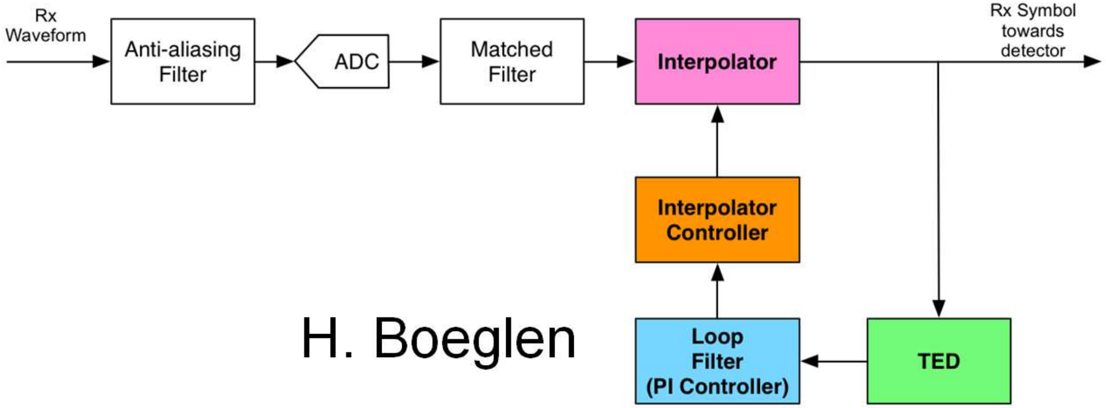
From a simulated to a real digital communication system: a QPSK modem design with GNU Radio



H. Boeglen

XLIM, UMR CNRS 6172, Université de Poitiers, France

European GNU Radio Days 2023, Paris, France

Outline

1 – Introduction and motivation

2 – QPSK modem: the main building blocks

3 – The fully digital QPSK synchronizer.

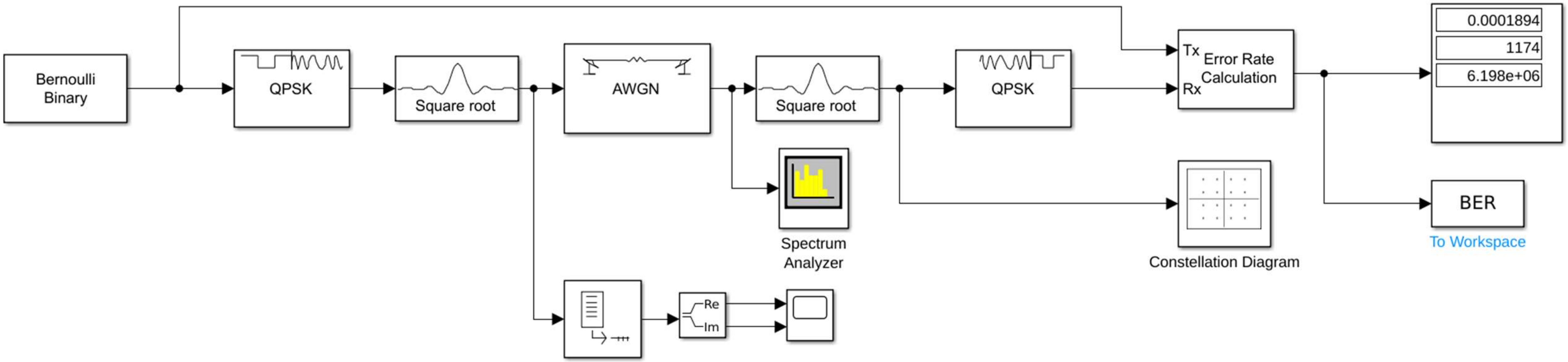
4 – GNU Radio demo with real hardware

5 – Conclusion

Outline

- ❑ **I have been teaching digital communications (DC) at undergraduate and graduate levels for more than 15 years.**

- ❑ **Extensive use of Matlab/Simulink and the Communications toolbox:**



➔Very good tool to illustrate the main DC concepts by simulation.

