

Background to Software Defined Radio

Without the Maths!

Dr John Worsnop CEng MIET G4BAO



Bravo Alpha Oscar 2018

Background to Software Defined Radio

Without the Maths!

Dr John Worsnop CEng MIET G4BAO

This presentation is released under a Creative Commons licence.

Details of how and where you may re use or modify this this can
be found at

<http://creativecommons.org/licenses/by-sa/4.0/>



Back to basics

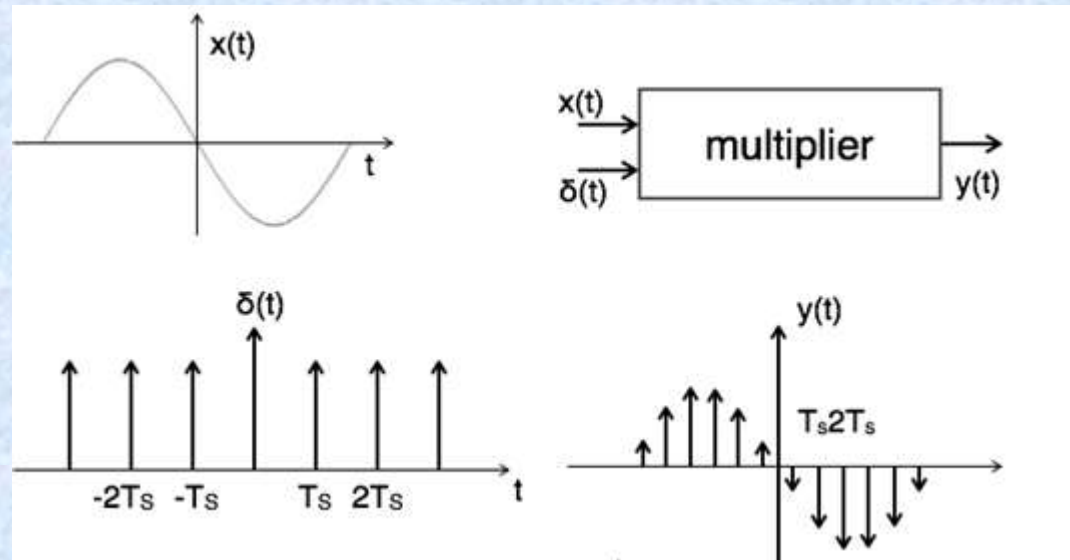
- To make a radio, what do we need?
- Signal processing blocks
 - Amplification
 - Filtering
 - Mixing
 - Other processing, such as noise reduction

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 $\delta(t)$
- 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”

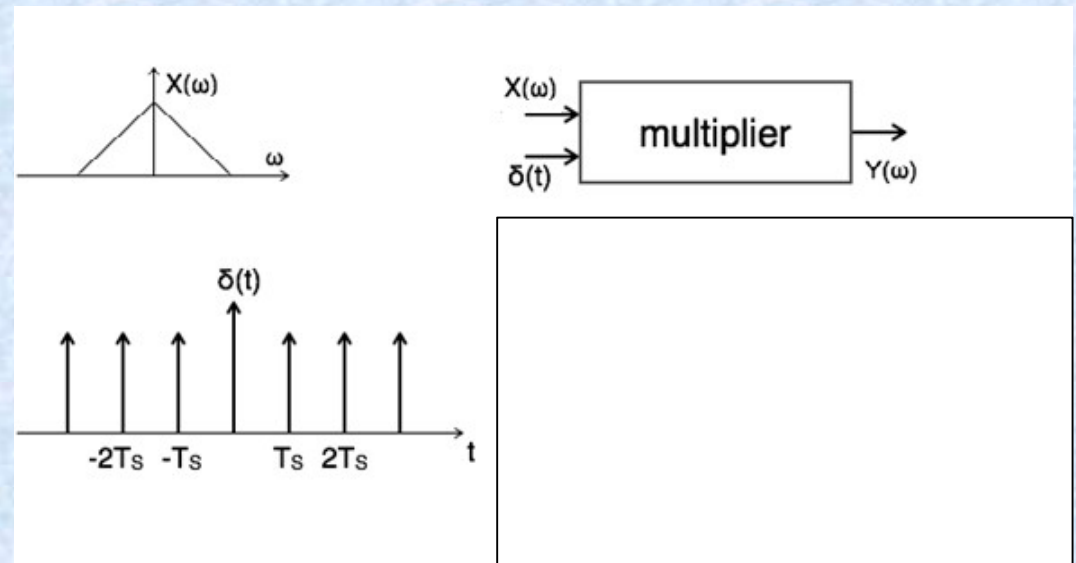
Take an example of colour information stored in an image.

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 $x(\omega)$ can be sampled by multiplying it with an impulse train $\delta(t)$
- When recovered this results in a sampled spectrum $Y(\omega)$
- Once you have as set of sample values (numbers) you can perform arithmetical operations on them



