UG0943
User Guide
CNN Accelerator for PolarFire FPGA

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

50200943. 1.0 12/20

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.0

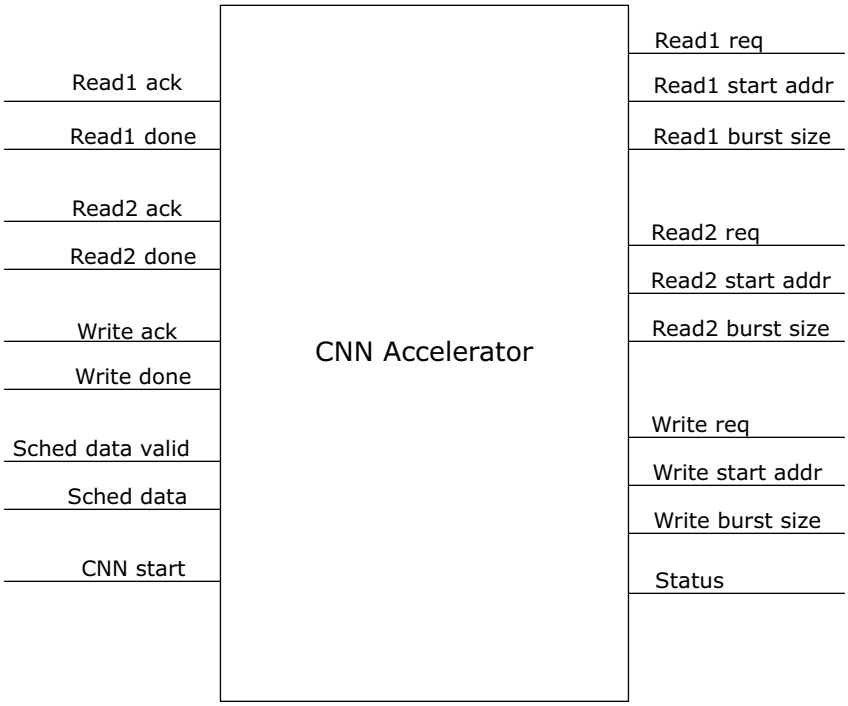
The first publication of this document.

2 Introduction

The CNN Accelerator IP provides hardware acceleration for inferencing Convolution Neural Networks (CNN) on PolarFire® FPGA. The CNN accelerator performs several DSP operations in a single clock cycle to achieve acceleration. A CNN consist of several types of layers connected in sequence like Convolution, Maxpool, ReLU, Fully connected layer, etc. A convolution layer uses Kernels with coefficients called as weights. The IP executes some of these layers sequentially and some of the layers simultaneously. The output of each layer called activations is stored in DDR and used as input to the next layer. The weights of the CNN are stored in DDR and are read along with the input corresponding to a convolution layer. The scheduler inside the CNN IP manages sequencing of a frame start, execution of different layers till the final output is computed.

The CNN accelerator IP interfaces to a DDR arbiter that enables multiple reads and writes. The IP uses two read channels, one to read the layer inputs and the other to read the network weights. One write channel is used by the IP to write the activations to DDR. The IP expects the input image to be scaled and as per the network input required to be stored in DDR. The scheduler that sequences different layers is configured by the input pins. Typically, a Processor subsystem or UART can be used to generate the data used for configuring the scheduler. The status output represents the number of the layer that the CNN IP is currently running.

Figure 1 • **CNN Accelerator IP Block Diagram**



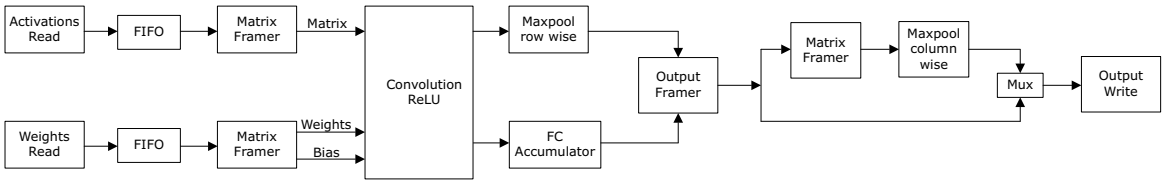
3 Hardware Implementation

This section describes the implementation of the CNN Accelerator IP.

