# CNN Accelerator IP Core

# User Guide

FPGA-IPUG-02037 Version 1.1

September 2018

# Contents

1. Introduction ..... 4
1.1. Quick Facts ..... 4
1.2. Features ..... 4
2. Functional Descriptions ..... 5
2.1. Overview ..... 5
2.2. Interface Descriptions ..... 6
2.2.1. Control and Status Interface ..... 8
2.2.2. Input Data Interface ..... 9
2.2.3. Result Interface ..... 9
2.2.4. DRAM Interface ..... 9
2.3. Clock Domain ..... 9
2.4. Reset Behavior ..... 10
2.5. Register Description ..... 11
2.6. Operation Sequence ..... 11
2.6.1. Command Format ..... 11
2.6.2. Input Data Format ..... 11
2.6.3. Output Data Format ..... 12
2.7. Supported Commands ..... 12
3. Parameter Settings ..... 13
4. IP Generation and Evaluation ..... 15
4.1. Licensing the IP ..... 15
4.2. Generation and Synthesis ..... 15
4.2.1. Getting Started ..... 15
4.2.2. Configuring the IP Core in Clarity ..... 16
4.2.3. Instantiating the IP Core ..... 16
4.3. Running Functional Simulation ..... 17
4.4. Hardware Evaluation ..... 17
5. Ordering Part Number ..... 18
References ..... 19
Technical Support Assistance ..... 19
Appendix A. Resource Utilization ..... 20
Revision History ..... 21

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

# Figures

Figure 2.1. Functional Block Diagram ..... 5
Figure 2.2. CNN Accelerator IP Core Interface Diagram ..... 6
Figure 2.3. Control and Status Interface Timing Diagram ..... 8
Figure 2.4. Input Data Interface Timing Diagram ..... 9
Figure 2.5. Result Interface Timing Diagram ..... 9
Figure 2.6. Reset Timing Diagram ..... 10
Figure 2.7. Reset Timing Diagram ..... 10
Figure 2.8. Command format ..... 11
Figure 3.1. CNN Accelerator IP Core Configuration User Interface ..... 13
Figure 4.1. CNN Accelerator IP Core in Clarity Designer Catalog Tab ..... 15

# Tables

Table 1.1. Quick Facts ..... 4
Table 2.1. CNN Accelerator IP Core Signal Descriptions ..... 6
Table 3.1. Attributes Table ..... 13
Table 3.2. Attributes Descriptions ..... 14
Table 4.1. File List ..... 16
Table A.1. Performance and Resource Utilization[1] ..... 20

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

# 1. Introduction

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 convolution layer, pooling layer, batch normalization layer, and full connect layer by executing sequence code with weight value which is generated by Lattice Neural Network Compiler. The engine is optimized for convolutional neural network, so it can be used for vision-based application 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 FPGA devices, and implemented using the Lattice Diamond® Software Place and Route tool integrated with the Synplify Pro® synthesis tool.

## 1.1. Quick Facts

Table 1.1 presents a summary of the CNN Accelerator IP Core.

**Table 1.1. Quick Facts**

| | | |
|---|---|---|
| **IP Requirements** | FPGA Families Supported | ECP5, ECP5-5G |
| **Resource Utilization** | Targeted Device | Full configuration: 85k devices in ECP5 families<br>Reduced configuration: All devices in ECP5 families |
| | Supported User Interface | AXI3, Native interfaces as described in Interface Descriptions section. |
| | Resources | See Table A.1. |
| **Design Tool Support** | Lattice Implementation | Lattice Diamond Design Suite 3.10 |
| | Synthesis | Lattice Synthesis Engine |
| | | Synopsys® Synplify Pro, M-2017.03LR-SP1-1 |
| | Simulation | For a list of supported simulators, see the Lattice Diamond User Guide. |

## 1.2. Features

The key features of the CNN Accelerator IP Core include:

- Support for 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)
- Dynamic support for 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
- Optimization for 3 x 3 2D convolution calculation
- Dynamic support for various 1D convolution from 1 to 72 taps
- Support for max pooling with overlap (For example, kernel 3, stride 2)

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

# 2. Functional Descriptions

## 2.1. Overview

The CNN Accelerator IP Core performs a series of calculations per command sequence that is generated by the Lattice Neural Network Compiler tool. Commands must be written at DRAM address specified by the i_code_base_addr signal which is accessible through AXI BUS. Input data may be read from DRAM at a pre-defined address or directly written through the input data write port. After command code and input data are available, CNN Accelerator IP Core starts calculation at the rising edge of start signal. During calculation, intermediate data and final result may be transferred to DRAM or fed out through the result write port. All operations are fully-programmable by command code.