The Nyquist limit

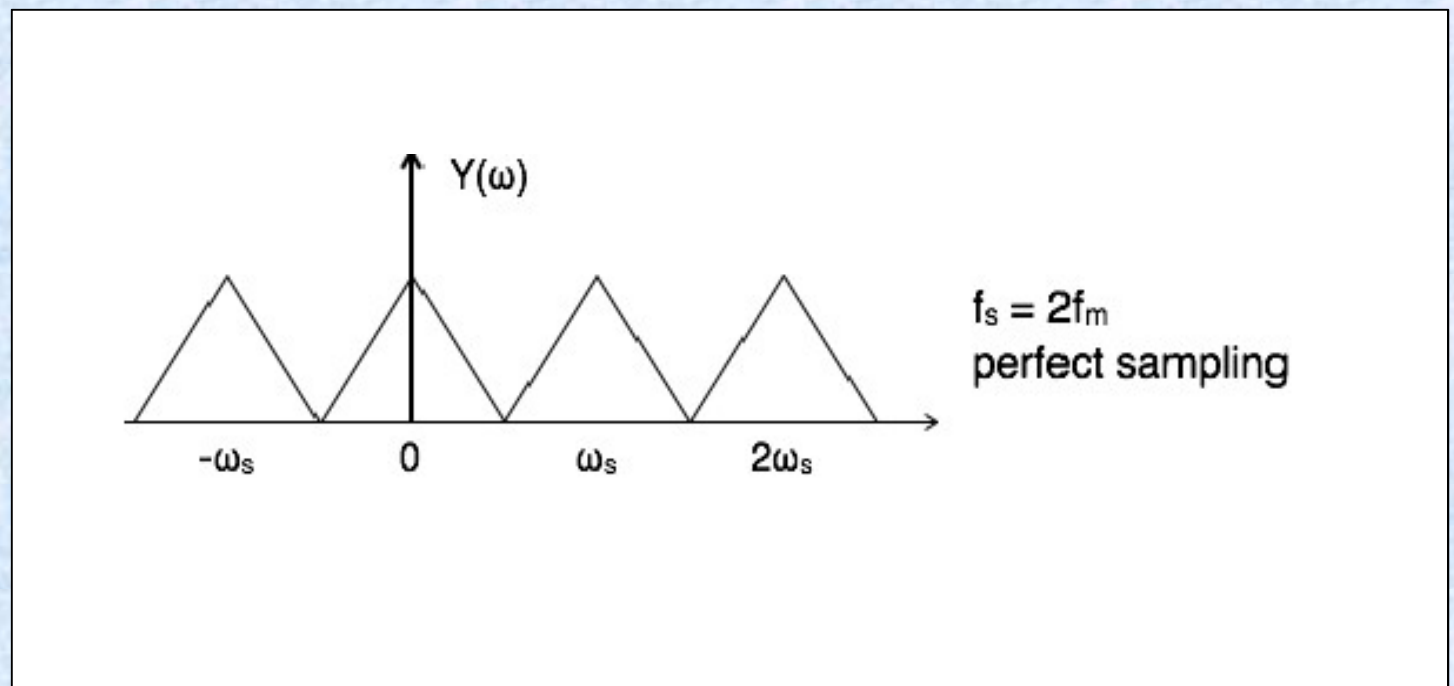
Nyquist's Sampling theorem

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

$$f_s \geq 2BW$$

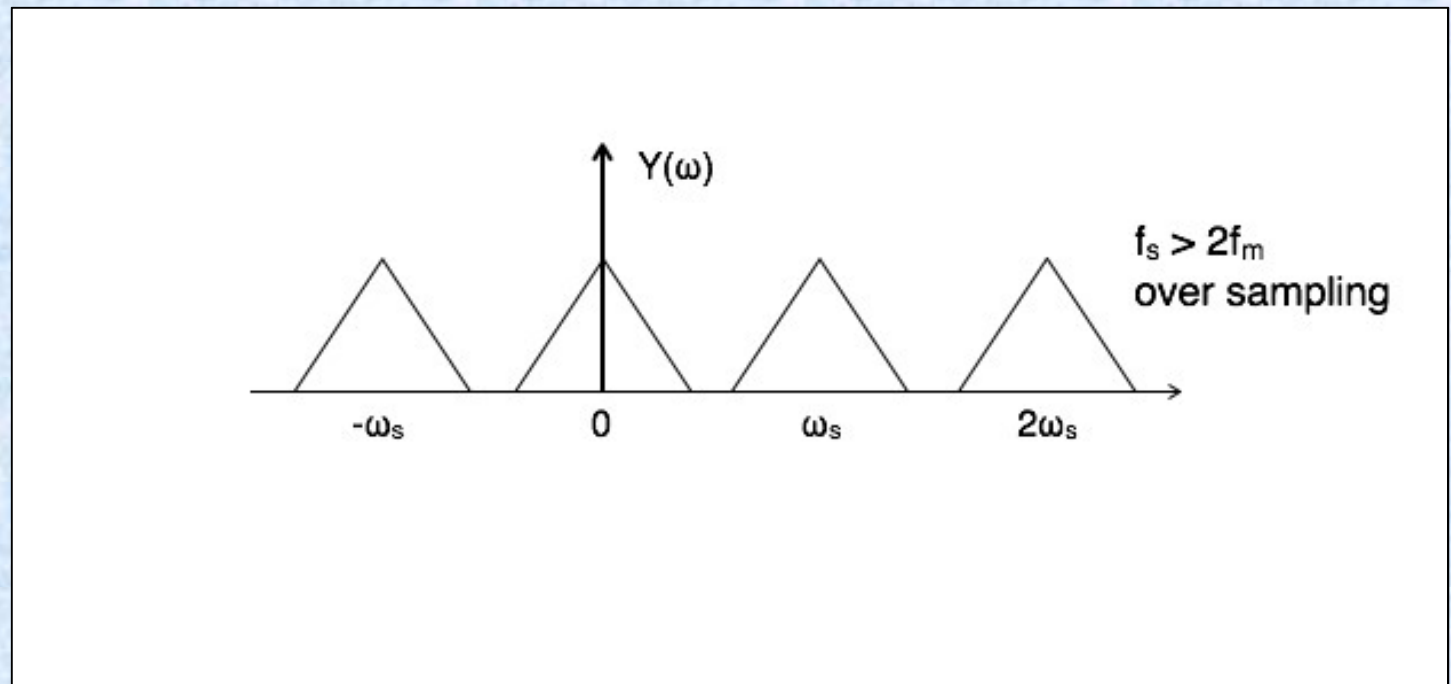
Over and under sampling

- Perfect sampling



Over and under sampling

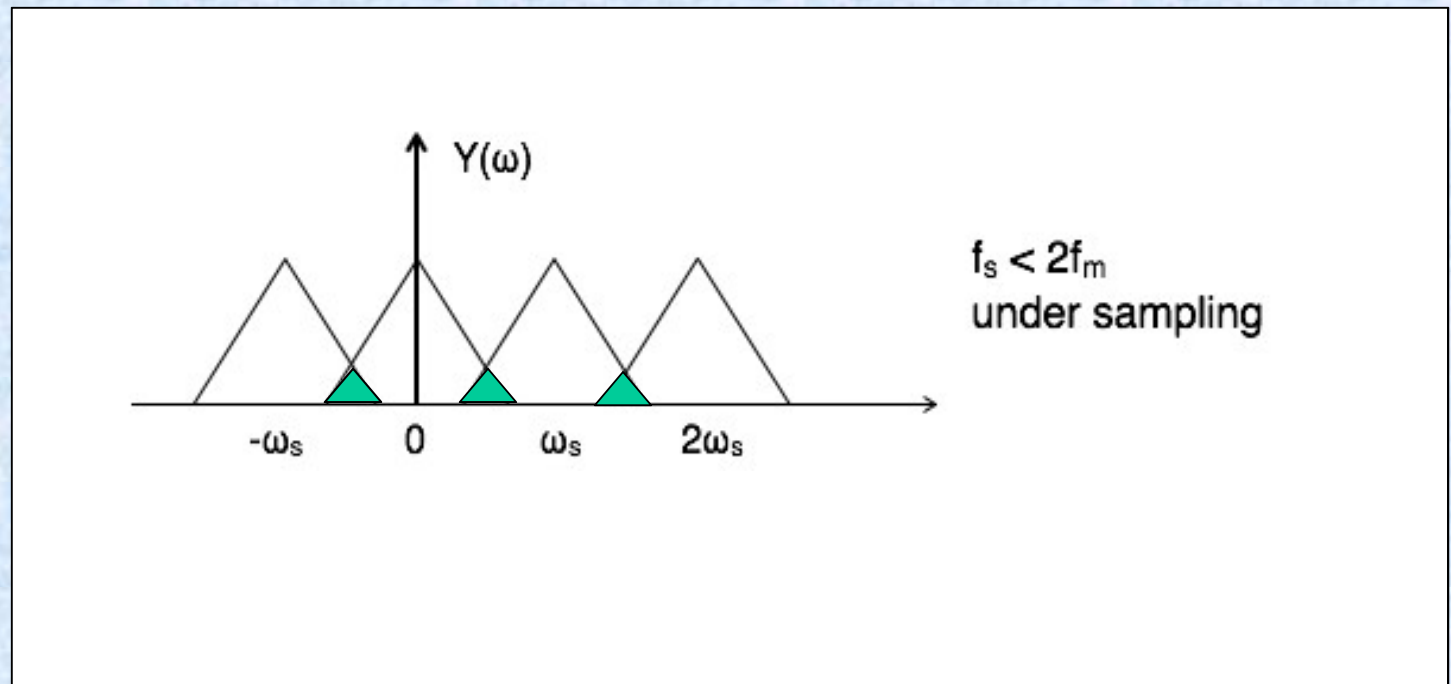
- Over sampling



- Original signal recovered normally

