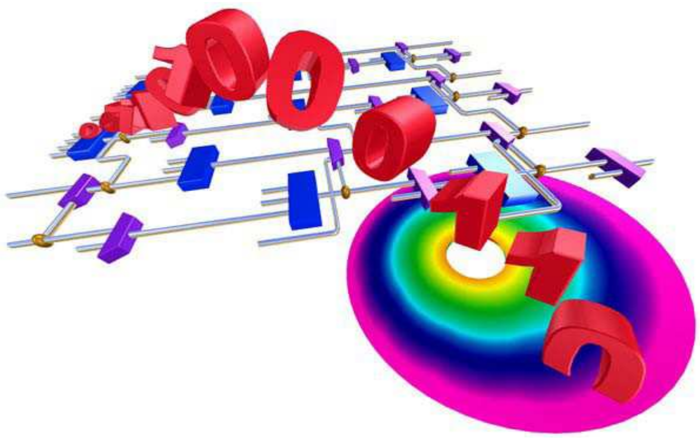
Implementation of Filters



Implementation of a Digital Receiver with FPGAs
University of Campinas

Peter Thorwartl

Introduction

- In this section we will review some filter types that are particularly relevant and efficient for FPGA implementation.
  - *Transpose Structures*
  - *Moving Average*
  - *CIC Filters*
  - *Differentiators and Integrator*
  - *Oversampled* (with sigma delta inputs)
  - *"Efficient" Digital FIRs*: Halfband, Symmetric
  - *Decimation and Interpolation Filters*
- Most of these filters are low cost to implement as they have no (costly) multiplications.

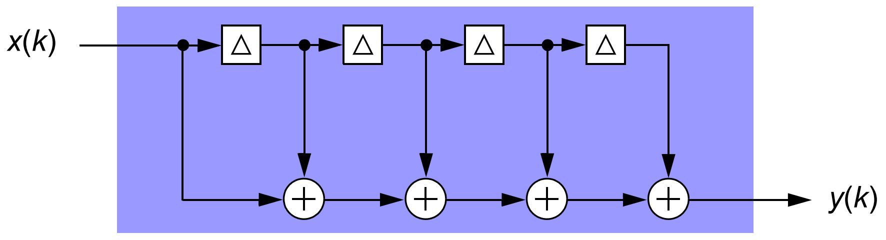
Notes:

In this section we will review the characteristics of various low cost FIR filters in the time and frequency domain.

These filters are low cost by virtue of have simple weights of 1, 0, or -1. Multiplying by these values of done at the cost of just a few gates rather than a full multiplier.

Moving Average

- All weights set to one:

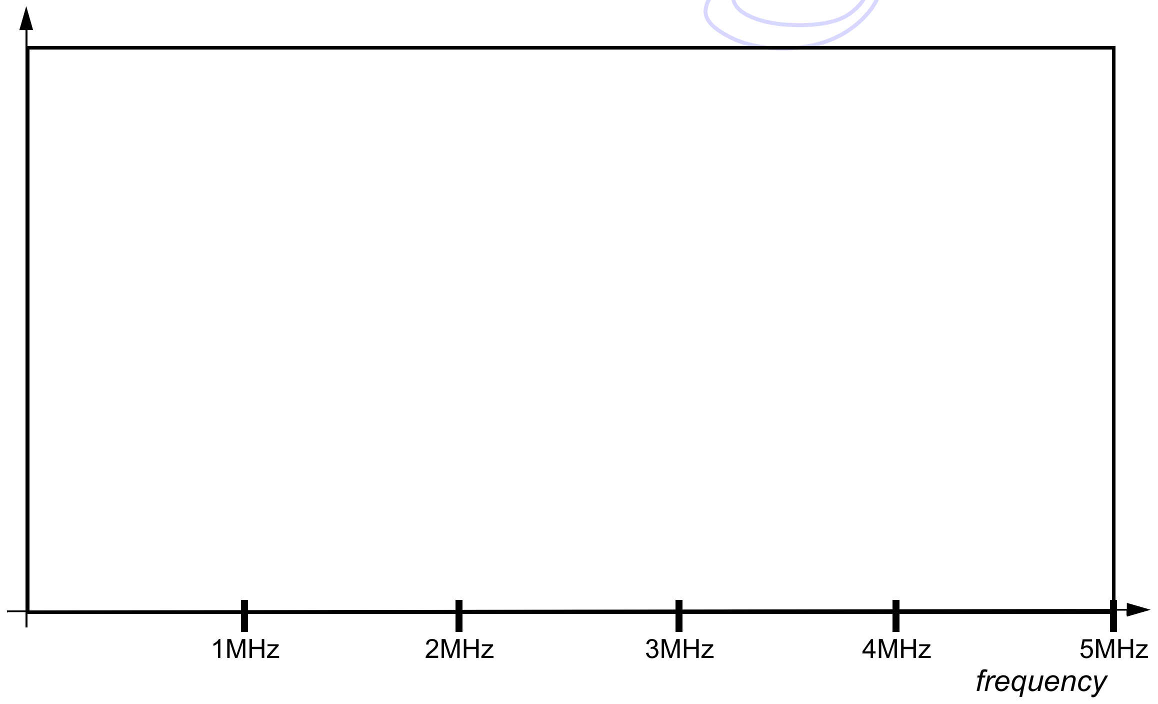


- Simple “low pass” characteristic
- Low cost - no multiplies required.

This filter might preferably be implemented use a power of two number of weights - why?

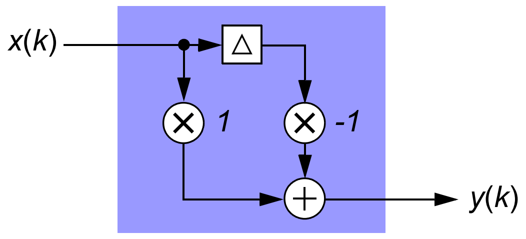
Notes:
This filter is a very simple low pass characteristic.

SystemView Examples....



Differentiator

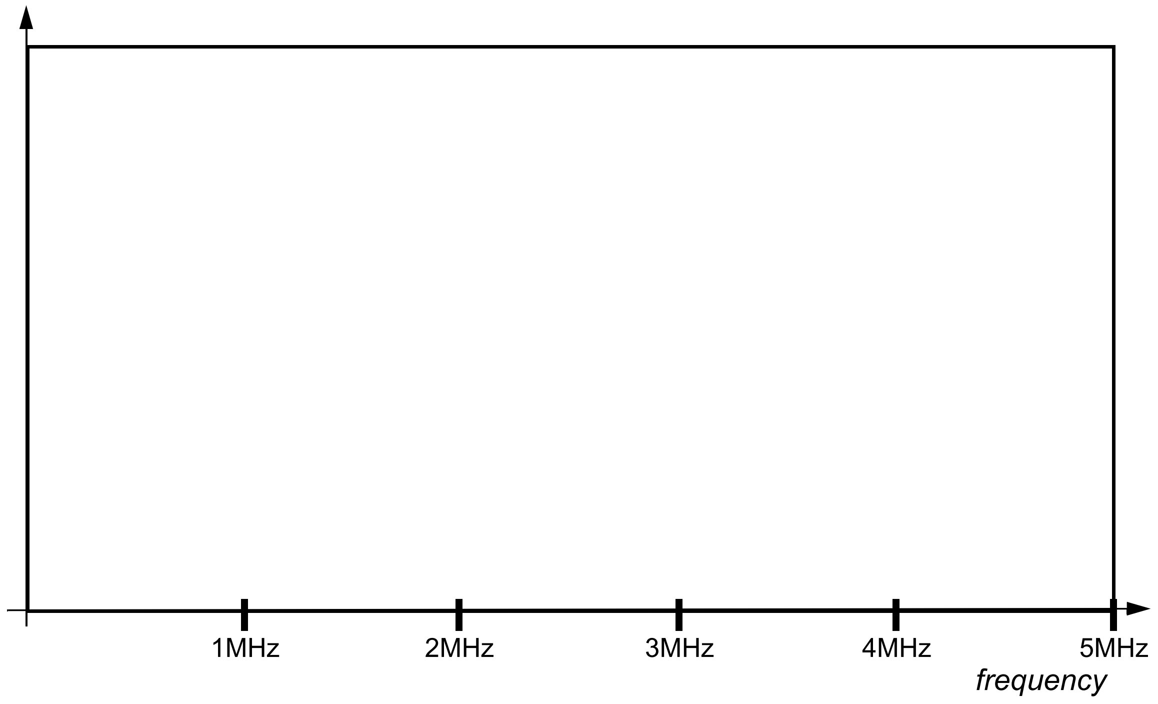
- Two weight filter, with values of 1 and -1:



- Simple “high pass” characteristic
- Low cost - no multiplies required.

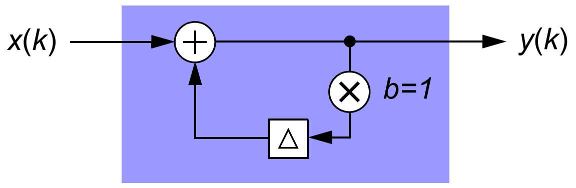
Notes:

SystemView Examples....



Integrator

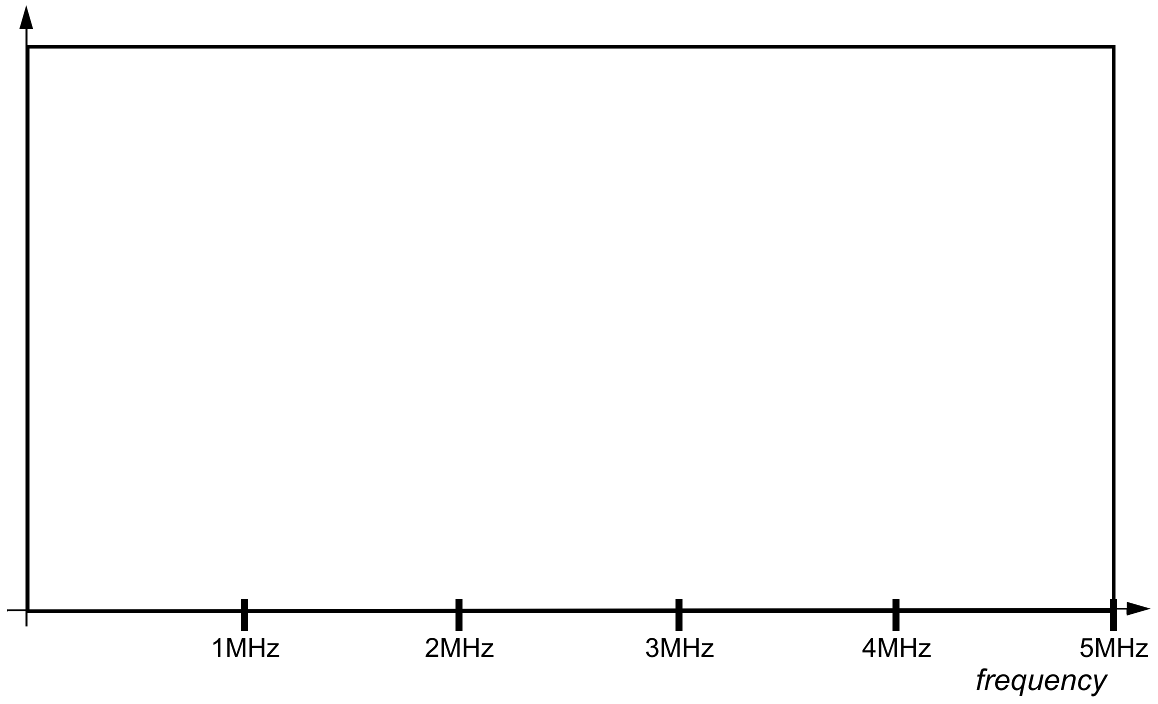
- Single weight IIR filter:



- “Low pass” characteristic (infinite gain at DC)
- Low cost - no multiplies required
- Set *b* just less than 1 produces a leaky integrator.
  .

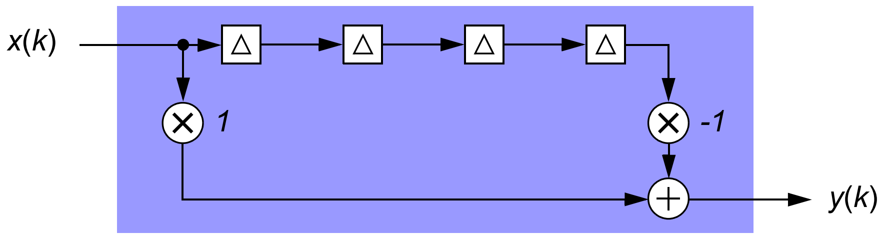
Notes:

SystemView Examples....



