite 애플리케이션을 빌드합니다. Zephyr RTOS 는 이 빌드의 일부로 빌드됩니다. 출력은 elf 파일입니다.

### 3 단계. SweRVolf 에서 프로그램을 로드합니다.

이번 단계에서는 2 단계에서 생성된 elf 파일을 SweRVolf 에 로드합니다.

## 2. 요구사항

이 실습을 완료하려면 다음을 설치해야 합니다:

- Vivado 2019.2 Web Pack (설치가이드(04 페이지) 참조)
- Verilator (v4.106) (설치가이드(08 페이지) 참조)
- FuseSoC (설치가이드(08 페이지) 참조)
- OpenOCD (RISC-V-specific version) (설치가이드(09 페이지) 참조)
- Zephyr Prerequisites (설치가이드(09 페이지) 참조)
- Zephyr SDK (v0.12.4) (설치가이드(10 페이지) 참조)
- PuTTY (설치가이드(10 페이지) 참조)

**중요:** RVfpga-SoC Labs 을 시작하기 전에 RVfpga-SoC 설치 가이드를 완료하는 것이 좋습니다.

아직 설치하지 않았다면 RVfpga-SoC 설치 가이드의 지침에 따라 Xilinx 의 Vivado 및 Verilator 를 설치하십시오. Imagination 의 대학 프로그램(IUP)에서 다운로드한 RVfpga-SoC 폴더를 컴퓨터에 복사했는지 확인합니다.

## 3. Tensorflow 의 Hello World 예제

이 실습에서는 Tensorflow 환경만 설정하고 간단한 Hello-World 텐서(tensor) 작업을 실행합니다.

Hello World 예제는 마이크로컨트롤러용 TensorFlow Lite 사용의 기본 사항을 보여주기 위해 설계되었습니다. 이 프로그램은 사인 함수를 복제하는 모델을 훈련하고 실행합니다. 즉, 단일 숫자를 입력으로 사용하고 숫자의 사인 값을 출력합니다.

자세한 내용은 이 링크에서 TensorFlow 의 공식 문서를 참조하세요. [link](#).

## 4. Tensorflow 를 위한 환경 설정

Ubuntu 터미널을 열고 다음 단계를 완료하십시오.

**1 단계. "SweRVolf" 디렉토리로 이동합니다.** 다음 Shell 변수를 설정해야 합니다. 이를 위해 다음을 실행합니다.

- `export WORKSPACE=$(pwd)`
- `export SWERVOLF_ROOT=$WORKSPACE/fusesoc_libraries/swervolf`
- `export ZEPHYR_BASE=$WORKSPACE/zephyr`

터미널 창에서 "`printenv < variable-name >`" 명령을 입력하여 Shell 변수가 성공적으로 설정되었는지 확인할 수도 있습니다.

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ export WORKSPACE=$(pwd)
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ export SWERVOLF_ROOT=$(pwd)/fusesoc_libraries/swervolf
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ export ZEPHYR_BASE=$WORKSPACE/zephyr
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 2. 셸 변수 설정

**2 단계. Tensorflow GitHub 저장소를 복제합니다.**

- `git clone https://github.com/tensorflow/tensorflow`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ git clone https://github.com/tensorflow/tensorflow
Cloning into 'tensorflow'...
remote: Enumerating objects: 1155704, done.
remote: Counting objects: 100% (193/193), done.
remote: Compressing objects: 100% (141/141), done.
remote: Total 1155704 (delta 62), reused 118 (delta 51), pack-reused 1155511
Receiving objects: 100% (1155704/1155704), 683.05 MiB | 157.00 KiB/s, done.
Resolving deltas: 100% (942622/942622), done.
Checking out files: 100% (25049/25049), done.
```

그림 3. Tensorflow

이제 "tensorflow" 디렉토리로 이동합니다.

- `cd tensorflow`

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ cd tensorflow/
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$
```

그림 4. "tensorflow" 디렉토리로 이동

다음 명령으로 저장소의 "v2.5.0" 분기를 확인하십시오.

