**Gam\_Interface IP**

**Inputs**

1. S00\_AXI
   1. [IP: axi\_interconnect\_1] M08\_AXI → S00\_AXI
2. gam\_new\_subset\_0 [1-bit]
   1. [IP: Gamma\_Imp\_0] gam\_new\_subset\_0 → gam\_new\_subset\_0
3. subset\_done\_0 [1-bit]
   1. [IP: SubsetCoordsMulti\_0] sub\_done\_0 → subset\_done\_0
4. num\_of\_subsets\_0 [32-bits]
   1. [IP: ParametersMulti\_0] num\_of\_subsets\_0 → num\_of\_subsets\_0
5. subset\_counter\_0 [32-bits]
   1. [IP: SubsetCoordsMulti\_0] subset\_counter\_0 → subset\_counter\_0
6. gam\_subset\_number\_0 [32-bits]
   1. [IP: Gamma\_Imp\_0] gid\_0 → gam\_subset\_number\_0
7. parameters\_done\_0 [1-bit]
   1. [IP: ParametersMulti\_0] param\_done\_0 → parameters\_done\_0
8. param\_dout\_0 [32-bits]
   1. [IP: blk\_mem\_gen\_9] doutb → param\_dout\_0
9. base\_address\_0 [32-bits]
   1. [IP: SubsetCoordsMulti\_0] base\_address\_0 → base\_address\_0
10. num\_pxl\_int\_in\_0 [32-bits]
    1. [IP: SubsetCoordsMulti\_0] num\_pxl\_Int\_0 → num\_pxl\_int\_in\_0
11. num\_pxl\_FP\_in\_0 [32-bits] num\_pxl\_FP\_0 → num\_pxl\_FP\_in\_0
    1. [IP: SubsetCoordsMulti\_0]
12. s00\_axi\_aclk [1-bit]
    1. [IP: zynq\_ultra\_ps\_e\_0] pl\_clk0 → s00\_axi\_aclk
13. 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. Gamma\_Imp\_0
5. SubsetCoordsMulti\_0
6. ParametersMulti\_0
7. blk\_mem\_gen\_9 [BRAM 9]

**Outputs**

1. param\_ea\_0 [1-bit]
   1. param\_ea\_0 → enb [IP: blk\_mem\_gen\_9]
2. param\_wea\_0 [4-bits]
   1. param\_wea\_0 → web [IP: blk\_mem\_gen\_9]
3. param\_addr\_0 [32-bits]
   1. param\_addr\_0 → addrb [IP: blk\_mem\_gen\_9]
   2. param\_addr\_0 → probe\_in# [IP: VIO]
4. gam\_cx\_0 [32-bits]
   1. gam\_cx\_0 → cx\_0 [IP: Gamma\_Imp\_0]
   2. gam\_cx\_0 → probe\_in# [IP: VIO]
5. gam\_cy\_0 [32-bits]
   1. gam\_cy\_0 → cy\_0 [IP: Gamma\_Imp\_0]
   2. gam\_cy\_0 → probe\_in# [IP: VIO]
6. gam\_interface\_done\_0 [1-bit]
   1. gam\_interface\_done\_0 → gam\_interface\_done\_0 [IP: Gamma\_Imp\_0]
7. base\_addr\_out\_0 [32-bits]
   1. base\_addr\_out\_0 → base\_address\_0 [IP: Gamma\_Imp\_0]
   2. base\_addr\_out\_0 → probe\_in# [IP: VIO]
8. num\_pxl\_int\_out\_0 [32-bits]
   1. num\_pxl\_int\_out\_0 → num\_pxl\_Int\_0 [IP: Gamma\_Imp\_0]
   2. num\_pxl\_int\_out\_0 → probe\_in# [IP: VIO]
9. num\_pxl\_FP\_out\_0 [32-bits]
   1. num\_pxl\_FP\_out\_0 → num\_pxl\_FP\_0 [IP: Gamma\_Imp\_0]
   2. num\_pxl\_FP\_out\_0 → probe\_in# [IP: VIO]

