Advanced CNN Accelerator IP

User Guide

FPGA-IPUG-02224-2.0

December 2023

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, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. 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. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. 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.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents....3
Acronyms in This Document....5
1. Introduction....6
1.1. Quick Facts....6
1.2. Features....6
2. Functional Descriptions....7
2.1. Overview....7
2.2. Interface Descriptions....8
2.2.1. Control and Status Interface....11
2.2.2. Input Data Interface....12
2.2.3. Result and Debug Interface....12
2.2.4. DRAM Interface....12
2.3. Attribute Summary....13
2.4. Clock Domain....14
2.5. Reset Behavior....14
2.6. Register Description....15
2.7. Operation Sequence....15
2.7.1. Command Format....15
2.7.2. Input Data Format....16
2.7.3. Output Data Format....16
2.8. Supported Commands....16
3. Core Generation, Simulation, and Validation....17
3.1. Licensing the IP....17
3.2. Generation and Synthesis....17
3.3. Running Functional Simulation....19
3.4. Hardware Evaluation....19
4. Hardware Validation....20
5. Ordering Part Number....21
Appendix A. Resource Utilization....22
References....23
Technical Support Assistance....24
Revision History....25

Figures

Figure 2.1. Functional Block Diagram of Advanced CNN Accelerator 7
Figure 2.2. Order of Layers for Advanced CNN Accelerator 8
Figure 2.3. Advanced CNN Accelerator IP Core Interface Diagram 8
Figure 2.4. Control and Status Interface Timing Diagram 11
Figure 2.5. General Purpose Output Sample Application 11
Figure 2.6. Result and Debug Interface Timing Diagram 12
Figure 2.7. Reset Timing Diagram 15
Figure 2.8. Command Format 16
Figure 3.1. Module/IP Block Wizard 17
Figure 3.2. Configure User Interface of Advanced CNN Accelerator IP Core 18
Figure 3.3.Check Generating Result 18

Tables

Table 1.1. Quick Facts 6
Table 2.1. Advanced CNN Accelerator IP Core Signal Descriptions 9
Table 2.2. Attributes Table 13
Table 3.1. Generated File List 19
Table A.1. Performance and Resource Utilization[1] 22

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
ALU	Arithmetic Logic Unit
CNN	Convolutional Neural Network
DRAM	Dynamic Random Access Memory
EBR	Embedded Block RAM
FIFO	First In, First Out
FPGA	Field-Programmable Gate Array
LMMI	Lattice Memory Mapped Interface
ML	Machine Learning
OPN	Ordering Part Number
SPD	Scratch Pad

1. Introduction

The Lattice Semiconductor Advanced CNN Accelerator IP Core is a calculation engine for Deep Neural Network with fixed point weight. It calculates full layers of Neural Network including convolution layer, pooling layer, batch normalization layer, and fully connected layer by executing a sequence of firmware code with weight value, which is generated by Lattice SensAI™ 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 and can be targeted for CertusPro™-NX and Lattice Avant™-AT-E devices. It is implemented using the Lattice Radiant™ software Place and Route tool integrated with the Synplify Pro® synthesis tool.

1.1. Quick Facts

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

Table 1.1. Quick Facts

IP Requirements	Supported FPGA Family	CertusPro-NX, and Avant-AT-E
Resource Utilization	Targeted Devices	LFCPNX-100, LAV-AT-E70
	Supported User Interface	AXI4, LMMI (Lattice Memory Mapped Interface) Native interfaces as described in Interface Descriptions section.
	Resources	See Table A.1.
Design Tool Support	Lattice Implementation	IP Core v1.0.0 – Lattice Radiant Software 2022.1 or later IP Core v2.0.0 – Lattice Radiant Software 2022.1 or later
	Synthesis	Synopsys® Synplify Pro for Lattice
	Simulation	For the list of supported simulators, see the Lattice Radiant Software user guide.

1.2. Features

The key features of the Advanced CNN Accelerator IP Core include:

- Supports higher throughput 64-bit data path engine for Avant-AT-E, and 32-bit data path engine for Avant-AT-E and CertusPro-NX devices.
- Supports convolution layer, max pooling layer, global average pooling layer, batch normalization layer, and full connect layer.
- Supports AXI4 for external memory interface.
- Supports Vector ALU (Arithmetic Logic Unit) for enhanced pixelwise operations, and accelerated pre/post ML image processing algorithms.
- Provides configurable bit width of activation (16/8-bit).
- Provides configurable number of memory blocks for tradeoff between resource and performance.
- Supports 3 × 3, 1 x 1, 5 x 5, and 7 x 7 2D convolution calculation for 8-bit input.
- Supports 3 x 3, and 5 x 5 convolution with 16-bit input for higher output accuracy.
- Supports 2 × 2 stride=2 max pooling/unpooling with max argument.
- Supports parameterizable kernel size, stride=1, 1D max pooling.
- Provides configurable LMMI interface access mode: byte, half-word (16-bit) or word (32-bit).
- Supports configurable maximum burst length (32, 256) for AXI4 Interface.
- Supports up to 4 convolution engines to improve performance.
- Supports usage of four parallel ports for 1 × 1 convolution.
- Provides general purpose output signal for controlling external logic through command code.
- Supports lookup table-based activation functions after convolution, and fully connected layers.
- Supports new Filter module in Vector ALU for specialized kernels like min/max/median.

