Sampling Rate Converter with FIR design

July 1st, 2017

O.D

History

Date	Draft	Author	Comment
July 1st, 2017	0	O.D	Creation

Definitions

- A sample rate converter (**SRC**) is a filter structure that performs resampling of the input signal
- In general finite impulse response (**FIR**) filtering is used
- A FIR filter has a finite impulse response of length N and filtering process is a **convolution** with the input samples:

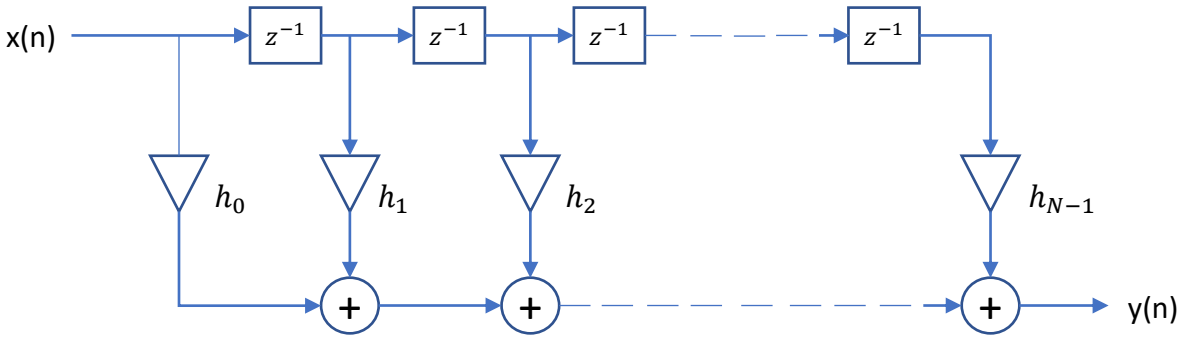
$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

- In practice, the SRC is implemented using a **polyphase** structure

FIR filter

- The FIR **order** is equal to the FIR **taps** count – 1

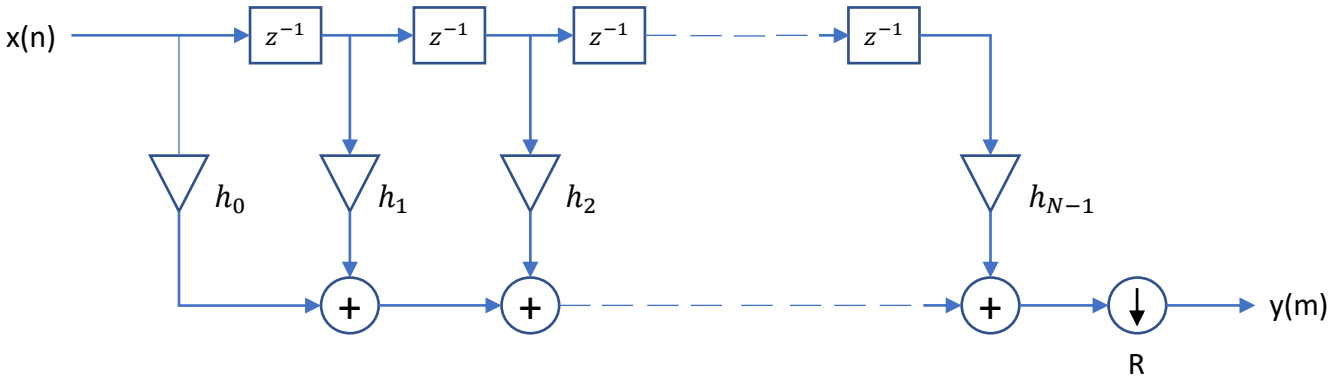
- FIR **canonical** structure:



- Polyphase structure is equivalent to the classical structure, it‘s just an efficient way to save MIPS and increase throughput

Decimation filter: canonical structure

- A **decimation** filter is a low pass FIR filter followed by a decimation



- The FIR canonical structure is implemented into a more efficient way to avoid computing the output samples that are dropped by the decimator: polyphase structure

Decimation filter: polyphase structure

- Consider an input buffer of 32 samples which are going to be filtered with a 16 taps FIR filter and then, decimated by a ratio of R = 8