**Associated IPs (outputs):**

1. blk\_mem\_gen\_9 [BRAM 9]
2. Gamma\_Imp\_0
3. VIO

**IP Description**

The Gam\_Interface\_0 IP is responsible for sending a few dedicated parameters to the Gamma\_Imp\_0 IP. The parameters that the Gamma\_Imp\_0 IP requires are the X and Y subset center points, the number of pixels in integer format, the number of pixels in IEEE 754 floating-point format, and the base address of where the pixels are located within the BRAMs. This IP, along with Coords\_Interface\_0 IP, was created because we cannot have multiple IP’s driving addresses to a single BRAM module that contains the user-defined parameters data. The solution around this was to create a few “interface” IPs that handle this addressing between their “parent” IPs. The user-defined parameters file represents a very small and insignificant amount of data, so we saw no problem with having this data duplicated into 3 separate BRAM modules. BRAM 9 is responsible for the parameter data that the Gam\_Interface\_0 IP uses. The IP waits for the parameter data to be finished with its writing of data into the BRAMs, after which it goes through a sequence of states that pull out the information required (listed above).

The Gam\_Interface\_0 works by accepting inputs from the SubsetCoordsMulti\_0 IP and its connected BRAM (9). This IP is crucial because it is what allows the ability to have multiple subsets within a DIC. It communicates with the SubsetCoordsMulti\_0 IP closely because it receives the X and Y subset coordinates which define where each subset is located within a frame; this happens in the signal defined as “base\_address”. The Gam\_Interface\_0 IP holds onto the location of every subset and sends this data over to the Gam\_Imp\_0 IP as it's needed to perform the correlation on that subset. The IP requires the following inputs: clock, gam\_new\_subset, susbet\_done, num\_of\_subsets, subset\_Counter, gam\_subset\_number, parameters\_done, param\_dout, base\_address, num\_pxl\_Int\_in, and num\_pxl\_FP\_in. Sub\_done lets the Gam\_Interface IP know that the SubsetCoordsMulti IP is done processing all of the X and Y subset coordinates. Gam\_new\_subset comes from the Gamma\_Imp\_0 IP and lets the Gam\_Interface IP know when it requires a new set of subset center points for correlation. The number of subsets lets the IP know how many subsets it should expect to hold and iterate through; this value comes from the parameters IP. The subset\_counter signal comes from the SubsetCoordsMulti IP and helps both IPs keep track of which subset they are working on. The gam\_subset\_number signal works with the Gamma\_Imp IP and lets it know which subset is currently running correlation so that the Gam\_Interface IP can assure that it has fed the proper subset to the Gamma IP as well as prepare to send the next subset over. The base\_address signal that comes from SubsetCoordsMulti is responsible for holding the base address of the actual information of the X and Y subset center points that are then saved within an internal register within the Gam\_Interface IP and are then looped over and sent as needed to the primary Gamma IP. The number of pixels in a subset for both integer and floating-point format is computed by the SubsetCoordsMulti IP and is required for knowing the number of pixels contained within each subset. The Gamma\_Imp IP has not started when the SubsetCoordsMulti IP is finished so the Gam\_Interface IP acts as a buffer and holds onto these values for Gamma until it is ready to start. They are contained within the Gam\_Interface IP for simplicity so that we can control when we send the X and Y center points to the Gamma IP as well as the number of pixels in the subset, and assure they are being sent at the right time with the right subset. Gam\_Interface is active for as long as Gamma\_Imp is active. Gamma\_Imp will work on its correlation between two frames and then request to do its correlation on a subset which signals the Gam\_Interface to send over the needed subset values for as many exist. When Gamma\_Imp is finished computing the first subset, it will request the next. Sending the X and Y center points from Gam\_Interface to Gamma\_Imp is identical to how the previous IPs sent over the values. The bulk of this IPs logic is wrapped up in storing the number of pixels (in integer and floating-point format) in each subset into an indexed register as well as the coordinates for each subset so that they can be made ready for the Gamma\_Imp IP when it requires them.