2. Functional Descriptions

2.1. Overview

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

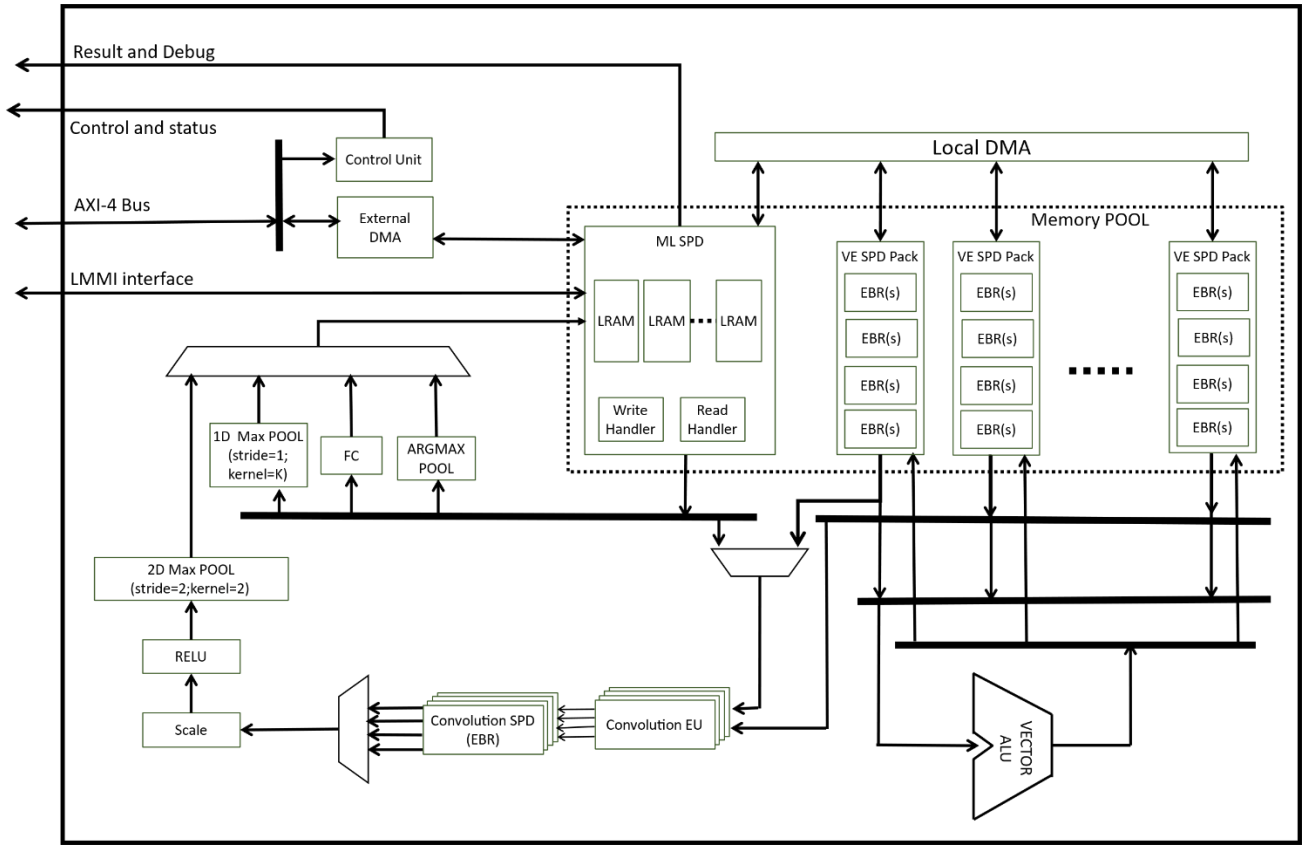

Figure 2.1. Functional Block Diagram of Advanced CNN Accelerator

In Figure 2.1, the internal storage memory pool consists of the Machine Learning (ML) SPD Scratch Pad (SPD) made of a configurable number of LRAMs and the VE SPD packs made of EBRs. Each VE SPD pack consists of fixed four VE SPDs, and each VE SPD consists of a configurable number of EBRs. The number of VE SPD packs can be configured as well. The input data must be present in ML SPD as the Activation Data Storage. It can feed data to the Convolution Engines. Fully Connected (FC) Engine, Argmax pool engine, or 1D Maxpool engine. The result of Fully Connected Engine, Argmax Pool engine or 1D Maxpool engine goes back to ML SPD. For the other path, the result of Convolution Engine is buffered to Convolution Scratch Storage, termed CONV SPD, and is transmitted to Scaler, which performs batch normalization. The Scaler can be bypassed by setting unity value, which is multiply by 1 and add by 0. That result goes to RELU and then to 2D Maxpool EU. Both RELU and 2D Maxpool EU has bypass option. Lastly, the result of 2-D Maxpool EU goes back to ML SPD.

For 3 × 3, 5 × 5 and 7 × 7 convolutions, data must reside in ML SPD. The VE SPD packs allow high bandwidth access for high throughput computations. Data must be moved into VE SPD from ML SPD using Local DMA transfer commands.

The Vector ALU supports high-performance pixel-wise arithmetic operations like elementwise addition. The Vector ALU also supports acceleration of image processing algorithms and other pre/post processing ML algorithms. The data must reside in VE SPD for use with Vector ALU.

The order of layers for CNN configuration is shown in Figure 2.2. Refer to this when designing the network model. The layers inside the parenthesis have bypass option.

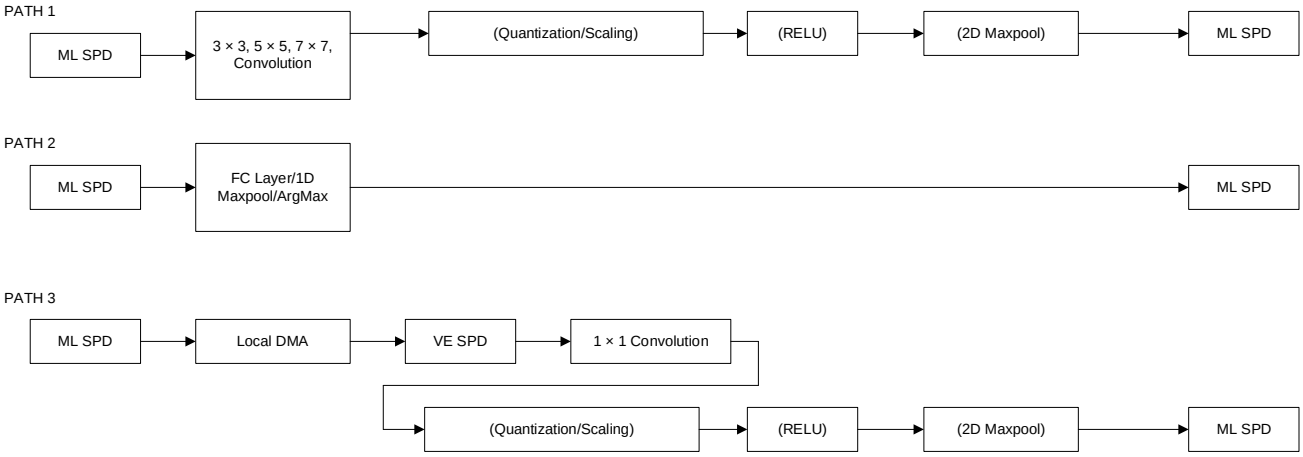

Figure 2.2. Order of Layers for Advanced CNN Accelerator

2.2. Interface Descriptions

Figure 2.3 shows the interface diagram for the Advanced CNN Accelerator IP Core. The diagram shows all of the available ports for the IP core.

