Building SONiC with Intel® P4 Studio SDE 9.7 for $P4_{16}$ Intel® Tofino™-Based Bare-Metal Switches

Application Note

Nov 2021

Intel Confidential

Legal Disclaimer

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel, Tofino, and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contents

Revision History

Date	Revision	Description
Nov 2021	001	Initial release

1 Introduction

This document contains instructions to build SONiC with Intel® P4 Studio Software Development Kit (SDE) 9.7.

General Note: The terms “Intel P4 Studio Software Development Environment”, “ Intel P4 Studio SDE”, and “Intel P4 Studio” are all used in this document, and in all cases, they refer to the same product: Intel P4 Studio Software Development Environment.

See the Intel P4 Studio SDE release notes for release-specific SONiC information, including which profiles are supported.

NOTE: Python3 support was added in Intel P4 Studio SDE 9.4.0. These instructions now use `python3` instead of `python`.

This document covers the following key items:

- Downloading essential software
- Intel P4 Studio SDE build steps, including building the P4 profiles
- Building SONiC
- Installing and running SONiC
- SONiC notes including how to enter the Barefoot shell, quagga shell, key logfiles, and how to access the Redis database.
- How to switch between different profiles in SONiC

1.1 Downloading Essential Software

Before building SONiC, take the following steps to create the correct build environment:

Download SDE, BSP, and scripts:

- Intel P4 Studio SDE 9.7.0: Intel Part # 669803
- SDE 9.7.0 BSP (either Wedge or custom BSP): Intel Part # 669804
- Download the scripts: Intel Part # 686219

1.1.1 List of Files in the Scripts zip File

The `scripts-for-sonic-sde-<SDE_number>.zip` file will contain:

- `make_sde_deb.sh`
- `make_platform_deb.sh`
- `uname-deb10`

1.1.2 Expected Files After Download is Complete

After the downloads are complete, the following files will have been downloaded:

- `bf-reference-bsp-9.X.Y.tgz`
- `bf-sde-9.X.Y.tgz`
- `make_sde_deb.sh`
- `make_platform_deb.sh`
- `uname-deb10`

Move these files from the download directory to the build directory (named "sde" in the list below) so that the directory structure will be:

```
sde

        ├── bf-reference-bsp-9.X.Y.tgz

        ├── bf-sde-9.X.Y.tgz

        ├── make_sde_deb.sh

        ├── make_platform_deb.sh

        └── uname-deb10
```

(*X* and *Y* stand for the specific release numbers.)

1.2 The Debian 10 Build Environment

There are two options available to build $P4_{16}$ profiles from the Intel P4 Studio SDE release package in Debian 10 (Buster):

1. In Debian 10 as a Host OS or VM
2. In Debian 10 Docker container (see Section 1.2.1 - "Using a Debian 10 Docker Container")

For option 1, no additional steps are required. For option 2 setup is required. Setup and also directions for re-entering the docker container after exit are in the next section.

1.2.1 Using a Debian 10 Docker Container

When the Docker Container option is chosen, then it must be setup before use. This section also includes directions to re-enter the Docker Container as needed.

1.2.1.1 Setting up A Debian 10 Docker Container

These steps are only needed if a Docker Container is used to build the $P4_{16}$ profile.

```
DEV_USER="\
    apt update;
    apt install -y sudo;
    echo '%sudo ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers;
    adduser --disabled-password --gecos '' ${USER};
    adduser ${USER} sudo;
    groupmod -g $(id -g ${USER}) ${USER};
    usermod -u ${UID} -g $(id -g ${USER}) ${USER};
    su ${USER};
"
docker run -v $(readlink -f .):/sde --name debian-buster-sde -ti
debian:buster bash -c "${DEV_USER}"

cd /sde
sudo mv /bin/uname /bin/uname.old
sudo cp uname-deb10 /bin/uname
sudo apt install -y dh-make python
sudo apt install -y linux-headers-amd64
```

1.2.1.2 Entering a Debian 10 Docker Container

After following the steps above, the docker container will have been started. If you exit the docker container shell, use the following commands to re-enter the shell.

```
# (Optional) Start docker container
docker start debian-buster-sde

# (Optional) Enter docker container shell
docker exec -ti --privileged -u $USER debian-buster-sde bash
```

2 SDE Build Steps

This section describes how to set up the Intel P4 Studio SDE build environment, and then build the P4 profiles.

