altera
An Intel Company

FPGA AI Suite

Getting Started Guide

Updated for FPGA AI Suite: **2024.3**

Online Version **768970**
Send Feedback **2024.12.16**

Contents

1. FPGA AI Suite Getting Started Guide

The *FPGA AI Suite Getting Started Guide* provides the following information:

- An overview of the FPGA AI Suite
- Installation instructions and a list of prerequisites
- A tutorial that walks you through the process of running inference on a ResNet-50 graph, including performance and area estimation

About the FPGA AI Suite Documentation Library

Documentation for the FPGA AI Suite is split across a few publications. Use the following table to find the publication that contains the FPGA AI Suite information that you are looking for:

Table 1. FPGA AI Suite Documentation Library

Title and Description	
Release Notes Provides late-breaking information about the FPGA AI Suite including new features, important bug fixes, and known issues.	Link
Getting Started Guide Get up and running with the FPGA AI Suite by learning how to initialize your compiler environment and reviewing the various design examples and tutorials provided with the FPGA AI Suite	Link
IP Reference Manual Provides an overview of the FPGA AI Suite IP and the parameters you can set to customize it. This document also covers the FPGA AI Suite IP generation utility.	Link
Compiler Reference Manual Describes the use modes of the graph compiler (`dla_compiler`). It also provides details about the compiler command options and the format of compilation inputs and outputs.	Link
PCIe-based Design Example User Guide Describes the design and implementation for accelerating AI inference using the FPGA AI Suite, Intel® Distribution of OpenVINO™ toolkit, and a Terasic* DE10-Agilex Development Board.	Link
SoC-based Design Example User Guide Describes the design and implementation for accelerating AI inference using the FPGA AI Suite, Intel Distribution of OpenVINO toolkit, and an Arria® 10 SX SoC FPGA Development Kit (DK-SOC-10AS066S) or Agilex™ 7 FPGA I-Series Transceiver-SoC Development Kit.	Link

Intel Distribution of OpenVINO toolkit Requirement

To use the FPGA AI Suite, you must be familiar with the Intel Distribution of OpenVINO toolkit.

FPGA AI Suite Version 2024.3 requires the Intel Distribution of OpenVINO toolkit Version 2023.3 LTS. For OpenVINO documentation, refer to https://docs.openvino.ai/2023.3/documentation.html.

2. FPGA AI Suite Components

The FPGA AI Suite consists of several components to help you enable AI on your Intel FPGAs.

The FPGA AI Suite consists of the following components:

- **IP**

  The FPGA AI Suite IP is an RTL-instantiable IP with AXI interfaces. These interfaces can be instantiated into a generic FPGA system.

  For a list of models supported by the FPGA AI Suite IP, refer to "Supported Models" in the *FPGA AI Suite IP Reference Manual*

- **Compiler** (`dla_compiler` command)

  The FPGA AI Suite compiler is a multipurpose tool that you can use for the following tasks with the FPGA AI Suite:

  — **Generate architectures**

  Use the compiler to generate an IP parameterization that is optimized for a given machine learning (ML) model or set of models, while attempting to fit the FPGA AI Suite IP block into a given resource footprint.

  — **Estimate IP performance**

  Use the compiler to get an estimate of the performance of the AI IP block for a given parameterization.

  — **Estimate IP FPGA resource consumption**

  Use the compiler to get an estimate of the FPGA resources (ALMs, M20k blocks, and DSPs) required for a given IP parameterization.

  — **Create an ahead-of-time compiled graph**

  Create a compiled version of an ML model that includes the instructions necessary to control the IP on the FPGA. The compiled binary includes both the instructions to control the IP during inference and the model weights. This compiled model is suitable for use by the Example Designs or in a production deployment.

  An OpenVINO plugin is distributed with the compiler that allows the PCIe Example Design to access the compiler in a just-in-time fashion.

- **IP generation tool**

The FPGA AI Suite IP generation tool customizes the FPGA AI Suite IP based on an input architecture file. The generated IP is placed into an IP library that you can import into an FPGA design with Platform Designer, use directly in a pure RTL design, or both.

- **Example designs**

The following design examples show how you can use the FPGA AI Suite IP:

- **PCIe-based design example for Agilex 7 devices**

  This design example targets the Terasic DE10-Agilex Development Board (DE10-Agilex-B2E2). This example is distributed with the FPGA AI Suite.

  This example highlights how to instantiate the FPGA AI Suite IP and provides an example runtime that shows you how to control the FPGA AI Suite IP and pass the model configuration to the IP to program it and run inference on a neural network.

- **SoC design example for Arria 10 devices**

  This design example is based on the Arria 10 SX SoC FPGA Development Kit (DK-SOC-10AS066S). This example is distributed with the FPGA AI Suite.

  This example demonstrates how to use the ARM* SoC as a host for the FPGA AI Suite, as well as how to perform inference in a streaming data flow.

- **SoC design example for Agilex 7 devices**

  This design example is based on the Agilex 7 FPGA I-Series Transceiver-SoC Development Kit (DK-SI-AGI027FC). This example is distributed with the FPGA AI Suite.

  This example demonstrates how to use the ARM SoC as a host for the FPGA AI Suite, as well as how to perform inference in a streaming data flow.

- **Streaming hostless design examples for Agilex 7 devices**

  This design example is based on the Agilex 7 FPGA I-Series Development Kit ES2 (DK-DEV-AGI027RBES). This example is distributed with the FPGA AI Suite.

  This example demonstrates how to create an instance of the FPGA AI Suite IP that does not have any dependency on external memory. Input (output) data are streamed into (out of) the IP. The IP is manually controlled by using the Quartus® Prime System Console.

- **JTAG design example for Agilex 5 E-Series devices**

  This design example is based on the Agilex 5 FPGA E-Series 065B Premium Development Kit (DK-A5E065BB32AES1). This example is distributed with the FPGA AI Suite.

  This example demonstrates how to instantiate one instance of the FPGA AI Suite IP on anAgilex 5 E-Series device. It places configurations and data on external DDR memory and allows an external host to interact with the memory and FPGA AI Suite IP via JTAG.

- **ARM-based design example**

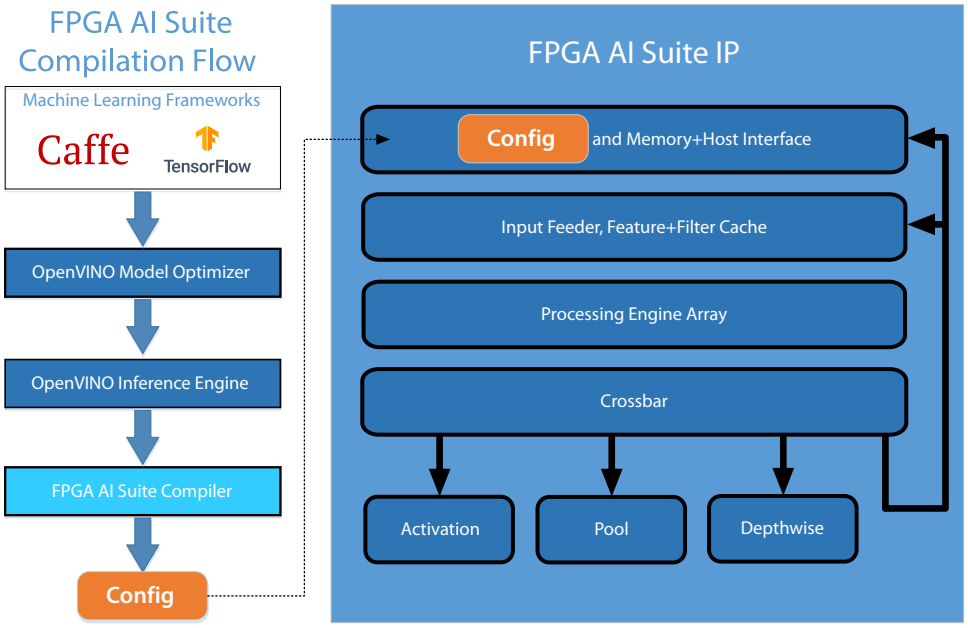
  This design example is based on using an Arria 10 SoC FPGA.

  This example highlights how to use the FPGA AI Suite IP in a fully embedded FPGA design, with direct camera input and a video IP pipeline connected to the FPGA AI Suite IP.

  This example design is available separately and is not provided with the FPGA AI Suite.

The following diagram illustrates the connection between two primary components of the FPGA AI Suite – the compiler and the IP:

Figure 1. Connection Between FPGA AI Suite Compiler and FPGA AI Suite IP

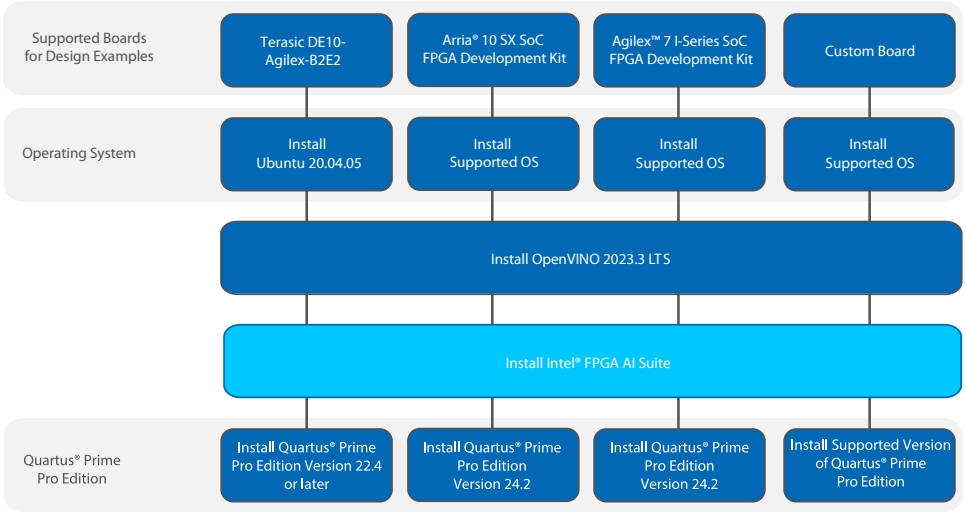


3. FPGA AI Suite Installation Overview

The FPGA AI Suite compiler, IP, and associated tools can be installed separately from the design examples. The FPGA AI Suite PCIe-based design example adds hardware prerequisites and additional software prerequisites, and has additional installation steps.