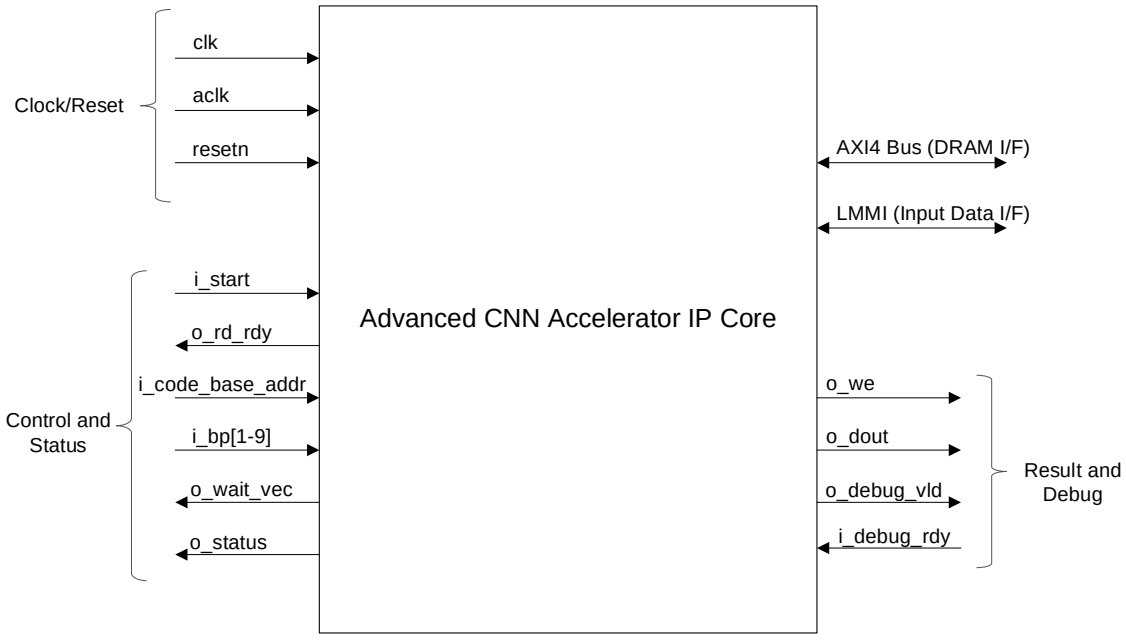

Figure 2.3. Advanced CNN Accelerator IP Core Interface Diagram

Table 2.1. Advanced CNN Accelerator IP Core Signal Descriptions

Pin Name	Direction	Function Description
Clock/Reset		
clk	Input	System clock 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. 0 – Resets all ports and sets internal registers to the default values. 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 Advanced 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_bp[31:0]	Input	This signal specifies external memory base pointer address for DMA read/write which enables efficient management of available external memory, and sharing of data in multiple ML engine scenario.
i_start	Input	Start execution signal. Level sensitive. Must deassert after o_rd_rdy going 0.
o_rd_rdy	Output	Ready signal 0 – Engine is busy/running. 1 – Engine is idle and ready to get input. External logic should write input data to internal memory only during o_rd_rdy is high.
o_status[10:0]	Output	Debug information [0] – If bit value is 1, convolution engine is running. [1] – If bit value is 1, full connect engine is running. [2] – If bit value is 1, Local DMA is moving data between ML SPD and VE SPD. [3] – If bit value is 1, ML IP is executing the current firmware instructions. [4] – If bit value is 1, command FIFO is reading. [5] – If bit value is 1, external memory DMA transfer is ongoing. [6] – If bit value is 1, DMA command is issued. [7] – If bit value is 1, engine is waiting for command FIFO to fill. [8] – If bit value is 1, engine is waiting for internal operations to complete. [9] – If bit value is 1, VE SPD pool is running. [10] – If bit value is 1, convolution result is being transferred to ML SPD through scale/RELU/pool.
o_gpo[31:0]	Output	General Purpose Output signal. Use for communication from firmware to outside block such as informing the type of current output, or informing the firmware version.
Input Data (LMMI)		
i_lmmi_request	Input	Start transaction
I_lmmi_wr_rdn	Input	Write = HIGH, Read = LOW
i_lmmi_offset[20:0]	Input	Address offset for accessing the internal LRAMs in Activation Data Storage. **Note:** Full 21 bits are used in 64-bit mode, and lower 20 bits of offset ([19:0]) are used in 32-bit mode.
I_lmmi_wdata[31:0]	Input	Write data
o_lmmi_ready	Output	Target Ready signal. When o_lmmi_ready and i_lmmi_request are both HIGH, it indicates write transaction is completed and address phase of ready transition is completed.
o_lmmi_rdata_valid	Output	Indicates read transaction is complete and o_lmmi_rdata contains valid data.
o_lmmi_rdata[31:0]	Output	Read data
Result and Debug		
o_we	Output	Write enable of result, indicates result data is valid. 0 – Result data is NOT valid. 1 – Result data is valid
o_dout[15:0]	Output	IF o_we is asserted, o_dout[15:0] contains result data.

Pin Name	Direction	Function Description
		IF o_debug_vld is asserted, o_dout[15:0] contains debug data.
_debug_rdy	Input	Ready signal for one burst read of debug data 0 – External logic is not ready to receive 1 burst of debug data. 1 – External logic is ready to receive 1 burst of debug data. The recommendation is to connect to 1 if debug feature is not used so that the operation is not halted when the IP Core sends debug data.
o_debug_vld	Output	Indicates that o_dout[15:0] is not result data, but debug data. 0 – Debug data is NOT valid. 1 – Debug data is valid
DRAM I/F (AXI4 Bus)		
aclk	Input	AXI4 clock signal. Fully asynchronous from clk. Recommend using DRAM system clock.
A2M_AWID[7:0]	Output	AXI4 write address channel, write address ID signal
A2M_AWADDR[31:0]	Output	AXI4 write address channel, write address signal
A2M_AWREGION[3:0]	Output	AXI4 write address channel, region identifier signal
A2M_AWLEN[7:0]	Output	AXI4 write address channel, burst length signal
A2M_AWSIZE[2:0]	Output	AXI4 write address channel, burst size signal
A2M_AWBURST[1:0]	Output	AXI4 write address channel, burst type signal
A2M_AWLOCK	Output	AXI4 write address channel, lock type signal
A2M_AWCACHE[3:0]	Output	AXI4 write address channel, memory type signal
A2M_AWPROT[2:0]	Output	AXI4 write address channel, protection type signal
A2M_AWQOS[3:0]	Output	AXI4 write address channel, quality of service signal
A2M_AWVALID	Output	AXI4 write address channel, write address valid signal
A2M_AWREADY	Input	AXI4 write address channel, write address ready signal
A2M_wid[7:0]	Output	AXI4 write data channel, write ID tag signal
A2M_WDATA[63:0]	Output	AXI4 write data channel, write data signal
A2M_WSTRB[7:0]	Output	AXI4 write data channel, write strobe signal
A2M_WLAST	Output	AXI4 write data channel, write last signal
A2M_WVALID	Output	AXI4 write data channel, write valid signal
A2M_WREADY	Input	AXI4 write data channel, write ready signal
A2M_BID[7:0]	Input	AXI4 write response channel, response ID tag signal
A2M_BRESP[1:0]	Input	AXI4 write response channel, write response signal
A2M_BVALID	Input	AXI4 write response channel, write response valid signal
A2M_BREADY	Output	AXI4 write response channel, response ready signal
A2M_ARID[7:0]	Output	AXI4 read address channel, read address ID signal
A2M_ARADDR[31:0]	Output	AXI4 read address channel, read address signal
A2M_ARREGION[3:0]	Output	AXI4 read address channel, region identifier signal
A2M_ARLEN[7:0]	Output	AXI4 read address channel, burst length signal
A2M_ARSIZE[2:0]	Output	AXI4 read address channel, burst size signal
A2M_ARBURST[1:0]	Output	AXI4 read address channel, burst type signal
A2M_ARLOCK	Output	AXI4 read address channel, lock type signal
A2M_ARCACHE[3:0]	Output	AXI4 read address channel, memory type signal
A2M_ARPROT[2:0]	Output	AXI4 read address channel, protection type signal
A2M_ARQOS[3:0]	Output	AXI4 read address channel, quality of service signal
A2M_ARVALID	Output	AXI4 read address channel, read address valid signal
A2M_ARREADY	Input	AXI4 read address channel, read address ready signal
A2M_RID[7:0]	Input	AXI4 Read data channel, read ID tag signal
A2M_RDATA[63:0]	Input	AXI4 Read data channel, read data signal
A2M_RRESP[1:0]	Input	AXI4 Read data channel, read response signal

Pin Name	Direction	Function Description
A2M_RLAST	Input	AXI4 Read data channel, read last signal
A2M_RVALID	Input	AXI4 Read data channel, read valid signal
A2M_RREADY	Output	AXI4 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 .

