Cascaded Integrator Comb

SystemC Approach

Author:
Ahmed Shahein

email:
ahmed.shahein@ieee.org

June 8, 2012

1 Introduction

Cascaded Integrator Comb (CIC) filter is one of the most popular filters in literature. Its main advantages are a) no multipliers b) implemented as recursive (IIR) or non-recursive (FIR) as well. The attached code is corresponding to the non-recursive implementation which is depicted by (1) and shown in Fig. 1.

Recursive representation

$$H(z) = \left(\frac{1}{\alpha}\frac{1-z^{-\alpha}}{1-z^{-1}}\right)^N \tag{1}$$

where $\alpha = M \times D$, M decimation factor and D differential factor where $D \in \{1, 2\}$.

Non-Recursive representation

$$H(z) = \left(\frac{1}{\alpha}\sum_{i=0}^{\alpha-1} z^{-i}\right)^N \tag{2}$$

Non-Recursive representation

$$H(z) = \left(\prod_{i=0}^{\beta-1}(1+z^{-2^i})\right)^N \tag{3}$$

where $\beta = log_2(\alpha)$.

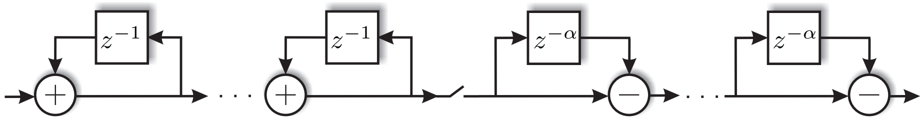

Figure 1: IIR-FIR CIC implementation.

2 Getting Stared

The SystemC model for the CIC is developed in structural hierarchal manner. In other words, each component is design individually in a header file. Further, each component has its own source file to test the corresponding block separately from the top-level entity. The top-level entity "**cicDecimator.h**" combines all the sub-blocks "**integrator.h**", "**comb.h**" and "**downsample.h**" to form a CIC decimation filter. The test-bench is given in "cicDecimator.cpp".

3 How to use the code?

You can customize the code to fits your specs. You need to enter/change the decimation factor M and the filter order or the number of stages N. You can do so by replacing M at "**downsample.h**"

Input	1st Integrator Output	2nd Integrator Output	3rd Integrator Output	1st Comb Output	2nd Comb Output	3rd Comb Output
0	0	0	0	0	0	0
1	1	1	1	1	1	1
1	2	3	4	3	2	1
1	3	6	10	6	3	1
1	4	10	20	10	4	1
1	5	15	35	15	5	1
1	6	21	56	21	6	1
1	7	28	84	28	7	1

Figure 2: 3 stages CIC ($N = 3, M = 1$) estimated output stage by stage.

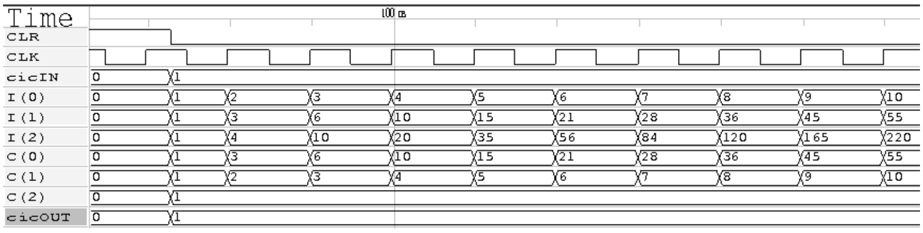

Figure 3: CIC simulated ($N = 3, M = 1$) output stage by stage.

Input	1st Integrator Output	2nd Integrator Output	3rd Integrator Output	M	1st Comb Output	2nd Comb Output	3rd Comb Output
0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0
1	2	3	4	4	4	4	4
1	3	6	10	4	4	4	4
1	4	10	20	20	16	12	8
1	5	15	35	20	16	12	8
1	6	21	56	56	36	20	8
1	7	28	84	56	36	20	8
1	8	36	120	120	64	28	8

Figure 4: CIC decimator ($N = 3, M = 2$) estimated output stage by stage.

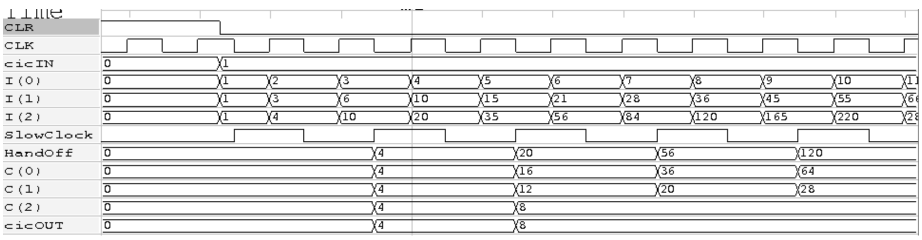

Figure 5: CIC decimator ($N = 3, M = 2$) simulated output stage by stage.

```
#define M 2
```

Then at "**cicDecimator.h**" replace the N

```
#define N 3
```

Finally, you can change the clock frequency and enter your stimuli at the "**cicDecimator.cpp**"

```
sc_clock  CLK("CLK", 10, SC_NS);
.
.
```

```
.
cicIN = 0;
```