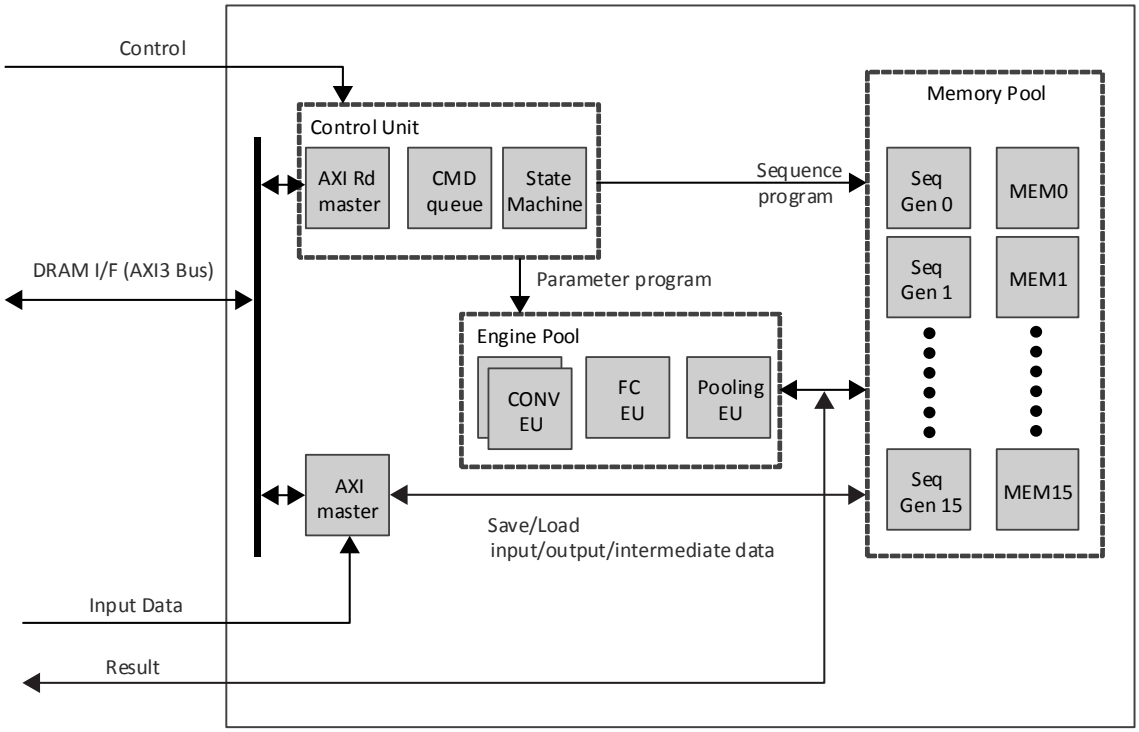


**Figure 2.1. Functional Block Diagram**

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

## 2.2. Interface Descriptions

Figure 2.2 shows the interface diagram for the CNN Accelerator IP Core. The diagram shows all of the available ports for the IP core.

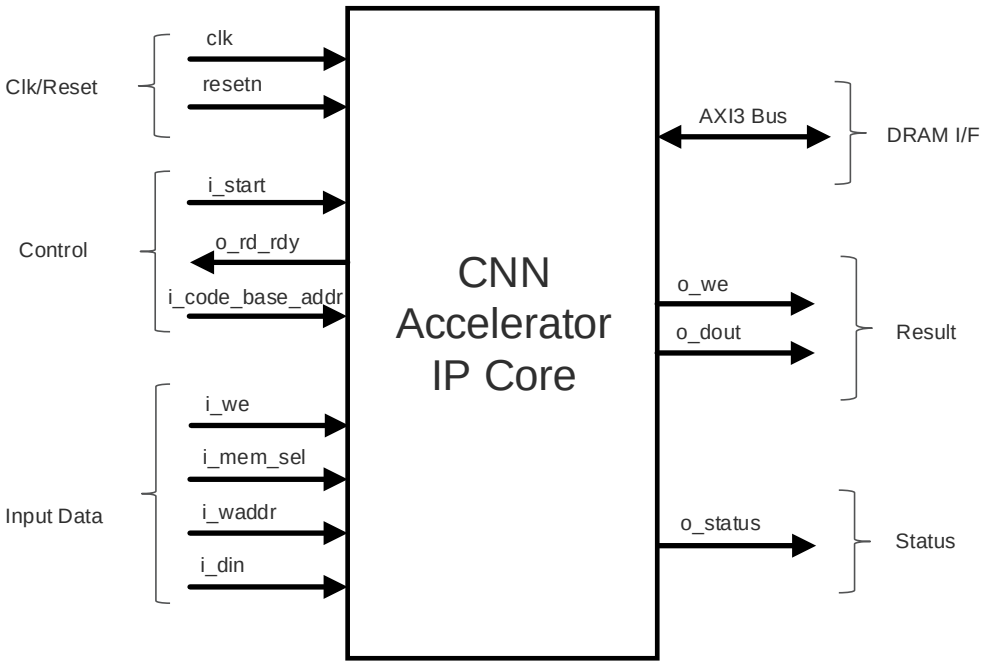


**Figure 2.2. CNN Accelerator IP Core Interface Diagram**

**Table 2.1. CNN Accelerator IP Core Signal Descriptions**

| Pin Name | Direction | Function Description |
|---|---|---|
| **Clock/Reset** | | |
| clk | Input | System clock<br>Frequency can be chosen by trade-off between power and performance. |
| resetn | Input | Active low system reset that is synchronous to clk signal and is asynchronous to aclk signal.<br>[0]: Resets all ports and sets internal registers to their default values.<br>[1]: Reset is NOT active |
| **Control and Status** | | |
| i_code_base_addr[31:0] | Input | This signal specifies the base/start address that is read by CNN Accelerator IP Core to get the command code. External logic should write the command code to this address. This signal must be set before start of operation. |
| i_start | Input | Start execution signal. Level sensitive. Must deassert after o_rd_rdy going 0. |
| o_rd_rdy | Output | Ready signal.<br>[0]: Engine is busy/running.<br>[1]: Engine is idle and ready to get input.<br>External logic should write input data to internal memory only during o_rd_rdy is high. |
| o_status | Output | Debug information<br>[0]: If bit value is 1, engines are running except full connect engine.<br>[1]: If bit value is 1, full connect engine is running.<br>[2]: If bit value is 1, AXI bus transfer is ongoing.<br>[3]: If bit value is 1, during engine is running.<br>[4]: If bit value is 1, command FIFO is reading.<br>[5]: If bit value is 1, during DRAM access.<br>[6]: If bit value is 1, DRAM command is issued. |