Note: All the commands of this section are executed in Debian 10 Docker container or Debian 10 host OS.

2.1 Setup the Intel P4 Studio SDE Build Environment

Note: The old build tool `p4studio_build` is no longer available from SDE 9.6.0, the replacement is new build tool `p4studio`. Please refer to *P4 Studio SDE Build Tool (p4studio) User Guide* (#631379) for detailed information about the `p4studio` tool.

Note: The dependencies installation takes about 30 minutes, depending on the performance and network of the build host.

```
cd /sde

# Setup build env
buildNumber=9.7.0
WORKSPACE=`pwd`
export SDE=$WORKSPACE/bf-sde-$buildNumber
export SDE_INSTALL=$SDE/install
export PATH=$SDE_INSTALL/bin:$PATH

# Extract SDE sources
tar -vxzf bf-sde-$buildNumber.tgz

# To install SDE dependencies
cd $SDE/p4studio
sudo ./install-p4studio-dependencies.sh
./p4studio dependencies install
```

2.2 Building P4 Profiles

There are two options for building P4 profiles:

1. Building a single P4 profile

3. Building multiple P4 profiles

After the profile(s) are built, the Intel P4 Studio SDE build is complete. The next step is to build SONiC (see Section 3 - “Building SONiC”).

2.2.1 Option1: Building a Single P4 Profile

In this option, build the Intel P4 Studio SDE with a single P4 profile, for example, the X1_PROFILE for Tofino:

```
# Build SDE with specific profile(e.g Tofino X1 profile)

cd $SDE/p4studio

./p4studio configure switch asic --bsp-path bf-reference-bsp-$buildNumber
.tgz
./p4studio build x1_tofino
```

Create the Platform and SAI .deb packages:

```
export USER="root"
cd $SDE
mkdir -p tools/sonic
cp ../make_platform_deb.sh tools/sonic/
cp ../make_sde_deb.sh tools/sonic/
```

```
cd $SDE/tools/sonic
chmod +x make_platform_deb.sh
chmod +x make_sde_deb.sh

./make_platform_deb.sh
./make_sde_deb.sh
```

After these steps the following files will have been created: `bfnplatform_1.0.0_amd64.deb` and `bfnsdk_1.0.0_amd64.deb.`

2.2.2 Option 2: Build Multiple P4 Profiles

This procedure can support building a unified SONiC Debian package that contains multiple P4 data planes.

The data planes are enabled by various profiles available with the `switch.p4-16` program as part of Intel P4 Studio Software Development Environment. Some or all available `switch.p4-16` profiles can be added to SONiC Debian package.

Select P4 profile and start to build Intel P4 Studio SDE (for example Tofino X1_PROFILE and X2_PROFILE):

```
# build Tofino X1 profile

cd $SDE/p4studio
./p4studio configure switch asic --bsp-path bf-reference-bsp-
$buildNumber.tgz
./p4studio build x1_tofino

cd $SDE
cp -r install install_x1_profile

# build Tofino X2 profile

cd $SDE/p4studio
./p4studio build x2_tofino

cd $SDE
cp -r install install_x2_profile
```

Create Platform and SAI .deb packages:

```
export USER="root"
cd $SDE
mkdir -p tools/sonic
cp ../make_platform_deb.sh tools/sonic/
cp ../make_sde_deb.sh tools/sonic/

cd $SDE/tools/sonic
chmod +x make_platform_deb.sh
chmod +x make_sde_deb.sh
```

Pack multiple profiles into one Debian file, and specify the default profile:

```
./make_platform_deb.sh -p x1_profile -p x2_profile --default-profile
x1_profile

./make_sde_deb.sh -p x1_profile -p x2_profile --default-profile
x1_profile
```

After these steps the following files will have been created: `bfnplatform_1.0.0_amd64.deb` and `bfnsdk_1.0.0_amd64.deb`.

