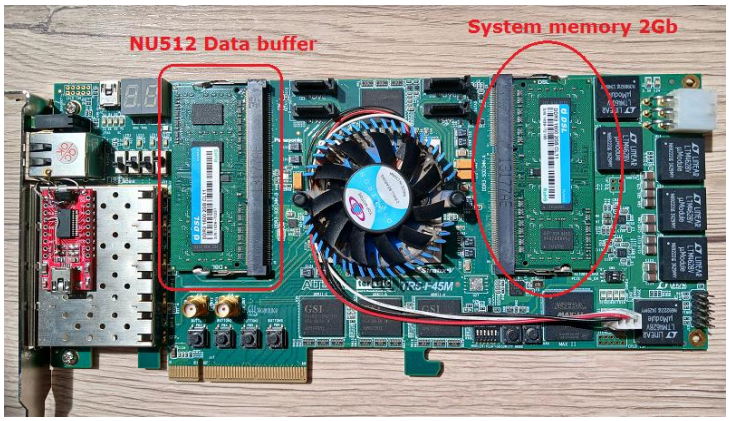
Unit NU512.

The new version of the block was developed to speed up the calculation of convolutions, acceleration of forward and back propagation of the multi-layer perceptron (MLP), edge extraction, pooling. I also wanted to add the ability to work with data in the format with a floating point of half precision - fp16. Now the NU512 block uses 2 data formats fp16 and fp32 (half and single precision). Its architecture was developed in such a way that there was a possibility of practical implementation on the DE5-NET platform with the Stratix V 5SGXEA7N2F45C2 chip and at the same time the architecture had the ability to scale when porting to more capacious and modern platforms.

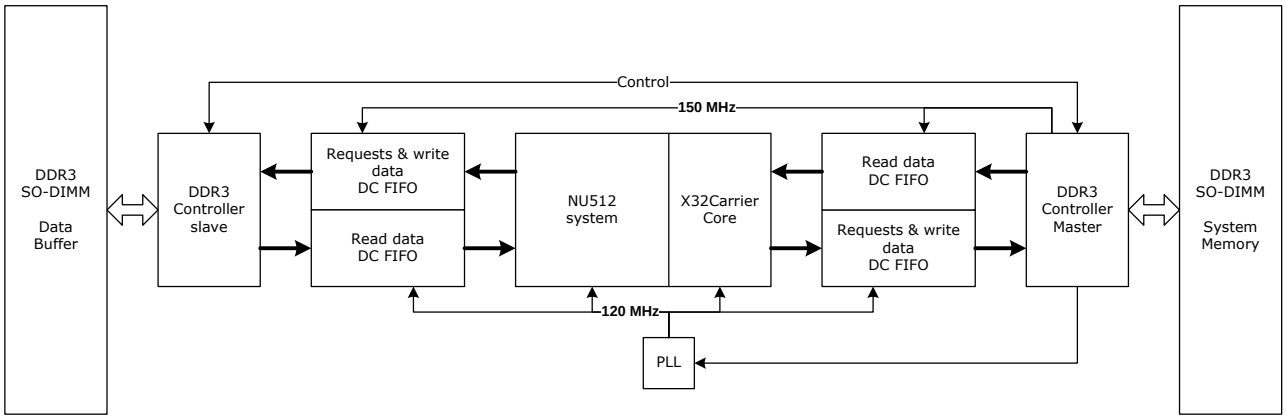
The DE5-NET platform has 2 DDR3 memory channels for 2 SO-DIM modules.



One of the channels was used as the system memory of the X32Carrier core. The second channel was used as data memory for the NU512 block. DDR3 SDRAM Controller with UniPHY v16.1 was used to connect to the memory. Its Avalon-MM interface was configured for Quarter rate mode and 512-bit bus width, operating at 150 MHz. The 5SGXEA7N2F45C2 crystal resources were used at 95%.

```
+-----------------------------------------------------------------------------+
; Fitter Summary                                                              ;
+---------------------------------+-------------------------------------------+
; Fitter Status                   ; Successful - Thu Apr 24 04:58:19 2025       ;
; Quartus Prime Version           ; 16.1.2 Build 203 01/18/2017 SJ Standard Edition ;
; Revision Name                   ; X32CsLNU512                               ;
; Top-level Entity Name           ; TOP                                       ;
; Family                          ; Stratix V                                 ;
; Device                          ; 5SGXEA7N2F45C2                            ;
; Timing Models                   ; Final                                     ;
; Logic utilization (in ALMs)     ; 223,197 / 234,720 ( 95 % )                ;
; Total registers                 ; 178632                                    ;
; Total pins                      ; 286 / 1,064 ( 27 % )                      ;
; Total virtual pins              ; 0                                         ;
; Total block memory bits         ; 39,633,248 / 52,428,800 ( 76 % )          ;
; Total RAM Blocks                ; 2,560 / 2,560 ( 100 % )                   ;
; Total DSP Blocks                ; 249 / 256 ( 97 % )                        ;
; Total HSSI STD RX PCSs          ; 0 / 48 ( 0 % )                            ;
; Total HSSI 10G RX PCSs          ; 0 / 48 ( 0 % )                            ;
; Total HSSI GEN3 RX PCSs         ; 0 / 48 ( 0 % )                            ;
; Total HSSI PMA RX Deserializers ; 0 / 48 ( 0 % )                            ;
; Total HSSI STD TX PCSs          ; 0 / 48 ( 0 % )                            ;
; Total HSSI 10G TX PCSs          ; 0 / 48 ( 0 % )                            ;
; Total HSSI GEN3 TX PCSs         ; 0 / 48 ( 0 % )                            ;
; Total HSSI PMA TX Serializers   ; 0 / 48 ( 0 % )                            ;
; Total HSSI PIPE GEN1_2s         ; 0 / 48 ( 0 % )                            ;
; Total HSSI GEN3s                ; 0 / 48 ( 0 % )                            ;
; Total PLLs                      ; 2 / 92 ( 2 % )                            ;
; Total DLLs                      ; 1 / 4 ( 25 % )                            ;
+---------------------------------+-------------------------------------------+
```

Such high resource utilization did not allow me to make the 150MHz clock, which synchronizes Avalon-MM interfaces of DDR3 controllers, the main clock signal of the entire system. I had to lower the main clock of the system to 120MHz and use Dual-clock FIFO to connect the system with two DDR3 controllers.