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

| Pin Name | Direction | Function Description |
|---|---|---|
| | | [7]: If bit value is 1, engine is waiting for FIFO fill. |
| **Input Data** | | |
| i_mem_sel[3:0] | Input | Memory select signal for routing the write access to one of the 16 internal memories from Memory Pool.<br>[0]: MEM0 is selected for write<br>[1]: MEM1 is selected for write<br>…<br>[15]: MEM15 is selected for write |
| i_we | Input | Write enable signal for internal memory<br>[0]: No write transaction<br>[1]: Write transaction is enabled, i_din and i_waddr are valid |
| i_waddr[12:0] | Input | Write address signal for internal memory |
| i_din[15:0] | Input | Input data signal for internal memory |
| **Result** | | |
| o_we | Output | Write enable of result, indicates result data is valid<br>[0]: Result data is NOT valid<br>[1]: Result data is valid |
| o_dout[15:0] | Output | Result data |
| **DRAM I/F (AXI3 Bus)** | | |
| aclk | Input | AXI3 clock signal. Fully asynchronous from clk. Recommend to use DRAM system clock. |
| A2M_AWID[7:0] | Output | AXI3 write address channel, write address ID signal<br>*Constant output: 0x10* |
| A2M_AWADDR[31:0] | Output | AXI3 write address channel, write address signal |
| A2M_AWLEN[7:0] | Output | AXI3 write address channel, burst length signal |
| A2M_AWSIZE[2:0] | Output | AXI3 write address channel, burst size signal<br>*Constant output: 3'b011 (8 bytes per beat)* |
| A2M_AWBURST[1:0] | Output | AXI3 write address channel, burst type signal<br>*Constant output: 3'b01 (INCR)* |
| A2M_AWLOCK[1:0] | Output | AXI3 write address channel, lock type signal<br>*Constant output: 3'b00 (Normal Access)* |
| A2M_AWCACHE[3:0] | Output | AXI3 write address channel, memory type signal<br>*Constant output: 0x00 (Device Non-bufferable)* |
| A2M_AWPROT[2:0] | Output | AXI3 write address channel, protection type signal<br>*Constant output: 3'b000 (Secure Access)* |
| A2M_AWVALID | Output | AXI3 write address channel, write address valid signal |
| A2M_AWREADY | Input | AXI3 write address channel, write address ready signal |
| A2M_WID[7:0] | Output | AXI3 write data channel, write ID tag signal<br>*Constant output: 0x10* |
| A2M_WDATA[ 63:0] | Output | AXI3 write data channel, write data signal |
| A2M_WSTRB[ 7:0] | Output | AXI3 write data channel, write strobe signal<br>*Constant output: 0xFF* |
| A2M_WLAST | Output | AXI3 write data channel, write last signal |
| A2M_WVALID | Output | AXI3 write data channel, write valid signal |
| A2M_WREADY | Input | AXI3 write data channel, write ready signal |
| A2M_BID[7:0] | Input | AXI3 write response channel, response ID tag signal |
| A2M_BRESP[1:0] | Input | AXI3 write response channel, write response signal |
| A2M_BVALID | Input | AXI3 write response channel, write response valid signal |
| A2M_BREADY | Output | AXI3 write response channel, response ready signal<br>*Constant output: 1'b1 (*AXI3 write response channel is ignored*)* |
| A2M_ARID[7:0] | Output | AXI3 read address channel, read address ID signal |

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

| Pin Name | Direction | Function Description |
|---|---|---|
| | | 0x00: AXI Master for Memory Pool<br>0x10: AXI Rd Master in Control Unit<br>Refer to Figure 2.1 for the AXI Masters |
| A2M_ARADDR[31:0] | Output | AXI3 read address channel, read address signal |
| A2M_ARLEN[7:0] | Output | AXI3 read address channel, burst length signal |
| A2M_ARSIZE[2:0] | Output | AXI3 read address channel, burst size signal<br>*Constant output: 3'b011 (8 bytes per beat)* |
| A2M_ARBURST[1:0] | Output | AXI3 read address channel, burst type signal<br>*Constant output: 3'b01 (INCR)* |
| A2M_ARLOCK[1:0] | Output | AXI3 read address channel, lock type signal<br>*Constant output: 3'b00 (Normal Access)* |
| A2M_ARCACHE[3:0] | Output | AXI3 read address channel, memory type signal<br>*Constant output: 0x01 (Device Bufferable)* |
| A2M_ARPROT[2:0] | Output | AXI3 read address channel, protection type signal<br>*Constant output: 3'b010 (Non-secure Access)* |
| A2M_ARVALID | Output | AXI3 read address channel, read address valid signal |
| A2M_ARREADY | Input | AXI3 read address channel, read address ready signal |
| A2M_RID[7:0] | Input | AXI3 Read data channel, read ID tag signal |
| A2M_RDATA[ 63:0] | Input | AXI3 Read data channel, read data signal |
| A2M_RRESP[1:0] | Input | AXI3 Read data channel, read response signal |
| A2M_RLAST | Input | AXI3 Read data channel, read last signal |
| A2M_RVALID | Input | AXI3 Read data channel, read valid signal |
| A2M_RREADY | Output | AXI3 Read data channel, read ready signal |

### 2.2.1. Control and Status Interface

After reset or when engine is idle, o_rd_rdy is high. During this state, external logic may perform the following:

1. Write input data through Input Data interface
2. Set the start address of command code to i_code_base_addr signal

After the above steps, external logic must assert i_start signal. Engine starts execution when it gets i_start = 1 and o_rd_rdy goes 0. During execution, each bit of o_status indicates activity of sub calculation engine or AXI BUS. After finishing execution, that is by getting finish command, the CNN Accelerator IP Core asserts o_rd_rdy and waits for the next execution. Repeat from asserting i_start. This is shown in Figure 2.3.