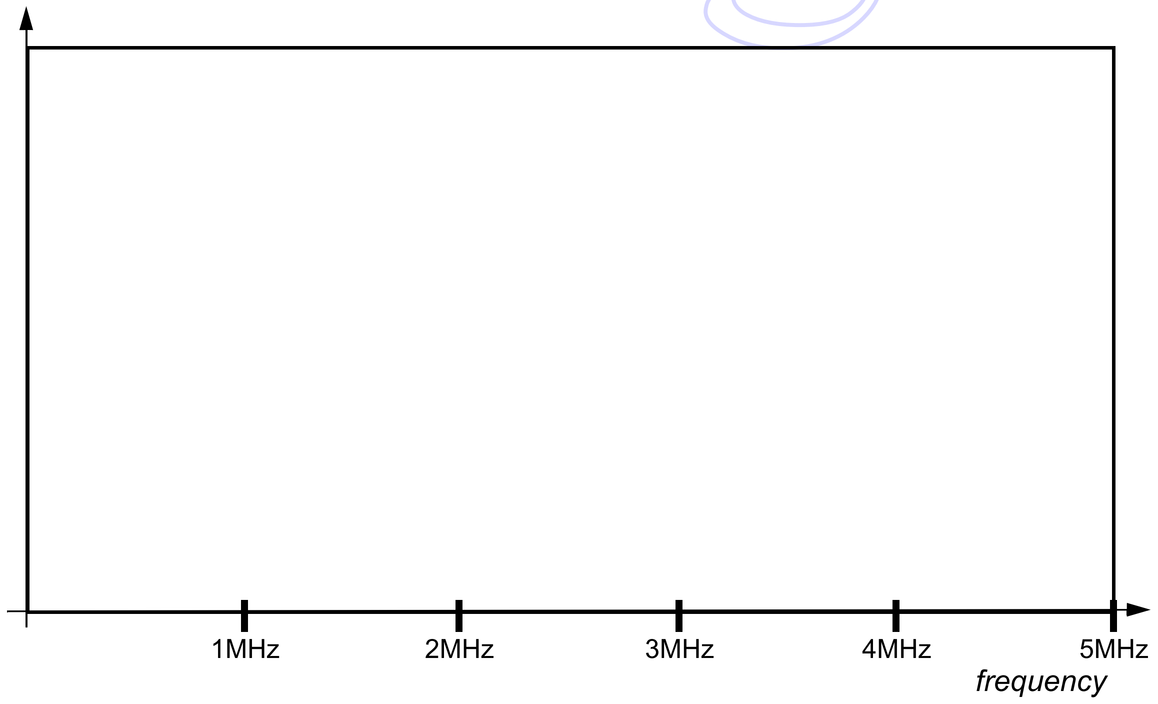
Comb Filter

- Weights set to 1 and -1 at either end of the filter.



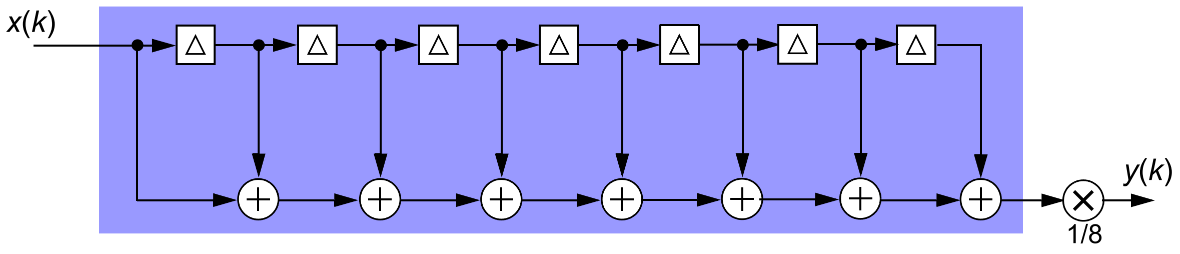
- Simple multichannel (comb like) frequency response
- Low cost - no multiplies required.

Notes:



Eight Weight Moving Average

- Consider again the moving average (MA); all weights of "1"

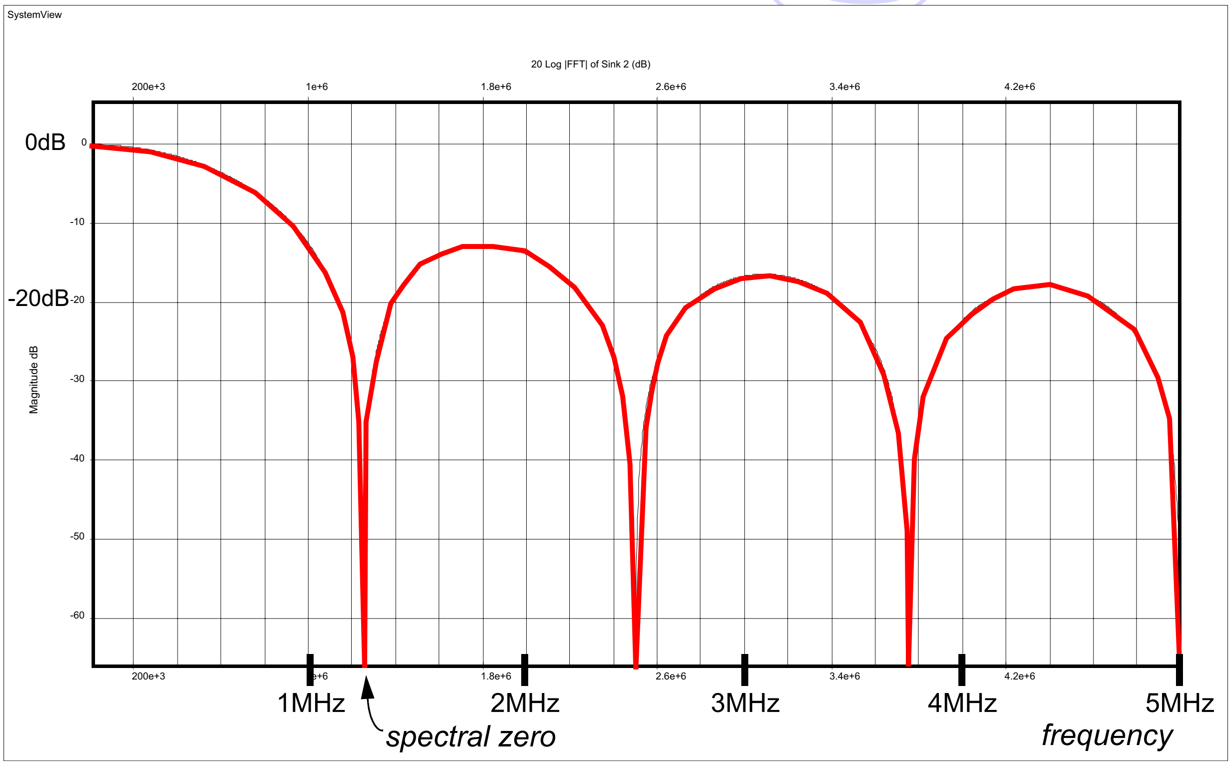


$$(H(z) = 1 + z^{-1} + z^{-2} + z^{-3} + z^{-4} + z^{-5} + z^{-6} + z^{-7})\frac{1}{8}$$

- True moving average if we scale the output by $\frac{1}{8}$ (left shift 3 places)- equivalent to all weights being 1/8.
- In the spectrum the moving average filter has *N-1* spectral zeroes from 0 to f_s. In our case $N = 8$, we can see 4 spectral zeroes from 0 to $f_s/2$.

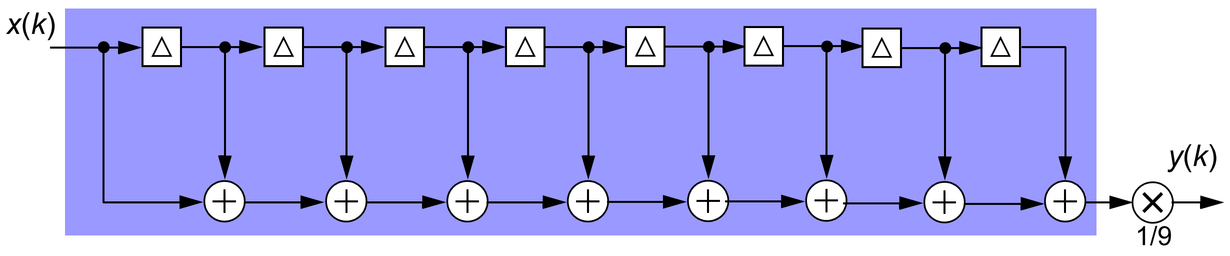
Notes:
For ease of numerical representation, we choose $f_s = 10,000,000$

We can see four spectral zeroes between 0 and $f_s/2$, i.e. 8-1=7 spectral zeroes between 0 and f_s.



Nine Weight Moving Average (MA)

- All weights are “1”



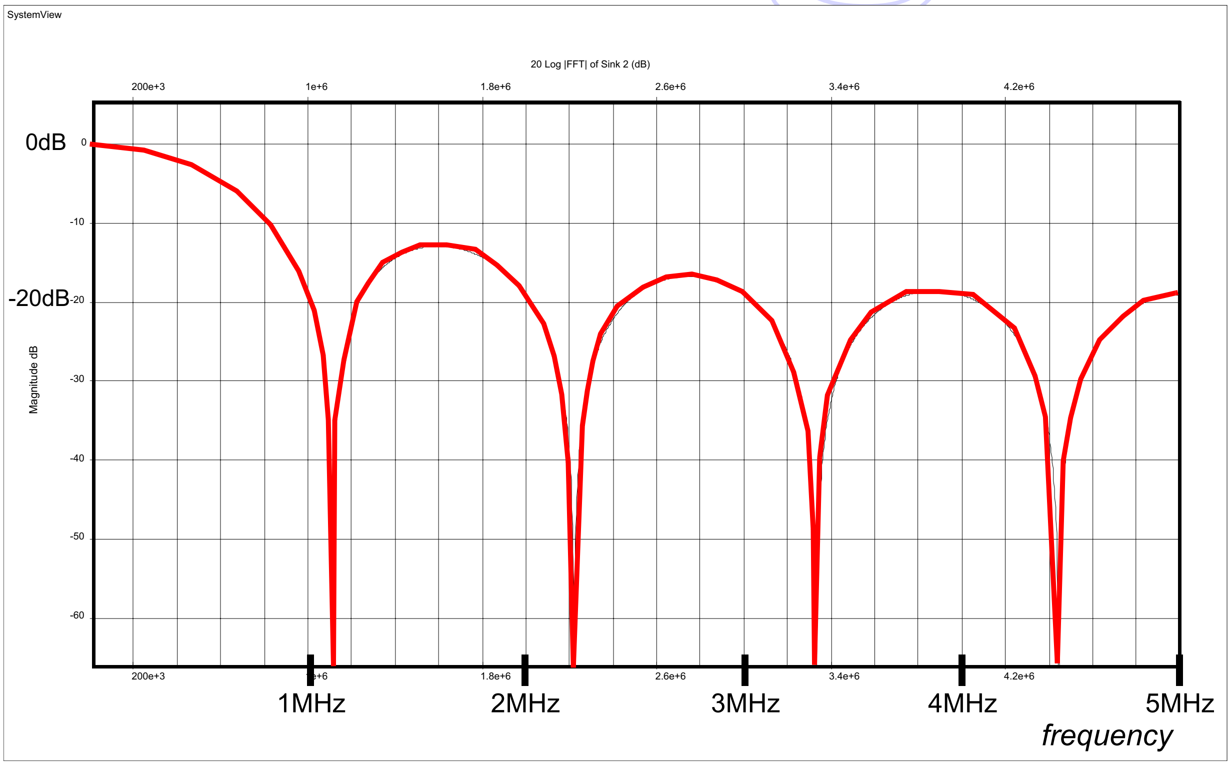
$$H(z) = (1 + z^{-1} + z^{-2} + z^{-3} + z^{-4} + z^{-5} + z^{-6} + z^{-7} + z^{-8})\frac{1}{9}$$

- Multiplying by 1/9 is not so convenient......

Notes:

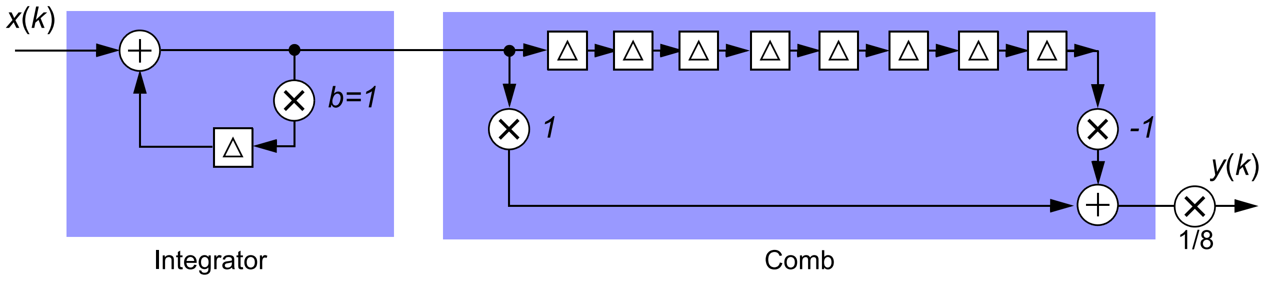
For ease of numerical representation, we choose $f_s = 10,000,000$

We can see 4 spectral zeroes between 0 and $f_s/2$, i.e. 9-1=8 spectral zeroes between 0 and f_s.



