Background to Software Defined Radio

Without the Maths!

Dr John Worsnop CEng MIET G4BAO

Background to Software Defined Radio

Without the Maths!

Dr John Worsnop CEng MIET G4BAO

Back to basics

- To make a radio, what do we need?
- Signal processing blocks
  - Amplification
  - Filtering
  - Mixing
  - Other processing, such as noise reduction

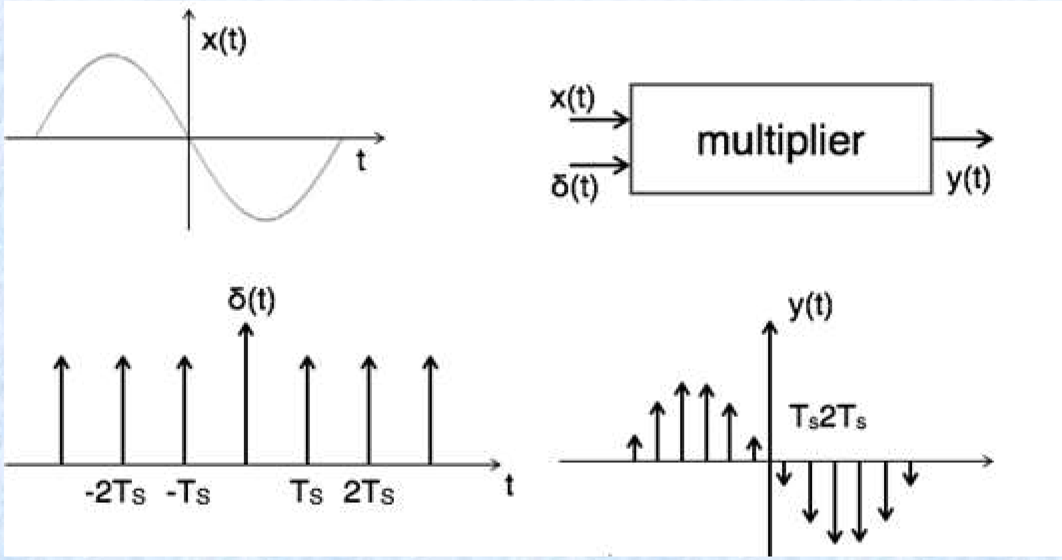
Background to SDR

- Sample rates and bit depths
- The Nyquist limit
- Time and frequency domains
- Transformation between them
  - How to use FFT and IFFT, not why they work
- Filtering basics
- Windowing
- Multiplication as the equivalent of mixing

Signal Sampling

- A time varying signal x(t) can be sampled by multiplying it with an impulse train δ(t)

- This results in a train of samples y(t)



Bit Depth

As each sample is stored as binary number, we refer to the number of bits used as “Bit depth”

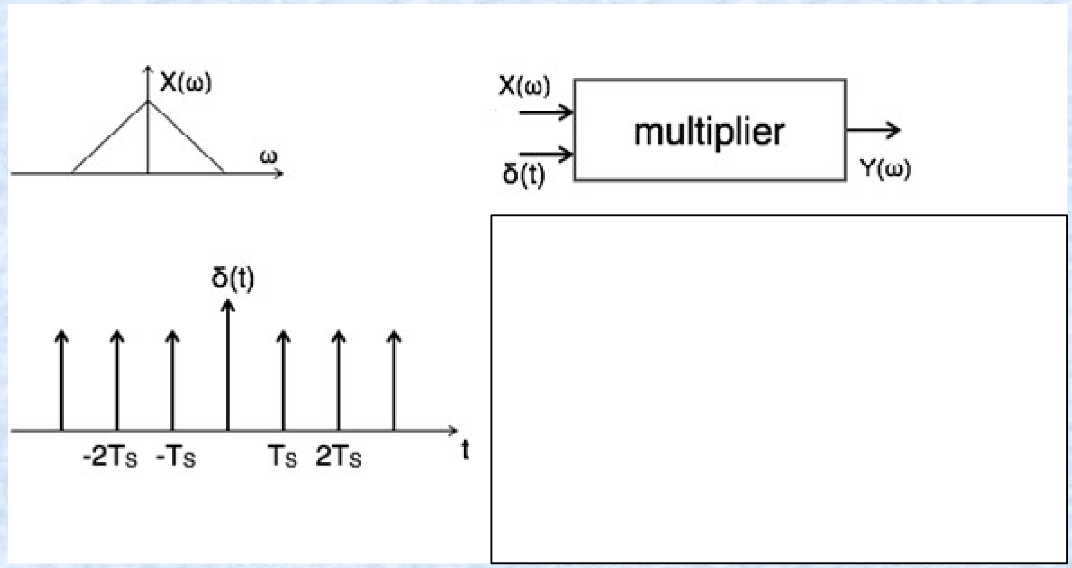
Take an example of colour information stored in an image.

The higher the **bit depth** of an image, the more colours it can store. The simplest image, a 1 **bit** image, can only show two colours, black and white.

Similarly a signal sampled with a bit depth of 3 can represent 2^3 (8) amplitude levels when restored.

Signal Sampling

- Similarly a complex input spectrum x(w) can be sampled by multiplying it with an impulse train δ(t)
- When recovered this results in a sampled spectrum Y(ω)
- Once you have as set of sample values (numbers) you can perform arithmetical operations on them



The Nyquist limit

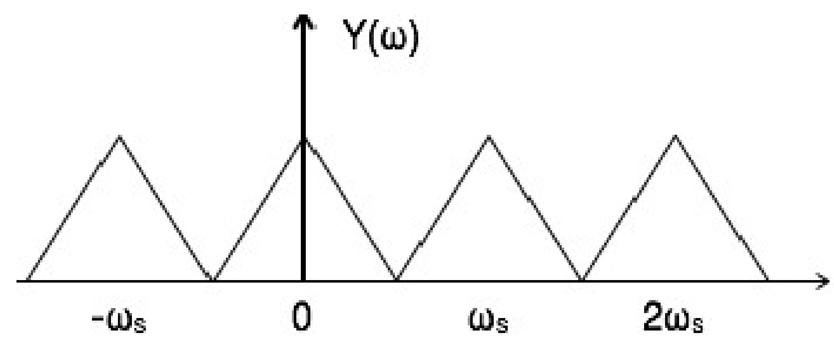
Nyquist's Sampling theorem

Any continuous time signal can be sampled and recovered when the number of samples per second is greater than or equal to twice the bandwidth of the original signal. (Or highest frequency component) if the signal goes down to zero