3.1 Design Description

The two DDR read channels **Image Read** and **Weights Read** read the image data and the weights data stored in DDR at a clock frequency of the DDR interface. A CDC FIFO converts the data from the DDR interface clock to the CNN system clock. The matrix frame frames the 3x3 matrix from the image data that will be used for convolution. The matrix framer implements the zero padding and convolution stride. The weight framer loads the weights values of filters used for convolution. The output framer arranges the convolution output into activation maps and stores them in LSRAM. A 3x3 matrix framer frames the matrix with zero padding and stride according to the network layer. The maxpool module finds the maximum of the 3x3 matrix and generates the final output. If a network layer does not use maxpool operation, the output can be directly selected from LSRAM through the multiplexer at the output.

Figure 2 • **CNN Accelerator IP Internal Structure**



The scheduler module controls the sequence of execution of each layer. For every layer, the scheduler provides the DDR address to read the image and weights and address to write the final output of the engine. It also configures the matrix framer for zero padding and stride, the selection of final output through mux. The convolution type - 2D convolution, Depth-wise convolution, and Point-wise convolution are configured through the scheduler. The scheduler data is loaded through the inputs of the IP corresponding to the scheduler.

Types of layers supported by the CNN engine are as follows:

- Convolution - stride1/stride2, Zero padding (5,5,5,5) or No zero padding
  - Kernel size - 3x3, 5x5, 7x7, 9x9
- 3x3 Max pooling - stride1/stride2 after convolution
- Leaky relu after 3x3 convolution
- Relu and Relu Max
- 3x3 Depth wise convolution - stride1/stride2 with zero padding
- Pointwise convolution
- Fully connected
- Global average pooling -7x7

3.2 Memory Components

The CNN Accelerator IP requires the following components to run a network:

- **Network Data**: This defines the structure of the CNN and the DDR memory map of network weights and activations.
- **Weights Data**: This contains the data of weights, biases, scale factors, etc of all the layers of the network.
- **Weights Info**: This contains the details of mapping SPI content of network weights to the DDR memory.

The above three components are generated as a single hex file from the SDK tool flow that can be loaded into the SPI flash.

3.3 Inputs and Outputs

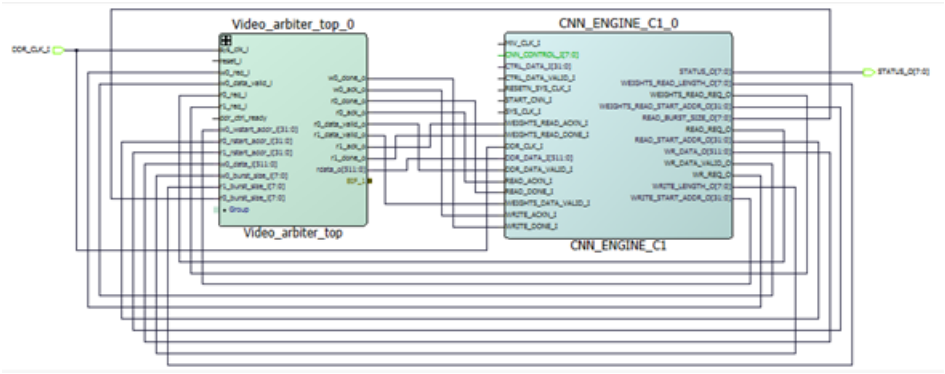
The following table shows the input and output ports of the CNN accelerator IP.

Table 1 • **Input and Output Ports of the CNN Accelerator IP**

Signal Name	Direction	Width	Description
RESETN_SYS_CLK_I	Input	-	Active low synchronous reset signal to design with respect to SYS_CLK_I
SYS_CLK_I	Input	-	System clock
DDR_CLK_I	Input	-	DDR clock
MiV_CLK_I	Input	-	Mi-V clock
CTRL_DATA_I	Input	32 bits	Control data input for scheduler
CTRL_DATA_VALID_I	Input	-	Valid signal for data input to scheduler
START_CNN_I	Input	-	Start signal to run CNN Accelerator for one frame
DDR_READ_CHANNEL1	Bus		Read channel1 bus to be connected to video arbiter for DDR read operation
DDR_READ_CHANNEL2	Bus		Read channel2 bus to be connected to video arbiter for DDR read operation
STATUS_O	Output	7 bits	Status register representing the number of the layer currently running in the CNN Accelerator. The rising edge of STATUS_O(7) denotes completion of one frame by CNN Accelerator.
DDR_WRITE_CHANNEL_O	Bus	-	Write channel bus to be connected to video arbiter for DDR write operation

The interface of the CNN IP with Video arbiter is shown in Figure 3.

