## Controller – LPE

### General Overview

This module, which is abbreviated to dla\_crtl\_lpe, computes address of global buffer needed by LPE and generates corresponding control signals according to the related registers and some control signals to decide the behaviors of LPE

### I/O Definition

**Table 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Signals** | **Direction** | **Bit Width** | **Description** |
| **Global Signals** | | | |
| clk | input | 1 |  |
| rst | input | 1 |  |
| **Instruction Signals** | | | |
| start\_lpe | input | 1 | Operation start signal of LPE |
| stgr\_lpe\_src\_addr | input | 1 | Start address of the source data |
| stgr\_lpe\_src\_len | input | 13 | The length of the source data rectangle |
| stgr\_lpe\_src\_skip | input | 13 | The address gap between two source data rows |
| stgr\_lpe\_src\_iter | input | 13 | The number of rows of the source data rectangle |
| stgr\_lpe\_dest\_addr | input | 13 | Start address of the destination data |
| stgr\_lpe\_dest\_len | input | 13 | The length of the destination data rectangle |
| stgr\_lpe\_dest\_skip | input | 13 | The address gap between two destination data rows |
| stgr\_lpe\_dest\_iter | input | 13 | The number of rows of the destination data rectangle |
| stgr\_lpe\_overlay | input | 7 | Indicates the number of slices of a kernel |
| stgr\_lpe\_relu | input | 1 | Directly connected to “ctrl\_lpe\_relu” |
| stgr\_lpe\_acc\_bypass | input | 1 | Directly connected to “ctrl\_lpe\_acc\_bypass” |
| ctrl\_adt\_fifo\_set | input | 1 | Enable signal of writing data to the fifo behind the adder tree in LPE |
| stgr\_lpe\_sprmps | input | 2 | Directly connected to “ctrl\_lpe\_sprmps” |
| **Ctrl Signals** | | | |
| ctrl\_lpe\_gb\_addr | output | 13 | The address of global buffer that LPE needs to read from or write to |
| ctrl\_lpe\_gb\_ren | output | 1 | read enable signal for LPE when  it needs to read data from global buffer. The corresponding address signal has a delay of one clock cycle |
| ctrl\_lpe\_gb\_wen | output | 1 | write enable signal for LPE when it needs to write data to global buffer. The corresponding address signal has a delay of one clock cycle |
| ctrl\_adt\_fifo\_ren | output | 1 | Enable signal for LPE to read data from the fifo behind the adder tree |
| ctrl\_lpe\_relu | output | 1 | Enable relu function of LPE |
| ctrl\_lpe\_acc\_bypass | output | 1 | Bypass the accumulater behind the adder tree in LPE |
| ctrl\_lpe\_sprmps | output | 2 | Indicates the work modes of LPE, there are three modes: “image”, “none” and “all” |
| complete\_lpe | output | 1 | Indicates the complete of current operation of LPE |

### Description

This module is mainly made up by four blocks: b\_ctrl\_lpe\_raddr, b\_crtl\_lpe\_waddr, b\_ctrl\_lpe\_fsm and b\_ctrl\_lpe\_complete. The function of b\_ctrl\_lpe\_raddr is to compute the correct address of global buffer when the LPE needs to read data from and the function of b\_ctrl\_lpe\_waddr is to compute the correct address of global buffer when the LPE needs to write data to global buffer. Block b\_ctrl\_lpe\_fsm is a finite state machine, which is used to control the work state of LPE. Finally, the function of b\_ctrl\_lpe\_complete is to decide the value of complete\_lpe which indicates whether LPE completes current operation. The details of these blocks will be illustrated below.

#### b\_ctrl\_lpe\_raddr and b\_ctrl\_lpe\_waddr

The architectures of b\_ctrl\_lpe\_raddr and b\_ctrl\_lpe\_waddr are same. Use b\_ctrl\_lpe\_raddr for an example, this block uses two registers named gb\_raddr and gb\_rleap(when LPE needs to write multiple results to global buffer, gb\_rleap stores the offsets between two continuous results) to store parts of the final address, which equal to the sum of gb\_raddr and gb\_rleap. The block use several counters to decide the values of gb\_raddr and gb\_rleap in different situations. Picture below show the diagram of b\_ctrl\_lpe\_raddr.