- FIR filtering:

$$y(0) = h(0)x(0) + h(1)x(-1) + h(2)x(-2) + \cdots + h(15)x(-15)$$
$$y(1) = h(0)x(1) + h(1)x(0) + h(2)x(-1) + \cdots + h(15)x(-14)$$
$$y(2) = h(0)x(2) + h(1)x(1) + h(2)x(0) + \cdots + h(15)x(-13)$$
$$\cdots$$
$$y(8) = h(0)x(8) + h(1)x(7) + h(2)x(6) + \cdots + h(15)x(-7)$$
$$\cdots$$
$$y(16) = h(0)x(16) + h(1)x(15) + h(2)x(14) + \cdots + h(15)x(1)$$
$$\cdots$$
$$y(24) = h(0)x(24) + h(1)x(23) + h(2)x(22) + \cdots + h(15)x(9)$$
$$\cdots$$
$$y(31) = h(0)x(31) + h(1)x(30) + h(2)x(29) + \cdots + h(15)x(16)$$

- Decimation: keep only output decimated samples y_dec = {y(0), y(8), y(16) and y(24)}

Decimation filter: polyphase structure

- We notice in this case after filtering process we keep only 4 filtered samples over 32

- So, instead of filtering each sample which is useless, the polyphase structure will filter only the useful samples:

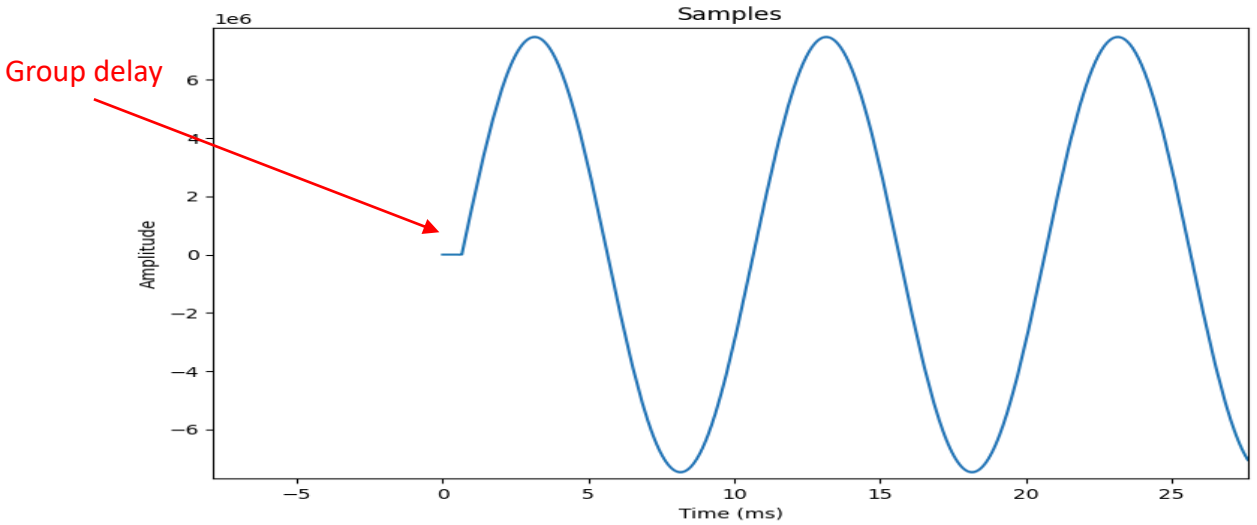
$$y\_dec(0) = h(0)x(0) + h(1)x(-1) + h(2)x(-2) + \cdots + h(15)x(-15)$$
$$y\_dec(1) = h(0)x(8) + h(1)x(7) + h(2)x(6) + \cdots + h(15)x(-7)$$
$$y\_dec(2) = h(0)x(16) + h(1)x(15) + h(2)x(14) + \cdots + h(15)x(1)$$
$$y\_dec(3) = h(0)x(24) + h(1)x(23) + h(2)x(22) + \cdots + h(15)x(9)$$

- We notice here that the polyphase implementation consists of just increasing the input buffer offset by 8 which corresponds exactly to the decimation factor!