Operations performed by the NU512

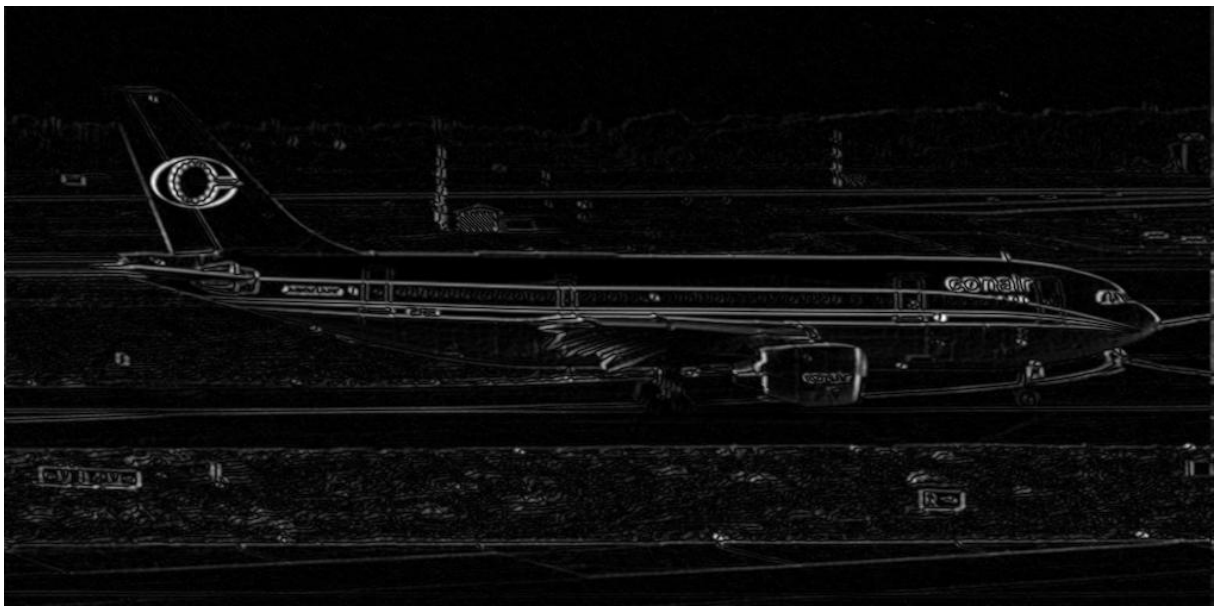
- Loading data from system memory into the data buffer. The source data can be in the following formats: uint8, uint16, int8, int16, fp16, fp32;
- Unloading data from the data buffer into system memory with translation from fp16 or fp32 format into int8, int16, fp16, fp32 formats;
- Convolution. Kern sizes: 3*3, 5*5 and 7*7 Convolution step 1*1;
- Edge extraction using a combination of the horizontal Sobel operator and the vertical Sobel operator.
- Pooling with a step of 2*2, by minimum, maximum and average values.
- Forward propagation of a MLP, a set of 8 widely used activation functions.
- Back propagation (training) of a MLP.

NU512 test results.

The system had a main clock 120МГц. To test the operations of data loading/unloading, convolutions, edge detection and pooling, a grayscale 8-bit photograph with a size of 1024*512 pixels was used.

It was processed with a 7x7 kern, designed to reduce noise.

After noise suppression, an edge detection operation was applied to the image.



And the last stage is checking the pooling. The image was reduced by 4 times in area.

Performance of convolutions, edge detection and pooling.

Operation	***FLOPS***	***FLOps/clock***	***Pixels/clock***
Convolution 3*3 fp16	9 171 659 248.52	76.43	4.496
Convolution 5*5 fp16	8 854 655 785.73	73.79	1.506
Convolution 7*7 fp16	12 654 934 022.24	105.46	1.087
Convolution 3*3 fp32	5 224 082 236.51	43.534	2.5608
Convolution 5*5 fp32	8 416 707 551.55	70.139	1.431
Convolution 7*7 fp32	12 193 045 812.45	101.61	1.0475
Edge detection	10 718 914 845.51	89.32	4.466
Pooling average fp16	1 346 591 865.36	11.22	3.74
Pooling average fp32	579 170 562.51	4.826	1.609

How can we use the data written in the last column of the table? For example, let's calculate the execution time of the pooling operation on an array of 1016*504. Data format fp32. The resulting array will have a dimension of 508*252=128016 pixels. The number of clocks that the block will spend on this operation will be equal to 128016/1.609=79562.46. If the clock is equal to 150 MHz, then the execution time of such a pooling operation will be 79562.46/150e6=530.42 μs.

Data transfers performance.

The maximum data transfer rate is limited to 480e6 bytes/sec with the main clock at 120 MHz. This is due to the fact that the NU512 block has a limitation on the maximum data exchange rate with the system memory. Access to the system memory is possible once every 2 clock cycles over a 64-bit channel. This is done specifically to prevent the channel from being blocked for the processor core, allowing it to work in parallel with the NU512 equipment.

So, Max.transfer rate = 8(bytes)*Fclk/2➔ 8*120e6/2=480e6 bytes/sec.

The Pixel/clock parameter allows you to calculate the time or data transfer rate at a different main clock frequency than 120 MHz.

Data format	***Bytes/sec***	***Pixels/sec***	***Pixel/clock***
Loading data from system memory into the data buffer.			
uint8→fp32	346 957 845.28	346 957 845.28	2.89
fp32→fp32	331 094 411.11	82 773 602.78	0.69
fp16→fp16	326 100 278.55	163 050 139.27	1.359
Transfer from data buffer to system memory			
fp32→fp32	479 432 528.88	119 858 132.22	0.999
fp32→int16	479 437 566.91	239 718 783.46	1.998
fp16→fp32	478 191 607.87	119 547 901.97	0.996
fp16→fp16	479 730 185.50	239 865 092.75	1.999

Test of the MLP.

For testing, the EMNIST-Digits dataset was chosen, which contained 240,000 training images and 40,000 test images. The images looked something like this:

These are mirror images of handwritten digits along the vertical axis, rotated 90 degrees counterclockwise, but this does not matter for MLP training. The image has a dimension of 28*28. The MLP was chosen in the configuration:

- Input layer – 784 elements in fp16 format (uint8→fp16 when loading the image into the data buffer);
- The inner layer had 2048 neurons, each with 784 inputs;
- The output activation function was chosen to be tangent hyperbolic, the result was presented in fp32 format;

- Output layer of 10 neurons, each of which had 2048 inputs;
- Output activation function is sigmoid;
- The data was fed to the MLP input without any pre-processing at all.

