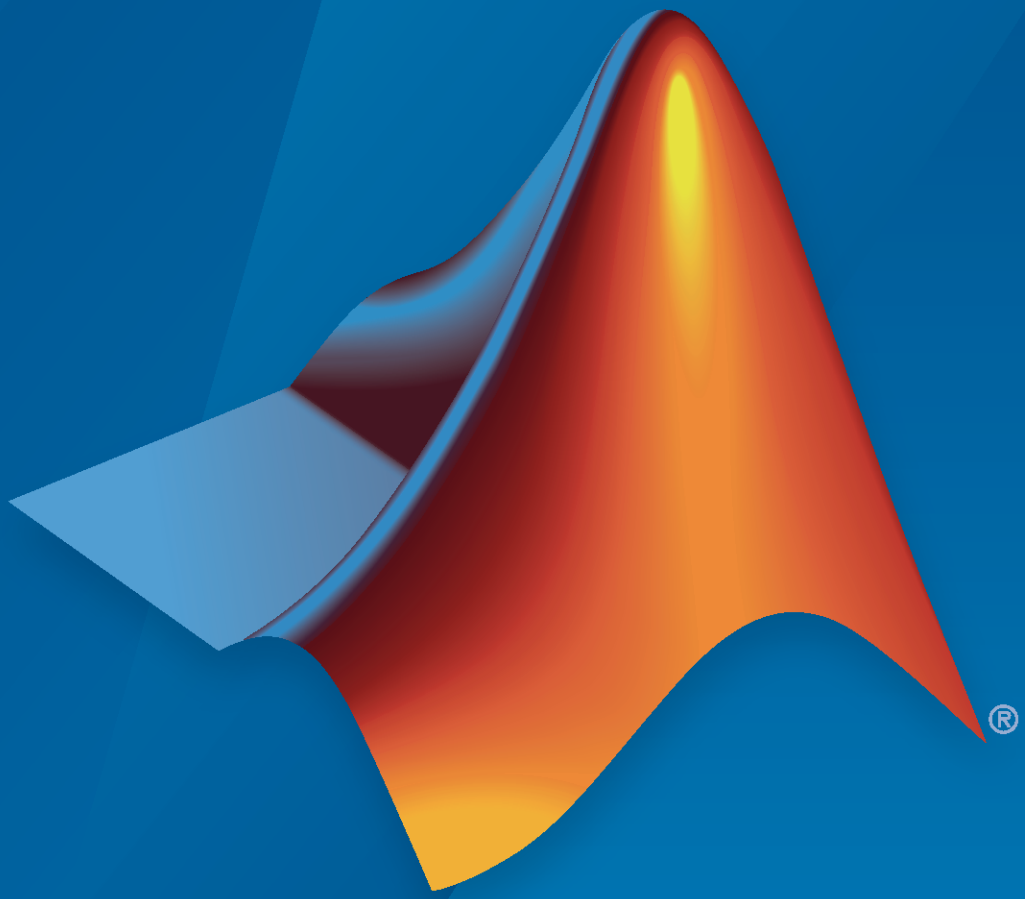
Deep Learning HDL Toolbox™

Getting Started Guide



MATLAB®

R2022a

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Deep Learning HDL Toolbox™ Getting Started Guide

Revision History

September 2020	Online only	New for Version 1.0 (Release 2020b)
March 2021	Online only	Revised for Version 1.1 (Release 2021a)
September 2021	Online only	Revised for Version 1.2 (Release R2021b)
March 2022	Online only	Revised for Version 1.3 (Release R2022a)

Contents

1 Install and Set Up Prerequisite Products and Third-Party Tools

2 Tutorials

Install and Set Up Prerequisite Products and Third-Party Tools

Deep Learning HDL Toolbox Product Description

Prototype and deploy deep learning networks on FPGAs and SoCs

Deep Learning HDL Toolbox™ provides functions and tools to prototype and implement deep learning networks on FPGAs and SoCs. It provides pre-built bitstreams for running a variety of deep learning networks on supported Xilinx® and Intel® FPGA and SoC devices. Profiling and estimation tools let you customize a deep learning network by exploring design, performance, and resource utilization tradeoffs.

Deep Learning HDL Toolbox enables you to customize the hardware implementation of your deep learning network and generate portable, synthesizable Verilog® and VHDL® code for deployment on any FPGA (with HDL Coder™ and Simulink®).

Use Deep Learning on FPGA Bitstreams

The Deep Learning HDL Toolbox hardware support packages provide bitstreams that you can use to deploy various deep learning networks on the target platform.

This table illustrates the mapping between the target boards, data types, and bitstream names.