Over and under sampling

- Under sampling



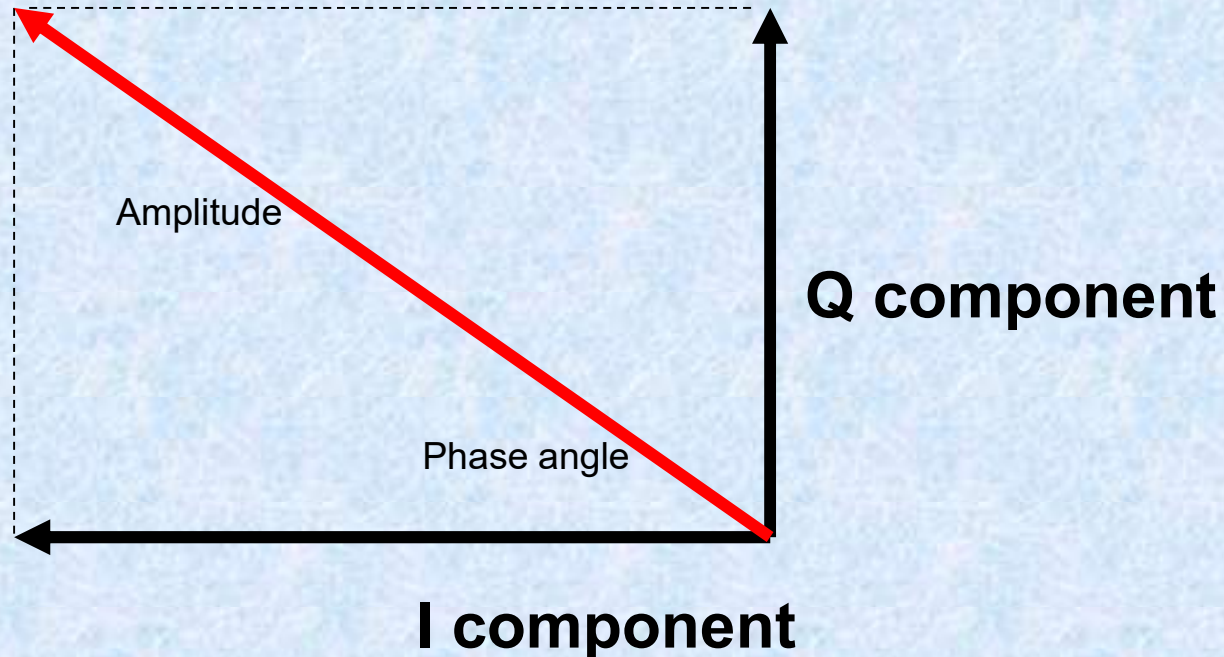
- Causes “aliasing” or “frequency foldback” distortion on the recovered signal

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.

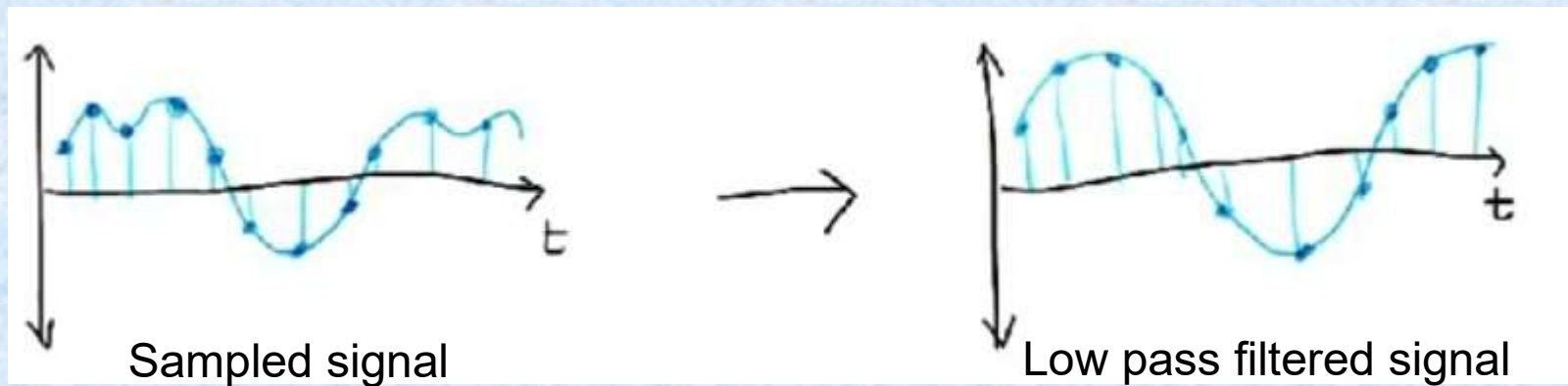
Real and floating point numbers

- 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”.

