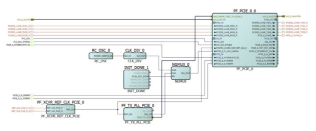
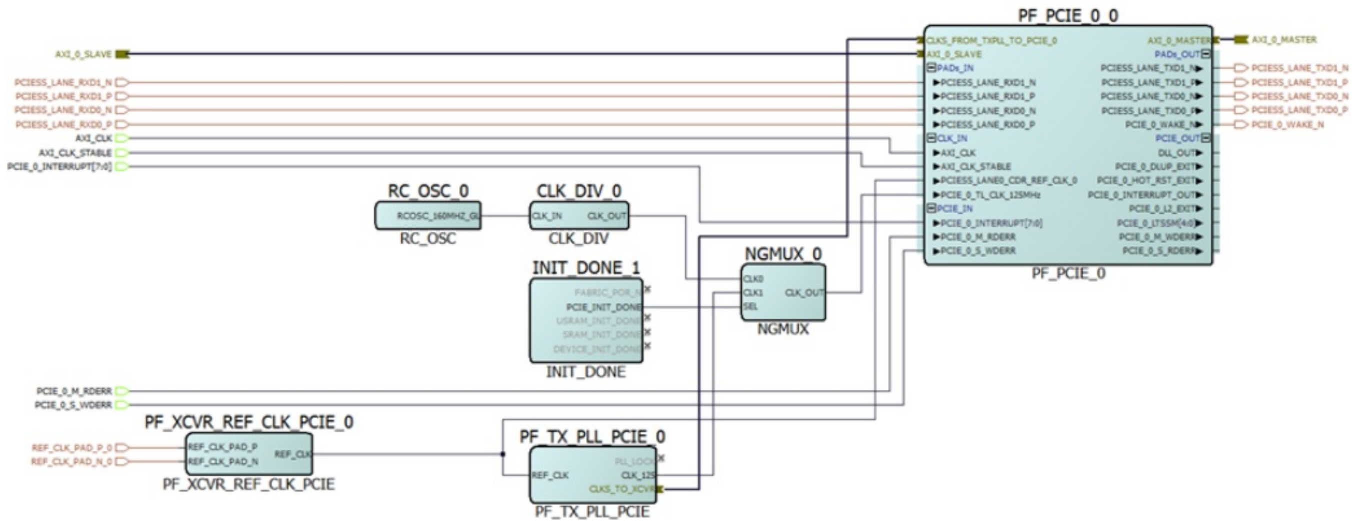


Microsemi UG0943 CNN Accelerator for PolarFire FPGA User Guide

Contents

Microsemi UG0943 CNN Accelerator for PolarFire FPGA



Product Information

The product is the CNN Accelerator for PolarFire FPGA, developed by Microsemi. It is a hardware implementation that provides accelerated processing for Convolutional Neural Networks (CNNs). The CNN Accelerator IP block diagram shows the internal structure of the accelerator, including components such as Activations Read FIFO, Weights Read FIFO, Matrix Framer, Convolution ReLU, Maxpooling, FC Accumulator, Output Framer, and Output Write.

Product Usage Instructions

Hardware Implementation

To implement the CNN Accelerator IP, follow these steps:

1. Ensure you have the necessary hardware components to support the CNN Accelerator for PolarFire FPGA.
2. Connect the necessary inputs and outputs to the accelerator, as described in the Inputs and Outputs section of the user manual.
3. Configure the memory components by specifying the appropriate configuration parameters. Refer to the Configuration Parameters section for details.
4. Refer to the Timing Diagrams section to understand the timing requirements of the accelerator.
5. Monitor the resource utilizations of the accelerator using the Resource Utilizations section.

Supported Layers

The CNN engine of the accelerator supports the following types of layers:

- Convolutional Layers
- ReLU Layers
- Max pooling Layers
- Fully Connected (FC) Layers

For detailed information on each layer type and how to configure them, refer to the Design Description section of the user manual.

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.

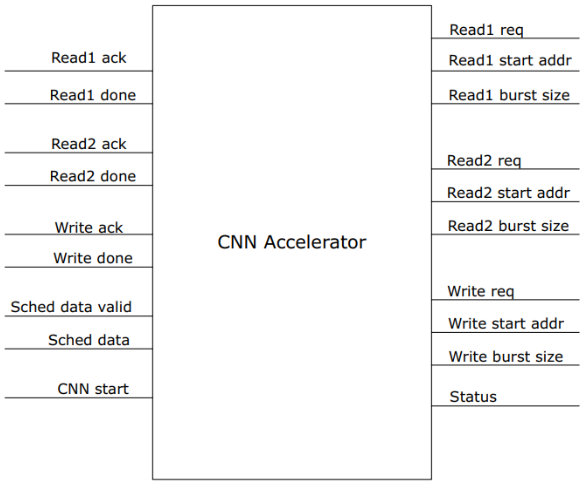
Revision 1.0
The first publication of this document.

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, and 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 layer that the CNN IP is currently running.

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



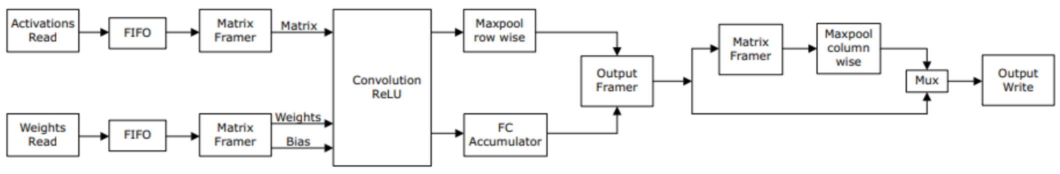
Hardware Implementation

This section describes the implementation of the CNN Accelerator IP.

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 3×3 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 3×3 matrix framer frames the matrix with zero padding and stride according to the network layer. The maxpool module finds the maximum of the 3×3 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 – 3×3, 5×5, 7×7, 9×9
- 3×3 Max pooling – stride1/stride2 after convolution
- Leaky relu after 3×3 convolution
- Relu and Relu Max
- 3×3 Depth wise convolution – stride1/stride2 with zero padding
- Pointwise convolution
- Fully connected
- Global average pooling -7×7

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
- **Weights Info**: This contains the details of mapping SPI content of network weights to the DDR

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

Inputs and Outputs

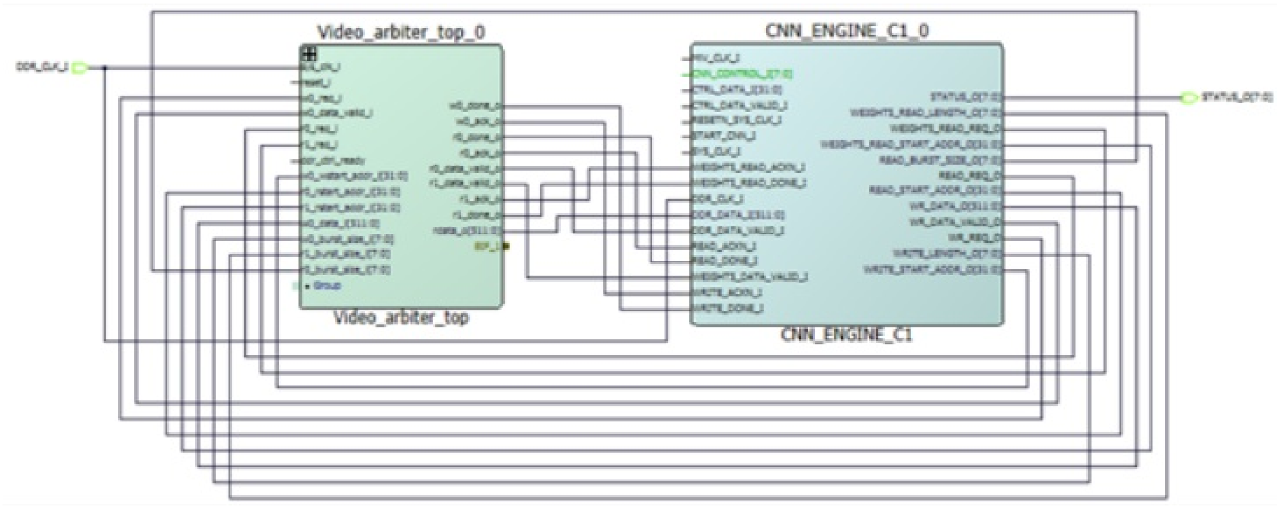
The following table shows the input and output ports of the CNN accelerator IP.

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 t o 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 currentl y running in the CNN Accelerator. The rising edge of STATU S_O(7) denotes completion of one frame by CNN Accelerato r.
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



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 convolution operation
- **G_MXP_EN**: Enable to support Maxpool operation
- **G_GAVG_POOLING_EN:** Enable to support Global average pooling operation

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

Figure 4: Timing Diagram of Read Channel

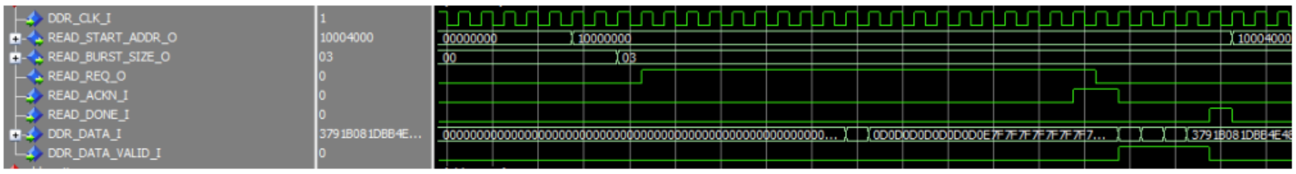
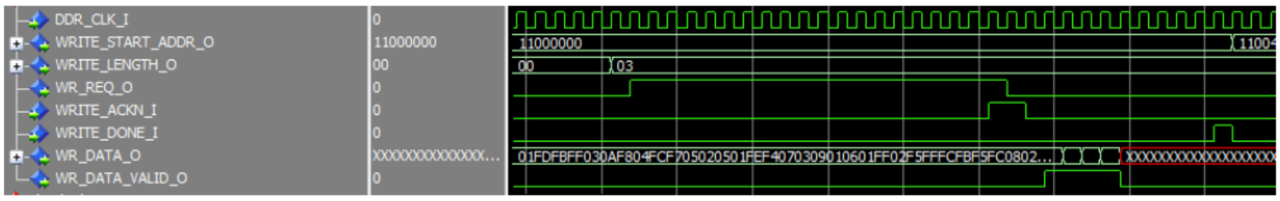


Figure 5: Timing Diagram of Write Channel



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
SRAM	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
SRAM	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	

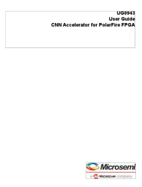
Note: The variation in resource utilization is achieved by choosing the optimal settings of the CNN IP for a particular network. Network latency is 1/FPS; networks are run with a batch size of 1.

Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

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

Documents / Resources

 UG0943 User Guide CNN Accelerator for PolarFire FPGA	Microsemi UG0943 CNN Accelerator for PolarFire FPGA [pdf] User Guide UG0943 CNN Accelerator for PolarFire FPGA, UG0943, CNN Accelerator for PolarFire FPGA, CNN Accelerator, Accelerator

References

- Microsemi \| Semiconductor & System Solutions \| Power Matters