- **Up to this year in the 2nd year of the IoT master's degree two disjoint topics teached in the DC course:**

  - ➔ **Basics + ECC (me)**
  - ➔ **Hardware aspects: linearity in RF amplifiers (my collegue)**

- **Everything is teached using Matlab/Simulink simulation tools.**

- **My collegue has discovered the power of GNU Radio in a DPD (digital pre-distorsion) research project.**

- **We have decided to give more coherence to the course by:**

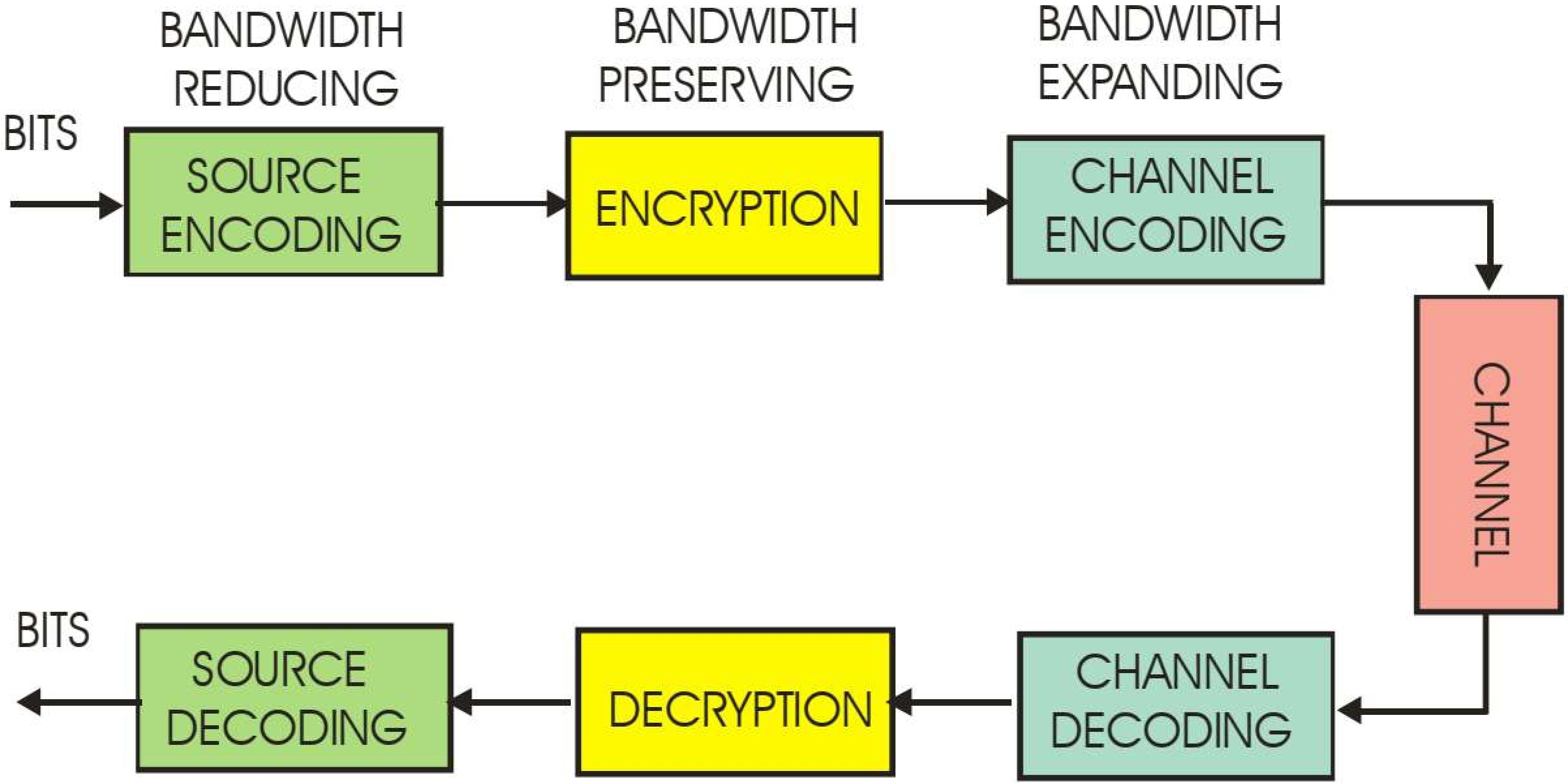
➔ **Replacing Matlab/Simulink by GNU Radio as much as possible (at last; I do not want to have trouble with JM ☺).**

