**axis\_pfb\_readout\_v2**

Introduction

axis\_pfb\_readout\_v2 can separately demodulate four superposed frequencies. This ip enables reading out multiple qubits using just one ADC, which is good for scaling up to larger qubit number.

Specs

To know how to program values for the following quantities from python, see the section *qick python library*.

* DDS Frequency (fs is sampling rate of ADC): max input frequency is fs/2; min input frequencies may be limited by external RF circuits (e.g. baluns).
* Power: input power range -5V ~ 2.1V (limited by ADC) [1].
* Waveform length: max waveform length is limited by axis\_avg\_buffer ip, which is also introduced in this thesis.
* No two frequencies can be in same bin, where the size of each bin fbin is fs / 16, where fs is sampling rate of the ADC. The first bin is from -fbin/2 to fbin/2, second is from fbin/2 to fbin/2 + fbin, and so on.

How it work (optional)

Assume that:

* ADC Sampling rate fs = 800 MHz. The frequency range of interest will be 0 ~ fs/2 or 0 - 400 MHz.
* Spectrum of input signal: containing 8 components.

The spectrum is shown below.

|  |
| --- |
| C:\Users\Richard\Downloads\Untitled.png |
| x units: MHz, y unit: dBm. |

To be able to seperate out the components, we first divide it into several bins. In this example, we divide it into 8 bin, each is 50 MHz broad, centered at 0, 50, 100, …, 350 MHz. Need to make sure no two components fall in the same bin.

|  |
| --- |
| C:\Users\Richard\Downloads\2.png |
| x units: MHz, y unit: dBm. |

Next, process each bin independently. Steps for processing each bin is as follow:

Step1. shift i’th bin down such that the bin is centered at 0 MHz.

|  |
| --- |
| C:\Users\Richard\Downloads\3.png |
|  |

Step2. use DDS to remove offset of component frequencies from their bin center frequencies.

|  |
| --- |
| C:\Users\Richard\Downloads\4.png |
|  |

How to get started using it (zcu216, vivado2020.2)

The demo codes are here:

<https://github.com/Ri-chard-Wu/thesis/tree/master/codes/axis_pfb_readout_v2-test-216>

The bitstream files are also included. If you don’t know how to use the bitstream files, see the section *generate bitstream & load with pynq*.

The demo is as follow: We combine the output of 4 DAC with a 4-way combiner, and feed the combined signal into one ADC, as shown in the figure below.

|  |
| --- |
|  |
| Circled in red on the left: Combine signals from 4 DAC with a 4-way combiner. Circled in red on the right: feed combined signals into one ADC. |

Next, we let the four DAC’s play pulse at the same time with different frequencies, and then let the ADC acquire the signal. Following shows the acquired pulses for three different combinations of 4 frequencies:

|  |
| --- |
|  |
|  |

We see that the pulse’s shape looks good.

how to include it in firmware (zcu216)

IP core settings (double click on the ip):

"interleaved input" checked.

RFDC (Zynq Ultrascale+ RF Data Converter) ADC settings:

|  |  |
| --- | --- |
|  | * Digital Output Data: I/Q * Decimation Mode: 2x * Samples per AXI4-Stream Cycle: 8 * Mixer Type: Coarse * Mixer Mode: Real->I/Q * Frequency: -Fs/4 |
|  |  |

RFDC ADC tile clocking settings:

* For details about the fields see the section *rfdc settings*.
* For demo purposes, just select *Clock Source* to be the tile itself, and select *Distribution Clock* to be *off*. For details, see the section *rfdc settings*.

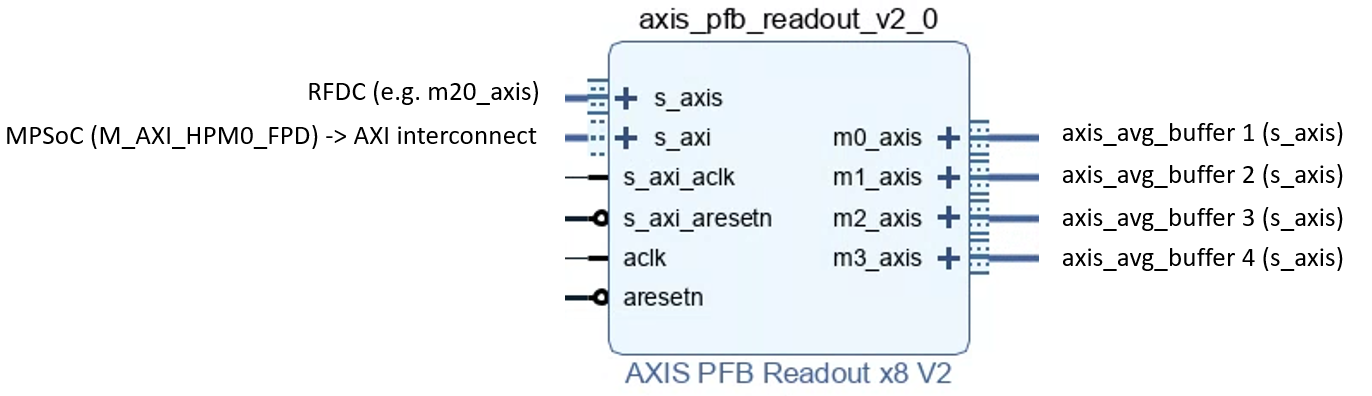


Wirings:

For full details, you can re-create the vivado block design using the scripts (bd\_216 … .tcl, proj\_216 … .tcl) at:

<https://github.com/Ri-chard-Wu/thesis/tree/master/codes/axis_pfb_readout_v2-test-216>

If you don’t know how to use the scripts, see the section *export & re-create vivado block design*.



References

[1] Zynq UltraScale+ RFSoC Data Sheet: DC and AC Switching Characteristics (DS926).