$$fs \geq 2\boldsymbol{BW}$$

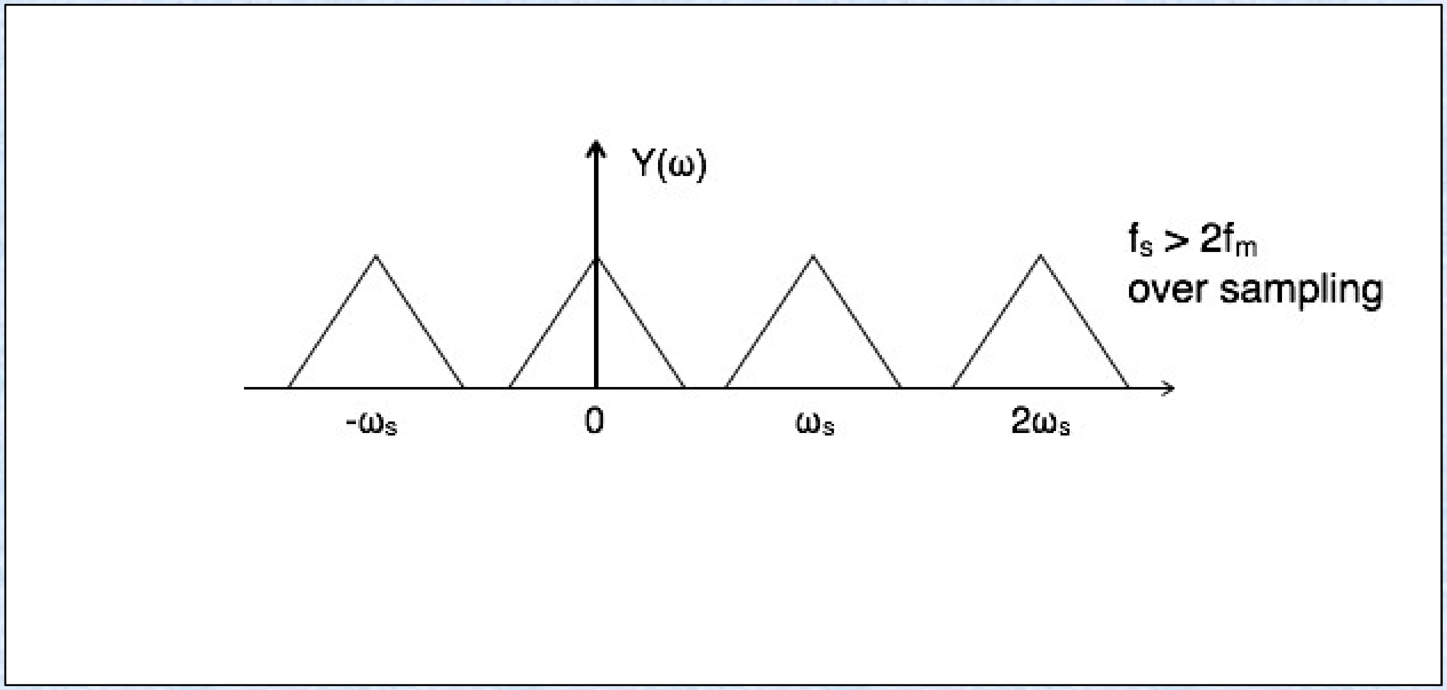
Over and under sampling

- Perfect sampling



Over and under sampling

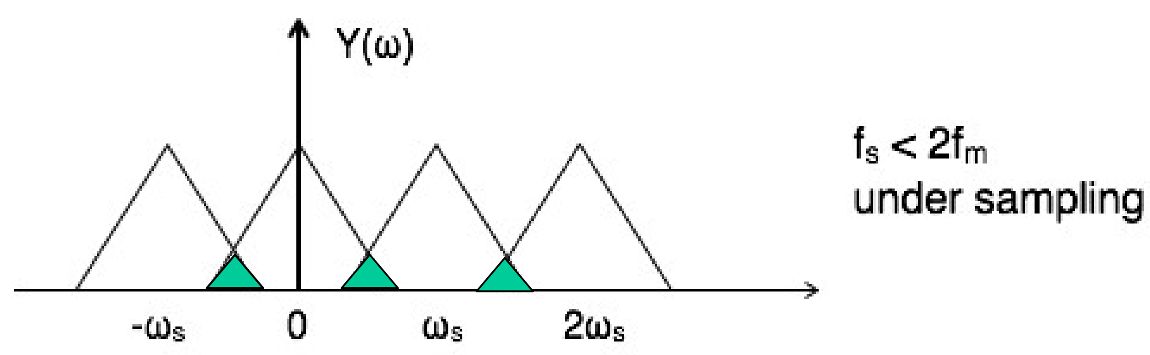
- Over sampling



- Original signal recovered normally

Over and under sampling

- Under sampling



- Causes “aliasing” or “frequency foldback” distortion on the recovered signal

Time and frequency domains

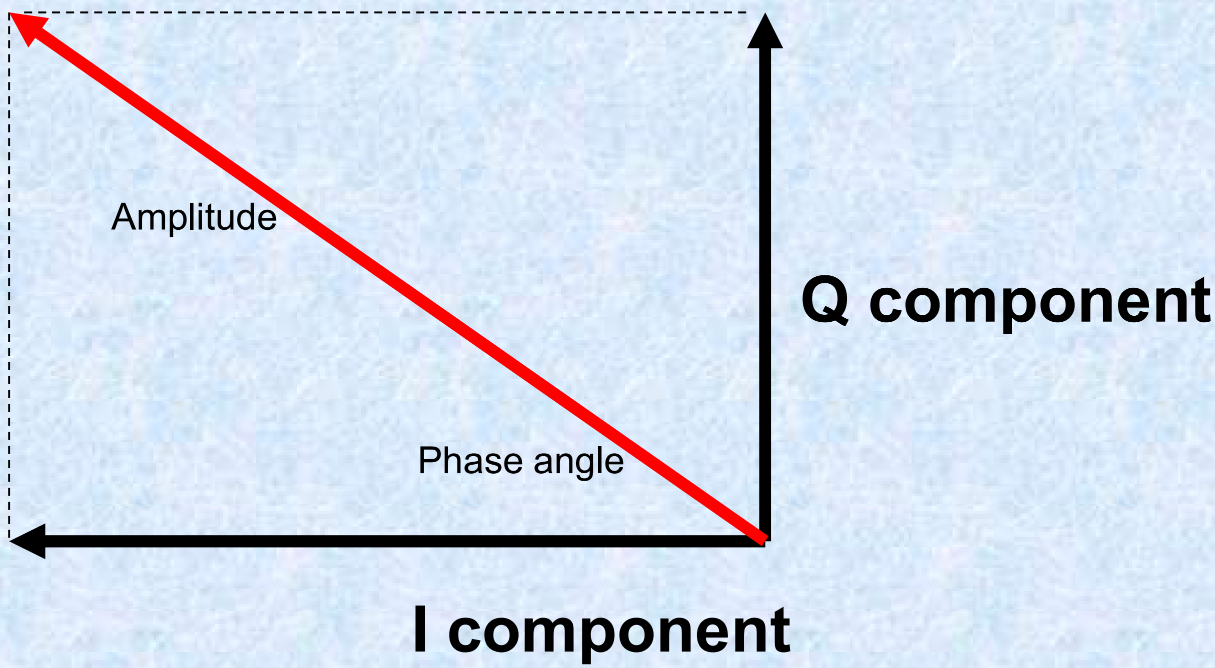
- Any signal can be represented in:
- The time domain (Oscilloscope display)
  - Amplitude vs time
- Or
- The frequency domain (Spectrum Analyser display)
  - Amplitude vs frequency

- It can be a vector signal (magnitude and phase)

- Or a scalar signal (amplitude only)

Time and frequency domains

- A vector (or “complex”) signal can be represented as two components 90 degrees out of phase:
- In maths it can be manipulated as a complex number



- This is usually called an “**IQ**” signal
  – (in **P**hase / in **Q**uadrature)

Transformation between time and frequency domains

- The **Fast Fourier Transform and Inverse Fast Fourier Transform** are very fast computer algorithms that perform a generalized mathematical process called the **Discrete Fourier transform (DFT)** to convert signals from one domain to another (time to frequency).

- **How to use FFT and IFFT**
- **FFT** converts time domain vector signal to frequency domain vector signal.
- **IFFT** converts frequency domain vector signal to time domain vector signal.

Real and floating point numbers

- A **Floating point** number is a computer science concept.
- You can't encode actual **real numbers** to store them in a computer because nearly all of them have, conceptually at least, an **infinite number** of decimal or binary places.
- So you make do with a binary version of scientific notation (like 6e-2 seconds to represent 60milliseconds) called “floating point”.

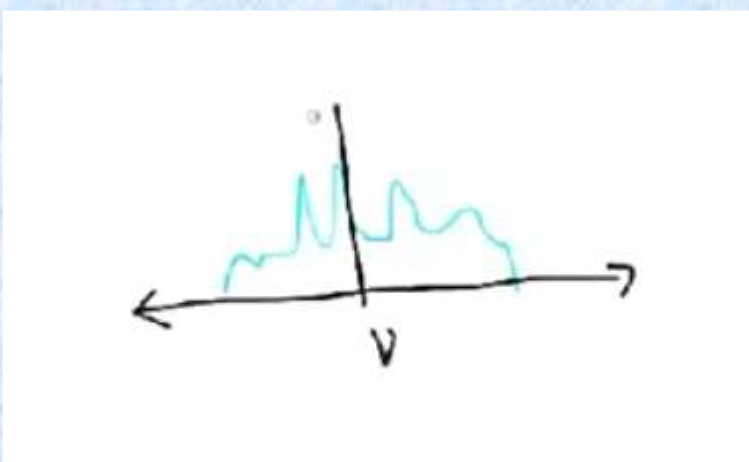
Filtering basics

Short intro to FIR filters (Finite impulse response filters)

A filter may be used to clean up a signal as below

Sampled signal   Low pass filtered signal

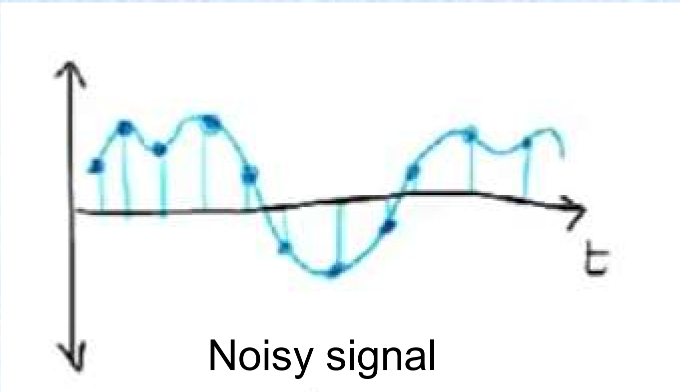
Take the FFT of the signal to get a spectrum



Filtering basics

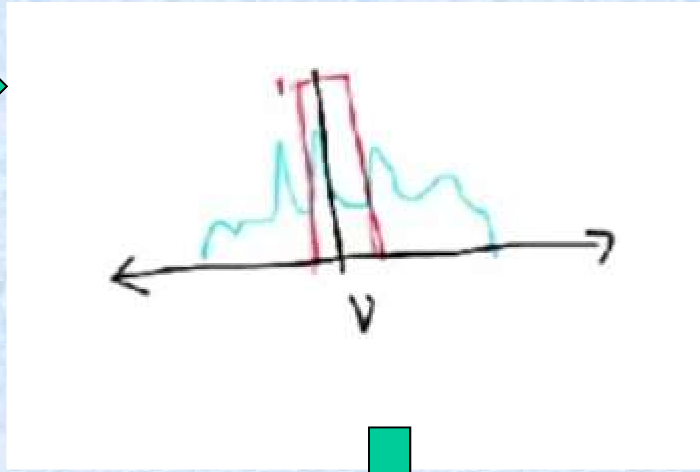
Multiply it by the FFT of the desired filter response

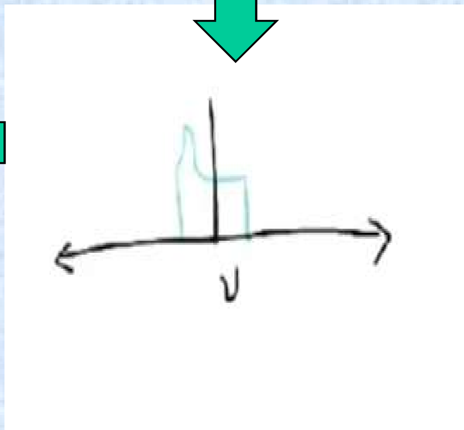
Leaving just the filtered signal



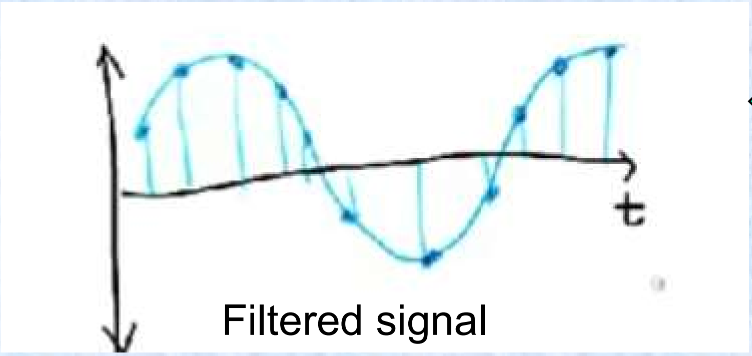
Noisy signal

FFT and
Multiply by filter





Inverse FFT



Filtered signal

Filtering basics

See ssb.grc

And

https://www.youtube.com/watch?v=NvRKtdrssFA

Filtering basics

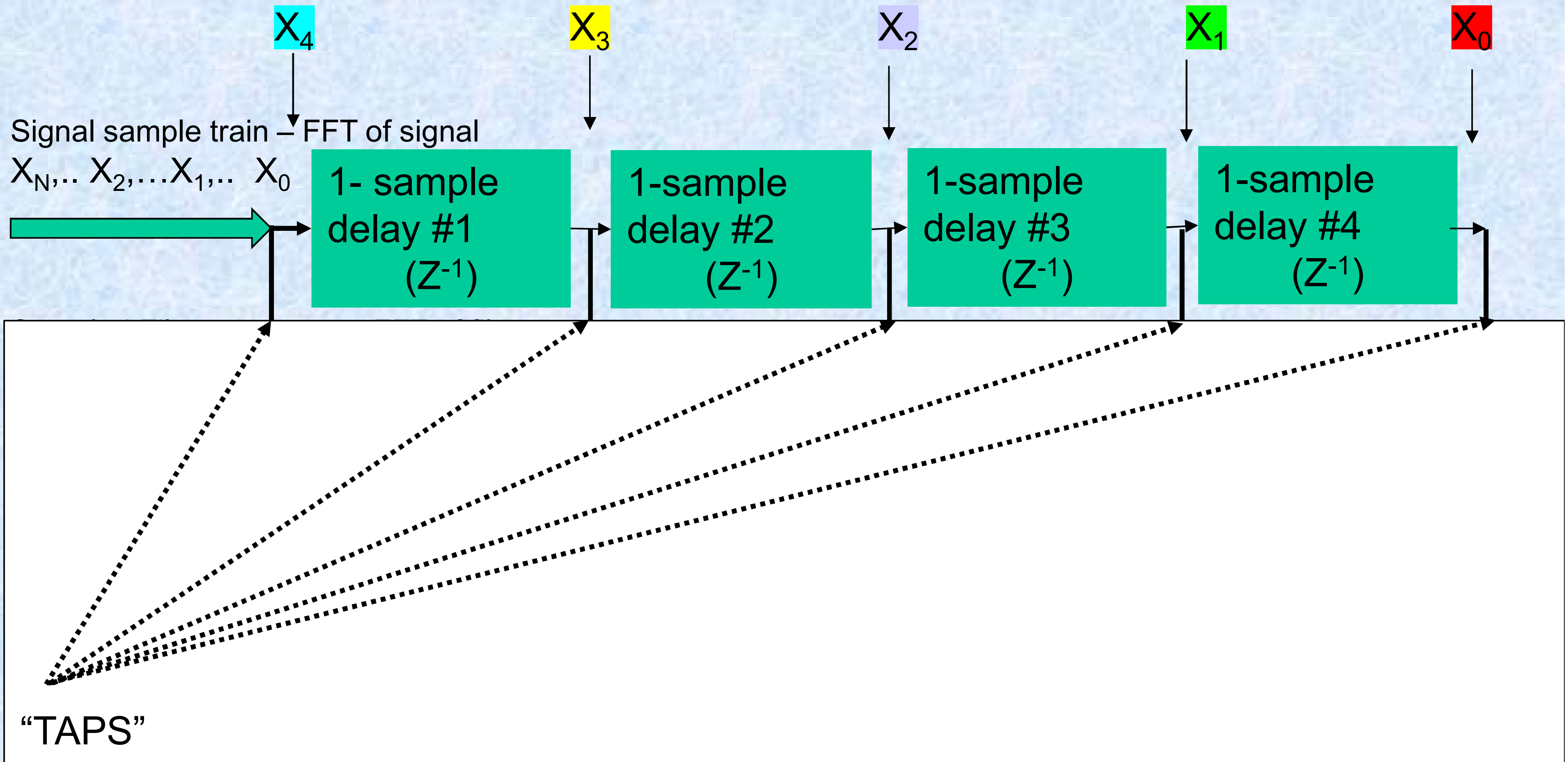
Convolution is a mathematical operation on two functions to produce a third function that expresses how the shape of one is modified by the other.

With 2 time functions, if you “slide one past the other” in time, (multiplying one by the other as you go),the convolution is the area under the curve. As per this animation link https://youtu.be/C1N55M1VD2o

Multiplication in the Fourier domain is Convolution

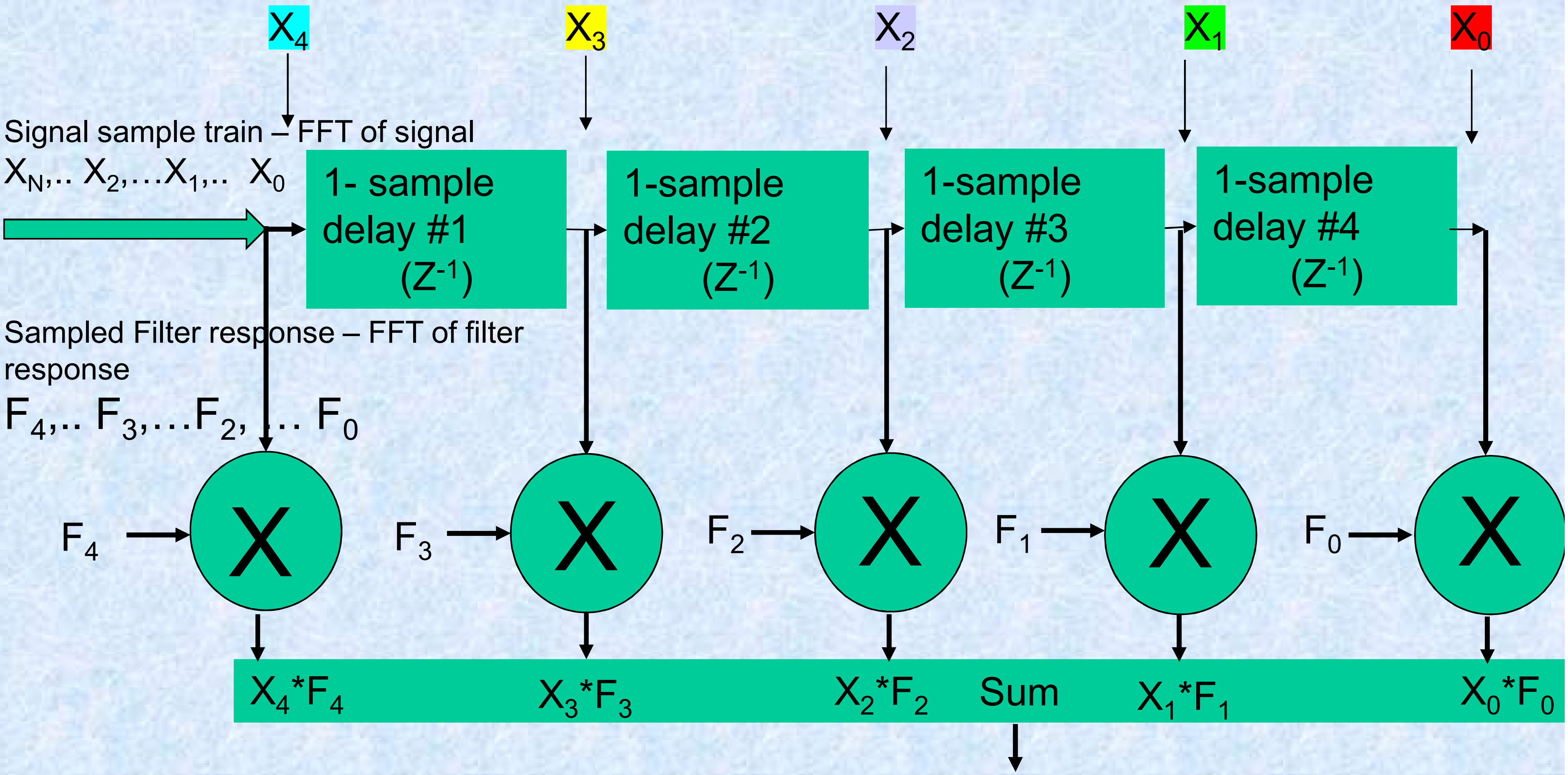
Filtering basics

To implement the convolution on a sampled signal



Filtering basics

To implement the convolution on a sampled signal

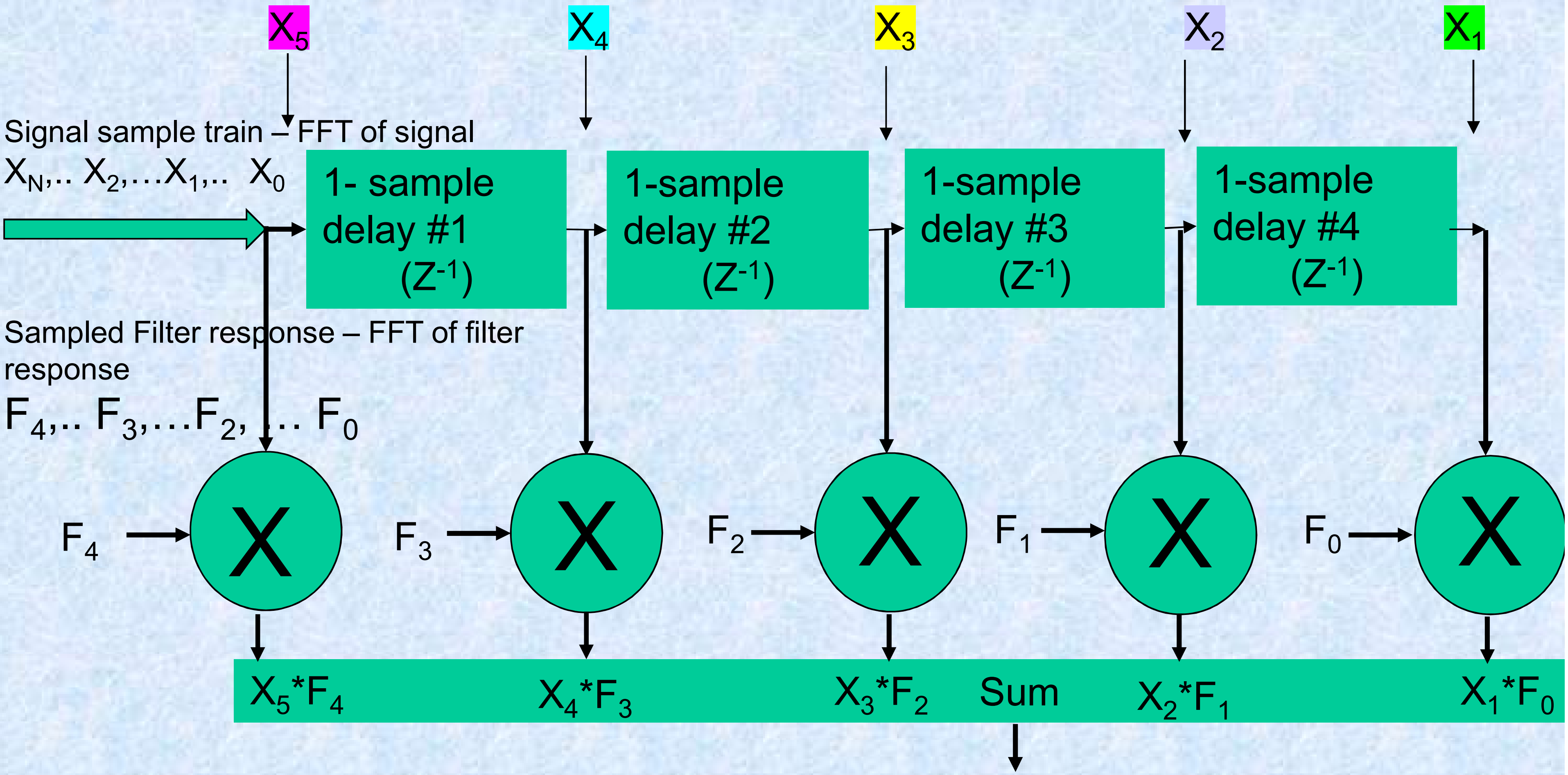


Sample 1 of FFT of filtered signal

Time index 1

Filtering basics

To implement the convolution on a sampled signal

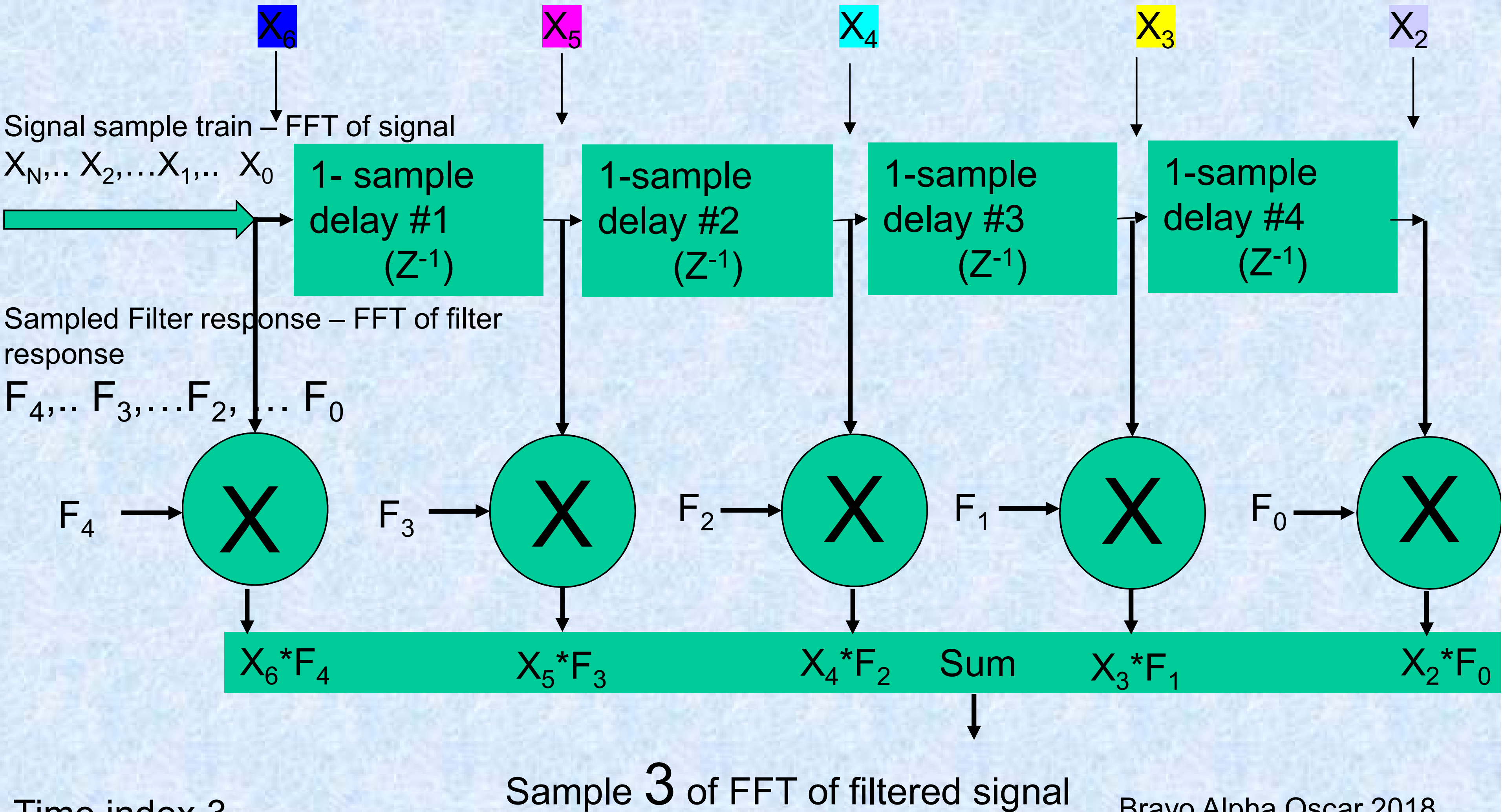


Sample 2 of FFT of filtered signal

Time index 2

Filtering basics

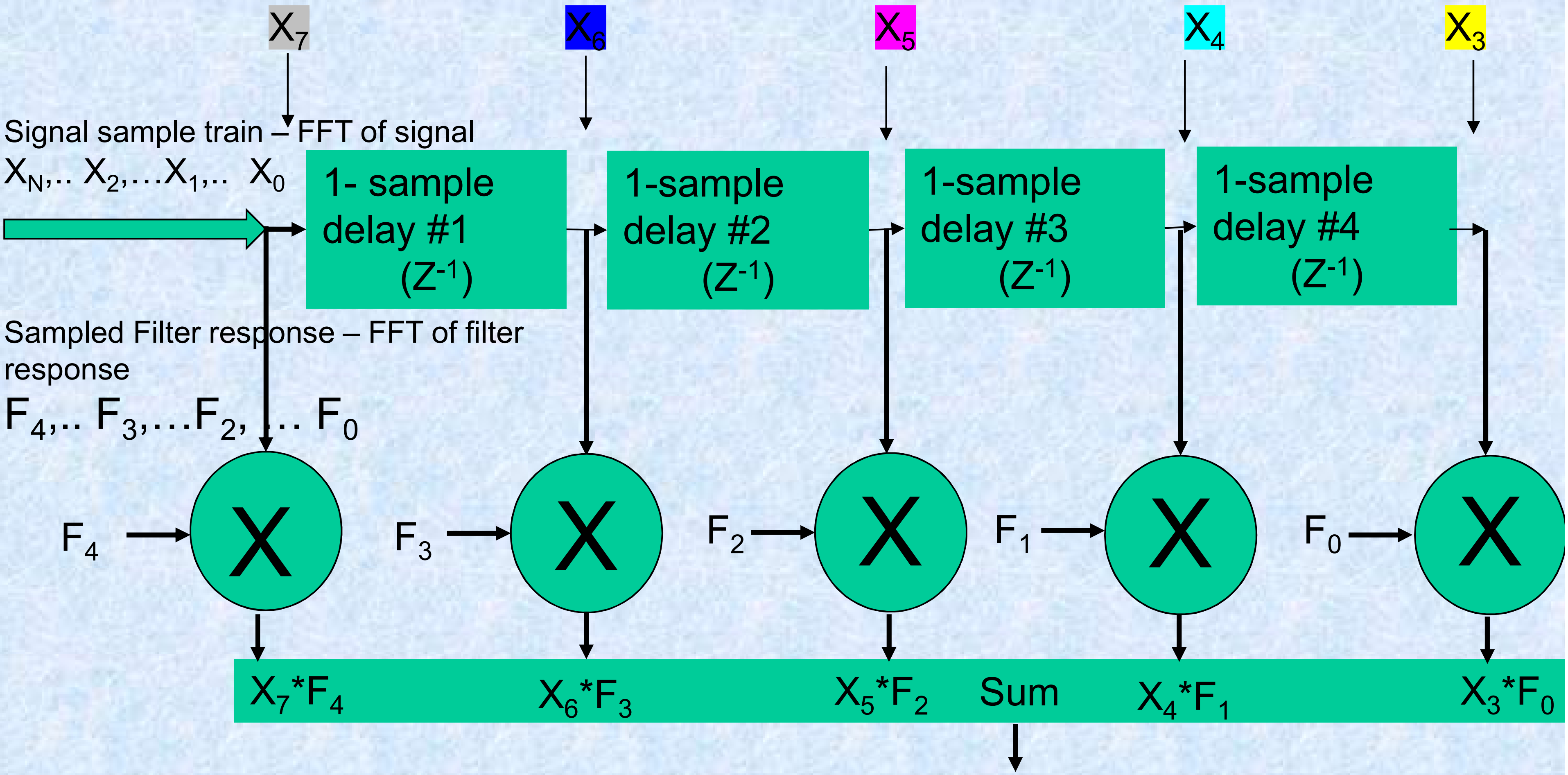
To implement the convolution on a sampled signal



Time index 3

Filtering basics

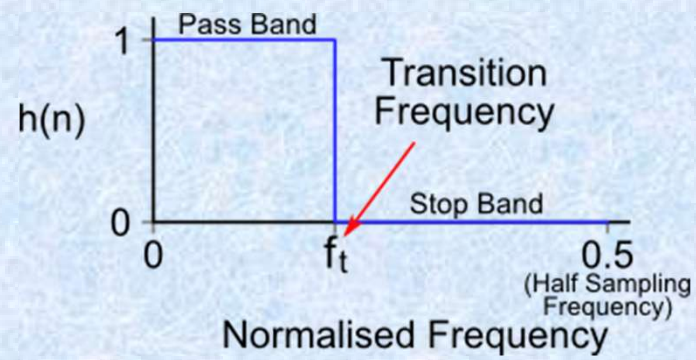
To implement the convolution on a sampled signal



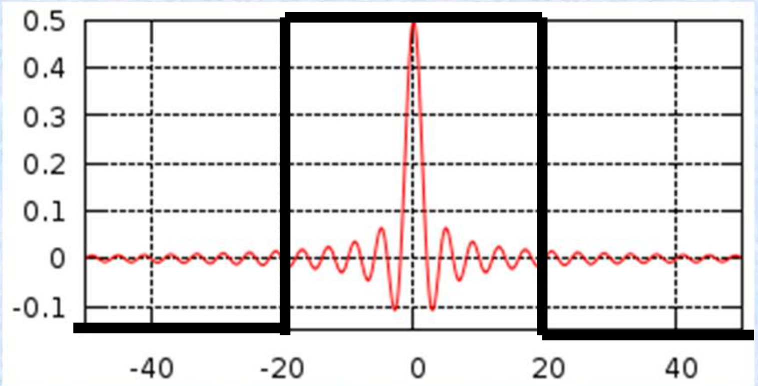
Sample 4 of FFT of filtered signal

Windowing

- **Ref :-** http://www.labbookpages.co.uk/audio/firWindowing.html
- The frequency response of an ideal low pass filter is a rectangle.

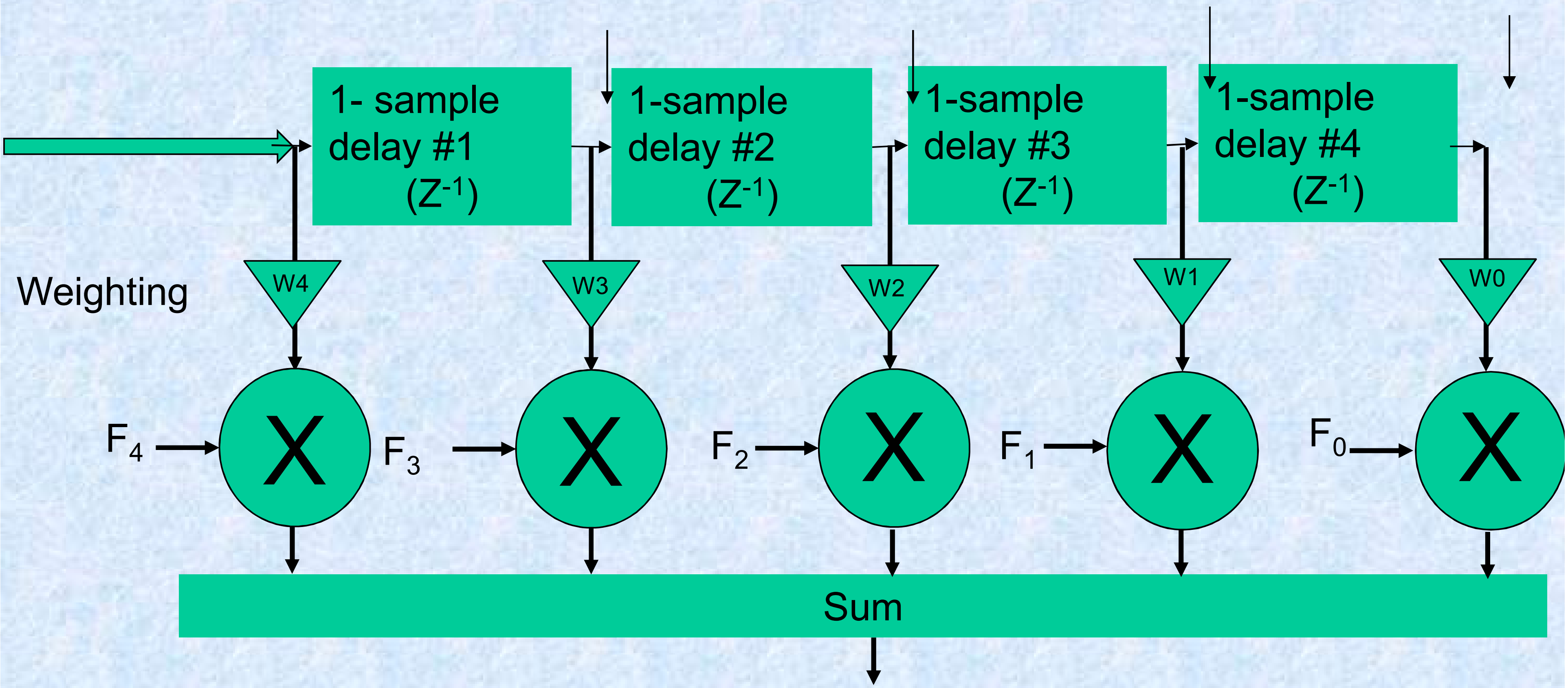


- It's Impulse response is a sinx/x or "sinc" function. It extends to infinity left and right
- This "infinite impulse response" is not possible with a FIR filter.
- To get over this we apply a "window" to the sinc function response by weighting each sample after it's delay.
- A rectangular window would just crop the sync function beyond + and – 20



Windowing

Each sample is “weighted by the Window function

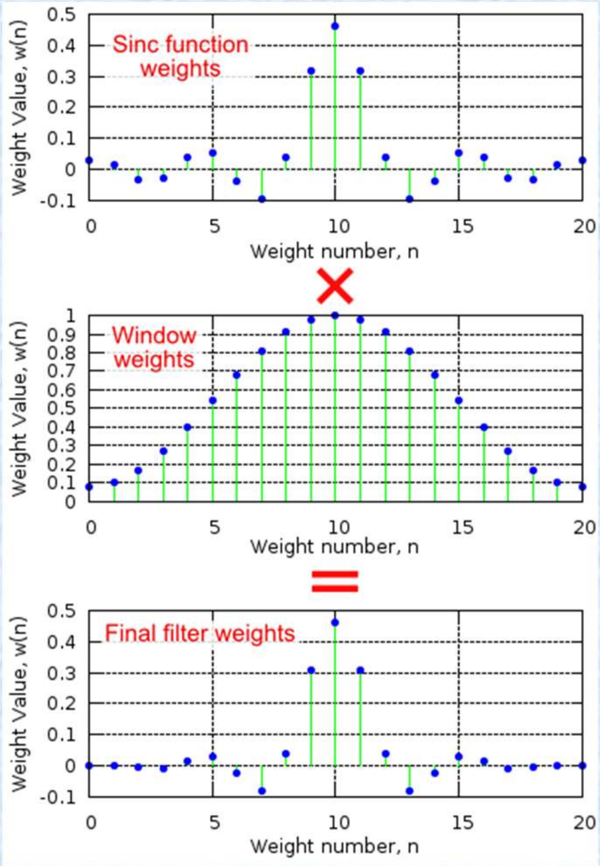


Windowing

- Ref :- http://www.labbookpages.co.uk/audio/firWindowing.html

- Applying a window to the sinc function weights provides extra control over the characteristics of the filter. The image to the right illustrates the process.
- First, the normal sinc weights are calculated, then the window weights are calculated, in this case a Hamming Window has been used, the equation is below.
- The two sets of weights are multiplied together to create the final set of filter weights.

$$w(n) = 0.54 - 0.46\, cos\left(\frac{2\pi n}{M}\right)$$



Filtering basics

Tap coefficients and samples can be real or complex

"D**ecimation**" is the process of reducing the sampling rate. In practice, this usually implies lowpass-**filtering** a signal, then throwing away some of its samples.

The decimation factor is simply the ratio of the input sample rate to the output rate. It is usually symbolized by "M", so (input rate/output rate)=M.

Filtering basics

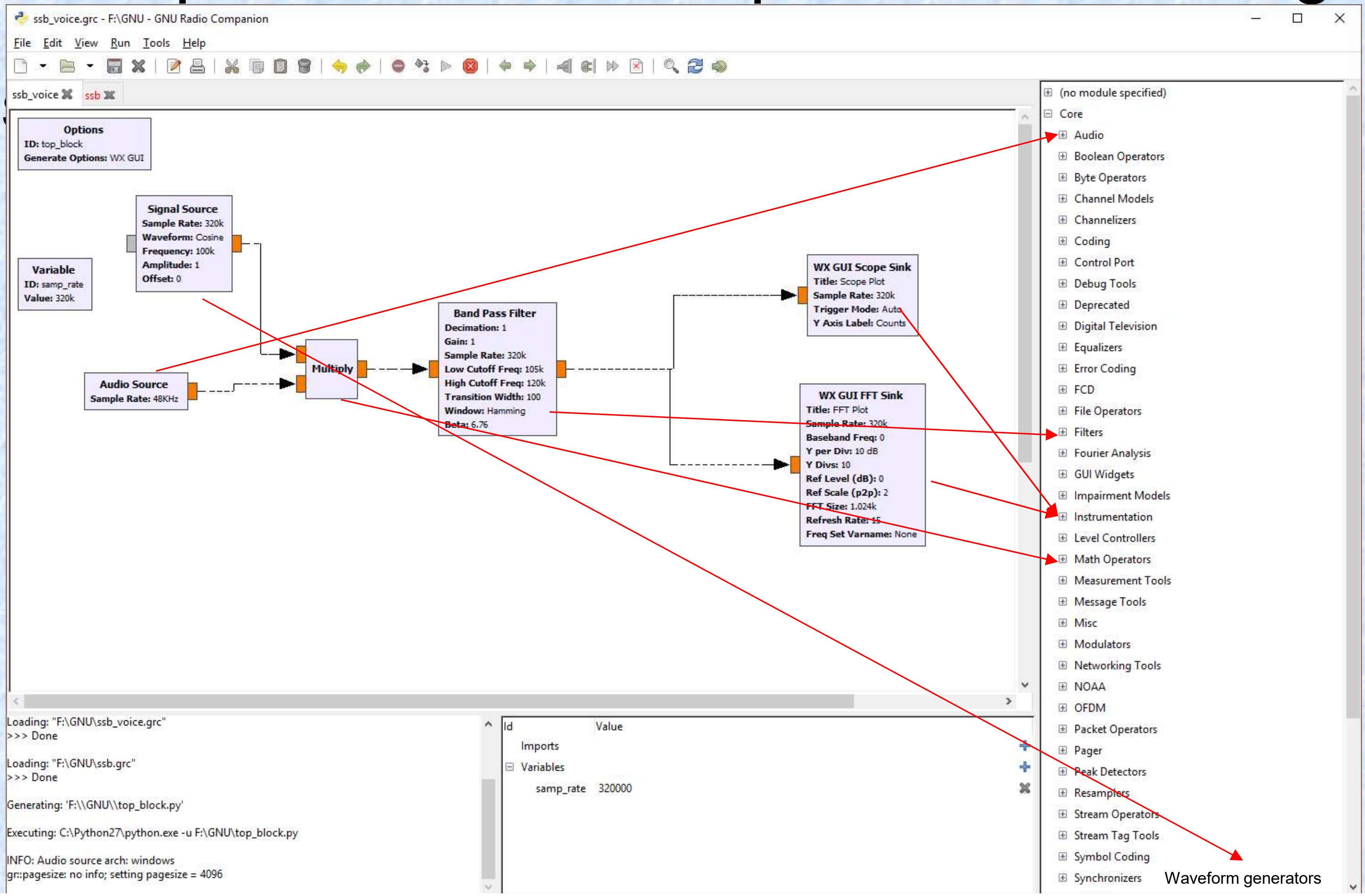
And now some good news!

The Gnu radio companion has a whole selection of standard filters plus a **filter design tool** to protect you from all this!

Signal mixing

- To finish this section
- We now have the basics of waveforms and filtering
- Finally – to represent a mixer, we just need mathematical multiplication

Multiplication as the equivalent of mixing



Summary

- We've looked at the tools needed to make a radio in the digital domain

- Maths of signal processing
  - Time and frequency domains
  - FFT / IFT to convert between them
  - Amplification – just scaling
  - Filtering – sample, shift, multiply and add
  - Mixing – just multiplication
  - Other processing, such as noise reduction is just clever filtering!