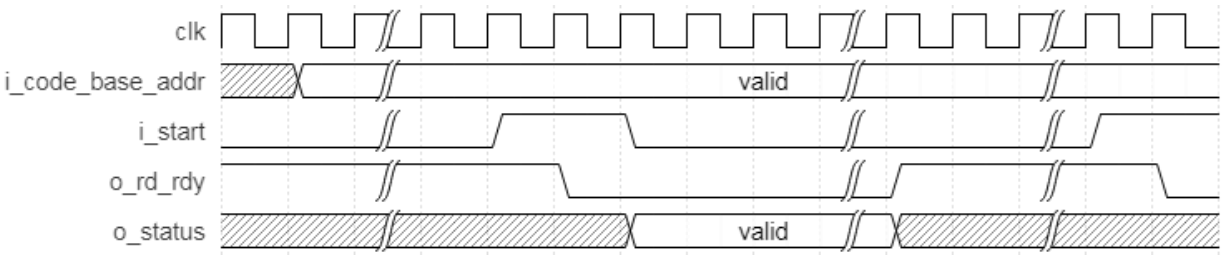


**Figure 2.3. Control and Status Interface Timing Diagram**

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

### 2.2.2. Input Data Interface

Input data can be written to DRAM by external logic. In this case, loading from DRAM for input data must be in command codes. Also, if input data is small enough to hold internal memory, writing to DRAM and reading back to CNN Accelerator IP Core may be waste of cycle time and energy. In that case, external logic can write input data to internal memory of CNN Accelerator IP Core during idle state (o_rd_rdy is high). Interface is based on simple SRAM interface with memory ID as shown in Figure 2.4. Memory ID and address must be matched to command code. Because it is SRAM interface, there is no required order or rule; any random access including memory ID. Overwriting of same address is also accepted.

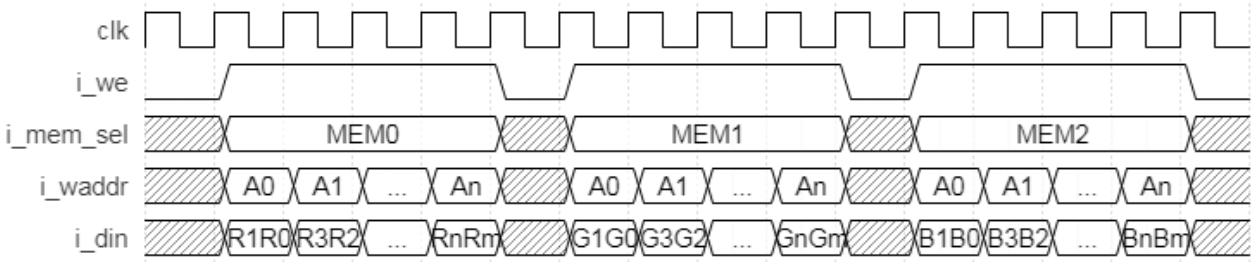


**Figure 2.4. Input Data Interface Timing Diagram**

### 2.2.3. Result Interface

Result, that is, 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 can also simply feed result data to external logic through this interface. Interface consists o_we as valid indicator and o_dout as 16-bit data as shown in Figure 2.5. Usually, it is a single burst series of 16-bit data. Also, it is fully programmable by command code.

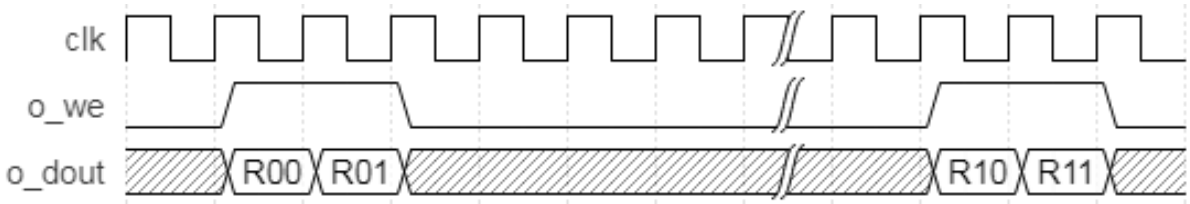


**Figure 2.5. Result Interface Timing Diagram**

### 2.2.4. DRAM Interface

Command code must be written in DRAM before execution of CNN Accelerator IP Core. Input data may be written in DRAM too. During execution of CNN Accelerator IP Core, it reads command code from DRAM and does calculation with internal sub execution engine per command code. Intermediate data may be transferred from/to DRAM per command code.

Please refer to AXI3 Protocol Specification for the timing diagram of DRAM Interface.

## 2.3. Clock Domain

The clk and aclk domains are shown Figure 2.6. The difference in clock is absorbed by Lattice Dual Clocked FIFO IP (FIFO_DC) and is implemented in AXI Rd Master and AXI Master sub-blocks.