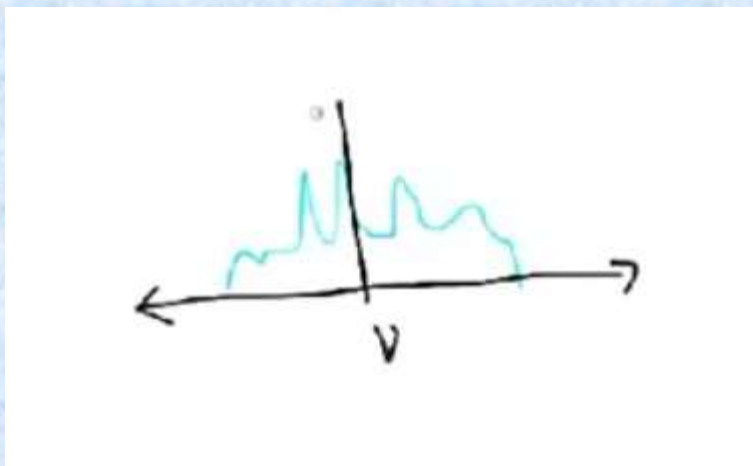
Filtering basics

Short intro to FIR filters (Finite impulse response filters)

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



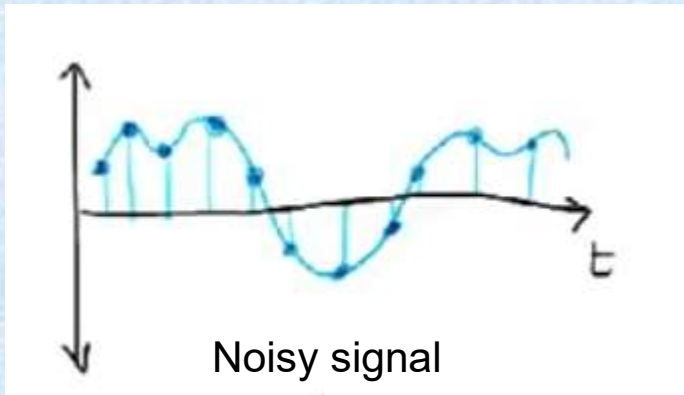
Take the FFT of the signal to get a spectrum



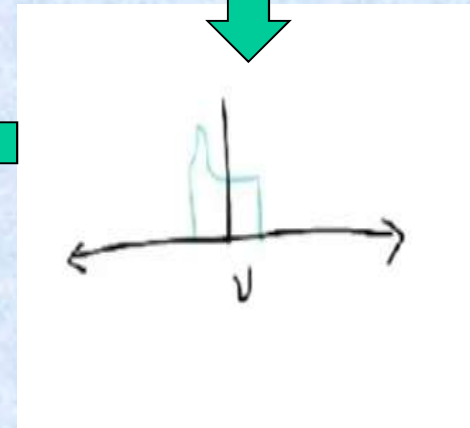
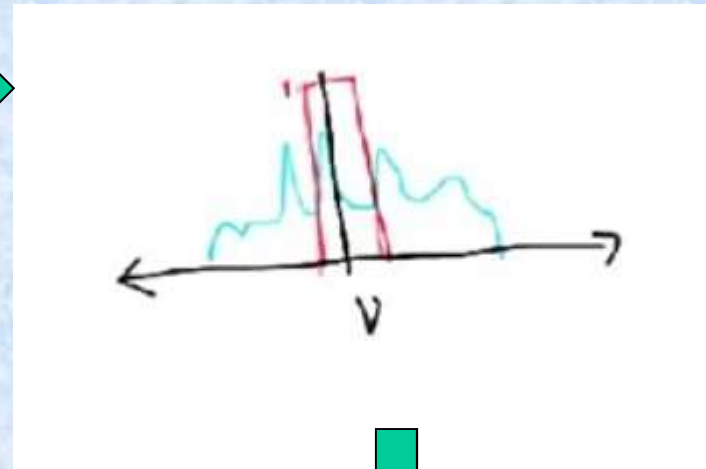
Filtering basics

Multiply it by the FFT of the desired filter response

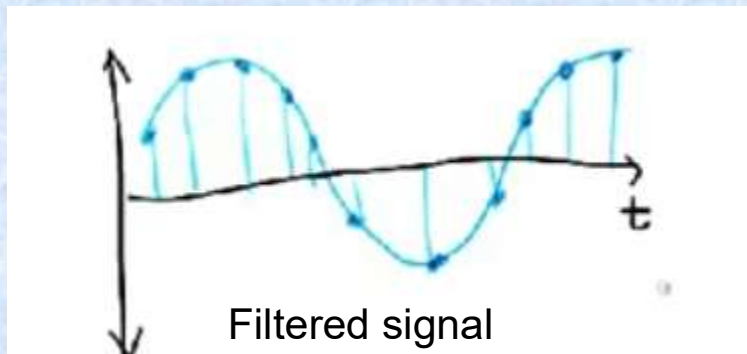
Leaving just the filtered signal



FFT and
Multiply by filter



Inverse FFT



Filtering basics

See ssb.grc

And

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

Filtering basics

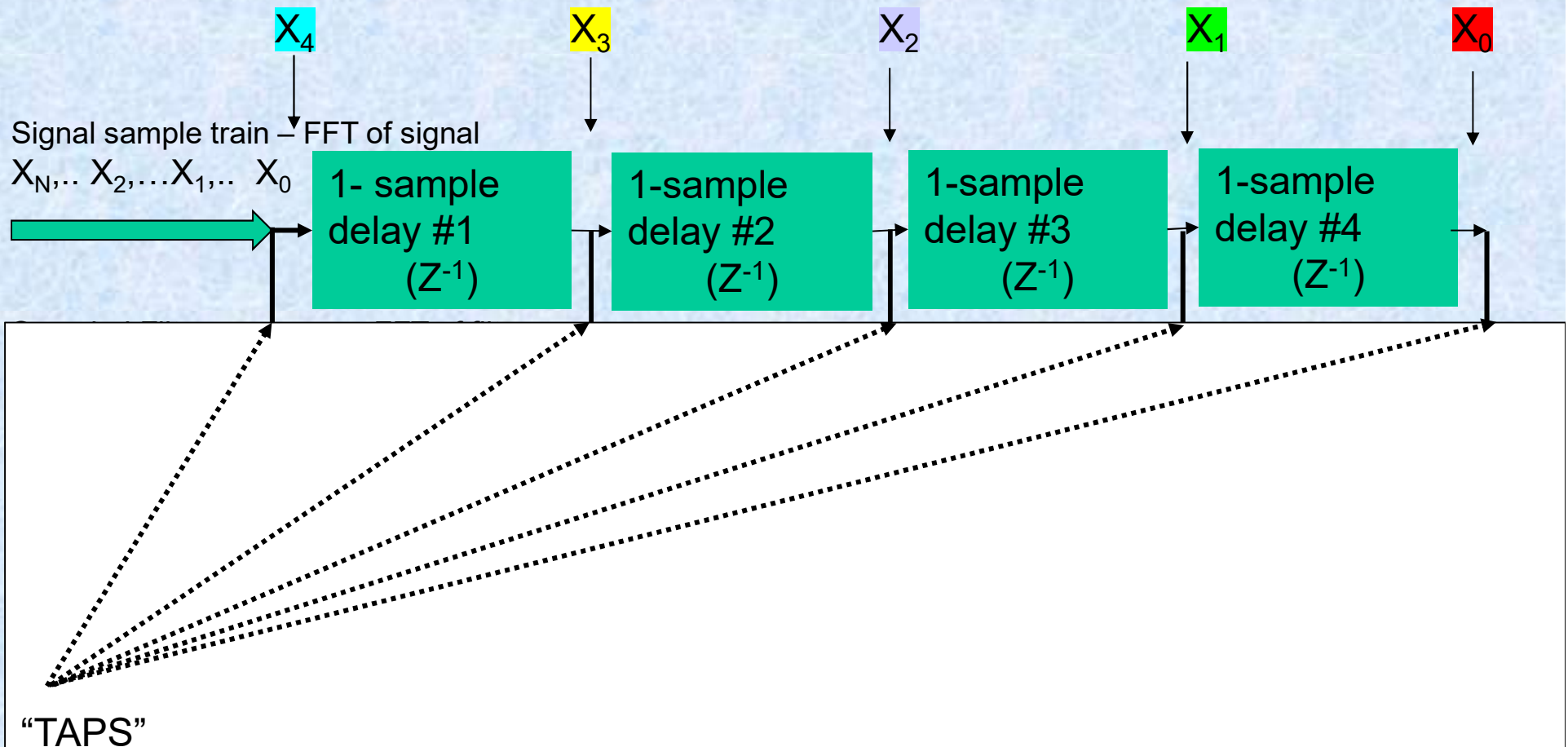
Convolution is a mathematical operation on two functions to produce a third function that expresses how the shape of one is modified by the other.