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 calculation engine or AXI BUS. After finishing execution that is by getting the finish command, the Advanced CNN Accelerator IP Core asserts o_rd_rdy and waits for the next execution. Repeat from asserting i_start as shown in Figure 2.4.

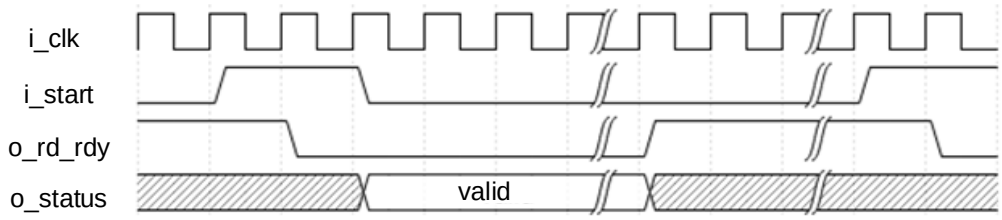

Figure 2.4. Control and Status Interface Timing Diagram

2.2.1.1. General Purpose Output

The general-purpose output signal, o_gpo signal is controlled by the Lattice sensAI Neural Network Compiler software. One possible application of this signal is shown in Figure 2.5.In this example, different post processing operation needs to be performed on certain outputs. The o_gpo signal may be used to select which post process to perform.

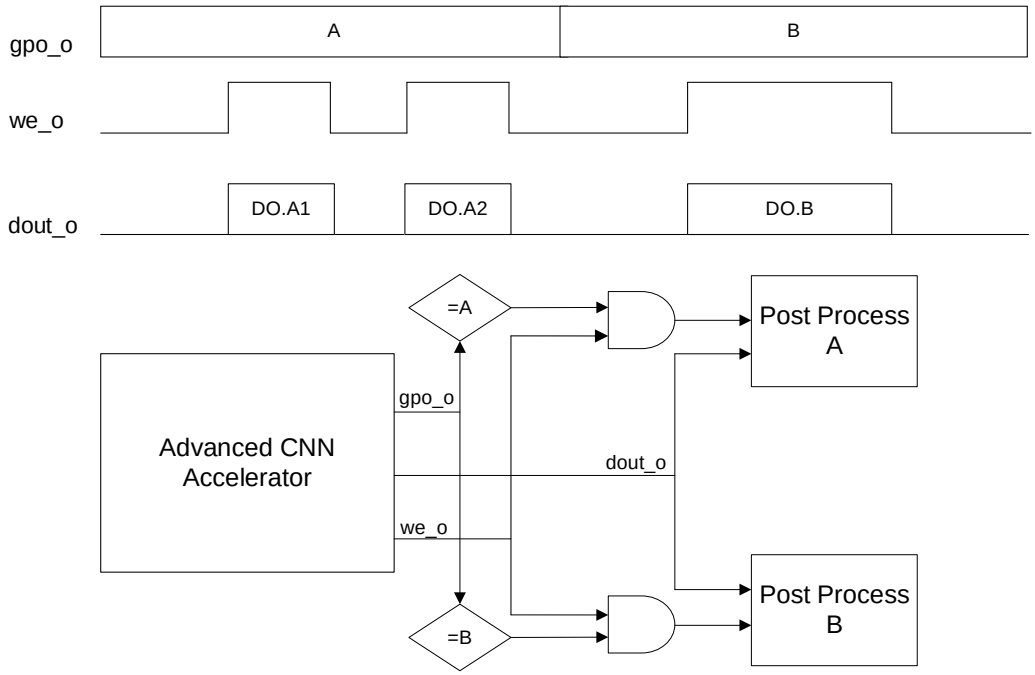

Figure 2.5. General Purpose Output Sample Application

2.2.2. Input Data Interface

Input data can be written to the DRAM by external logic. In this case, loading from the DRAM for input data must be in command codes. This is done by enabling the *Store Input* option and setting the target DRAM address in the Project Implementation window of the Lattice sensAI Neural Network Compiler software. In this case, the external logic needs to store the image data to target the DRAM address.
In addition, if input data is small enough to hold in internal memory, writing to the DRAM and reading back to the Advanced CNN Accelerator IP Core may be a waste of cycle time and energy. In that case, the *Store Input* option should be disabled and the external logic should write input data to the internal memory of the Advanced CNN Accelerator IP Core during idle state (o_lmmi_ready is high). The Input Data Interface is LMMI. Refer to Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039) for more information on LMMI and its timing diagram. The read latency from i_lmmi_request to o_lmmi_ready is four clock cycles. Memory ID and memory address are mapped to the i_lmmi_offset as described in Table 2.1. They should be matched to command code. The input data interface of this IP core is 32-bit wide, and can be configured for 8-bit, 16-bit, or 32-bit write using the “LMMI Write mode” attribute setting. Similarly, this also applies for *LMMI Read mode*.

Depending on the access mode setting, the LMMI offset is byte addressable (in BYTE mode), half-word addressable (in HWORD mode) and word addressable (in WORD mode). For example in BYTE mode, the LMMI offset 0 means 0^{th} byte, offset 1 means first byte and so on. In WORD mode, offset 0 means 0^{th} 32-bit word, offset 1 means first 32-bit word and so on. There is no required order or rule for writing the input data to the internal memory; any random access including memory ID is acceptable. Overwriting of same address is also accepted. However, user must ensure that the input data format is satisfied before asserting i_start signal. Refer to Input Data Format section for details.

2.2.3. Result and Debug Interface

The result, that is, the final blob data of neural network and the debug information are outputted to external logic through this Result and Debug Interface. When o_we signal is asserted, the o_dout signal contains the valid final Blob data. The external logic must have enough buffer to sample the result data whenever o_we asserts. This is usually a single burst series of 16-bit data based on the command code. On the other hand, when o_debug_vld is asserted, the o_dout signal contains valid debug information. The IP Core keeps transmitting available debug information when i_debug_rdy is asserted. The IP Core halts until all debug information are transmitted. Thus, when debug information is not used, user must set i_debug_rdy to a fixed 1’b1 value so that it can flush-out debug information to proceed operation. The o_we and o_debug_vld signals never assert at the same time. Figure 2.6 shows an example timing diagram of Result and Debug interface with i_debug_rdy fixed to 1’b1.

