Compact CNN Accelerator IP Core

User Guide

FPGA-IPUG-02038-1.2

May 2019

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

1. Introduction 5
1.1. Quick Facts 5
1.2. Features 5
2. Functional Descriptions 6
2.1. Overview 6
2.2. Interface Descriptions 8
2.2.1. Control and Status 10
2.2.2. Input Data Interface 10
2.2.3. Command FIFO Interface 11
2.2.4. Result and Debug Interface 11
2.3. Reset Behavior 12
2.4. Register Description 12
2.5. Operation Sequence 12
2.5.1. Command Format 13
2.5.2. Input Data Format 13
2.5.3. Output Data Format 13
2.6. Supported Commands 13
3. Parameter Settings 14
4. IP Generation and Evaluation 16
4.1. Licensing the IP 16
4.2. Generation and Synthesis 16
4.3. Running Functional Simulation 18
5. Ordering Part Number 19
References 20
Technical Support Assistance 20
Appendix A. Resource Utilization 21
Revision History 22

Figures

Figure 2.1. Functional Block Diagram (Machine Learning Type == BNN) 6
Figure 2.2. Functional Block Diagram (Machine Learning Type == CNN) 7
Figure 2.3. Compact CNN Accelerator IP Core Interface Diagram 8
Figure 2.4. Control and Status Interface Timing Diagram 10
Figure 2.5. Input Data Interface Timing Diagram 10
Figure 2.6. Command FIFO Interface Timing Diagram 11
Figure 2.7. Result and Debug Interface Timing Diagram (No Debug Data) 11
Figure 2.8. Result and Debug Interface Timing Diagram (With Debug Data) 12
Figure 2.9. Reset Timing Diagram 12
Figure 2.10. Command Format 13
Figure 3.1. Compact CNN Accelerator IP Core Configuration User Interface 14
Figure 4.1. Module/IP Block Wizard 17
Figure 4.2. Check Generating Result 17

Tables

Table 1.1. Quick Facts 5
Table 2.1. Compact CNN Accelerator IP Core Signal Descriptions 9
Table 3.1. Attributes Table 14
Table 3.2. Attributes Descriptions 15
Table 4.1. Generated File List 18
Table A.1. Performance and Resource Utilization (Machine Learning Type == BNN)[1] 21
Table A.2. Performance and Resource Utilization (Machine Learning Type == CNN)[1] 21

1. Introduction

The Lattice Semiconductor Compact CNN Accelerator IP Core is a calculation engine for Deep Neural Networks with fixed point weight or binarized weight. It calculates many layers of neural networks including convolution, pooling, batch normalization, and full connect by executing sequence code with weight values which is generated by the Lattice Neural Network Compiler tool. The engine is optimized for convolutional neural networks, which is suitable for analysis of image data and also some applications where none image data can be represented visually. Typical applications include image classification, face detection, and key phrase detection. The engine can operate standalone without the addition of a seperate processor.

The design is implemented in Verilog HDL. It can be targeted to iCE40 UltraPlus™ FPGA devices and 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 Compact CNN Accelerator IP Core.

Table 1.1. Quick Facts

IP Requirements	FPGA Families Supported	iCE40 UltraPlus
Resource Utilization	Targeted Device	iCE40 UP5K
	Supported User Interface	Native interfaces, please refer to Interface Descriptions section.
	Resources	See Table A.1 and Table A.2.
Design Tool Support	Lattice Implementation	IP Core v1.0.x - Lattice Radiant Software 1.0 IP Core v1.1.x - Lattice Radiant Software 1.1
	Synthesis	Lattice Synthesis Engine
		Synopsys® Synplify Pro
	Simulation	For a list of supported simulators, see the Lattice Radiant Software 1.0 User Guide and Lattice Radiant Software 1.1 User Guide.

1.2. Features

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

- Support for binarized convolution layer, max/or pooling layer, binarizer and binarized full connect layer
- Configurable Blob memory type and size
- Optimization for 3 x 3 2D convolution calculation
- Dynamic support for various 1D convolution from 1 to 9 taps
- Support for fixed point convolution layer, max pooling layer and scaling layer
- Support for RELU and leaky RELU (fixed negative slope)
- Configurable size of scratch pad of convolution layer (1K entries, 4K entries)

2. Functional Descriptions

2.1. Overview

The Compact CNN Accelerator IP Core performs a series of calculations per a command sequence that is generated by the Lattice Neural Network Compiler tool. Commands must be fed through the command FIFO interface. Input data is directly written through input data write port. After command code and input data are available, the Compact CNN Accelerator IP Core starts calculation at the rising edge of start signal. During calculation, intermediate data and final result are fed out through the result write port. All operations are fully programmable by command code.

The Compact CNN Accelerator IP Core has different implementations for BNN and CNN depending on the Machine Learning Type attribute value. The functional block diagram for BNN and CNN implementation are shown in Figure 2.1 and Figure 2.2, respectively.

