## Controller – Overview

### Operation Set

In this design, computations are grouped by a set of functions which are supported by the blocks in the processing core. Each of the function set is called an *operation* (which is typed with italic capital letters). The operations supported by the DLA are listed in Table 1:

**Table 1 Operations of DLA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operations** | **Full name** | **Processing blocks** | **Control blocks** | **Descriptions** |
| *COMP\_CONV* | Computation of Convolution | KCE/LPE/WB/LB/GB | dla\_ctrl\_conv/dla\_wbload/dla\_wbload\_mng/dla\_ctrl\_lpe | Computing convolutions and pooling operations in one Round A, including weight loading and psum storage. |
| *COMP\_FC* | Computationof Full connection | KCE/LPE/ LB/GB | dla\_ctrl\_fc/dla\_wbload/dla\_wbload\_mng/dla\_ctrl\_lpe | Fully connected matrix multiplication in one Round A, including psum storage. |
| *COMP\_RESHAPE* | Reshape | KCE/LPE/ LB/GB | dla\_ctrl\_reshape/ dla\_ctrl\_lpe | Move data in one GB to selected GBs |
| *COMP\_APE* | Computation related to APE | APE/GB | dla\_ctrl\_ape | Do element-wise multiplication/add, instant No. multiplication/add, and piece-wise non-linear function with APE, including output feature storage to GBs |
| *MOV\_GB2LB* | GB to LB (uni-directional) | LB/GB | dla\_ctrl\_gb2lb | Cache GB data in corresponding LB |
| *MOV\_DDR2GB* | GB to DDR (bi-directional) | GB | dla\_hzzm\_mng | Move data between GB and off-chip DDR |
| *MOV\_SOC2GB* | GB to SOC (bi-directional) | GB/Auxiliary buffer | (Processor) | Move data between GB/Auxiliary buffer and memories in the SoC under the control of processor |

### Architecture



**Figure 1. Block diagram of DLA controller**

Figure 1 shows the architecture of DLA controller. There are three kinds of building blocks in the controller: 1) registers, 2) gateways, and 3) control logic.

1. **Registers**

There are two kinds of registers: surface registers and staging registers.

The surface registers have independent address apace and are accessible by the CPU in the system. Each operation has its private surface registers. Before the operation, parameter that will be used during the operation are transferred to corresponding registers by the CPU. The parameters that are transferred are called “***Instruction***” (which is different from conventional instruction concept used in CPU), and this process is called ***Instruction Issue***. Only when the *Instruction Issue* is finished will the DLA begin to process the corresponding operation.

The staging registers are adopted as static instruction storage during operation processing. After *Instruction Issue*, the CPU will kick off the operation by copying the instructions in the surface registers into staging registers, if the DLA is idle. Then the DLA start to process the operation, during which the instructions in the staging registers keep unchanged. At the same time, CPU can issue the next instruction into the surface registers. This is to explore the parallelism between DLA processing and CPU instruction issue.

1. **Gateways**

There are two kinds of gateways in the DLA controller: RIF Bus and Arbiters.