Cascade Integrator Comb (CIC)

- Generate a MA impulse response with CIC structure (see Slide 9.6)



$$H(z) = \left(\frac{1}{1-z^{-1}}\right)(1-z^{-8}) = \frac{1-z^{-8}}{1-z^{-1}}$$

- Note that: $\frac{1-z^{-8}}{1-z^{-1}} = 1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5}+z^{-6}+z^{-7}$

- i.e ***an integrator and M comb weight CIC* = *M-1 weight MA***

August 2005, For Academic Use Only, All Rights Reserved

Notes:

$$\frac{1-z^{-8}}{1-z^{-1}} = 1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5}+z^{-6}+z^{-7}$$

$$1-z^{-8} = (1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5}+z^{-6}+z^{-7})(1-z^{-1})$$

$$\begin{aligned}1-z^{-8} = 1 &+ \cancel{z^{-1}} + \cancel{z^{-2}} + \cancel{z^{-3}} + \cancel{z^{-4}} + \cancel{z^{-5}} + \cancel{z^{-6}} + \cancel{z^{-7}} \\ &- \cancel{z^{-1}} - \cancel{z^{-2}} - \cancel{z^{-3}} - \cancel{z^{-4}} - \cancel{z^{-5}} - \cancel{z^{-6}} - \cancel{z^{-7}} - z^{-8}\end{aligned}$$

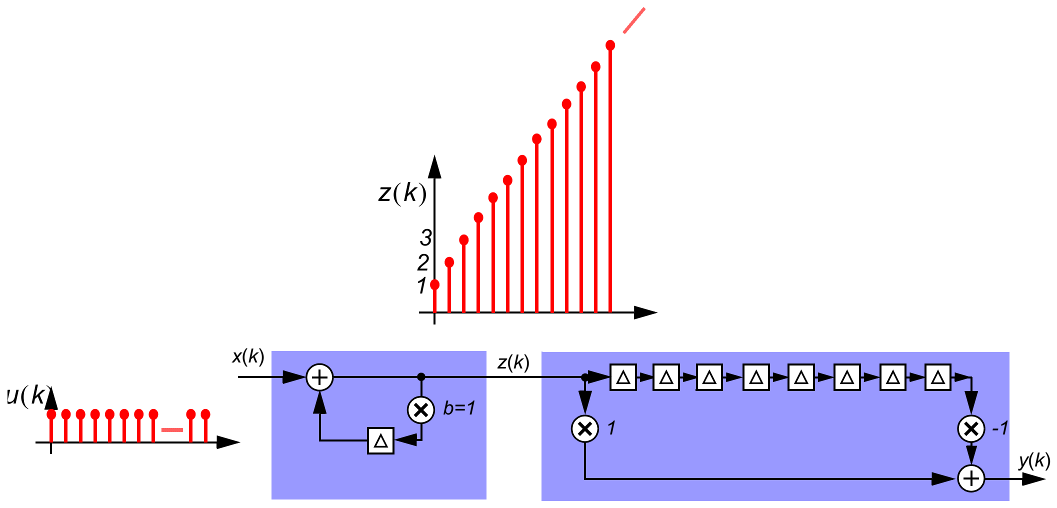
It is interesting to note that the integrator has infinite gain at DC and the comb filter has zero gain an DC!

CIC Advantages: CIC has Only two additions compare to 8 additions in MA.

CIC Disadvantages: CIC requires 9 storage registers, and MA requires only 7 storage register.

Integrator Overflow

- The integrator of the CIC has infinite gain at DC (0 Hz).
- Therefore consider the input of a ***step signal*** to the CIC:



- The integrator output "grows" unbounded for the step input.

Notes:

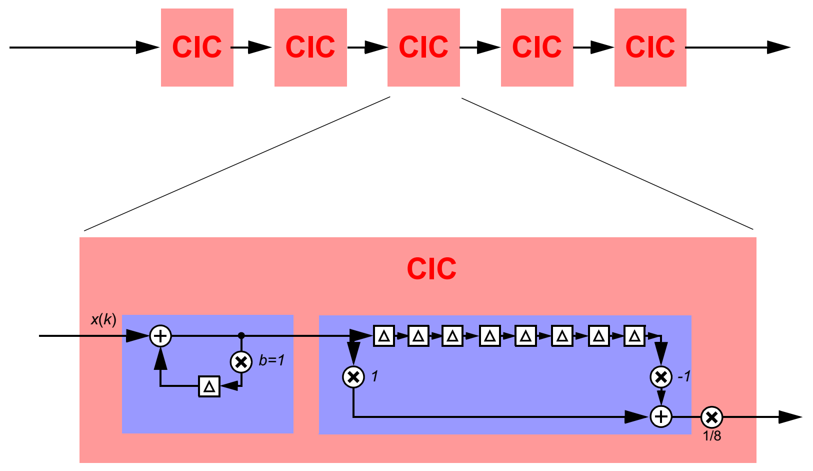
Eventually the integrator output will overflow.....

To address this we can use modulo arithmetic.

See ***SystemView examples***.....

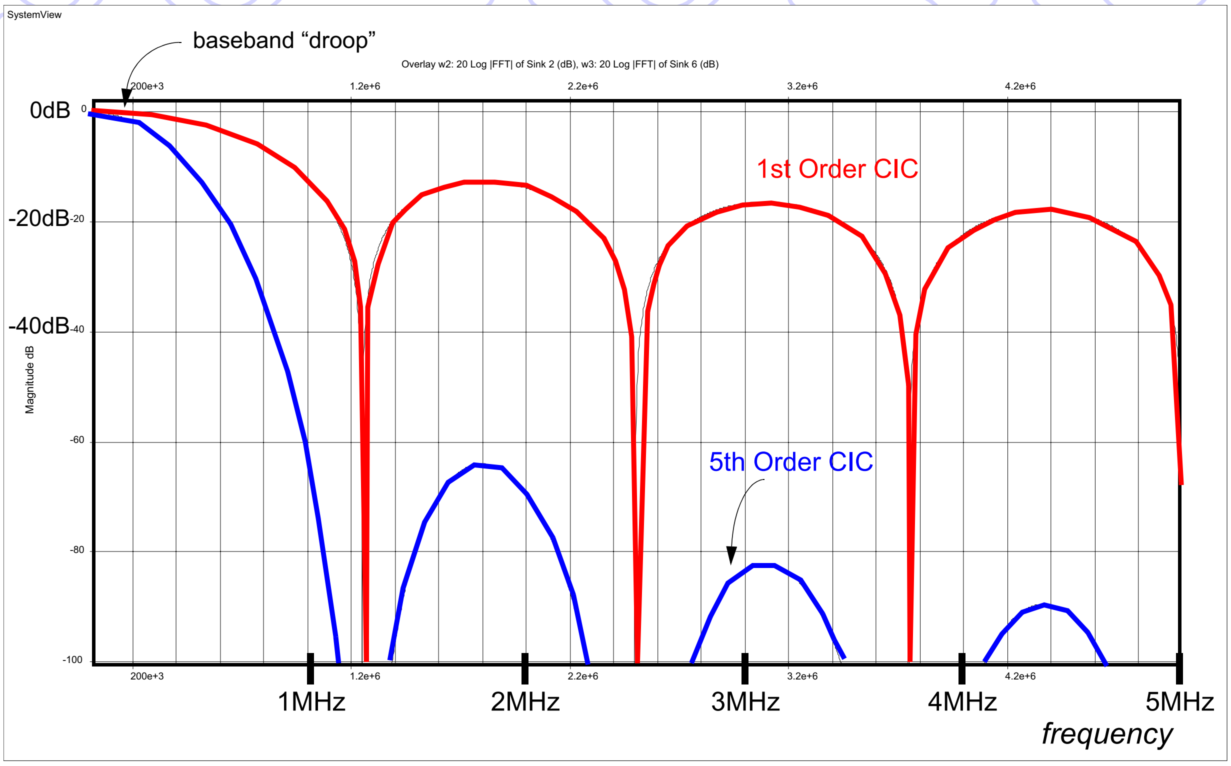
Cascade of CICs

- We can cascade CIC filters to produce "better" low pass characteristics:
- Cascade of 5 CICs of 8th order MA filters:



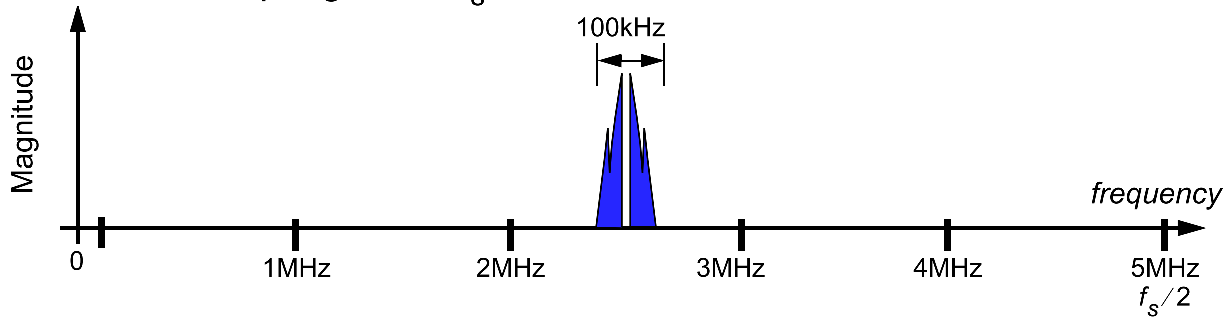
- Note however the baseband droop is "worse".

Notes:
Plots of CIC and cascade of 5 CICs for 8th order moving average.



Recovery of an IF modulated Signal

- Consider the following scenario:
    - Signal of interest centered at $f_c = 2.5\text{MHz}$
    - Signal bandwidth = 100kHz
    - Sampling rate, $f_s = 10\text{MHz}$

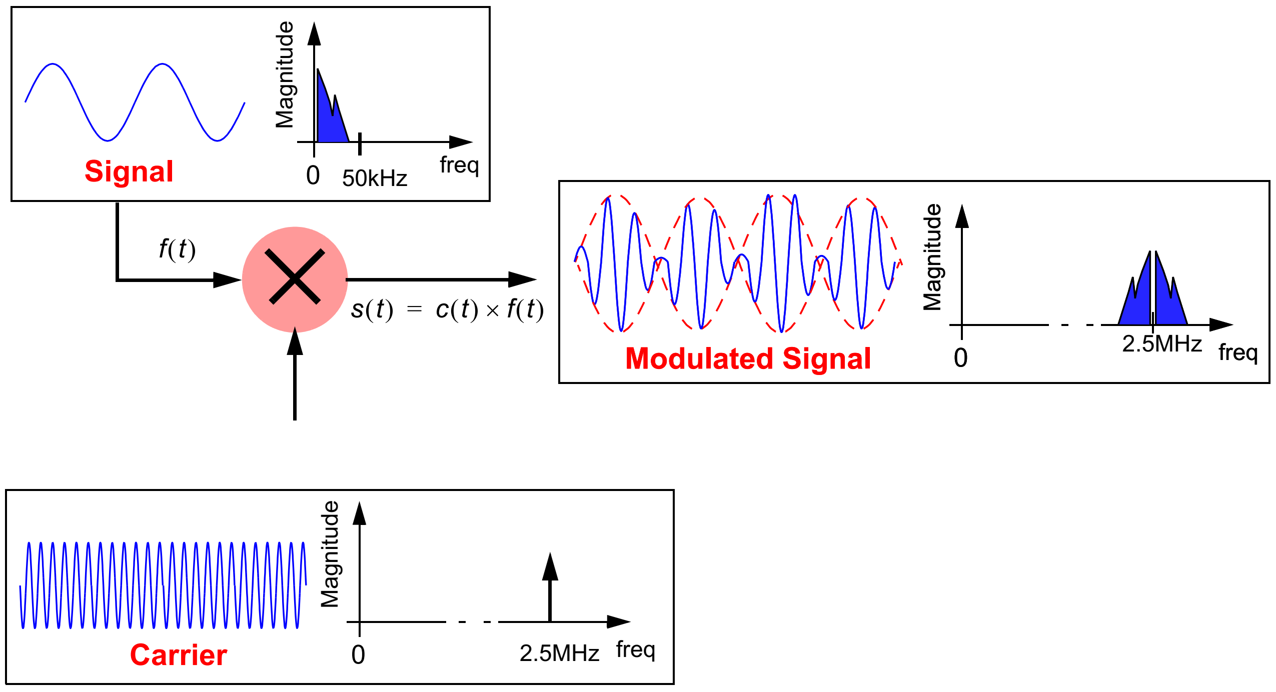


- Requirement is to recover the IF signal at baseband frequencies using as low computation as possible.

Notes:

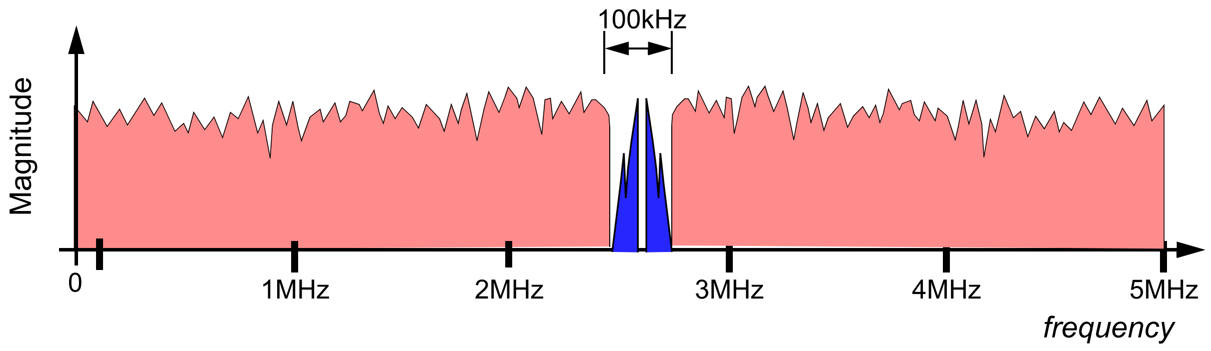
This bandpass signal has been created by simple amplitude modulation:

Amplitude of a “high” frequency carrier sinusoid is varied in proportion to the amplitude of signal with lower frequency components.