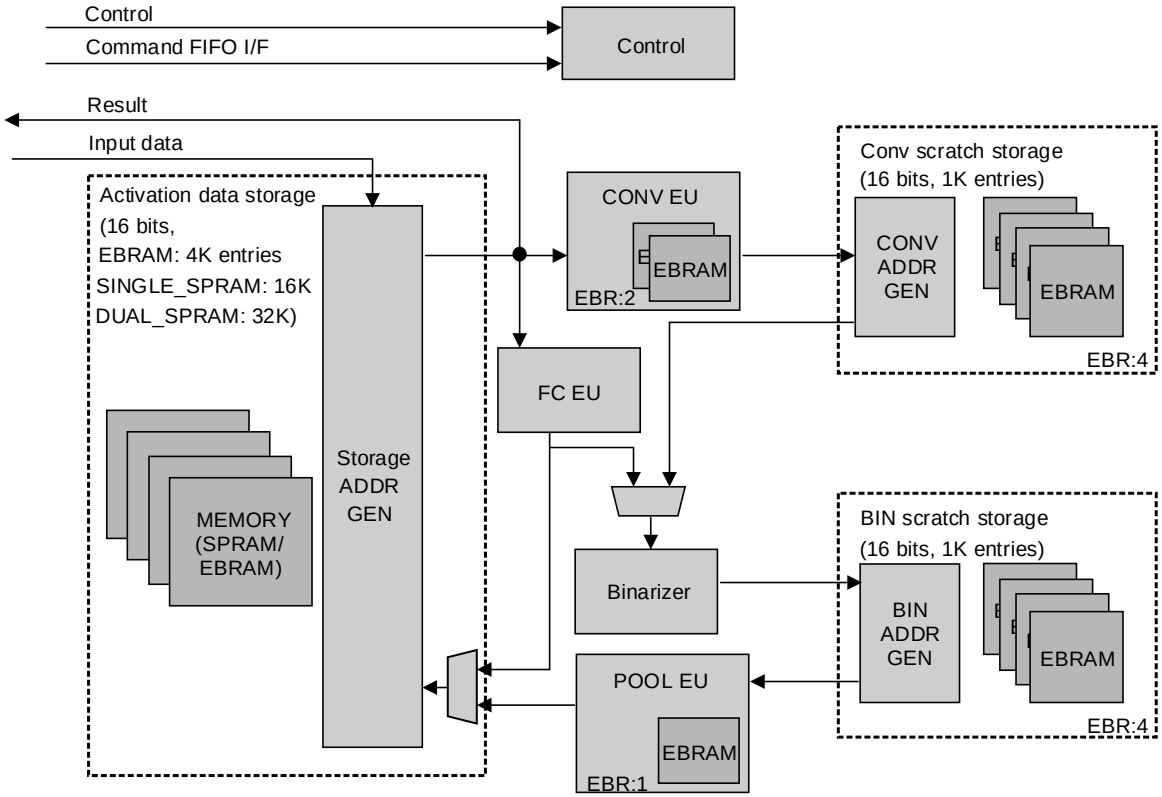

Figure 2.1. Functional Block Diagram (Machine Learning Type == BNN)

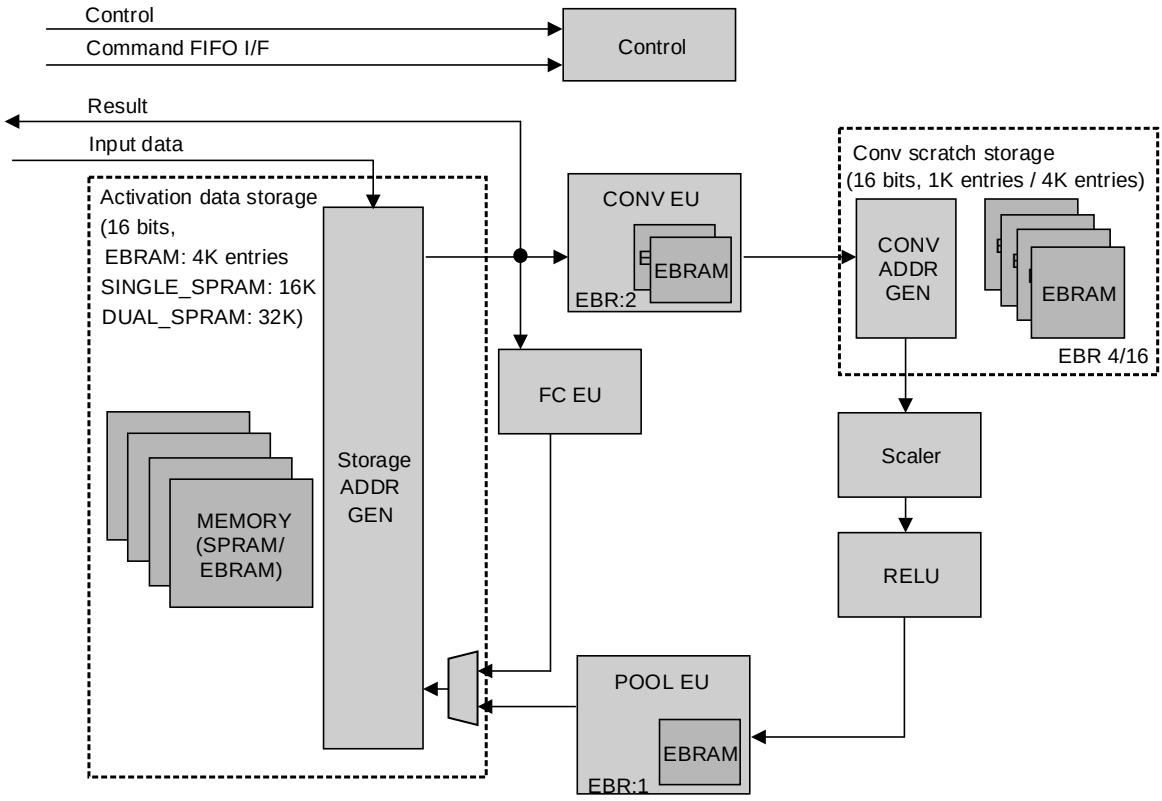

Figure 2.2. Functional Block Diagram (Machine Learning Type == CNN)

