**Interface IP**

**Inputs**

1. S00\_AXI
   1. [IP: axi\_interconnect\_1] M02\_AXI → S00\_AXI
2. grad\_busy\_0 [1-bit]
   1. [IP: GradientsMulti\_1] grad\_busy\_0 → grad\_busy\_0
3. grad\_wea\_ints\_0 [1-bit]
   1. [IP: GradientsMulti\_1] ints\_wea\_0 → grad\_wea\_ints\_0
4. grad\_addr\_ints\_0 [17-bits]
   1. [IP: GradientsMulti\_1] addr\_ints\_0 → grad\_addr\_ints\_0
5. gamma\_addr\_ints\_ref\_0 [17-bits]
   1. [IP: Gamma\_Imp\_0] addr\_ref\_ints\_0 → gamma\_addr\_ints\_ref\_0
6. gamma\_addr\_ints\_def\_0 [17-bits]
   1. [IP: Gamma\_Imp\_0] addr\_def\_ints\_0 → gamma\_addr\_ints\_def\_0
7. frame\_counter\_0 [32-bits]
   1. [IP: GradientsMulti\_1] out\_frame\_counter\_0 → frame\_counter\_0
8. img\_in\_0\_0 [32-bits]
   1. [IP: blk\_mem\_gen\_0] doutb → img\_in\_0\_0
9. img\_in\_1\_0 [32-bits]
   1. [IP: blk\_mem\_gen\_1] doutb → img\_in\_1\_0
10. s00\_axi\_aclk [1-bit]
    1. [IP: zynq\_ultra\_ps\_e\_0] pl\_clk0 → s00\_axi\_aclk
11. 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. GradientsMulti\_1
5. Gamma\_Imp\_0
6. blk\_mem\_gen\_0 [BRAM 0]
7. blk\_mem\_gen\_1 [BRAM 1]

**Outputs**

1. ref\_img\_out\_0 [32-bits]
   1. ref\_img\_out\_0 → dout\_ints\_0 [IP: GradientsMulti\_1]
   2. ref\_img\_out\_0 → dout\_ref\_ints\_0 [IP: Gamma\_Imp\_0]
2. def\_img\_out\_0 [32-bits]
   1. def\_img\_out\_0 → dout\_def\_ints\_0 [IP: Gamma\_Imp\_0]
3. out\_grad\_wea\_ints\_0 [4-bits]
   1. out\_grad\_wea\_ints\_0 → web [IP: blk\_mem\_gen\_0]
   2. out\_grad\_wea\_ints\_0 → web [IP: blk\_mem\_gen\_1]
4. out\_grad\_gamma\_addr\_ints\_ref\_0 [32-bits]
   1. out\_grad\_gamma\_addr\_ints\_ref\_0 → addrb [IP: blk\_mem\_gen\_0]
5. out\_gamma\_addr\_ints\_def\_0 [32-bits]
   1. out\_gamma\_addr\_ints\_def\_0 → addrb [IP: blk\_mem\_gen\_1]

**Associated IPs (outputs):**

1. blk\_mem\_gen\_0 [BRAM 0]
2. blk\_mem\_gen\_1 [BRAM 1]

**IP Description**

The Interface IP is a simple IP that handles addressing and sending of image data to both the GradientsMulti\_1 IP and the Gamma\_Imp\_0 IP. This IP was the foundation for the Coords\_Interface\_0 and Gam\_Interface\_0 IP. The logic for the Interface IP is a small state machine that starts when the signal called “new\_frame” that is defined as an input signal but is really an AXI slave register that is written to by the echo.c server on the ARM. When the echo server receives a new frame, it writes to the interfaces “new\_frame” register that initiates its function. If a new frame is received then the IP sets an internal register called “waiting” to ‘0’ and then begins to read the two input signals “grad\_wea\_ints” and “grad\_busy”. The first state can be reduced as it is not totally necessary - it is currently set up the way it is for debugging purposes. Essentially the GradientsMulti\_1 IP will output a signal that tells the Interface IP to perform a read or write from BRAM 0 or BRAM 1 (which contains the reference and deformed images). However, this signal from the GradientsMulti\_1 IP is currently always set to ‘0’ which means the Interface IP should always set BRAM 0 and BRAM 1 to read mode so that they can send the image data to GradientsMulti\_1 and Gamma\_Imp\_0. State 2 reads the input signal “grad\_busy” which if it is set to a ‘1’ means the GradientsMulti\_1 IP is busy and that it needs the image data from whichever BRAM contains the reference image. The input “frame\_counter” comes from the GradientsMulti\_1 IP and tells the Interface\_IP which frame number the correlation is on and sets the corresponding address to the needed BRAM. The frame\_counter variable allows the IP to switch between reading image data and writing addresses to BRAM 0 and BRAM 1. As images come in for DIC from the PC, the first image is considered to be the reference image while the second image is considered to be the deformed image. Whenever the first correlation has been performed on the first two frames, the reference image is no longer needed and the deformed image becomes the “new” reference image; one new image comes into the FPGA and is considered to be the new deformed image. Instead of writing the entire deformed image to the reference images dedicated BRAM, the Interface IP acts as a “ping-pong” buffer that takes the data from both the reference and deformed image BRAMs and flips their context as needed. The frame\_coutner variable is set and updated in the echo.c server every time a new frame has been received and is ultimately written into the GradientsMulti\_1 IP and wired over to the Interface IP. Based on the frame number, the IP will switch the BRAM inputs and set them to the proper output. For frames 1 and 2: reference image in = reference image out; deformed image in = deformed image out. For frames 2 and 3: deformed image in = reference image out; reference image in = deformed image out. This pattern continues. The output from the Interface IP “out\_grad\_gamma\_addr\_ints\_0” is an address that's index is dependent on the output from the GradientsMulti\_1 IP, “grad\_addr\_ints”, that tells the Interface IP which line of the BRAM to read from. The address is multiplied by the number 4 because the BRAM address space is byte enabled. So, if the Gradients IP wants to access the 3rd line of data in BRAM, it will send the number 2 over (starts from 0), which is multiplied by 4 to yield 8, which is the address of line 3 of the BRAM. If the GradientsMulti\_1 IP is not busy, it means that it is finished processing the gradient values and that the Gamma\_Imp\_0 IP is busy. If this is the case, then the else condition is met that sets the addresses for both BRAMs 0 and 1 from Gamma\_Imp\_0s index signals, “gamma\_addr\_ints\_ref” and “gamma\_addr\_ints\_def”. This is because the Gamma\_Imp\_0 IP requires both the reference image and the deformed image for correlation.

After state 2 handles the addressing for both of the BRAMs, it moves onto state 3 and state 4 which are empty “dummy” states that allow two clock cycles for the address data to be set properly. Once the FSM goes to state 5, it looks at the frame counter variable again to switch the incoming BRAM 0 and 1 data to the proper reference and deformed data outputs. After this, the FSM jumps back to state 2 where it handles the next line of addressing for whichever IP has control.