**Design Specification of**

**Deep Learning Accelerator**

**(ARTOSYN)**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Writer | Date | Description |
| V0.1 | Sha Shen | 2017/12/1 | Initial |
| V0.2 | Sha Shen | 2017/11/15 | Add chapter 5-7 |
|  |  |  |  |

## Table of contents

[Table of contents 1](#_Toc501215800)

[Chapter 1. Overview 2](#_Toc501215801)

[Chapter 2. Feature list 3](#_Toc501215802)

[Chapter 3. Hardware architecture 4](#_Toc501215803)

[Chapter 4. NN generator 8](#_Toc501215804)

## Chapter 1. Overview

More and more new deep neuro networks emerge in the field of machine vision. The computational complexity of these new networks continues to soar up. The traditional CPU, DSP or GPU in embedded systems cannot afford such computational burden. They are also not efficient methods in terms of power consumption as well as DDR access.

The DLA (Deep Learning Accelerator) described in this document is specific for the inference of various deep neuro networks which include (but not limited to) AlexNet, GoogLeNet, VGG, ResNet and yolo. Dedicated hardware acceleration will be available for 2D convolution, activation, pooling and normalization and other essential parts of a deep neuro network.

The building blocks inside DLA should be scheduled by external DSP/CPU. It cannot accomplish the inference of a deep neuro network by itself. This DLA is better to be used as a coprocessor of DSP or CPU. But it is not tightly coupled with any specific CPU/DSP. The AXI 4 bus interface of the DLA is provided in order for both the data input/output and DLA configuration.

A typical system which incorporates the DLA is shown in Fig. 1



Fig. 1 system diagram using DLA

The DLA will deliver up to 16TOPS (measured by INT4x4 MAC) at 1GHz working frequency with the peak power of 1W. It will increase the image processing power by 10 times if compared with the XM4 DSP used in Sirius chip.

(盘古，ADAS，1080p@30fps，2W，250 objects/frame, 1Tops)

## Chapter 2. Feature list

The DLA has the following features:

* Different convolution layer（stride 1/2/3）:
  + direct
  + Winograd（NO SUPPORT）
  + Image-input
  + Batching
* Hybrid precision MAC supporting INT16/8/4/2/binary
* Padding type：zero/copy
* On-chip convolution buffer up to 2MB (shared SRAM)
* Support various activation layer:
  + ReLu/PReLu/Leaky ReLu/ReLu6/CReLu
  + ELU/SELU
  + Softmax(???)
  + Lookup table based: tanh/sigmoid
* Linear function layer:
  + precision scaling
  + batch normalization
  + bias addition
  + element-wise operation
* max/min/average pooling layer，stride 2/3/4/5
* Local Response normalization
* De-convolution
* Image input format
  + RGB/YUV/raw
  + 8bit/10bit
  + Planar/semi-planar/Interleave
* AXI4 master interface for image/feature/weight in/out
* AXI4 slave interface for buffer filling, register configuring, interrupt status report

## Chapter 3. Hardware architecture

The internal hardware architecture is shown in Fig. 3.1



Fig. 3.1 Top diagram of DLA

3.1 input/output port

All the input/output ports are described in Table 3.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Port name | Dir | Width | Description |  |
| Clk\_bus | I | 1 | Bus interface Clock input |  |
| Clk\_dla | I |  | DLA core clock |  |
| rstn | I | 1 | System reset, active low |  |
| dla\_intr | O | 1 | Interruption output |  |
| m\_arid | O | 10 |  |  |
| m\_araddr | O | 32 |  |  |
| m\_arlen | O | 8 |  |  |
| m\_arsize | O | 3 |  |  |
| m\_arbusrt | O | 2 |  |  |
| m\_arvalid | O | 1 |  |  |
| m\_arready | I | 1 |  |  |
| m\_rid | I | 10 |  |  |
| m\_rdata | I | 128 |  |  |
| m\_rresp | I | 2 |  |  |
| m\_rlast | I | 1 |  |  |
| m\_rvalid | I | 1 |  |  |
| m\_rready | O | 1 |  |  |
| m\_awid | O | 10 |  |  |
| m\_awaddr | O | 32 |  |  |
| m\_awlen | O | 8 |  |  |
| m\_awsize | O | 3 |  |  |
| m\_awbusrt | O | 2 |  |  |
| m\_awvalid | O | 1 |  |  |
| m\_awready | I | 1 |  |  |
| m\_wid | O | 10 |  |  |
| m\_wdata | O | 128 |  |  |
| m\_wstrb | O | 16 |  |  |
| m\_wlast | O | 1 |  |  |
| m\_wvalid | O | 1 |  |  |
| m\_wready | I | 1 |  |  |
| m\_bid | I | 10 |  |  |
| m\_bresp | I | 2 |  |  |
| m\_bvalid | I | 1 |  |  |
| m\_bready | O | 1 |  |  |
| s\_arid | I | 10 |  |  |
| s\_araddr | I | 32 |  |  |
| s\_arlen | I | 8 |  |  |
| s\_arsize | I | 3 |  |  |
| s\_arbusrt | I | 2 |  |  |
| s\_arvalid | I | 1 |  |  |
| s\_arready | O | 1 |  |  |
| s\_rid | O | 10 |  |  |
| s\_rdata | O | 32 |  |  |
| s\_rresp | O | 2 |  |  |
| s\_rlast | O | 1 |  |  |
| s\_rvalid | O | 1 |  |  |
| s\_rready | I | 1 |  |  |
| s\_awid | I | 10 |  |  |
| s\_awaddr | I | 32 |  |  |
| s\_awlen | I | 8 |  |  |
| s\_awsize | I | 3 |  |  |
| s\_awbusrt | I | 2 |  |  |
| s\_awvalid | I | 1 |  |  |
| s\_awready | O | 1 |  |  |
| s\_wid | I | 10 |  |  |
| s\_wdata | I | 32 |  |  |
| s\_wstrb | I | 16 |  |  |
| s\_wlast | I | 1 |  |  |
| s\_wvalid | I | 1 |  |  |
| s\_wready | O | 1 |  |  |
| s\_bid | O | 10 |  |  |
| s\_bresp | O | 2 |  |  |
| s\_bvalid | O | 1 |  |  |
| s\_bready | I | 1 |  |  |
| APB BUS |  |  |  |  |
| DEBUG BUS |  |  | Handshaking signals  Buffer info  Interrupt info |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### 3.2 Description of each sub-block

#### 3.2.1 Compress/ Decompress

#### 3.2.2 CDMA

3.2.2.1 Introduction

CDMA is responsible to read features and weights from DDR, and write them to CBUF in required format.

3.2.2.2 Features

* Support ping-pang structure with cooperation with CSC.
* Support to hold 4 tasks to do the data transfer

3.2.2.3 Input/Output Ports

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Direction** | **Width** | **Description** |
| clk\_i | in | 1 | clock input |
| rstn\_i | in | 1 | reset input |
|  | | | |
| cdma2csc\_full\_0 | out | 1 | Ping buffer full indication |
| cdma2csc\_full\_1 | out | 1 | Pang buffer full indication |
| csc2cdma\_emp\_0 | in | 1 | Ping buffer empty indication |
| csc2cdma\_emp\_1 | in | 1 | Pang buffer empty indication |
|  |  |  |  |
| cdma\_arvalid\_o | out | 1 | CDMA read address valid |
| cdma\_araddr\_o | out | 32 | CDMA read address |
| cdma\_arlen\_o | out | 4 | CDMA read length |
| cdma\_arready\_i | in | 1 | CDMA read address ready |
| cdma\_rvalid\_i | in | 1 | CDMA read data valid |
| cdma\_rdata\_i | in | 128 | CDMA read data |
| cdma\_rlast\_i | in | 1 | CDMA read data last |
| cdma\_rready\_o | out | 1 | CDMA read data ready |
|  |  |  |  |
| cdma\_awvalid\_o | out | 1 | CDMA write address valid |
| cdma\_awaddr\_o | out | 32 | CDMA write address |
| cdma\_awlen\_o | out | 4 | CDMA write length |
| cdma\_awready\_i | in | 1 | CDMA write address ready |
| cdma\_wvalid\_o | out | 1 | CDMA write data valid |
| cdma\_wdata\_o | out | 128 | CDMA write data |
| cdma\_wlast\_o | out | 1 | CDMA write data last |
| cdma\_wready\_i | in | 1 | CDMA write data ready |
| cdma\_bvalid\_i | in | 1 | CDMA write respond valid |
| cdma\_bresp\_i | in | 2 | CDMA write respond |
| cdma\_bready\_o | out | 1 | CDMA write respond ready |