- `git checkout -b v2.5.0`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$ git checkout -b v2.5.0
Switched to a new branch 'v2.5.0'
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$
```

## 그림 5. git checkout

**3 단계.** TensorFlow 에서 Zephyr SweRVolf 에 대한 지원을 추가하려면 이 TensorFlow 저장소에 몇 개의 파일을 복사해야 합니다.

첫 번째 파일은 hello\_world 예제의 **"Makefile.inc"** 파일입니다. 다음 경로로 이동하여 **"Makefile.inc"**를 복사합니다.

- [RVfpgaSoCPath]/RVfpgaSoC/Labs/LabResources/Lab5/Makefile.inc

이제 **"Makefile.inc"** 파일을 다음 위치에 붙여넣습니다(그림 6 참조).

- [RVfpgaSoCPath]/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/examples/hello\_world/zephyr\_riscv/

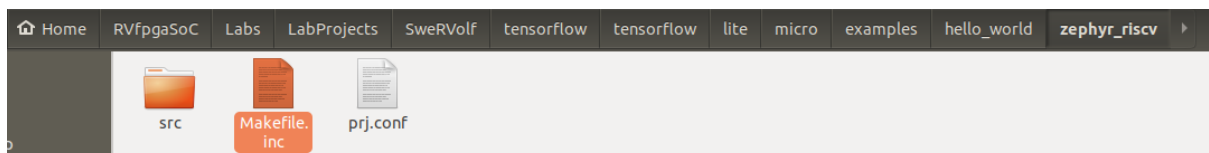


그림 6. Makefile.inc

두 번째 파일은 **"zephyr\_swervolf\_makefile.inc"** 파일입니다. 다음 경로로 이동하여 **"zephyr\_swervolf\_makefile.inc"**를 복사합니다.

- [RVfpgaSoCPath]/RVfpgaSoC/Labs/LabResources/Lab5/zephyr\_swervolf\_makefile.inc

이제 **"zephyr\_swervolf\_makefile.inc"** 파일을 다음 위치에 붙여 넣습니다(그림 7 참조).

- [RVfpgaSoCPath]/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/tools/make/targets/

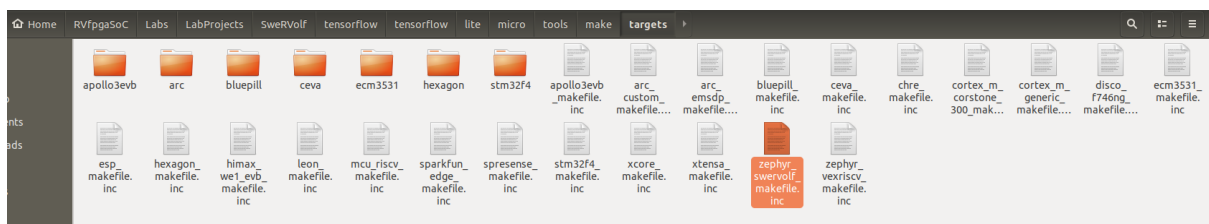


그림 7. zephyr\_swervolf\_makefile.inc

**4 단계.** 필요한 패키지를 설치합니다.

다음 명령을 사용하여 "WORKSPACE" 디렉토리로 이동합니다.

```
> cd ..
```

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$ cd ..
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 8. WORKSPACE 디렉터리로 이동

다음 명령으로 필요한 패키지를 설치합니다.

- `sudo apt install git cmake ninja-build gperf ccache dfu-util device-tree-compiler wget python python3-pip python3-setuptools python3-tk python3-wheel xz-utils file make gcc gcc-multilib locales tar curl unzip xxd make autoconf g++ flex bison virtualenv`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ sudo apt install git cmake ninja-build gperf ccache dfu-util device-tree-compiler wget python python3-pip python3-setuptools python3-tk python3-wheel xz-utils file make gcc gcc-multilib locales tar curl unzip xxd make autoconf g++ flex bison virtualenv
[sudo] password for hamza:
Reading package lists... Done
Building dependency tree
Reading state information... Done
autoconf is already the newest version (2.69-11).
bison is already the newest version (2:3.0.4.dfsg-1build1).
ccache is already the newest version (3.4.1-1).
device-tree-compiler is already the newest version (1.4.5-3).
flex is already the newest version (2.6.4-6).
make is already the newest version (4.1-9.1ubuntu1).
python is already the newest version (2.7.15-rc1-1).
python3-setuptools is already the newest version (39.0.1-2).
xz-utils is already the newest version (5.2.2-1.3).
dfu-util is already the newest version (0.9-1).
```

그림 9. 패키지 설치

**5 단계.** 가상 환경을 만듭니다.

먼저 다음 명령을 사용하여 zephyr 디렉터리로 이동합니다.