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

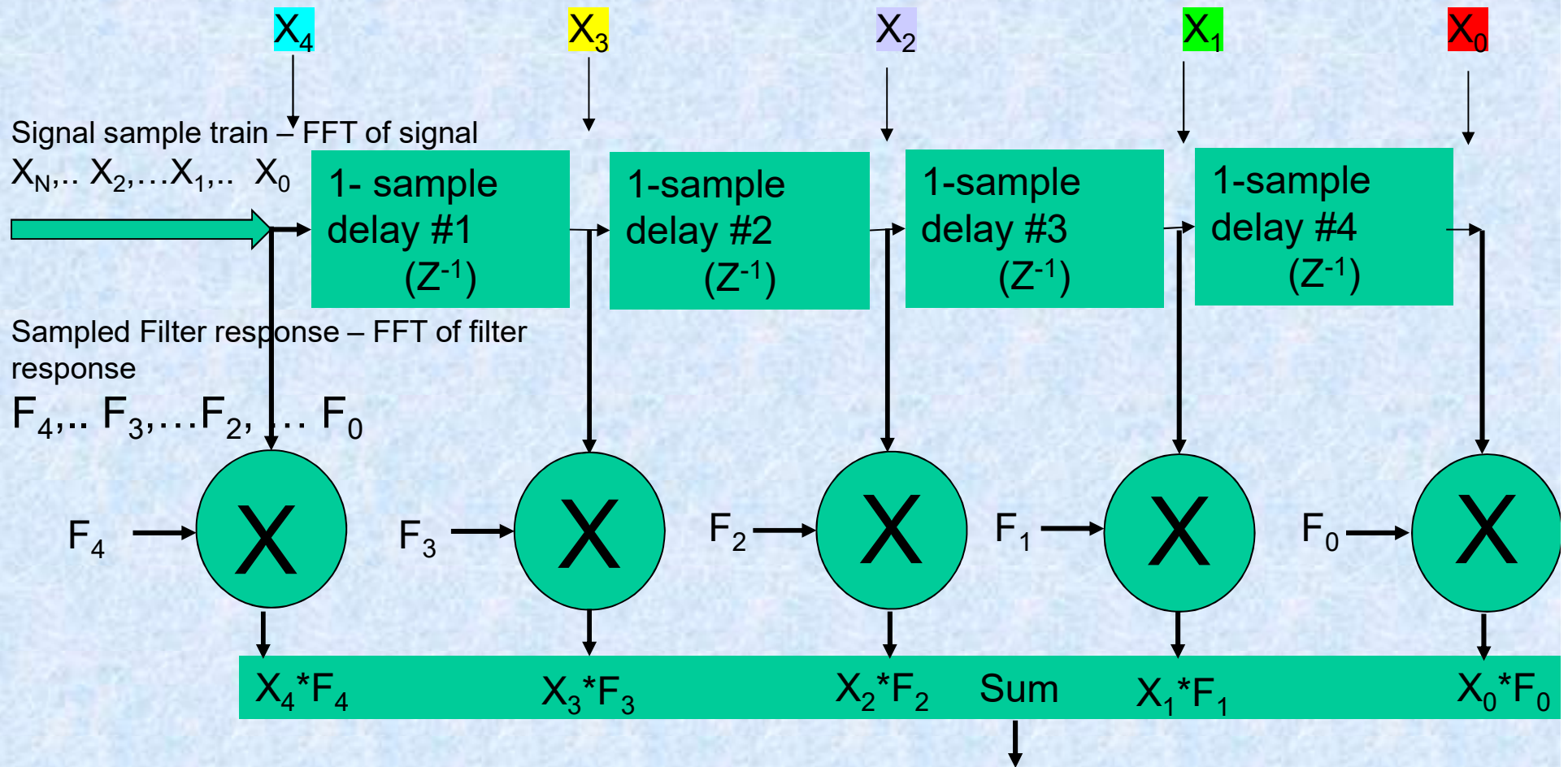
Filtering basics

To implement the convolution on a sampled signal



Filtering basics

To implement the convolution on a sampled signal



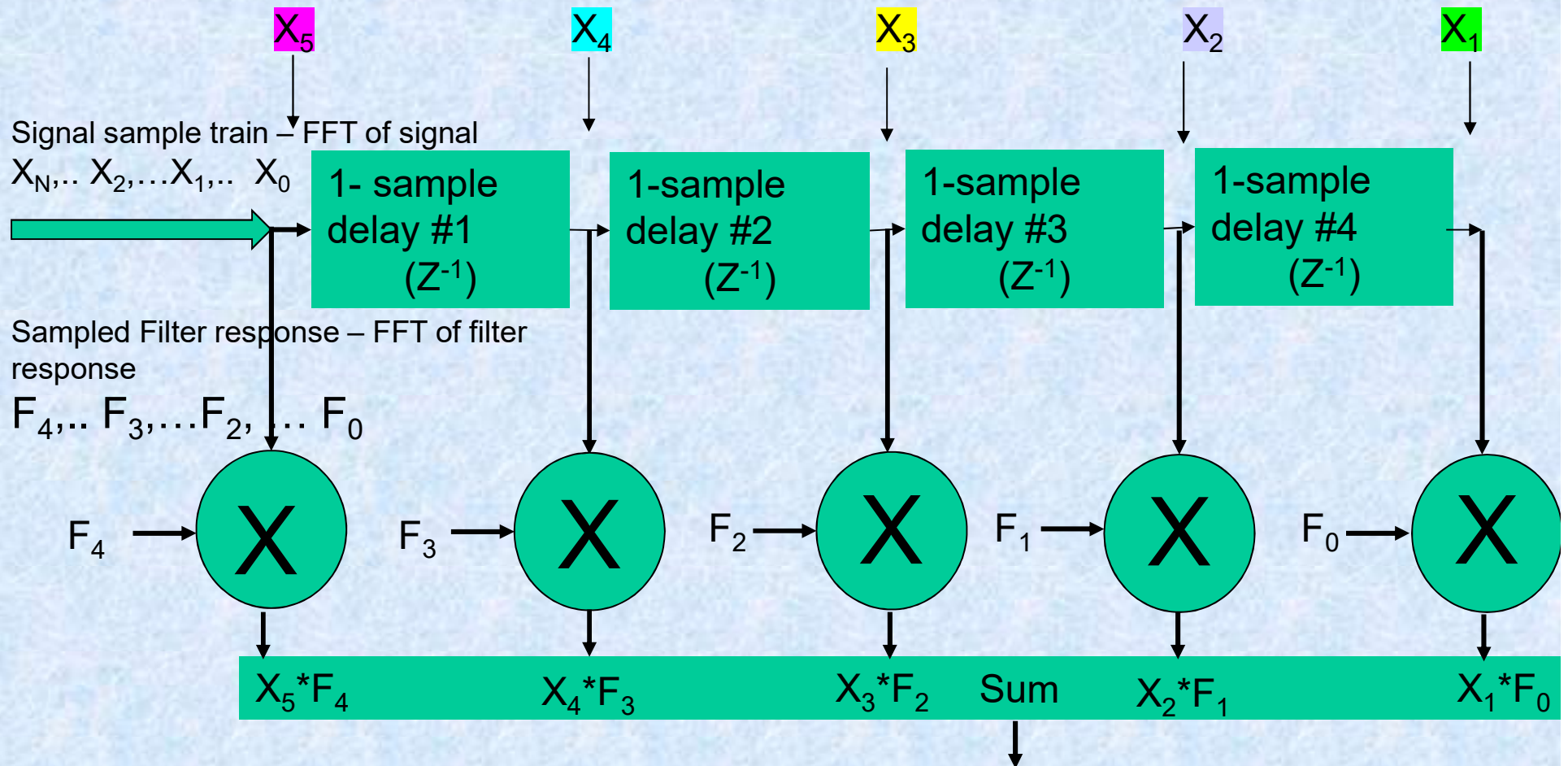
Time index 1