Figure 1. Diagram of b\_ctrl\_lpe\_raddr

Depends on the ctrl\_lpe\_gb\_ren signal, only one address(generated by b\_ctrl\_lpe\_raddr or b\_ctrl\_lpe\_waddr) can be passed to the LPE as ctrl\_lpe\_gb\_addr.

#### b\_ctrl\_lpe\_fsm

Sometimes, the results of adder tree need to do addition with the number read from the corresponding global buffer before write back to the global buffer. But sometimes the results of adder tree can directly write back to the global buffer. The pipelines of two data-paths are different, so the controller use a fsm to help generate correct control signals at the right time according to whether the addition is needed. Picture below illustrates the state transitions of this fsm.



Figure 2. Diagram of b\_ctrl\_lpe\_fsm

The time sequences of some control signals depend on the value of stgr\_lpe\_acc\_bypass, Figure 3 and Figure 4 show the difference.

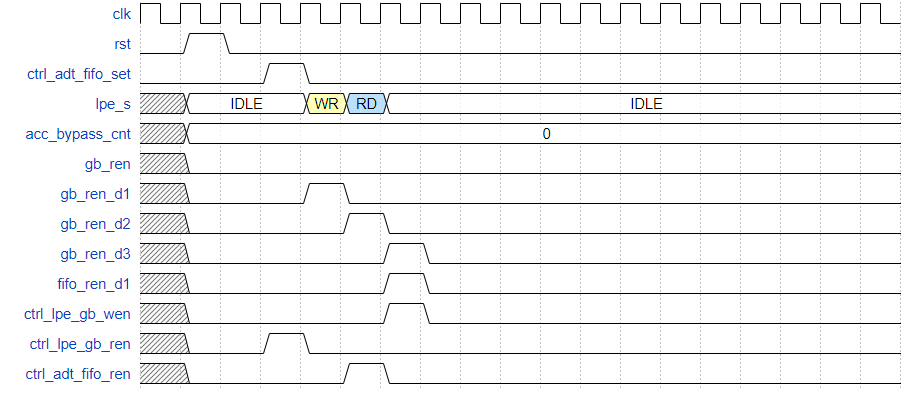


Figure 3. ACC\_BYPASS(cnt\_leap\_max = 0)

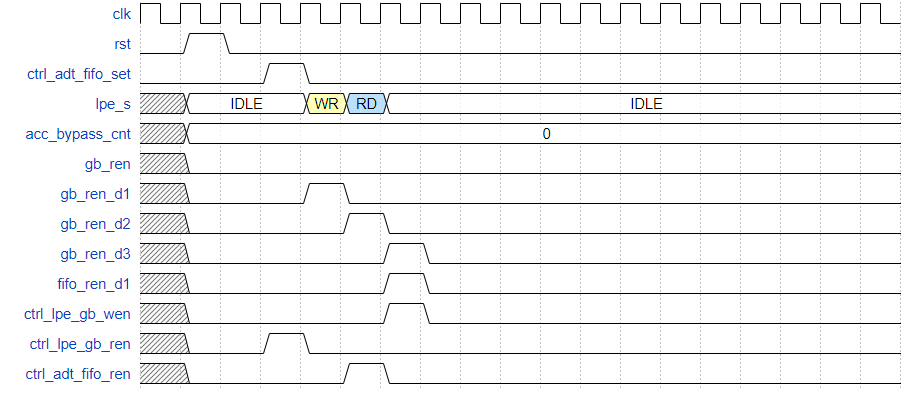


Figure 4. ACC NOT BYPASS(cnt\_leap\_max = 3)

#### b\_ctrl\_lpe\_complete

This block uses write enable signal of global buffer and values of several counters to judge if LPE complete current operation then outputs complete\_lpe signal. The logic of this block is below.



Figure 5. Diagram of b\_ctrl\_lpe\_complete

#### Other Logic

If the kernel of a convolution operation is large, the kernel will be divided into several slices. Ane the LPE handles the results caused by slices of the same kernel with the same read address and write address. In the controller for LPE, two counters(cnt\_src\_overlay and cnt\_dest\_overlay) are used to indicate whether all slices are handled. The logic of the counter is below(counters for read address generation and write address generation are same).



Figure 6. The Logic of cnt\_src\_overlay