- `cd zephyr`

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ cd zephyr/
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$
```

그림 10. zephyr 디렉터리로 이동

다음 명령을 사용하여 zephyr 디렉터리 내에 가상 환경을 만듭니다.

- `virtualenv venv-zephyr`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$ virtualenv venv-zephyr
created virtual environment CPython3.6.9.final.0-64 in 374ms
creator CPython3Posix(dest=/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/venv-zephyr, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/hamza/.local/share/virtualenv)
added seed packages: pip==21.0.1, setuptools==54.2.0, wheel==0.36.2
activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$
```

그림 11. venv-zephyr 만들기

**Step 6.** 다음 명령어를 입력하여 마지막 단계에서 생성한 가상 환경을 활성화합니다.



```
> source venv-zephyr/bin/activate
```

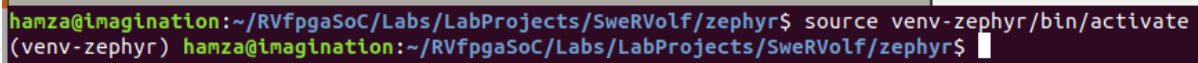


그림 12. venv-zephyr 활성화

**7 단계.** 다음 명령을 사용하여 "requirements.txt" 파일에 나열된 필수 패키지를 설치합니다.

```
> pip3 install -r scripts/requirements.txt
```

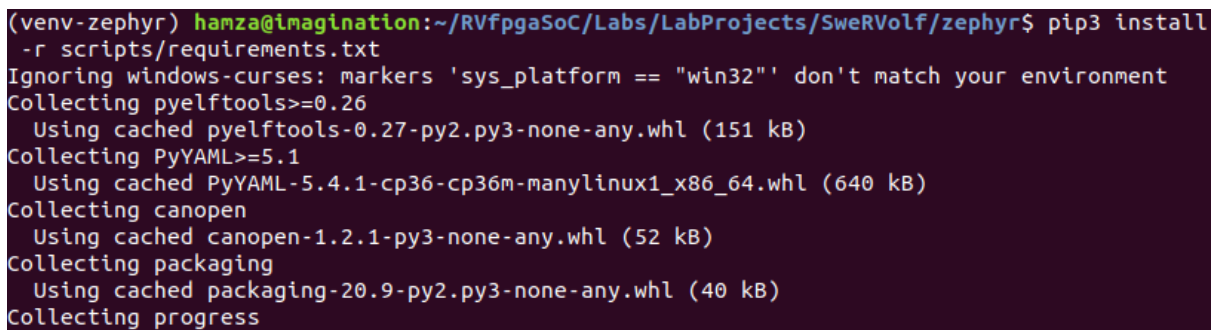


그림 13. 필수 패키지 설치

이제 이 터미널 탭을 닫고 "hello\_world" 예제를 빌드할 기본 터미널 탭으로 돌아갈 수 있습니다.

## 5. Swervolf 를 위한 Hello World 예제 빌드

이 섹션에서는 SweRVolf 에 대한 "hello\_world" 예제를 빌드합니다. "hello\_world" 예제에 대해 "zephyr.bin" 및 "zephyr.elf" 파일을 생성합니다.

