**GradientsMulti IP**

**Inputs**

1. S00\_AXI
   1. [IP: axi\_interconnect\_1] M01\_AXI → S00\_AXI
2. param\_ready\_0 [1-bit]
   1. [IP: ParametersMulti\_0] param\_done\_0 → param\_ready\_0
3. width\_0 [32-bits]
   1. [IP: ParametersMulti\_0] width\_0 → width\_0
4. height\_0 [32-bits]
   1. [IP: ParametersMulti\_0] height\_0 → height\_0
5. dout\_ints\_0 [32-bits]
   1. [IP: MUX\_0] ref\_img\_out\_0 → dout\_ints\_0
6. s00\_axi\_aclk [1-bit]
   1. [IP: zynq\_ultra\_ps\_e\_0] pl\_clk0 → s00\_axi\_aclk
7. s00\_axi\_aresetn [1-bit]
   1. [IP: rst\_ps8\_0\_100M] peripheral\_aresetn → s00\_axi\_aresetn

**Associated IPs (inputs):**

1. zynq\_ultra\_ps\_e\_0
2. rst\_ps8\_0\_100M
3. axi\_interconnect\_1
4. ParametersMulti\_0
5. Interface\_0

**Outputs**

1. grad\_ea\_0 [1-bit]
   1. grad\_ea\_0 → ena [IP: blk\_mem\_gen\_3]
   2. grad\_ea\_0 → enb [IP: blk\_mem\_gen\_3]
   3. grad\_ea\_0 → ena [IP: blk\_mem\_gen\_4]
   4. grad\_ea\_0 → enb [IP: blk\_mem\_gen\_4]
2. ints\_ea\_0 [1-bit]
   1. ints\_ea\_0 → enb [IP:blk\_mem\_gen\_0]
   2. ints\_ea\_0 → enb [IP:blk\_mem\_gen\_1]
3. grad\_wea\_0 [1-bit]
   1. grad\_wea\_0 → wea [IP: blk\_mem\_gen\_3]
   2. grad\_wea\_0 → web [IP: blk\_mem\_gen\_3]
   3. grad\_wea\_0 → wea [IP: blk\_mem\_gen\_4]
   4. grad\_wea\_0 → web [IP: blk\_mem\_gen\_4]
4. ints\_wea\_0 [1-bit]
   1. ints\_wea\_0 → grad\_wea\_ints\_0 [IP: Interface\_0]
5. addr\_ints\_0 [17-bits]
   1. addr\_ints\_0 → grad\_addr\_ints\_0 [IP: Interface\_0]
6. addr\_grad\_x\_0 [17-bits]
   1. addr\_grad\_x\_0 → addra [IP: blk\_mem\_gen\_3]
7. addr\_grad\_y\_0 [17-bits]
   1. addr\_grad\_y\_0 → addra [IP: blk\_mem\_gen\_4]
8. din\_grad\_x\_0 [32-bits]
   1. din\_grad\_x\_0 → dina [IP: blk\_mem\_gen\_3]
9. din\_grad\_y\_0 [32-bits]
   1. din\_grad\_y\_0 → dina [IP: blk\_mem\_gen\_4]
10. din\_ints\_0 [32-bits]
    1. din\_ints\_0 → dinb [IP: blk\_mem\_gen\_0]
    2. din\_ints\_0 → dinb [IP: blk\_mem\_gen\_1]
11. grad\_done\_0 [1-bit]
    1. grad\_done\_0 → ready\_Grad\_0 [IP: Gamma\_Imp\_0]\
    2. grad\_done\_0 → probe\_in# [IP: VIO]
12. out\_frame\_counter\_0 [32-bits]
    1. out\_frame\_counter\_0 → frame\_counter\_0 [IP: MUX\_0]
    2. out\_frame\_counter\_0 → probe\_in# [IP: VIO]
13. grad\_busy\_0 [1-bit]
    1. grad\_busy\_0 → grad\_busy\_0 [IP: Interface\_0]
    2. grad\_busy\_0 → grad\_busy\_0 [IP: Counter\_0]
14. grad\_idle\_counter\_0 [128-bits]
    1. grad\_idle\_counter\_0 → probe\_in# [IP: VIO]

**Associated IPs (outputs):**

1. blk\_mem\_gen\_0 [BRAM 0]
2. blk\_mem\_gen\_1 [BRAM 1]
3. blk\_mem\_gen\_3 [BRAM 3]
4. blk\_mem\_gen\_4 [BRAM 4]
5. Interface\_0
6. Gamma\_Imp\_0

**IP Description**

This IP is responsible for computing the gradients of the images (gradients are the differences between the intensities of two adjacent pixels both in x-direction and y-direction). In this case, by considering the optimization method and the correlation routine, we use the reference image for computing the gradients. Whenever a new reference image is stored in BRAM 0 or BRAM 1, this IP should start computing its gradients. In general, GradientsMulti\_0 IP reads the intensities from BRAM 0 or 1, iterates over the whole image in both x and y directions and computes the difference between the intensities of each pixel and the pixel below it (for x-direction gradients) and the intensity of each pixel and its right-hand side pixel (for y-direction gradients), and then saves the result in BRAM 3 and BRAM 4 for x-direction and y-direction, respectively. In the first state, the param\_ready\_0 and internal register new\_frame is checked to make sure that the IP starts when height\_0 and width\_0 are set by the ParametersMulti\_0 IP. BRAM control signals of grad\_ea\_0, ints\_ea\_0, grad\_wea\_0, and ints\_wea\_0 are also set within the first state. The loop over the whole image starts at the second state (when we know we have had a new reference frame saved into BRAMs). Using two nested for loops the pixels are accessed and the floating-point subtractor is called. The address to the BRAM 0 or 1 (addr\_ints\_0) is set based on the loop counters and image width and height. It is important to note that based on the DICe source code, a border of width two pixels from the frame edges is considered for computing the gradients. So, the gradients of the pixels within the border of the image are computed using a simple subtraction between the intensities and for the pixels that are not on the border, a weighted subtraction is conducted. The internal registers of clock counters within the IP are used to count the clock cycles when we read the intensities from BRAM 0 or 1. It is because the read operation in BRAMs takes two clock cycles. So, we wait for two clock cycles to have the valid data on BRAM data out port (dout\_ints\_0) then the subtractor is called. This IP also is responsible for forwarding the frame\_counter to the Interface\_0 IP for switching the BRAMs between the reference and deformed frames. The frame\_counter variable is an internal slave register to the GradientsMulti\_0 IP, not the Interface\_0, just because for some reason we could not add this slave register directly to the Interface\_0 IP; it was a way around a bug we faced during development. So, GradientsMulti\_0 does not need and use this variable and it acts like a buffer which forwards the frame\_counter to the Interface\_0 IP.