Figure 3 • **CNN Accelerator IP interface with Video arbiter**



3.4 Configuration Parameters

The following table shows the description of the configuration parameters used in the hardware implementation of CNN accelerator. These are generic parameters and can be varied as per the requirement of the application.

Table 2 • **Configuration Parameters**

Name	Description
G_PW	Product width or convolution output bit width
G_DWC	Enable to support Depth wise convolution operation
G_MXP_EN	Enable to support Maxpool operation
G_GAVG_POOLING_EN	Enable to support Global average pooling operation

3.5 Timing Diagrams

The following figures show the timing diagrams of read and write channels.

Figure 4 • **Timing Diagram of Read Channel**

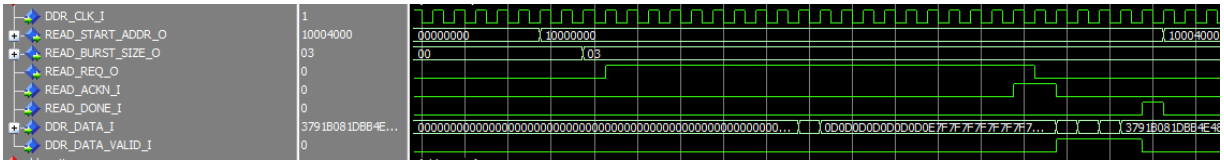
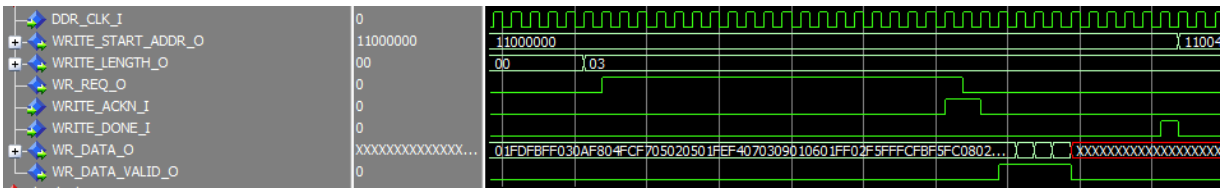


Figure 5 • **Timing Diagram of Write Channel**



3.6 Resource Utilizations

The CNN accelerator IP is implemented on PolarFire FPGA (MPF300T - 1FCG1152E package). The following tables show the resource utilization of CNN Accelerator IP.

Table 3 • **G_PW = 30, G_DWC = 1, G_MXP_EN = 1, G_GAVG_POOLING_EN = 1**

LUT	37840
DFF	34832
MATH	152
LSRAM	116
uSRAM	45

Table 4 • **G_PW = 25, G_DWC = 1, G_MXP_EN = 1, G_GAVG_POOLING_EN = 1**

LUT	36059
DFF	34434
MATH	152
LSRAM	114
uSRAM	45

Table 5 • **G_PW = 30, G_DWC = 0, G_MXP_EN = 1, G_GAVG_POOLING_EN = 1**

LUT	30497
DFF	29856
MATH	152
LSRAM	116
uSRAM	45

Table 6 • **G_PW = 30, G_DWC = 1, G_MXP_EN = 0, G_GAVG_POOLING_EN = 1**

LUT	34260
DFF	32338
MATH	152
LSRAM	95
uSRAM	45

Table 7 • **G_PW = 30, G_DWC = 1, G_MXP_EN = 1, G_GAVG_POOLING_EN = 0**

LUT	36438
DFF	34262
MATH	152
LSRAM	116
uSRAM	0

Table 8 • **Performance and Resource Utilization of the IP for Example Networks**

	Tiny YOLO v2 COCO	**Mobilenet v1**	**Resnet50**
Frames/sec @200 MHz	**15.5 FPS**	**54 FPS**	**7 FPS**
LUT	28642	32330	36059
DFF	29128	31791	34434
MATH	152	152	152
LSRAM	114	93	114
uSRAM	0	45	45

Note: The variation in the resource utilization is achieved by choosing optimal settings of the CNN IP for a particular network. Network latency is 1/FPS; networks are run with a batch size of 1.