This document has for aim to help Icestudio users to make a numerical filter working, and gives developers some informations about what has already been made and what still needs work.

I'm not expert in signal processing and HDL, then I don't plan to improve and extend this library in short-term. Do not hesitate to take over this project as you wish if you have the skills.

Filtering theory

The Laplace-transform of a signal f is given by :

$$\mathcal{L}(f) : p \to \langle f | e^{-p^* x} \rangle = \int_{\mathbb{R}} f(x) e^{-px} dx$$

Where p is a complex number. An important property is :

$$\mathcal{L}\left(\frac{df}{dt}\right) = pf$$

A filter h is a signal-processing system which describes an ordinary differential equation, and can also be wrote as a Laplace transform (which is an algebric function).

When the differential equation is linear, its Laplace transform is a rationnal fraction. For example, a first-order low-pass :

$$\frac{dy}{dt} + \frac{y}{\tau} = \frac{x}{\tau}$$

$$\mathcal{L}(h) : p \to \frac{1}{1 + \tau p}$$

However, for numerical computing applications as FPGA it's better tu use z-transform which is closer to hardware. The idea is to consider a discretized signal f where :

$$f_{i+j} = z^j f_i$$

Now, we need an integration method. The simplest is the Newton one :

$$\frac{df}{dt} = \frac{f_i - f_{i-1}}{T_e}$$

In the frequency-domain :

$$p = \frac{1 - z^{-1}}{T_e}$$

Which gives a way to obtain z-transform knowing Laplace transform. For our first-order filter example :

$$\mathcal{Z}(h)(z) = \frac{1}{1 + \tau \frac{1 - z^{-1}}{T_e}}$$

$$\mathcal{Z}(h)(z) = \frac{1}{1 + \frac{\tau}{T_e} - \frac{\tau}{T_e} z^{-1}}$$

$$\mathcal{Z}(h)(z) = \frac{\frac{1}{1 + \frac{\tau}{T_e}}}{1 - \frac{\frac{\tau}{T_e}}{1 + \frac{\tau}{T_e}} z^{-1}}$$

Where T_e is the sampling time (the clock period). We will place ourselves in a unit time-system in which $T_e = 1$, but for concretes application you'll have to take it into account.

$$\mathcal{Z}(h)(z) = \frac{\frac{1}{1+\tau}}{1 - \frac{\tau}{1+\tau} z^{-1}}$$

Another commonly-used integration method is the Tustin one, which corresponds to :

$$z = \frac{1 + p\frac{T_e}{2}}{1 - p\frac{T_e}{2}}$$

If we use this method, we'll find another z-transform for the same continuous filter, but quite close in their coefficients.

Finally, we'll distinguish two sorts of numerical filters :

— Finite impulse response filters :

$$\mathcal{Z}(h)(z) = \sum_{i=0}^{\infty} a_i z^{-i}$$

— Infinite impulse response filters :

$$\mathcal{Z}(h)(z) = \frac{\sum_{i=0}^{\infty} b_i z^{-i}}{1 + \sum_{i=1}^{\infty} a_i z^{-i}}$$

Implementation

Numbers must be coded in a signed fixed-point formats, on n bits (I think *yosys* doesn't support *real* Verilog type). All blocks have a *width* parameter which must be affected to $n \leq 64$.

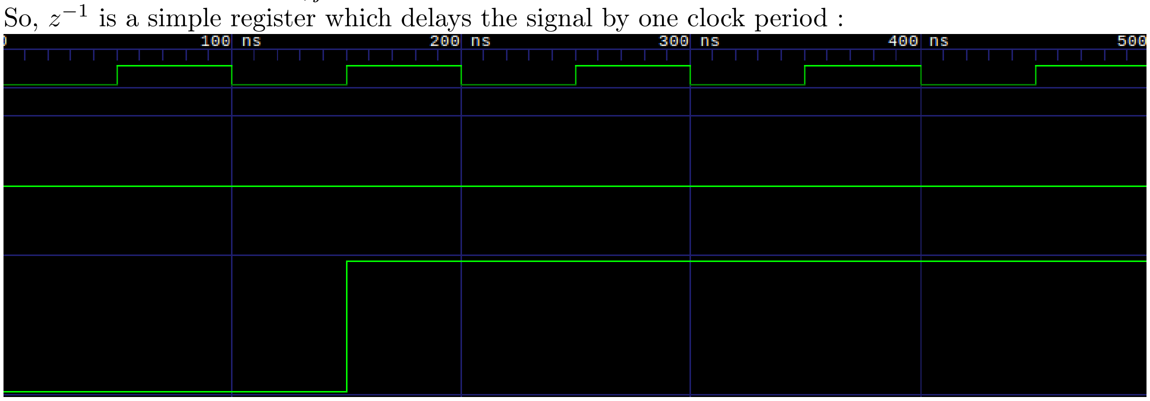
Clocks are chained to generate positive edge when the task of a block takes end and the output register is filled ; then, the next block can begin.