Target Board	Data Type	Bitstream Name
Xilinx Zynq® -7000 ZC706	`single`	`'zc706_single'`
Xilinx Zynq-7000 ZC706	`int8`	`'zc706_int8'`
Xilinx Zynq UltraScale™ ZCU102	`single`	`'zcu102_single'`
Xilinx Zynq UltraScale ZCU102	`int8`	`'zcu102_int8'`
Intel Arria® 10 SoC development kit	`single`	`'arria10soc_single'`
Intel Arria 10 SoC development kit	`int8`	`'arria10soc_int8'`

For an example that illustrates how you can use these bitstreams names and deploy your network when running the workflow, see “Prototype Deep Learning Networks on FPGA and SoCs Workflow”.

Configure FPGA Boards

Prepare your target FPGA boards for deploying a deep learning network by configuring them to connect to your host computer.

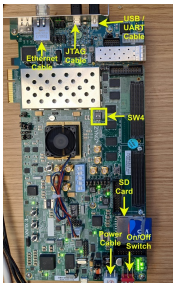
Xilinx Zynq-7000 ZC706 Evaluation Board

To set up the board:

1. Plug in the power cord, and then connect the host computer to the FPGA board by using a JTAG cable.
2. Specify the SW4 switch settings to use the Digilent USB-TO-JTAG interface.

Configuration Source	SW4 switch 1	SW4 switch 2
None	0	0
Cable Connector J3	1	0
Digilent USB-TO-JTAG Interface	0	1
JTAG (flying lead)Header J62	1	1

This graphic shows the configuration settings for the Xilinx Zynq-7000 ZC706 Evaluation Board.



To learn more about the board configuration, see the Xilinx ZC706 Evaluation Board User Guide.

Intel Arria 10 SoC Development Kit

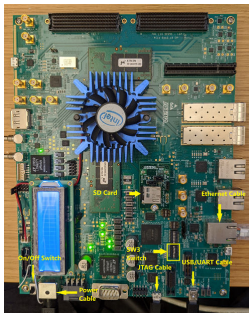
To set up the board:

1. Plug in the power cord, and then connect the host computer to the FPGA board by using a JTAG cable.
2. Specify the SW3 switch settings.

Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
Off	On	On	On	On	Off	Off	Off

3. Connect two DDR4 plugin boards to the memory plugin slot.

This graphic shows the configuration settings for the Intel Arria 10 SoC development kit.



To learn more about the board configuration, see the Arria 10 SoC Development Kit User Guide.

Xilinx Zynq UltraScale+ MPSoC ZCU102 FPGA Development Board

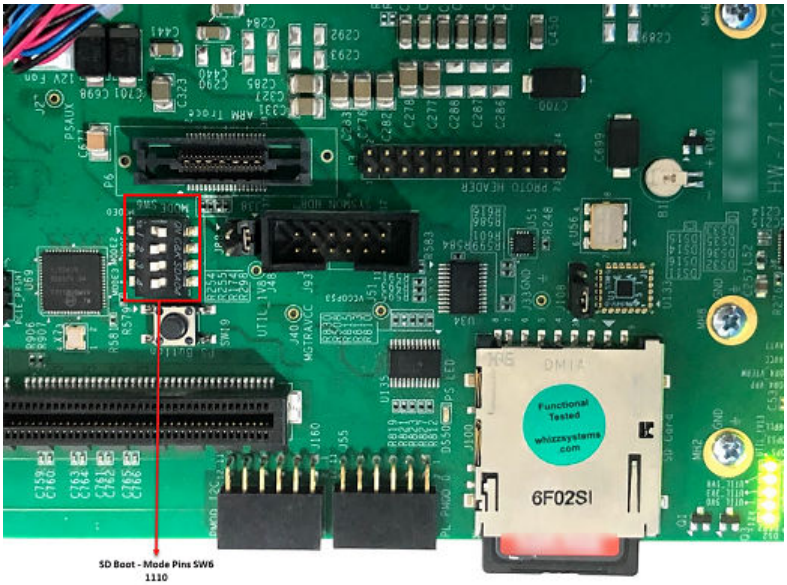
To set up the board:

1. Plug in the power cord. If using JTAG, connect the FPGA board to the host computer by using a JTAG cable. If using Ethernet, connect the FPGA board to the host computer by using an Ethernet cable.
2. Configure the SW6 switch.

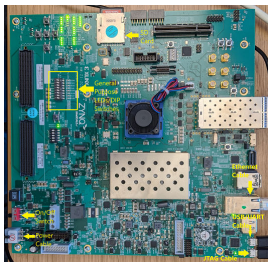
Boot Mode	Mode Pins [3:0]	SW6 Switch Position [3:0]
JTAG	0, 0, 0, 0	on, on, on, on
QSPI32	0, 0, 1, 0	on, on, off, on
SD	1, 1, 1, 0	off, off, off, on

The SW6 default position is QSPI32. For the SW6 DIP switch, moving the switch towards the ON label is 0.

This graphic shows the location of the SW6 switch.



This graphic shows the configuration settings for the Xilinx Zynq UltraScale+™ MPSoC ZCU102 FPGA development board.



To learn more about the ZCU102 hardware setup, refer to the Xilinx documentation.

JTAG Connection

Vendor	Required Hardware	Required Software
Intel	USB Blaster I or USB Blaster II download cable	• USB Blaster I or II driver • For Windows® operating systems: Quartus® Prime executable directory must be on system path. • For Linux® operating systems: versions below Quartus II 13.1 are not supported. Quartus II 14.1 is not supported. Only 64-bit Quartus is supported. Quartus library directory must be on `LD_LIBRARY_PATH` *before* starting MATLAB®. Prepend the Linux distribution library path before the Quartus library on `LD_LIBRARY_PATH`. For example, `/lib/x86_64-linux-gnu:$QUARTUS_PATH`.
Xilinx	Digilent® download cable • If your board has an onboard Digilent USB-JTAG module, use a USB cable • If your board has a standard Xilinx 14 pin JTAG connector, use with HS2 or HS3 cable from Digilent	• For Windows operating systems: Xilinx Vivado® executable directory must be on system path. • For Linux operating systems: Digilent Adept 2. For the installation steps, see “Install Digilent Adept 2 Runtime” (HDL Verifier Support Package for Xilinx FPGA Boards).
	FTDI USB-JTAG cable • Supported for boards with onboard FT4232H, FT232H, or FT2232H devices implementing USB-to JTAG	Install these D2XX drivers. • For Windows operating systems: 2.12.28 (64 bit) • For Linux operating systems: 1.4.22 (64 bit) For the installation guide, see D2XX Drivers from the FTDI Chip website.
Microsemi®	JTAG connection not supported	

Note When simulating your FPGA design through Digilent JTAG cable with Simulink or MATLAB, you cannot use any debugging software that requires access to the JTAG; for example, Vivado Logic Analyzer.

Tutorials

Try Deep Learning on FPGA with Only Five Additional Lines of MATLAB Code

Use the Deep Learning HDL Toolbox to deploy a pretrained deep learning network to a target board and identify objects on a live webcam connected to the development computer by adding only five lines of MATLAB code to the “Try Deep Learning in 10 Lines of MATLAB Code” example.

1. To connect to a webcam and load the pretrained ResNet-18 network:.

   ```
   camera = webcam; % Connect to the camera
   net = resnet18;   % Load the neural network
   ```

   If you need to install the webcam and ResNet-18 add-ons, a message appears with a link to help you download the free add-ons using Add-On Explorer. Alternatively, see Deep Learning Toolbox Model *for ResNet-18 Network* and MATLAB Support Package for USB Webcams for installation instructions.

   After you install Deep Learning Toolbox Model for ResNet-18 network, you can use it to classify images. ResNet-18 is a pretrained model that has been trained on a subset of the ImageNet database. The model is trained on more than a million images and can classify images into 1000 object categories such as keyboard, mouse, cup, pencil, and so on.
2. To set up the interface to the target board, create the workflow object, and deploy the network to the target board:

   ```
   hT = dlhdl.Target('Xilinx',Interface = 'Ethernet');
   hW = dlhdl.Workflow('Network',net,'Bitstream','zcu102_single','Target',hT);
   hW.deploy;
   ```
3. To show and classify live images:

   ```
   while true
       im = snapshot(camera);       % Take a picture
       image(im);                   % Show the picture
       im = imresize(im,[224 224]); % Resize the picture for ResNet-18
       [prediction, speed] = hW.predict(single(im),'Profile','on');
       [val, idx] = max(prediction);
       label = net.Layers(end).ClassNames{idx}; % Classify the image
       title(char(label));          % Show the class label
       drawnow
      end
   ```

   Point the webcam at an object. The pretrained deep learning network reports what class of object it thinks the webcam is showing, classifying images until you press **Ctrl+C**. The code resizes the image for the network by using `imresize`.

   For example, the network correctly classifies a coffee mug. Experiment with objects in your surroundings to see how accurate the network is.

For next steps, see “Deep Learning on FPGA Solution”.

See Also

`resnet18` | `dlhdl.Workflow` | `dlhdl.Target`

More About

- “Prototype Deep Learning Networks on FPGA and SoCs Workflow”
- “Supported Networks, Layers, Boards, and Tools”