Read Me

The files attached can be used for behavioral simulation and to check that the values are correct. The connect_tb.v testbench file does not check for correction of output. It is just there to show what the input and output format should be. If you need to check for correction then check the waveform for the same and it would give a clear indication of the output. connect.v is the module which connect_tb.v testbench calls. A few things to note,

- The input format for the activation is that it takes number of FFT points (defined as FFTCHNL in the code and is 8 in the case) * no. of input channels * 32 bits (16 bit data) value.
- The input format for kernel input is that it takes the value for a kernel pixel across all input and output channels, hence it takes FFTCHNL*FFTCHNL number of clocks for the whole kernel data to fit inside.
- The FFTCHNL can be changed but if you do so, please recreate the 1D FFT code for 16 input and not 8 input.
- Because we are taking so many input data per clock, there is a good chance the code would not synthesize for more than 4 input and output channels. There is an easy fix for that. Change the input format for the code (connect.v file), also in that file make sure that you do not start the systolic array computation before getting all the data, so the signals would change a bit, new signals might be needed to be added. Because the kernels would be reused for all the images, hence this delay for waiting for the input would be a one time delay. As for activation, this would also need to be reduced in size as not instead of bringing FFTCHNL number of pixels across all input channels now we would need to bring the data in chunks, though this is not that much of an issue, let us say we have 256 channels and we can only bring 16 channels worth data, because of the pipelined nature of FFT we can bring the whole data in 16 cycles. Just make sure that the FFT output of the data is fit in the correct index.
- Once the FFT and kernel are in memory, the code starts FFTCHNL*FFTCHNL number of systolic arrays and store the result in another memory. There should not be an issue with this step till 128 input channels. For 256 input channels the whole convolution step need to be divided into 2 steps of 128 channels each, for this again some new signals need to be used.

THe output is stored in memory so no need to worry about the validity of data.

- Please make sure that you have the same number of input and output channels. This is because in the systolic array code right now the input and output channels are assumed to be same. This again can be easily changed, just add a new parameter for output channel and make sure that you change the row parameter for matrix and vector. Currently they both share the same parameter hence the issue.

- From the systolic array the output is sent to memory and from there it is sent to IFFT. We do not need to worry about this. The output from that is available as output of the design. This step might again require that the output data be partitioned because of the limitation because of the number of pins.