Decimation filter: polyphase structure

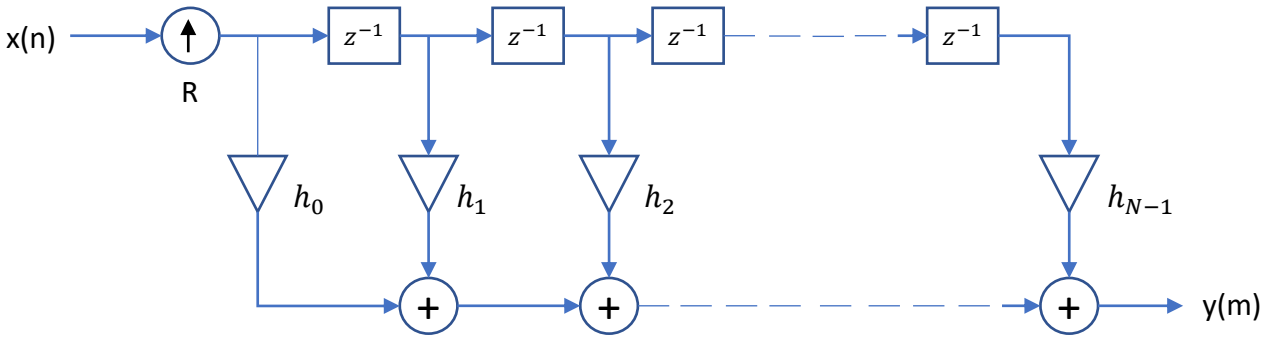
- Notice that FIR **delay line** is initialized with 0, that's the reason why the signal appears with a small delay at the beginning

- It's called the FIR **group delay** and it's equal to (N/2)x(1/R)



Interpolation filter: canonical structure

- An **interpolation** filter is an interpolator which inserts zeros between the input signal samples, followed by a FIR low pass filter



- The FIR canonical structure is implemented into a more efficient way to avoid the multiplication by zero: polyphase structure

Interpolation filter: polyphase structure

- Consider an input buffer of 32 samples which are going to be interpolated by a ratio of R = 3 and then filtered with a 24 taps FIR filter

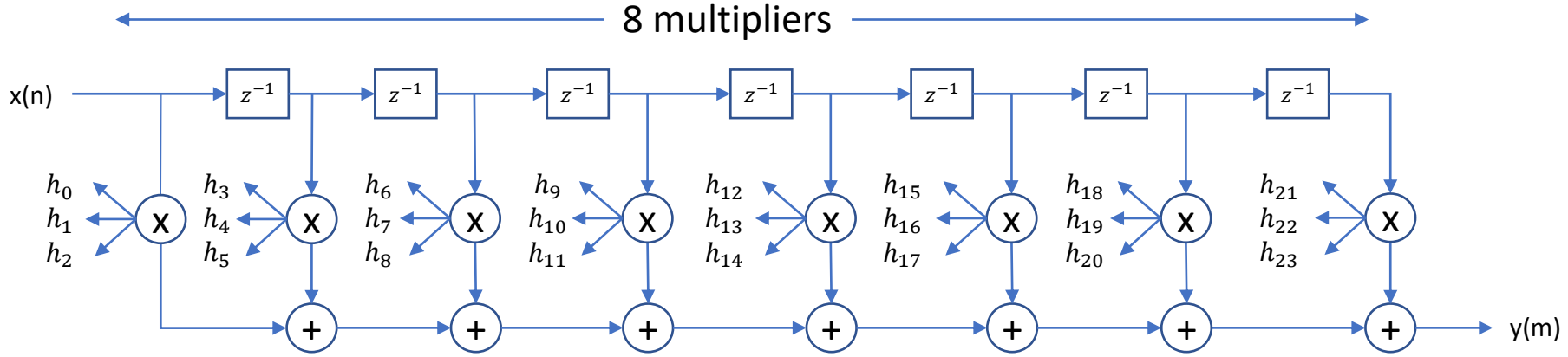
- Interpolator with R-1 zero insertion: x_inp = {x(0), 0, 0, x(1), 0, 0, x(2), 0, 0, x(3), 0, 0, . . .}