Sample 1 of FFT of filtered signal

Bravo Alpha Oscar 2018

Filtering basics

To implement the convolution on a sampled signal



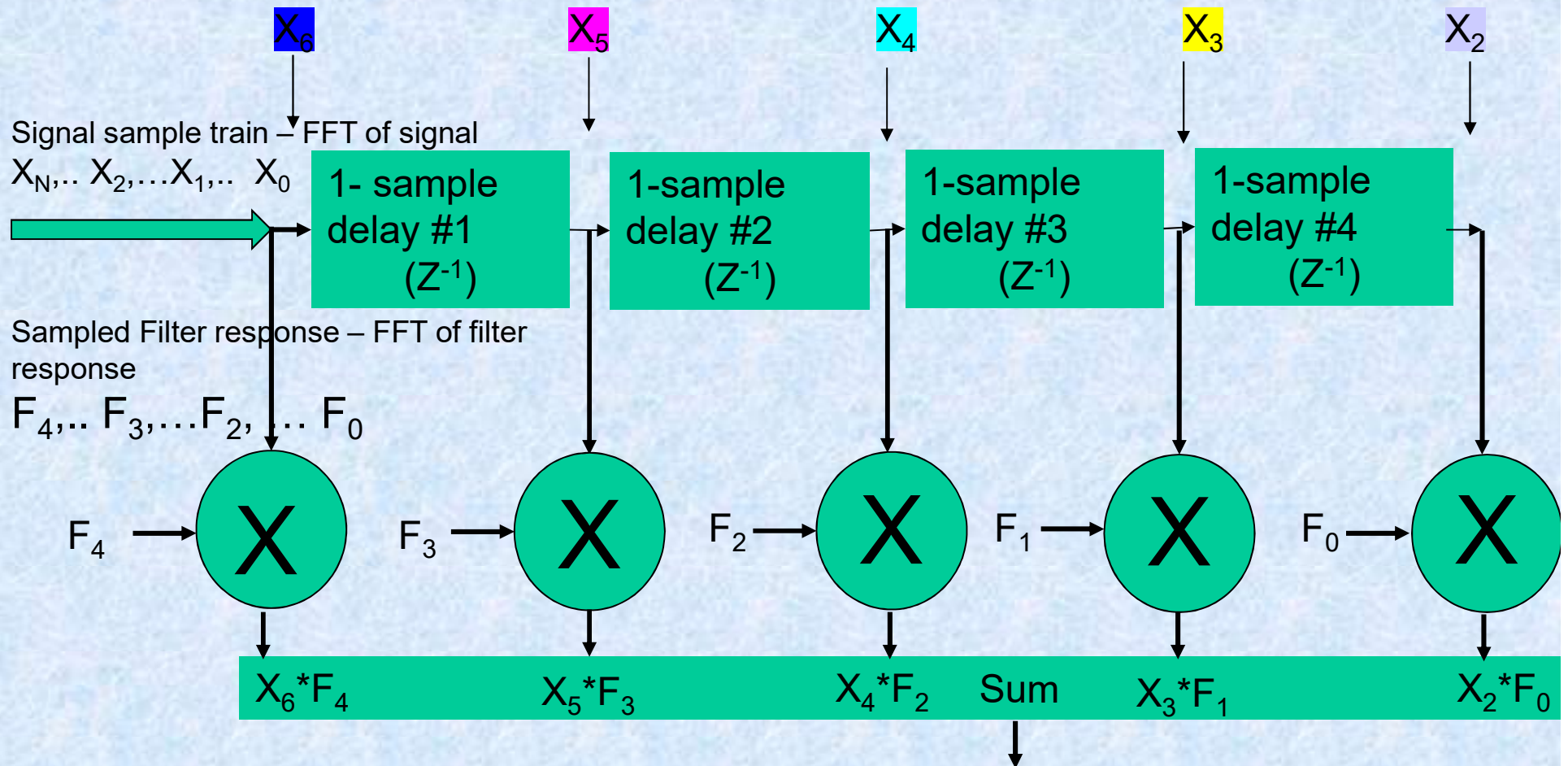
Time index 2

Sample 2 of FFT of filtered signal

Bravo Alpha Oscar 2018

Filtering basics

To implement the convolution on a sampled signal



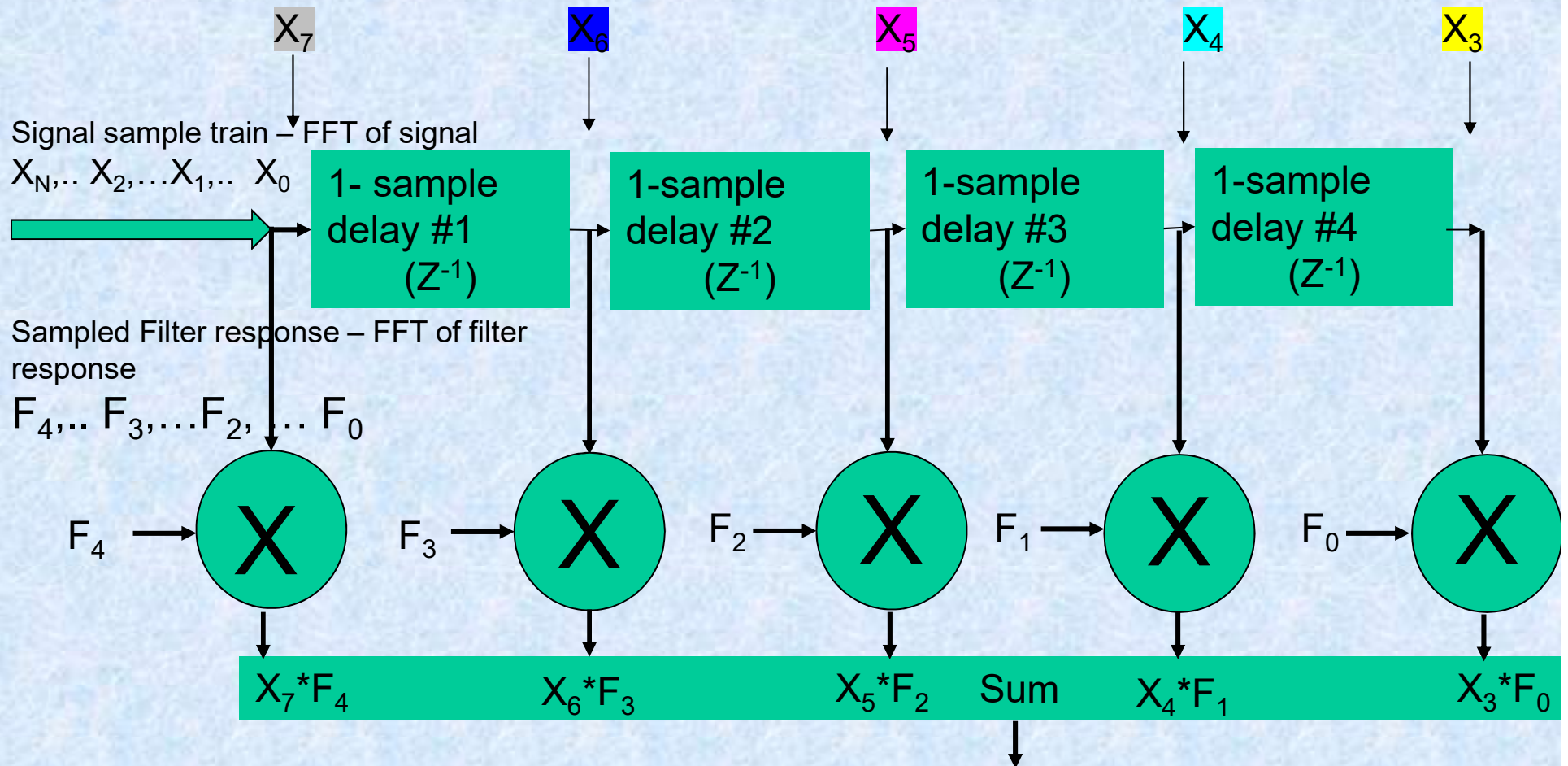
Time index 3

Sample 3 of FFT of filtered signal

Bravo Alpha Oscar 2018

Filtering basics

To implement the convolution on a sampled signal



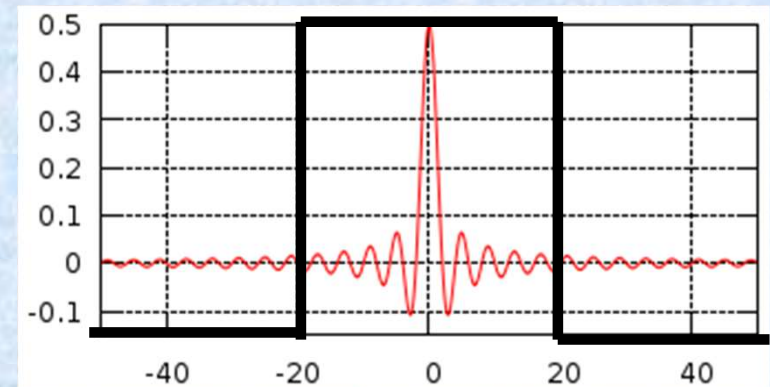
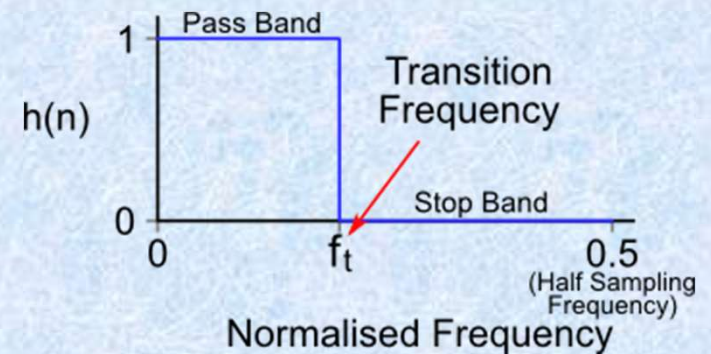
Time index 4

Sample 4 of FFT of filtered signal

Bravo Alpha Oscar 2018

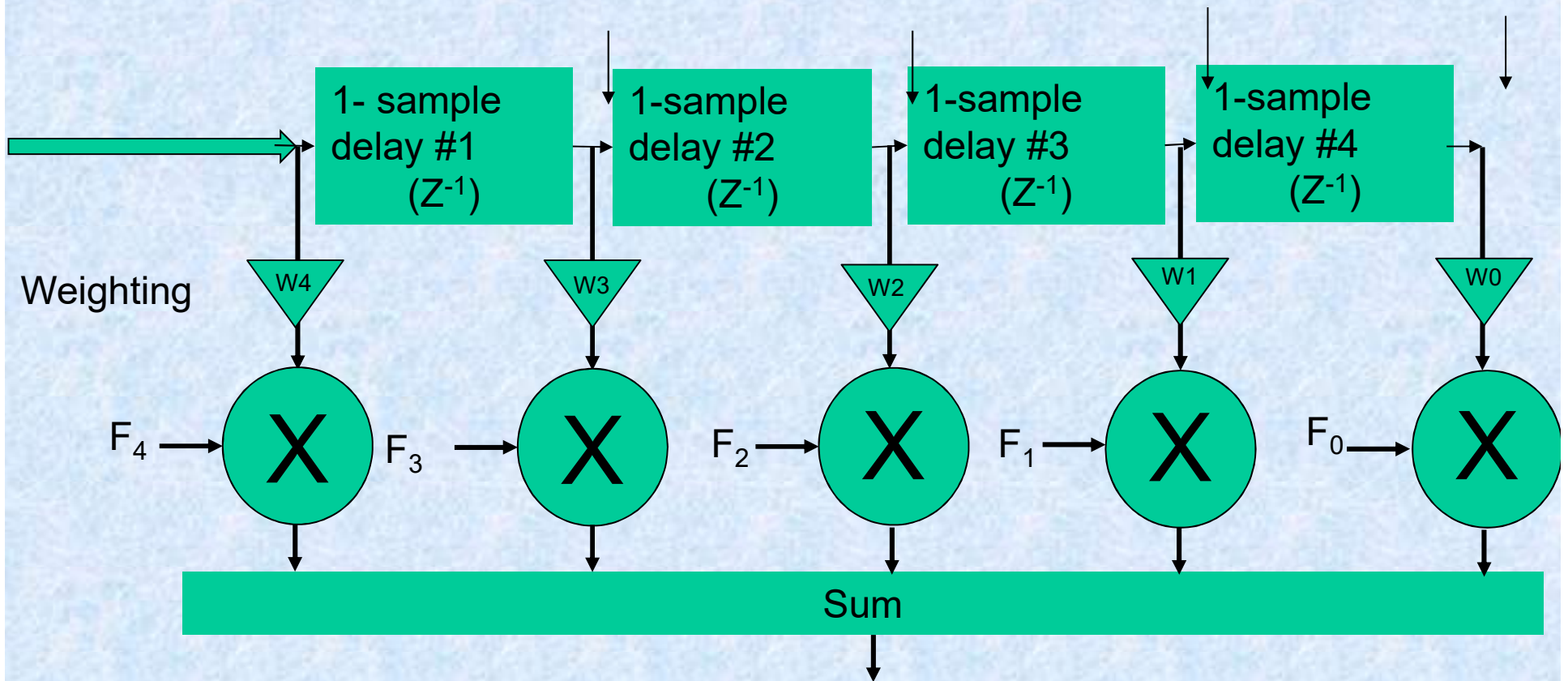
Windowing

- 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 sinc/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 sinc function beyond + and - 20



Windowing

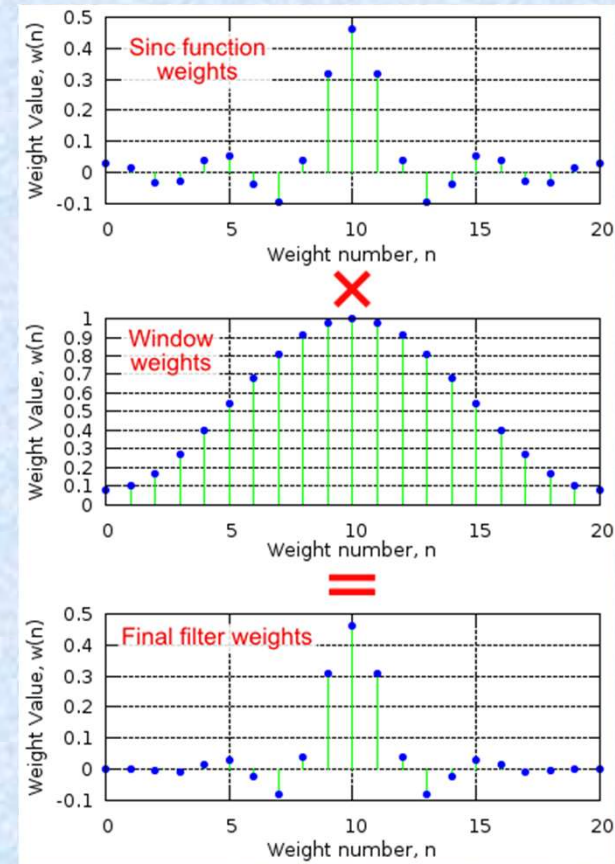
Each sample is “weighted by the Window function



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

“**Decimation**” 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 $(\text{input rate}/\text{output rate})=M$.

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

The screenshot displays the GNU Radio Companion interface for a project named 'ssb_voice.grc'. The main workspace shows a signal flow graph with the following components and connections:

- Signal Source**: Sample Rate: 320k, Waveform: Cosine, Frequency: 100k, Amplitude: 1, Offset: 0.
- Variable**: ID: samp_rate, Value: 320k.
- Audio Source**: Sample Rate: 48KHz.
- Multiply**: Receives input from both the Signal Source and the Audio Source.
- Band Pass Filter**: Decimation: 1, Gain: 1, Sample Rate: 320k, Low Cutoff Freq: 105k, High Cutoff Freq: 120k, Transition Width: 100, Window: Hamming, Beta: 6.76.
- WX GUI Scope Sink**: Title: Scope Plot, Sample Rate: 320k, Trigger Mode: Auto, Y Axis Label: Counts.
- WX GUI FFT Sink**: Title: FFT Plot, Sample Rate: 320k, Baseband Freq: 0, Y per Div: 10 dB, Y Divs: 10, Ref Level (dB): 0, Ref Scale (p2p): 2, FFT Size: 1.024k, Refresh Rate: 15, Freq Set Varname: None.

Red arrows indicate the following connections:

- From the **Multiply** block to the **Math Operators** category in the component palette.
- From the **Band Pass Filter** block to the **Filters** category.
- From the **WX GUI Scope Sink** block to the **Instrumentation** category.
- From the **WX GUI FFT Sink** block to the **Instrumentation** category.
- From the **Signal Source** block to the **Waveform generators** category.

The component palette on the right lists various categories such as Core, Audio, Boolean Operators, Byte Operators, Channel Models, Channelizers, Coding, Control Port, Debug Tools, Deprecated, Digital Television, Equalizers, Error Coding, FCD, File Operators, Filters, Fourier Analysis, GUI Widgets, Impairment Models, Instrumentation, Level Controllers, Math Operators, Measurement Tools, Message Tools, Misc, Modulators, Networking Tools, NOAA, OFDM, Packet Operators, Pager, Peak Detectors, Resamplers, Stream Operators, Stream Tag Tools, Symbol Coding, and Synchronizers.

The console at the bottom shows the following output:

```
Loading: "F:\GNU\ssb_voice.grc"
>>> Done

Loading: "F:\GNU\ssb.grc"
>>> Done

Generating: 'F:\GNU\top_block.py'

Executing: C:\Python27\python.exe -u F:\GNU\top_block.py

INFO: Audio source arch: windows
gr::pagesize: no info; setting pagesize = 4096
```

Id	Value
Imports	
Variables	
samp_rate	320000

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!



Bravo Alpha Oscar 2018