3.2.2.4 Register File

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Width** | **Default** | **R/W** | **Description** |
| start\_en | 1 | 1’b0 | R/W | Start indication to kick off the dma. |
| start\_address0 | 32 | 32’h0 | R/W | The start address of task0. |
| transfer\_size0 | 32 | 32’h0 | R/W | The total transfer size of task0. |
| start\_address1 | 32 | 32’h0 | R/W | The start address of task1. |
| transfer\_size1 | 32 | 32’h0 | R/W | The total transfer size of task0. |
| start\_address2 | 32 | 32’h0 | R/W | The start address of task2. |
| transfer\_size2 | 32 | 32’h0 | R/W | The total transfer size of task0. |
| start\_address3 | 32 | 32’h0 | R/W | The start address of task3. |
| transfer\_size3 | 32 | 32’h0 | R/W | The total transfer size of task0. |

增加分tile的参数

#### 3.2.3 Convolution Buffer

3.2.3.1 Introduction

Convolution buffer is responsible to buffer the features and weights which are used to do the convolution. It’s total 2MByte, including 8x16 banks. And can be accessed by two AXI ports and two special read ports for CSC.

3.2.3.2 Features

* 2 dimension bank split. (8x16)
* The Y-axis bank split is for two purpose. One is for the simultaneous read of features and weights. The other is for the ping-pang buffer implementation of features and weights.
* The X-axis bank split is for two purpose, too. One is for power saving when MAC matrix is split to several engines to read feature. The other is for the 0-skip operation.
* There are 4 groups of ports can access the buffer, one AXI for the SOC system, another AXI for the modules in DLA and two special read ports for CSC to do convolution.

3.2.3.3 Architecture



3.2.3.4 Input/Output Ports

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Direction** | **Width** | **Description** |
| clk\_i | in | 1 | clock input |
| rstn\_i | in | 1 | reset input |
|  | | | |
| csc2cbuf\_X\_wt\_addr\_req\_i | in | 1 | Weight address request (X=0-7) |
| csc2cbuf\_X\_wt\_addr\_i | in | 14 | Weight address (X=0-7) |
| cbuf2csc\_X\_wt\_addr\_ready\_o | out | 1 | Ready to receive weight address (X=0-7) |
| cbuf2csc\_X\_wt\_req\_o | out | 1 | Weight request (X=0-7) |
| cbuf2csc\_X\_wt\_o | out | 128 | Weight (X=0-7) |
| csc2cbuf\_X\_wt\_ready\_i | in | 1 | Ready to receive weight (X=0-7) |
| csc2cbuf\_X\_ft\_addr\_req\_i | in | 1 | Feature address request (X=0-7) |
| csc2cbuf\_X\_ft\_addr\_i | in | 14 | Feature address (X=0-7) |
| cbuf2csc\_X\_ft\_addr\_ready\_o | out | 1 | Ready to receive feature address (X=0-7) |
| cbuf2csc\_X\_ft\_req\_o | out | 1 | Feature request (X=0-7) |
| cbuf2csc\_X\_ft\_o | out | 128 | Feature (X=0-7) |
| csc2cbuf\_X\_ft\_ready\_i | in | 1 | Ready to receive feature (X=0-7) |
|  | | | |
| *DLA\_AXI\_SLAVE* |  |  | dla axi slave ports to access buffer |
|  |  |  |  |
| *SOC\_AXI\_SLAVE* |  |  | *soc axi slave ports to access buffer* |

3.2.3.5 Description

(1) MAC timing

MAC matrix can be partitioned to improve the utilization of MAC PE as descripted in 3.2.4 and 3.2.5.

* MAC\_C\_Partition = 2’b00





* MAC\_C\_Partition = 2’b01





1. Storage Format
2. Feature

* MAC\_C\_Partition = 2’b00



* MAC\_C\_Partition = 2’b01



* MAC\_C\_Partition = 2’b10
* MAC\_C\_Partition = 2’b11

1. Weight

* MAC\_C\_Partition = 2’b00



* MAC\_C\_Partition = 2’b01



* MAC\_C\_Partition = 2’b10



* MAC\_C\_Partition = 2’b11



#### 3.2.4 CSC

3.2.4.1. Introduction

CSC (Convolution Sequence controller) is a unit that controls data/weight sequence from CBUF to CMAC. This unit can automatically choose data/weight stored in CBUF, add paddings, and send data/weight to CMAC. CSC would not change any data itself but only move data from CBUF to CMAC.

3.2.4.2. Feature

* One set register configure one convolutional layer (or de-convolutional layer, atrous/dilated layer, fully connected layer).
* 1024 bit width CBUF data/weight read, 1024 bit width CMAC data/weight write.
* 16/32/64/128 hardware output channel while engine number 1/2/4/8
* Support almost unlimited weight kernel size and feature map size by using advanced operation mechanism.
* Support image input mode (small input channel number)
* Support 3D convolutional computation
* Automatically add zero padding
* Support index mode that weight is based on look-up table. Index address width range is 1/2/3/4/5/6 bit. Index data width range is 8/16 bit.
* Support skipping zero-feature map operation under certain situation.(optional)

3.2.4.3. Interface List

|  |  |  |  |
| --- | --- | --- | --- |
| **CSC&CMAC** | | | |
| **name** | **width** | **I/O** | **description** |
| csc2cmac\_wt | 1024 | O | Weight for CMAC computing |
| csc2cmac\_dat | 1024 | O | Feature map for CMAC computing |
| csc2cmac\_req | 1 | O | Request signal to CMAC |
| cmac2csc\_ready | 1 | I | Ready signal from CMAC |
| csc2cmac\_engine7..0\_num | 8 | O | Indicates which register will be accumulated into for the current feature map computing. This number is mainly used for skipping feature zero operation |
|  |  |  |  |
| **CSC&CBUF signal** | | | |
| **name** | **width** | **I/O** | **description** |
| csc2cbuf\_engine7..0\_wt\_addr\_req | 1 | O | Weight address request signal to CBUF of No.7..0 engine |
| cbuf2csc\_engine7..0\_wt\_addr\_ready | 1 | I | Weight address ready signal from CBUF of No.7..0 |
| csc2cbuf\_engine7..0\_wt\_addr | 14 | O | Weight address to CBUF of No.7..0 |
| csc2cbuf\_engine7..0\_dat\_addr\_req | 1 | O | Feature map address request signal to CBUF of No.7..0 engine |
| cbuf2csc\_engine7..0\_dat\_addr\_ready | 1 | I | Feature map address ready signal from CBUF of No.7..0 |
| csc2cbuf\_engine7..0\_dat\_addr | 14 | O | Feature map address to CBUF of No.7..0 |
| cbuf2csc\_engine7..0\_wt\_req | 1 | I | Weight request signal from CBUF of No.7..0 |
| csc2cbuf\_engine7..0\_wt\_ready | 1 | O | Weight ready signal to CBUF of No.7..0 |
| cbuf2csc\_engine7..0\_wt | 128 | I | Weight from CBUF of No.7..0 |
| cbuf2csc\_engine7..0\_dat\_req | 1 | I | Feature map request signal from CBUF of No.7..0 |
| csc2cbuf\_engine7..0\_dat\_ready | 1 | O | Feature map ready signal to CBUF of No.7..0 |
| cbuf2csc\_engine7..0\_dat | 128 | I | Feature map from CBUF of No.7..0 |
|  |  |  |  |
| **CSC&CSB signal** | | | |
| **name** | **width** | **I/O** | **description** |
| csb2csc\_req | 1 | I | Register configure signal form CSB |
| csc2csb\_ready | 1 | O | Register configure ready signal to CSB |
| csb2csc\_wr\_en | 1 | I | Register configure write enable signal, 1 write mode, 0 read mode |
| csb2csc\_addr | 6 | I | Register configure write/read address from CSB |
| csb2csc\_wr\_data | 32 | I | Register configure write data from CSB |
| csc2csb\_rd\_data | 32 | I | Register configure read data to CSB |
| csc2csb\_stride\_done | 1 | O | Posedge trigger. Interrupt signal to CSB for one feature map stride done. |
| csc2csb\_feature\_done | 1 | O | posedge trigger. Interrupt signal to CSB for one feature map done. |
| csc2csb\_round\_done | 1 | O | posedge trigger. Interrupt signal to CSB for one weight round done. |
| csc2csb\_layer\_done | 1 | O | posedge trigger. Interrupt signal to CSB for one layer done. |
|  |  |  |  |
| **CSC system signal** | | | |
| **name** | **width** | **I/O** | **description** |
| clk | 1 | I | System clk |
| rstn | 1 | I | Active low reset signal which is synchronized with clk |