The FPGA AI Suite has different software dependencies depending on the design example. The following diagram describes an installation flow for the different design example hardware.

Figure 2. Installation Flows for FPGA AI Suite



If you want to install only the FPGA AI Suite compiler, IP, and associated tools, go to Installing the FPGA AI Suite Compiler and IP Generation Tools on page 9.

Quartus Prime Pro Edition is required only if you want to build the example designs or if you want to integrate the FPGA AI Suite IP into your designs.

If you want to run the PCIe-based design example, complete these steps:

1. Installing the FPGA AI Suite Compiler and IP Generation Tools on page 9
2. Installing the FPGA AI Suite PCIe-Based Design Example Prerequisites on page 20

If you want to run the SoC design example, complete the steps in Installing the FPGA AI Suite Compiler and IP Generation Tools on page 9. After complete these steps, follow the instructions in *FPGA AI Suite SoC Design Example User Guide*.

3.1. FPGA AI Suite Directory Structure

After you install the FPGA AI Suite you have the following directories in your <*ai_suite_root*>, which is the value assigned to the COREDLA_ROOT environment variable:

Table 2. FPGA AI Suite Directory Structure

Directory	Description
`$COREDLA_ROOT/`	Base directory
`$COREDLA_ROOT/bin`	Tool executables (`dla_compiler` command)
`$COREDLA_ROOT/lib`	Libraries (compiler API, OpenVINO plugins, etc)
`$COREDLA_ROOT/inc`	Public API header files
`$COREDLA_ROOT/util`	Additional runtime modules
`$COREDLA_ROOT/thirdparty`	Third-party packages used by the runtime
`$COREDLA_ROOT/area_model`	Data files
`$COREDLA_ROOT/demo`	Sample image files and bitstreams
`$COREDLA_ROOT/example_architectures`	Architecture files, each which defines a specific IP parameterization
`$COREDLA_ROOT/example_ip_cores`	Generated IP cores corresponding to the example architectures
`$COREDLA_ROOT/fpga`	Source files used by the IP generator
`$COREDLA_ROOT/licenses`	License information for FPGA AI Suite and included libraries
`$COREDLA_ROOT/platform`	BSP source directory
`$COREDLA_ROOT/runtime`	Directory with OpenVINO demos and low-level runtime for FPGA interfacing
`$COREDLA_ROOT/dla_plugin`	Directory for OpenVINO heterogeneous compilation and inference integration with the FPGA

4. Installing the FPGA AI Suite Compiler and IP Generation Tools

This section describes how to install the FPGA AI Suite compiler (`dla_compiler` command) and IP Generation Tools.

If you want to use the PCIe-based design example, ensure that you have met the prerequisites outlined in Installing the FPGA AI Suite PCIe-Based Design Example Prerequisites on page 20 before you follow the instructions in this section.

To install the FPGA AI Suite compiler and IP generation tools:

Tip: The steps here outline the installation process. You can follow the remaining parts of this section in sequence without returning to this topic to complete your installation.

1. Review Supported FPGA Families on page 9 to ensure that your target FPGA device is supported.
2. Ensure that your system meets the operating system requirements as outlined in Operating System Prerequisites on page 10.
3. Install FPGA AI Suite.

   Tip: If you want to install FPGA AI Suite in a location other than the default directory, follow the instructions in Installing the FPGA AI Suite Into an Alternative Location on page 12.
4. Install OpenVINO Toolkit.
5. Install Quartus Prime Pro Edition.

   Important: Quartus Prime Pro Edition is required for the FPGA AI Suite design examples. Otherwise, Quartus Prime Pro Edition is an optional component.
6. Set the required environment variables.
7. If you are using Red Hat* Enterprise Linux* 8.7, install Intel oneAPI Threading Building Blocks.
8. Finalize your installation by verifying that FPGA AI Suite commands run correctly.

4.1. Supported FPGA Families

The FPGA AI Suite supports the following FPGA device families:

- Agilex 5
- Agilex 7
- Arria 10
- Cyclone® 10 GX
- Stratix® 10

The FPGA AI Suite is compatible with Quartus Prime Pro Edition Versions 19.2 - 24.3.

The FPGA AI Suite design examples (PCIe and SoC) require specific versions of Quartus Prime Pro Edition.

4.2. Operating System Prerequisites

The FPGA AI Suite supports the following operating systems:

- Red Hat Enterprise Linux 8.7
- Ubuntu* 20.04 LTS

  Ubuntu 20.04 LTS is supported natively and in a Windows Subsystem for Linux 2 (WSL 2) environment.

  In a WSL 2 environment, only the FPGA AI Suite compiler and IP generation tools are supported.
- Ubuntu 22.04 LTS

  Ubuntu 22.04 LTS is supported only natively. It is not supported in a WSL 2 environment. For support in a WSL 2 environment, use Ubuntu 20.04 LTS.

Important: The PCIe-based design examples require specific versions operating systems. This is required only to use the specific PCIe example design listed. For the operating system requirements for the PCIe-based design examples, refer to the following table:

Table 3. PCIe-Based Design Example Operating System Requirements

Design Example	Operating System Requirement
PCIe-based design example for Agilex 7 devices	Ubuntu 20.04.5

For details about the operating system requirements for the PCIe-based design example, refer to PCIe-based Design Example Operating System Prerequisites on page 20.

4.3. Installing the FPGA AI Suite with System Package Management Tools

The FPGA AI Suite is provided as a package that you can install using your system package management tools. The default location for the installation is `/opt/intel`. If you want to install FPGA AI Suite in a different location, follow the instructions in Installing the FPGA AI Suite Into an Alternative Location on page 12.

To install the FPGA AI Suite with system package management tools:

1. Download the FPGA AI Suite 2024.3 installation package for your operating system from the following Intel FPGA Software Download Center URL:

   https://plan.seek.intel.com/Intel_FPGA_AI_Download_Enroll

   WSL 2 Users: Use a web browser in your Windows environment to download the file. Ensure that you download **FPGA AI Suite for Ubuntu 20**.

   Ubuntu 22.04 Users:* Download the package file to `/tmp`.
2. Install FPGA AI Suite according to the instruction for your operating system:

- **Red Hat Enterprise Linux**: Follow the instructions in Installing FPGA AI Suite on Red Hat Enterprise Linux Operating Systems on page 11.
- **Ubuntu Linux**: Follow the instructions in Installing FPGA AI Suite on Ubuntu Operating Systems on page 11.
- **Window Subsystem for Linux 2 (WSL 2)**: Follow the instructions in Installing FPGA AI Suite in a Windows Subsystem for Linux 2 Environment on page 11.

Installing FPGA AI Suite on Red Hat Enterprise Linux Operating Systems

For Red Hat Enterprise Linux, the FPGA AI Suite is provided as an RPM package that you can install with the `dnf` command. The `dnf` command automatically installs several dependencies.

Some of the dependencies are in the CodeReady Linux Builder repository. Enable this repository with the following command:

```
sudo subscription-manager repos \
--enable codeready-builder-for-rhel-8-x86_64-rpms
```

To install the package into the default location in `/opt/intel/`, locate the `.rpm` file and run the following command:

```
sudo dnf install <full path to fpga-ai-suite.rpm>
```

Installing FPGA AI Suite on Ubuntu Operating Systems

For Ubuntu operating systems (including those running in a WSL 2 environment), the FPGA AI Suite is provided as a package that you can install with the `apt` command. The `apt` command automatically installs several dependencies.

To install the package into the default location in `/opt/intel/`, locate the `.deb` file and run the following commands:

```
sudo apt-get install --reinstall libappstream4
sudo apt update
sudo apt install <full path to fpga-ai-suite.deb>
```

Installing FPGA AI Suite in a Windows Subsystem for Linux 2 Environment

The FPGA AI Suite supports Ubuntu 20.04 LTS running in a Windows Subsystem for Linux 2 (WSL 2) environment. Ensure that your WSL 2 environment has Ubuntu 20.04 LTS installed and that you have internet access from your WSL 2 environment.

To install FPGA AI Suite in your WSL 2 environment:

1. Move the downloaded `.deb` file from its downloaded location in your Windows file system to a location in your Linux file system. You can drag-and-drop the file using Windows File Explorer. You can find the Linux file system the **Linux** entry in the File Explorer navigation pane.
2. Start a WSL 2 terminal session.
3. In the WSL2 terminal session, install the package into the default location in `/opt/intel/` by locating the `.deb` file in your Linux file system and running the following commands:

```
sudo apt-get install --reinstall libappstream4
sudo apt update
sudo apt install <full path to fpga-ai-suite.deb>
```

Next Step

After installing FPGA AI Suite, continue your installation with the steps in Installing OpenVINO Toolkit on page 12.

4.3.1. Installing the FPGA AI Suite Into an Alternative Location

If you prefer to install the FPGA AI Suite into an alternative location, you can use the following commands:

- Ubuntu 20.04:

```
sudo apt update
sudo apt install libnuma-dev cmake git curl python3.8 python3.8-venv \
                 graphviz unzip lsb libboost-all-dev
dpkg-deb -R <full path to fpga-ai-suite.deb> \
           <full path to alternative location>
```

- Ubuntu 22.04:

```
sudo apt update
sudo apt install libnuma-dev cmake git curl graphviz unzip lsb libboost-all-
dev
dpkg-deb -R <full path to fpga-ai-suite.deb> \
           <full path to alternative location>
```

- Red Hat Enterprise Linux operating systems:

```
sudo dnf install numactl-devel cmake git curl python38 graphviz \
                 gcc gcc-c++ redhat-lsb tbb-devel gflags-devel boost-devel
rpm -i --prefix=<full path to alternative location> \
               <full path to fpga-ai-suite.rpm>
```

4.4. Installing OpenVINO Toolkit

The FPGA AI Suite requires the OpenVINO Toolkit 2023.3 for Linux.