Recovery of an IF modulated Signal

- When the signal is received, the spectrum outside of the 50kHz band of interest is likely to be occupied with other signals and noise:



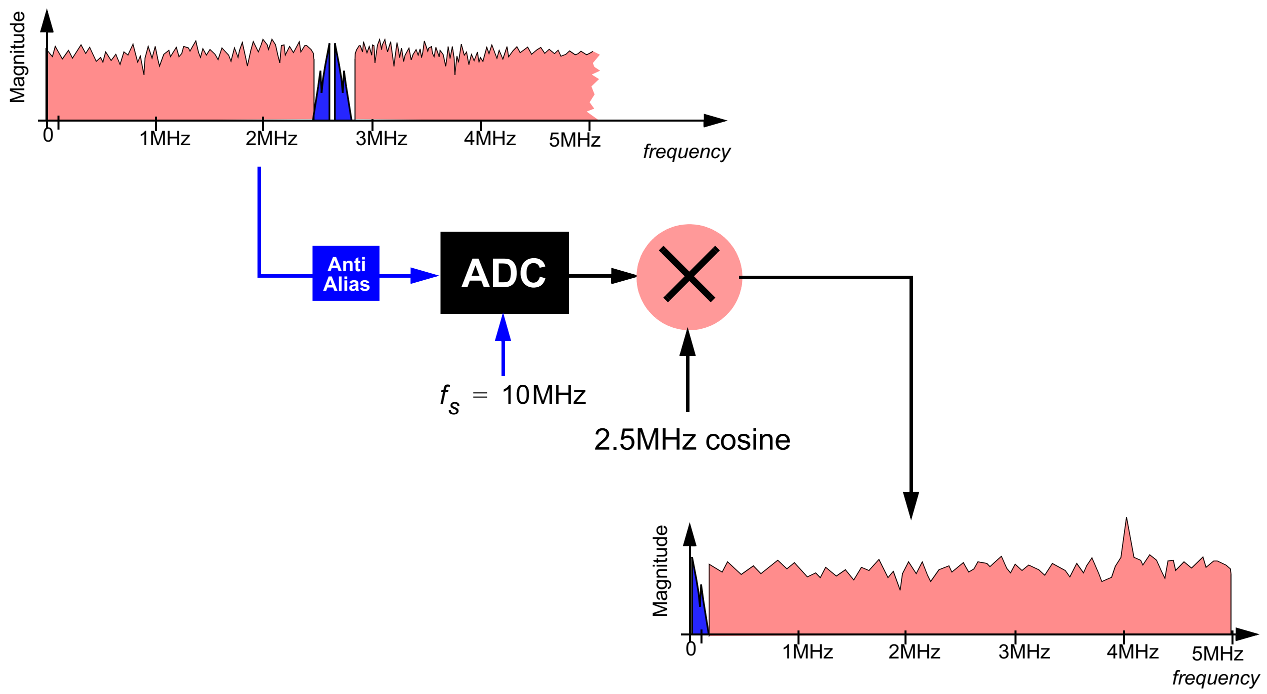
- To recover we require to demodulate to baseband and then low pass filter to recover the signal.

Notes:

Top

Demodulation of Signal

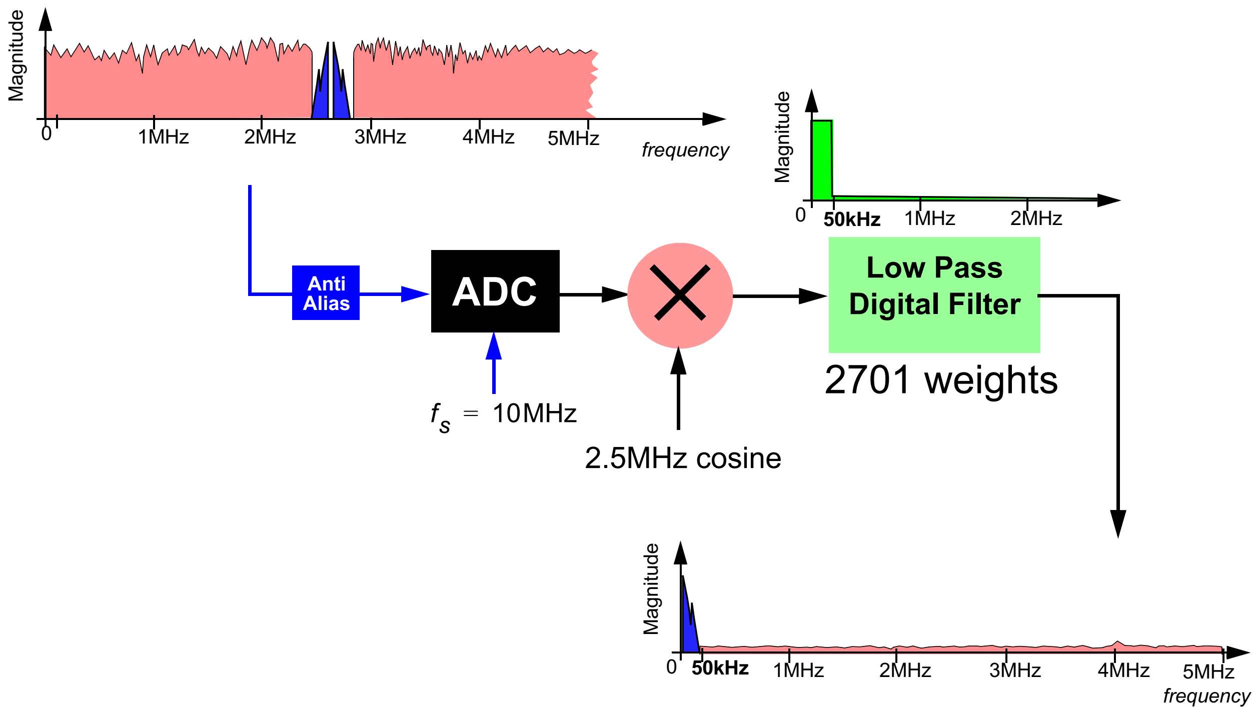
- Sampling with a high frequency ADC we can first digitally demodulate the signal:



Notes:

Demodulation of Signal

- ....then low pass filter:

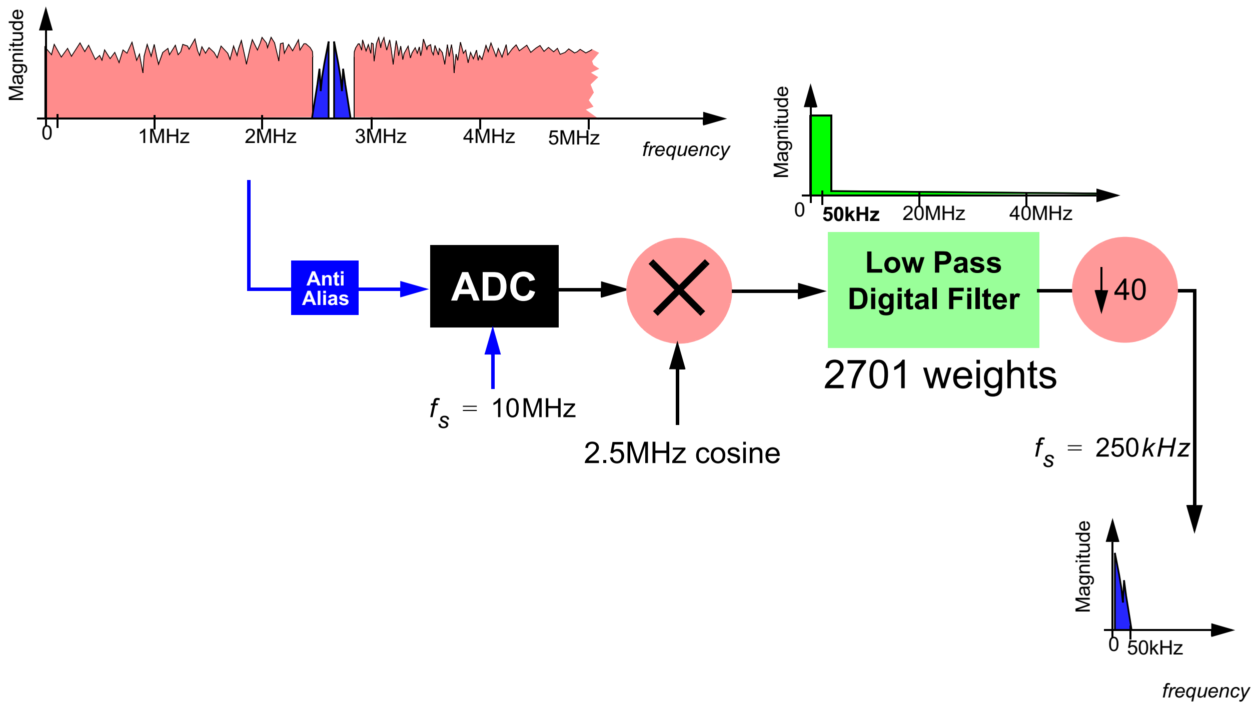


Notes:
Cost of Digital Filter

MACs/sec = 10,000,000 x 2701 = 27,010,000,000 = ***27 billion MACs/sec***!

But remember the Downsampling...!

- ....then low pass filter:



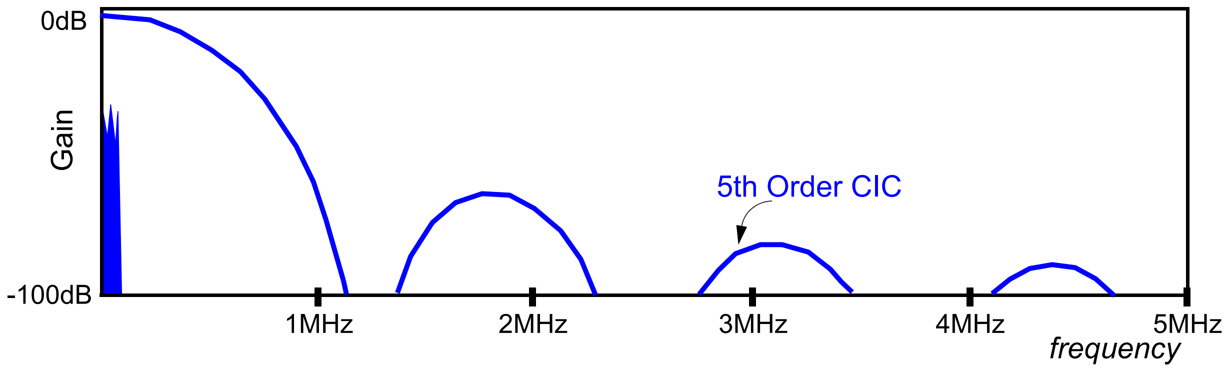
Notes:
In this example the final required sample rate is 250kHz and hence as we have bandlimited we can now downsample by a factor of 40.

Cost of Digital Filter

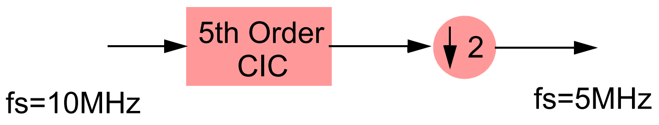
MACs/sec = 10,000,000/40 x 2701 = 270,100,000 = ***675 million MACs/sec***!

CIC stage for Decimation

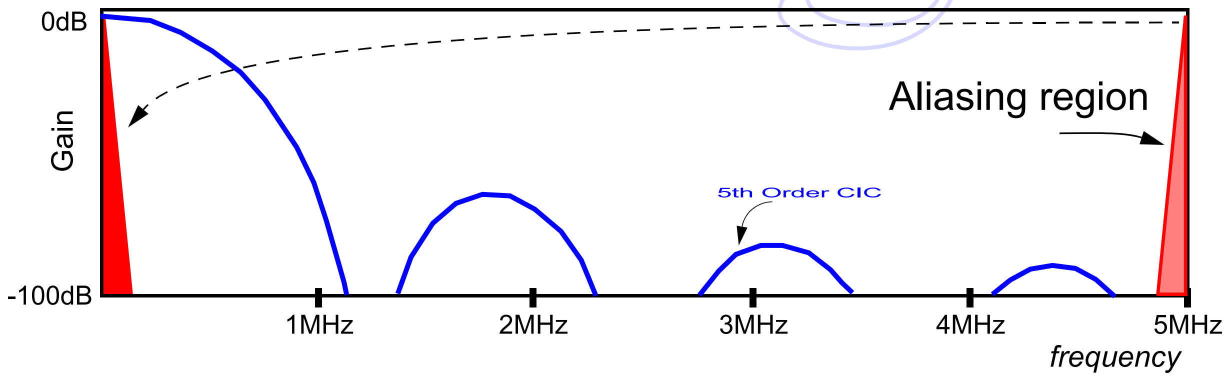
- Consider now designing the low pass filter using a cascade of low cost simpler filters. Is there a cost saving?