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

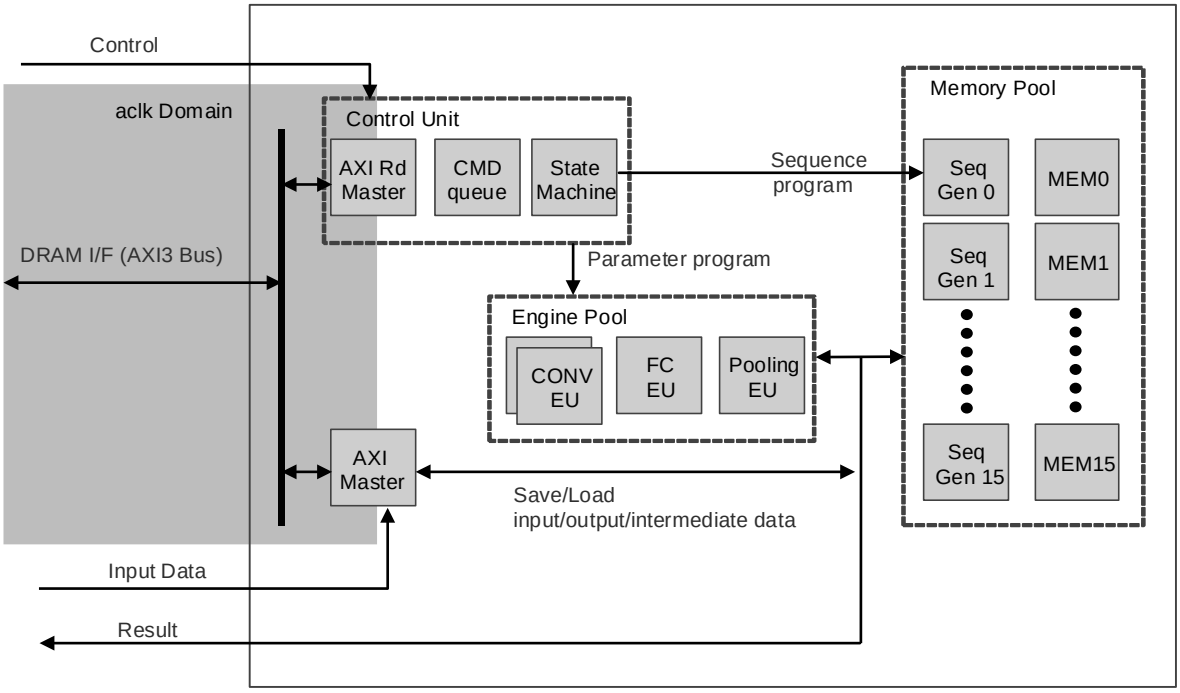


**Figure 2.6. Reset Timing Diagram**

## 2.4. Reset Behavior

When resetn signal asserts, output ports return to logic 0 in the next cycle. When resetn deasserts, output ready signals assert in the next cycle. A timing diagram of reset during AXI3 access is shown as an example in Figure 2.7. Not all AXI3 output signals are shown in this figure. The clk and aclk signals are 50% out-of-phase to show asynchronous relationship. The minimum resetn assert period is 1 cycle of the slower clock between clk and aclk.

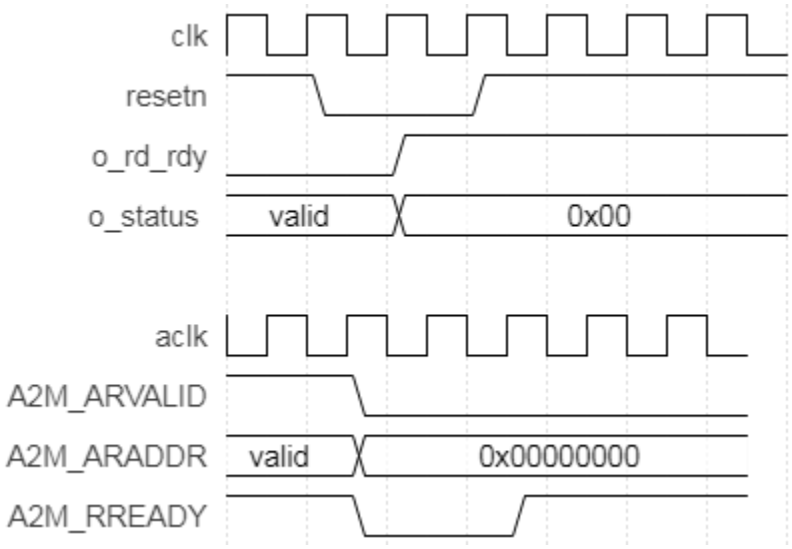


**Figure 2.7. Reset Timing Diagram**

Some AXI3 output signals are constant outputs; these are not affected by reset. Please refer to Table 2.1 for the AXI3 output signals that are constant.

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

## 2.5. Register Description

CNN Accelerator IP Core has no user-configurable register.

## 2.6. Operation Sequence

Operation sequence must be executed in the following steps:

1. Assert Reset.
2. Deassert Reset, i_start must be deasserted.
3. Write command sequence code which is generated by the Lattice Neural Network tool into DRAM starting at the address specified by i_code_base_addr signal.
4. Check whether o_rd_rdy is high or not. o_rd_rdy must be high. Otherwise, go back to step 1.
5. Write input data into DRAM at the proper address, which is decided by command sequence, or directly write into the internal memory block of the CNN Accelerator IP Core through input data ports.
6. Assert i_start and check o_rd_rdy. o_rd_rdy signal should be 0 after asserting i_start.
7. Deassert i_start.
8. Check o_we if code has direct output commands. Collect o_dout while o_we == 1.
9. Check o_rd_rdy and read result from DRAM if command code has storing result to DRAM code after o_rd_rdy going high.
10. Repeat from step 5.

### 2.6.1. Command Format

Command is a sequence of 32-bit data with or without additional parameters or weights as shown in Figure 2.8. It should be loaded at DRAM address specified by i_code_base_addr signal 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).

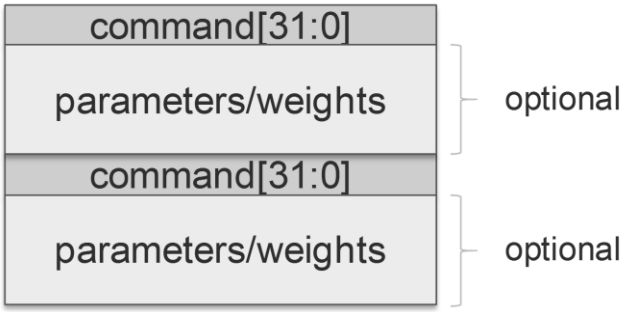


**Figure 2.8. Command format**

### 2.6.2. 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, the 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. Another example is object detection neural network 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 CNN Accelerator IP Core.

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

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

### 2.6.3. 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 outputs 2-beat burst (two consecutive) of 16-bit data, the 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 in little-endian order.

## 2.7. Supported Commands

Command sequences are generated by Lattice Neural Network Compiler Software User Guide (FPGA-UG-02052).

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

# 3. Parameter Settings

The Clarity Designer tool is used to create IP and architectural modules in the Diamond software. You may refer to the IP Generation and Evaluation section on how to generate the IP.

Table 3.1 provides the list of user configurable attributes for the CNN Accelerator IP Core. The attribute values are specified using the IP core Configuration user interface in Clarity Designer as shown in Figure 3.1.