10 neurons of the last layer == 10 digits, 0-9. The perceptron response to the action was selected by the maximum value at the output of one of the 10 neurons. The result of the MLP operation in the form of a set of 10 numbers in fp32 format was unloaded into the system memory, where the numbers were processed programmatically, using the standard set of machine instructions of the X32Carrier core, and a decision was made on which digit was depicted. For example, if neuron 3 produced the largest number, then the presented data set was considered an image of the digit 3. MLP training was carried out in the simplest way:

- The MLP was presented with the next image of a number and the forward propagation was calculated;
- the calculation result was unloaded into the system memory;
- the errors of the output layer were calculated: MLP Error = (expected value 0 or 1) - (MLP response) Thus, neurons whose indices were not equal to the true value of the digit in the presented image received negative error values, and neurons whose indices were equal to the value of the depicted digit received a positive error value;
- the set of errors was loaded into the MLP and the backpropagation procedure (training) was started;
- the cycle was repeated over all 240,000 training images, and once one cycle was completed, the next one was started from the very beginning of the dataset.

During this infinite learning cycle, the correctness of the direct propagation result was assessed each time. And if after the next direct propagation it was found that the maximum number of correct results had been obtained for the last 1000 images, then the current MLP configuration was saved, and the training continued. Thus, after a short time (about 30 minutes), a configuration of MLP coefficients was formed that theoretically gave the best recognition result.

As a result of such training, a set of coefficients was obtained that allowed us to obtain a probability of correct answers of 94.48% on the test set.

Fashion recognition.

The second experiment with MLP was class recognition of clothing, shoes and accessories.

The perceptron configuration was the same. 784 input elements - 1st layer, 2048 neurons in the hidden layer and 10 neurons in the output layer. The hidden layer also had the TanH activation function, and the output - sigmoid.

Result: 79.44% correct answers.

MLP Performance.

Number of elementary operations (+ - * /) in direct propagation:
2048*(784*2+28)+10*(2048*2+26)=3309828, where:

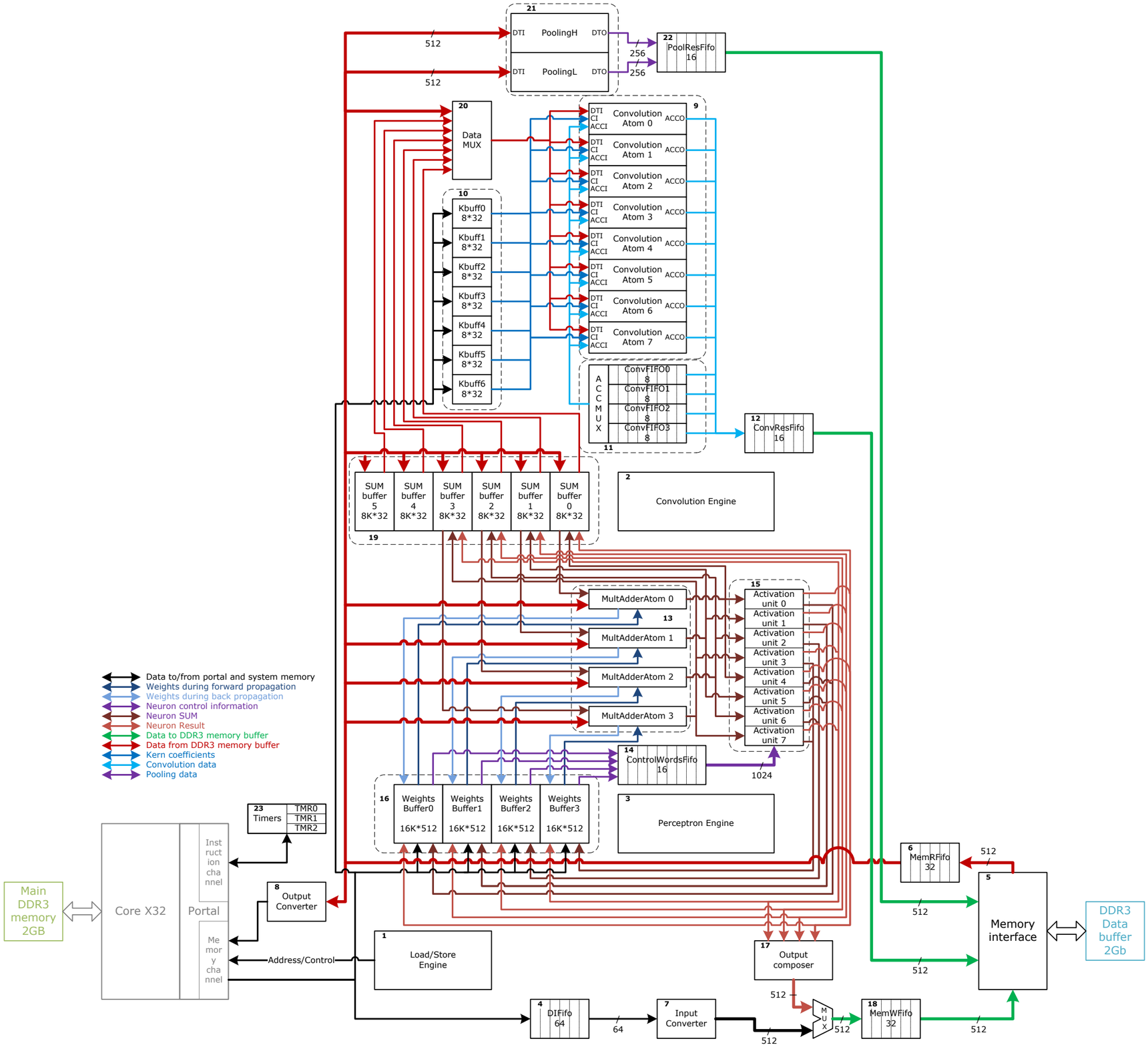
- 2048 – number of neurons in the hidden layer;
- 784*2 – the number of operations of multiplying input data by weight coefficients and summing products and fixed offset;
- 28 – the number of elementary operations in calculating the activation function TanH;
- 10 – number of neurons in the output layer;
- 2048*2 – the number of operations of multiplication of the results of the previous layer and operations of summation of products and fixed offset;
- 26 – the number of elementary operations performed when calculating the sigmoid activation function.