3.2.4.4. Register

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **width** | **value** | **description** |
| num\_stride\_output\_kernel | 8 | 1…255 | Number of output kernel in one stripe computation |
| f\_pingpong | 8 | 1…255 | 0, means weight pingpong firstly if weight and data both need pingpong  1, means data pingpong firstly if weight and data both need pingpong |
| kernel\_stride\_x | 8 | 1…255 | Weight stride in horizontal direction |
| kernel\_stride\_y | 8 | 1…255 | Weight stride in vertical direction |
| kernel\_size\_x | 8 | 1…255 | Weight size in horizontal direction |
| kernel\_size\_y | 8 | 1…255 | Weight size in vertical direction |
| len\_dat\_stride\_norm | 3 | 4/8/16/32/64/128 | Length of feature map computing for one weight value during normal convolutional stage |
| num\_dat\_stride\_norm | 8 | 1…255 | Number of feature map computing for one weight value during normal convolutional stage |
| len\_dat\_stride\_last | 8 | 1…255 | Length of feature map computing for one weight value during last convolutional stage |
| num\_dat\_stride\_last | 8 | 1…255 | Number of feature map computing for one weight value during last convolutional stage |
| num\_input\_folding | 8 | 1…255 | The value indicates folding number for input channel |
| cbuf\_data0\_address | 16 | 0…214 | Head address of cbuf for feature map 0, ping-pong address. |
| Cbuf\_data1\_address | 16 | 0…214 | Head address of cbuf for feature map 1, ping-pong address. |
| Cbuf\_weight0\_address | 16 | 0…214 | Head address of weight for feature map 0, ping-pong address. |
| Cbuf\_weight1\_address | 16 | 0…214 | Head address of weight for feature map 1, ping-pong address. |
| tile\_dat\_type | 8 | 1…255 | Type of data tile  000:    001:    010:    011:    100:    101:    110: |
| tile\_size\_x\_1 | 8 | 1…255 | Size of tile 1 in horizontal direction |
| tile\_size\_y\_1 | 8 | 1…255 | Size of tile 1 in vertical direction |
| tile\_size\_x\_2 | 8 | 1…255 | Size of tile 2 in horizontal direction |
| tile\_size\_y\_2 | 8 | 1…255 | Size of tile 2 in vertical direction |
| tile\_size\_x\_3 | 8 | 1…255 | Size of tile 3 in horizontal direction |
| tile\_size\_y\_3 | 8 | 1…255 | Size of tile 3 in vertical direction |
| tile\_size\_x\_4 | 8 | 1…255 | Size of tile 4 in horizontal direction |
| tile\_size\_y\_4 | 8 | 1…255 | Size of tile 4 in vertical direction |
| tile\_size\_x\_5 | 8 | 1…255 | Size of tile 5 in horizontal direction |
| tile\_size\_y\_5 | 8 | 1…255 | Size of tile 5 in vertical direction |
| tile\_size\_x\_6 | 8 | 1…255 | Size of tile 6 in horizontal direction |
| tile\_size\_y\_6 | 8 | 1…255 | Size of tile 6 in vertical direction |
| tile\_size\_x\_7 | 8 | 1…255 | Size of tile 7 in horizontal direction |
| tile\_size\_y\_7 | 8 | 1…255 | Size of tile 7 in vertical direction |
| tile\_size\_x\_8 | 8 | 1…255 | Size of tile 8 in horizontal direction |
| tile\_size\_y\_8 | 8 | 1…255 | Size of tile 8 in vertical direction |
| tile\_size\_x\_9 | 8 | 1…255 | Size of tile 9 in horizontal direction |
| tile\_size\_y\_9 | 8 | 1…255 | Size of tile 9 in vertical direction |
| tile\_num\_x\_c | 8 | 1…255 | Number of tile 5 in horizontal direction |
| tile\_num\_y\_c | 8 | 1…255 | Number of tile 5 in vertical direction |
| num\_wt\_tile | 8 | 1…255 | Number of weight tile |
| tile\_num\_wt \_output \_kernel | 8 | 1…255 | Number of output weight kernel in tile |
| Computation type | 8 | 1…255 | Computation type：  0：normal convolutional computation  dilated convolutional computation  de-convolutional computation  1: fc  2: image input |
| compute\_data\_precision | 3 | 1/2/4/8/16 | Feature map precision for CMAC computing |
| compute\_weight\_precision | 3 | 1/2/4/8/16 | Weight precision for CMAC computing |
| index\_enable | 1 | 0/1 | Enable signal for index mode |
| index\_width | 3 | 1/2/3/4/5/6/7/8 | Index address width |
| padding\_t | 8 | 0…255 | Size of adding 0 padding on top side of feature map |
| padding\_b | 8 | 0…255 | Size of adding 0 padding on bottom side of feature map |
| padding\_l | 8 | 0…255 | Size of adding 0 padding on left side of feature map |
| padding\_r | 8 | 0…255 | Size of adding 0 padding on right side of feature map |
| padding\_x | 8 | 0…255 | Size of adding 0 padding between features on horizontal direction for de-convolutional computation |
| padding\_y | 8 | 0…255 | Size of adding 0 padding between features on vertical direction for de-convolutional computation |
| dilated\_x | 8 | 0…255 | Size of adding 0 padding between features on horizontal direction |
| dilated\_y | 8 | 0…255 | Size of adding 0 padding between features on vertical direction |
| num\_engine | 8 | 0…255 | Number of input engine  00: 1  01: 2  10: 4  11: 8 |
| new\_layer\_enable | 1 | posedge | Posedge signal indicates new layer and all other register will be set new value. |

3.2.4.5. Functional Description

1) Convolutional Operation Mechanism

The state-of-the-art convolutional sequence mechanism is used in CSC to deliver nearly unlimited both weight kernel size and feature map size. The main operation shows below,

***Step1***

Choose input feature with hardware\_input\_channel from entire input channel. Mark this feature as ***F***. Choose one of weight kernel marked as ***W***. Then ***W*** slides horizontally on ***F*** with distance of *len\_dat\_stride\_norm*. (Note that if last sliding distance is smaller than *len\_dat\_stride\_norm* , this stride should be merged into former stride and its sliding distance is *len\_dat\_stride\_last*).

***Step 2***

Repeat ***Step 1*** until all input channel feature map with corresponding ***W*** have done.

***Step 3***

Then ***W*** continue slide from previous stop point of feature map, and repeat ***Step 1,2***

***Step 4***

If ***W*** has gone through all ***F***, choose another new W, repeat ***Step 1,2,3*** until all weight kernel have been used.

For more detail information, some figures have been illustrated as below,

Fig. 1 shows a base demo of one input channel and one output channel,

kernel\_size\_x = 3

kernel\_size\_y=3

tile\_size\_x/ tile\_size\_x\_rb = 13



Fig. 1

Fig. 2, Fig.3, Fig.4 show a demo of one output channel base operation

num\_input\_folding=3

kernel\_size\_x = 3

kernel\_size\_y=3



Fig.2



Fig.3



Fig.4

Fig.5 show a demo of one input channel and one output channel

num\_input\_folding=1

kernel\_size\_x = 3

kernel\_size\_y=3

tile\_size\_x/tile\_size\_x\_rb=8

len\_dat\_stride\_norm=8

num\_data\_stride\_norm=3

len\_dat\_stride\_last=12

num\_data\_stride\_last=1



Fig.5

Fig.6 shows timing diagram from CSC to CMAC interface. Weights store into CAMC firstly, then corresponding data run while next stride weights refresh per clock cycle simultaneously. Note that the same color for weight and data means they are one pair of MAC input.



Fig.6

2) Tile Partitioning

Under some case that feature map size is very huge, especially for large input channel, the feature map should be divided into some small tiles so that the CBUF can meet bandwidth requirement.

Four types of tiles should be required if necessary, that is, upper-left, upper-right, lower-left and lower-right which are illustrated as Fig.7. CSC will load all the data/weight by counting *tile\_num\_x* and *tile\_num\_y*. CSC will transfer data according to the order left->right and top->bottom.



