# 1. Some theory background.

## 1.1 Transfer function

$$H(z) = \left( \frac{1 - z^{-rg}}{1 - z^{-1}} \right)^m \quad [1]$$

## 1.2 Truncation

In interpolation mode truncation not used. Full output data width is:

$$dw_{out} = dw_{max} = dw + log2\left( \frac{r^m}{r} \right)$$

In decimation mode $dw_{max}$ is large for practical cases. Truncation may be used at each stage reducing register widths. See [2] for details.

# 2. Core description.

## 2.1 Source files.

- cic_package.sv — function set for CIC-decimator register's width calculation
- comb.sv — comb module
- integrator.sv — integrator module
- downsampler.sv — downsampling register module (used in CIC-decimator)
- cic_i.sv — parametrizable CIC integrator module
- cic_d.sv — parametrizable CIC decimator module
- cic_i_tb.sv, cic_d_tb.sv — testbench examples
- cic_i_tb_run.tcl, cic_d_tb_run.tcl — simulation scripts

## 2.2 Core parameters

- dw — input data width for integrator module
- idw/odw — input/output data width for integrator module
- m — CIC-filter order (m combs + m integrators)
- r — interpolation/decimation ratio
- g — differential delay in coms

# 3. Getting started (ModelSim example).

## 3.1 Setup core

mkdir ~/cic_test

cd ~/cic_test

svn co http://opencores.org/ocsvn/cic_core/cic_core

### 3.2 Create project in ModelSim.

Specify folder (cic_test), add source files from cic_core/trunk/sim and cic_core/trunk/src.

### 3.3 Run test.

In ModelSim console type «do cic_core/trunk/sim/cic_d_tb_run.tcl» (CIC-decimator example) or «do cic_core/trunk/sim/cic_i_tb_run.tcl» (CIC-interpolator).

[1] http://www.design-reuse.com/articles/10028/understanding-cascaded-integrator-comb-filters.html

[2] Hogenauer, E. B., "An Economical Class of Digital Filters for Decimation and Interpolation", IEEE® Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, No. 2, April 1981, pp. 155-162.