- If we low pass filter the signal of interest with the 5th order CIC then downsample by 2 to 5MHz, then the aliasing of higher frequency signals comes from frequency regions where the energy is very low.

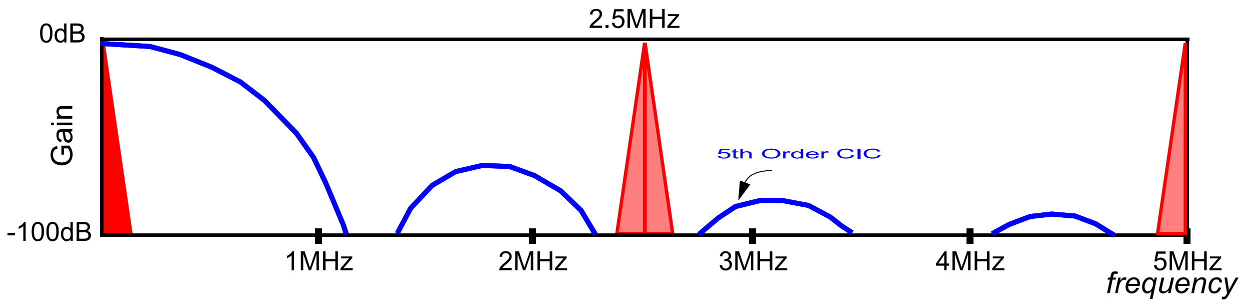


Notes:

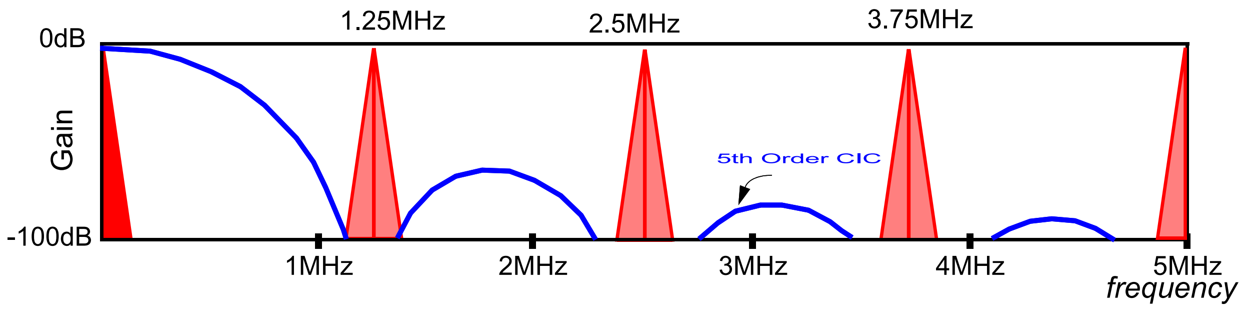


Final Stage Decimation

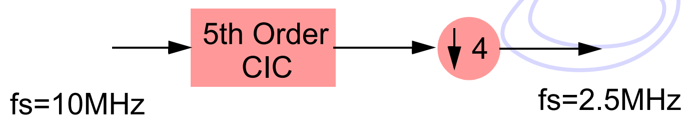
- Noting the other *empty* spectral regions we could downsample by 4:

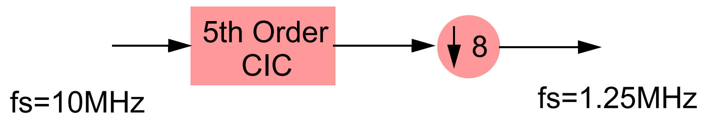


....in fact we could probably downsample by 8:



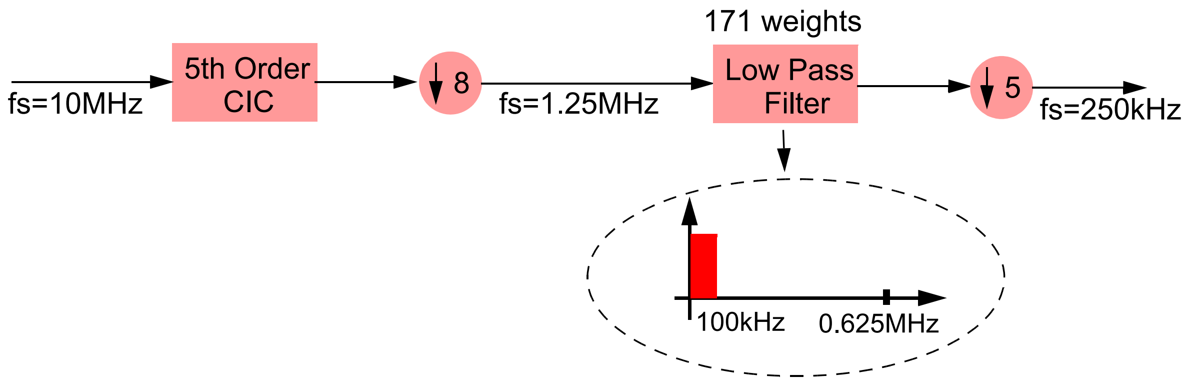
Notes:



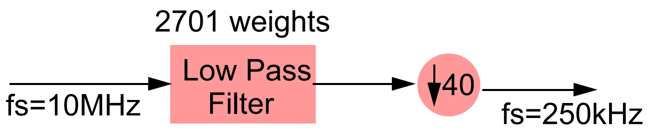


Downsampling Values

- We can then perform a final stage of decimation using a standard low pass filter:



- Therefore we are now *anticipating* that the above staged decimation is similar to the one step decimation presented earlier (and shown below):



Notes:

Cost Comparison

- One stage ***Low Pass Filter*** decimation:
  - 2701 weights, 10MHz sampling, Downsample 40

$$\frac{2701 \times 10,000,000}{40} = 675.25 \text{ million MACs/sec}$$

- ***5th Order CIC and low pass*** (at f_s = 1.25MHz)
  - 171 weights, 1.25MHz sampling, Downsample 5

$$\frac{171 \times 1,250,000}{5} = 42.75 \text{ million MACs/sec}$$

  - 5 CICs with 2 adds each at 10MHz = 100 million adds/sec
- ***Computation reduced by a factor of almost 16!***

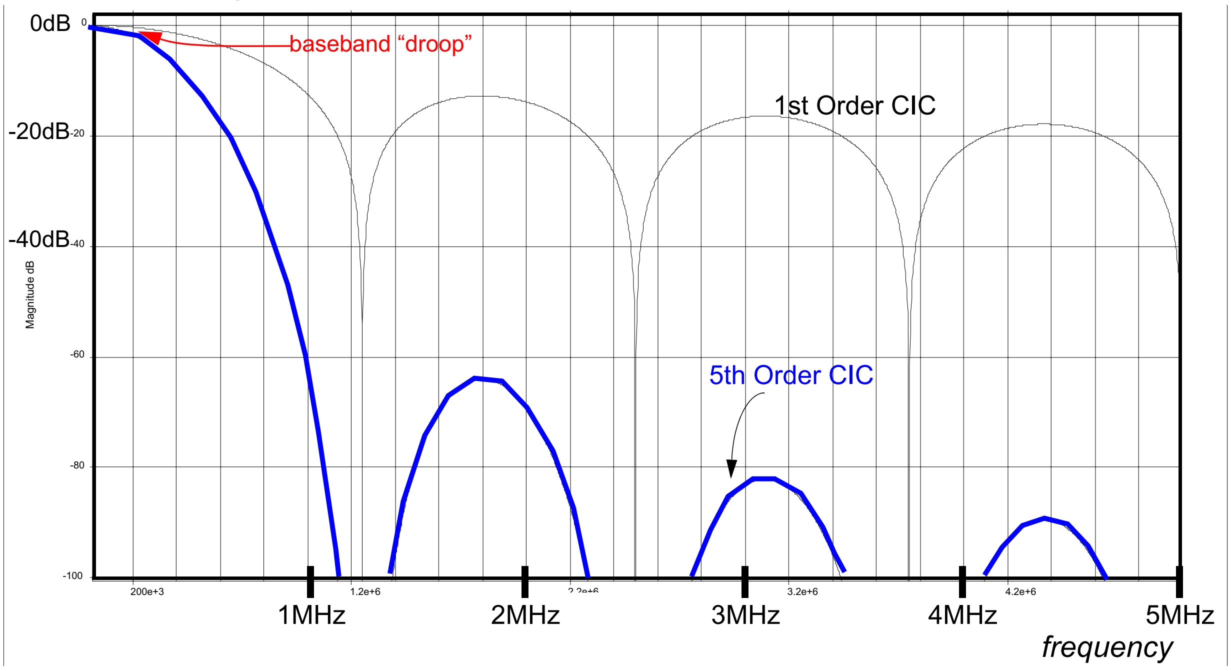
Notes:

Top

9.20

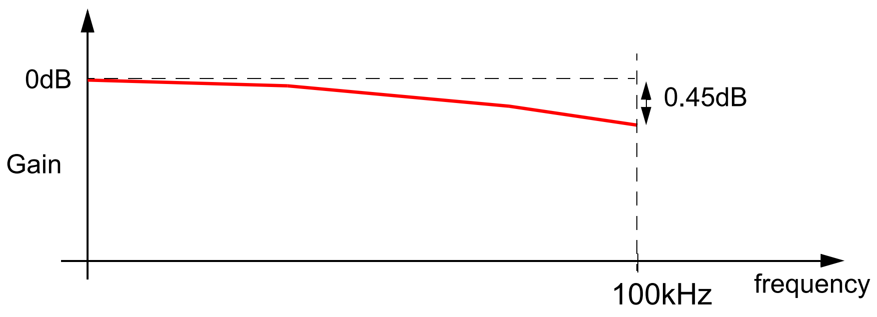
CIC Droop

- One difference have ignored so far is the “droop” at low frequencies of the CIC low pass filter:



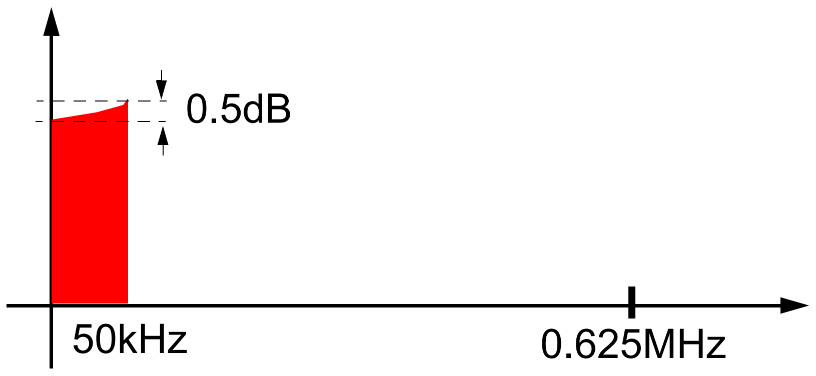
Notes:

Careful viewing in SystemView shows that the droop is around 0.5dB:



Correcting the Droop

- So how do we correct the droop?
- Incorporate a “lift” in passband of the final stage low pass filter:



- ....therefore the decision of this final stage filter must be done very carefully to correct for the droop.

Notes:

CIC Filter study

- In the next few slides the Cascaded Integrator-Comb filter is examined in more detail. This will cover the following areas:
  - Introduction to the CIC filter and some examples of where it may be used
  - An examination of word length growth in CIC filters and how ‘bit-pruning’ may be used to reduce resource consumption
  - The Sharpened CIC (SCIC) filter structure: how it differs from the CIC filter and where its use is appropriate
  - A discussion of the costs and benefits of CIC and SCIC filters compared with non-recursive, ‘moving average’-based filter structures

Notes:

Some of material in this section has been gathered from the following sources:

[1] E. B. Hogenauer, “An Economical Class of Digital Filters for Interpolation and Decimation,” *IEEE Trans. Acoustics, Speech and Signal Processing* vol. ASSP-29, no. 2, pp. 155-162, April 1981.

[2] Chapter 11, “CIC Filters” of “Multirate Signal Processing for Communication Systems” by fred harris, ISBN: 0131465112.

[3] A. Y. Kwentus, Z. Jiang and A. N Willson, Jr, “Application of Filter sharpening to Cascaded Integrator-Comb Decimation Filters,” *IEEE Trans. Signal Processing*, vol. 45, no. 2, pp. 457-467, February 1997

[4] H. Oh, S. Kim and G. Choi, “On the Use of Interpolated Second-Order Polynomials for Efficient Filter Design in Programmable Downconversion,” *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 551-560, April 1999

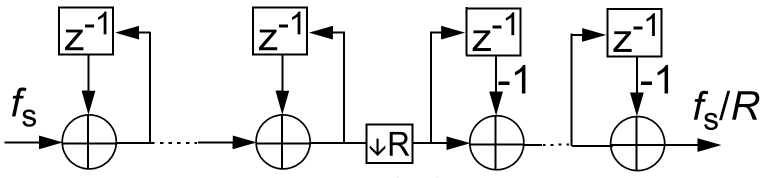
[5] J. F. Kaiser and R. W. Hamming, “Sharpening the Response of a Symmetric Nonrecursive Filter by Multiple Use of the Same Filter,” *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-25, no. 5, pp. 415-422, October 1977.