2.2. Interface Descriptions

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

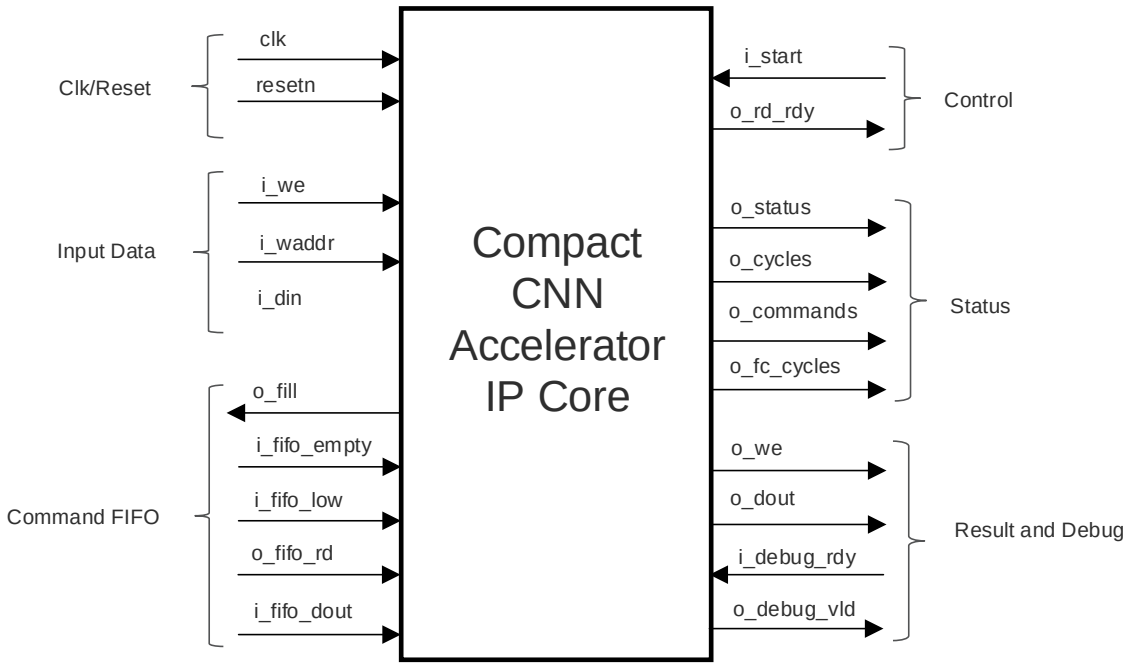

Figure 2.3. Compact CNN Accelerator IP Core Interface Diagram

Table 2.1. Compact 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 0: Resets all ports and sets internal registers to their default values. 1: Reset is NOT active
Control and Status		
i_start	Input	Start execution signal. This signal is level sensitive and 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[7:0]	Output	Debug information [0]: Indicates activity of engines except full connect engine [1]: Indicates activity of full connect engine [2~7]: 0 (for future usage)
o_cycles[31:0]	Output	Indicates the number of clock cycles that Compact CNN Accelerator IP Core is running. The count is reset to 32'b0 on i_start assertion.
o_commands[31:0]	Output	Indicates the number of commands that Compact CNN Accelerator IP Core processed. The count is reset to 32'b0 on i_start assertion.
o_fc_cycles[31:0]	Output	Indicates the number of clock cycles that fully connected later is running. The count is reset to 32'b0 on i_start assertion.
Input Data		
i_we	Input	Write enable signal for internal memory 0 : No write transaction, i_waddr and i_din are ignored 1 : Write transaction is enabled, i_din and i_waddr are valid
i_waddr[15:0]	Input	Write address signal for internal memory.
i_din[15:0]	Input	Input data signal for internal memory.
Command FIFO		
o_fill	Output	Request for filling the external command FIFO 0 : Request to flush and fill the external command FIFO External command fifo must be flushed and the command code generated by Lattice Neural Network Compiler must be loaded to external command FIFO. 1 : Normal FIFO operation
i_fifo_empty	Input	Indicates whether external command FIFO is empty or not. 0 : External FIFO is not empty 1 : External FIFO is empty
i_fifo_low	Input	Indicates whether external command FIFO level is low or not. 0 : FIFO level is not low 1 : FIFO level is low If fifo level is not low, fifo must consecutively provide one full set of full connect weights, that is, by one 32-bit data per two cycles.
o_fifo_rd	Output	Read request to external command FIFO 0 : No Read request 1 : Read request to external FIFO is active Value of i_fifo_dout is sampled when i_fifo_empty = 0 and o_fifo_rd = 1.
i_fifo_dout[31:0]	Input	External command FIFO read data

Pin Name	Direction	Function Description
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. IF o_debug_vld is asserted, o_dout[15:0] contains debug data.
i_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. Recommend to connect to 1 if debug feature is not used.
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

2.2.1. Control and Status

After reset or when engine is idle, o_rd_rdy is high. During this state, external logic may write input data through the Input Data interface. After writing input data is completed, external logic must assert the i_start signal. The engine starts execution when it reaches i_start = 1 and o_rd_rdy goes 0 during execution. During execution, each bit of o_status indicates activity of sub calculation engine. After completing execution, that is, by getting to finish command, Compact CNN Accelerator IP Core asserts o_rd_rdy (goes idle) and waits for the next execution.

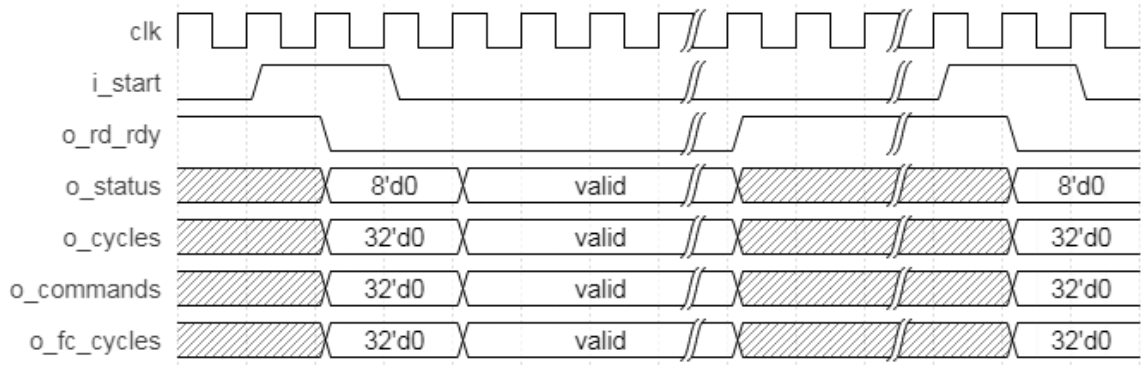

Figure 2.4. Control and Status Interface Timing Diagram

2.2.2. Input Data Interface

External logic should write input data to internal memory of Compact CNN Accelerator IP Core only during idle state (o_rd_rdy is high). Writing to internal memory while o_rd_rdy is low is ignored. The address must be matched to command code. Input Data Interface is based on simple SRAM interface as shown in Figure 2.5. Since input data is written to internal SRAM, there is no required order or rule. Any random access is acceptable. Overwriting of the same address is also accepted.

