



# Face Tracking Using Convolutional Neural Network Accelerator IP

Reference design supported by Neural Network Compiler, CNN IP, or Compact CNN IP 1.1. Not recommended for new designs.

## Reference Design

FPGA-RD-02037-1.2

March 2021

## Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

## Contents

Acronyms in This Document .....	4
1. Introduction .....	5
2. Related Documentation .....	6
2.1. Soft IP Document .....	6
2.2. Diamond Document .....	6
3. Software Requirements .....	6
4. Hardware Requirements .....	6
5. Reference Design Overview .....	7
5.1. Block Diagram .....	7
5.2. Top Level Blocks .....	7
6. CNN Accelerator Engine .....	8
7. SD Card Loader .....	11
8. AXI Slave and DDR3 Memory Interface .....	12
9. CSI2 to DVI Interface .....	12
10. Video Processing Module .....	12
References .....	14
Technical Support Assistance .....	14
Revision History .....	15

## Figures

Figure 1.1. Lattice Embedded Vision Development Kit with MicroSD Card Adapter Board .....	5
Figure 5.1. Face Tracking Reference Design Block Diagram .....	7
Figure 6.1. CNN Accelerator IP Core Generation User Interface .....	8
Figure 6.2. CNN Accelerator Soft IP Inputs and Outputs .....	9
Figure 6.3. Command Format .....	10
Figure 7.1. Neural Network Compiler Output File Generation and Flow .....	11
Figure 10.1. Face Tracking Design *.yml File Code Snippet .....	12

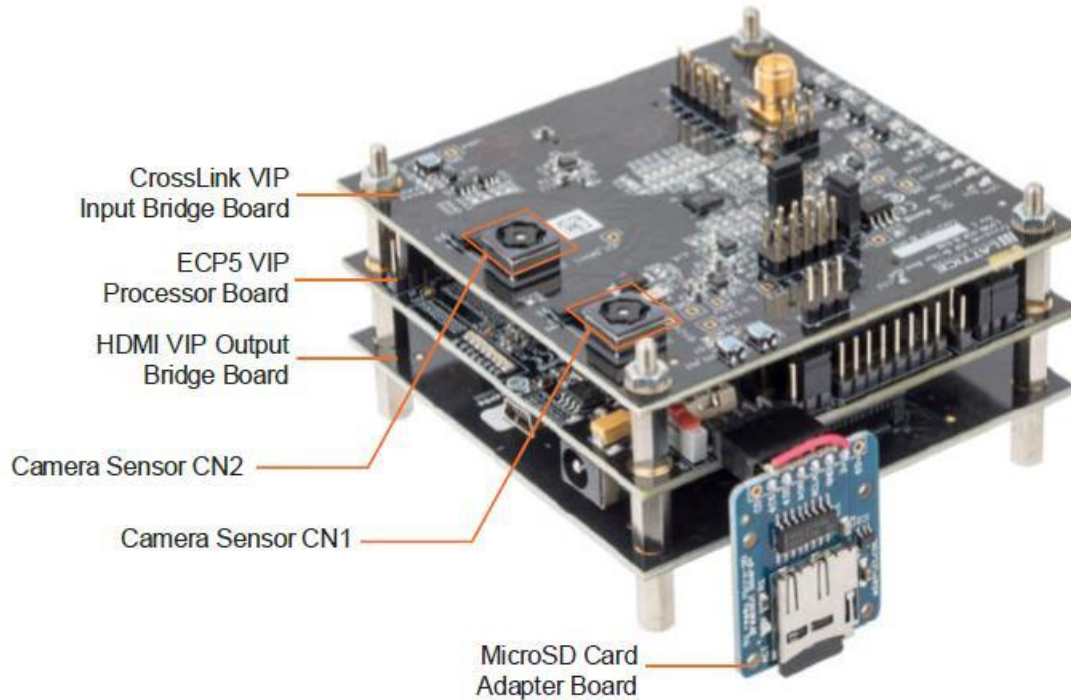
## Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AXI	Advanced Extensible Interface
CNN	Convolutional Neural Network
DRAM	Dynamic Random Access Memory
FPGA	Field-Programmable Gate Array
DVI	Digital Visual Interface
pASSP	Programmable Application Specific Standard Product
PIP	Picture In Picture
SPI	Serial Peripheral Interface
SD Card	Secure Digital (Memory) Card

## 1. Introduction

This document describes the Face Tracking machine learning neural network reference design. This reference design can be implemented on Lattice's Embedded Vision Development Kit (EVDK), featuring the CrossLink™ and ECP5™ FPGAs.