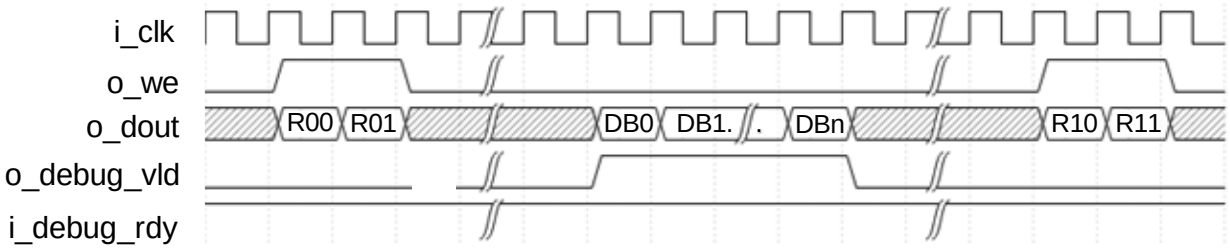

Figure 2.6. Result and Debug Interface Timing Diagram

2.2.4. DRAM Interface

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

Refer to AXI4 Protocol Specification for the timing diagram of DRAM Interface.

2.3. Attribute Summary

The configurable attributes of the Advanced CNN Accelerator IP Core are shown and described in Table 2.2 . The attributes can be configured through the IP Catalog's Module/IP Block wizard of the Lattice Radiant software.

Table 2.2. Attributes Table

Attribute	IP Parameter	Selectable Values	Default	Description
Number of LRAMs	LRAM_NUM	[0 -16]	7	Number of LRAMs in the ML SPD.
Number of VE SPD Packs	SPD_NUM	[0 -16]	8	1 VE SPD Pack = 4 VE SPDs. Number of EBRs in a VE SPD is the attribute below.
Max AXI4 external memory DMA Burst	MAX_BURST	[0-255]	31	Max burst limit on AXI4 bus during DMA transactions with external memory.
LMMI read mode	LMMI_RMODE	BYTE, HWORD, WORD	BYTE	Access width mode for LMMI read interface (reading data out of the IP). BYTE: byte mode ; HWORD: 16-bit half word mode ; WORD: 32-bit word mode
LMMI Write mode	LMMI_WMODE	BYTE, HWORD, WORD	BYTE	Access width mode for LMMI write interface (writing data into the IP). BYTE: byte mode ; HWORD: 16 bit half word mode ; WORD: 32 bit word mode
No. of Convolution engines	CONV_NUM	[0-4]	1	Number of parallel convolution engines. Higher compute throughput can be achieved with more number of engines, at the expense of higher utilization of DSP resources.
Enable 4 parallel ports for 1x1 convolution	CONV_M4	Checked, Unchecked	Unchecked	Enables four parallel ports internally for high bandwidth data accesses for 1 × 1 convolutions.
Enable Vector ALU	EN_VE	Checked, Unchecked	Checked	Enables the Vector Engine for pixelwise ALU operations.
Enable 5 × 5 convolution	EN_CONV5	Checked, Unchecked	Checked	Enables 5 × 5 convolutions inside the Conv EU.
Enable 7 × 7 convolution	EN_CONV7	Checked, Unchecked	Checked	Enables 7 × 7 convolutions inside the Conv EU.
Enable Argmax pool	EN_ARGMAX_POOL	Checked, Unchecked	Checked	Enables the Argmax pool compute module.
Enable Maxpool stride=1 module	EN_MAXPOOL_S1	Checked, Unchecked	Checked	Enables the 1-D stride=1 maxpool compute engine.
Engine Datapath	VE_TYPE	32 bit/64 bit	32 bit	Datapath mode for Avant-AT-E devices. Only supports 32-bit engine for CertusPro-NX devices currently.
Enable Filter module in Vector ALU	EN_VE_FILT	Checked, Unchecked	Unchecked	Enables the Filter module in Vector ALU to support specialized kernels like min/max/median.
Enable activation function in scale module	LUT_SCALE_ACT_FUNC	Checked, Unchecked	Unchecked	Enables lookup table based activation function in scale module.
Lookup table address width(For scale)	LUT_SCALE_ACT_ADDR_WIDTH	integer	12	Number of bits for lookup table address. Address width = $clog2(Lookup table depth)

Attribute	IP Parameter	Selectable Values	Default	Description
Lookup table data width(For scale)	LUT_SCALE_ACT_WIDTH	[8 or 16]	8	Number of data bits for each lookup table entry.
Scale Lookup table init file relative path(For scale)	LUT_SCALE_ACT_INIT_FILE	String	“”	Path to lookup table initialization file in hex format. Each line should have data-width-bits worth of hex value. Number of lines in file = Lookup table depth. **Note:** The file path should be supplied as a string relative to the Radiant project directory.
Enable MSB clipping(For scale)	LUT_SCALE_ACT_CLIP_EN	Checked, Unchecked	Unchecked	Clip MSB for LUT addressing. Useful for Sigmoid-like functions to not store positive and negative saturated curves in memory, and clamp them to fixed values. It uses the available LUT memory locations to store nonlinear portions of curves.
Enable activation function in FC module	LUT_FC_ACT_FUNC	Checked, Unchecked	Unchecked	Enables lookup table based activation function in Fully connected layer.
Lookup table address width(for FC)	LUT_FC_ACT_ADDR_WIDTH	integer	12	Number of bits for lookup table address. Address width = $clog2(Lookup table depth)
Lookup table data width(for FC)	LUT_FC_ACT_WIDTH	Fixed at 8	8	Number of data bits for each lookup table entry
FC Lookup table init file relative path	LUT_FC_ACT_INIT_FILE	String	“”	Path to lookup table initialization file in hex format. Each line should have data-width-bits worth of hex value. Number of lines in file = Lookup table depth. **Note:** The file path should be supplied as a string relative to the Radiant project directory.
Enable MSB clipping(For FC)	LUT_FC_ACT_CLIP_EN	Checked, Unchecked	Unchecked	Clip MSB for LUT addressing. Useful for Sigmoid-like functions to not store positive and negative saturated curves in memory, and clamp them to fixed values. It uses the available LUT memory locations to store nonlinear portions of curves.

2.4. Clock Domain

The IP uses two clocks clk and aclk for processing and AXI interface respectively. The difference in clock is absorbed by Lattice Dual Clocked FIFO IP (FIFO_DC) and is implemented in the DMA sub block.

2.5. 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 AXI4 access is shown as an example in Figure 2.7. Only a few 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 one cycle of the slower clock between clk and aclk.

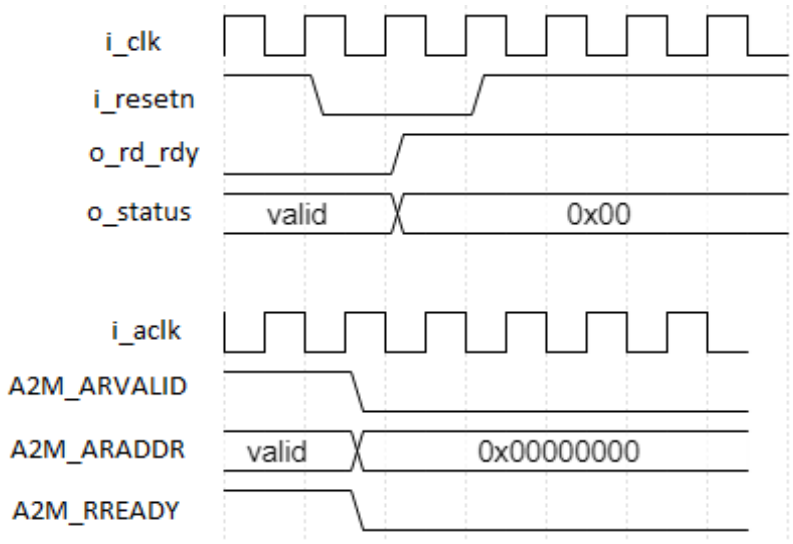

Figure 2.7. Reset Timing Diagram

2.6. Register Description

Advanced CNN Accelerator IP Core has no user-configurable register.

2.7. Operation Sequence

Operation sequence must be executed in the following steps. i_debug_rdy is tied to 1 in this example.

1. Assert Reset.
2. Deassert Reset. i_start must be deasserted.
3. Write command sequence code, which is generated by the Lattice sensAI Neural Network Compiler software into the DRAM starting at the address specified by i_code_base_addr signal.
4. Check that o_rd_rdy is high; if not, go back to step 1.
5. Write input data into the DRAM at the proper address, which is decided by command sequence, or directly write into the internal memory block of the Advanced CNN Accelerator IP Core through the input data LMMI interface.
6. Assert i_start and check o_rd_rdy. The o_rd_rdy signal deasserts to 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. Repeat from step 5, for the next set of input to the ML IP.

2.7.1. 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. The command is generated by the Lattice sensAI Neural Network Compiler software. For more information, refer to Lattice sensAI Neural Network Compiler Software User Guide (FPGA-UG-02052).

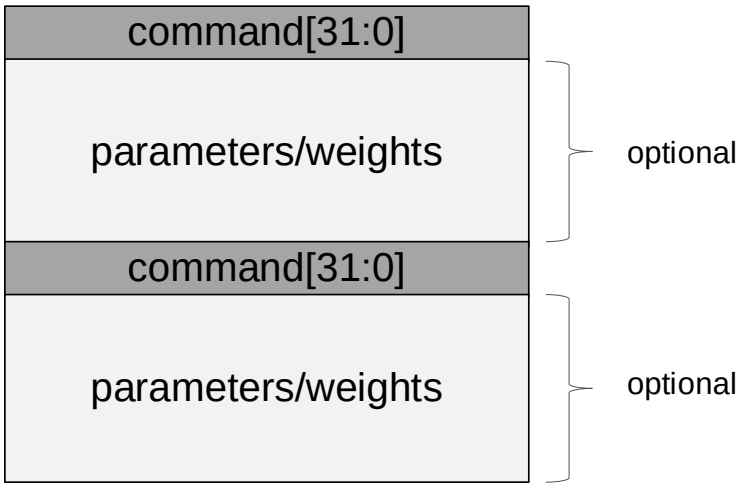

Figure 2.8. Command Format

2.7.2. 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 the Lattice Advanced CNN Accelerator IP Core through the input data write interface. If input data is 8-bit use only the lower 8-bit of the input data interface and set the upper 24 bits to 24'h00 (and likewise for half word/16 bit write). For example, face detection neural network may take 32 × 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 × 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 Advanced CNN Accelerator IP Core.

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.

The input data must agree with the Byte Mode attribute settings.

2.7.3. Output Data Format

The output data is a sequence of 16-bit data which is controlled by commands. The 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.

2.8. Supported Commands

The command sequences are generated by Lattice sensAI Neural Network Compiler Software (FPGA-UG-02052).

3. Core Generation, Simulation, and Validation

This section provides information on how to generate the Advanced CNN Accelerator IP Core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software 2.0 User Guide.

3.1. Licensing the IP

An IP core-specific device-specific license string is required to enable full, unrestricted use of the Lattice Advanced CNN Accelerator IP Core in a complete, top-level design. User may refer to the instructions on how to obtain licenses for Lattice IP cores at Lattice Software Licensing web page.

User may download and generate the Advanced 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 Advanced 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 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.

3.2. Generation and Synthesis

The Lattice Radiant software allows user to customize and generate modules and IPs and integrate them into the device's architecture. The procedure for generating the Advanced CNN Accelerator IP Core in Lattice Radiant software is described below.

To generate the Advanced CNN Accelerator IP Core:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click on **Advanced_CNN_Accelerator** under **IP, DSP** category. The **Module/IP Block Wizard** opens as shown in Figure 3.1. Enter values in the **Component name** and the **Create in** fields and click **Next**.

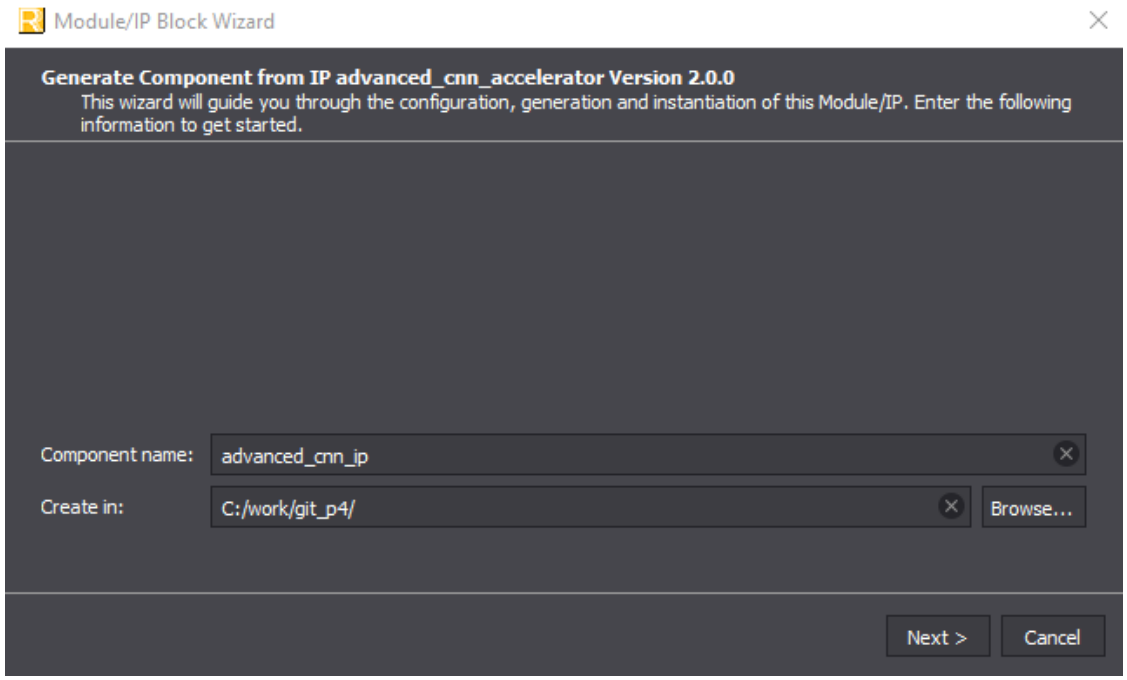

Figure 3.1. Module/IP Block Wizard

3. In the module's dialog box of the **Module/IP Block Wizard** window, customize the selected Advanced CNN Accelerator IP Core using drop-down menus and check boxes. As a sample configuration, see Figure 3.2. For configuration options, see the Attribute Summary section.

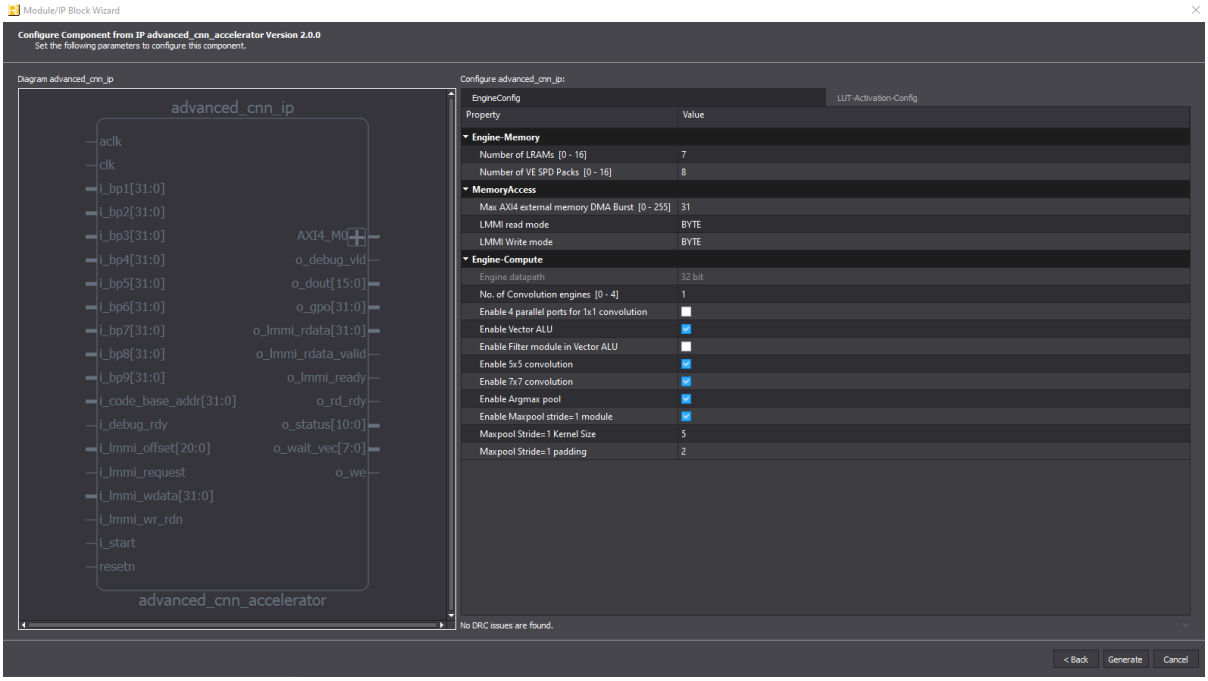

Figure 3.2. Configure User Interface of Advanced CNN Accelerator IP Core

4. Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 3.3.

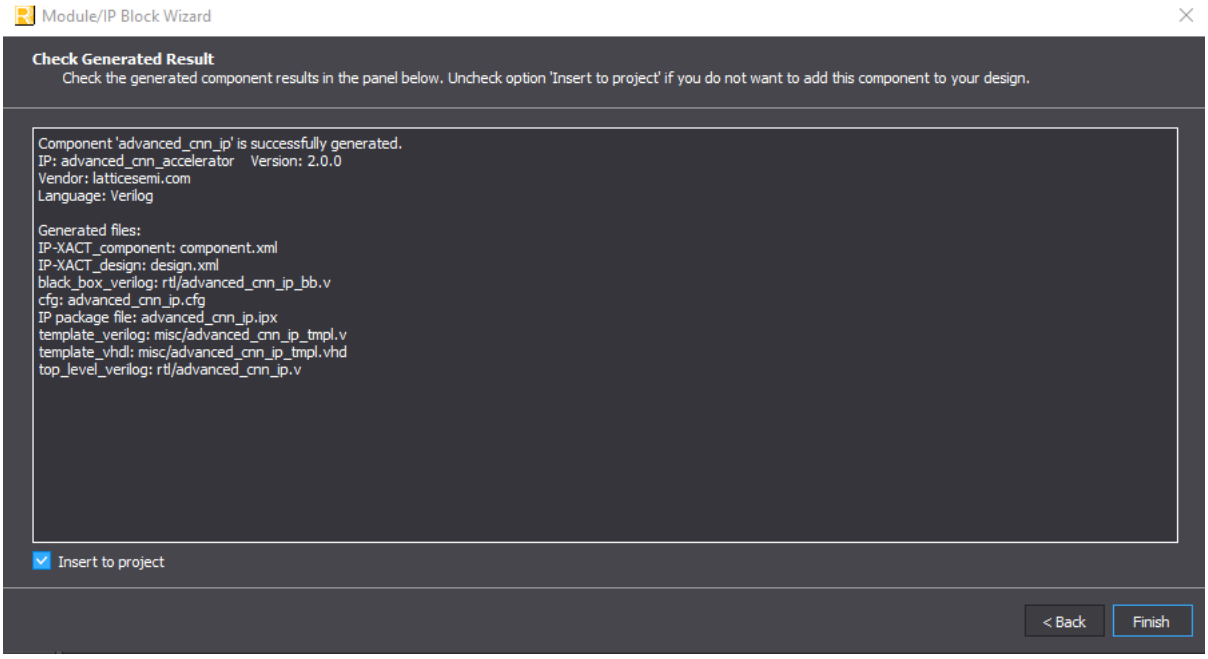

Figure 3.3.Check Generating Result

5. Click the **Finish** button. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 3.1.

The generated Advanced CNN Accelerator IP Core package includes the closed-box (<Component name>_bb.v) and instance templates (<Component name>_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (<Component name>.v) that can be used as an instantiation template for the IP core is also provided. User may also use this top-level reference as the starting template for the top-level of the complete design. The generated files are listed in Table 3.1.

Table 3.1. Generated File List

Attribute	Description
<Component name>.ipx	Contains the information on the files associated to the generated IP.
<Component name>.cfg	Contains the parameter values used in IP configuration.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	Provides an example RTL top file that instantiates the IP core.
rtl/<Component name>_bb.v	Provides the synthesis closed-box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	Provide instance templates for the IP core.