[6] Y. Gao, L Jia, J. Isoaho and H. Tenhunen, “A Comparison Design of COmb Decimators for Delta-Sigma Analog-Digital Converters”, *Analog Integrated Circuits and Signal Processing*, vol. 22, no. 1 pp. 51-60, January 2000

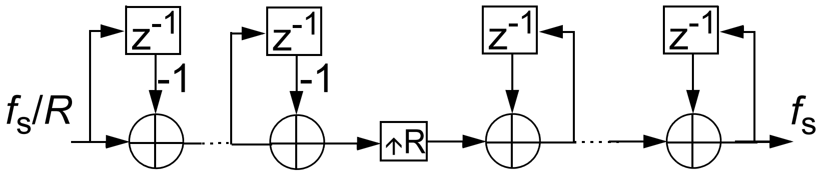
Reference is made to these texts throughout this section where appropriate.

CIC Filter Detailing Study - Introduction

- The Cascaded Integrator-Comb (CIC) filter is a multirate filter consisting of *N* integrator sections running at high sample rate, f_s, and *N* comb sections, which run at a rate of f_s/R
- Inexpensive: requires no multipliers and only minimum storage
- The filter may be configured as an anti-aliasing decimating filter, or as an anti-imaging interpolation filter:



Decimating CIC Filter



Interpolation CIC Filter

Notes:

CIC filters use a property of multirate systems known as the noble identities to allow the efficient positioning of the up/downsampler:

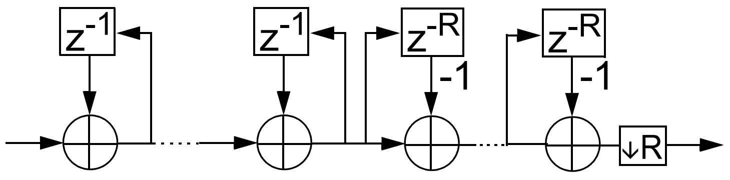
$$X(z) \rightarrow \boxed{\downarrow R} \rightarrow \boxed{H(z)} \rightarrow Y(z) \equiv X(z) \rightarrow \boxed{H(z^R)} \rightarrow \boxed{\downarrow R} \rightarrow Y(z)$$

1st noble identity (sample rate reduction)

$$X(z) \rightarrow \boxed{\uparrow R} \rightarrow \boxed{H(z^R)} \rightarrow Y(z) \equiv X(z) \rightarrow \boxed{H(z)} \rightarrow \boxed{\uparrow R} \rightarrow Y(z)$$

2nd noble identity (sample rate increase)

This allows the sample rate changer to be located between the integrators and combs, thus allowing the structure to be independent of decimation ratio. However, for analysis purposes, it is easier to examine the filter with all sections referenced to the high sample rate. Thus, for the decimating case, we have the following structure:



Decimating CIC Structure for analysis purposes only

Introduction - CIC Transfer Function

- Mathematically, the CIC filter is equivalent to a cascade of N 'moving average' filters, which means that, despite the fact that it contains recursive integrator sections, it is still an FIR filter

- Considering the case where N=1:

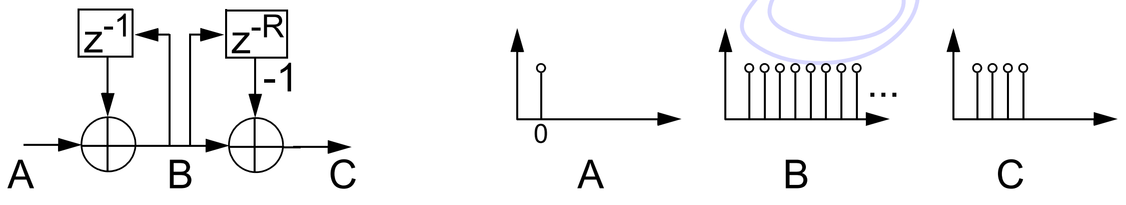
$$H_{MA}(z) = \frac{1}{R}\sum_{k=0}^{R-1} z^{-k}$$

$$H_{CIC}(z) = \frac{1}{R}\left(\frac{1-z^{-R}}{1-z^{-1}}\right)$$

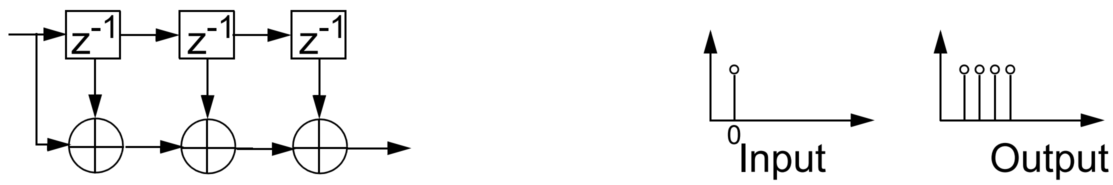
- H_{MA} and H_{CIC} are actually equivalent. This can be demonstrated by examining the impulse response of the two filters

Notes:

Consider the following CIC and moving average filters for the case of N=1, R=4 (the sample rate changer has been omitted as analysis is facilitated by referencing all components to the high sample rate):



CIC Filter and responses



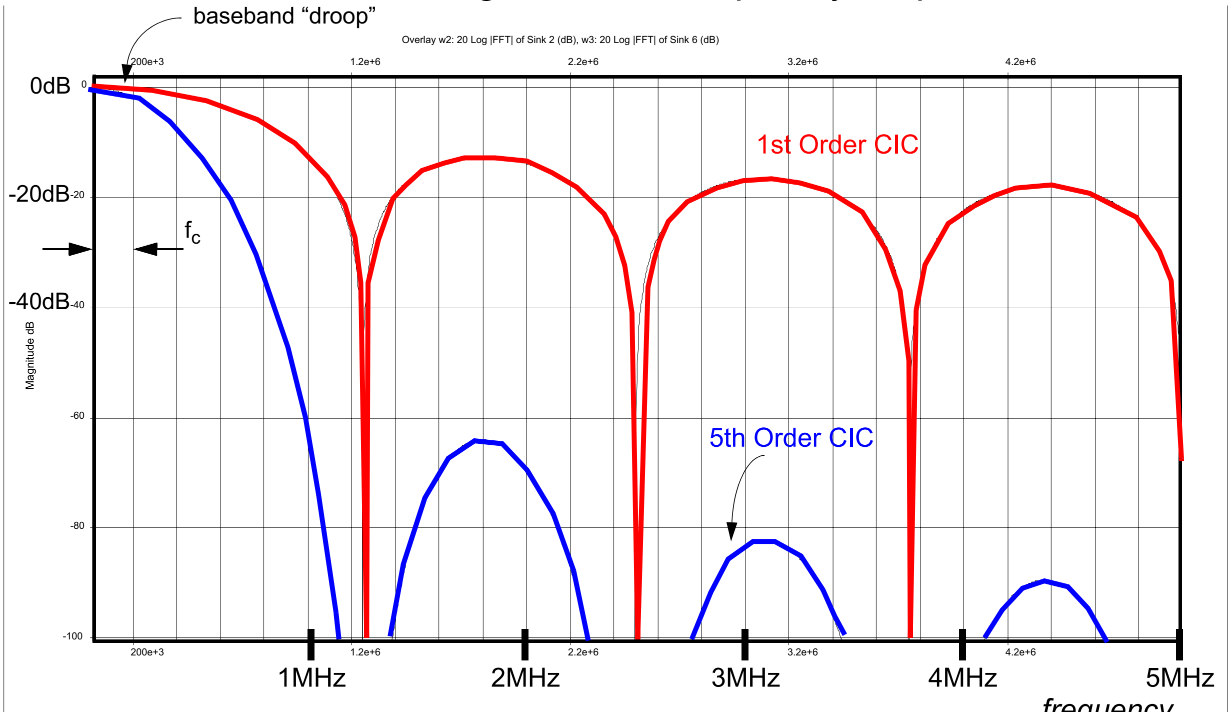
Moving average filter and responses

The moving average and CIC filters produce the same impulse response, but it can be seen that they are different internally. The first R samples emerging from the last register in the comb section are zero, each of which, when subtracted from the constant values from the integrator, results in the first four samples shown in the diagram at point C. Beyond this point in time, internally, the comb filter is still subtracting by constantly cancelling the outputs at B, the result being that all further outputs at C are zero

In contrast, in the moving average filter, no further internal processing is carried out after the output of the last non-zero sample.

Introduction - Frequency Response

- Features of the Decimating CIC filter frequency response



Notes:

The portion of the frequency response, f_c, represents the passband of the complete multirate rate filter (CIC filter followed by subsequent filtering to further isolate the passband). Typically, f_c is equal to $f_s/8R$ or $f_s/16R$.

It can be seen that there are nulls in the frequency response. These occur at integer multiples of f_s/R. The nulls provide 'natural' alias rejection as they coincide with the aliased components of the baseband signal. The fact that CIC filters provide this good anti-aliasing property with only minimal arithmetic and storage requirements is what has led to them becoming widely used in multirate signal processing.

The passband properties of the CIC filter are less desirable, although in a multirate decimation filter, anti-aliasing is more important at this point, as a flat passband at this early stage would be both costly and unnecessary. The passband properties of the overall decimation filter can easily be improved in a latter stage of the filter. There is a droop in the passband that is dependent on the decimation ratio, *R*, and the number of cascaded sections, *N*. Assuming constant *N*, for decimation ratios up to around 16, there is a marked difference in this droop. For decimation ratios greater than 16, droop remains fairly constant. The droop can be compensated for by designing a 'lift' in the frequency response of the post-CIC filtering. For multirate filters with programmable decimation ratios, the compensating filter must also be programmable. Other strategies exist for compensating for passband droop, which will be examined later.

CIC Bit Growth - Decimation

- The gain through a decimating CIC filter is given by $G = R^N$, where R and N are the decimation ratio and number of integrator/comb sections, respectively
- As we use a binary number system (2's complement), it follows that, for input bit width B_{IN}, the bit width at the filter output, B_{OUT}, should be:

$$B_{OUT} = B_{IN} + \lceil \log_2 G \rceil = B_{IN} + \lceil N\log_2 R \rceil$$

- The bit width B_{OUT} is also the required bit width at *each stage* of the filter
- If the value of B_{OUT} is greater than the number of bits required at the filter output, then some LSBs in earlier may be discarded, or *pruned*, according to a methodology proposed by Hogenauer [1]

Notes:

In order to permit full bit growth through the decimating CIC filter, the bit width at each stage must also be set to B_{OUT}. As the integrators have an infinite dc gain, it follows that the most significant bit at output must also be the most significant bit at the integrator sections of the filter. In order to allow a propagation path from the integrators through to the output, it also follows that B_{OUT} must be the required bit width for the comb sections as well.

For higher order filters and large decimation ratios, the size of B_{OUT} can be quite considerable. For example a CIC filter with N=5, R=32 and an input bit width of B_{IN} = 16, requires

$$B_{OUT} = B_{IN} + \lceil N\log_2 R \rceil = 16 + \lceil 5\log_2 32 \rceil = 46 \text{ bits}$$

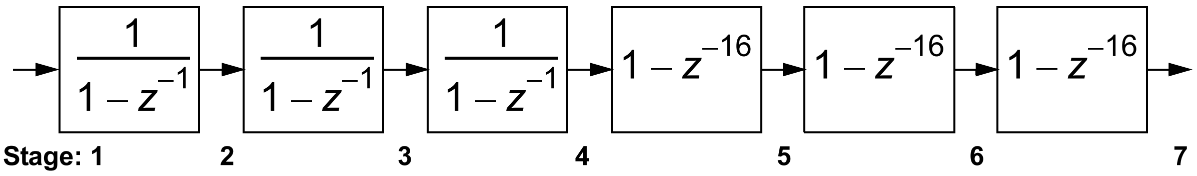
at each filter stage for full bit growth.

If the bit width required at the filter output is less than B_{OUT}, Hogenauer's pruning technique can provide a significant hardware saving by truncating (rounding is unnecessary) LSBs from intermediate filter stages. The pruning is carried out such that the quantization error introduced by discarding intermediate LSBs is not greater than the error introduced by truncating/rounding at the filter output.

Much of the following explanation on Hogenauer pruning is taken from [1] and [2]

CIC Decimation Filter - Pruning LSBs

- Consider a 3-stage CIC filter with a decimation factor of 16:



- If we want $B_{IN}=B_{OUT}=16$ bits, then the bit width at each stage must be truncated as follows:

Stage	Bit Width	Discarded Bits
1	27	1
2	24	4
3	21	7
4	20	8
5	19	9
6	18	10
7 (output)	16	12

Notes:

It will now be shown how we arrived at the number of bits to discard at each section by regarding each set of discarded bits as a noise source at the filter output. We begin by introducing some of the mathematics that will help us to do this.

Firstly, we understand that the probability distribution of an error introduced by truncating at stage *i* is assumed to be uniform with a width given by $E_i = 2^{B_i}$, where B_i is the number of bits discarded at stage *i.* The standard deviation of this error is $\sigma_i^2 = E_i^2/12$.