**1 단계.** 먼저 tensorflow 디렉토리로 이동합니다.

```
> cd ../tensorflow/
```

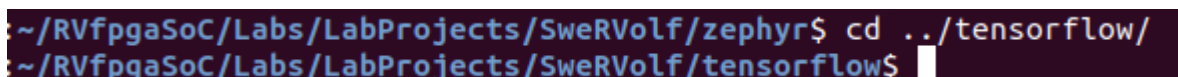


그림 14. "tensorflow" 디렉토리로 이동

**2 단계.** 이 실습에서는 SweRVolf 용 Hello World 를 구축할 것입니다. 이것은 다음 명령으로 수행됩니다.

```
> make -f tensorflow/lite/micro/tools/make/Makefile  
TARGET=zephyr_swervolf BUILD_TYPE=debug hello_world_bin
```

```
(venv-zephyr) hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$ make -f tensorflow/lite/micro/tools/make/Makefile
TARGET=zephyr_swervolf BUILD_TYPE=debug hello_world_bin
--2021-05-20 19:44:08-- http://mirror.tensorflow.org/github.com/google/flatbuffers/archive/dca12522a9f9e37f126ab925fd385c807ab4f84e.zip
Resolving mirror.tensorflow.org (mirror.tensorflow.org)... 216.58.210.80, 2a00:1450:4018:803::2010
Connecting to mirror.tensorflow.org (mirror.tensorflow.org)|216.58.210.80|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1760478 (1.7M) [application/zip]
Saving to: '/tmp/tmp.kvIHESzFqo/dca12522a9f9e37f126ab925fd385c807ab4f84e.zip'

/tmp/tmp.kvIHESzFqo/dca12522a9f9 100%[=====] 1.68M 1.83MB/s in 0.9s

2021-05-20 19:44:09 (1.83 MB/s) - '/tmp/tmp.kvIHESzFqo/dca12522a9f9e37f126ab925fd385c807ab4f84e.zip' saved [1760478/1760478]

Cloning into 'tensorflow/lite/micro/tools/make/downloads/pigweed'...
remote: Sending approximately 15.02 MiB ...
remote: Counting objects: 33, done
remote: Finding sources: 100% (33/33)
remote: Total 22687 (delta 9753), reused 22681 (delta 9753)
Receiving objects: 100% (22687/22687), 15.01 MiB | 1.78 MiB/s, done.
Resolving deltas: 100% (9753/9753), done.
Note: checking out '47268dff45019863e20438ca3746c6c62df6ef09'.
```

### 그림 15. hello\_world 빌드 예제

종속성에 대한 일부 틀 모음을 다운로드해야 하므로 몇 분 정도 걸립니다. 완료되면 다음과 같은 경로 내에 생성된 일부 폴더가 표시되어야 합니다.

- tensorflow/lite/micro/tools/make/gen/zephyr\_swervolf\_x86\_64\_debug/hello\_world/

이러한 폴더에는 생성된 프로젝트 및 소스 파일이 포함됩니다.

```
[ 98%] Building C object zephyr/CMakeFiles/zephyr_final.dir/misc/empty_file.c.obj
[ 98%] Building C object zephyr/CMakeFiles/zephyr_final.dir/isr_tables.c.obj
[100%] Linking CXX executable zephyr.elf
Generating files from zephyr.elf for board: swervolf_nexys
make[3]: Leaving directory '/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/build'
[100%] Built target zephyr_final
make[2]: Leaving directory '/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/build'
make[1]: Leaving directory '/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/build'
```

### 그림 16. hello\_world 예제 빌드 완료

결과 바이너리(zephyr.bin 및 zephyr.elf)는 다음 경로에서 생성됩니다.

- tensorflow/lite/micro/tools/make/gen/zephyr\_swervolf\_x86\_64\_debug/hello\_world/build/zephyr

**3 단계.** 이제 다음 명령을 입력하여 가상 환경을 종료할 수 있습니다.

➤ deactivate

## 6. Verilator 에서 Hello World 예제 실행하기

이 섹션에서는 "zephyr.bin" 파일을 ".hex" 파일로 변환한 다음 SweRVolf 용 시뮬레이터를 실행하는 동안 초기 램(ram) 파일로 로드합니다.

1 단계. "hello\_world" 프로젝트 디렉토리로 이동합니다. 다음 명령을 입력하여 해당 디렉토리를 입력하십시오.

```
> cd
tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_de
bug/hello_world/
```

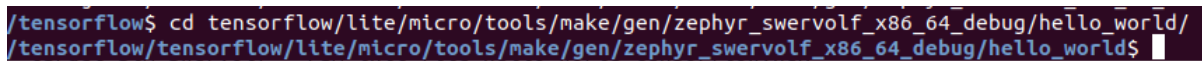


그림 17. "hello\_world" 프로젝트 경로

2 단계. ".bin" 파일을 ".hex" 파일로 변환합니다. ".hex" 파일을 생성하려면 hello\_world 디렉토리에서 다음 명령을 실행하십시오 :

```
> python3 $SWERVOLF_ROOT/sw/makehex.py build/zephyr/zephyr.bin
>
/home/<Username>/RVfpgaSoC/Labs/LabProjects/SweRVolf/hello_worl
d_tensorflow.hex
```

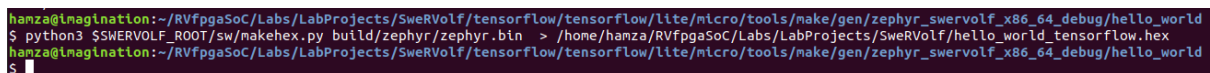


그림 18. ".bin"을 ".hex"로 변환

3 단계. "WORKSPACE" 디렉토리로 이동합니다.