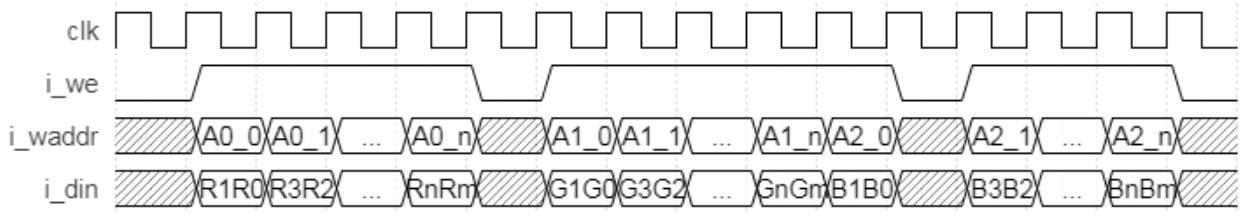

Figure 2.5. Input Data Interface Timing Diagram

2.2.3. Command FIFO Interface

External command FIFO must provide command code that is generated by the Lattice Neural Network Compiler tool. When o_fill signal is at logic 0, external logic should load the command code to external command FIFO. After the i_start control signal is asserted, o_fill go to logic 1 and normal FIFO operation proceeds as shown in Figure 2.6. When i_fifo_empty is low, i_fifo_dout contains a valid command. The command is latched/sampled when i_fifo_empty is low and i_fifo_rd is high. The external command FIFO holds current command (value of i_fifo_dout) when i_fifo_empty and o_fifo_rd are both low.

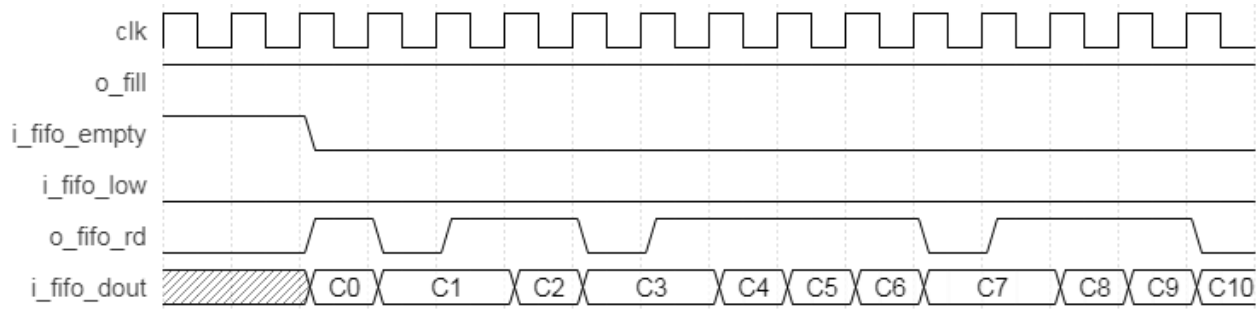

Figure 2.6. Command FIFO Interface Timing Diagram

2.2.4. Result and Debug Interface

This interface transmits the result, that is, by the final Blob data of the neural network. Interface consists of o_we as valid indicator and o_dout as 16-bit data. Usually, it may be a single burst series of 16-bit data. The amount of output data is programmable by command code. This interface does not have a ready signal. Thus, the receiving module should be able to receive the entire Blob.

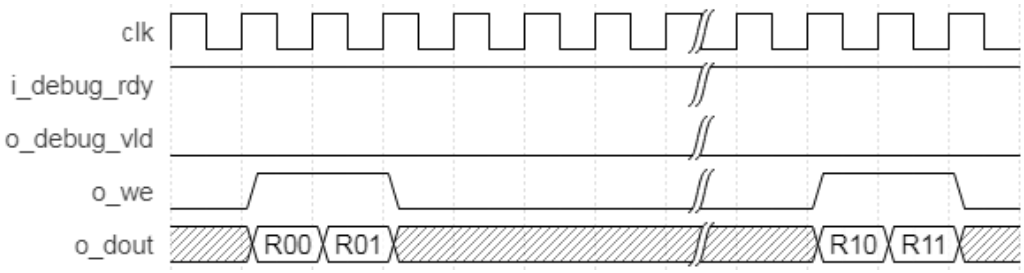

Figure 2.7. Result and Debug Interface Timing Diagram (No Debug Data)

Compact Compact CNN Accelerator IP Core may transfer debug data through o_dout port by asserting o_debug_vld signal per command code. The amount of debug data is also set in the command code. Debug data or intermediate result can be transferred in Result and Debug Interface without affecting normal operation. Unlike the Result data, transfer of debug data can be controlled by i_debug_rdy signal. Initially, external logic must be able to accept one burst read of debug data, and should assert i_debug_rdy signal to 1. If buffer is not available for next consecutive burst read, external logic must deassert i_debug_rdy signal to 0, in order to hold operation of Compact CNN Accelerator IP Core. A sample transmission of debug data and result data is shown in Figure 2.8. In this example, only 2 bursts of debug data are set in the command code thus, only 2 bursts are transmitted even if i_debug_rdy signal is still asserted.

