**ParametersMulti IP**

**Inputs**

1. S00\_AXI
   1. [IP: axi\_interconnect\_1] M04\_AXI → S00\_AXI
2. data\_in\_0 [32-bits]
   1. [IP: blk\_mem\_gen\_2] doutb → data\_in\_0
3. s00\_axi\_aclk [1-bit]
   1. [IP: zynq\_ultra\_ps\_e\_0] pl\_clk0 → s00\_axi\_aclk
4. 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. blk\_mem\_gen\_2 [BRAM 2]

**Outputs**

1. we\_0 [4-bits]
   1. we\_0 → web [IP: blk\_mem\_gen\_2]
2. ea\_0 [1-bit]
   1. ea\_0 → enb [IP: blk\_mem\_gen\_2]
3. addr\_0 [32-bits]
   1. addr\_0 → addrb [IP: blk\_mem\_gen\_2]
4. num\_of\_bits\_0 [32-bits]
   1. \*\* UNUSED
5. num\_of\_pxl\_0 [32-bits]
   1. num\_of\_pxl\_0 → num\_pixels\_0 [IP: Gamma\_Imp\_0]
6. num\_of\_subsets\_0 [32-bits]
   1. num\_of\_subsets\_0 → num\_of\_subsets\_0 [IP: Gam\_Interface\_0]
   2. num\_of\_subsets\_0 → probe\_in# [IP: VIO]
7. width\_0 [32-bits]
   1. width\_0 → width\_0 [IP: GradientsMulti\_1]
   2. width\_0 → probe\_in# [IP: VIO]
8. height\_0 [32-bits]
   1. height\_0 → height\_0 [IP: GradientsMulti\_1]
   2. height\_0 → probe\_in# [IP: VIO]
9. param\_bram\_dout\_0 [32-bits]
   1. param\_bram\_dout\_0 → dinb [IP: blk\_mem\_gen\_2]
10. optimization\_method\_0 [32-bits]
    1. optimization\_method\_0 → optimization\_method\_0 [IP: Gamma\_Imp\_0]
11. correlation\_routine\_0 [32-bits]
    1. correlation\_routine\_0 → correlation\_routine\_0 [IP: Gamma\_Imp\_0]
12. param\_done\_0 [1-bit]
    1. param\_done\_0 → param\_ready\_0 [IP: GradientsMulti\_1]
    2. param\_done\_0 → parameters\_done\_0 [IP: Gam\_Interface\_0]
    3. param\_done\_0 → parameters\_done\_0 [IP: Coords\_Interface\_0]
    4. param\_done\_0 → param\_ready\_0 [IP: SubsetCoordsMulti\_0]
    5. param\_done\_0 → ready\_Params\_0 [IP: Gamma\_Imp\_0]
    6. param\_done\_0 → probe\_in# [IP: VIO]

**Associated IPs (outputs):**

1. blk\_mem\_gen\_2 [BRAM 2]
2. Gamma\_Imp\_0
3. Gam\_Interface\_0
4. SubsetCoordsMulti\_0
5. GradientsMulti\_1
6. Coords\_Interface\_0
7. VIO

**IP Description**

The ParametersMulti\_0 IP is a simple IP that is responsible for sending user-defined parameter data to multiple IPs that require it to perform DIC. The IP has one input, labeled “data\_in” that is connected to BRAM 2. Before any frames are sent to the FPGA from the Python script on the PC, first a .txt file is defined and sent over ethernet to the FPGA and saved into BRAM 2. The Parameters IP then is responsible for reading most of this data from the BRAM to the Gamma\_Imp\_0, Gam\_Interface\_0, SubsetCoordsMulti\_0, GradientsMulti\_1, and Coords\_Interface\_0 IPs. The code within the Parameters IP is relatively straight forward. It starts with an address that reads from BRAM and then follows with two “dummy” states that are used to ensure that the data has enough time to be read from memory. After it has read the first line of data, it increments the address to pull out the next line of data and then the pattern continues. \*NOTE: two dummy states are used after each data assignment state because BRAM requires two clock cycles to perform a read operation. However, if URAM is used as the source of parameter data then these dummy states can be reduced from two states to one state because URAM only requires one clock cycles for a read operation.

The Parameters IP has an internal AXI slave register defined as an input labeled “param\_start”. This register is written to by the echo.c server whenever it has verified that all user-defined parameter data has been received from the PCs Python script. The Parameters IP is responsible for collecting the following data from BRAM and sending to various other IPs: num\_of\_bits (number of bits in a frame), num\_of\_pxl (number of pixels in a frame), num\_of\_subsets (number of subsets defined for correlation; also known in DICe as regions of interest (ROIs)), width\_ (the width of the image in pixels), height\_ (the height of the image in pixels), optimization\_method (defines which method of correlation will be used; gradient-based: 0, or simplex-based: 1), and correlation\_routine (defines which correlation method will be used; fast method - 0 [for gradient-based optimization], or robust method - 1 [for simplex based optimization]). \*NOTE: gradient-based optimization tracking is used for high frame counts with low subset counts. With our current design, no IP requires the need for the number of bits. Currently the Gamma\_Imp\_0 IP requires the data: correlation\_routine, optimization\_method, and the num\_of\_pxl.

The Gam\_Interface\_0 IP requires the num\_of\_subsets signal so it can interface the subsets with the Gamma\_Imp\_0 IP. The height\_ and width\_ signals are required by the GradientsMulti\_1 IP to determine the gradients of each pixel in the image based on the image size. The signal   
param\_bram\_dout\_0” was defined to get rid of an error that appeared that required the BRAM 2 dinb port to be connected; this usually shouldn’t be necessary. Lastly, the “param\_done\_0” signal is set at the last state of the Parameters IP and is used to tell the GradientsMulti\_1, SubsetCoordsMulti\_0, Coords\_Interface\_0, Gam\_Interface\_0, and Gamma\_Imp\_0 IPs that they may be potentially ready to start their processing because the user-defined parameters data is ready to be used. Most of these IPs also wait for image data from other IPs before starting - but they need to know that the parameters are set.