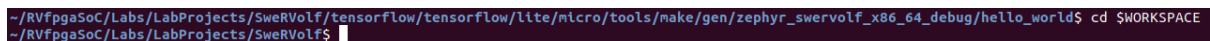


그림 19. venv-zephyr 에 패키지 설치하기

4 단계. 시뮬레이터에서 ".hex" 파일을 로드합니다.

```
> fusesoc run --target=sim swervolf --
ram_init_file=hello_world_tensorflow.hex
```

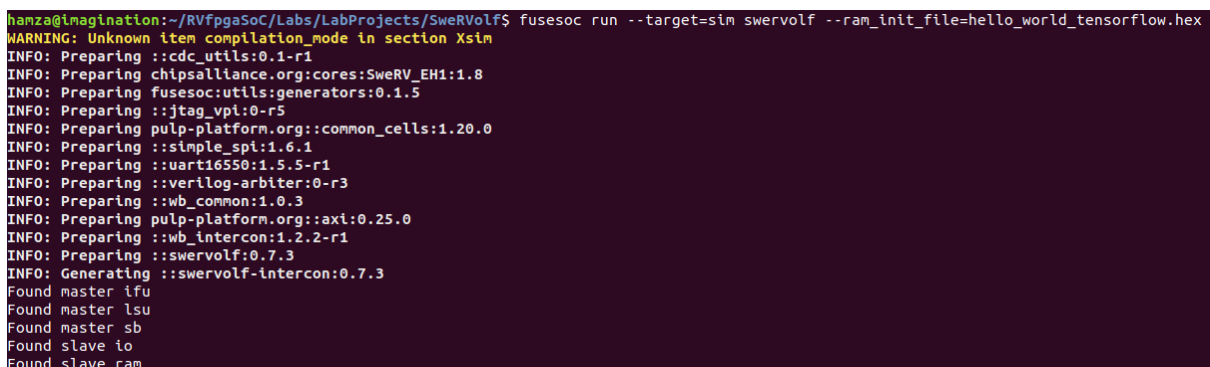


그림 20. 시뮬레이터에서 ".hex" 파일 로드

hello\_world 예제의 출력을 볼 수 있습니다(그림 21 참조). 프로그램은 "sine" 함수의 X 및 Y 값을 인쇄합니다.

```
g++ jtagServer.o jtag_common.o tb.o verilated.o verilated_dpi.o verilated_vcd_c.o Vswervolf_core_tb_ALL.a -o Vswervolf_core_tb
make[1]: Leaving directory '/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/build/swervolf_0.7.3/sin-verilator'
INFO: Running
INFO: Running simulation
Loading RAM contents from /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/hello_world_tensorflow.hex
Releasing reset
*** Booting Zephyr OS build zephyr-v2.4.0 ***
x_value: 1.0*2^-127, y_value: 1.0*2^-127

x_value: 1.2566366*2^-2, y_value: 1.4910772*2^-2

x_value: 1.2566366*2^-1, y_value: 1.1183078*2^-1

x_value: 1.8849551*2^-1, y_value: 1.677462*2^-1

x_value: 1.2566366*2^0, y_value: 1.9316229*2^-1
```

그림 21. "hello\_world" 출력

"ctrl + c"를 눌러 프로그램을 종료합니다.

## 7. Nexys A7 보드에서 Hello World 예제 실행하기

이 섹션에서는 OpenOCD 를 사용하여 보드에서 "hello\_world" 프로젝트를 실행합니다.