Fig.7

3) CBUF Read Mechanism

CSC imports data/weight from CBUF. For data, 2 addresses *cbuf\_data0\_address* and *cbuf\_data1\_address* are required to implement ping-pong operation. For weight, head address *cbuf\_weight\_address*, number of weight kernel in one weight-reloading round *num\_output\_wt* and number of weight-reloading round *num\_output\_wt\_round* are required. The CBUF reading mechanism shows in Fig.8. The relationship between these parameters are,

*num\_output\_wt\_round* = total\_wt/( *num\_output\_wt* \*1024bit/ *compute\_weight\_precision*)



Fig.8

4) Fully Connected Operation Mechanism

Fully connected layer operation is similar to convolutional layer operation. The difference is that fully connected layer has no weight sharing operation. In other words, every feature map point has only one unique weight. So feature map refresh not every clock cycle but every weight reloaded cycle (this cycle determined by the hardware output channel, it is 16). Feature map should wait all weight loading into CMAC and then updates, wait next weighting loading. Fully connected mechanism shows in fig.9,

Fig.10 shows timing diagram from CSC to CMAC interface. Weights store into CAMC firstly, then corresponding data run while next stride weights refresh per clock cycle simultaneously. Note that it is different from convolutional operation. Feature data no longer refresh per clock cycle but only refresh one stride ends.



Fig.9



Fig.10

5) Adding Zero Padding

CSC automatically adds zero padding on the feature map by setting padding\_x, padding\_y, *padding\_t*, *padding\_l*, *padding\_r* and *padding\_b*. If padding parameter is set zero, it means no padding on this side or no padding between features. For example, *padding\_t* =0, it means there is no need for adding zero padding on top side. Fig.11 shows the details,



Fig.11

6) Input Channel Partitioning

DLA supports input channel partitioning function that can improve MAC hardware utilization. Parameter *num\_engine* determines how many partitions MAC array is divided into. According to different number of engine configuration, CSC will adjust CBUF read strategy and CMAC data/weight sending strategy. Fig.12 illustrates the demo under *num\_engine*=1 and *num\_engine*=4, When number of engine is 1, CSC will read data from CBUF every entry. When number of engine is 4, CSC will read data CBUF every entry and automatically choose available data to move them into CMAC.



Fig.12

7) Weight Index Mode

Weight index mode is implemented when parameter *index \_enable* is high. CSC under this mode reads data from CBUF as index address and fetches real weight from weight index table and then CSC sends weights to CMAC. Weight index table is configured by system register and its behavior just likes a look up table. The address width is determined by parameter *index\_width*. Fig.13 shows the details.



Fig.13

7) Image Input Mode



8) Deconvolutional computation

9) 3D convolutional computation

10)Batch mode

11) Skip Feature Zero(Optional)

#### 3.2.5 CMAC/ACCU

3.2.5.1 Feature

(1) support int16xint16, int8xint8, int4xint4 (int1xint(1248)、int2xint(248)、int4xint8)

(int4xint4 16Tops).

(2) support (C means input channel number, K means output channel number)

* C=64, K=16
* C=32, K=32
* C=16, K=64
* C=8, K=128

(under int8xint8, C multiple 2，K multiple 2; under int4xint4, C multiple 4, K multiple 4)

(3) support configurable truncation and saturation of ACCU

3.2.5.2 Architecture Diagram

The internal architecture of MAC/ACCU is shown in Fig. 3.2.5.2.1



Fig. 3.2.5.2.1 Architecture of MAC/ACCU

3.2.5.3 Input/Output

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Dir | Width | Description |
| clk\_i | I | 1 | clock for MAC and ACCU |
| rstn\_i | I | 1 | reset for MAC and ACCU, active low |
| wt\_csc\_i | I | 1024 | weight data |
| actv\_csc\_i | I | 1024 | activation data |
| csc2mac\_valid\_i | I | 1 | indicate the csc is valid to transfer data to mac |
| mac2csc\_ready\_o | O | 1 | indicate the mac is ready to receive data from csc |
| mac2sdp\_valid\_o | O | 1 | indicate the mac is valid to transfer data to sdp |
| sdp2mac\_ready\_i | I | 1 | indicate the sdp is ready to receive data from mac |
| mac2sdp\_data\_o | O | 512 | for activation module data |
| interrupt\_o | O | 8 | interrupt information of MAC/ACCU |

3.2.5.4 Register

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **R/W** | **Description** |
| precision | 8 | R/W | The precision of the operator |
| ck\_conf | 8 | R/W | 00：set C=64、K=16  01：set C=32、K=32  10：set C=16、K=64  11：set C=8、K=128  (under int8xint8, C multiple 2, K multiple 2  under int4xint4, C multiple 2, K multiple 2) |
| sliding\_conf | 8 | R/W | each feature data’s sliding number  (4\8\16\32\64\128) |
| feature\_size\_x | 8 | R/W | the width of feature’s horizontal direction |
| feature\_size\_y | 8 | R/W | the height of feature’s vertical direction |
| kernel\_size\_x | 8 | R/W | the width of kernel’s horizontal direction |
| kernel\_size\_y | 8 | R/W | the height of kernel’s vertical direction |
| kernel\_stripe\_x | 8 | R/W | the stripe number of kernel’s horizontal direction |
| kernel\_stripe\_y | 8 | R/W | the stripe number of kernel’s vertical direction |
| tile\_size\_x | 8 | R/W | the width of tile’s horizontal direction |
| tile\_size\_y | 8 | R/W | the height of tile’s vertical direction |
| tile\_num\_x | 8 | R/W | the tile’s number of horizontal direction |
| tile\_num\_y | 8 | R/W | the tile’s number of vertical direction |
| tile\_size\_x\_r | 8 | R/W | the width of the right tile’s horizontal direction |
| tile\_size\_y\_r | 8 | R/W | the height of the right tile’s vertical direction |
| tile\_num\_y\_r | 8 | R/W | the right tile’s number |
| tile\_size\_x\_b | 8 | R/W | the width of the below tile’s horizontal direction |
| tile\_size\_y\_b | 8 | R/W | the height of the below tile’s vertical direction |
| tile\_num\_x\_b | 8 | R/W | the below tile’s number |
| tile\_size\_x\_rb | 8 | R/W | the width of the low right tile’s horizontal direction |
| tile\_size\_y\_rb | 8 | R/W | the height of the low right tile’s vertical direction |
| precision\_conf\_i | 8 | R/W | calculation precision information for MAC |
| truncation\_conf\_i | 8 | R/W | truncation information for ACCU |
| saturation\_conf\_i | 8 | R/W | saturation information for ACCU |

3.2.5.5 Calculation Mode

1. C=64, K=16

The data arrangement of mac’s calculation is decided as c=64, k=16.



1. C=32, K=32

The data arrangement of mac’s calculation is decided as c=32, k=32.



1. C=16, K=64

The data arrangement of mac’s calculation is decided as c=16, k=64.



1. C=8, K=128

The data arrangement of mac’s calculation is decided as c=8, k=128.



3.2.5.6 Assemble Buffer

The Sram bank number is decided by the maximum calculation K value, and the address space of each SRAM bank is decided by maximum sliding\_conf\_i (register)’s value.



Fig. 3.2.5.6.1 The Date in Feature which Stored in SRAM



Fig. 3.2.5.6.2 The Temporary(ACCU) Data Mapping in SRAM

3.2.5.7 Accumulator Unit

The Accumulator Unit has 128 basic accumulator unit, which add the data between partial sum (SRAM) and partial sum (MAC). Meanwhile, it accomplishes the function of saturation and truncation of the result of adder. The result will transfer to the SRAM or SDP.



Fig. 3.2.1.7.1 Basic Accumulator Unit

#### 3.2.6 SDP

3.2.6.1 Introduction

SDP (Single Data Processor) is responsible to realize the activation layer and linear function layer. All the operations in this module is on single point and there are total 16 16bit-points that can be operated on in parallel.

3.2.6.2 Features

* The input points can be from CONV or memory.
* The output points that have been processed can be transferred to PDP directly or written to memory.
* The order of every linear operations is flexible to configure.
* Using two look-up-tables to support nonlinear active functions.
* The operations of unit0 and unit1 are based on kernels or layers, while the operations of unit2 are based only on layers.