2.2.3 Build Tofino2 P4 Profiles

This procedure can support building a unified SONiC Debian package that contains multiple P4 data planes.

The data planes are enabled by various profiles available with the `switch.p4-16` program as part of Intel P4 Studio Software Development Environment. Some or all available `switch.p4-16` profiles can be added to SONiC Debian package.

Select P4 profile and start to build Intel P4 Studio SDE (for example Y1_PROFILE):

```
# build Tofino2 Y1 profile

cd $SDE/p4studio
./p4studio configure switch asic newport --bsp-path bf-reference-bsp-
$buildNumber.tgz
./p4studio build y1_tofino2
```

Create Platform and SAI .deb packages:

```
export USER="root"
cd $SDE
mkdir -p tools/sonic
cp ../make_platform_deb.sh tools/sonic/
cp ../make_sde_deb.sh tools/sonic/

cd $SDE/tools/sonic
chmod +x make_platform_deb.sh
chmod +x make_sde_deb.sh

./make_platform_deb.sh
./make_sde_deb.sh
```

After these steps the following files will have been created: `bfnplatform_1.0.0_amd64.deb` and `bfnsdk_1.0.0_amd64.deb`.

You also can building a unified SONiC Debian package that contains multiple P4 data planes for Tofino2, the procedure is similar as Section 2.2.2 - "Option 2: Build Multiple P4 Profiles".

3 Building SONiC

Note: All the commands of this section are executed outside the Docker container.

First, get the build image from the SONiC code repository:

```
# Get SONiC code repository

git clone https://github.com/Azure/sonic-buildimage

cd sonic-buildimage/

export SONIC_BUILD=`pwd`

# (Optional) Checkout a specific commit id.

git checkout 48ba459f9

# git submodule update

git submodule update --init --recursive
```

Note: The commit ID 48ba459f9 is tested starting with Intel P4 Studio SDE 9.7.0.

Install the Debian packages and specify the PATH:

```
# Copy SDE/BSP deb file to your SONiC build directory.

docker cp debian-buster-sde:/sde/bf-sde-
9.7.0/tools/sonic/bfnsdk_1.0.0_amd64.deb ${SONIC_BUILD}

docker cp debian-buster-sde:/sde/bf-sde-
9.7.0/tools/sonic/bfnplatform_1.0.0_amd64.deb ${SONIC_BUILD}
```

Edit `the bfn-sai.mk` makefile to specify the Debian packages PATH:

```
vim platform/barefoot/bfn-sai.mk
```

Add the following text to the `bfn-sai.mk` makefile:

```
BFN_SAI = bfnsdk_1.0.0_amd64.deb
$(BFN_SAI)_PATH = /sonic

$(BFN_SAI)_DEPENDS += $(LIBNL_GENL3_DEV)
$(eval $(call add_conflict_package,$(BFN_SAI),$(LIBSAIVS_DEV)))
$(BFN_SAI)_RDEPENDS += $(LIBNL_GENL3)

#SONIC_ONLINE_DEBS += $(BFN_SAI)
SONIC_COPY_DEBS +=$(BFN_SAI)
$(BFN_SAI_DEV)_DEPENDS += $(BFN_SAI)
```

Edit the `bfn-platform.mk` makefile:

```
vim platform/barefoot/bfn-platform.mk
```

Add the following text to `bfn-platform.mk`:

```
BFN_PLATFORM = bfnplatform_1.0.0_amd64.deb
$(BFN_PLATFORM)_PATH = /sonic

#SONIC_ONLINE_DEBS += $(BFN_PLATFORM)
SONIC_COPY_DEBS += $(BFN_PLATFORM)
$(BFN_SAI_DEV)_DEPENDS += $(BFN_PLATFORM)
```

3.1 Start SONiC Build

Note: The SONiC build may take about 2-3 hours to complete, depending on the performance and network of the build host.

Execute `make init` once after cloning the repository:

```
make init
```

Load overlay module (only the first time):

```
sudo modprobe overlay
```

Configure the Barefoot Networks platform:

```
make configure PLATFORM=barefoot
```

Build the SONiC image:

```
make target/sonic-barefoot.bin
```

After the build, `sonic-barefoot.bin` will have been created in the `${SONIC_BUILD}/target` directory.

4 Install and Run SONiC

SONiC can be installed and run from either the ONIE environment or the SONiC environment.

4.1 Install SONiC From ONIE Environment

Note: If ONIE is not installed, check the ODM instructions to obtain and install latest ONIE.

Note: If a NOS is already installed, then it must be erased and re-installed. During the boot process, in the ONIE boot menu: choose "ONIE: Uninstall OS" to erase the existing NOS. Reboot the system and choose "ONIE: Install OS" in the ONIE boot menu.

After ONIE has booted, stop auto discovery using the command:

```
onie-discovery-stop
```

Copy the generated `sonic-barefoot.bin` to ONIE via `scp` or a USB device:

```
scp <user>@<ip>:<path>/sonic-barefoot.bin sonic-barefoot.bin
```

Install the SONiC image:

```
chmod +x sonic-barefoot.bin

./sonic-barefoot.bin
```

Manually reboot under ONIE after SONiC installation succeeds.

```
reboot
```

After installation is complete, log in with these credentials:

```
admin/YourPaSsWoRd
```

4.2 Install New SONiC Image From SONiC Environment

Note: If you already have a SONiC installed, you also can install SONiC image from the SONiC environment as an option.

```
# Copy the generated sonic-barefoot.bin to target via scp or USB disk.

scp <user>@<ip>:<path>/sonic-barefoot.bin sonic-barefoot.bin.

# Install the SONiC image

sudo sonic_installer install sonic-barefoot.bin
```

5 Other SONiC Notes

To enter the Barefoot shell:

```
docker exec -ti syncd /opt/bfn/install/bin/bfshell
```

To enter the `frr` shell, use the `vtysh (docker exec -ti bgpd vtysh)` command.

For general system log: view `/var/log/syslog`.

For the SAI calling log: view `/var/log/swss/sairedis.rec`.

To access the Redis Database: use the `redis-cli` command.

6 Switching Between Different Profiles in SONiC

Edit `/etc/sonic/config_db.json` to include the `p4_profile` attribute (see example below).

Note: If the p4_profile attribute is not specified in `config_db.json`, the switch will use the default profile which is specified when creating SAI deb packages.

Note: the profile name must match the install folder name: **`install_<profile name>`** ***. For example, if x1_profile is the profile name, then the install folder name is*** **`install_x1_profile`*****.***

```
"DEVICE_METADATA": {
    "localhost": {
        "bgp_asn": "65100",
        "hostname": "sonic",
        "hwsku": "montara",
        "mac": "00:90:fb:60:e2:26",
        "platform": "x86_64-accton_wedge100bf_32x-r0",
        "p4_profile": "x1_profile",
        "type": "LeafRouter"
    }
},
```

Load the updated `config_db.json` by using the command `sudo config load -y`.

Reboot the switch for the new profile to take effect.

7 Upgrading/Deploy Barefoot Intel P4 Studio SDE Debian Package in SONiC

```
# Copy deb packages to your SONiC box
SONIC_BOX_IPADDR="dd.dd.dd.dd"

scp bfnsdk_1.0.0_amd64.deb admin@${SONIC_BOX_IPADDR}:/home/admin/
scp bfnplatform_1.0.0_amd64.deb admin@${SONIC_BOX_IPADDR}:/home/admin/

# Login to the box, deploy SDE and restart syncd

# Copy the deb packages to syncd container
docker cp bfnsdk_1.0.0_amd64.deb syncd:/
docker cp bfnplatform_1.0.0_amd64.deb syncd:/

# Enter syncd container shell, and install the SDE deb packages.
docker exec -ti syncd bash
dpkg -i bfnsdk_1.0.0_amd64.deb
dpkg -i bfnplatform_1.0.0_amd64.deb

exit

# Restart system or swss service
sudo systemctl restart swss
```