**Table 3.1. Attributes Table**

| Attribute | Selectable Values | Default | Dependency on Other Attributes |
|---|---|---|---|
| Machine Learning Type | CNN, BNN | CNN | — |
| No. of Convolution Engines | 1 - 8 | 8 | — |
| No. of Internal Storage of Blob | 2 - 16 | 16 | — |
| BNN Blob Type | +1/-1, +1/0 | +1/-1 | Valid only when Machine Learning Type = "BNN" |

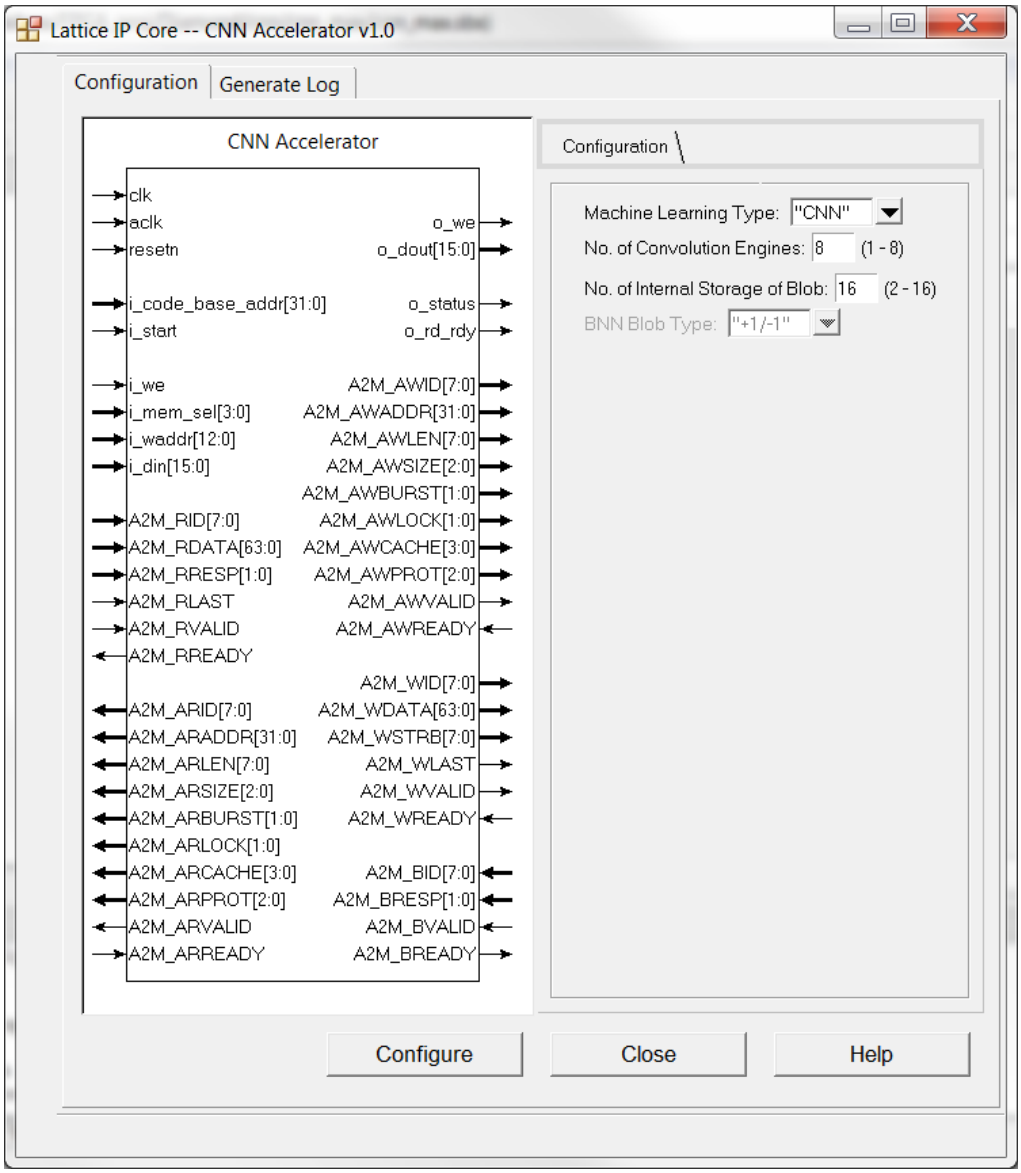


**Figure 3.1. CNN Accelerator IP Core Configuration User Interface**

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

**Table 3.2. Attributes Descriptions**

| Attribute | Description |
|---|---|
| Machine Learning Type | This option allows you to choose between CNN engine and BNN engine.<br>CNN engine always uses 16-bit fixed point weight, while BNN only uses 1-bit weight. Based on required performance, accuracy and available capacity, you should select Machine Learning Type. |
| No. of Convolution Engines | This option allows you to specify the number of convolution engines.<br>For 1D convolution:<br>• 1-9 taps can be performed for each engine<br>• Up to 72 taps by chaining 8 engines<br>For 2D Convolution:<br>• 3 x 3 convolution per cycle per engine<br>You should trade-off between required size and performance. |
| No. of Internal Storage of Blob | Each storage can store up to 16 kB. You should trade-off between required size and performance. In order to fully utilize convolution engines, number of storage must be larger than number of convolution engines. Recommend set 2x of number of convolution engines. |
| BNN Blob Type | Selects the type of binary blob data, either +1/-1 or +1/0. This setting should be matched to the Lattice Neural Network Compiler. |

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

# 4. IP Generation and Evaluation

This section provides information on how to generate the IP using the Lattice Diamond software, and how to run simulation, synthesis and hardware evaluation. For more details on the Lattice Diamond software, you may refer to the Lattice Diamond User Guide and Lattice Diamond Tutorial.

## 4.1. Licensing the IP

An IP core-specific device-specific license string is required to enable full, unrestricted use of the Lattice CNN Accelerator IP Core in a complete, top-level design. You may refer to the instructions on how to obtain licenses for Lattice IP cores at http://www.latticesemi.com/Products/DesignSoftwareAndIP.aspx.