**Figure 1.1. Lattice Embedded Vision Development Kit with MicroSD Card Adapter Board**

## 2. Related Documentation

### 2.1. Soft IP Document

- [CNN Accelerator IP Core User Guide \(FPGA-IPUG-02037\)](#)

### 2.2. Diamond Document

For more information on Lattice Diamond software, visit Lattice website at:

[www.latticesemi.com/Products/DesignSoftwareAndIP](http://www.latticesemi.com/Products/DesignSoftwareAndIP)

## 3. Software Requirements

- Lattice Diamond® Software Version 3.10
- Synplify Pro® Synthesis Tool

## 4. Hardware Requirements

- Lattice Embedded Vision Development Kit (LF-EVDK1-EVN)
- Mini-USB Cable (included in the Lattice Embedded Vision Development Kit)
- 12 V Power Supply (included with the Kit)
- HDMI Cable
- HDMI Monitor (1080p60)
- Micro-SD Card Adapter (MICROSD-ADP-ENV)
- Micro-SD Card. Standard Micro-SD card only.

## 5. Reference Design Overview

### 5.1. Block Diagram

Figure 5.1 shows the block diagram of the Face Tracking reference design.

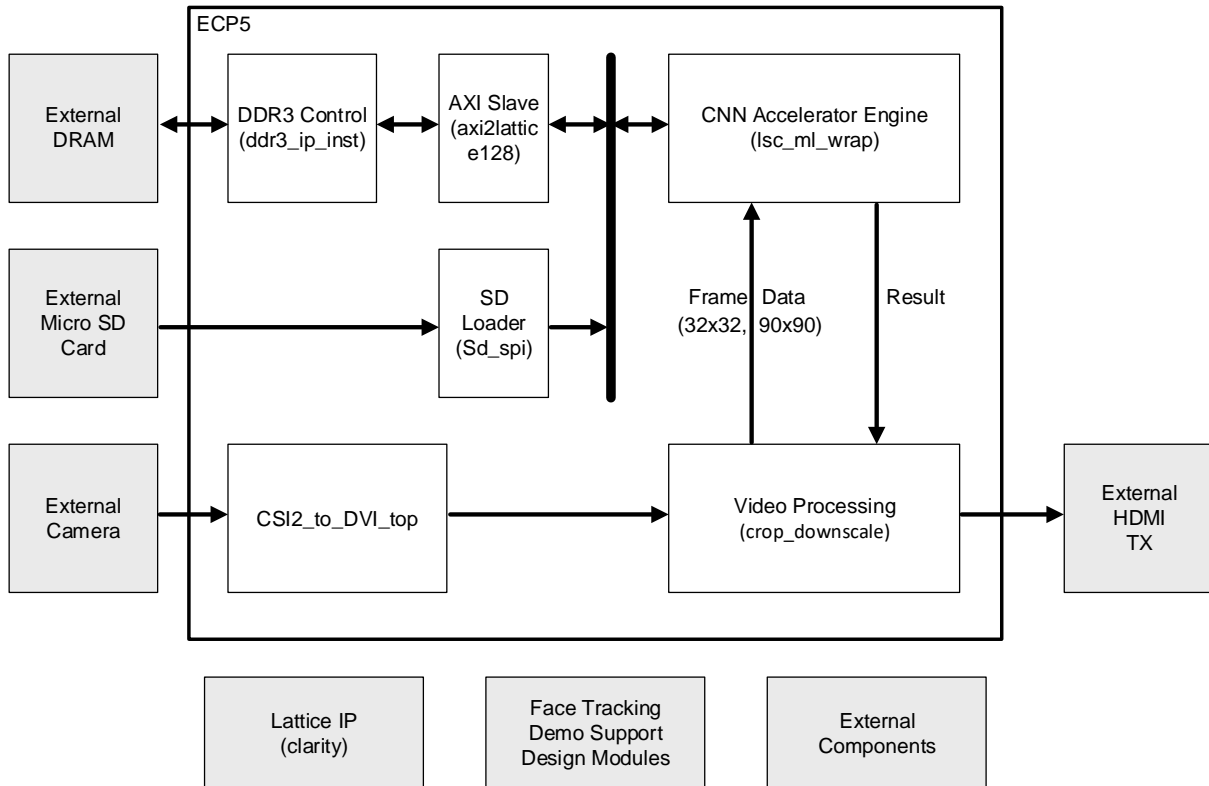


Figure 5.1. Face Tracking Reference Design Block Diagram

### 5.2. Top Level Blocks

The reference design uses ECP5-85 FPGA containing the following major blocks:

- CNN Accelerator Engine
- SD card to SPI interface
- AXI Slave interface
- DDR3 memory interface
- CSI2 to DVI interface
- Video processing module

## 6. CNN Accelerator Engine

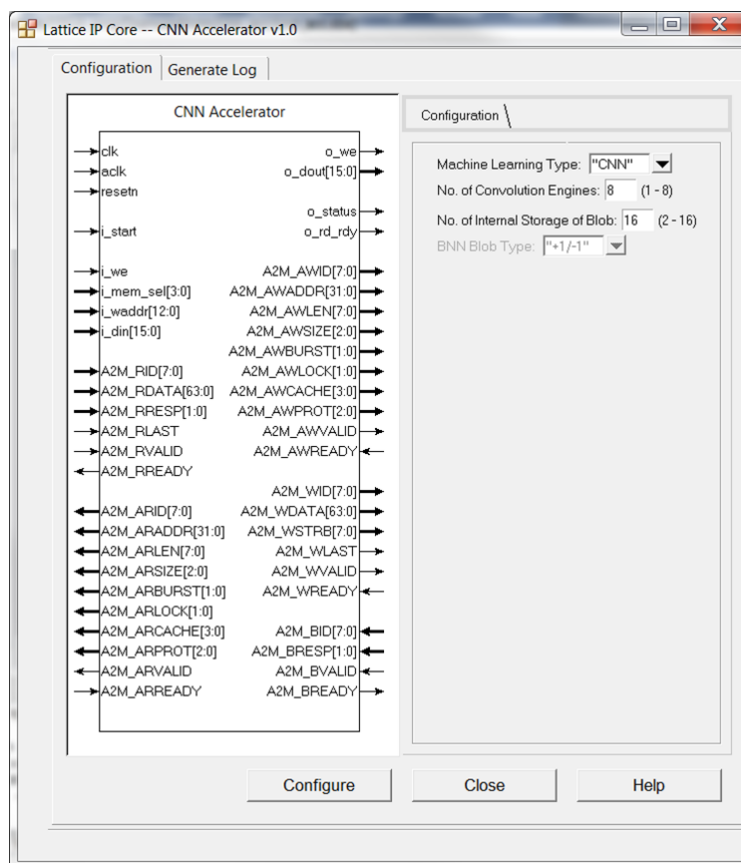
The Lattice Semiconductor CNN Accelerator IP Core is a calculation engine for Deep Neural Network with fixed point weight or binarized weight. It calculates full layers of Neural Network including the convolution layer, pooling layer, batch normalization layer, and full connect layer by executing sequence code with weight value generated by the Lattice Neural Network Compiler tool. The engine is optimized for convolutional neural network, hence it can be used for vision-based applications such as classification or object detection and tracking. The IP Core does not require an extra processor; it can perform all required calculations by itself.

The design is implemented in Verilog HDL. It can be targeted to ECP5 and ECP5-5G (LFE5U, LFE5UM, and LFE5UM5G) FPGA devices and implemented using the Lattice Diamond Software Place and Route tool integrated with the Synplify Pro synthesis tool.

The key features of the CNN Accelerator IP Core include:

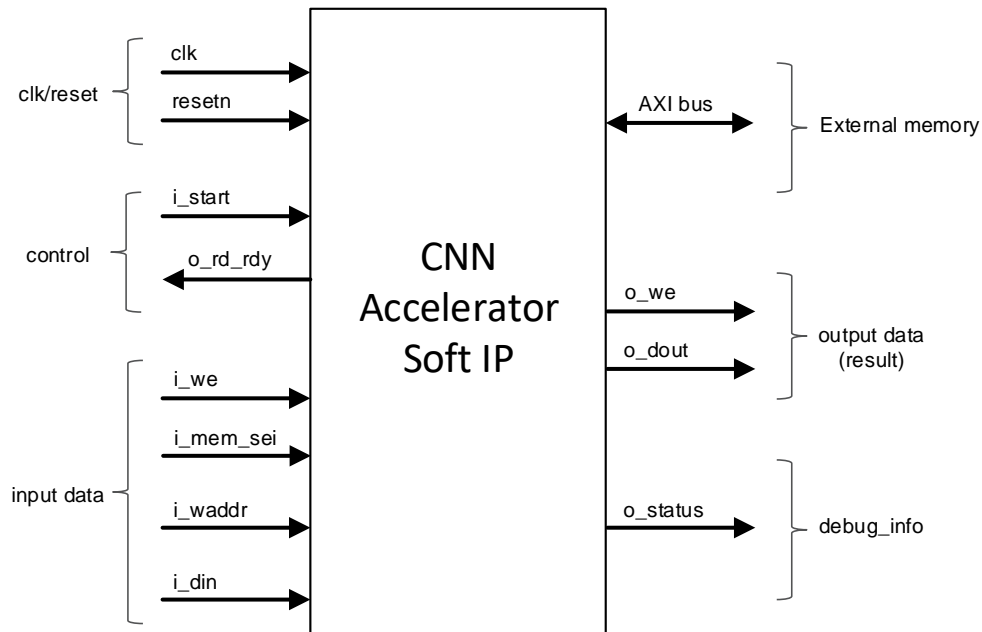
- Supports convolution layer, max pooling layer, batch normalization layer and full connect layer
- Configurable bit width of weight (16-bit, 1-bit)
- Configurable bit width of activation (16/8-bit, 1-bit)
- Dynamically support 16-bit and 8-bit width of activation
- Configurable number of memory blocks for tradeoff between resource and performance
- Configurable number of convolution engines for tradeoff between resource and performance
- Optimized for 3 x 3 2D convolution calculation
- Dynamically support various 1D convolution from 1 to 72 taps
- Support max pooling with overlap (For example, kernel 3 and stride 2)

Engine configuration parameters can be set using the Clarity Designer’s IP Core Configuration user interface as shown in [Figure 6.1](#).



**Figure 6.1. CNN Accelerator IP Core Generation User Interface**

Figure 6.2 shows the inputs and outputs of this IP module.



**Figure 6.2. CNN Accelerator Soft IP Inputs and Outputs**

#### Input Data Format:

Input data is a sequence of 8-bit or 16-bit data. Memory index and address are decided by Neural Network. Therefore, external block should process input raw data and write input data to Lattice CNN Accelerator IP Core through input data write interface. Since CNN Accelerator IP Core has only 16-bit width interface, external block should pack two of 8-bit data if 8-bit width is used for input data layer.

For example, face detection neural network may take 32 x 32 of R, G, B planes at memory index 0 with address 0x0000 for Red plane, 0x0400 for Green plane, and 0x0800 for Blue plane. Object detection neural network on the other hand may take 90 x 90 of R, G, B planes which are assigned to memory index 0, 1, and 2, respectively. Because memory assignment is defined by Neural Network, external block should handle input raw data and write it to proper position of internal memory of the CNN Accelerator IP Core.

Writing input data to DRAM and using Load command to fetch input data are also possible for the case of large input data. The IP core expects data in little-endian order.

#### Result:

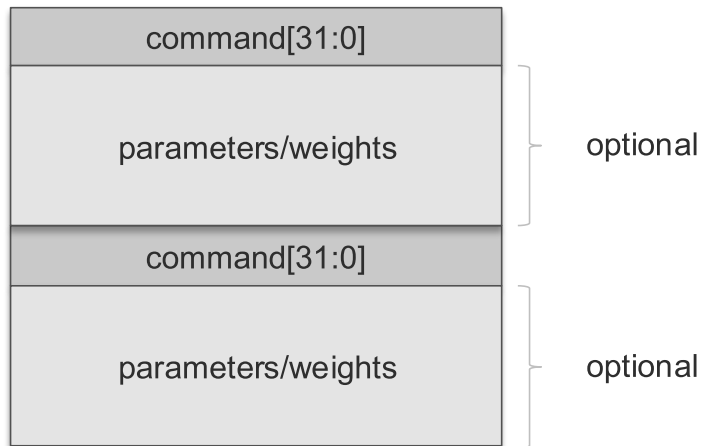
Result, that is, by final blob data of neural network can be written to DRAM per command code. In this case, external logic should read result data from DRAM. However, command code also can simply feed result data to external logic through this interface. Interface consists of `o_we` as valid indicator and `o_dout` as 16-bit data. Usually, it is a single burst series of 16-bit data and it's also fully programmable by command code.

#### Output Data Format:

Output data is a sequence of 16-bit data which is controlled by commands. Amount of data is also decided by Neural Network, that is, by output blobs. External block should interpret output sequence and generate usable information. For example, face detection design outputs 2-bit burst (two consecutive) of 16-bit data, first is confidence of non-face while the second one is confidence of face. Whenever the latter is larger than the former, conclusion is Face. The IP core outputs data is in little-endian order.

#### Command Format:

Command is a sequence of 32-bit data with or without additional parameters or weights as shown in Figure 6.3. It should be loaded at DRAM address 0x0000 before execution. Command is generated by the Lattice Neural Network Compiler tool. For more information, refer to [Lattice Neural Network Compiler Software User Guide \(FPGA-UG-02052\)](#).



**Figure 6.3. Command Format**

## 7. SD Card Loader

SD card interface in this design is used to get the input data into the CNN accelerator IP. SD card contains a binary file that is generated by Lattice Neural Network Compiler Tool.

Lattice Neural Network Compiler tool allows analyzing and compiling a trained neural network (such as what is generated by Caffe or TensorFlow tools) for use with select Lattice Semiconductor FPGA products.

Lattice Neural Network Compiler tool outputs three files:

- A hardware configuration file (\*.yml) that contains info on fixed point converted network and memory allocation.
- A firmware file (\*.lscml) that contains weights coming from a trained model file.
- A binary file (\*.bin) generated from firmware file (\*.lscml) for programming into SD card.

Figure 7.1 shows the output file generation flow of the Neural Network Compiler tool.

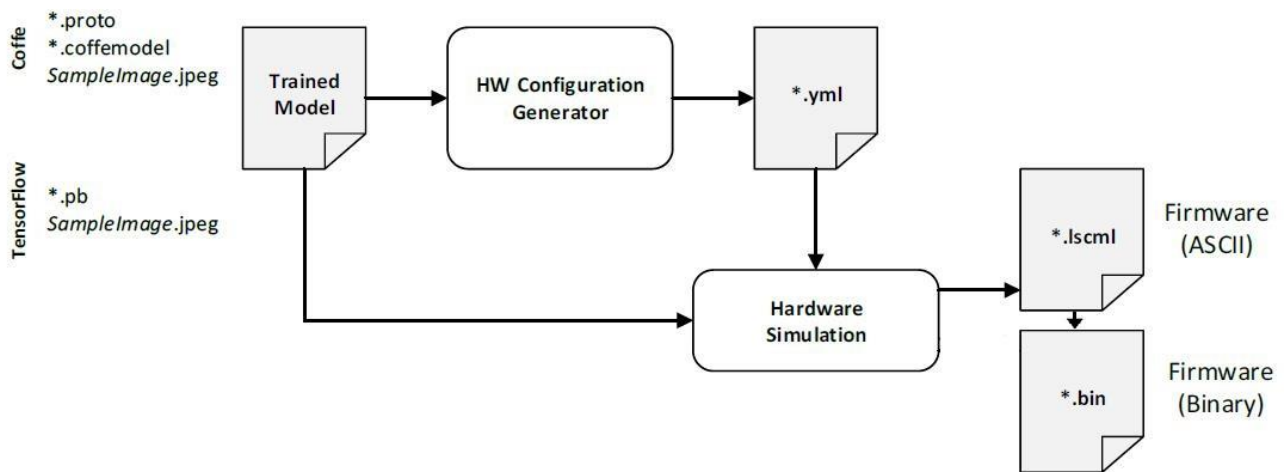


Figure 7.1. Neural Network Compiler Output File Generation and Flow

## 8. AXI Slave and DDR3 Memory Interface

AXI interface allows command code to be written in DRAM before execution of CNN Accelerator IP Core. Input data may also be written in DRAM. CNN Accelerator IP Core reads command code from DRAM, and performs calculations using internal sub execution engines. Intermediate data may also be transferred from/to DRAM per command code.