This number of elementary floating point operations was completed in 0.0002278 sec, which corresponds to the performance:
3309828/0.0002278=14529534679.54 FOPS or
14.53 GFLOPS
121.079 FOPS/clock

Number of operations in backpropagation:
2048*(10+9+1+1+2+784*2)+10*(1+1+2+2048*2)= 3299368, where:
2048 – number of neurons in the hidden layer;
10+9 – the number of operations required to calculate the error value of each of the 2048 neurons;
+1+1 – multiplying the error by the learning rate and multiplying by the derivative of the output function;
+2 – number of operations when calculating the derivative of the output function;

784*2 и 2048*2 – the number of operations of multiplying the scaled error by the value of the input data value and then summing it with the previous weighting factor.
This number of operations was completed in 0.000150292 sec., which corresponds to the performance of:
3299368/0.000150292=21953051393.2877332 FOPS or
21.95 GFLOPS
182.942 FOPS/clock

Architecture of the NU512.


21
DTI
PoolingH
DTO
DTI
PoolingL
DTO
512
512
256
256
22
PoolResFifo
16
20
Data
MUX
10
Kbuff0
8*32
Kbuff1
8*32
Kbuff2
8*32
Kbuff3
8*32
Kbuff4
8*32
Kbuff5
8*32
Kbuff6
8*32
9
DTI
CI
ACCI
Convolution
Atom 0
ACCO
Convolution
Atom 1
Convolution
Atom 2
Convolution
Atom 3
Convolution
Atom 4
Convolution
Atom 5
Convolution
Atom 6
Convolution
Atom 7
A
C
C
M
U
X
ConvFIFO0
8
ConvFIFO1
8
ConvFIFO2
8
ConvFIFO3
8
11
12
ConvResFifo
16
SUM
buffer
5
8K*32
SUM
buffer
4
8K*32
SUM
buffer
3
8K*32
SUM
buffer
2
8K*32
SUM
buffer
1
8K*32
SUM
buffer
0
8K*32
19
2
Convolution Engine
15
MultAdderAtom 0
13
MultAdderAtom 1
MultAdderAtom 2
MultAdderAtom 3
Activation
unit 0
Activation
unit 1
Activation
unit 2
Activation
unit 3
Activation
unit 4
Activation
unit 5
Activation
unit 6
Activation
unit 7
14
ControlWordsFifo
16
1024
16
Weights
Buffer0
16K*512
Weights
Buffer1
16K*512
Weights
Buffer2
16K*512
Weights
Buffer3
16K*512
3
Perceptron Engine
Data to/from portal and system memory
Weights during forward propagation
Weights during back propagation
Neuron control information
Neuron SUM
Neuron Result
Data to DDR3 memory buffer
Data from DDR3 memory buffer
Kern coefficients
Convolution data
Pooling data
23
Timers
TMR0
TMR1
TMR2
Main
DDR3
memory
2GB
Core X32
Portal
Inst
ruct
ion
cha
nnel
Me
mor
y
cha
nnel
8
Output
Converter
Address/Control
1
Load/Store
Engine
4
DIFifo
64
64
7
Input
Converter
512
17
Output
composer
512
M
U
X
512
18
MemWFifo
32
512
6
MemRFifo
32
512
512
5
Memory
interface
DDR3
Data
buffer
2Gb

1. Load/Store engine. The engine performs NULD, NUSD, NULP, NUSP operations and loading of error values in the perceptron backpropagation instruction - NUBP.
2. Convolution engine. Designed to perform convolution, pooling and edge detection. Executes NUCN, NUPL instructions.
3. Perceptron engine. Controls the operation of the perceptron in forward and backward propagation, NUFP and NUBP instructions.
4. Input queue. Used for temporary storage of data read from system memory, intended for transmission to an external data buffer.
5. Interface to DDR3 memory module. DDR3 memory module is used as external data storage buffer. Avalon-MM interface of DDR3 controller is 512 bits wide and uses Quad Rate mode for Avalon-MM interface.
6. Read data queue. Contains 512-bit data sets consisting of either 32 elements in fp16 format or 16 elements in fp32 format. This queue contains data intended for transfer to system memory, for performing convolution, pooling, and for performing forward and backward perceptron propagation.
7. Input converter. Converts data received from system memory to fp16 or fp32 format before writing to the external data buffer. Input data can be in uint8, uint16, fp16, fp32, int8, int16 formats.
8. Output converter. Used when unloading data from the buffer memory to the system memory. The converter can save data in the int8, int16, fp16 and fp32 formats.
9. Set of 8 blocks for calculating convolutions. These 8 blocks are capable of calculating 16 convolutions of 3*3 or 8 convolutions of 5*5 and 7*7.
10. Kern Buffers. 7 buffers contain kernel coefficients 3*3, 5*5 and 7*7.
11. Intermediate data storage queues and intermediate results multiplexer. The queues store the sums accumulated during processing of the next initial 512-bit data set. The same 512-bit data set coming from queue 6 contains operands for different kern rows. For example, with a kern size of 3*3, each initial data set is processed by 3 kern rows. The result of processing the 0th row is written to queue 11. The intermediate result of processing the previous data set is extracted from queue 11, summed with the result of processing the 1st row of the kern, and placed back in queue 11. The intermediate result of processing the two previous data sets is summed with the partial sum obtained during processing of the 3rd row of the kern and the current data set. The result of the sum is no longer written to the queue, but is transferred to queue 12 for subsequent writing to the buffer memory as a result.
12. Convolution Result Queue: This queue accumulates convolution results before writing them to the data buffer.
13. MultAdderAtoms are blocks that perform multiplication of input data by weight coefficients and summation of the obtained products. MLP is used in direct and reverse propagation. During reverse propagation, the blocks allow calculating neuron errors by multiplying the error of the neuron from the next layer by the corresponding weight coefficient and summing with the accumulated error. Each of the four blocks has 32 multipliers, which is designed for parallel multiplication of 32 pairs of