The standard deviation of the error at stage 7 can be readily determined, as we know in advance that we want an output of 16 bits, which means discarding 12 LSBs. Therefore we have:

$$\sigma_7^2 = E_7^2/12$$

$$\sigma_7^2 = 2^{2\times 12}/12$$

$$\sigma_7^2 = 2^{24}/12 = 1398101.33$$

which, as we will see, will come in useful later on.

CIC Decimation Filter - Pruning LSBs

- The total standard deviation error introduced by a noise source at stage *i* through to the filter output is given by:

$$\sigma_{T_i}^2 = \sigma_i^2 F_i^2 = \sigma_i^2 \sum_{k=0}^{L} h_i^2(k)$$

$$\text{where } L = \begin{cases} N(R-1)+i-1, & \text{i = 1,2,3} \\ 2N+1-i, & \text{i = 4,5,6} \end{cases}$$

- F_i^2 is the standard deviation error gain from stage *i* through to output, while the values $h_i(k)$ are transfer function coefficients at stage *i*

$$h_i(k) = \begin{cases} \sum_{d=0}^{\lfloor k/R \rfloor} (-1)^d \binom{N}{d} \binom{N-i+k-Rd}{k-Rd}, & \text{i = 1,2,3} \\ (-1)^k \binom{2N+1-i}{k}, & \text{i = 4,5,6} \end{cases}$$

Notes:

It is assumed that an equal level of noise will be contributed from each error source. The standard deviation from the first 2*N* (in this case 2*N*=6) error sources will thus be less than or equal to the standard deviation due to truncation or rounding at output:

$$\sigma_{T_i}^2 \le \frac{1}{2N}\sigma_{T_{2N+1}}^2 \quad \text{, i=1,2,...,2N}$$

$$\sigma_{T_i}^2 = \frac{1}{12} \times 2^{2B_i} F_i^2$$

Equating the RHS of the above two equations yields:

$$2^{2B_i} F_i^2 \le \frac{6}{N}\sigma_{T_{2N+1}}^2$$

In order to isolate B_i, the number of bits to be discarded at stage *i*, we take the log base 2 of both sides

$$B_i \le -\log_2 F_i + \log_2 \sigma_{T_{2N+1}} + \frac{1}{2}\log_2\left(\frac{6}{N}\right)$$

Finally, we apply the floor operator so as to obtain an integer value for B_i

$$B_i = \left\lfloor -\log_2 F_i + \log_2 \sigma_{T_{2N+1}} + \frac{1}{2}\log_2\left(\frac{6}{N}\right) \right\rfloor$$

From the previous slides, we have established how to calculate F_i^2 the standard deviation error gain, so F_i can be obtained by taking the square root of this value. At output (stage 7), F_7 reduces to 1, thus the total standard deviation error at output is simply the value of σ_7^2 that was calculated a few slides back.

CIC Decimation Filter - Pruning LSBs

- Calculation of LSBs to be discarded:

Stage	F_i	$-\log_2(F_i)$	$\sigma^2_{T_{2N+1}}$	$\log_2\sigma_{T_{2N+1}}$	$\frac{1}{2}\log_2\left(\frac{6}{N}\right)$	B_i
1	760.095	-9.570	1398101.33	10.208	0.5	1
2	64.125	-6.003	1398101.33	10.208	0.5	4
3	9.798	-3.292	1398101.33	10.208	0.5	7
4	4.472	-2.161	1398101.33	10.208	0.5	8
5	2.449	-1.292	1398101.33	10.208	0.5	9
6	1.414	-0.500	1398101.33	10.208	0.5	10
7	1	0	1398101.33	10.208	0.5	12

Notes:

CIC Bit Growth - Interpolation

- Unlike the CIC decimation filter, the CIC interpolation filter does not require a uniform bit width across all filter sections
- If we allow the combs to be numbered from 1 to *N* and the integrator sections to be numbered from *N*+1 to 2*N*, then the gain of the CIC interpolation filter at stage *i* is given by:

$$G_i = \begin{cases} 2^i, & i = 1, 2, \ldots, N \\ \dfrac{2^{2N-i}R^{i-N}}{R}, & i = N+1, \ldots, 2N \end{cases}$$

- For input bit width B_{IN}, in order to permit full bit growth, the required bit width at each filter stage is calculated as:

$$B_i = B_{IN} + \lceil \log_2 G_i \rceil$$

Notes:
Pruning cannot be performed at the output of the CIC interpolation filter as the integrators follow (rather than precede) the comb sections. Introducing a small quantization error in a comb stage of the CIC interpolation filter would cause the filter to be unstable, as the error would achieve infinite growth in the integrator sections.

The Sharpened CIC filter

- Earlier on in this section, when looking at the response of the CIC filter, it was noted that the passband, f_c, contained a decimation ratio-dependent 'droop'
- Generally, for CIC filters with programmable decimation ratios, the droop in the passband response will be compensated for using a second stage filter with *programmable* coefficients
- A filter with programmable coefficients brings additional hardware costs, for example, storage of several coefficient sets
- The next few slides introduce the Sharpened CIC (SCIC) filter, which can in certain circumstances reduce the hardware required for a multirate decimator design by eliminating the need for programmable passband droop correction

Notes:

The filter sharpening theory on which the SCIC filter was developed in the late 1970s by Kaiser and Hamming [5].

The theory centres on the idea of an *amplitude change function.* Applying a polynomial, *P*, to a filter transfer function *H(z)*, with slope=0 at *P[H(z)]*=0 *and P[H(z)]*=1 will result in a filter with improved passband *and* stopband properties. A full set of these polynomials, called Amplitude Change Functions is provided in [5], although only one is used in [3] for the SCIC filter.

An alternative CIC filter sharpening technique can be found in [4].

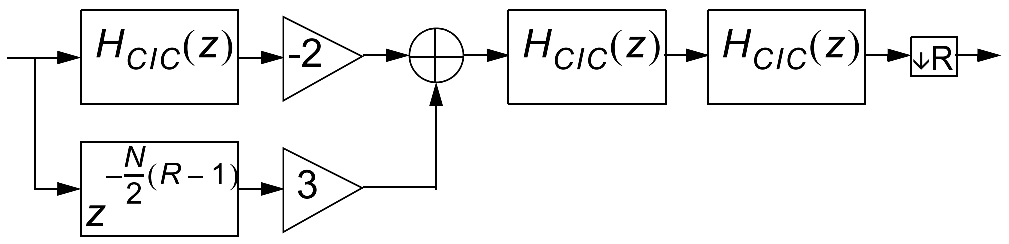
The SCIC Filter Transfer Function

- Deriving the transfer function of SCIC filter involves applying the simplest Kaiser and Hamming amplitude change polynomial to the transfer function of the CIC filter:

$$H_{CIC}(z) = \left(\frac{1}{R}\left(\frac{1-z^{-R}}{1-z^{-1}}\right)\right)^N$$

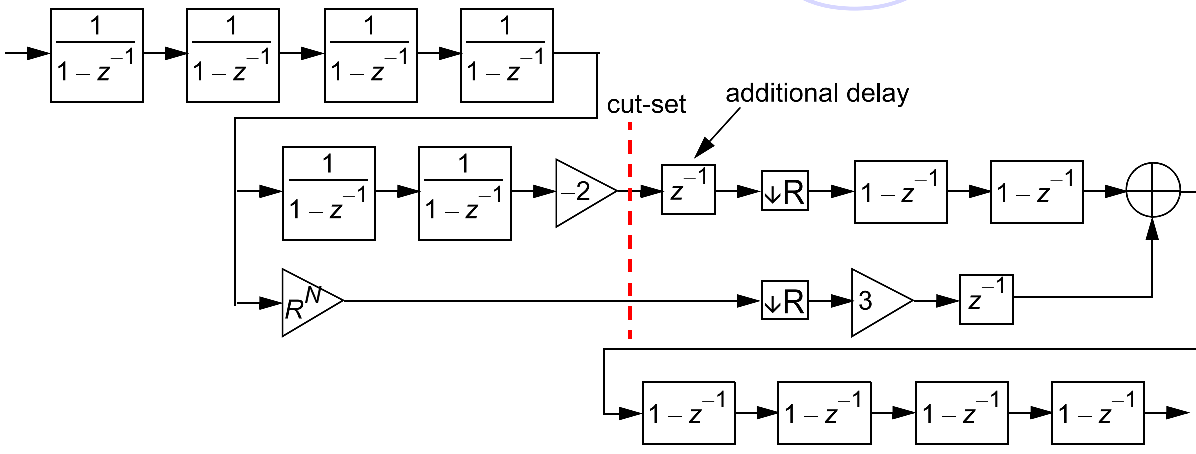
$$H_{SCIC} = 3H^2_{CIC}(z) - 2H^3_{CIC}(z)$$

- Thus, in a the form of a flow graph suitable for analysis we have:



Notes:

As with the CIC filter, the analysis form of the signal flow graph is not representative of how the filter will be implemented on hardware. The following signal flow graph, for the case of *N*=2, uses the noble identities to allow the downsampler to be moved between the integrator and comb sections, thus reducing storage requirements and permitting the same structure to be used for any decimation ratio.



Note that in the analysis flow graph, there is a delay of $\frac{N}{2}(R-1)$ required to manage the group delay of the first CIC filter. The addition of a delay at both branches in the *cut-set* in the flow graph just prior to the downsamplers allows this delay to be *NR*/2, which for *N*=2 results in a delay of *R*. Using the first noble identity, we can move the delay of *R* in the lower branch of the circuit to the other side of the downsampler, thus making it independent of *R*.

For odd values of *N*, the delay of $\frac{N}{2}(R-1)$ is not an integer for all values of *R*. In these cases, we cannot take advantage of the first noble identity to make the equalisation of group delay independent of *R,* thus the SCIC filter is only practical for even values of *N*.

SCIC Filter Passband

- Comparison made for different values of passband, f_c, between a CIC filter with N=4 and an SCIC filter with N=2

Filter	Passband Droop (dB)		Alias Rejection (dB)	
	$f_c = \frac{1}{8R}$	$f_c = \frac{1}{16R}$	$f_c = \frac{1}{8R}$	$f_c = \frac{1}{16R}$
CIC	0.8941	0.2230	68.3	94.1
SCIC	0.0640	0.0042	58.9	84.6

- Passband of SCIC filter is sufficiently flat to eliminate the need for programmable filters for some applications
- However, the SCIC filter requires two more integrators and two more comb sections than the CIC
- The benefit of using the SCIC filter must therefore be judged on how much hardware is saved further on in the processing chain

Notes:

The SCIC filter should not be regarded as a replacement for the CIC simply to avoid the problem of passband droop. There are limitations to the usefulness of the filter - we have already seen, for example, that SCIC filters with odd values of N are difficult to implement.

The number of integrators and combs required in an SCIC filter is 3N, so for a filter with N=4, 12 integrators and 12 combs would be needed!

Remember also that passband droop, besides being dependent on decimation ratio, R, is also dependent on the filter order, N. For values of N greater than 2, the SCIC will start to exhibit a level of passband droop that may also require programmable correction.

Finally, it should be remembered that in a complete multirate decimation filter, a flat passband is rarely important until the final stages of the filter. This is particularly true for the case of fixed decimation ratio systems, but for some programmable decimation ratio systems a hardware saving *may* be obtained by using an SCIC filter with fixed coefficient second stage filtering.

Recursive & Non-recursive architectures

- When introducing the CIC filter transfer function, it was noted that it was equivalent to a cascade of *N* moving average filters
- The main difference between the CIC and moving average based algorithms is the presence of *recursion* in the CIC architecture
- Although mathematically equivalent, as we will see over the following pages, each structure has benefits and drawbacks regarding circuit speed, power consumption, hardware area consumption and programmability of decimation ratio
- The technique of *stage-separating* the non-recursive architecture for low power consumption will be presented
- A non-recursive alternative to the SCIC filter will also be investigated

Notes:

Stage separation is the technique of factorising a filter transfer function and, through use of the noble identities, allowing much of the signal processing to be performed at a rate lower than input.