- FIR filtering:

$$y(0) = h(0)x\_inp(0) + h(1)x\_inp(-1) + h(2)x\_inp(-2) + \cdots + h(23)x\_inp(-23)$$
$$y(1) = h(0)\boldsymbol{x\_inp(1)} + h(1)x\_inp(0) + h(2)x\_inp(-1) + \cdots + h(23)x\_inp(-22)$$
$$y(2) = h(0)\boldsymbol{x\_inp(2)} + h(1)\boldsymbol{x\_inp(1)} + h(2)x\_inp(0) + \cdots + h(23)x\_inp(-21)$$
$$\dots$$
$$y(32) = h(0)\boldsymbol{x\_inp(32)} + h(1)\boldsymbol{x\_inp(31)} + h(2)x\_inp(30) + \cdots + h(23)x\_inp(9)$$
$$\dots$$
$$y(64) = h(0)\boldsymbol{x\_inp(64)} + h(1)x\_inp(63) + h(2)\boldsymbol{x\_inp(62)} + \cdots + h(23)\boldsymbol{xinp(41)}$$
$$\dots$$
$$y(93) = h(0)x\_inp(93) + h(1)\boldsymbol{x\_inp(92)} + h(2)\boldsymbol{x\_inp(91)} + \cdots + h(23)\boldsymbol{xinp(70)}$$
$$y(94) = h(0)\boldsymbol{x\_inp(94)} + h(1)x\_inp(93) + h(2)\boldsymbol{x\_inp(92)} + \cdots + h(23)\boldsymbol{xinp(71)}$$
$$y(95) = h(0)\boldsymbol{x\_inp(95)} + h(1)\boldsymbol{x\_inp(94)} + h(2)x\_inp(93) + \cdots + h(23)x\_inp(72)$$

Interpolation filter: polyphase structure

- Only indexes 0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93 are valid samples, all the rest is 0 => we deduce the following polyphase structure:

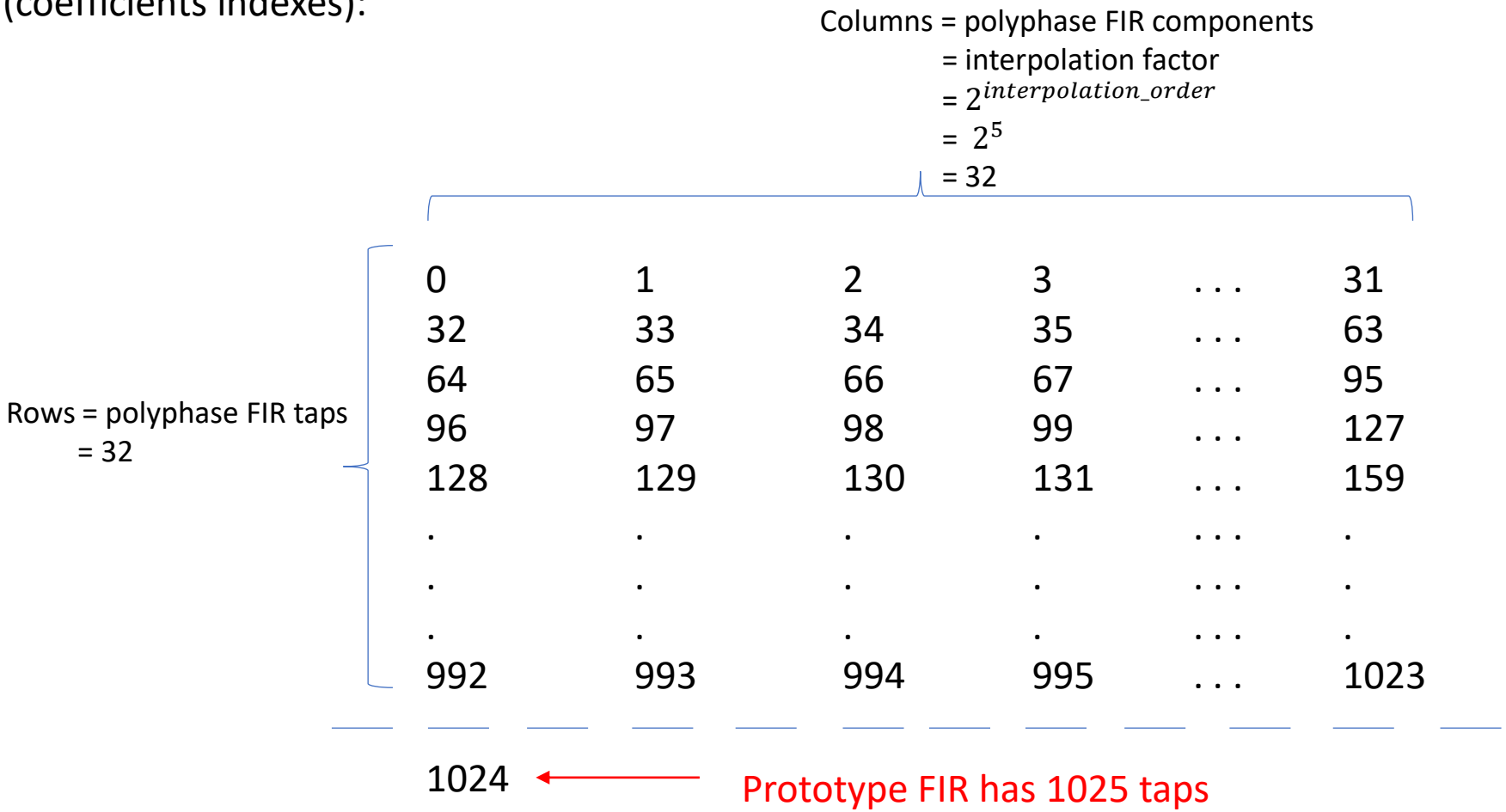


- We have 3 sub-filters of 8 taps that compose a **polyphase filter bank**

- We notice here that the polyphase implementation consists of filtering each input sample 3 times, which corresponds exactly to the interpolation factor, and then increasing the input buffer offset by 1!

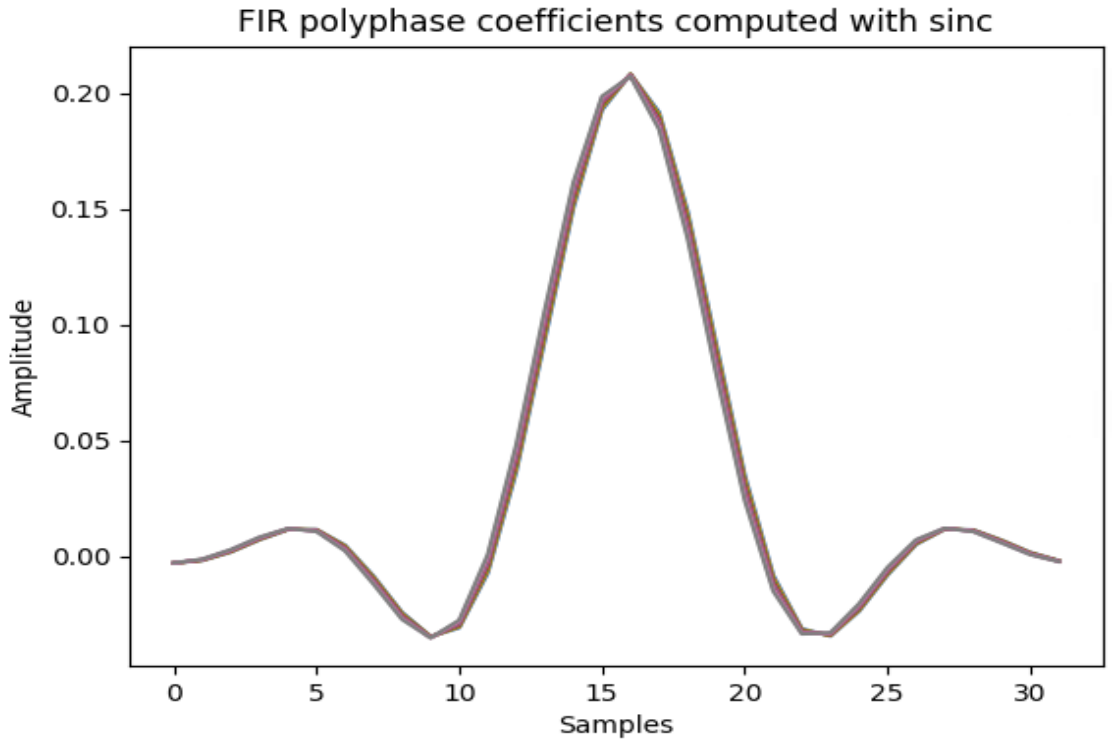
Polyphase FIR filter bank

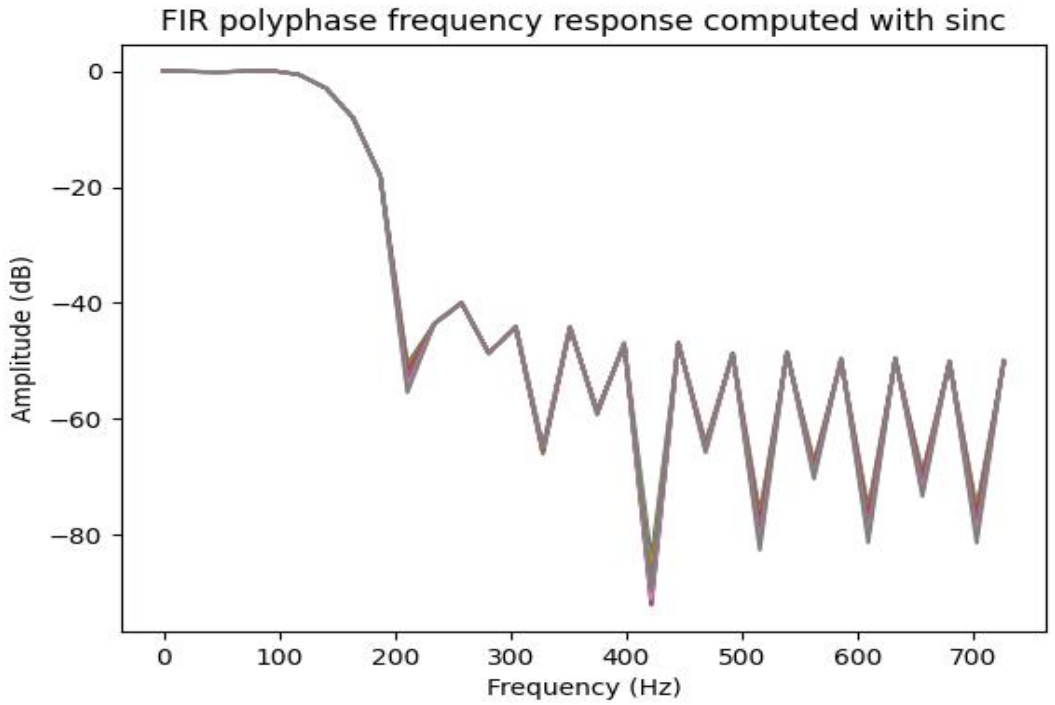
- E.g., we have a polyphase filter bank with 32 FIR filters of 32 taps, represented in the following FIR matrix (coefficients indexes):



Polyphase FIR filter bank

- The following plot shows the 32 polyphase components of our example:





Polyphase FIR filter bank

- In general case: $R = \frac{F_{in}}{F_{out}} = \frac{N}{D} = \frac{Offset\ increment\ per\ polyphase\ filters}{Polyphase\ filters\ count}$, e.g.:

R	Offset increment at each input sample	Polyphase FIR bank index
Up: 1/2	Total = 1: [0->1]	Total = 2: {32, 0}
Up: 1/3	Total = 1: [0->0->1]	Total = 3: {42, 63, 21}
Up: 2/3	Total = 2: [0->1->1]	Total = 3: {63, 42, 21}
Up: 3/5	Total = 3: [0->1->0->1->1]	Total = 5: {51, 25, 63, 38, 12}
Down: 2/1	Total = 2: [2]	Total = 1: {0}
Down: 3/1	Total = 3: [3]	Total = 1: {0}
Down: 3/2	Total = 3: [1->2]	Total = 2: {32, 0}
Down: 5/3	Total = 5: [1->2->2]	Total = 3: {63, 42, 21}
Down: 10/9	Total = 10: [1->1->1->1->1->1->1->1->2]	Total = 9: {14, 21, 28, 35, 42, 49, 56, 63, 7}

Polyphase FIR filter bank

- The offset increment and the polyphase bank index can be handled by a **phase accumulator of N-bits**, i.e., on each input sample we have:

$$\varphi(n) = \varphi(n-1) + \frac{F_{in}}{F_{out}}$$

- And:

$incr = \varphi(n)$ right shifted by **N** bits

$idx =$ interpolation_order LSbits of $\varphi(n)$ right shifted by $(\mathrm{N} -$ interpolation_order$)$ bits

$\mu = (\mathrm{N} - interpolation\_order)$ LSbits of $\varphi(n)$

Polyphase FIR filter bank

- For a better frequency resolution, we can interpolate different polyphase FIR banks
- In practice we use 2 kind of interpolations, linear or cubic (see Gardner reference)
- Cubic (Lagrange polynomials C_{-2}, C_{-1}, C_0, C_1):

$$FOR\ n = 0{:}nTaps$$
$$coeff(n) = C_{-2}(\mu) \cdot FIR_{n,idx-1} + C_{-1}(\mu) \cdot FIR_{n,idx} + C_1(\mu) \cdot FIR_{n,idx+1} + C_2(\mu) \cdot FIR_{n,idx+2}$$

- Linear (Taylor-Young):

$$FOR\ n = 0{:}\,nTaps$$
$$coeff(n) = FIR_{n,idx} + \mu \cdot \left(FIR_{n,idx+1} - FIR_{n,idx}\right)$$

Prototype FIR

- The polyphase filter bank is built from the prototype FIR filter

- The ideal prototype FIR is a low pass FIR defined by the gate function:

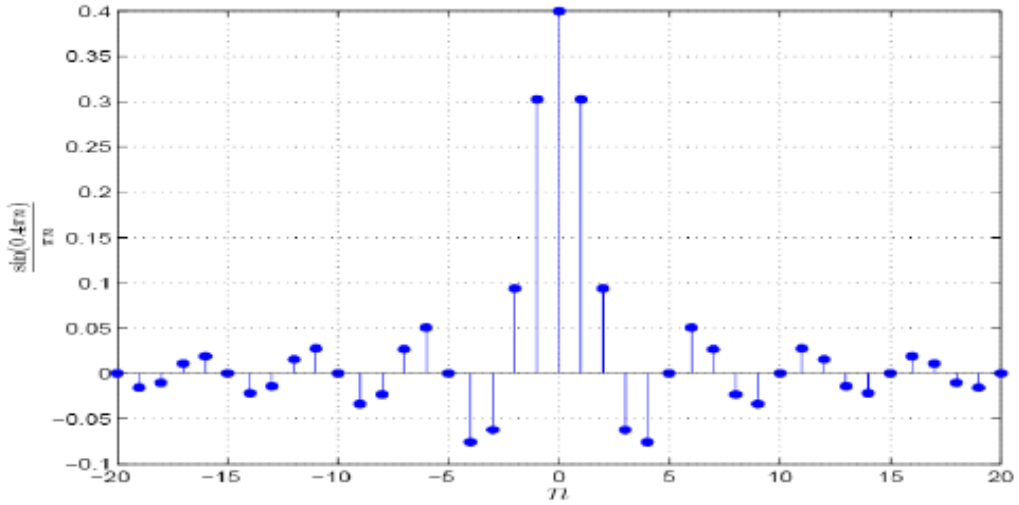
$$\mathrm{H}(\omega) = \begin{cases} 1, \ |\omega| \leq |\omega_c| \\ 0, \omega_c < |\omega| < \pi \end{cases}$$

Prototype FIR

- It can be shown easily that the FIR impulse response is:

$$h(n) = \frac{\omega_c}{\pi} \cdot \frac{\sin(n\omega_c)}{n\omega_c}$$

- Notice that the center tap is needed (but not necessary) in order to have a symmetrical FIR:



- In order to save resources, only the **half + center tap are used**, because of the FIR symmetry

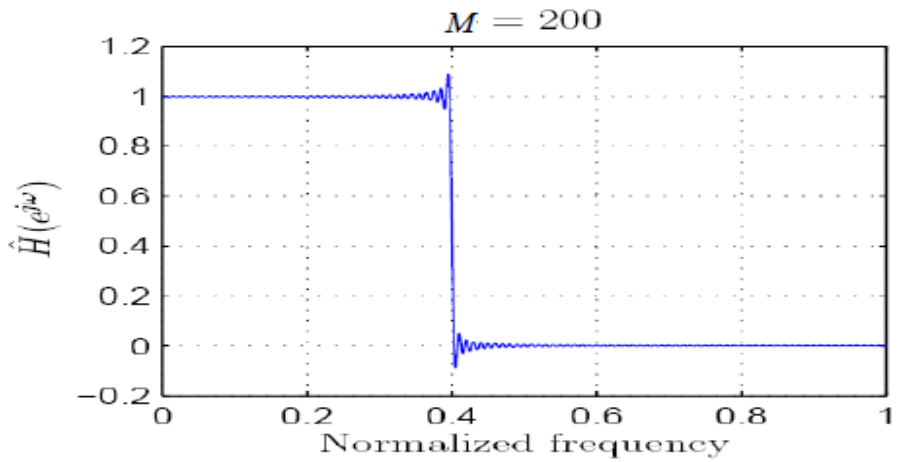
Prototype FIR

- So, the desired FIR impulse response has a **sinc** shape which is non-causal and infinite in duration => in practice it cannot be implemented

- We are going to approximate the FIR filter by **truncation**

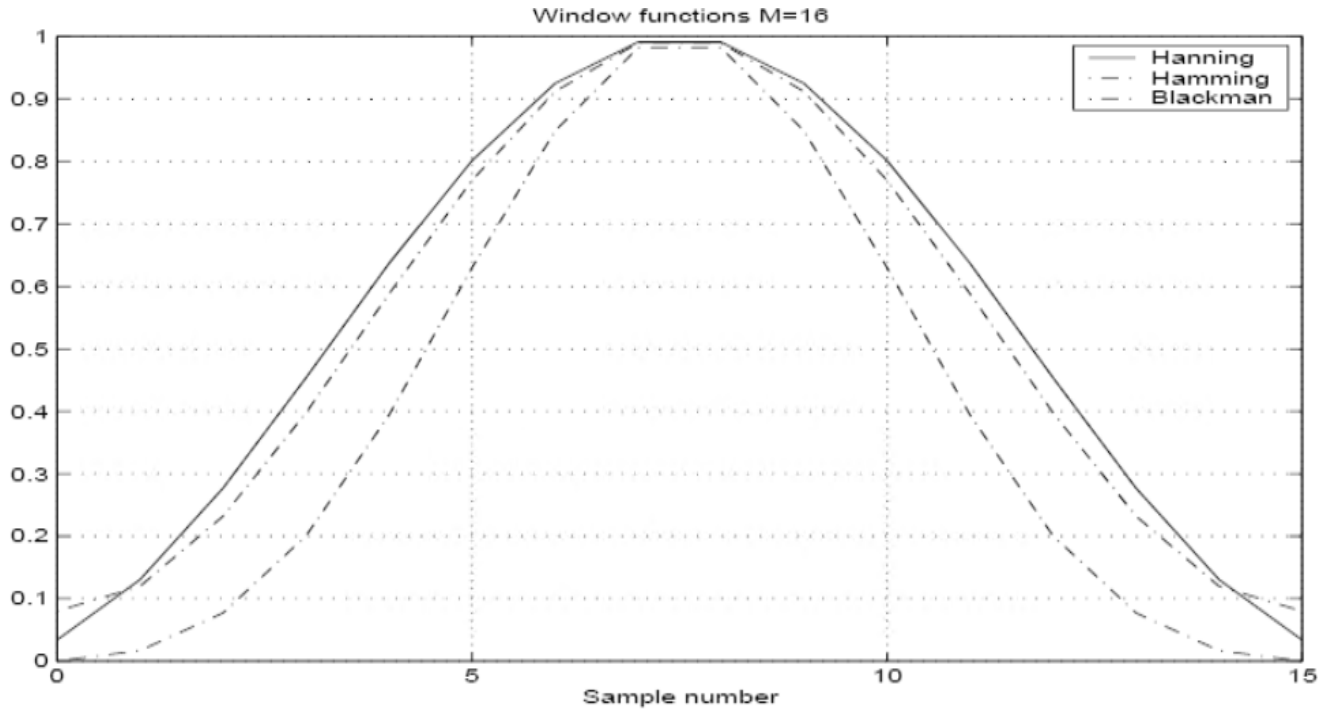
Prototype FIR

- The truncation will introduce **ripples** in both passband/stopband



Prototype FIR

- In order to reduce ripples, we need to use windowing: basically, the FIR impulse response is multiplied with a window coefficients



Prototype FIR

- The following plot shows the 1025 taps prototype FIR of our example built with Kaiser window: