**axis\_avg\_buffer**

Introduction

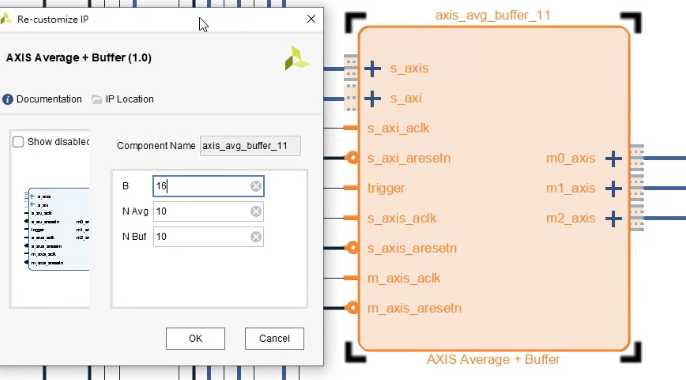
The readout ip (axis\_pfb\_readout\_v2, axis\_readout\_v3) is always processing (decimate, filter, demodulate, etc) input from ADC and send the produced outputs to axis\_avg\_buffer for averaging and temporary storage before being read back to PS side. Whether to store the processed output depends on whether a trigger signal is received. The trigger signal is sent by tproc (axis\_tproc64x32\_x8\_v1). The timing of sending trigger signal can be programed (see the section *axis\_tproc64x32\_x8\_v1* for detail). Number of data point and starting address in axis\_avg\_buffer to store data after a trigger signal is received is programed directly from python (with pynq’s mmio class)[[1]](#footnote-1).

Inside axis\_avg\_buffer there are two separate buffers, one is used to store data directly coming out from the readout ip’s (buf), another is used to store accumulated data (avg). For example, if we set length to N in python, then at the end to data acquisition, buf will contain 2N numbers (I and Q for each data point), while avg will only contain 2 number, corresponding to sum of all I and Q-components of all N acquired data points.

As for which buffer (avg or buf) to read into PS side, that depends out what python methods are called. See the section *qick python library* for details.

Specs

Note spec can vary depending on the three paremeters B, N\_Buf, N\_Avg. To be sure, you can re-create the vivado block design following the steps in the section *export & re-create vivado block design* and check it.



* Size of avg buffer: can store 210 number of I & Q data, both are 16-bits.
* Size of buf buffer: can store 210 number of I & Q data, both are 16-bits.

How to change buffer size

Double click the ip, a window will open up. Three parameter can be modified B, N\_Buf, N\_Avg, their meanings are as follow: The buf buffer can store 2N\_Buf number of I & Q data, both are 16-bits or B-bits (total size is 2N\_Buf \* 2B bits). Similarly for avg buffer. Once done, you can re-generate the bitstream (see the section *generate bitstream & load with pynq*) and then you can use it in measurement.

1. AxisAvgBuffer::config\_avg() or AxisAvgBuffer::config\_buf() -> SocIp:: \_\_setattr\_\_() -> pynq’s MMIO class. [↑](#footnote-ref-1)