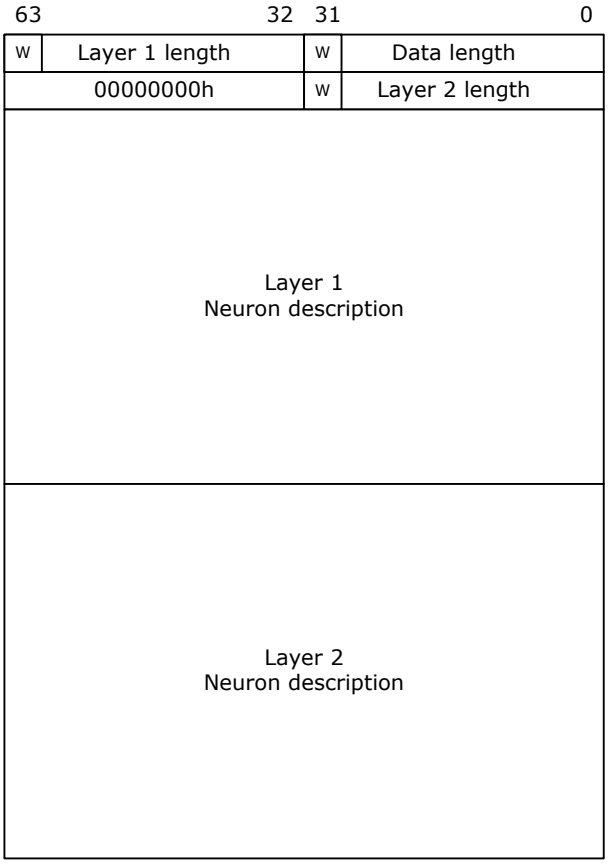
data and weight coefficients if the data is presented in the fp16 format. If the fp32 format is used, only 16 multipliers work. This allows processing the entire 512-bit data word in 1 cycle. The results of the multiplications are summed on the pipeline over several cycles. The total pipeline length of the MultAdderAtom block is 10. For a 512-word at the ReadMemFifo output, at each clock cycle the coefficients of the next neuron are selected from the coefficient buffer and a data pair is fed to the MultAdderAtom. When the coefficient groups for all neurons of the layer are passed through the MultAdderAtom, the processed data word leaves the input queue and the processing of the next 512-bit word and the corresponding coefficients begins. If the last set of coefficients is processed, the resulting sum is fed to the inputs of the activation blocks simultaneously with the neuron control parameters, the neuron parameters pass through the ControlWordsFifo queue.

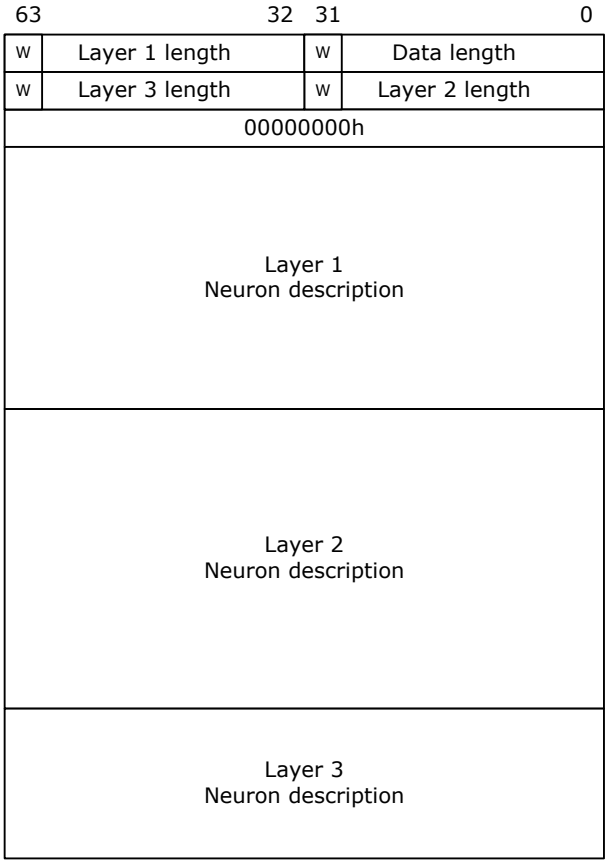
14. ControlWordsFifo is used to synchronize the simultaneous supply of the accumulated sum value and neuron parameters to the inputs of the activation blocks.
15. Activation blocks calculate the output functions of neurons. Each block is capable of calculating up to 4 different functions in parallel. In total, 8 blocks can calculate up to 32 activation functions in parallel. Each activation block contains an adder, a multiplier, and a divider. These blocks perform calculations in the fp32 floating-point format and have 3 pipeline levels, which, in addition to the level of the source operand multiplexer, form 4-level pipelines on which data of 4 output functions being calculated can simultaneously be located. The "e^x" function is calculated using a tabular method using 24 multiplication operations. The principle is simple: if the mantissa bit of the number x is 0, then the generated value is multiplied by 1, if the bit is 1, then the result is multiplied by the corresponding coefficient from the table. Activation blocks calculate the output functions during forward data propagation and calculate the derivatives of the activation functions during backward propagation. The table below shows the formulas for the functions in accordance with which the calculations are made. The natural logarithm is calculated using the series: $\ln(x) = 2 * [\frac{x-1}{x+1} + \frac{1}{3} * \left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5} * \left(\frac{x-1}{x+1}\right)^5 + \frac{1}{7} * \left(\frac{x-1}{x+1}\right)^7 + \cdots]$ The values 1/3, 1/5, 1/7, 1/9, 1/11, 1/13, 1/15 and 1/17 are retrieved from ROM.
16. The results of the activation functions and the values of the sums of the weighted input data are written to the coefficient buffers. They will be used if the backpropagation operation is started after the forward propagation. The activation blocks do not always produce results simultaneously, since different activation functions can be used or the formulas can use the division operation, the execution time of which depends on the number of 1's in the mantissas of the operands.
17. The results of the activation functions are passed to the input of the OutComposer result collector, which waits for the next 512-bit word to be filled and passes it to the write queue.
18. The write queue accumulates data and it is transferred to the memory controller in burst mode.

19. Sum Buffers in the amount of 6 pcs are used during convolution calculation, perceptron forward propagation and perceptron backpropagation. During convolution calculation, these buffers are used to store the last few elements of the current 512-bit word of the current matrix column's raw data in order to use them as raw data when processing the next matrix column. If the core size is 3*3, then only buffers 5 and 4 are used (4 is used when the data width is fp32), if the core size is 5*5, then buffers 5 and 4 are used if the data format is fp16 and buffers 5,4,3 and 2 if the data format is fp32. When using the 7*7 core, all 6 buffers are used if the data format is fp32. Only 4 buffers, 0-3, are used when calculating MLP. When calculating forward propagation, the buffers are used to temporarily store intermediate values of the sums of products of weight coefficients and data. When calculating backpropagation, buffers are used to store the derivatives of the activation functions multiplied by the neuron's error rate and the learning rate. These values are used later when recalculating the weights.
20. The data multiplexer prepares the source data for the convolution blocks depending on the data format and the kern size.
21. Two pooling units process two 512-bit words and form a single 512-bit result.
22. The pooling result is placed in PoolResFifo, from which the data is transferred for writing to DDR3 data buffer.
23. The timers count the running time of the Load/Store Engine, Convolution Engine, and Perceptron Engine in 200ns ticks. The contents of the timers can be read into a general-purpose register. The timers are reset to 0 by software.