Overflow isn't handled. Take care.

z^{-1} block

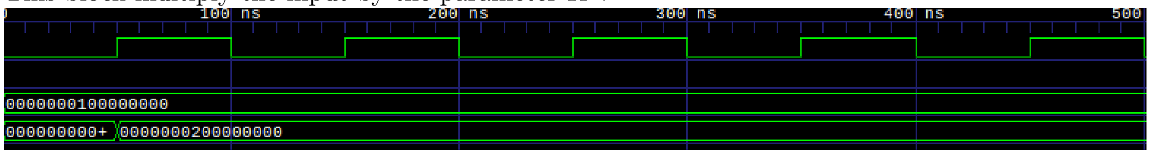
Remember that $z^j f_i = f_{i+j}$.

So, z^{-1} is a simple register which delays the signal by one clock period :



gain block

This block multiply the input by the parameter K :



add block

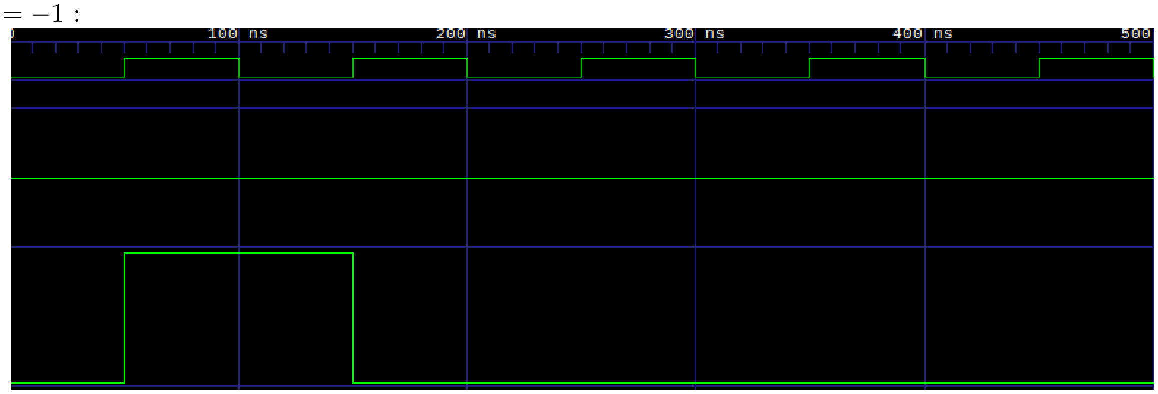
This block makes the sum of the inputs a and b.

filteringcell block

This is the core of all linear filtering structure. It's composed of one of each precedents blocks, and the value of the gain corresponds to one coefficient in a z-transform.

FIR blocks

A first-order and a second-order FIR are availables. Pure derivative is obtained for $b_0 = 1$ and $b_1 = -1$:

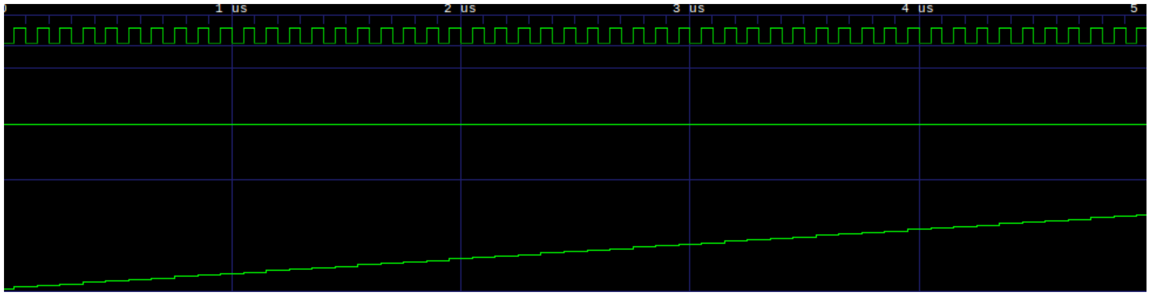


Indeed, derivative of a step is a Dirac impulse.

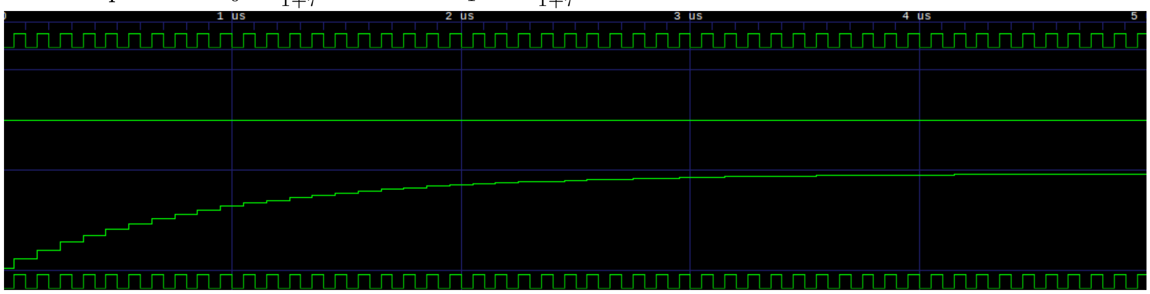
These blocks are simples chains of *filteringcell*s, whose the number corresponds to the order of the filter. I think we should propose a block with the order in parameter, but I don't know how to make it in Verilog.