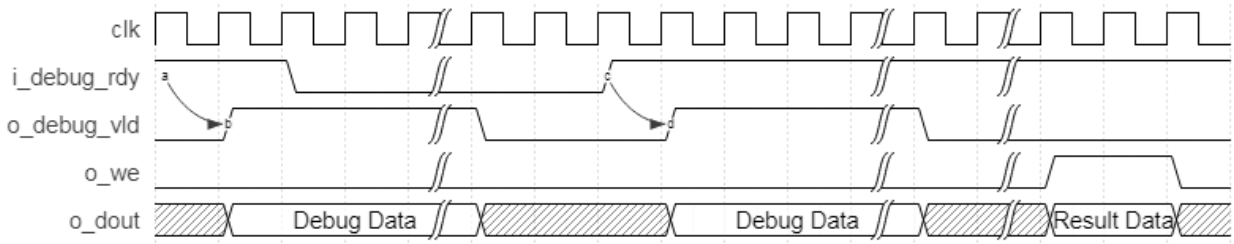

Figure 2.8. Result and Debug Interface Timing Diagram (With Debug Data)

2.3. Reset Behavior

When resetn signal asserts, output ports return to default value in the next cycle. The default value of o_rd_rdy signal is 1, while all other output signals are 0. A timing diagram of reset during operation is shown as an example in Figure 2.9. The minimum resetn assert period is 1 clk cycle.

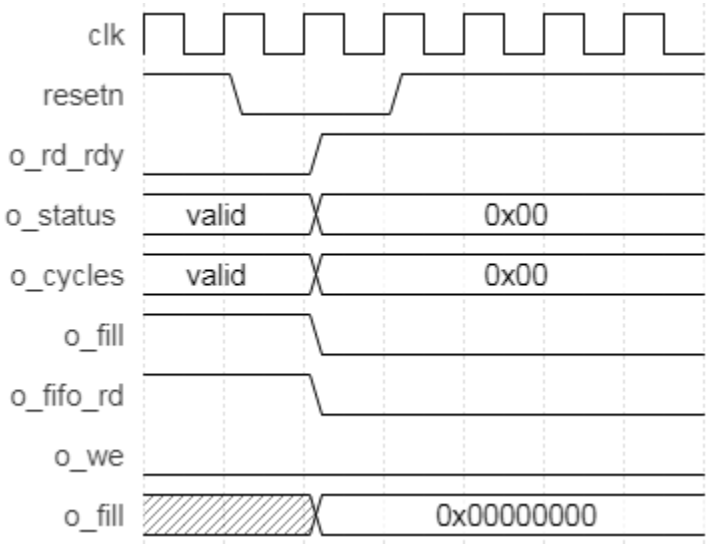

Figure 2.9. Reset Timing Diagram

2.4. Register Description

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

2.5. Operation Sequence

Operation must be executed in the following sequence:

1. Assert Reset.
2. Deassert Reset; i_start must be deasserted.
3. Write the command sequence code, which is generated by the Lattice Neural Network Compiler tool, into FIFO.
4. Check whether o_rd_rdy is high or not. o_rd_rdy must be high. Otherwise, go back to step 1.
5. Write into memory block of Compact CNN Accelerator IP Core through Input Data interface at proper address, which is decided by command sequence.
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 and collect o_dout while o_we == 1.
9. Repeat from step 5 for the next Input data.

2.5.1. Command Format

Command is a sequence of 32-bit data with or without additional parameters or weights. It should be loaded at address 0x0000 before execution. Engine expects command in little-endian order. Command is generated by the Lattice Neural Network Compiler tool. For more information, refer to Lattice SensAI Neural Network Compiler Software User Guide (FPGA_UG-02052).

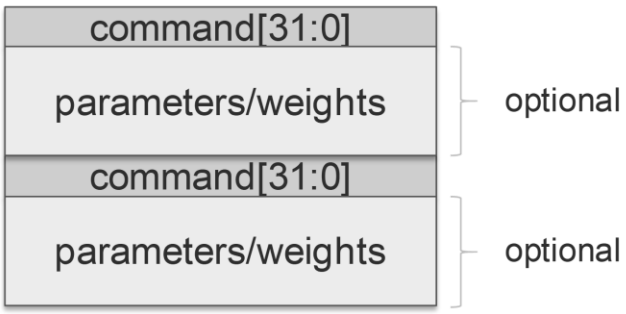

Figure 2.10. Command Format

2.5.2. Input Data Format

Input data is a sequence of 8-bit or 16-bit data. Address is decided by the neural network. Therefore, external block should process input raw data and write input data to the Compact CNN Accelerator IP Core through the input data write interface. Since IP core only has 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 with address 0x0000 for Red plane, 0x0400 for Green plane, and 0x0800 for Blue plane. Because memory assignment is defined by the neural network, external block should handle input raw data and write it to proper position of internal memory of Compact CNN Accelerator IP Core. The IP Core expects data in little-endian order.

2.5.3. Output Data Format

Output data is a sequence of 16-bit data which is controlled by commands. The amount of data is also decided by the neural network, that is, by output Blobs. External block should interpret output sequence and generate usable information. For example, face detection may have two classes. Therefore, output may be two of 16-bit data and final result may be simple comparison of these two 16-bit data. 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.

2.6. Supported Commands

Command sequences are generated by the Lattice Neural Network Compiler tool. For more information, refer to Lattice SensAI Neural Network Compiler Software User Guide (FPGA-UG-02052).

3. Parameter Settings

The Clarity Designer tool is used to create IP and architectural modules in the Lattice Radiant Software. 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 Compact CNN Accelerator IP Core. The attribute values are specified using the IP core Configuration user interface in the Lattice Radiant Software 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	—
Memory Type	EBRAM, DUAL_SPRAM, SINGLE_SPRAM	SINGLE_SPRAM	—
BNN Blob Type	+1/-1, +1/0	+1/0	Machine Learning Type == BNN
Scratch Pad Memory Size	1K, 4K	1K	Machine Learning Type == CNN