3.2.6.2 Architecture



3.2.6.3 Interface

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Direction** | **Width** | **Description** |
| clk\_i | in | 1 | clock input |
| rstn\_i | in | 1 | reset input |
|  | | | |
| conv2sdp\_valid\_i | in | 1 | Data from convolution valid indication |
| conv2sdp\_data\_i | in | 512 | Data from convolution (16x32bits) |
| sdp2conv\_ready\_o | out | 1 | SDP ready to receive data from convolution indication |
|  | | | |
| sdp2pdp\_valid\_o | out | 1 | Data from to PDP valid indication |
| sdp2pdp\_data\_o | out | 256 | Data to PDP (16x16bits) |
| pdp2sdp\_ready\_i | in | 1 | PDP ready to receive data from SDP indication |
|  | | | |
| sdp\_re0\_arvalid\_o | out | 1 | Read Engine0 read address valid |
| sdp\_re0\_araddr\_o | out | 32 | Read Engine0 read address |
| sdp\_re0\_arlen\_o | out | 4 | Read Engine0 read length |
| sdp\_re0\_arready\_i | in | 1 | Read Engine0 read address ready |
| sdp\_re0\_rvalid\_i | in | 1 | Read Engine0 read data valid |
| sdp\_re0\_rdata\_i | in | 256 | Read Engine0 read data |
| sdp\_re0\_rlast\_i | in | 1 | Read Engine0 read data last |
| sdp\_re0\_rready\_o | out | 1 | Read Engine0 read data ready |
|  | | | |
| sdp\_re1\_arvalid\_o | out | 1 | Read Engine1 read address valid |
| sdp\_re1\_araddr\_o | out | 32 | Read Engine1 read address |
| sdp\_re1\_arlen\_o | out | 4 | Read Engine1 read length |
| sdp\_re1\_arready\_i | in | 1 | Read Engine1 read address ready |
| sdp\_re1\_rvalid\_i | in | 1 | Read Engine1 read data valid |
| sdp\_re1\_rdata\_i | in | 256 | Read Engine1 read data |
| sdp\_re1\_rlast\_i | in | 1 | Read Engine1 read data last |
| sdp\_re1\_rready\_o | out | 1 | Read Engine1 read data ready |
|  | | | |
| sdp\_re2\_arvalid\_o | out | 1 | Read Engine2 read address valid |
| sdp\_re2\_araddr\_o | out | 32 | Read Engine2 read address |
| sdp\_re2\_arlen\_o | out | 4 | Read Engine2 read length |
| sdp\_re2\_arready\_i | in | 1 | Read Engine2 read address ready |
| sdp\_re2\_rvalid\_i | in | 1 | Read Engine2 read data valid |
| sdp\_re2\_rdata\_i | in | 256 | Read Engine2 read data |
| sdp\_re2\_rlast\_i | in | 1 | Read Engine2 read data last |
| sdp\_re2\_rready\_o | out | 1 | Read Engine2 read data ready |
|  | | | |
| sdp\_re3\_arvalid\_o | out | 1 | Read Engine3 read address valid |
| sdp\_re3\_araddr\_o | out | 32 | Read Engine3 read address |
| sdp\_re3\_arlen\_o | out | 4 | Read Engine3 read length |
| sdp\_re3\_arready\_i | in | 1 | Read Engine3 read address ready |
| sdp\_re3\_rvalid\_i | in | 1 | Read Engine3 read data valid |
| sdp\_re3\_rdata\_i | in | 256 | Read Engine3 read data |
| sdp\_re3\_rlast\_i | in | 1 | Read Engine3 read data last |
| sdp\_re3\_rready\_o | out | 1 | Read Engine3 read data ready |
|  | | | |
| sdp\_we\_awvalid\_o | out | 1 | Write Engine write address valid |
| sdp\_we\_awaddr\_o | out | 32 | Write Engine write address |
| sdp\_we\_awready\_i | in | 1 | Write Engine write address ready |
| sdp\_we\_wvalid\_o | out | 1 | Write Engine write data valid |
| sdp\_we\_wdata\_o | out | 256 | Write Engine write data |
| sdp\_we\_wready\_i | in | 1 | Write Engine write data ready |
| sdp\_we\_bvalid\_i | in | 1 | Write Engine write respond valid |
| sdp\_we\_bresp\_i | in | 2 | Write Engine write respond |
| sdp\_we\_bready\_o | out | 1 | Write Engine write respond ready |

3.2.6.4 Register File

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Width** | **Default** | **R/W** | **Description** |
| sdp\_precision | 2 | 2’b00 | R/W | The precision of the operator.  2’b00:16bit 2’b01:8bit 2’b10:4bit 2’b11:2bit |
| sdp\_width | 16 | 16’h0 | R/W | The input cube width. |
| sdp\_height | 16 | 16’h0 | R/W | The input cube height. |
| sdp\_channel | 16 | 16’h0 | R/W | The input cube channel number. |
| sdp\_unit0\_en | 3 | 3’b000 | R/W | The enable indication of unit0 bias/scale/shift.  [0]:bias [1]:scale [2]:shift |
| sdp\_unit0\_cfg | 3 | 3’b000 | R/W | The cfg of unit0 bias/scale/shift. If 0, the param is from register. If 1, the param is from re1.  [0]:bias [1]:scale [2]:shift |
| sdp\_unit1\_en | 3 | 3’b000 | R/W | The enable indication of unit1 bias/scale/shift.  [0]:bias [1]:scale [2]:shift |
| sdp\_unit1\_cfg | 3 | 3’b000 | R/W | The cfg of unit1 bias/scale/shift. If 0, the param is from register. If 1, the param is from re1.  [0]:bias [1]:scale [2]:shift |
| sdp\_unit2\_en | 3 | 3’b000 | R/W | The enable indication of unit2 bias/scale/shift.  [0]:bias [1]:scale [2]:shift |
| sdp\_relu\_en | 1 | 1’b0 | R/W | The enable indication of RELU. |
| sdp\_prelu\_en | 1 | 1’b0 | R/W | The enable indication of pRELU. |
| sdp\_prelu\_param | 16 | 16’h0 | R/W | The parameter of pRELU. |
| sdp\_lut\_en | 1 | 1’b0 | R/W | The enable indication of LUT. |
| sdp\_lut\_slopeX | 32 | 32’h0 | R/W | X=0-63. The slope value of look up table. |
| sdp\_lut\_biasX | 32 | 32’h0 | R/W | X=0-63. The bias value of look up table. |
| sdp\_ew\_en | 1 | 1’b0 | R/W | The enable indication of element-wise operation. |
| sdp\_ew\_type | 2 | 2’b00 | R/W | The type of the element-wise operation.  2’b00: add 2’b01: sub 2’b10: max 2’b11 min |
| sdp\_bias0\_param | 16 | 16’h0 | R/W | The parameter of the bias in unit0. |
| sdp\_scale0\_param | 16 | 16’h0 | R/W | The parameter of the scale in unit0. |
| sdp\_shift0\_param | 16 | 16’h0 | R/W | The parameter of the shifter in unit0. |
| sdp\_bias1\_param | 16 | 16’h0 | R/W | The parameter of the bias in unit1. |
| sdp\_scale1\_param | 16 | 16’h0 | R/W | The parameter of the scale in unit1. |
| sdp\_shift1\_param | 16 | 16’h0 | R/W | The parameter of the shifter in unit1. |
| sdp\_bias2\_param | 16 | 16’h0 | R/W | The parameter of the bias in unit2. |
| sdp\_scale2\_param | 16 | 16’h0 | R/W | The parameter of the scale in unit2. |
| sdp\_shift2\_param | 16 | 16’h0 | R/W | The parameter of the shifter in unit2. |
|  |  |  |  |  |
| sdp\_re0\_en | 1 | 1’b0 | R/W | The enable indication of read engine0. |
| sdp\_re1\_en | 1 | 1’b0 | R/W | The enable indication of read engine1. |
| sdp\_re2\_en | 1 | 1’b0 | R/W | The enable indication of read engine2. |
| sdp\_re3\_en | 1 | 1’b0 | R/W | The enable indication of read engine3. |
| sdp\_we\_en | 1 | 1’b0 | R/W | The enable indication of write engine. |
| sdp\_re0\_start\_addr | 32 | 32’h0 | R/W | The read start address of read engine0. |
| sdp\_re0\_size | 32 | 32’h0 | R/W | The read byte number of read engine0. |
| sdp\_re1\_start\_addr | 32 | 32’h0 | R/W | The read start address of read engine1. |
| sdp\_re1\_size | 32 | 32’h0 | R/W | The read byte number of read engine1. |
| sdp\_re2\_start\_addr | 32 | 32’h0 | R/W | The read start address of read engine2. |
| sdp\_re2\_size | 32 | 32’h0 | R/W | The read byte number of read engine2. |
| sdp\_re3\_start\_addr | 32 | 32’h0 | R/W | The read start address of read engine3. |
| sdp\_re3\_size | 32 | 32’h0 | R/W | The read byte number of read engine3. |
| sdp\_we\_start\_addr | 32 | 32’h0 | R/W | The write start address of write engine. |
| sdp\_we\_serial\_nu | 32 | 32’h0 | R/W | The serial number of input (16x16bits) which will be write to serial address. |
| sdp\_we\_interval | 32 | 32’h0 | R/W | The address interval between two serial inputs. |

#### 3.2.7 PDP

3.2.7.1 Introduction

PDP (Planar Data Processor) is responsible to realize the pooling layer.The operations in this module is on a planar and also total 16 16bit-points can be operated in parallel.

3.2.7.1 Features

* Pooling type support max/min/average.
* The input data can be from SDP or memory.
* According to the MAC’s process sequence, the operation here is split to 1-dimension first and then 2-dimension.
* There are total 7 line buffers to support the max pooling stride 8 when doing 2-dimension operation.

3.2.7.2 Architecture



3.2.7.3 Interface

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Direction** | **Width** | **Description** |
| clk\_i | in | 1 | clock input |
| rstn\_i | in | 1 | reset input |
| sdp2pdp\_valid\_i | in | 1 | Data from convolution valid indication |
| sdp2pdp\_data\_i | in | 256 | Data from convolution (16x16bits) |
| pdp2sdp\_ready\_o | out | 1 | PDP ready to receive data from SDP indication |
|  | | | |
| pdp\_re\_arvalid\_o | out | 1 | Read Engine read address valid |
| pdp\_re\_araddr\_o | out | 32 | Read Engine read address |
| pdp\_re\_arlen\_o | out | 4 | Read Engine read length |
| pdp\_re\_arready\_i | in | 1 | Read Engine read address ready |
| pdp\_re\_rvalid\_i | in | 1 | Read Engine read data valid |
| pdp\_re\_rdata\_i | in | 256 | Read Engine read data |
| pdp\_re\_rlast\_i | in | 1 | Read Engine read data last |
| pdp\_re\_rready\_o | out | 1 | Read Engine read data ready |
|  |  |  |  |
| pdp\_we\_awvalid\_o | out | 1 | Write Engine write address valid |
| pdp\_we\_awaddr\_o | out | 32 | Write Engine write address |
| pdp\_we\_awready\_i | in | 1 | Write Engine write address ready |
| pdp\_we\_wvalid\_o | out | 1 | Write Engine write data valid |
| pdp\_we\_wdata\_o | out | 256 | Write Engine write data |
| pdp\_we\_wready\_i | in | 1 | Write Engine write data ready |
| pdp\_we\_bvalid\_i | in | 1 | Write Engine write respond valid |
| pdp\_we\_bresp\_i | in | 2 | Write Engine write respond |
| pdp\_we\_bready\_o | out | 1 | Write Engine write respond ready |

3.2.7.4 Register File

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Width** | **Default** | **R/W** | **Description** |
| pdp\_precision | 2 | 2’b00 | R/W | The precision of the operator.  2’b00:16bit 2’b01:8bit 2’b10:4bit 2’b11:2bit |
| pdp\_width | 16 | 16’h0 | R/W | The input cube width. |
| pdp\_height | 16 | 16’h0 | R/W | The input cube height. |
| pdp\_channel | 16 | 16’h0 | R/W | The input cube channel number. |
| pdp\_pooling\_type | 2 | 2’00 | R/W | The type of pooling.  2’b00: No use 2’b01: max  2’b10: min 2’b11: average |
| pdp\_pooling\_size | 4 | 4’b0000 | R/W | The kernel size of pooling.. |
| pdp\_pooling\_stride | 4 | 4’b0000 | R/W | The stride of pooling. |
| pdp\_tile\_en | 1 | 1’b0 | R/W | Indicate if the cube is parted in to tiles. |
| pdp\_tile\_width | 16 | 16’h0000 | R/W | The width of usual tile. |
| pdp\_tile\_width\_last | 16 | 16’h0000 | R/W | The width of the left tile. |
| pdp\_tile\_height | 16 | 16’h0000 | R/W | The height of usual tile. |
| pdp\_tile\_height\_last | 16 | 16’h0000 | R/W | The height of the bottom tile. |
| pdp\_tile\_num\_x | 8 | 8’h00 | R/W | The number of tiles in the x dimension. |
| pdp\_tile\_num\_y | 8 | 8’h00 | R/W | The number of tiles in the y dimension. |
| pdp\_padding\_en | 1 | 1’b0 | R/W | The enable indication of padding. |
| pdp\_padding\_num\_l | 4 | 4’h0 | R/W | Padding number in the left. |
| pdp\_padding\_num\_r | 4 | 4’h0 | R/W | Padding number in the right. |
| pdp\_padding\_num\_u | 4 | 4’h0 | R/W | Padding number in the upside. |
| pdp\_padding\_num\_b | 4 | 4’h0 | R/W | Padding number in the bottom. |
|  |  |  |  |  |
| pdp\_re\_en | 1 | 1’b0 | R/W | The enable indication of read engine. |
| pdp\_we\_start\_addr | 32 | 32’h0 | R/W | The write start address of write engine. |
| pdp\_we\_serial\_nu | 32 | 32’h0 | R/W | The serial number of input (16x16bits) which will be write to serial address. |
| pdp\_we\_interval | 32 | 32’h0 | R/W | The address interval between two serial inputs. |

3.2.7.5 Line Buffer



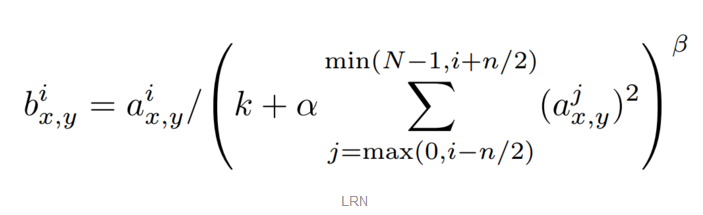
#### 3.2.8 CDP

3.2.8.1 Feature

* Cross-channel Data Processor
* Realize the local response normalization(LRN) function



Refer tf.nn.lrn



3.2.8.2 Architecture



Fig. 3.2.8.2.1 The Architecture of CDP



Fig. 3.2.8.2.2 The Diagram of LRN Equation

3.2.8.3 Input/Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Direction** | **Width** | **Description** |
| clk\_i | in | 1 | clock input |
| rstn\_i | in | 1 | reset input |
| **AXI\_Lite Read data port** | | | |
| **AXI\_Lite Write data port** | | | |

3.2.8.4 Register

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **R/W** | **Description** |
| precision | 4 | R/W | The precision of the operator |
| feature\_size\_x | 8 | R/W | the width of feature’s horizontal direction |
| feature\_size\_y | 8 | R/W | the height of feature’s vertical direction |
| channel\_size | 8 | R/W | the number of feature’s channel |
| lrn\_n\_number | 8 | R/W | the number of LRN caculation’s setting |
| read\_address | 32 | R/W | the beginning address for CDP to read feature |
| write\_address | 32 | R/W | the beginning address for CDP to write feature |

#### 3.2.9 Reshape

3.2.9.1 Feature

* Data Reshape Engine
* Splitting Mode (Splitting mode transforms a data cube into M-planar formats (NCHW). The number of planes is equal to channel size.)
* Slicing Mode (Slicing mode may be used to separate out different features or spatial regions of an image.)
* Merging Mode (Merging mode transforms a serial of planes to a feature data cube.)
* Contraction Mode (Contraction mode transforms mapping format are used to de-extend the cube. It’s a second hardware layer to support deconvolution. Normally, a software deconvolution layer has deconvolution x stride and y stride that are greater than 1; with these strides the output of phase I hardware-layer is a channel-extended data cube.)
* Reshape-transpose Mode (“Reshape-transpose” mode (common in deconvolutional networks) create output data with larger dimensions than the input dataset.)

3.2.9.2 Data Shape Diagram

(1) splitting mode

Refer tf.splite



Fig. 3.2.9.2.1 The Data Diagram of Splitting Mode

(2) slicing mode

Refer tf.slice



Fig. 3.2.9.2.2 The Data Diagram of Slicing Mode

(3) merging mode



Fig. 3.2.9.2.3 The Data Diagram of Merging Mode

(4) reshape-transpose mode



Fig. 3.2.9.2.4 The Data Diagram of Reshape-transpose Mode

3.2.9.3 Reshape Architecture



Fig. 3.2.9.3.1 The Architecture of Reshape

3.2.9.4 Input/Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Direction** | **Width** | **Description** |
| clk\_i | in | 1 | clock input |
| rstn\_i | in | 1 | reset input |
| **AXI\_Lite Read data port** | | | |
| **AXI\_Lite Write data port** | | | |

3.2.9.5 Register

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **R/W** | **Description** |
| precision | 4 | R/W | The precision of the operator |
|  |  |  |  |
| Splitting Mode | | | |
| feature\_size\_x | 8 | R/W | the width of feature’s horizontal direction |
| feature\_size\_y | 8 | R/W | the height of feature’s vertical direction |
| channel\_size | 8 | R/W | the number of feature’s channel |
| split\_number | 8 | R/W | the value of split number |
| ori\_address | 32 | R/W | the address of original feature’s starting point |
| new\_address | 32 | R/W | the address of new feature’s starting point |
|  |  |  |  |
| Slicing Mode | | | |
| ori\_feature\_size\_x | 8 | R/W | the width of original feature’s horizontal direction |
| ori\_feature\_size\_y | 8 | R/W | the height of original feature’s vertical direction |
| ori\_channel\_size | 8 | R/W | the number of original feature’s channel |
| new\_feature\_size\_x | 8 | R/W | the width of new feature’s horizontal direction |
| new\_feature\_size\_y | 8 | R/W | the height of original feature’s vertical direction |
| new\_channel\_size | 8 | R/W | the number of original feature’s channel |
| ori\_address | 32 | R/W | the address of original feature’s starting point |
| new\_address | 32 | R/W | the address of new feature’s starting point |
|  |  |  |  |
| Reshape-transpose Mode | | | |
| ori\_feature\_size\_x | 8 | R/W | the width of original feature’s horizontal direction |
| ori\_feature\_size\_y | 8 | R/W | the height of original feature’s vertical direction |
| ori\_channel\_size | 8 | R/W | the number of original feature’s channel |
| reshape\_type | 8 | R/W | the reshape type, according to Fig.3.2.9.2.4 |
| ori\_address | 32 | R/W | the address of original feature’s starting point |
| new\_address | 32 | R/W | the address of new feature’s starting point |

#### 3.2.10 AXI master interface

#### 3.2.11 AXI slave interface

#### 3.2.12 Interrupt system

#### 3.2.13 SCU

3.2.13.1. Introduction

SCU (System Controller Unit) has two main functions. Firstly, it is the register interface of entire deep learning accelerator for SOC. Second, it is the command processing unit that can control accelerator operation. SCU also sends interrupt signal to SOC according to work status of every sub blocks.

3.2.13.2. Feature

* Fixed command format for one layer configuration.
* Support at most 5 layers at once software configuration.
* AXI4 slave interface for SOC writing with fixed address.
* Hybrid control structure with central/branch unit.
* Inner APB bus between central and branch block.
* Shadow register for branch unit.

3.2.13.3. Architecture



Fig.1

Fig.1 shows overall diagram of SCU. Branch units have shadow registers for their high efficiently layer switching. Command buffer in central unit can store at most 5 layer commands in order to reduce the number of interrupts for CPU. AXI bus is used for communicating with external SOC. Internal APB bus is implemented for multi-slave APB device.

1) Central System Control Unit



Fig.2

Fig.2 shows the hardware diagram of central unit. Deep learning accelerator configuration is written by AXI slave write interface. Then central unit will send command to every block through APB interface. Also, external SoC can access APB interface to fetch current DLA configuration in sub block or central control unit. Note that there is a command buffer which can store extra command. FSM guarantees the entire implementation of central unit.

2) Branch Control Unit



Fig.3

Fig.3 shows the diagram of branch control unit. Central unit would write branch unit through APB slave interface and external SoC would implement read operation by accessing branch unit register. The shadow register mechanism is used in branch unit. Note that SoC can only read current register value. Sub block parameters are decoded from current register value and will switch into shadow register immediately if current operation is accomplished.

3.2.13.5. Interface

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Width | I/O | Comment |
| cdma\_int | 1 | I | cdma done interrupt signal |
| csc\_stride\_int | 1 | I | csc stride done interrupt signal |
| csc\_feature\_int | 1 | I | csc feature done interrupt signal |
| csc\_tile\_int | 1 | I | csc tile done interrupt signal |
| cmac\_int | 1 | I | cmac done interrupt signal |
| mics\_dma | 1 | I | mics done interrupt signal |
| sdp\_int | 1 | I | sdp done interrupt signal |
| pdp\_int | 1 | I | pdp done interrupt signal |
| cdp\_int | 1 | I | cdp done interrupt signal |
| rubik\_int | 1 | I | rubik done interrupt signal |
| bdma\_int | 1 | I | bdma done interrupt signal |
| block parameter bus |  |  | static parameters per layer for each block |
| AXI bus |  |  | AXI slave bus |
| … |  |  |  |

3.2.13.6. Register List

1)CDMA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Width** | **Default** | **R/W** | **Description** |
| MAC\_C\_Size | 2 | 2’b11 | RO | The size in channel dimension of MAC matrix.  2’b00: 8 2’b01: 16  2’b10: 32 2’b11: 64 |
| MAC\_K\_Size | 2 | 2’b11 | RO | The size in kernel dimension of MAC matrix  2’b00: 2 2’b01: 4  2’b10: 8 2’b11: 16 |
| C\_Partition | 2 | 2’b00 | R/W | The partition of MAC matrix in channel dimension.  2’b00: 1 2’b01:1/2  2’b10: 1/4 2’b11: 1/8  PS: if C\_SIZE\_MAC=8, only can be 1,  if C\_SIZE\_MAC=16, can be 1 1/2  if C\_SIZE\_MAC=32, can be 1 1/2 1/4  if C\_SIZE\_MAC=64, can be 1 1/2 1/4 1/8 |
| F\_in\_Width | 16 | 16’h0000 | R/W | The width of the input feature |
| F\_in\_Height | 16 | 16’h0000 | R/W | The height of the input feature |
| F\_in\_Channel | 16 | 16’h0000 | R/W | The channel number of the input feature |
| F\_out\_Channel | 16 | 16’h0000 | R/W | The channel number of the output feature |
| W\_Size | 16 | 16’h0000 | R/W | The size of the kernel |

2)CSC

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **width** | **value** | **description** |
| kernel\_stride\_x | 8 | 1…255 | Weight stride in horizontal direction |
| kernel\_stride\_y | 8 | 1…255 | Weight stride in vertical direction |
| kernel\_size\_x | 8 | 1…255 | Weight size in horizontal direction |
| kernel\_size\_y | 8 | 1…255 | Weight size in vertical direction |
| len\_dat\_stride\_norm | 3 | 4/8/16/32/64/128 | Length of feature map computing for one weight value during normal convolutional stage |
| num\_dat\_stride\_norm | 8 | 1…255 | Number of feature map computing for one weight value during normal convolutional stage |
| len\_dat\_stride\_last | 8 | 1…255 | Length of feature map computing for one weight value during last convolutional stage |
| num\_dat\_stride\_last | 8 | 1…255 | Number of feature map computing for one weight value during last convolutional stage |
| num\_input\_folding | 8 | 1…255 | The value indicates folding number for input channel |
| cbuf\_data0\_address | 16 | 0…214 | Head address of cbuf for feature map 0, ping-pong address. |
| Cbuf\_data1\_address |  |  | Head address of cbuf for feature map 1, ping-pong address. |
| Cbuf\_weight\_address | 16 | 0…214 | Weight address of cbuf |
| tile\_size\_x | 8 | 1…255 | Size of upper-left tile in horizontal direction |
| tile\_size\_y | 8 | 1…255 | Size of upper-left tile in vertical direction |
| tile\_num\_x | 8 | 1…255 | Number of upper-left tile in horizontal direction |
| tile\_num\_y | 8 | 1…255 | Number of upper-left tile in vertical direction |
| tile\_size\_x\_rb | 8 | 1…255 | Size of lower-right tile size in horizontal direction |
| tile\_size\_y\_rb | 8 | 1…255 | Size of lower-right tile in vertical direction |
| num\_output\_wt | 8 | 255 | The value indicates the number of weight address CSC processes per reloading weight |
| num\_output\_wt\_round | 8 | 255 | The value indicates the round number of load weight from CBUF |
| num\_engine | 2 | 1/2/4/8 | Number of engine for CMAC computing |
| compute\_weight\_precision | 3 | 1/2/4/8/16 | Weight precision for CMAC computing |
| compute\_data\_precision | 3 | 1/2/4/8/16 | Feature map precision for CMAC computing |
| index\_enable | 1 | 0/1 | Enable signal for index mode |
| index\_width | 3 | 1/2/3/4/5/6/7/8 | Index address width |
| padding\_t | 8 | 0…255 | Size of adding 0 padding on top side of feature map |
| padding\_b | 8 | 0…255 | Size of adding 0 padding on bottom side of feature map |
| padding\_l | 8 | 0…255 | Size of adding 0 padding on left side of feature map |
| padding\_r | 8 | 0…255 | Size of adding 0 padding on right side of feature map |
| padding\_x | 8 | 0…255 | Size of adding 0 padding between features on horizontal direction |
| padding\_y | 8 | 0…255 | Size of adding 0 padding between features on vertical direction |
| skip\_zero\_enable | 1 | 0/1 | Enable of skipping zero feature map computing |
| new\_layer\_enable | 1 | posedge | Posedge signal indicates new layer and all other register will be set new value. |

3) CMAC

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Dir | Width | Description |
| ck\_conf\_i | I | 8 | 00：set C=64、K=16  01：set C=32、K=32  10：set C=16、K=64  11：set C=8、K=128  (under int8xint8, C multiple 2  under int4xint4, C multiple 4) |
| sliding\_conf\_i | I | 8 | each feature data’s sliding number  (4\8\16\32\64\128) |
| feature\_size\_x | I | 8 | the width of feature’s horizontal direction |
| feature\_size\_y | I | 8 | the height of feature’s vertical direction |
| kernel\_size\_x | I | 8 | the width of kernel’s horizontal direction |
| kernel\_size\_y | I | 8 | the height of kernel’s vertical direction |
| kernel\_stripe\_x | I | 8 | the stripe number of kernel’s horizontal direction |
| kernel\_stripe\_y | I | 8 | the stripe number of kernel’s vertical direction |
| tile\_size\_x | I | 8 | the width of tile’s horizontal direction |
| tile\_size\_y | I | 8 | the height of tile’s vertical direction |
| tile\_num\_x | I | 8 | the tile’s number of horizontal direction |
| tile\_num\_y | I | 8 | the tile’s number of vertical direction |
| tile\_size\_x\_rb | I | 8 | the width of the low right tile’s horizontal direction |
| tile\_size\_y\_rb | I | 8 | the height of the low right tile’s vertical direction |
| precision\_conf\_i | I | 8 | calculation precision information for MAC |
| truncation\_conf\_i | I | 8 | truncation information for ACCU |
| saturation\_conf\_i | I | 8 | saturation information for ACCU |

4) SDP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Width** | **Default** | **R/W** | **Description** |
| precision |  |  | R/W | The precision of the operator |
| unit0\_en | 1 | 1’b0 | R/W | Unit0 enable indication |
| unit1\_en | 1 | 1’b0 | R/W | Unit1 enable indication |
| ew\_en | 1 | 1’b0 | R/W | Element-wise operation enable indication |
| lut\_en | 1 | 1’b0 | R/W | Look up table enable indication |
| lut\_slopeX | 32 | 32’h0 | R/W | X=0-63. The slope value of look up table. |
| lut\_biasX | 32 | 32’h0 | R/W | X=0-63. The bias value of look up table. |
| unit3\_bias | 32 | 32’h0 | R/W | Bias value of unit3 |
| unit3\_scale | 32 | 32’h0 | R/W | Scale value of unit3 |
| unit3\_shift | 32 | 32’h0 | R/W | Shift value of unit3 |

5) PDP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Width** | **Default** | **R/W** | **Description** |
| precision |  |  | R/W | The precision of the operator |
| pooling\_type | 2 | 2’00 | R/W | The type of pooling  2’b00: No use 2’b01: max  2’b10: min 2’b11: average |
| pooling\_size | 4 | 4’b0000 | R/W | The kernel size of pooling. |
| pooling\_stride | 4 | 4’b0000 | R/W | The stride of pooling |
| tile\_en | 1 | 1’b0 | R/W | Indicate if the cube is parted in to tiles |
| tile\_width | 16 | 16’h0000 | R/W | The width of usual tile |
| tile\_width\_last | 16 | 16’h0000 | R/W | The width of the left tile |
| tile\_height | 16 | 16’h0000 | R/W | The height of usual tile |
| tile\_height\_last | 16 | 16’h0000 | R/W | The height of the bottom tile |
| tile\_num\_x | 8 | 8’h00 | R/W | The number of tiles in the x dimension |
| tile\_num\_y | 8 | 8’h00 | R/W | The number of tiles in the y dimension |
| padding\_type | 1 | 1’b0 | R/W | The type of padding  1’b0: VALID 1’b1: SAME |
| padding\_num\_l | 4 | 4’h0 | R/W | Padding number in the left |
| padding\_num\_r | 4 | 4’h0 | R/W | Padding number in the right |
| padding\_num\_u | 4 | 4’h0 | R/W | Padding number in the upside |
| padding\_num\_b | 4 | 4’h0 | R/W | Padding number in the bottom |

6) CDP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Width** | **Default** | **R/W** | **Description** |
| precision |  |  | R/W | The precision of the operator |

7) Interrupt regsiter

|  |  |  |
| --- | --- | --- |
| Name | Width | Comment |
| cdma\_int | 1 | cdma done interrupt signal |
| csc\_stride\_int | 1 | csc stride done interrupt signal |
| csc\_feature\_int | 1 | csc feature done interrupt signal |
| csc\_tile\_int | 1 | csc tile done interrupt signal |
| cmac\_int | 1 | cmac done interrupt signal |
| mics\_dma | 1 | mics done interrupt signal |
| sdp\_int | 1 | sdp done interrupt signal |
| pdp\_int | 1 | pdp done interrupt signal |
| cdp\_int | 1 | cdp done interrupt signal |
| rubik\_int | 1 | rubik done interrupt signal |
| bdma\_int | 1 | bdma done interrupt signal |
| … |  |  |

3.2.13.4. AXI Command list

3.2.13.5. APB read address list

3.2.13.6. Function description

1) Accelerator Command format

Command format for AXI written is illustrated in Fig.4. 8bit msb is required for indicating which ID the command belongs to, and the left 120bit (15bytes) is set for the register value with corresponding ID. There are 16\*15=240 parameters/registers of deep learning accelerator in total.



Fig.4

Table 1 illustrates the detailed command format. ID1-ID15 only contain 16 bytes CNN configuration while ID0 contains 12 bytes CNN configuration, 23 bits interrupt mask and 1bit L\_EN signal. L\_EN signal means the layer enable signal. Interrupt mask is used to mask the interrupt signal from each sub block.

Table 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8bit | 120bit | | | | | | | | | | | | | | |  |
| ID | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | comments |
| 0 | CNN Configuration | | | | | | | | | | | | Interrupt mask | | L\_EN | Block Setting/  SCU Setting |
| 1 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 2 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 3 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 4 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 5 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 6 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 7 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 8 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 9 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 10 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 11 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 12 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 13 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 14 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| 15 | CNN Configuration | | | | | | | | | | | | | | | Block Setting |
| others | N/A | | | | | | | | | | | | | | | undefined |

2) Accelerator Command Implementation



Fig.5

Deep learning accelerator is programmed by register writing through external bus. One layer setting need one set of order-fixed command which means user only follow guide to configure register and implement layer-based computing. L\_EN in ID 0 is an enable signal of one layer setting, which indicates one layer configuration is available by previous register setting when L\_EN is high.

Fig.5(a) shows the one layer setting by full command way. Fig.5(b) shows multi-layer setting by full command way. If some parameters in deep learning accelerator have been configured by previous setting, there is no necessary to re-configure them unless the parameters need to be changed in new layer. The user can use part command way to configure registers. Fig.5(c) shows mult-layer setting by part command way.

3) Interrupt

Interrupt signal indicates whether the block has finished computing. There are 2 types of interrupts. One is between branch unit and central unit. The branch unit will arbitrate work done signal which are generated by each block. And then sends corresponding interrupt signal to central unit. Another is to external SOC interrupt signal. This type signal generated by central unit by current setting and inner interrupts signal. The implementation mechanism is illustrated in Fig.6



Fig.6

## Chapter 4. NN generator