IIR block

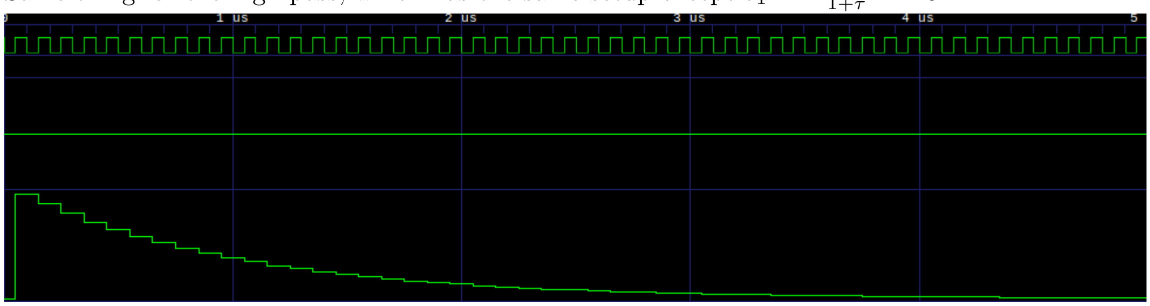
This block makes use of two FIR filters to build an IIR (two-poles, two-zeros). Let's start with the integrator ($b_0 = 1$ and $a_1 = -1$) :



The following picture is the step response of the first order we took in example. I took $\tau = 10$, which corresponds to $b_0 = \frac{1}{1+\tau} = 0.1$ and $a_1 = -\frac{\tau}{1+\tau} = -0.9$:



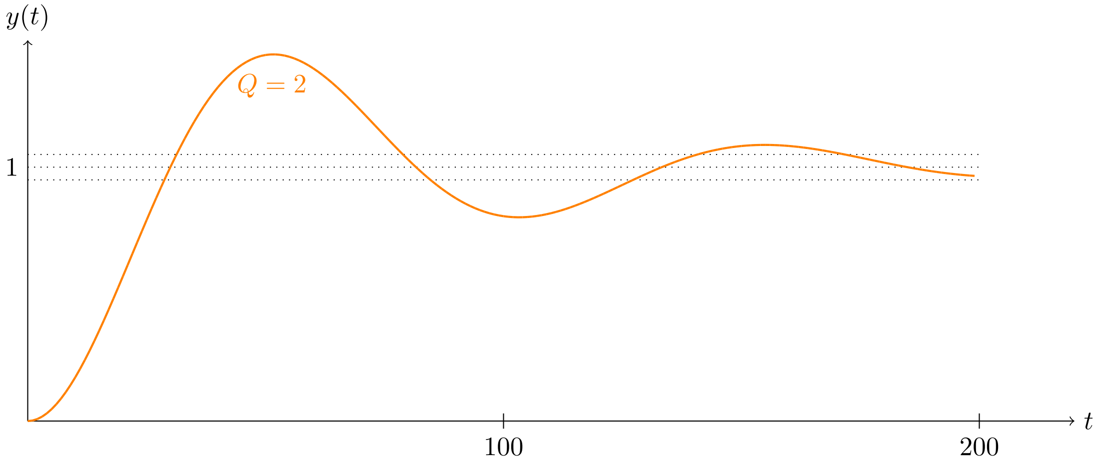
Same thing for the high-pass, which has the same setup except $b_1 = -\frac{1}{1+\tau} = -0.1$:



For another example, we'll take the following Laplace-transform (whose order is 2) :

$$\mathcal{L}(h)(p) = \frac{1}{1 + \frac{p}{Q\omega_0} + \frac{p^2}{\omega_0^2}}$$

With $Q = 2$ and $\omega_0 = \frac{2\pi}{100}$, the theorical step response is :



With the substitution $p = 1 - z^{-1}$, we'll find :

$$\mathcal{L}(h)(p) = \frac{\frac{1}{1+\frac{1}{Q\omega_0}+\frac{1}{\omega_0^2}}}{1 - \frac{\frac{1}{Q\omega_0}+\frac{2}{\omega_0^2}}{1+\frac{1}{Q\omega_0}+\frac{1}{\omega_0^2}} z^{-1} + \frac{\frac{1}{\omega_0^2}}{1+\frac{1}{Q\omega_0}+\frac{1}{\omega_0^2}} z^{-2}}$$

Which allows to calculate the coefficients :

$$\begin{cases} b_0 = 0.00381299972176139 \\ a_1 = -1.96203111287016 \\ a_2 = 0.965844112591921 \end{cases}$$