**1 단계.** Nexys A7 보드를 컴퓨터에 연결한 다음 Workspace 디렉터리에서 FPGA 빌드 명령을 실행합니다.

➤ `fusesoc run --target=nexys_a7 --run swervolf`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ fusesoc run --target=nexys_a7 --run swervolf
WARNING: Unknown item compilation_mode in section Xsin
INFO: Running
export HW_TARGET=; \
export JTAG_FREQ=; \
vivado -quiet -nolog -notrace -mode batch -source swervolf_0.7.3_pgm.tcl -tclargs xc7a100tcs9324-1 swervolf_0.7.3.bit
FuseSoC Xilinx FPGA Programming Tool
=====
```

그림 22. FPGA 빌드 실행

**2 단계.** OpenOCD 를 SweRVolf 와 연결합니다.

➤ `openocd -f $SWERVOLF_ROOT/data/swervolf_nexys_debug.cfg`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ openocd -f $SWERVOLF_ROOT/data/swervolf_nexys_debug.cfg
Open On-Chip Debugger 0.11.0-rc1+dev-01535-g3651cbfd (2021-01-30-12:10)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
DEPRECATED! use 'adapter driver' not 'interface'
DEPRECATED! use 'adapter speed' not 'adapter khz'
Info : ftdi: if you experience problems at higher adapter clocks, try the command "ftdi_tdo_sample_edge falling"
Info : clock speed 10000 kHz
Info : JTAG tap: riscv.cpu tap/device found: 0x13631093 (mfg: 0x049 (Xilinx), part: 0x3631, ver: 0x1)
Info : datacount=2 progbufsize=0
Warn : We won't be able to execute fence instructions on this target. Memory may not always appear consistent. (progbufsize=0, impebreak=0)
Info : Examined RISC-V core; found 1 harts
Info : hart 0: XLEN=32, misa=0x40001104
Info : starting gdb server for riscv.cpu on 3333
Info : Listening on port 3333 for gdb connections
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
```

그림 23. OpenOCD 연결

**3 단계.** "Ctrl + Shift + t"를 사용하여 새 터미널을 열고 다음 명령을 사용하여 OpenOCD 를 통해 디버그 세션에 연결합니다.

➤ telnet localhost 4444

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ telnet localhost 4444
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Open On-Chip Debugger
> █
```

그림 24. telnet 로컬 호스트 4444

OpenOCD 는 `load_image /path/to/file.elf` 를 실행하여 ELF 프로그램 파일 로드를 지원합니다. 경로는 OpenOCD 가 시작된 디렉토리에 상대적임을 기억하십시오.

➤ load\_image  
tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr\_swervolf\_x86\_64\_debug/hello\_world/build/zephyr/zephyr.elf

```
> load_image tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/build/zephyr/zephyr.elf
287072 bytes written at address 0x00000000
downloaded 287072 bytes in 27.439606s (10.217 KiB/s)
> █
```

그림 25. ".elf" 파일 로드

프로그램이 로드된 후 다음 명령을 사용하여 프로그램 카운터를 주소 0 으로 설정합니다.

➤ reg pc 0

```
> reg pc 0
pc (/32): 0x00000000
```

그림 26. 프로그램 카운터를 0 으로 설정

이제 다음 명령을 사용하여 프로그램을 시작하십시오.

➤ resume

```
> resume
> █
```

그림 27. 프로그램 시작

**4 단계.** "Ctrl + Shift + t"를 사용하여 새 터미널을 엽니다. 명령을 사용하여 "PuTTY"를 엽니다.

➤ sudo putty

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ sudo putty
(puTTY:3263): Gtk-CRITICAL **: 12:18:45.869: gtk_box_gadget_distribute: assertion 'size >= 0' failed in GtkScrollbar
```

그림 28. PuTTY 열기

여기에서 PuTTY 를 Nexys A7 보드용 직렬 콘솔로 사용할 것입니다.

**5 단계.** 다음 구성을 설정합니다 :

연결 유형을 "**Serial**"로 선택한 다음 **"/dev/ttyUSB1"**을 serial line 으로 입력하고 속도를 **"115200"**으로 설정합니다. 이제 "Open"를 클릭하여 직렬 콘솔을 시작합니다.

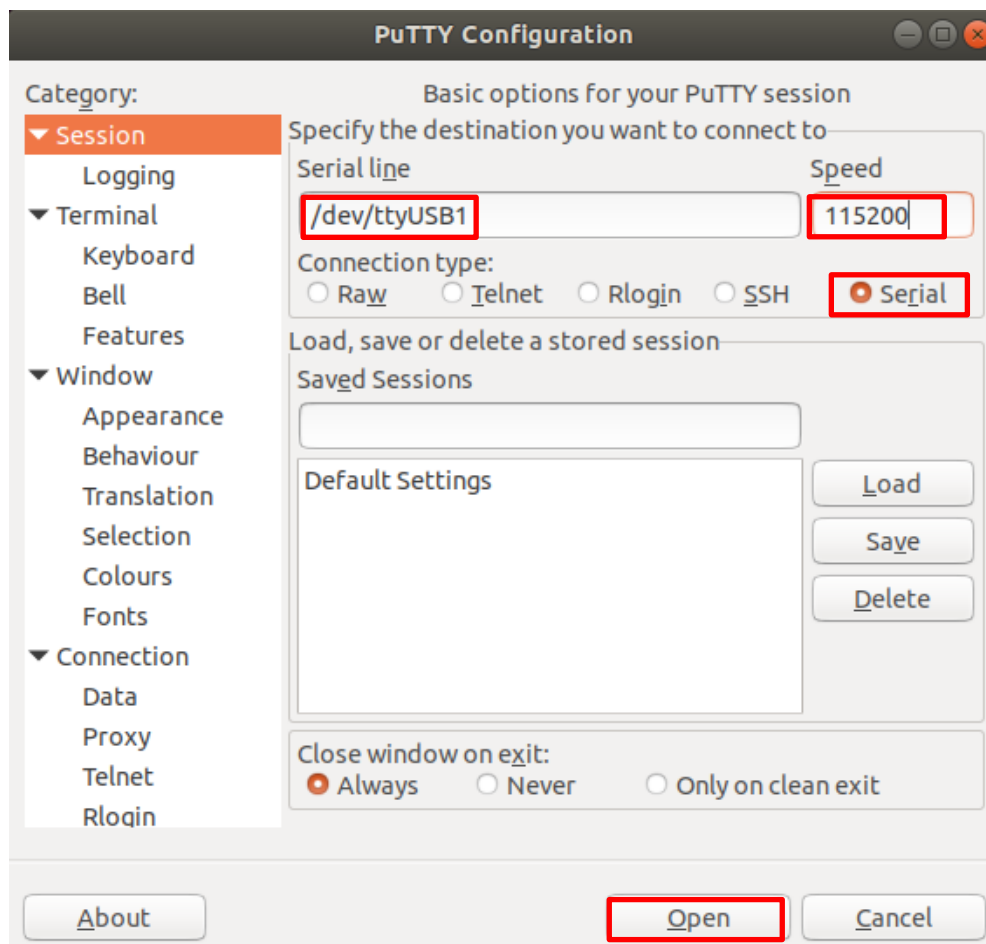


그림 29. PuTTY 구성

직렬 콘솔에서 hello\_world 예제의 출력을 볼 수 있습니다(그림 30 참조).

```

/dev/ttyUSB1 - PuTTY
x_value: 1.0995567*2^2, y_value: -1.9316229*2^-1
x_value: 1.1780966*2^2, y_value: -1.1098361*2^0
x_value: 1.2566366*2^2, y_value: -1.965511*2^-1
x_value: 1.3351763*2^2, y_value: -1.4910772*2^-1
x_value: 1.4137159*2^2, y_value: -1.0674761*2^-1
x_value: 1.4922558*2^2, y_value: -1.4233011*2^-2
x_value: 1.0*2^-127, y_value: 1.0*2^-127
x_value: 1.2566366*2^-2, y_value: 1.4910772*2^-2
x_value: 1.2566366*2^-1, y_value: 1.1183078*2^-1
x_value: 1.8849551*2^-1, y_value: 1.677462*2^-1
x_value: 1.2566366*2^0, y_value: 1.9316229*2^-1
x_value: 1.5707957*2^0, y_value: 1.0420598*2^0
x_value: 1.8849551*2^0, y_value: 1.9146791*2^-1
x_value: 1.0995567*2^1, y_value: 1.6435742*2^-1
x_value: 1.2566366*2^1, y_value: 1.0674761*2^-1
x_value: 1.4137159*2^1, y_value: 1.8977352*2^-3
x_value: 1.5707957*2^1, y_value: 1.0844199*2^-7
x_value: 1.7278753*2^1, y_value: -1.2199725*2^-2
x_value: 1.8849551*2^1, y_value: -1.0674761*2

```

**그림 30. 직렬 콘솔**

시뮬레이션 섹션에서 보았듯이 프로그램은 TensorFlow 모델이 그리는 사인 함수의 "X" 및 "Y" 좌표를 인쇄합니다.

**참고:** 직렬 콘솔을 열 수 없는 경우 직렬 라인으로 `"/dev/ttyUSB0"`을 입력 하십시오.

따라서 이 실습에서는 TensorFlow 의 "hello\_world" 예제를 Zephyr 애플리케이션으로 성공적으로 빌드한 다음 SweRVolf 에서 해당 예제를 실행했습니다.