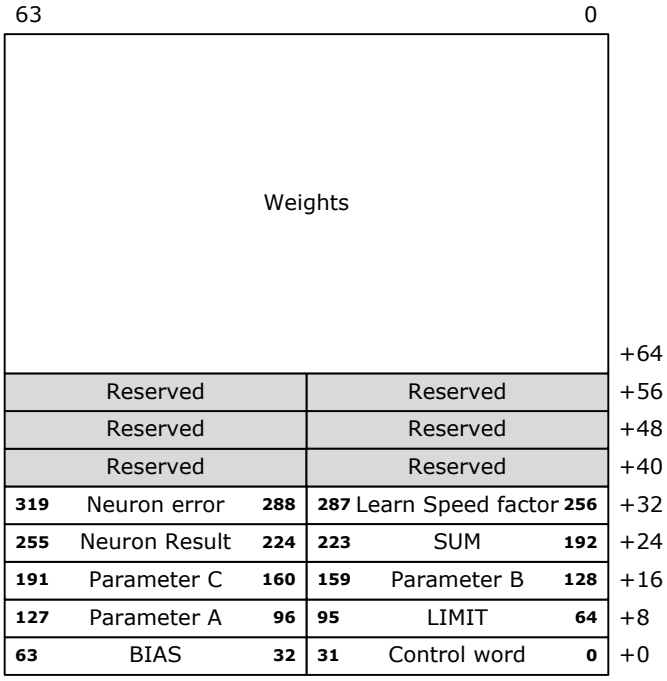
MLP control block format.

The header contains the description of the perceptron layers. Each 32-bit word contains a counter of data elements and a bit of data size. The data size bit is placed in bit 31, 0 – fp16, 1 – fp32. The description of the layers ends with a zero 32-bit word. Next, the neuron parameters are placed. The parameters are aligned to the boundary of 64-bit Qwords. If the description of the MLP layers contains an even number of layers, then after the zero termination word, another 32-bit word is placed for alignment to the boundary of 8 bytes.





Each neuron is described by a block of parameters followed by a set of coefficients.



The number of coefficients is equal to the length of the previous layer, and their format is equal to the format of the data of the previous layer. The neuron parameter block contains:

- The control word contains only 3 significant bits. Bits [2:0] define the activation function according to the table:

[2:0]	Function	Forward propagation	Derivative
0	Limit detector, ReLU, Leak ReLU, Linear	f(x)=C+A(x-limit), if x>=limit f(x)=B(x-limit), if x<limit	f'(x)=A, if x>=limit f'(x)=B, if x<limit
1	Softsign	f(x)=x/(|x|+1)	f'(x)=1/((|x|+1)^2)
2	ELU	f(x)=A(x-limit), if x>=limit f(x)=B(e^(x-limit)-1), if x<limit	f'(x)=A, if x>=limit f'(x)=f(x)+B, if x<limit
3	Sigmoid	f(x)=1/(1+e^-x)	f'(x)=f(x)*(1-f(x))
4	TanH	f(x)=2/(1 + e^(-2x)) – 1	f'(x)=1-(f(x))^2
5	Softplus	f(x)=ln(1+e^x)	f'(x)=1/(1+e^-x)
6	Swish	f(x)=x/(1+e^-x)	f'(x)=((e^-x)*(x + 1) + 1)/((1+e^-x)^2)
7	Gaussian	f(x)=e^-(x^2)	f'(x)=-2xe^-(x^2)

- Limit, Parameter A, Parameter B, Parameter C – activation function parameters;
- SUM – the value of the sum of the weighted input data of a neuron. This value is the initial operand of the activation function;
- Neuron result – the value of the activation function result;
- Learning speed – learning speed;
- Neuron error – neuron error value. For the last layer, the error value is loaded from system memory at the start of the NUBP instruction. For the remaining layers, the error values are calculated by NU512.

Weights can be in 16-bit or 32-bit format.

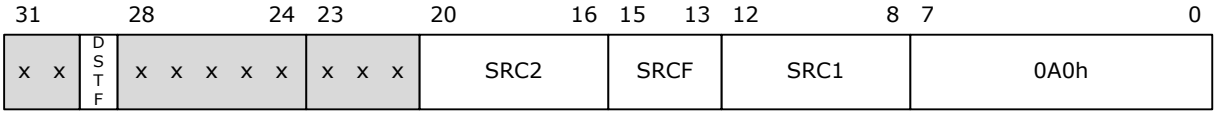
NU512 Instructions

NULD, NeuroUnit Load Data.

Mnemonic:

NULD SRC1,SRC2,SRCF,DSTF

Format:



Example:

nuld r4,r23,uint16,fp32

Description:

Instruction for loading data from system memory into the data buffer.

Instruction parameters:

- SRC1[31:0] – selector of the object in which the data is placed starting from offset zero.

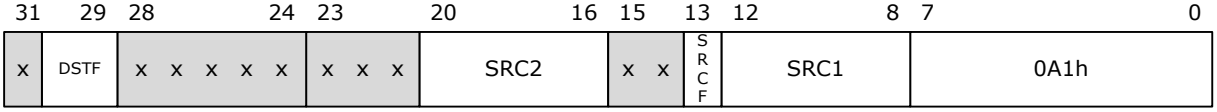
- SRC1[56:32] – address of loading data inside the data buffer. The value defines the address of a 512-bit cell.
- SRC2[31:0] – number of data elements in a row.
- SRC2[63:32] – number of rows, reduced by 1. If SRC2[63:32]=0, then there is only one row. This property allows loading both one-dimensional data blocks and two-dimensional arrays into the data buffer.
- SRCF – format of the source data: uint8 – unsigned 8 bits, uint16 – unsigned 16 bits, fp16 – floating point 16 bits, fp32 – floating point 32 bits, int8 – signed number 8 bits, int16 – signed number 16 bits.
- DSTF – format in which the data will be written to the buffer. Can be either fp16 or fp32.

NUSD, NeuroUnit Store Data.

Mnemonic:

NUSD SRC1,SRC2,SRCF,DSTF

Format:



Example:

nusd r5,r2,fp16,int16

Description:

The instruction unloads data from the data buffer into the system memory. Data from the internal fp16 or fp32 format is converted to the receiver format. The receiver format can be int8, int16, fp16 or fp32. This instruction does not output data in unsigned format. Instruction parameters:

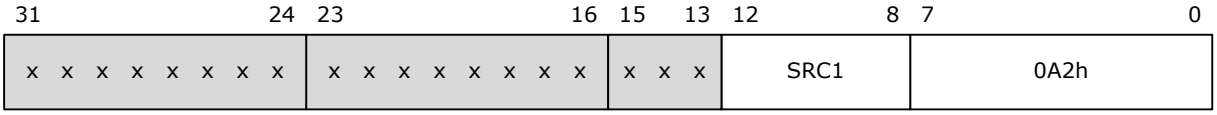
- SRC1[31:0] – selector of the object into which data is written, starting at offset zero;
- SRC1[56:32] – address of data placement in the buffer;
- SRC2[31:0] – number of data elements in each row;
- SRC2[63:32] – number of lines minus 1. If SRC2[63:32]=0, then only one line of data is transmitted;
- SRCF – format of the source data in the buffer. Can be either fp16 or fp32;
- DSTF – The format of the data in the destination object. Can be int8, int16, fp16, or fp32.

NULP, NeuroUnit Load Percetron

Mnemonic:

NULP SRC1

Format:



Example:

nulp r20

Description:

The instruction loads the perceptron data into the internal buffers of the NU512 block. The instruction parameter is a selector in the SRC1 register. The

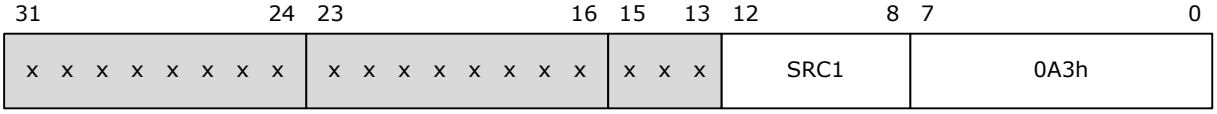
selector points to the object in which the perceptron data is located starting from offset zero.

NUSP, NeuroUnit Store Perceptron

Mnemonic:

NUSP SRC1

Format:



Example:

nusp r3

Description:

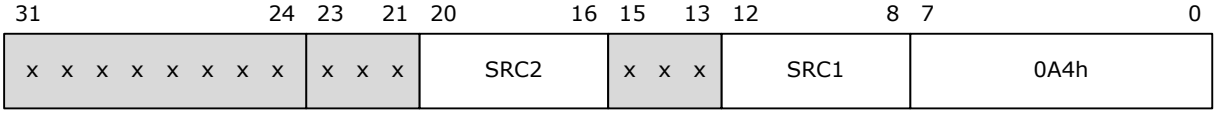
The instruction unloads the perceptron data into an object located in system memory. SRC1 contains the object selector.

NUBP, Neuro Unit Back Propagation

Mnemonic:

NUBP SRC1,SRC2

Format:



Example:

nubp r23,r17

Description:

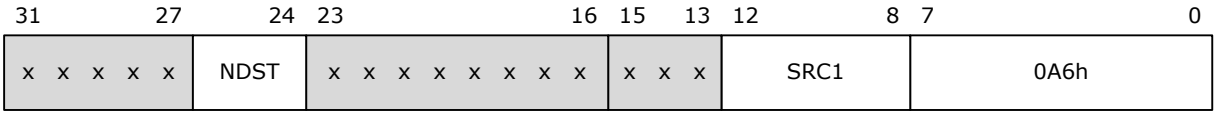
The instruction initiates loading of the error block into the control parameters of the last layer of neurons. After loading the errors, the backpropagation procedure begins. During this procedure, a sequential calculation of errors is performed for all neurons of all preceding layers, starting from N-1 to 1. After completing the calculation of errors for all neurons in all layers, the weight coefficients are updated for all neurons in all layers.

NUWR, NeuroUnit Write Register

Mnemonic:

NUWR NDST,SRC1

Format:



Example:

nuwr 0,r14

Description:

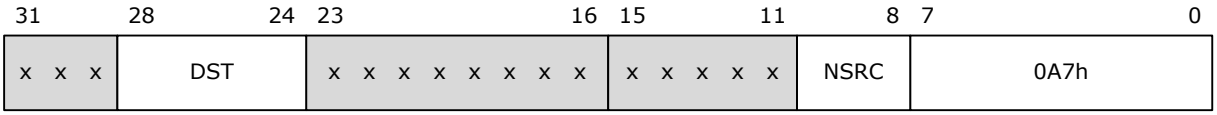
The instruction writes data to one of the NU512 control registers. In the current version of the block, writing any value to register 0 results in zeroing the TMR0, TMR1, TMR2 counters.

NURR, NeuroUnit Read Register

Mnemonic:

NURR DST,NSRC

Format:

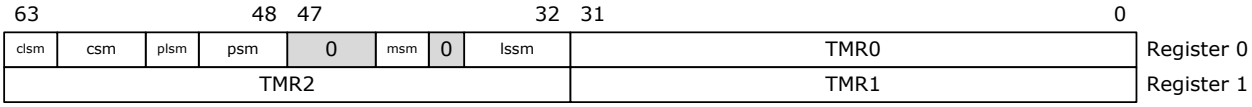


Example:

nurr r28,1

Description:

Reading a register. In the current version, only 2 registers are available: 0 and 1. Their format is shown in the figure below.



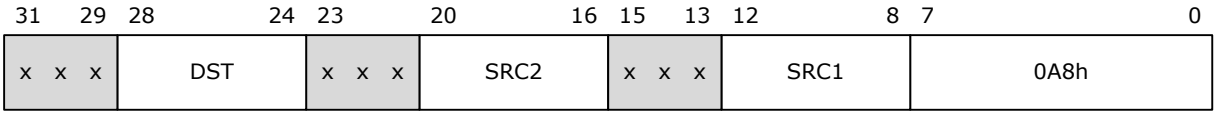
Register 0 contains the TMR0 timer and the state machine fields. TMR0 – counts 200ns ticks of the data load/store engine. TMR1 – counts ticks of convolution, pooling and edge detection. TMR2 counts ticks of forward and backward propagation MLP.

NUCN, NeuroUnit CoNvolution.

Mnemonic:

NUCN SRC1,SRC2,DST

Format:

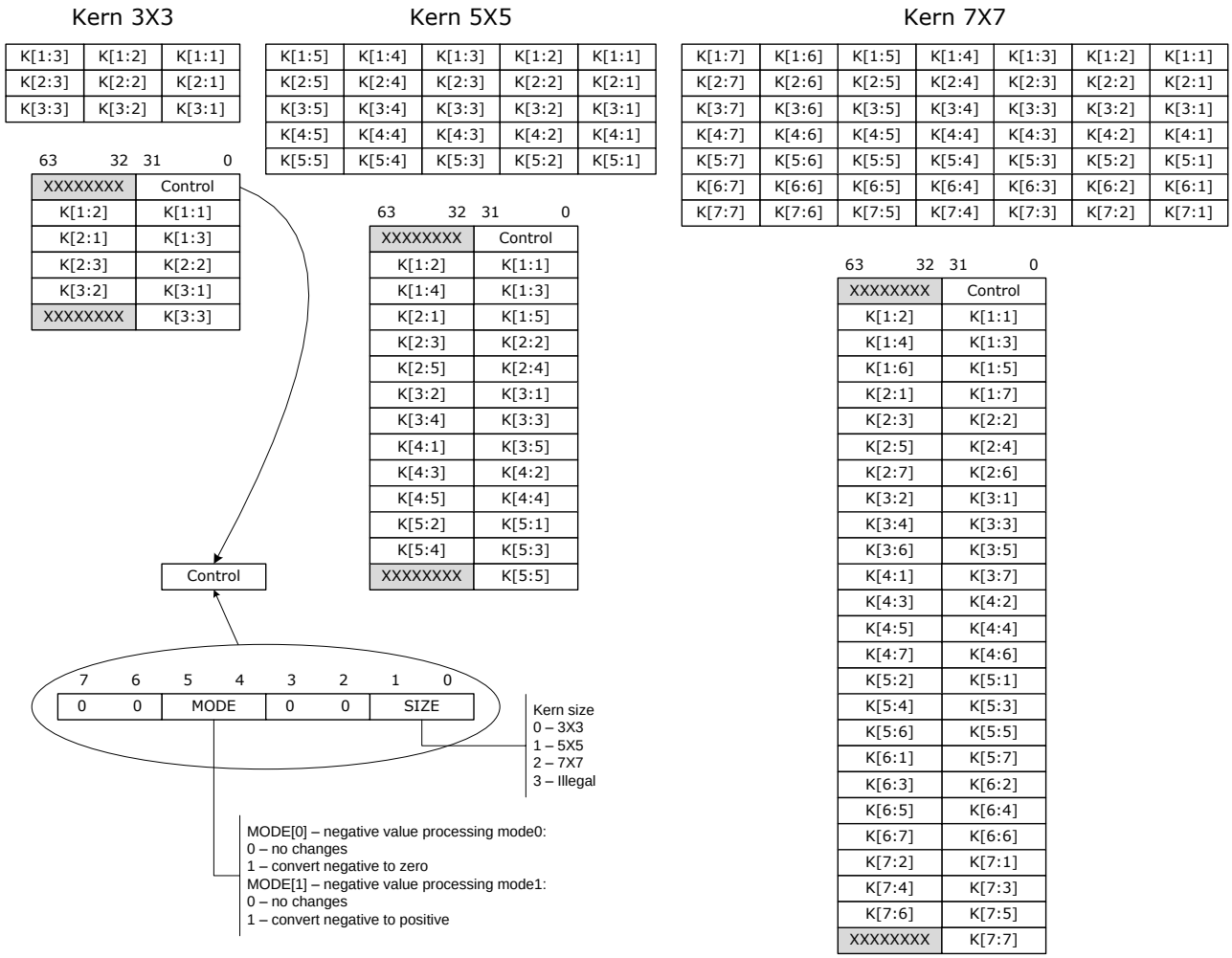


Example:

nucn r3,r29,r6

Description

The instruction starts the calculation of the convolution of a two-dimensional matrix with a 3*3 or 5*5 or 7*7 kern size. The register specified in the SRC1 field contains the address of the source data location in the data buffer (bits [24:0]), the horizontal width of the matrix (bits [43:32]) and the height of the matrix (bits [59:48]). Bit 62 of SRC1 is the data format 0➔fp16 and 1➔fp32, bit 63 is the edge detection mode control, if SRC1[63]=0 – convolution is performed, if SRC1[63]=1 – edge detection is performed. The SRC2 register contains the offset (bits [31:0]) and the object selector (bits [63:32]) that determine the placement of the kern data. The kern data has the format shown in the figure below.



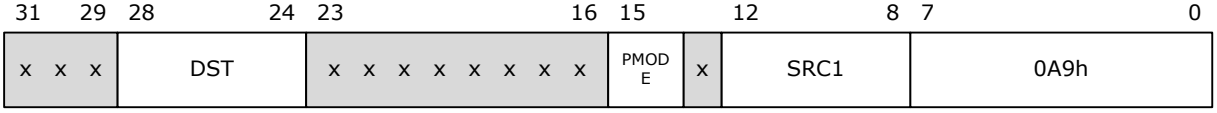
The DST register in bits [24:0] contains the address of the data buffer where the result of the convolution or edge detection will be placed.

NUPL, NeuroUnit PooLing

Mnemonic:

NUPL SRC1,DST,Pmode

Format:



Example:

nupl r13,r14,pmin

Description:

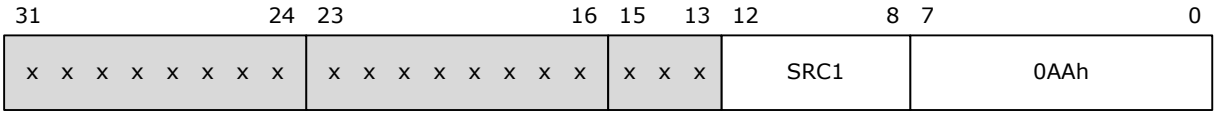
Pooling. The pooling operation can be performed in 3 modes: Pmode=0➔pmin – selects minimum values, Pmode=1➔pmax – selects maximum values, Pmode=2➔pavr – calculates the average value for 4 points. The SRC1 register contains the address of the initial data in the buffer (bits [24:0]), the horizontal length of the matrix (bits [43:32]), the vertical height (bits [59:48]), the data format (bit [62]) SRC1[62]=0➔FP16, SRC1[62]=1➔FP32. The DST register contains the address where the result will be written.

NUFP, NeuroUnit Forward Propagation

Mnemonic:

NUFP SRC1

Format:



Example:

nufp r17

Description:

Forward propagation. The SRC1 register contains the address of the source data. During forward propagation, the results of the intermediate layers' calculations and the results of the last layer are placed in the data buffer immediately after the source data, as shown in the figure below.