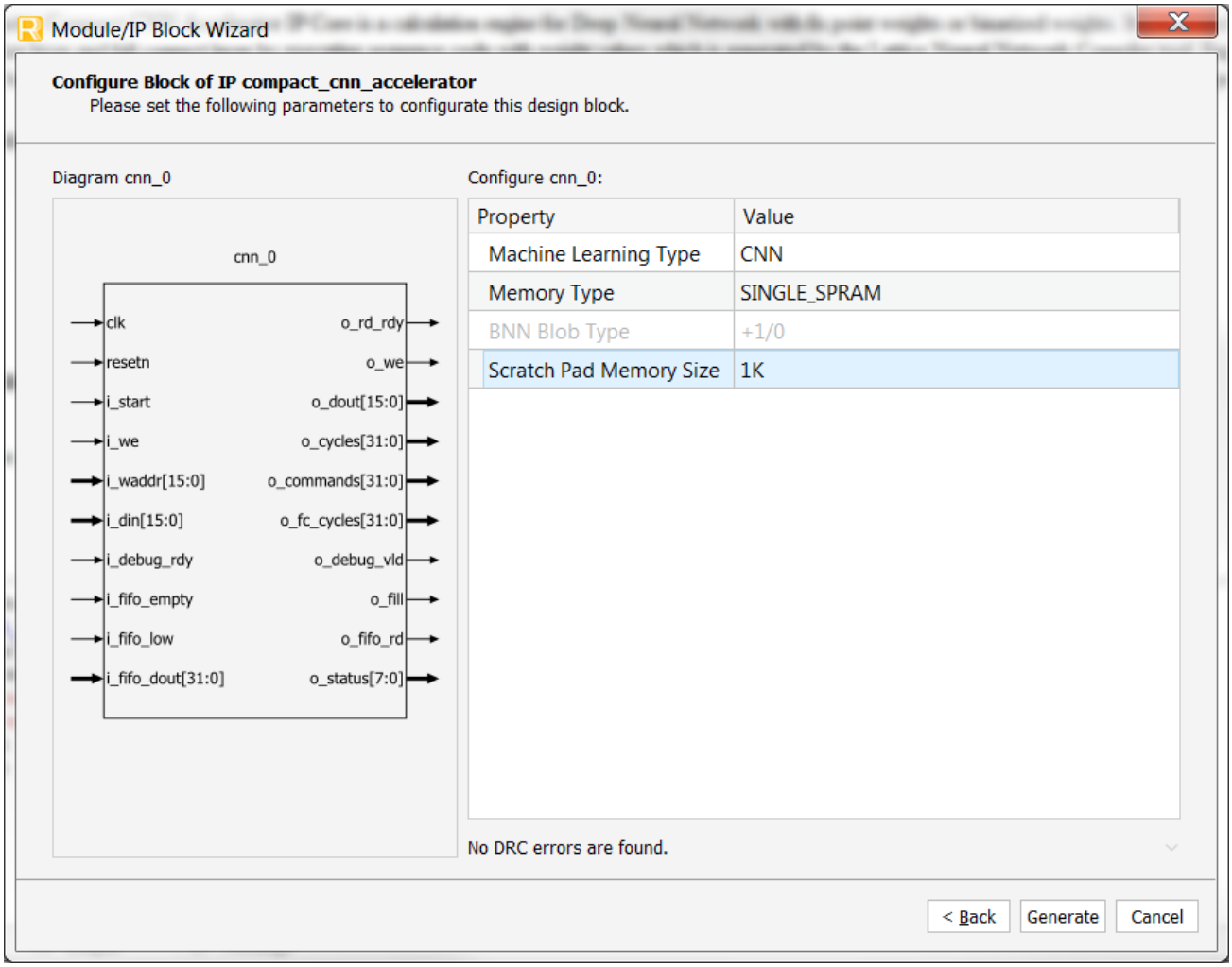

Figure 3.1. Compact CNN Accelerator IP Core Configuration User Interface

Table 3.2. Attributes Descriptions

Attribute	Description
Machine Learning Type	Selects the machine learning engine type between CNN and BNN
Memory Type	EBRAM: Use 16 EBRAM. Total size is 8 kB. SINGLE_SPRAM: Use 1 SPRAM. Total size is 32 kB. DUAL_SPRAM: Use 2 SRPAMs. Total size is 64 kB. Use SINGLE_SPRAM or DUAL_SPRAM based on required size of peak Blob data EBRAM option is for special usage when no SPRAM is available
BNN Blob Type	Selects the type of binary blob data, either +1/–1 or +1/0. This setting should be matched to Lattice Neural Network Compiler. This setting is only valid when Machine Learning Type == BNN.
Scratch Pad Memory Size	Configures the memory size of scratch pad of convolution. 1K: Size of scratch pad memory is 2 kB (1 k x 2 bytes). 4K: Size of scratch pad memory is 8 kB (4 k x 2 bytes). This setting is only valid when Machine Learning Type == CNN. The BNN engine uses fixed scratch memory of 1K (2 kB) for the BIN and convolution functions.

4. IP Generation and Evaluation

This section provides information on how to generate the IP using the Lattice Radiant Software, and how to run simulation, synthesis, and hardware evaluation. For more details on the Lattice Radiant Software, refer to the Lattice Radiant Software 1.0 User Guide and Lattice Radiant Software 1.0 Tutorial.

4.1. Licensing the IP

IP core v1.1.x requires an IP core-specific license string and a Lattice Radiant software 1.1 license patch to enable full use of the Lattice Compact CNN Accelerator IP Core in a complete, top-level design. Without a license string and the software license patch, the bitstream file is not generated. Compact CNN Accelerator IP Core license string is available in 30-day evaluation license and full license with yearly renewal. The evaluation license may be used only for the purpose of checking the usability of the IP core. The bitstream generated using evaluation license must not be used in actual product.

If you want to use the Compact CNN Accelerator IP Core in your product, you may obtain a 30-day evaluation license by going to http://www.latticesemi.com/Support/Licensing/IPCore/CompactCNN and entering your 12-digit hexadecimal Host Physical Address (MAC address).

Ensure that you are logged in using your organization/company email as the license file will be sent to this email.

If you have purchased and received the IP Core product, you may obtain the full license by going to http://www.latticesemi.com/en/Support/Licensing/IPCore/IPCoreNew and providing all required information.