**Figure 2 Flowchart of DLA operations**

### I/O Defination

|  |  |  |  |
| --- | --- | --- | --- |
| **Signals** | **Direction** | **Bit Width** | **Description** |
| **Global Signals** | | | |
| clk | input | 1 |  |
| rst | input | 1 |  |
| **SOC-HZZ Slave Interface** | | | |
| hzzs\_mosi | input | 32 |  |
| hzzs\_miso | output | 32 |  |
| hzzs\_mosi\_en | output | 1 | tri\_state input enable |
| hzzs\_miso\_en | output | 1 | tri\_state output enable |
| hzzs\_miso\_valid | output | 1 |  |
| hzzs\_mosi\_valid | input | 1 |  |
| **DDR-HZZ Master Interface** | | | |
| hzzm\_miso\_clk | input | 1 |  |
| hzzm\_mosi | output | HZZ\_T2D\_WIDTH |  |
| hzzm\_miso | input | HZZ\_T2D\_WIDTH |  |
| hzzm\_mosi\_en | output | 1 |  |
| hzzm\_miso\_en | output | 1 |  |
| hzzm\_miso\_valid | input | 1 |  |
| hzzm\_miso\_valid | output | 1 |  |
| **Global Buffer Interface – MOV SOC2GB** | | | |
| bif\_gb\_soc2gb\_gb\_sel | output | 1 |  |
| bif\_gb\_soc2gb\_addr | output | 13 |  |
| bif\_gb\_soc2gb\_ram\_sel | output | 16 |  |
| bif\_gb\_soc2gb\_wen | output | 1 |  |
| bif\_gb\_soc2gb\_ren | output | 1 |  |
| bif\_gb\_soc2gb\_wdata | output | 256 |  |
| bif\_gb\_soc2gb\_rdata | input | 16 \* 16 |  |
| **Global Buffer Interface – MOV DDR2GB** | | | |
| bif\_gb\_ddr2gb\_ab\_sel | output | 1 |  |
| bif\_gb\_ddr2gb\_addr | output | 13 |  |
| bif\_gb\_ddr2gb\_wdata | output | HZZ\_T2D\_WIDTH |  |
| bif\_gb\_ddr2gb\_wen | output | 1 |  |
| bif\_gb\_ddr2gb\_ram\_idx | output | 4 |  |
| bif\_gb\_ddr2gb\_ren | output | 1 |  |
| bif\_gb\_ddr2gb\_rdata | input | HZZ\_T2D\_WIDTH |  |
| **Global Buffer Interface – MOV GB2LB** | | | |
| bif\_gb\_gb2lb\_ren | output | 1 |  |
| bif\_gb\_gb2lb\_addr | output | 13 |  |
| **Global Buffer Interface – LPE** | | | |
| bif\_gb\_lpe\_addr | output | 13 |  |
| bif\_gb\_lpe\_ren | output | 1 |  |
| bif\_gb\_lpe\_wen | output | 1 |  |
| **Global Buffer Interface – COMP APE** | | | |
| bif\_gb\_ape\_addr | output | 13 |  |
| bif\_gb\_ape\_wen | output | 1 |  |
| bif\_gb\_ape\_ren | output | 1 |  |
| **Local Buffer Interface – MOV GB2LB** | | | |
| bif\_lb\_gb2lb\_addr | output | 11 |  |
| bif\_lb\_gb2lb\_wen | output | 1 |  |
| **Local Buffer Interface – COMP CONV** | | | |
| bif\_lb\_conv\_hori | output | 6 |  |
| bif\_lb\_conv\_vert | output | 6 |  |
| bif\_lb\_conv\_ren | output | 1 |  |
| bif\_lb\_conv\_pad\_num | output | 16 |  |
| bif\_lb\_conv\_pad\_enable | output | 1 |  |
| **Local Buffer Interface – COMP FC** | | | |
| bif\_lb\_fc\_addr | output | 11 |  |
| bif\_lb\_fc\_ren | output | 1 |  |
| **Local Buffer Interface – COMP RESHAPE** | | | |
| bif\_lb\_reshape\_addr | output | 11 |  |
| bif\_lb\_reshape\_ren | output | 1 |  |
| **Weight Buffer Interface – Weight Buffer LOAD** | | | |
| bif\_wb\_wbload\_wdata | output | 256 |  |
| bif\_wb\_wbload\_waddr | output | 4 |  |
| bif\_wb\_wbload\_wen | output | 16 |  |
| **Weight Buffer Interface – COMP CONV** | | | |
| bif\_wb\_conv\_raddr | output | 4 |  |
| bif\_wb\_conv\_ren | output | 1 |  |
| bif\_wb\_conv\_switch | output | 1 |  |
| **Weight Buffer Interface – COMP FC** | | | |
| bif\_wb\_fc\_ren | output | 1 |  |
| bif\_wb\_fc\_switch | output | 1 |  |
| **CPE Control Signals** | | | |
| ctrl\_pool\_mode | output | 1 |  |
| ctrl\_conv\_kpe\_rst | output | 1 |  |
| ctrl\_conv\_adt\_fifo\_set | output | 1 |  |
| ctrl\_fc\_kpe\_acc\_rst | output | 1 |  |
| ctrl\_fc\_adt\_fifo\_set | output | 1 |  |
| ctrl\_fc\_ifmap\_is\_zero | input | 1 |  |
| ctrl\_fc\_ifmap\_enable | output | 1 |  |
| ctrl\_reshape\_adt\_fifo\_set | output | 1 |  |
| **LPE Control Signals** | | | |
| ctrl\_adt\_fifo\_set | input | 1 |  |
| ctrl\_adt\_fifo\_ren | output | 1 |  |
| ctrl\_lpe\_relu | output | 1 |  |
| ctrl\_lpe\_acc\_bypass | output | 1 |  |
| ctrl\_lpe\_sprmps | output | 2 |  |
| **APE Control Signals** | | | |
| ctrl\_ape\_sa\_en | output | 1 |  |
| ctrl\_ape\_ab\_en | output | 1 |  |
| ctrl\_ape\_sb\_en | output | 1 |  |
| ctrl\_ape\_mul\_en | output | 2 |  |
| ctrl\_ape\_add\_en | output | 1 |  |
| ctrl\_ape\_mode | output | 3 |  |
| ctrl\_ape\_imm | output | 16 |  |
| **Staging Registers - GLOBAL** | | | |
| stgr\_enable\_row | output | 16 |  |
| stgr\_enable\_col | output | 16 |  |
| stgr\_buf\_mask | output | 256 |  |
| stgr\_status | output | 3 |  |
| stgr\_pool\_enable | output | 1 |  |
| **Staging Registers - PRECISION** | | | |
| stgr\_precision\_ape\_shift | output | 4 |  |
| stgr\_precision\_kpe\_shift | output | 4 |  |
| stgr\_precision\_ifmap | output | 1 |  |
| stgr\_precision\_weight | output | 2 |  |
| **Interrupt** | | | |
| interrupt | output | 1 |  |