## 9. CSI2 to DVI Interface

This module implements a bridge function that converts the camera input's MIPI CSI data to DVI output using CrossLink and Si1136 HDMI transmitter.

## 10. Video Processing Module

The crop\_downscale module provides all the necessary functions needed to manage the process of inputting data, receiving output data, and generating a composite image for output to the HDMI interface. Three examples are included in the design:

- crop\_downscale.v – crops input to 32x32
- crop\_downscale\_key.v – crops input to 90x90
- crop\_downscale\_keyL.v – crops input to 224x224

The face tracking demo uses “crop\_downscale\_key.v”. Key functions of the code include:

- Capturing a downscaled image from the camera input module and saving it to a frame buffer.
- Writing the frame buffer data into CNN accelerator engine during the blanking period.
- Buffering the output after completion of the image data processing.
- Creating a PIP (Picture-in-Picture) bounding box with green borders and outputting the composite image.

Output from CSI2\_to\_DVI\_top module is a stream of data that reflects the camera image. Input image is then downscaled to 90x90 pixels, stored in a frame buffer and passed to output.

Image data is written from the frame buffer into the CNN acceleration engine prior to the start of the processing. Data is then formatted for compatibility with the trained network.

The \*.yml file provides majority of the information needed for understanding how the input data should be prepared. A snippet of the code in \*.yml file for face tracking design is shown in [Figure 10.1](#).

```
1  cfrac: 1024
2  Mean: 128
3  Scale: 255
4  Input Size: [1, 3, 90, 90]
5  num_ebr: 16
6  ebr_blk_size: 16384
7  num_conv_eng: 8
8  binary_mode: false
9  blob:
10  data:
11  |   frac: 8
12  |   memblks: 3
13  |   extmem: 0x00c000
14  |   depth_per_mem: 1
```

Figure 10.1. Face Tracking Design \*.yml File Code Snippet

- Input Size: [1, 3, 90, 90] -- Indicates one input array consisting of 3 layers of dimensions 90x90.
- memblks: 3 – Total number of memory blocks needed.
- depth\_per\_mem: 1 – Number of memory blocks allocated to each memory layer.
- frac: 8 – Number of bits that is allocated to the fractional component. It is equal to the minimum number of bits to represent this number minus 1. In this case, 3 bits to represent 8-1=7.
- num\_ebr: 16 – Number of memory blocks.

**Note:** Despite the variable name, this does not tie directly to the number of Embedded Block Ram (EBR) used in the design.

- ebr\_blk\_size: 16384 – This defines the size of the memory blocks in Bytes. Note the blocks have a width of 16 bits and the depth is variable.

CNN accelerator engine's port results, o\_we and o\_dout[15:0], can be used to output any number of results.

Designer can add a read command to allow reading any data based on the neural network design.

In face tracking design, we use 4 results to be transferred from CNN accelerator engine to crop\_downscale\_key module. These are used to create a green bounding box around the face on the output image stream:

- X pixels (from image top left to box center)
- Y pixels (from image top left to box center)
- Box X (dimension in pixels)
- Box Y (dimension in pixels)

## References

For more information on the FPGA device, visit: [http://www.latticesemi.com/Products/FPGAandCPLD/ECP5\\_](http://www.latticesemi.com/Products/FPGAandCPLD/ECP5_)

For complete information on Lattice Diamond Project-Based Environment, Design Flow, Implementation Flow and Tasks, as well as on the Simulation Flow, see the [Lattice Diamond User Guide](#).

## Technical Support Assistance

Submit a technical support case through [www.latticesemi.com/techsupport](http://www.latticesemi.com/techsupport).

## Revision History

### Revision 1.2, March 2021

Section	Change Summary
All	Corrected type Salve to <i>Slave</i> across the document.

### Revision 1.1, May 2019

Section	Change Summary
All	<ul style="list-style-type: none"> <li>Added <i>Reference design supported by Neural Network Compiler, CNN IP, or Compact CNN IP 1.1. Not recommended for new designs</i> on the front cover page.</li> <li>Added <a href="#">Disclaimers</a> section.</li> <li>Updated the back cover page of the document.</li> </ul>
Revision History	Updated revision history table to new template.

### Revision 1.0, May 2018

Section	Change Summary
All	Initial release



[www.latticesemi.com](http://www.latticesemi.com)