The IP core v1.1.x supports Lattice Radiant Software 1.1 and future versions. For more details, refer to the Lattice Radiant Software 1.1 User Guide and Lattice Radiant Software 1.1 Tutorial.

You can obtain the Lattice Radiant 1.1. software patch file from the Lattice website through Lattice Radiant 1.1 Software Patch.

4.2. Generation and Synthesis

The Lattice Radiant Software allows customization and generation of modules. The procedure for generating Compact CNN Accelerator IP Core in Lattice Radiant Software is described below:

1. Create a new Lattice Radiant Software project or open an existing project.
2. In the IP Catalog tab, double-click on **Compact_CNN_Accelerator** under IP, DSP category. The Module/IP Block Wizard - appears as shown in Figure 4.1. Enter the values in the **Instance name** and the **Create in** fields.

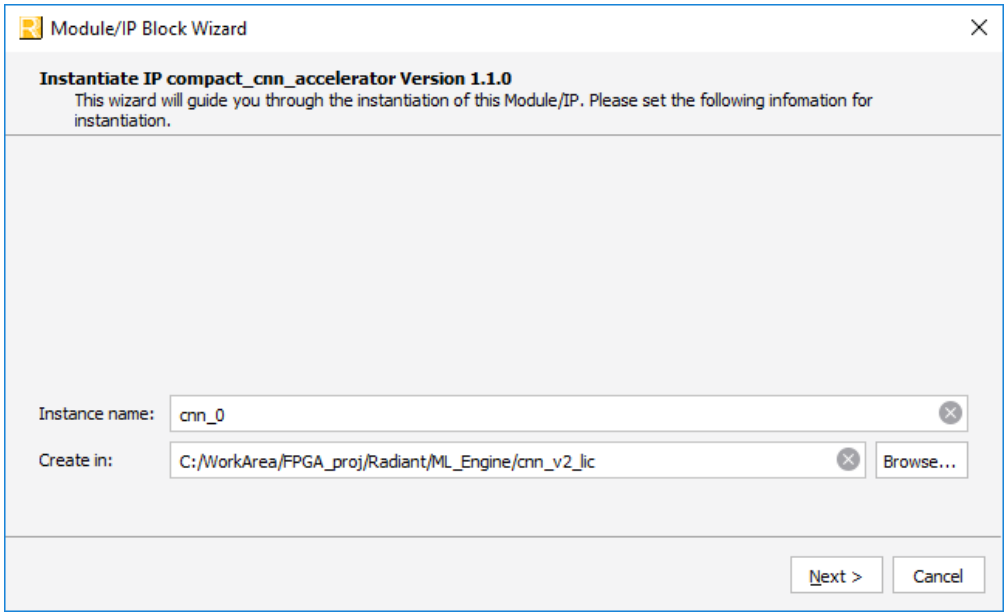

Figure 4.1. Module/IP Block Wizard

3. Click **Next**.
4. Update the configuration parameters as necessary as shown in Figure 3.1. .
5. Click the **Generate** button to generate the IP. Confirm the generation result as shown in Figure 4.2. Check the **Insert to project** option to add the generated IP to the project.
6. Click **Finish**. All the generated files are placed under the directory paths in the Create in and the Instance name fields shown in Figure 4.1.

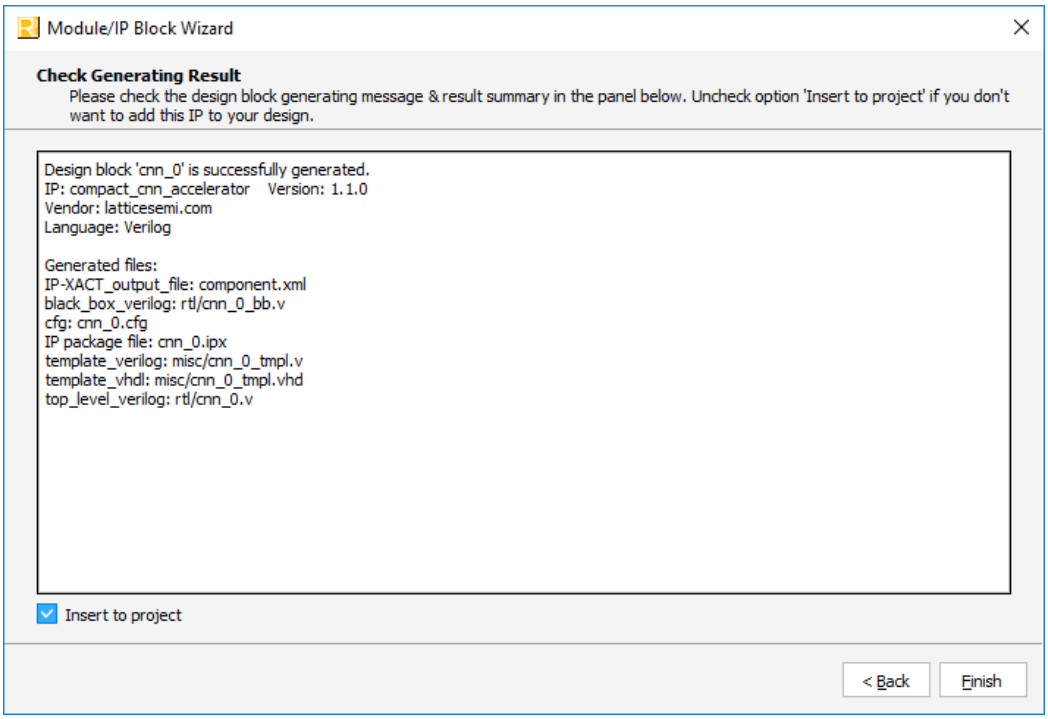

Figure 4.2. Check Generating Result

The generated Compact CNN Accelerator IP Core package includes black-box (<Instance Name>_bb.v) and instance templates (<Instance 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 (<Instance Name>.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. The generated files are listed in Table 4.1.