You may download and generate the CNN Accelerator IP Core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license string. The CNN Accelerator IP Core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core which operate in hardware for a limited time (approximately four hours) without requiring an IP license string. See Hardware Evaluation section for further details. However, a license string is required to enable timing simulation to open the design in the Diamond software, and to generate bitstream file that does not include the hardware evaluation timeout limitation.

**Note:** All IP has a license whether in eval mode or full mode. Difference is license string.

## 4.2. Generation and Synthesis

### 4.2.1. Getting Started

The CNN Accelerator IP Core is available for download in the Lattice IP Server using the Diamond Clarity Designer tool. The IP files can be automatically installed using InstallShield® technology in any customer-specified directory. After the IP core is installed, the IP core is listed in the Catalog tab of the Clarity Designer user interface, under DSP category as shown in Figure 4.1.

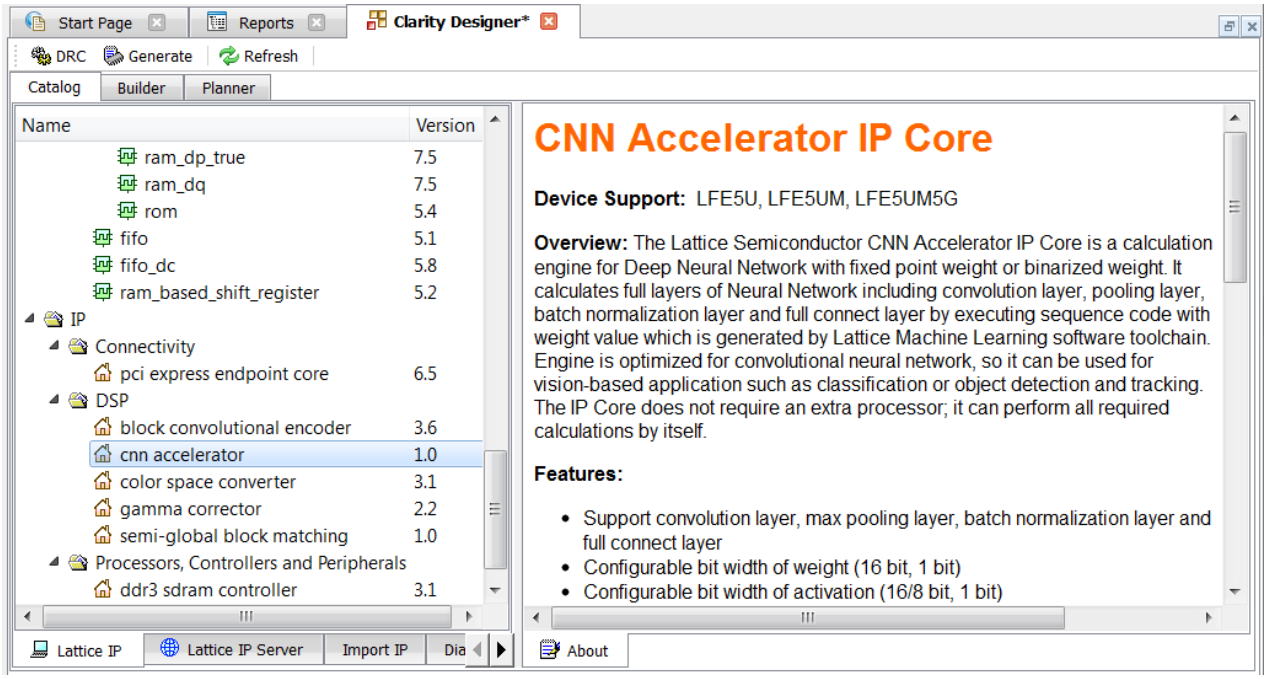


**Figure 4.1. CNN Accelerator IP Core in Clarity Designer Catalog Tab**

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

### 4.2.2. Configuring the IP Core in Clarity

The CNN Accelerator IP Core should be configured and generated before it can be used in the Diamond project. This is done via the CNN Accelerator IP Core configuration user interface (see Figure 3.1); it provides options for setting the desired parameters and invoking the IP core generator.

To configure the CNN Accelerator IP Core:

1. Double-click the CNN Accelerator IP Core in the Catalog tab of the Diamond Clarity tool.
2. Specify the following in the dialog box:
   - Instance Path
   - Instance Name

   **Note:** All generated files are placed under the directory <Instance Path>/<Instance Name>.
3. Enter configuration parameters.
4. Click **Customize.** This closes the dialog box and launches the CNN Accelerator IP Core configuration user interface.
5. Enter the desired parameters in the CNN Accelerator IP Core configuration user interface. Ensure that same parameters are used in the Lattice Neural Network Compiler to be able to generate compatible command sequence code.
6. Click the **Generate** button, and close the user interface to generate the IP core and supporting files.

Table 4.1 provides a list of key files and directories created by the IPexpress tool and how they are used.

**Table 4.1. File List**

| Attribute | Description |
|---|---|
| <Instance Name>.lpc | This file contains the tool options used to recreate or modify the core in the Clarity tool. |
| <Instance Name>.ngo | This file provides the synthesized IP core. |
| <Instance Name>_bb.v | This file provides the synthesis black box for the user's synthesis. |
| <Instance Name>_inst.v | This file provides an instance template for the IP core. |
| <Instance Name>_top.v | This file provides an example RTL top file that instantiates the IP core. |
| beh_rtl.v* | This file provides cycle-accurate simulation model for the IP core.<br>The top level module name in this file is <Instance Name>. |
| generate_core.tcl | This file is created when the user interface **Generate** button is pushed.<br>This file may be run from command line. |
| <Instance Name>_generate.log | This is the synthesis and map log file. |
| <Instance Name>_gen.log | This is the IP Core generation log file. |

***Note:** This file is located in the following path:
<Instance Path>/<Instance Name>/ml_engine_ecp5_eval/<Instance Name>/src/beh_rtl/<device>

### 4.2.3. Instantiating the IP Core

The generated CNN Accelerator IP Core package includes black-box (<Instance Name>_bb.v) and instance (<Instance Name>_inst.v) templates that can be used to instantiate the core in a top-level design. A sample of RTL top-level reference source file (<Instance Name>_top.v) that can be used as an instantiation template for the IP core is also provided. You may also use this top-level reference as the starting template for the top-level for their complete design.

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

## 4.3. Running Functional Simulation

The CNN Accelerator IP Core does NOT contain a sample test bench for performing simple simulation test. However, a cycle-accurate simulation model (beh_rtl.v) is provided as shown in Table 4.1. This may be instantiated in a test bench. To successfully elaborate the simulation model:

1. Instantiate GSR and PUR in testbench top RTL file.
   The following codes may be copied and paste to testbench top RTL file:
   ```
   PUR PUR_INST(<active_low_reset_signal>);
   GSR GSR_INST(<active_low_reset_signal>);
   ```
2. Compile the simulation primitives of selected device. These are located in the following path:
   <Diamond Install Path>/cae_library/simulation/verilog/<device>

## 4.4. Hardware Evaluation

The CNN Accelerator IP Core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase of an IP license. It may also be used to evaluate the core in hardware in user-defined designs. Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

# 5. Ordering Part Number

The Ordering Part Numbers (OPN) for CNN Accelerator IP Core targeting ECP5 and ECP5-5G FPGA devices are the following:

- CNN-ACCEL-E5-U – Project License
- CNN-ACCEL-E5-UT – Site License

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

# References

For more information on the FPGA device, visit: 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.

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

# Appendix A. Resource Utilization

Table A.1 shows configuration and resource utilization for the ECP5UM using Lattice Diamond 3.10.0.111.0. The following settings are used in generating this data:

- Synthesis Tool: Synplify Pro
- Device Part No.: LFE5UM-85F-8BG756I
- BNN Blob Type: +1/0 (For Machine Learning Type: BNN)

**Table A.1. Performance and Resource Utilization[1]**

| No. of Convolution Engines | No. of Internal Storage of Blob[2] | Register | LUTs | Slices | Block RAMs | clk Fmax[3] (MHz) |
|---|---|---|---|---|---|---|
| **Machine Learning Type: CNN** | | | | | | |
| 1 | 2 | 3607 | 4829 | 3601 | 25 | 150.218 |
| 4 | 8 | 9064 | 13661 | 9890 | 76 | 144.509 |
| 8 | 16 | 16482 | 25889 | 18456 | 144 | 121.892 |
| **Machine Learning Type: BNN** | | | | | | |
| 1 | 2 | 3472 | 4871 | 3524 | 25 | 119.26 |
| 4 | 8 | 9764 | 15294 | 10808 | 76 | 117.426 |
| 8 | 16 | 17898 | 28600 | 20094 | 144 | 109.242 |

**Notes:**

1. Performance may vary when using a different software version or targeting a different device density or speed grade.
2. It is recommended to use No. of Internal Storage of Blob = 2 * No. of Convolution Engines.
3. Fmax is generated when the FPGA design only contains the CNN Accelerator IP Core. These values may be reduced when user logic is added to the FPGA design.

For more information on Lattice Diamond Software, visit the Lattice web site at www.latticesemi.com/Products/DesignSoftwareAndIP.

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

# Revision History

**Revision 1.1, September 2018**

| Section | Change Summary |
|---|---|
| Functional Description | Updated Overview section. Updated status signal description in Table 2.1. CNN Accelerator IP Core Signal Descriptions. |
| Appendix A. Resource Utilization | Updated Table A.1. Performance and Resource Utilization[1] per result of CNN Accelerator IP Core v1.1. |
| Revision History | Updated revision history table to new template. |

**Revision 1.0, May 2018**

| Section | Change Summary |
|---|---|
| All | Initial release |

© 2018 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

7th Floor, 111 SW 5th Avenue
Portland, OR 97204, USA
T 503.268.8000
www.latticesemi.com

# OCEAN CHIPS
## Океан Электроники
## Поставка электронных компонентов

Компания «Океан Электроники» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Поставка оригинальных импортных электронных компонентов напрямую с производств Америки, Европы и Азии, а так же с крупнейших складов мира;
- Широкая линейка поставок активных и пассивных импортных электронных компонентов (более 30 млн. наименований);
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Помощь Конструкторского Отдела и консультации квалифицированных инженеров;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Поставка электронных компонентов под контролем ВП;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- При необходимости вся продукция военного и аэрокосмического назначения проходит испытания и сертификацию в лаборатории (по согласованию с заказчиком);
- Поставка специализированных компонентов военного и аэрокосмического уровня качества (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Actel, Aeroflex, Peregrine, VPT, Syfer, Eurofarad, Texas Instruments, MS Kennedy, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits, General Dynamics и др.);

Компания «Океан Электроники» является официальным дистрибьютором и эксклюзивным представителем в России одного из крупнейших производителей разъемов военного и аэрокосмического назначения «**JONHON**», а так же официальным дистрибьютором и эксклюзивным представителем в России производителя высокотехнологичных и надежных решений для передачи СВЧ сигналов «**FORSTAR**».

AVIC JONHON

**«JONHON»** (основан в 1970 г.)

Разъемы специального, военного и аэрокосмического назначения:

(Применяются в военной, авиационной, аэрокосмической, морской, железнодорожной, горно- и нефтедобывающей отраслях промышленности)

**«FORSTAR»** (основан в 1998 г.)

ВЧ соединители, коаксиальные кабели, кабельные сборки и микроволновые компоненты:

(Применяются в телекоммуникациях гражданского и специального назначения, в средствах связи, РЛС, а так же военной, авиационной и аэрокосмической отраслях промышленности).

FORSTAR®

Телефон: 8 (812) 309-75-97 (многоканальный)
Факс: 8 (812) 320-03-32
Электронная почта: ocean@oceanchips.ru
Web: http://oceanchips.ru/
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, д. 2, корп. 4, лит. А