3.3. Running Functional Simulation

The Advanced CNN Accelerator IP does not have a functional simulation testbench.

3.4. Hardware Evaluation

The Advanced 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.

4. Hardware Validation

This IP is validated through the *Machine Vision: Bardcode Detection and Reading* reference design on VVML CPNX-100 board and *Object Classification: Video Stream Analysis* reference design on Lattice Avant-AT-E evaluation board. Refer to Lattice SensAI Stack Reference Designs for the demos, and details on the configuration used for validation.

5. Ordering Part Number

The Ordering Part Numbers (OPN) for the Advanced CNN Accelerator IP Core are the following:

- CNNADV-ACCEL-CPNX-UT – Advanced CNN Accelerator IP for CertusPro-NX – Multi-site Perpetual License
- CNNADV-ACCEL-CPNX-US – Advanced CNN Accelerator IP for CertusPro-NX – Single Machine Annual License
- CNNADV-ACCEL-AVE-UT – Advanced CNN Accelerator IP for Lattice Avant-AT-E – Multi-site Perpetual License
- CNNADV-ACCEL-AVE-US – Advanced CNN Accelerator IP for Lattice Avant-AT-E – Single Machine Annual License

Appendix A. Resource Utilization

Table A.1 shows configuration and resource utilization for LFCPNX-100 and LAV-AT-E70 using Synplify Pro Synthesis Engine of Lattice Radiant software.

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

Configuration	FPGA	clk, aclk (MHz)[2]	Registers	LUTs	LRAMs	EBRs	DSP MULT (in terms of 18x18 MULT)
VE_TYPE = "VE64" (64b engine) LRAM_NUM : 8 SPD_NUM : 8 CONV_NUM : 1 CONV_M4 : 1 EN_VE : 0 EN_VE_FILT : 0 EN_CONV5 : 1 EN_CONV7 : 1 EN_ARGMAX_POOL : 0 EN_MAXPOOL_S1 : 0 (Both Scale and FC LUT activations disabled)	LAV-AT-E70	200, 100	50512	46039	(Avant-AT-E has no LRAMs, so ML SPD is made of EBRs)	407	282 (Uses DOT PROD DSP primitive)
VE_TYPE = "VE32" (32b engine) LRAM_NUM : 8 SPD_NUM : 8 CONV_NUM : 1 CONV_M4 : 1 EN_VE : 0 EN_VE_FILT : 0 EN_CONV5 : 1 EN_CONV7 : 1 EN_ARGMAX_POOL : 0 EN_MAXPOOL_S1 : 0 (Both Scale and FC LUT activations disabled)	LAV-AT-E70	200, 100	27393	25574	(Avant-AT-E has no LRAMs, so ML SPD is made of EBRs)	271	74 (Uses DOT PROD DSP primitive)
VE_TYPE = "VE32" (32b engine) LRAM_NUM : 7 SPD_NUM : 8 CONV_NUM : 1 CONV_M4 : 0 EN_VE : 1 EN_VE_FILT:0 EN_CONV5 : 1 EN_CONV7 : 1 EN_ARGMAX_POOL : 1 EN_MAXPOOL_S1 : 1 (Both Scale and FC LUT activations disabled)	LFCPNX-100	96, 96	36546	50006	7	174	91

Notes:

1. Performance may vary when using a different software version or targeting a different device density or speed grade.
2. The *clk* and *aclk* numbers are from timing closure in ML demo and reference designs released with sensAI 6.0.

References

- Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039)
- Lattice sensAI Neural Network Compiler Software User Guide (FPGA-UG-02052)
- Lattice Radiant Software 2.0 User Guide
- Lattice SensAI Stack Reference Designs
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- CertusPro-NX web page
- Avant-E web page
- Lattice sensAI Stack web page
- Lattice Radiant Software web page
- Lattice Insights web page for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 2.0, December 2023

Section	Change Summary
All	• Changed the document name from *Advanced CNN Accelerator IP Core - Lattice Radiant Software* to *Advanced CNN Accelerator IP*. • Minor adjustments to ensure the document is consistent with Lattice Semiconductor's inclusive language policy.
Disclaimers	Updated boilerplate.
Inclusive Language	Added boilerplate.
Acronyms in This Document	Added Arithmetic Logic Unit (ALU).

Introduction	• Added *IP Core v2.0.0 – Lattice Radiant Software 2022.1 or later*, and link to Lattice Radiant Software web page in Table 1.1.
	• Added new key features in the Features section:
	  • Supports higher throughput 64-bit data path engine for Avant-AT-E, and 32-bit data path engine for Avant-AT-E and CertusPro-NX devices
	  • Supports Vector ALU (Arithmetic Logic Unit) for enhanced pixelwise operations, and accelerated pre/post ML image processing algorithms
	  • Supports 3 x 3, and 5 x 5 convolution with 16-bit input for higher output accuracy
	  • Supports lookup table-based activation functions after convolution, and fully connected layers
	  • Supports new Filter module in Vector ALU for specialized kernels like min/max/median
	• Added the sentence *The Vector ALU also supports acceleration of image processing algorithms and other pre/post processing ML algorithms* to Overview section.

Interface Descriptions	• Added the sentence *which enables efficient management of available external memory, and sharing of data in multiple ML engine scenario* to the functional description of *i_bp[31:0]* pin in Table 2.1. • Updated the names of *o_status[10:0]*, and *i_lmmi_offset[20:0]* pins and their functional description in Table 2.1. • Added [31:0] to the pin names for *I_lmmi_wdata*, and *o_lmmi_rdata* in Table 2.1.
Attribute Summary	• Removed attributes *Number of EBRs in VE SPD*, and *Avant Mode* from Table 2.2. • Added new attributes *Engine Datapath*, *Enable Filter module in Vector ALU*, *Enable activation function in scale module*, *Lookup table address width(For scale)*, *Lookup table data width(For scale)*, *Scale Lookup table init file relative path(For scale)*, *Enable MSB clipping(For scale)*, *Enable activation function in FC module*, *Lookup table address width(For FC)*, *Lookup table data width(For FC)*, *FC Lookup table init file relative path*, and *Enable MSB clipping(For FC)* to Table 2.2.
Core Generation, Simulation, and Validation	Updated Figure 3.1, Figure 3.2, and Figure 3.3 in Generation and Synthesis section.
Hardware Validation	Added this section.
Ordering Part Number	Updated this section.
Resource Utilization	Updated Table A.1 by replacing the old configurations with new configurations, and updated the device name from *Avant-E-500* to *LAV-AT-E70*.
References	Added links to *Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039)*, *Lattice sensAI Neural Network Compiler Software User Guide (FPGA-UG-02052)*, *Lattice Radiant Software 2.0 User Guide*, *Lattice SensAI Stack Reference Designs*, *Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)*, *Lattice sensAI Stack web page*, *Lattice Radiant Software web page*, and *Lattice Insights web page*.

Revision 1.0, March 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com