Table 4.1. Generated File List

Attribute	Description
<Instance Name>.ipx	This file contains the information on the files associated to the generated IP.
<Instance Name>.cfg	This file contains the parameter values used in IP configuration.
rtl/<Instance Name>.v	This file provides an example RTL top file that instantiates the IP core.
rtl/<Instance Name>_bb.v	This file provides the synthesis black box for the user's synthesis.
misc/<Instance Name>_tmpl.v misc /<Instance Name>_tmpl.vhd	These files provide instance templates for the IP core.

4.3. Running Functional Simulation

- The Compact CNN Accelerator IP Core does NOT contain a sample test bench.

5. Ordering Part Number

The Ordering Part Numbers (OPN) for Compact CNN Accelerator IP Core v1.1.x targeting iCE40UP FPGA devices are the following:

- CNN-CPACCEL-UP-U – Project License
- CNN-CPACCEL-UP-UT – Site License

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.

Appendix A. Resource Utilization

Table A.1 and Table A.2 show configuration and resource utilization for the iCE40 UltraPlus using Lattice Radiant Software 1.0.0.350.0. The following settings are used in generating this data:

- Synthesis Tool: Lattice Synthesis Engine
- Device Part No.: iCE40 UP5K-SG48I

Table A.1. Performance and Resource Utilization (Machine Learning Type == BNN)[1]

Memory Type	BNN Blob Type	Register	LUT4s	EBR	SRAM	clk Fmax[2] (MHz)
EBRAM	+1/0	1822	2419	27	0	41.762
DUAL_SPRAM	+1/0	1803	2447	11	2	31.565
SINGLE_SPRAM	+1/0	1802	2430	11	1	41.103
SINGLE_SPRAM	+1/–1	1992	2706	11	1	40.748

Notes:

1. Performance may vary when using a different software version or targeting a different device density or speed grade.
2. Fmax is generated when the FPGA design only contains Compact CNN Accelerator IP Core, these values may be reduced when user logic is added to the FPGA design.

Table A.2. Performance and Resource Utilization (Machine Learning Type == CNN)[1]

Memory Type	Scratch Pad[3]	Register	LUT4s	EBR	SRAM	clk Fmax[2] (MHz)
EBRAM	1K	1725	2816	23	0	28.164
DUAL_SPRAM	1K	1706	2867	7	2	27.672
SINGLE_SPRAM	1K	1705	2841	7	1	26.782
SINGLE_SPRAM	4K	2052	3989	19	1	25.950

Notes:

1. Performance may vary when using a different software version or targeting a different device density or speed grade.
2. Fmax is generated when the FPGA design only contains Compact CNN Accelerator IP Core, these values may be reduced when user logic is added to the FPGA design.
3. The *K* value in Scratch Pad is equivalent to kilobyte. For example, 1K is equal to **1 kB** of scratch pad memory.

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

Revision History

Revision 1.2, May 2019

Section	Change Summary
All	• Added Disclaimers section. • Updated last page of the document.
Quick Facts	• Added IP Core version – Radiant software version compatibility.
Licensing the IP	• Added licensing information.
Ordering Part Number	• Added Ordering Part Number information.

Revision 1.1, September 2018

Section	Change Summary
All	Renamed the IPUG from BNN Accelerator IP Core to Compact CNN Accelerator IP Core. Added support for convolutional neural network with fix point weights.
Introduction	Updated Features section. Added additional features.
Functional Description	• Updated text, Figure 2.1. Functional Block Diagram (Machine Learning Type == BNN) and Figure 2.2. Functional Block Diagram (Machine Learning Type == CNN) in Overview section. • Updated Figure 2.3. Compact CNN Accelerator IP Core Interface Diagram and Table 2.1. Compact CNN Accelerator IP Core Signal Descriptions in Interface Descriptions section.
Parameter Settings	Updated Figure 3.1. Compact CNN Accelerator IP Core Configuration User Interface, Table 3.1. Attributes Table, and Table 3.2. Attributes Descriptions.
Appendix A. Resource Utilization	• Updated Table A.1. Performance and Resource Utilization (Machine Learning Type == BNN)[1]. • Added Table A.2. Performance and Resource Utilization (Machine Learning Type == CNN)[1].

Revision 1.0, May 2018

Section	Change Summary
All	Initial release

www.latticesemi.com

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Development Software ***category:***

Click to view products by Lattice ***manufacturer:***

Other Similar products are found below :

RAPPID-560XBSW RAPPID-567XFSW DG-ACC-NET-CD SRP004001-01 SW006021-1NH SW163052 SYSWINEV21 Core429-SA SW500006-HPA CWP-BASIC-FL W128E13 CWP-PRO-FL AD-CCES-NODE-1 NT-ZJCAT1-EV4 CWA-BASIC-FL RAPPID-567XKSW CWA-STANDARD-R SW89CN0-ZCC CWA-LS-DVLPR-NL VDSP-21XX-PCFLOAT RAPPID-563XMSW IPS-EMBEDDED SDAWIR-4532-01 MPROG-PRO535E AFLCF-08-LX-CE060-R21 WS02-CFSC1-EV3-UP SYSMAC-STUDIO-EIPCPLR LIB-PL-PC-N-1YR-DISKID LS1043A-SWSP-PRM SW006026-COV MG2213 38100-401 1120270005 1120270006 MIKROBASIC PRO FOR FT90X (USB DONGLE) MIKROC PRO FOR AVR (USB DONGLE LICENSE) MIKROC PRO FOR FT90X (USB DONGLE) MIKROBASIC PRO FOR AVR (USB DONGLE LICEN MIKROBASIC PRO FOR FT90X MIKROC PRO FOR DSPIC30/33 (USB DONGLE LI MIKROC PRO FOR FT90X MIKROC PRO FOR PIC32 (USB DONGLE LICENSE 52202-588 MIKROPASCAL PRO FOR ARM (USB DONGLE LICE MIKROPASCAL PRO FOR FT90X MIKROPASCAL PRO FOR FT90X (USB DONGLE) MIKROPASCAL PRO FOR PIC32 (USB DONGLE LI SW006021-2H SW006023-3 CWP-STANDARD-FL