To prepare OpenVINO Toolkit for FPGA AI Suite:

1. Install the required OpenVINO Toolkit components.
2. Configure your OpenVINO environment.

   You must configure the OpenVINO environment each time you want to use the OpenVINO Toolkit. For convenience, you can add the configuration to your `.bashrc` file.

The installer might issue an error message that indicates the OpenCL is not installed and is needed for GPU operation of OpenVINO. You can safely ignore this error message. The FPGA AI Suite does not require OpenVINO GPU support.

Recommended Python Version

OpenVINO Toolkit supports Python versions 3.7-3.10. The recommended version for your operating system is as follows:

- **Red Hat Enterprise Linux 8**: Python 3.8.
- **Ubuntu 20**: Python 3.8.
- **Ubuntu 22**: Python 3.10.

Installing Required OpenVINO Toolkit Components

FPGA AI Suite requires you to install the OpenVINO Runtime and OpenVINO Development Tools separately.

If you plan run the FPGA AI Suite in a WSL 2 environment, ensure that you complete these instructions from an Ubuntu command prompt in your WSL 2 environment and not from a Windows command line.

The location where you install the OpenVINO Runtime is often referred to as `<openvino_installdir>` in FPGA AI Suite documentation. Typically, this location is `/opt/intel` or `/home/<user>/intel`.

To install the required OpenVINO Toolkit components:

1. [**RHEL 8 and Ubuntu 20 only**] Create a Python 3.8 virtual environment and start it:

   Important: This step is not required on Ubuntu 22.04. If you use Ubuntu 22.04, skip to the next step where you install the OpenVINO 2023.3 Runtime.

   a. Install Python 3.8:

      Red Hat Enterprise Linux*:

      ```
      sudo yum install python3.8
      ```

      Ubuntu Linux 20:

      ```
      sudo apt update
      sudo apt install python3.8 python3.8-venv -y
      ```

   b. Verify your Python installation with the following command:

      ```
      python3.8 -V
      ```

      This command should return output indicating that you are using Python 3.8. For example:

      ```
      Python 3.8.16
      ```

   c. Create a new directory and switch to it:

      ```
      mkdir ~/build-openvino-dev && cd ~/build-openvino-dev
      ```

d. Create a Python 3.8 virtual environment:

```
python3.8 -m venv openvino_env

source openvino_env/bin/activate
```

e. Confirm that you are in the new Python virtual environment:

```
python -V
```

This command should return output indicating that you are using Python 3.8. For example:

```
Python 3.8.5
```

2. Install the OpenVINO 2023.3 Runtime:

a. If you do not have an `opt/intel` folder, create one as follows:

```
sudo mkdir /opt/intel
```

b. If you do not have a `~/Downloads` folder, create one as follows:

```
mkdir ~/Downloads
```

c. Go to your `Downloads` folder:

```
cd ~/Downloads
```

d. Download the OpenVINO 2023.3 Runtime archive file for your system, extract the files, and move them to the `/opt/intel` folder as follows:

- **Red Hat Enterprise Linux 8**:

```
curl -L https://storage.openvinotoolkit.org/\
repositories/openvino/packages/2023.3/linux/\
l_openvino_toolkit_rhel8_2023.3.0.13775.ceeafaf64f3_x86_64.tgz \
--output openvino_2023.3.0.tgz

tar -xf openvino_2023.3.0.tgz

sudo mv \
l_openvino_toolkit_rhel8_2023.3.0.13775.ceeafaf64f3_x86_64 \
/opt/intel/openvino_2023.3.0
```

- **Ubuntu 20**:

```
curl -L https://storage.openvinotoolkit.org/\
repositories/openvino/packages/2023.3/linux/\
l_openvino_toolkit_ubuntu20_2023.3.0.13775.ceeafaf64f3_x86_64.tgz \
--output openvino_2023.3.0.tgz

tar -xf openvino_2023.3.0.tgz

sudo mv \
l_openvino_toolkit_ubuntu20_2023.3.0.13775.ceeafaf64f3_x86_64 \
/opt/intel/openvino_2023.3.0
```

- **Ubuntu 22**:

```
curl -L https://storage.openvinotoolkit.org/\
repositories/openvino/packages/2023.3/linux/\
l_openvino_toolkit_ubuntu22_2023.3.0.13775.ceeafaf64f3_x86_64.tgz \
--output openvino_2023.3.0.tgz

tar -xf openvino_2023.3.0.tgz
```

```
sudo mv \
l_openvino_toolkit_ubuntu22_2023.3.0.13775.ceeafaf64f3_x86_64 \
/opt/intel/openvino_2023.3.0
```

e. Install system dependencies required by the OpenVINO Runtime by running the provided script:

- **Red Hat Enterprise Linux**:

```
cd /opt/intel/openvino_2023.3.0/install_dependencies/

sed -i -e 's|http://mirror.centos.org/centos/8-stream/AppStream| \
https://repo.almalinux.org/almalinux/8/AppStream|' \
./install_openvino_dependencies.sh

sed -i -e 's/yum install "\$iopt" "\${pkgs\[@\]}"/yum \
install \${iopt:+\$iopt} "\${pkgs[@]}"/' \
./install_openvino_dependencies.sh
sudo -E ./install_openvino_dependencies.sh
```

- **Ubuntu Linux**:

```
cd /opt/intel/openvino_2023.3.0/install_dependencies/

sudo -E ./install_openvino_dependencies.sh
```

f. Create a symbolic link:

```
cd /opt/intel

sudo ln -s openvino_2023.3.0 openvino_2023
```

Important: If you have already installed a previous release of OpenVINO 2023, a symbolic link to the `openvino_2023` folder might already exist. Unlink the previous link with the `sudo unlink openvino_2023` command, and then re-run the commands in this step.

g. Configure the OpenVINO Runtime environment variables with the following command:

```
source /opt/intel/openvino_2023/setupvars.sh
```

3. Install the OpenVINO Development Tools:

a. Upgrade the Python `pip` command:

```
python -m pip install --upgrade pip
```

b. Install the `openvino-dev` package:

```
pip install "openvino-dev[<frameworks>]==2023.3.0"
```

Where *<frameworks>* is a comma-separated list of the deep learning frameworks that you want your OpenVINO development tools to support. The following values are supported: `caffe`, `kaldi`, `mxnet`, `onnx`, `pytorch`, `tensorflow`, `tensorflow2`.

For example, to install the OpenVINO development tools to support the Caffe, PyTorch*, and TensorFlow* frameworks, run the following command:

```
pip install "openvino-dev[caffe, pytorch, tensorflow]==2023.3.0"
```

Tip: This command might produce the following error message:

```
ERROR: tensorflow 2.13.1 has requirement numpy<=1.24.3,>=1.22, but
you'll have numpy 1.24.4 which is incompatible
```

Thjs error message is pip-specific indicating that some packages in the local Python environment do not conform to OpenVINO development tools requirements. You can safely ignore this warning message. The `openvino-dev` package is still successfully installed.

4. Install OpenCV as follows:
   - On Red Hat Enterprise Linux systems, OpenCV is available from the CodeReady Linux Builder repository. Enable the repository and install OpenCV with the following commands:

     ```
     sudo subscription-manager repos \
     --enable codeready-builder-for-rhel-8-x86_64-rpms

     sudo yum update

     sudo yum install opencv opencv-devel

     pip3 install opencv-python
     ```

   - On Ubuntu Linux systems, OpenCV is available in the default repositories. Install OpenCV with the following commands:

     ```
     sudo apt update

     sudo apt install libopencv-dev python3-opencv
     ```

   If you need a version of OpenCV optimized for your hardware, you might need to build OpenCV from source. For instructions, refer to "Approach #2: Build OpenCV against specific version of OpenVINO" section of *OpenCV usage with OpenVINO*.

Configuring the OpenVINO Environment

You must configure the OpenVINO environment each time you want to use the OpenVINO Toolkit. For convenience, you can add the configuration to your `.bashrc` file.

To configure the OpenVINO environment, start a Linux terminal session and run the following commands:

```
source ~/build-openvino-dev/openvino_env/bin/activate

unset python_version

source <openvino_installdir>/openvino_2023/setupvars.sh
```

You can add these comments to your `.bashrc` file to configure the OpenVINO environment for you automatically when you log on.

Next Step

After installing OpenVINO toolkit, continue your installation with the steps in Installing Quartus Prime Pro Edition Software on page 17.

Related Information

Installing Intel Distribution of OpenVINO toolkit

4.5. Installing Quartus Prime Pro Edition Software

The FPGA AI Suite is compatible with Quartus Prime Pro Edition for Linux Versions 19.2 - 24.3. FPGA AI Suite does not support Quartus Prime Pro Edition for Windows.

WSL 2 Restriction: In a Windows Subsystem for Linux 2 (WSL 2) environment, you must install Quartus Prime Pro Edition for Linux in your WSL 2 environment. You cannot use a Quartus Prime Pro Edition for Windows installation with FPGA AI Suite in the WSL 2 environment on the same system.

Tip: When designing a solution with the FPGA AI Suite, use the latest supported version of Quartus Prime Pro Edition.

Ensure that the `QUARTUS_ROOTDIR` environment variable is set correctly. For example, you can add the following to your `~/.bashrc` file to set the target directory appropriately:

```
export QUARTUS_ROOTDIR="<quartus_install_path>/quartus/"
```

Important: The FPGA boards required for the FPGA AI Suite design examples require specific versions of Quartus Prime Pro Edition. This requirement applies only to the design examples and does not apply to the FPGA AI Suite IP and compiler. For the Quartus Prime Pro Edition requirements for the design examples, refer to the following table:

Table 4. Quartus Prime Pro Edition Requirements for FPGA AI Suite Design Example Boards

Design Example Board	Quartus Prime Pro Edition Requirement
Agilex 5 FPGA E-Series 065B Premium Development Kit (DK-A5E065BB32AES1)	Quartus Prime Pro Edition Version 24.3
Agilex 7 FPGA I-Series Development Kit ES2 (DK-DEV-AGI027RBES) (PCIe-Attached)	Quartus Prime Pro Edition Version 24.1
Agilex 7 FPGA I-Series Development Kit ES2 (DK-DEV-AGI027RBES) (Hostless)	Quartus Prime Pro Edition Version 24.3
Agilex 7 FPGA I-Series Transceiver-SoC Development Kit (DK-SI-AGI027FC)	Quartus Prime Pro Edition Version 24.3 (Ashling* RiscFree* IDE for Intel FPGAs is also required)
Arria 10 SX SoC FPGA Development Kit (DK-SOC-10AS066S)	Quartus Prime Pro Edition Version 24.3 (Ashling RiscFree IDE for Intel FPGAs is also required)
Intel FPGA SmartNIC N6001-PL Platform (without Ethernet controller) (PCIe-Attached)	Quartus Prime Pro Edition Version 24.1
Intel FPGA SmartNIC N6001-PL Platform (without Ethernet controller) (Hostless)	Quartus Prime Pro Edition Version 24.3
Terasic DE10-Agilex Development Board (DE10-Agilex-B2E2)	Quartus Prime Pro Edition Version 22.4 or later. This document assumes that Version 24.3 is used.

Next Step

After installing Quartus Prime Pro Edition, continue your installation with Setting Required Environment Variables on page 18.

4.6. Setting Required Environment Variables

Set the OpenVINO toolkit and FPGA AI Suite environment variables by running the following commands from a bash-style shell (do not use `/bin/dash` or `/bin/tcsh`) or by adding the commands to your `.bashrc` file. If you did not install OpenVINO toolkit into the default location, adjust the second command appropriately.

```
source /opt/intel/openvino_2023/setupvars.sh
source /opt/intel/fpga_ai_suite_2024.3/dla/setupvars.sh
```

Tip: If you have multiple versions of FPGA AI Suite installed on your system, you can use `source` commands to switch between them. To switch, source the environment variable setup scripts for the version of FPGA AI Suite that you want to run. You must ensure that you also have the required OpenVINO toolkit version available.

For example, if you want to run FPGA AI Suite 2023.3, run the following commands:

```
source /opt/intel/openvino_2022/setupvars.sh
source /opt/intel/fpga_ai_suite_2023.3/dla/setupvars.sh
```

In some older versions of FPGA AI Suite. the environment variable setting script is called `init_env.sh` and might be in a slightly different path (`dla/bin/init_env.sh`).

Next Step

After setting the required environment variables, if you are using Red Hat Enterprise Linux 8.7, continue your installation with the steps in Installing Intel Threading Building Blocks (TBB) on page 18. Otherwise, complete your installation with the steps in Finalizing Your FPGA AI Suite Installation on page 19.

4.7. Installing Intel Threading Building Blocks (TBB)

FPGA AI Suite requires some functions provided by Intel Threading Building Blocks (TBB).

For Ubuntu 20 and Ubuntu 22 systems, Intel TBB is installed as part of the FPGA AI Suite installation package.

For all other operating systems, FPGA AI Suite provides a script to download and install Intel TBB 2020 Update 3 (v2020.3) on your system.

To run the script, issue the following command:

```
$COREDLA_ROOT/bin/build_tbb.sh
source $INTEL_OPENVINO_DIR/setupvars.sh
```

For more information about Intel TBB, refer to the oneAPI open source oneTBB GitHub repository.

Next Step

After installing Intel TBB, complete your installation with the steps in Finalizing Your FPGA AI Suite Installation on page 19.

4.8. Finalizing Your FPGA AI Suite Installation

To finalize your FPGA AI Suite Installation, verify your installation by running a `dla_compiler` command.

You can verify your installation by running the following command:

```
dla_compiler --fanalyze-area --march $COREDLA_ROOT/*/AGX7_Generic.arch
```

The `dla_compiler` command prepares deep learning models for consumption by the FPGA AI Suite IP, produces performance estimates, and produces area (or FPGA resource usage) estimates. This example command produces an area estimate for the IP configuration stored in the file `AGX7_Generic.arch`.

If this command runs successfully, your FPGA AI Suite installation was successful.

5. Installing the FPGA AI Suite PCIe-Based Design Example Prerequisites

This section describes the steps to install the prerequisites for the following design example:

- PCIe-based design example for Agilex 7 devices

If you want to use the PCIe-based design examples, you must meet these prerequisites before you install the FPGA AI Suite.

If you do not want to use the PCIe-based design examples, skip this section and go to Installing the FPGA AI Suite Compiler and IP Generation Tools on page 9.

To learn more about the PCIe-based design example, refer to *FPGA AI Suite PCIe-Based Design Example User Guide*.

5.1. PCIe-based Design Example Hardware Prerequisites

The minimum hardware requirement for the design examples are as follows:

- Desktop computer or server with at least 1 free PCIe slot
- Minimum RAM: 48GB
- Minimum storage: 100GB SSD

Supported Boards

The PCIe-based design examples requires the following board:

- Terasic DE10-Agilex Development Board

  This board is required by the PCIe-based design example for Agilex 7 devices.

5.2. PCIe-based Design Example Operating System Prerequisites

The operating system prerequisites for the PCIe-based design example depend on the target FPGA device.

PCIe-based Design Example for Agilex 7 Operating System Prerequisites

The PCIe-based example design for Agilex 7 devices requires Ubuntu 20.04 LTS.

For installation instructions, refer to https://ubuntu.com/tutorials/install-ubuntu-desktop.

5.3. PCIe-based Design Example Software Prerequisites

The PCIe-based design examples require the following software:

- CMake 3.10.2 or higher
- `libnuma-dev` package

The `apt` tool attempts to install these prerequisites automatically when you install the FPGA AI Suite following the instructions in Installing the FPGA AI Suite with System Package Management Tools on page 10.

Depending on the target board for the PCIe-based design example, you must meet additional software prerequisites:

- Additional Software Prerequisites for the PCIe-based Design Example for Agilex 7 Devices on page 21

5.3.1. Additional Software Prerequisites for the PCIe-based Design Example for Agilex 7 Devices

The kernel driver for the Terasic BSP must be installed according to instructions provided by Terasic. Follow the instructions that follow, or contact your Terasic representative for additional details.

Important: If you follow the Terasic instructions, the Terasic instructions (on page 35) say to override the $LD_LIBRARY_PATH environment variable. Do not override the environment variable. Append paths to the $LD_LIBRARY_PATH environment variable instead.

A copy of the verified Terasic kernel driver is distributed as a separate file: `DE10_Agilex_revC_linux_BSP_21.2.zip`. Obtain this file from your Terasic representative.

Ubuntu Kernel Prerequisites

The kernel driver is tested on the Ubuntu 20.04.5 LTS distribution with kernel version 5.15.0. You can confirm the kernel version of your Ubuntu system with the following command:

```
uname -r
```

Terasic Kernel Driver Software Prerequisites

To install the Terasic kernel driver, install the following software dependencies:

- Quartus Prime Pro Edition Programmer and Tools Version 22.4 or later.

  The instructions in this document assume that Quartus Prime Pro Edition Version 24.3 is used.

  Download the software from the **Additional Software** tab at the following URL: https://www.intel.com/content/www/us/en/software-kit/current/657471.html.
- Intel FPGA RTE for OpenCL™ Version 21.2.

  Download the software from **RTE** tab at the following URL: https://www.intel.com/content/www/us/en/software-kit/670235/intel-fpga-sdk-for-opencl-pro-edition-software-version-21-2.html.
- Intel FPGA Download Cable II (formerly USB-Blaster II) Driver for Linux.

  Configuration steps are available the following URL: https://www.intel.com/content/www/us/en/support/programmable/support-resources/download/dri-usb-b-lnx.html

After you have installed the Quartus Programmer, set up the required environment variable using the following command:

```
export PATH="$PATH:"<path to Quartus Programmer installation folder>/qprogrammer/quartus/bin
```

Your system must recognize the Intel FPGA Download Cable II to install the Terasic kernel driver and program your board.

To verify the connection of the Intel FPGA Download Cable II, issue the `jtagconfig` command. You should see the DE10-Agilex device listed. An example of the expected output is as follows:

```
$ jtagconfig
1) DE10-Agilex [1-11.1]
C24A0DD   AGFB014R24(A|B)
```

The list index number ("1" in the example) is also the cable number, which can be used to reprogram the board in Terasic Driver Installation on page 22.

Terasic Driver Installation

After you meet the Terasic kernel driver software prerequisites, complete the following steps:

1. Set up the environment with the following command:

   ```
   source <full path to opencl rte installation>/init_opencl.sh
   ```

2. Unzip the board support package in the current directory and set up the following environment variables (The value $PWD in the commands that follow must correspond to the directory in which the BSP packages was unzipped):

   ```
   tmp1=$PWD/DE10_Agilex_revC_linux_BSP_21.2/de10_agilex
   export AOCL_BOARD_PACKAGE_ROOT=$tmp1
   tmp1=$ALTERAOCLSDKROOT/host/linux64/lib
   tmp2=$AOCL_BOARD_PACKAGE_ROOT/linux64/lib
   export LD_LIBRARY_PATH=$tmp1:$tmp2:$LD_LIBRARY_PATH
   ```

3. Install the driver with the following command:

```
aocl install
```

4. Rescan the PCIe bus or reboot the system to program the board. To verify that the driver bonds to the card, program the testing bitstream to the board with the following command:

```
quartus_pgm -c <cable_number> -m jtag \
  -o "p;$AOCL_BOARD_PACKAGE_ROOT/bringup/B2E2_8GBx4/top.sof"
```

5. Perform a soft reboot of your host system for the system to recognize the Terasic DE10-Agilex Development Board:

```
reboot
```

   You can verify that the board is recognized by the host system with the following command:

```
lspci | grep Altera
```

   If you do not see the board information when scanning the PCIe bus, try rebooting your system or reinstalling the board. In some cases, you might need to move the card to a different PCIe slot.

For more details and information about the Terasic kernel driver, refer to the Terasic OpenCL BSP Guide, *DE10-Agilex_OpenCL_21.2.pdf*, in the kernel driver package.

Ubuntu Kernel Update Requirements

If you update your Ubuntu kernel, you must reinstall the Terasic kernel driver.

Bitstream Generation Requirements

If you plan to use only the precompiled bitstreams supplied with the FPGA AI Suite, then no further steps are required.

If you plan to generate bitstreams corresponding to custom FPGA AI Suite IP architectures for the PCIe-based example design for Agilex 7 devices, then you must install Quartus Prime Pro Edition Version 24.3 and setup your PATH environment variable.

For Quartus Prime Pro Edition Version 24.3 prerequisites and installation instructions, refer to *Intel FPGA Software Installation and Licensing*.

Download the Quartus Prime Pro Edition software from the following URL: https://www.intel.com/content/www/us/en/software-kit/current/657471.html

Bitstream generation works with Quartus Prime Pro Edition Version 24.3, even though the Terasic driver requires Intel FPGA RTE for OpenCL Version 21.2.

You must also add the Platform Designer command (`qsys`) to your PATH environment variable. Add the following commands to your `.bashrc` file, and adjust the paths depending on your installation locations, if needed:

```
export PATH="$PATH:"~/intelFPGA_pro/quartus_24.3/qsys/bin/
export PATH="$PATH:"~/intelFPGA_pro/quartus_24.3/quartus/bin/
```

Reboot Requirements

Any time that you reboot your system after powering down, you must reprogram the board with the following command:

```
quartus_pgm -c <cable_number> -m jtag \
     –o "p;$AOCL_BOARD_PACKAGE_ROOT/bringup/B2E2_8GBx4/top.sof"
```

After you successfully install the board, continue by installing the compiler and IP generation tool as described in Installing the FPGA AI Suite Compiler and IP Generation Tools on page 9.

6. FPGA AI Suite Quick Start Tutorial

The examples in this section assume that you have installed the FPGA AI Suite according to the instructions in Installing the FPGA AI Suite Compiler and IP Generation Tools on page 9.

Ensure that you the required environment variables set. For details, refer to Setting Required Environment Variables on page 18.

A quick command to check that the FPGA AI Suite installation was successful is as follows:

```
dla_compiler --fanalyze-area --march $COREDLA_ROOT/*/AGX7_Generic.arch
```

6.1. Creating a Working Directory

This tutorial requires a writeable working directory that contains a copy of the example runtime and the demonstration files.

If you want to use `~/coredla_work/` as your working directory, run the following commands:

```
mkdir ~/coredla_work
cd ~/coredla_work
source dla_init_local_directory.sh
```

The `dla_init_local_directory.sh` script does the following tasks:

- Sets `COREDLA_WORK` environment variable to point at working directory.
- Creates a locally writeable copy of the `runtime` directory.
- Creates a locally writeable copy of the `demo` directory.

To re-enter this work directory after opening a new shell, `source` the OpenVINO `setupvar.sh` script and the FPGA AI Suite `init_env.sh` script, as described in Setting Required Environment Variables on page 18. Then, run the following commands:

```
cd ~/coredla_work
source ./coredla_work.sh
```

6.2. Preparing OpenVINO Model Zoo

These instructions assume that you have a copy of OpenVINO Model Zoo 2023.3 in your `$COREDLA_WORK/demo/open_model_zoo/` directory.

To download a copy of Model Zoo, run the following commands:

```
cd $COREDLA_WORK/demo
git clone https://github.com/openvinotoolkit/open_model_zoo.git
cd open_model_zoo
git checkout 2023.3.0
```

6.3. Preparing a Model

A model must be converted from a framework (such as TensorFlow, Caffe, or Pytorch) into a pair of `.bin` and `.xml` files before the FPGA AI Suite compiler (`dla_compiler` command) can ingest the model.

The following commands download the ResNet-50 TensorFlow model and run Model Optimizer:

```
source ~/build-openvino-dev/openvino_env/bin/activate
omz_downloader --name resnet-50-tf \
    --output_dir $COREDLA_WORK/demo/models/
omz_converter --name resnet-50-tf \
    --download_dir $COREDLA_WORK/demo/models/ \
    --output_dir $COREDLA_WORK/demo/models/
```

The `omz_downloader` command downloads the trained model to `$COREDLA_WORK/demo/models` folder. The `omz_converter` command runs model optimizer that converts the trained model into intermediate representation `.bin` and `.xml` files in the `$COREDLA_WORK/demo/models/public/resnet-50-tf/FP32/` directory.

The directory `$COREDLA_WORK/demo/open_model_zoo/models/public/resnet-50-tf/` contains two useful files that do not appear in the `$COREDLA_ROOT/demo/models/` directory tree:

- The `README.md` file describes background information about the model.
- The `model.yml` file shows the detailed command-line information given to Model Optimizer (`mo.py`) when it converts the model to a pair of `.bin` and `.xml` files

For a list OpenVINO Model Zoo models that the FPGA AI Suite supports, refer to the *FPGA AI Suite IP Reference Manual*.

6.4. Running the Graph Compiler

The FPGA AI Suite compiler (`dla_compiler` tool) can estimate the performance of a graph and IP architecture combination. The tool can also produce a compiled network that the runtime uses in AOT (ahead-of-time) mode to control inference on the IP.

The following commands generate an AOT file for Agilex 7 devices:

```
cd $COREDLA_WORK/demo/models/public/resnet-50-tf/FP32
dla_compiler \
            --march $COREDLA_ROOT/example_architectures/AGX7_Performance.arch \
            --network-file ./resnet-50-tf.xml \
            --foutput-format=open_vino_hetero \
            --o $COREDLA_WORK/demo/RN50_Performance_b1.bin \
            --batch-size=1 \
            --fanalyze-performance
```

These command specify the following input files:

- The `.arch` file specifies the configuration of the IP architecture (configuration parameters such as large, small, and which activations to use).
- The `.xml` file (along with the `.bin`) describes the network.
- The `.bin` file stores certain parameters along with the model weights.

The FPGA AI Suite compiler uses these files to analyze the performance estimate.

The output file `RN50_Performance_b1.bin` is the compiled network and contains the instructions and graph weights necessary to control the FPGA AI Suite IP on the FPGA device. This output file is specific to the `.arch` file and the graph `.xml` file.

The compiler dot graph shows which layers run on the FPGA and which layers run on the CPU. Using the Graphviz package, the dot file format can be converted into an SVG graphics file named `ResNet-50.svg` that shows each layer. You can view SVG graphics in most web browsers. Use the following command to convert the dot graph file into an SVG file:

```
dot -Tsvg hetero_subgraphs_resnet-50-tf.dot -o ResNet-50.svg
```

6.5. Preparing an Image Set

This section describes how to prepare an image set for classification graphs that requires 224x224 input and have been trained on the ImageNet classifications. For the `yolo-v3-tf` and `yolo-v3-tiny-tf` graphs, the instructions in the *FPGA AI Suite PCIe Example Design User Guide* describe how to prepare an image set, ground truth data, and how to call the `dla_benchmark` application.

The FPGA AI Suite includes sample validation images in the `$COREDLA_ROOT/demo/sample_images/` folder.

The contents are as follows:

`$COREDLA_WORK/demo/sample_images/`	Sample image directory. A file naming convention is used to provide a sort order for the images.
`$COREDLA_WORK/demo/sample_images/ground_truth.txt`	Ground truth file.
`$COREDLA_WORK/demo/sample_images/TF_ground_truth.txt`	Ground truth file suitable for graphs from the TensorFlow framework.

Image classification graphs trained in the TensorFlow framework require ground truth files that account for a difference in how TensorFlow numbers the output categories (an off-by-one difference). The sample image set includes two ground truth files to account for this.

Due to the small number of images in the sample image set, the `dla_benchmark` commands in this section for the sample image set `-niter=8`. This setting limits the number of inferences executed by the FPGA AI Suite IP.

To help performance benchmarking, the `dla_benchmark` demo generates random image data if it requires more input images than are available in the image set.

Optionally, you can run the demo with the ILSVRC2012 image set. To use this image set, manually download it as follows:

1. The image set is available from https://image-net.org/download-images.php.
2. Download the **2012** "Validation images" and the **2017** development kit.
3. Convert the 2012 images to `.bmp` format, resizing the smallest dimension to 256 while preserving aspect ratio and center crop to 224x224.

The image preprocessing can be done with many different tools. One of the common Linux tools is ImageMagick. For details, refer to https://imagemagick.org/.

The following commands use ImageMagick to prepare the images:

```
mkdir imagenet_jpg
cd imagenet_jpg
tar xf /path/to/ILSVRC2012_img_val.tar
mkdir ../imagenet
echo *.JPEG | xargs mogrify \
    -path ../imagenet \
    -resize '256x256^' \
    -gravity Center \
    -crop 224x224+0+0 \
    -format bmp
```

Create the `TF_ground_truth.txt` file from the 2017 development with these commands:

```
tar zxf ILSVRC2017_devkit.tar.gz
cd ILSVRC/devkit/data
export cnt=1
cp ILSVRC2015_clsloc_validation_ground_truth.txt TF_ground_truth.txt
cat map_clsloc.txt | sort | (
    while read a; do
        orig=$(echo $a | awk '{print $2}')
        sed -i -e "s/^$orig\$/_$cnt/" TF_ground_truth.txt
        cnt=$(($cnt+1))
    done
)
sed -i -e 's/^_//' TF_ground_truth.txt
```

If you want the demo to use the ImageNet images instead of the sample images bundled with the FPGA AI Suite make the following changes to the `dla_benchmark` example commands in Performing Inference on the PCIe-Based Example Design on page 30:

1. Set `-niter=5000` instead of `-niter=8`.
2. Set `-groundtruth_loc` to specify the `TF_ground_truth.txt` created earlier.
3. Set `-i` to specify the directory containing the converted ILSVRC2012 `.bmp` images.

   The `.bmp` format is lossless and preserves as much details as possible.

6.6. Programming the FPGA Device

If you do not have an FPGA board installed, you can still run inference on the software emulation model. For instruction, skip ahead to Performing Inference Without an FPGA Board on page 36.

To complete this section of the tutorial, you must have installed the following supported design example:

- PCIe-based design example for Agilex 7 devices

  This design example requires the Terasic DE10-Agilex Development Board.

For installation instructions, see FPGA AI Suite Installation Overview on page 7.

Important: The instructions here assume that your machine has only a single board installed.

The PCIe-based design examples requires a runtime layer that must be compiled before the design is usable. The runtime for the design examples typically needs to be compiled only once. However, a new FPGA bitstream must be generated whenever any of the following parameters change:

- Number of IP instances built in the design example.
  - PCIe-based design example for Agilex 7 devices supports up to 4 instances.
- Configuration of the IP architecture (as specified by a `.arch` file)
- Quartus fitter seed (or any other "try-harder" compilation options)

Depending on your design example, follow the instructions in the following section:

- Programming the Agilex 7 Device (Terasic DE10-Agilex Development Board) on page 29

Programming the Agilex 7 Device (Terasic DE10-Agilex Development Board)

To program the DE10-Agilex-B2E2 board required by the PCIe-based design example for Agilex 7 devices:

1. Build the runtime with the following commands:

```
cd $COREDLA_WORK/runtime
rm -rf build_Release
./build_runtime.sh -target_de10_agilex
```

2. Program the bitstream onto the FPGA device with the following commands:

```
jtagdir=$COREDLA_WORK/runtime/build_Release/fpga_jtag_reprogram
bitsdir=$COREDLA_WORK/demo/bitstreams
$jtagdir/fpga_jtag_reprogram $bitsdir/AGX7_Performance.sof
curarch=$COREDLA_ROOT/example_architectures/AGX7_Performance.arch
```

The bitstream completely reprograms the board, which causes it to disconnect from the PCIe bus. The board is kept accessible with the `$COREDLA_WORK/runtime/fpga_jtag_reprogram` tool.

Precompiled bitstreams are provided in the `$COREDLA_WORK/demo/bitstreams/` directory. Some bitstreams instantiate a large IP, while others instantiate a small IP. There are also differences in which bitstreams support which neural networks. For details, refer to the *FPGA AI Suite IP Reference Manual*.

6.7. Performing Inference on the PCIe-Based Example Design

Performing Inference Using JIT Mode

The JIT (just-in-time) mode causes the `dla_benchmark` demonstration application to call the `dla_compiler` command in a just-in-time way to compile the neural net graph.

If you do not have images and ground truth files, you can skip the optional `-i` and `-groundtruth_loc` parameter entries in the command that follows. If you skip these parameters, the `dla_benchmark` demonstration application generates randomized image data.

The value for `$curarch` must match the bitstream that you programmed in Programming the FPGA Device on page 28.

```
imagedir=$COREDLA_WORK/demo/sample_images
xmldir=$COREDLA_WORK/demo/models/public/
$COREDLA_WORK/runtime/build_Release/dla_benchmark/dla_benchmark \
    -b=1 \
    -m $xmldir/resnet-50-tf/FP32/resnet-50-tf.xml \
    -d=HETERO:FPGA,CPU \
    -niter=8 \
    -plugins $COREDLA_WORK/runtime/plugins.xml \
    -arch_file $curarch \
    -api=async \
    -perf_est \
    -nireq=4 \
    -bgr \
    -i $imagedir \
    -groundtruth_loc $imagedir/TF_ground_truth.txt
```

Performing Inference Using AOT Mode

In AOT (ahead-of-time) mode, the `dla_benchmark` demonstration application uses a compiled network that was previously produced by the `dla_compiler` compiler command when you followed the steps in Running the Graph Compiler on page 26.

To use AOT mode instead of JIT mode:

1. Add the `-cm` argument to specify the name of the file containing the compiled network
2. Remove the `-perf_est` flag. The `dla_benchmark` demonstration application does not estimate performance in AOT mode.

If you omit `-i` and `-groundtruth_loc` arguments, the `dla_benchmark` demonstration application generates random input data that is useful only for performance benchmarking.

```
gt_file=$COREDLA_WORK/demo/sample_images/TF_ground_truth.txt
$COREDLA_WORK/runtime/build_Release/dla_benchmark/dla_benchmark \
    -b=1 \
    -cm $COREDLA_WORK/demo/RN50_Performance_b1.bin \
    -d=HETERO:FPGA,CPU \
    -niter=8 \
    -plugins $COREDLA_WORK/runtime/plugins.xml \
    -arch_file $curarch \
    -api=async \
    -nireq=4 \
```

```
    -bgr \
    -i $COREDLA_WORK/demo/sample_images/ \
    -groundtruth_loc $gt_file
```

The `-cm` argument points to the `.bin` file that you created in Running the Graph Compiler on page 26.

Inference APIs

The easiest way to evaluate the ability of the FPGA AI Suite to perform inference is to use the `dla_benchmark` demonstration application that is included in the example runtime and is built as part of the steps described in Programming the FPGA Device on page 28.

The example runtime also includes instructions on how to use the OpenVINO Python API to execute inference using the JIT style described in Performing Inference Using JIT Mode on page 30.

These instructions are located in `$COREDLA_WORK/runtime/python_demos/README.md`.

6.8. Building an FPGA Bitstream for the PCIe Example Design

To complete this portion of the tutorial, you must meet the following prerequisites:

- You must have a license for bitstream generation of the FPGA AI Suite IP.
- You must have a specific version of Quartus Prime Pro Edition installed:
  - The PCIe-based design example for Agilex 7 devices requires Quartus Prime Pro Edition Version 22.4 or later. This document assumes that Version 24.3 is used.
- You must have the following paths included in your $PATH environment variable:
  - `quartus/bin`
  - `qsys/bin`

If you do not have a license for FPGA AI Suite, the generated IP has a built-in inference-count limitation. Any inference operations that occur after the limit is reached generate an error message that indicates that a license is required. To reset the inference count limit, you must reprogram the bitstream onto the FPGA device.

Building an FPGA Bitstream for the PCIe-Based Design Example for Agilex 7 Devices

Run the following command for the PCIe-Based Design Example for Agilex 7 devices:

```
cd $COREDLA_WORK/demo
dla_build_example_design.py \
  -a $COREDLA_ROOT/example_architectures/AGX7_Performance.arch \
  --build -ed 3 -n 4 \
  --build-dir build_AGX7_Performance \
  --output-dir $COREDLA_WORK/demo/my_bitstreams
```

If the AOCL_BOARD_PACKAGE_ROOT environment variable is not set, the `dla_build_example_design.py` command returns an error message. To set this environment variable, review the instructions in Additional Software Prerequisites for the PCIe-based Design Example for Agilex 7 Devices on page 21.

If you want to see the full set of supported options, run the following command:

```
dla_build_example_design.py --help
```

The `dla_build_example_design.py` command provides a `--seed` option that you can use to vary the Quartus Prime random seed.

This commands places the bitstreams into the `my_bitstreams` directory. The bitstream is named `AGX7_Performance.sof`.

After the bitstream is built, you must program it onto the FPGA following the instructions in section Programming the FPGA Device on page 28.

Building an FPGA Bitstream for the PCIe-Based Design Example for Agilex 7 Devices (WSL 2)

Important: These instructions apply to running Ubuntu Linux 20.04 In a Windows Subsystem for Linux 2 (WSL 2) environment, For these instructions, you must install Quartus Prime Pro Edition for Windows on the same system as your WSL 2 environment.

To build the FPGA bitstream for the PCIe-Based Design Example for Agilex 7 devices in a WSL 2 environment:

1. On your Microsoft* Windows system, start an Ubuntu 20.04 terminal session.
2. At the Ubuntu command prompt, run the following command:

```
cd $COREDLA_WORK/demo
dla_build_example_design.py \
  -a $COREDLA_ROOT/example_architectures/AGX7_Performance.arch \
  --build -ed 3 -n 4 \
  --build-dir build_AGX7_Performance \
  --output-dir $COREDLA_WORK/demo/my_bitstreams \
  --wsl true
```

3. Follow the instructions provided by the command output. The provided instructions guide you through the following tasks:
   - Copying the finalizing command from the command output.
   - Pasting the finalizing command and running it at a Windows command prompt.

6.9. Building the Example FPGA Bitstreams

If you want to regenerate the sample bitstreams instead of using the supplied bitstreams, use the `build_bitstreams.sh` script in the `$COREDLA_ROOT/example_architectures/` folder.

6.10. Preparing a ResNet50 v1 Model

OpenVINO Model Zoo 2023.3 does not include a ResNet50 v1 model.

The following commands create `graph.xml` and `graph.bin` files for ResNet50 v1, using the `mo_caffe.py` command from OpenVINO Model Optimizer. These commands assume that you have enabled OpenVINO Model Optimizer as described Preparing OpenVINO Model Zoo on page 25.

```
wget https://www.deepdetect.com/models/resnet/ResNet-50-model.caffemodel
```

```
wget https://raw.githubusercontent.com/yihui-he/\
resnet-imagenet-caffe/master/resnet_50/ResNet-50-deploy.prototxt
```

```
source ~/build-openvino-dev/openvino_env/bin/activate
mo \
    --input_shape=[1,3,224,224] \
    --mean_values=[103.53,116.28,123.675] \
    --input=data \
    --output=prob \
    --input_model=ResNet-50-model.caffemodel \
    --input_proto=ResNet-50-deploy.prototxt \
    --model_name=graph
```

6.11. Performing Inference on the Inflated 3D (I3D) Graph

Before you try the instructions in this section, ensure that you have completed the following tasks:

- Set up OpenVINO Model Zoo as described in Preparing a Model on page 26.
- Set up Model Optimizer as described in Preparing OpenVINO Model Zoo on page 25.
- Program the FPGA and initialize the `$curarch` environment variable as described in Programming the FPGA Device on page 28

Remember: A model must be converted from a framework (such as Tensorflow, Caffe, or PyTorch) into a `.bin`/`.xml` file pair before the FPGA AI Suite compiler (`dla_compiler` command) can ingest the model.

Preparing the `i3d-rgb-tf` Model

The following commands download the `i3d-rgb-tf` TensorFlow model and run Model Optimizer:

```
source ~/build-openvino-dev/openvino_env/bin/activate
omz_downloader --name i3d-rgb-tf \
   --output_dir $COREDLA_WORK/demo/models/
omz_converter --name i3d-rgb-tf \
   --download_dir $COREDLA_WORK/demo/models/ \
   --output_dir $COREDLA_WORK/demo/models/
```

After running these commands, you can find the model in the following directory:

```
demo/models/public/i3d-rgb-tf/FP32/
```

Inference on the `i3d-rgb-tf` Model and Accuracy Calculations

This section details how to run inference on `i3d-rgb-tf` with the sample videos and ground truth files provided. For 3D input, the batch size determines how many "clips" a video should be broken down into.

To run inference on the `i3d-rgb-tf` model with 3 clips per video, run the following commands:

```
cd $COREDLA_WORK/runtime/build_Release/dla_benchmark
gnd=$COREDLA_WORK/demo/sample_videos/groundtruth_batch_size_3.txt
./dla_benchmark \
   -b=3 \
   -niter=4 \
   -m $COREDLA_WORK/demo/models/public/i3d-rgb-tf/FP32/i3d-rgb-tf.xml \
   -d=HETERO:FPGA,CPU \
   -i=$COREDLA_WORK/demo/sample_videos/ \
   -plugins=../../plugins.xml \
   -arch_file=$curarch \
   -api=async \
   -nireq=4 \
   -perf_est \
   -groundtruth_loc=$gnd \
   -folding_option=1
```

To run inference on the `i3d-rgb-tf` model with 1 clip per video, change the ground truth file (`-groundtruth_loc`) and the `-b` flag as follows:

```
cd $COREDLA_WORK/runtime/build_Release/dla_benchmark
gnd=$COREDLA_WORK/demo/sample_videos/groundtruth_batch_size_1.txt
./dla_benchmark \
   -b=1 \
   -niter=4 \
   -m $COREDLA_WORK/demo/models/public/i3d-rgb-tf/FP32/i3d-rgb-tf.xml \
   -d=HETERO:FPGA,CPU \
   -i=$COREDLA_WORK/demo/sample_videos/ \
   -plugins=../../plugins.xml \
   -arch_file=$curarch \
   -api=async \
   -nireq=4 \
   -perf_est \
   -groundtruth_loc=$gnd \
   -folding_option=1
```

Important: Loading the video data with the `dla_benchmark` command places a high memory demand on the host. Using values for the `-niter` option that are less than 100 results in manageable memory requirements for most hosts. If the memory usage is too high, the operating system might abruptly end the running `dla_benchmark` command.

Ground truth files are included for both 3 clips per video, and 1 clip per video. To use a different number of clips per video:

1. Create a new ground truth text file that contains the correct action class for each video being repeated the same amount of time as the amount of clips per video

   See the `groundtruth_batch_size_3.txt` file for reference. The ground truths are repeated 3 times each in that file.
2. Change the `-groundtruth_loc` option to point to the new ground truth text file that you created.

To run the `dla_benchmark` command on the CPU, change the `-d=HETERO:FPGA,CPU` option to `-d=HETERO:CPU`.

6.12. Performing Inference on YOLOv3 and Calculating Accuracy Metrics

This tutorial targets Agilex 7 devices and assumes that you have the following prerequisites completed:

- You have a copy of of OpenVINO Model Zoo 2023.3 in your `$COREDLA_ROOT/demo/open_model_zoo/` directory.
- You have completed the steps in Preparing OpenVINO Model Zoo on page 25.
- You have programmed the Agilex 7 FPGA device with a bitstream that corresponds to `AGX7_Performance.arch`. For example instructions, refer to Building an FPGA Bitstream for the PCIe Example Design on page 31.

The steps in the sections that follow guide you through preparing a YOLOv3 model for inference, preparing a COCO validation dataset and annotations, as well as calculating mAP and COCO AP metrics.

6.12.1. Preparing a YOLOv3 Model

As stated in Preparing a Model on page 26, a model must be converted from a framework (such as TensorFlow, Caffe, or Pytorch) into a pair of `.bin` and `.xml` files before the FPGA AI Suite compiler (`dla_compiler` command) can ingest the model.

The following commands download the YOLOv3 TensorFlow model and run Model Optimizer:

```
source ~/build-openvino-dev/openvino_env/bin/activate
omz_downloader --name yolo-v3-tf \
    --output_dir $COREDLA_WORK/demo/models/
omz_converter --name yolo-v3-tf \
    --download_dir $COREDLA_WORK/demo/models/ \
    --output_dir $COREDLA_WORK/demo/models/
```

These commands create `.bin` and `.xml` files in the `demo/models/public/yolo-v3-tf/FP32/` directory.

6.12.2. Preparing a COCO Validation Dataset and Annotations

Use the publicly available COCO 2017 validation images as input to the model and the COCO 2017 annotations as the ground-truth.

You can download the images from the following URL: http://images.cocodataset.org/zips/val2017.zip.

You can download the annotations from the following URL: http://images.cocodataset.org/annotations/annotations_trainval2017.zip.

1. Build the runtime with the following commands:

```
cd $COREDLA_WORK/runtime
rm -rf build_Release
./build_runtime.sh -target_de10_agilex
```

2. Download and extract both `.zip` files into the `coco-images` directory:

```
cd $COREDLA_WORK/runtime
mkdir coco-images
cd coco-images
```

```
wget http://images.cocodataset.org/zips/val2017.zip
wget http://images.cocodataset.org/annotations/annotations_trainval2017.zip
unzip annotations_trainval2017.zip
unzip val2017.zip
```

3. The `dla_benchmark` application allows only plain text ground truth files, so use the `convert_annotations.py` script to setup the groundtruth directory as follows:

```
cd $COREDLA_WORK/runtime/coco-images
mkdir groundtruth
python3 \
  ../dla_benchmark/convert_annotations.py \
  annotations/instances_val2017.json \
  groundtruth
```

6.12.3. Inference on YOLOv3 and Calculating Accuracy Scores

To run inference on YOLOv3 and calculate the mAP and COCO AP scores, run the following commands:

```
cd $COREDLA_WORK/runtime
./build_Release/dla_benchmark/dla_benchmark \
   -b=1 \
   -niter=5000 \
   -m $COREDLA_WORK/demo/models/public/yolo-v3-tf/FP32/yolo-v3-tf.xml \
   -d=HETERO:FPGA,CPU \
   -i=./coco-images/val2017 \
   -plugins=./plugins.xml \
   -arch_file=$COREDLA_ROOT/example_architectures/AGX7_Performance.arch \
   -yolo_version=yolo-v3-tf \
   -api=async \
   -groundtruth_loc=./coco-images/groundtruth \
   -nireq=4 \
   -enable_object_detection_ap \
   -perf_est \
   -bgr
```

The `dla_benchmark` command prints the mAP and COCO AP scores and saves a text file called `ap_report.txt` that contains the scores in the current working directory.

To enable the accuracy checking routine for object detection graphs such as YOLOv3, use the `-enable_object_detection_ap=1` option of the `dla_benchmark` command. This flag directs the `dla_benchmark` command to calculate the mAP and COCO AP for object detection graphs.

Also, specify the version of the YOLO graph that you provide to the `dla_benchmark` command with the `-yolo_version= yolo-v3-tf` option.

The input images folder is specified with `-i=./coco-images` and the ground truth annotations is specified with `-groundtruth_loc=./groundtruth`. If you chose to save the images or the ground truth annotations to a location other than the ones specified in this tutorial, update these parameters to point to the correct location.

6.13. Performing Inference Without an FPGA Board

The software emulation model is a C++ software model of the FPGA AI Suite IP that is bit-accurate(*). The emulation models relatively low-level transactions and, in some cases, models processing delays of modules.

To use the software emulation model:

1. Build the runtime with the following commands:

```
cd $COREDLA_WORK/runtime
rm -rf build_Release
./build_runtime.sh -target_emulation
```

2. Run inference with the `-niter=1` and `-nireq=1` options (because the software model is reasonably slow) with the following commands:

```
modeldir=$COREDLA_WORK/demo/models/public
imagedir=$COREDLA_WORK/demo/sample_images
curarch=$COREDLA_ROOT/example_architectures/AGX7_Performance.arch
gnd=$imagedir/TF_ground_truth.txt

cd $COREDLA_WORK/runtime/build_Release/dla_benchmark
./dla_benchmark \
   -b 1  `# Run only a single image` \
   -niter 1  `# Run only a single image` \
   -nireq 1  `# Running emulator: so -nireq=1` \
   -m $modeldir/resnet-50-tf/FP32/resnet-50-tf.xml \
    `# Same as when running on hardware - specify the graph` \
   -d HETERO:FPGA,CPU \
    `# Same as when running on hardware - \
    use FPGA if possible, fallback to CPU` \
   -i $imagedir  \
   `# Same as when running on hardware - specify image directory` \
   -arch_file $curarch \
   `# Same as when running on hardware - specify .arch file` \
   -dump_output \
   `# Dump output result.txt file` \
   -plugins emulation \
   `# Use the software emulator` \
   -groundtruth_loc $gnd
   `# Location of the ground truth file for $imagedir`
```

(*) Minor rounding differences between software emulation and hardware will typically result in differences of less than two units of least precision (ulps.

7. Running the Hostless DDR-Free Design Example

The FPGA AI Suite provides a design example to demonstrate hostless and DDR-free operation of the FPGA AI Suite IP. Graph filters, bias, and FPGA AI Suite IP configurations are stored in on-chip memory on the FPGA device instead of DDR memory on the board.

For more details about DDR-free operation, refer to DDR-Free Operation in the *FPGA AI Suite IP Reference Manual*.

Hardware Requirements

This design example requires the following hardware:

- Agilex 7 FPGA I-Series Development Kit ES2 (DK-DEV-AGI027RBES)
- Intel FPGA Download Cable

Software Requirements

This design example requires the following software:

- FPGA AI Suite
- Quartus Prime Programmer (either standalone or as part of Quartus Prime Design Suite).
- Quartus Prime System Console (either standalone or as part of Quartus Prime Design Suite).

Procedure

To run the hostless DDR-free design example with a ResNet-18 PyTorch Model:

1. Download and prepare the ResNet-18 PyTorch Model with the OpenVINO Model Optimizer with the following commands:

```
source ~/build-openvino-dev/openvino_env/bin/activate

omz_downloader --name resnet-18-pytorch \
    --output_dir $COREDLA_WORK/demo/models/

omz_converter --name resnet-18-pytorch \
    --download_dir $COREDLA_WORK/demo/models/ \
    --output_dir $COREDLA_WORK/demo/models/
```

   Important: The OpenVINO Open Model Zoo (OMZ) PyTorch models do not include a softmax operation at the end of the model.

2. Generate the parameter ROMs as `.mif` files by running the FPGA AI Suite compiler with the following command:

```
dlac \
  --batch-size=1 \
  --network-file <path/to/graph> \
  --march $COREDLA_ROOT/example_architectures/
```

```
AGX7_Streaming_Ddrfree_Resnet18.arch \
  --foutput-format=open_vino_hetero \
  --o <compiler output .bin file name>  \
  --fplugin HETERO:FPGA \
  --dumpdir $COREDLA_ROOT/resnet-18-dlac-out/
```

The `.mif` files are created a directory called `parameter_rom` in the folder specified by the `--dumpdir` option.

For details about creating the `.mif` files required for DDR-free operation, refer to "Generating Artifacts for DDR-Free Operation" in the *FPGA AI Suite Compiler Reference Manual*.

3. Build the example design with the following command:

```
dla_build_example_design.py \
  -n 1 \
  --arch=$COREDLA_ROOT/example_architectures/
AGX7_Streaming_Ddrfree_Resnet18.arch \
  --build --build-dir=<path/to/build/dir> \
  --example-design-id=0_STREAMING \
  --seed=1 \
  --parameter-rom-dir $COREDLA_ROOT/resnet-18-dlac-out/parameter_rom/
```

Building the example design creates the bitstream needed to program the FPGA device.

For more information about the `dla_build_example_design` command, refer to "Build Script" in *FPGA AI Suite PCIe-based Design Example User Guide*.

4. Program the FPGA device with the Quartus Prime Programmer.

The bitstream used to program the device is *<path/to/build/dir>*`/hw/output_files/top.sof`.

Program the FPGA device with the following command:

```
quartus_pgm -c 1 -m jtag -o "p;top.sof@1"
```

For more information about the Quartus Prime Programmer, refer to *Quartus Prime Pro Edition User Guide: Programmer*.

5. Use the Quartus Prime System Console to run inference on the example design.

Because this example design is hostless, operations that typically come from the host are performed through Quartus Prime System Console instead. For more information about the Quartus Prime System Console, refer to "Analyzing and Debugging Designs with System Console" in *Quartus Prime Pro Edition User Guide: Debug Tools*.

Use the System Console to complete the following steps:

a. Store input features in the FPGA on-chip memory.
b. Prime the FPGA AI Suite IP registers for inference.
c. Configure an ingress Modular Scatter-Gather DMA (mSGDMA) core to read the input features from on-chip memory and stream data into the FPGA AI Suite IP.
d. Configure an egress mSGDMA core to stream data from the FPGA AI Suite IP into on-chip memory.
e. Read the inference results from on-chip memory.

The design example provides a System Console script to automate these operations for you. You can find the script in the `$CORDLA_ROOT/runtime/stream/ed0_streaming_example` folder.

To use the design example System Console script:

a. Run the following commands:

```
cd <quartus-install-path>

system-console –-script=system_console_script.tcl <path-to-img.bin> \
               <#-of-inferences> <output-channels> <output-height> \
               <output-width>
```

The design example Quartus Prime System Console script generates a file called `output.bin` that contains the raw inference results.

b. (Optional) To measure the performance of the design example, run the following commands:

```
cd <quartus-install-path>

system-console –-script=system_console_perf.tcl <path-to-img.bin>
```

6. Postprocess the raw inference output for readability with the following command:

```
python3 $COREDLA_ROOT/bin/streaming_post_processing.py <path-to-output.bin>
```

This script cleans the raw output binary file by script removing some invalid bytes and storing an FP16 formatted `result_hw.txt` file for readability.

A. FPGA AI Suite Getting Started Guide Archives

For the latest and previous versions of this user guide, refer to FPGA AI Suite Getting Started Guide. If an FPGA AI Suite software version is not listed, the user guide for the previous software version applies.

B. FPGA AI Suite Getting Started Guide Document Revision History

Document Version	FPGA AI SuiteVersion	Changes
2024.12.16	2024.3	• Updated "Installing Quartus Prime Pro Edition Software" to update the Quartus Prime Pro Edition version required by the various FPGA boards supported by the FPGA AI Suite design examples.
2024.11.25	2024.3	• Added description of the JTAG design example for Agilex 5 E-Series devices to "FPGA AI Suite Components".
2024.09.06	2024.2	• Replaced occurrences of "memoryless" with "DDR-free". • Replaced occurrences of "software reference model" with "software emulation model".
2024.07.31	2024.2	• Added "Running the Hostless Memoryless Design Example". • Added "Performing Inference Without an FPGA Board". • Updated supported OpenVINO version to 2023.3 LTS. • Revised "Setting Required Environment Variables". • Removed references to Intel PAC with Arria 10 GX FPGA. This card is no longer supported by FPGA AI Suite. • Removed references to Ubuntu 18. This operating system is no longer supported by FPGA AI Suite. • Replaced references to the `-plugins_xml_file` option with the `-plugins` option. The `-plugins_xml_file` option is deprecated and will be removed in a future release.
2024.03.29	2024.1	• Added support for Ubuntu 22.04.
2024.02.12	2023.3.1	• Updated for additional Agilex 7 support.
2024.02.02	2023.3	• Corrected OpenVINO version in graphic in "FPGA AI Suite Installation Overview".
2023.12.01	2023.3	• Added instructions for a WSL 2 environment to "Building an FPGA Bitstream for the PCIe Example Design".
2023.09.06	2023.2.1	• Updated supported OpenVINO version to 2022.3.1 LTS. • Updated installation instructions for Ubuntu 18 operating systems.
2023.07.17	2023.2	• Corrected instructions in "Installing OpenVINO Toolkit".
2023.07.03	2023.2	• Added "Installing FPGA AI Suite on Windows Subsystem for Linux". • Revised "OpenVINO Toolkit" for OpenVINO 2022.3 LTS. • Renamed "Installing the FPGA AI Suite" to "FPGA AI Suite Installation Overview" to better reflect the contents of that section. • Updated supported OpenVINO version to 2022.3 LTS. • Updated OpenVINO installation paths to `/opt/intel/openvino_2023`.

continued...

Document Version	FPGA AI SuiteVersion	Changes
		• Updated FPGA AI Suite installation paths to `/opt/intel/fpga_ai_suite_2023.2`. • Changed occurrences of `tools/downloader/downloader.py` to `omz_downloader`. • Changed occurrences of `tools/downloader/converter.py` to `omz_converter`.
2023.04.05	2023.1	• Added "Performing Inference on the i3d Graph". • Renamed the`dlac` command. The FPGA AI Suite compiler command is now `dla_compiler`. • Updated the Intel Agilex product family name to "Intel Agilex® 7."
2022.12.23	2022.2	• Update for the Debian package-based install. • Add walk-through for validating a YOLOv3 model. • Update to account for the new Terasic DE10-Agilex BSP ZIP file. • Renamed `dla_create_and_build_pcie_ed.py` to `dla_build_example_design.py`
2022.04.27	2022.1	• Minor fixes to some file names, including one for the running inference in the AOT flow on the FPGA.
2022.04.12	2022.1	• Notes on installing CMake for CentOS 7. • Clarifications to the Model Zoo and Model Optimizer install instructions.
2022.04.08	2022.1	• Extensively revised installation instructions. • Added installation information for the PCI-based design examples. • Enhanced details for Intel Agilex devices. • Added a quick-start tutorial. • Removed supplementary performance details.
2021.09.10	2021.2	• Added information for initial Intel Agilex device family support.
2021.04.30	2021.1	• Expanded list of tasks performed by the Intel FPGA AI Suite Compiler. • Distinguished more clearly between requirements for the IP and requirements for the Example Designs.
2020.12.04	2020.2	• Renamed the script that sets the Intel FPGA AI Suite environment variables to init_env.sh. It was called coredla_developer_init_env.sh in earlier releases.
2020.10.30	2020.1	• Initial release.