Consider the following example:

Recall that $H(z) = \left(\frac{1-z^{-R}}{1-z^{-1}}\right)^N = \left(\sum_{k=0}^{R-1} z^{-k}\right)^N$

For non-prime values of the decimation ratio, *R*, the filter may be separated into a cascade of *d* filters, where *d* is the number of prime integer factors of *R*. If *R* is a power-of-two, then $R=2^d$

So now we have: $H(z) = \left(\sum_{k=0}^{2^d-1} z^{-k}\right)^N = \prod_{i=0}^{d-1} \left(1+z^{-2^i}\right)^N$

For the case of d=3, that is $R=2^3=8$, $H(z)$ can be expanded thus:

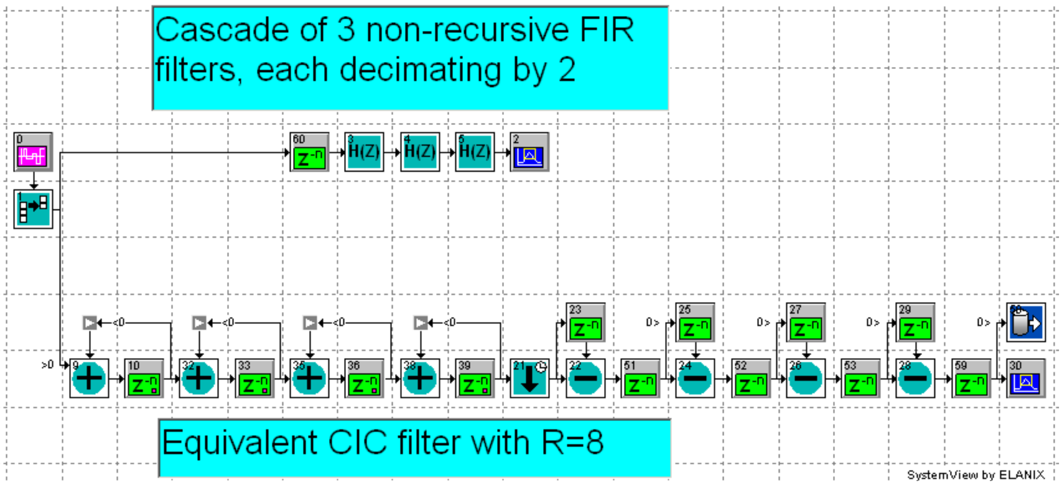
$$H(z) = \left(1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5}+z^{-6}+z^{-7}\right)^N = \left(1+z^{-1}\right)^N\left(1+z^{-2}\right)^N\left(1+z^{-4}\right)^N$$

Rather than perform all filtering before downsampling, the values of the delays in the factorised version of the transfer function allow us to conveniently exploit the first noble identity to separate the filter into stages, most of which run at a rate less than the input rate, thus lowering the power consumption of the circuit:

$$\begin{tikzcd} {} \arrow[r] & \boxed{H(z)} \arrow[r] & \boxed{\downarrow 8} \arrow[r] & \equiv \arrow[r] & \boxed{(1-z^{-1})^N} \arrow[r] & \boxed{\downarrow 2} \arrow[r] & \boxed{(1-z^{-1})^N} \arrow[r] & \boxed{\downarrow 2} \arrow[r] & \boxed{(1-z^{-1})^N} \arrow[r] & \boxed{\downarrow 2} \arrow[r] & {} \end{tikzcd}$$

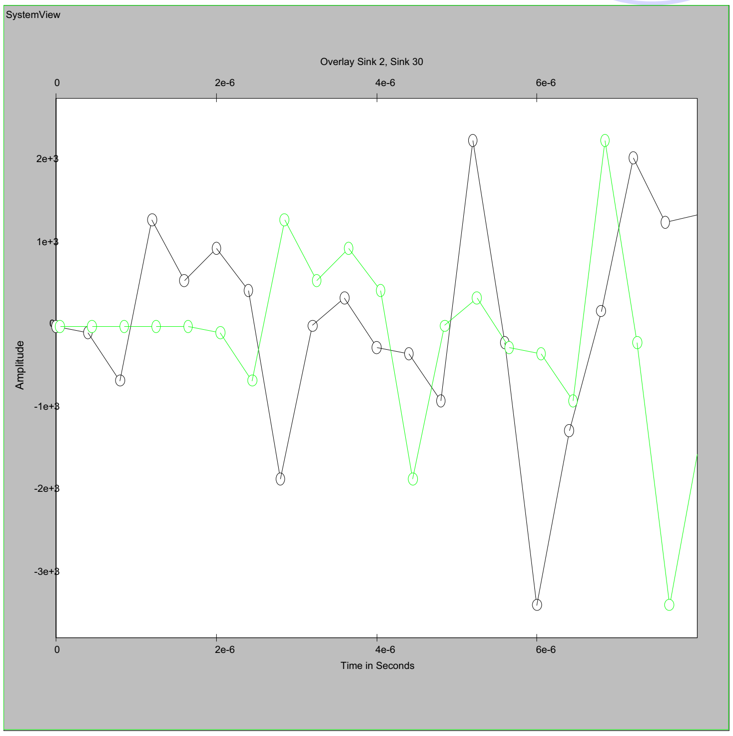
Power, Area and Speed Comparison

- Using SystemView and HDS, we can design a stage separated, non-recursive moving average filter and an equivalent CIC filter with *R*=8, *N*=4:


Cascade of 3 non-recursive FIR filters, each decimating by 2
Equivalent CIC filter with R=8
SystemView by ELANIX

- Rather than implement each non-recursive FIR filter as a cascade of 4 filters with coefficients (1,1), each will be implemented on hardware as a single full parallel, transpose form FIR filter with coefficients (1,4,6,4,1)

Notes:
Mathematically, we have shown these two systems to be equivalent, but just to demonstrate this fact, we can input a PN sequence to each filter (as shown in the SystemView system diagram). The outputs of both filters are identical, as shown below, except that the output from the CIC design, plotted in green, lags that of the CIC by several samples due to pipelining.



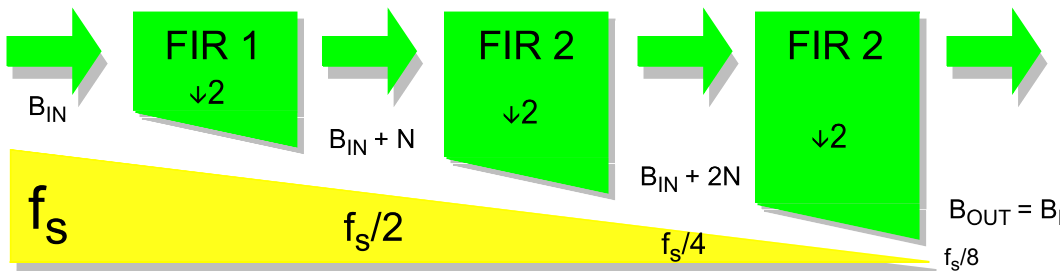
Power, Area and Speed Comparison

- Both designs taken through to “Place and Route” stage for Virtex 2 FPGA in Xilinx ISE
- Xilinx XPower used to obtain approximate indication of power consumption of each design after post-Place and Route

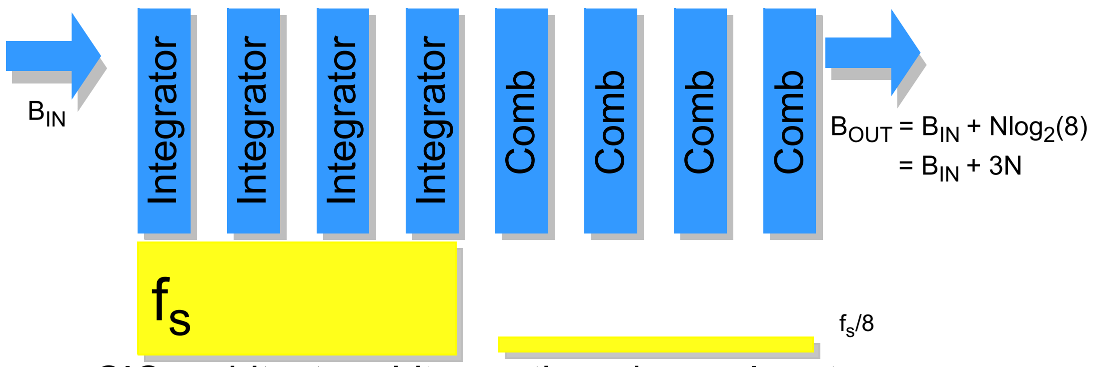
Filter Architecture	***FPGA Area Consumed (Slices)***	**Maximum Sample Rate (MHz)**	***Power Consumption (mW)***
CIC Filter	85	260	642
Non-recursive FIR Filter	240	330	514

- Example only shows results for a single case, refer to [6] for more complete data comparing power and area consumption and circuit speed of the CIC and non-recursive FIR architectures

Notes:
The difference in power consumption between the two filters can be understood by examining the following diagram of bit growth and circuit speed for the previous example. Note that in the non-recursive architecture, the increase in processing required as the bit width grows is offset by the reduction in sample rate at each stage. Power consumption in the CIC filter is dominated by the four integrator sections running at the input sample rate.



Non-recursive architecture bit growth and sample rate



CIC architecture bit growth and sample rate

A non-recursive 'SCIC' filter

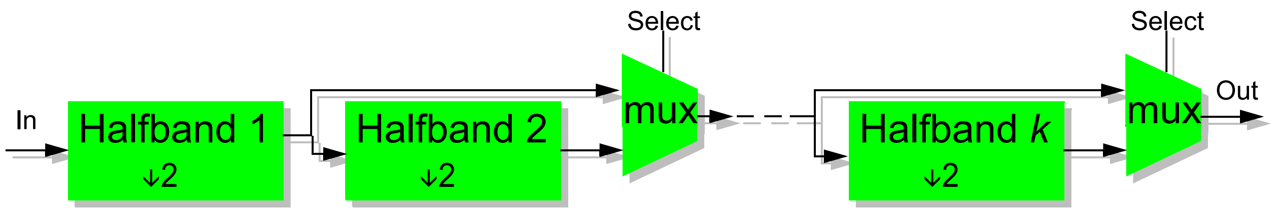
- For the CIC filter, we could produce a mathematically identical, non-recursive filter by *stage-separating* the filter transfer function
- In exchange for increased area consumption, the non-recursive architecture gave us:
    - A higher maximum circuit speed
    - A lower power consumption
- Mathematically, stage-separating the SCIC filter in this way is not possible, however it is possible to achieve a non-recursive near equivalent to the SCIC filter using a cascade of very simple *halfband* filters, each of which will decimate its input signal by 2
- This filter can also be programmable (recall that, to be of benefit, the SCIC filter must be programmable) for power-of-two decimation ratios by selecting or de-selecting each halfband filter in the chain

Notes:

When the SCIC filter is decimating by 2, its mathematically equivalent transversal filter is a halfband filter with the coefficients (-2,0,18,32,18,0,-2).

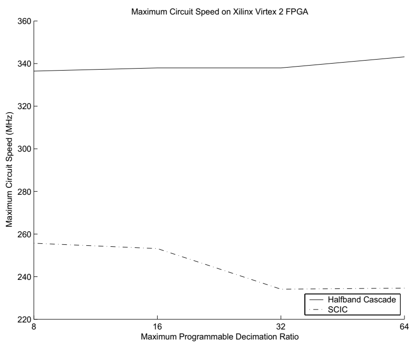
Unlike the non-recursive equivalent to the CIC filter, cascading two of these halfband filters does not produce a filter that is mathematically equivalent to a SCIC filter decimating by 4. However, from a functional perspective, the difference is small enough for us to consider a cascade of these halfband filters to be, in some cases, a viable alternative to the SCIC filter.

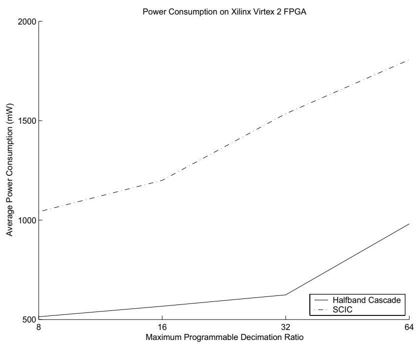
A schematic of the filter is shown below. Programmable power-of-two decimation ratios from 2^1 upwards are achieved by selecting or de-selecting halfband filters in the processing chain using the multiplexers. A maximum decimation ratio of 2^k can be obtained.

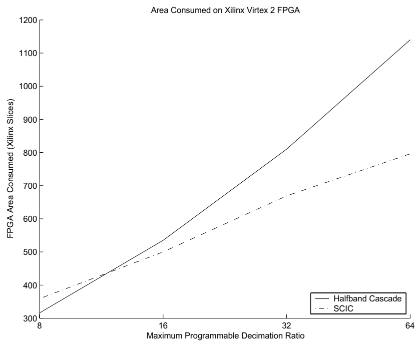


FPGA Implementation

- Graphs show results of ‘Post Place and Route’ data for the SCIC and halfband cascade on Xilinx Virtex 2 FPGA







Notes:

Data on FPGA area consumed was gathered from the “Map Report” stage of the Xilinx ISE design flow.

Data on maximum circuit speed was obtained from the “Post Place and Route Static Timing” stage of the Xilinx design flow.

Power consumption data was obtained by running a post “Place and Route” simulation and using the generated data to allow Xilinx XPower to make a “reasonable” calculation of the power consumed in each circuit.

Recursive & Non-recursive architectures

- The last few pages of theory and examples have compared the recursive CIC and SCIC architectures with some non-recursive FIR-based alternatives
- We can conclude that the recursive CIC and SCIC structures have the following advantages:
    - Low area consumption
    - Structures are not as restricted to power-of-two decimation ratios as non-recursive FIR alternatives
- We can also conclude that, in the cases examined, non-recursive, transversal FIR-based cascades have the following advantages:
    - Low power consumption
    - High maximum circuit speed

Notes:

Conclusions

- In this section we introduced the Cascaded Integrator-Comb filter as an efficient anti-alias/anti imaging filter in a multirate system
- Wordlength growth in CIC filters was examined and we looked at how ‘bit-pruning’ could reduce hardware resource consumption
- The Sharpened CIC (SCIC) filter structure was presented as a possible alternative to the CIC filter in situations that satisfied the following minimum criteria:
    - The circuit requires to be programmed for *several* decimation ratios
    - A *programmable* coefficient second stage filter would *normally* be used to compensate for CIC passband droop if
- Finally, we looked at the advantages and disadvantages of non-recursive transversal FIR filter based alternatives to the CIC and SCIC filters

Notes: