**axis\_signal\_gen\_v6**

Introduction

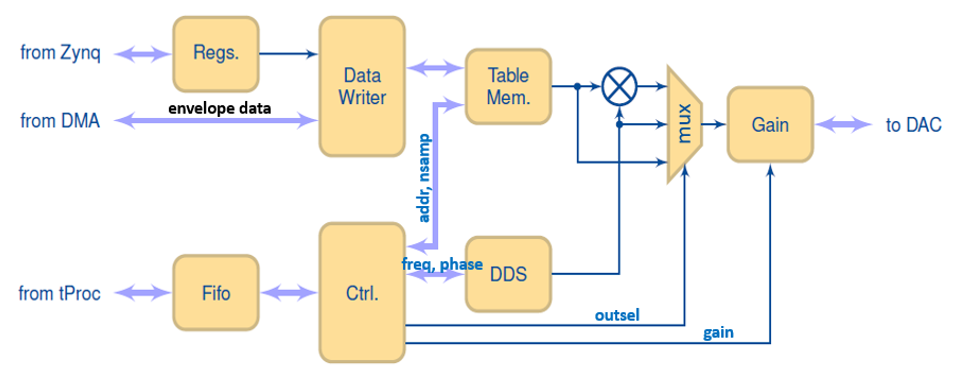
This ip receives waveform info (explained below) from tproc, and play pulse accordingly. This ip can also store envelope data.

Specs

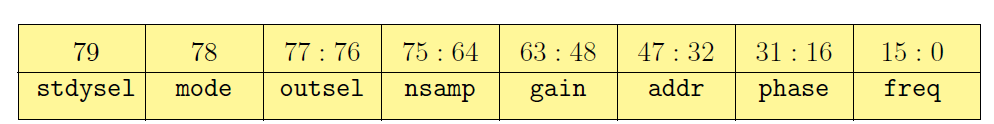
* Phase[[1]](#footnote-1): 0 to 360 degree, resolution is 32-bits (step 360/232).
* Gain: 16-bits integer (-32768~32767). (TODO: include GFP table)
* DDS Frequency (fs is sampling rate of DAC, and fdds = fs)[[2]](#footnote-2): from 0 to fdds, resolution is 32-bits (step fdds/232­­­).

(Note: DAC’s actual output frequency is DDS frequencies, together with their images (see *Sampling & re-construction* section).)

How it work



* Before writing envelope into *Table Mem*, the qick’s python lib will first write starting address and length of the envelope data into *Regs*, then begin to transfer envelope through “from DMA” into *Table Mem*.
* Waveform info is sent in from “from tProc” and queued in *Fifo*. The *Ctrl* will take out and process the queued waveform info one by one from *Fifo*.
* The waveform info contains



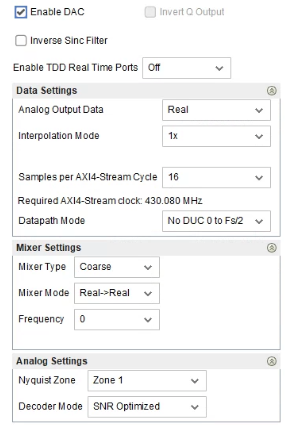
* + For *DDS* block: phases, frequency.
  + For *Table Mem*: starting address and length to readout the envelope to be multiplied with DDS’s output (sinusoidal).
* The *DDS* block contains 16 DDS’s. The DAC’s sampling rate fs need to be equal to faclk \* 16.

how to include it in firmware (zcu216, vivado2020.2)

IP core settings (double click on the ip):



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



RFDC DAC 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*.



1. relevant qick functions: QickConfig::deg2reg(). [↑](#footnote-ref-1)
2. relevant qick functions: QickConfig::freq2reg(), QickConfig::freq2int(). [↑](#footnote-ref-2)