➔ **Giving more space to implementation with the design of a real world QPSK modem (GNU Radio + hardware (Adalm-Pluto) up to the linearized PA).**

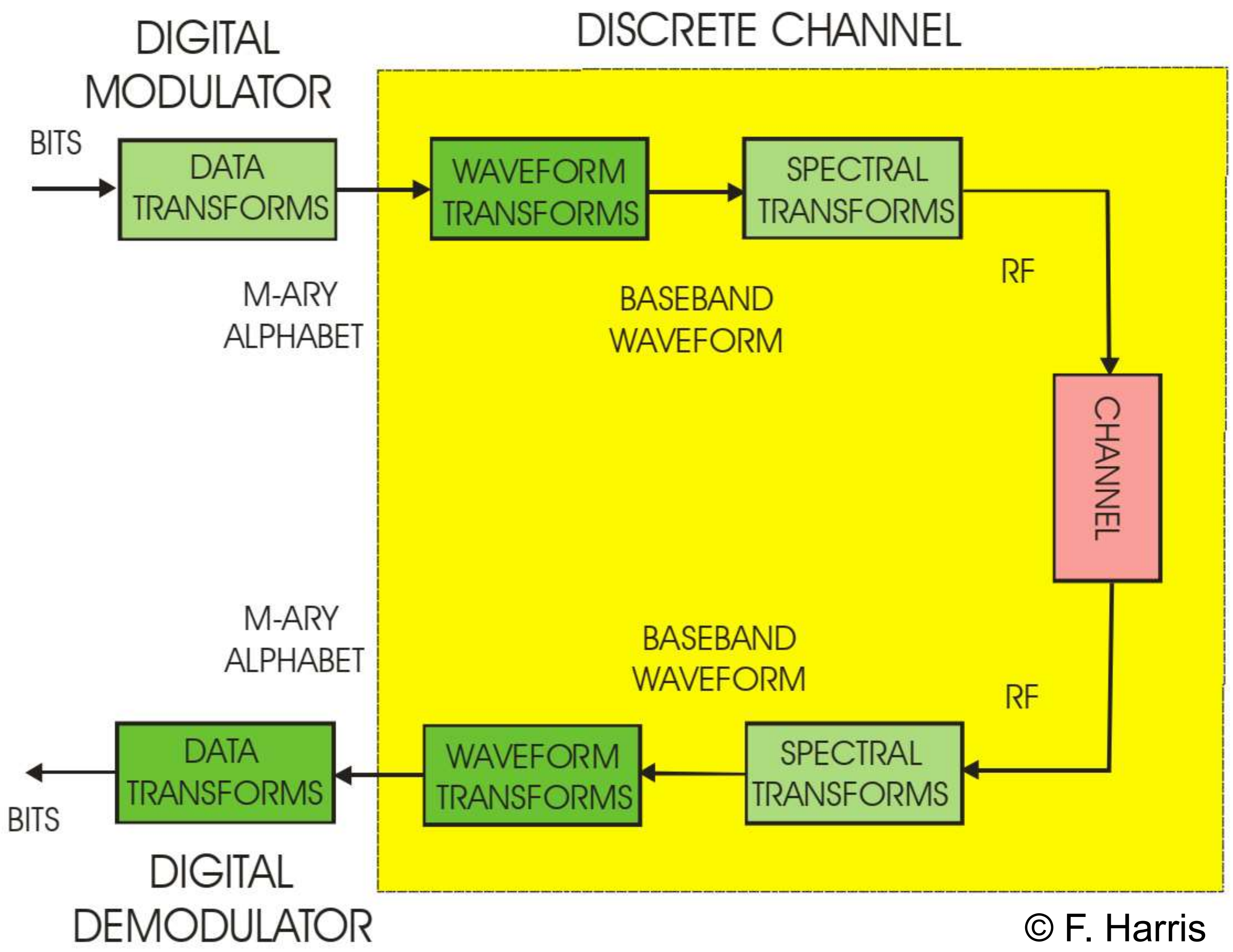
- **So this presentation will be about the difficulties you'll face when you go from simulation to implementation!**
- **Since it is the topic of these 2 days we will focus mainly on synchronization aspects.**
- **Well, no choice: no synchronization means no real world implementation anyway.**

Outline

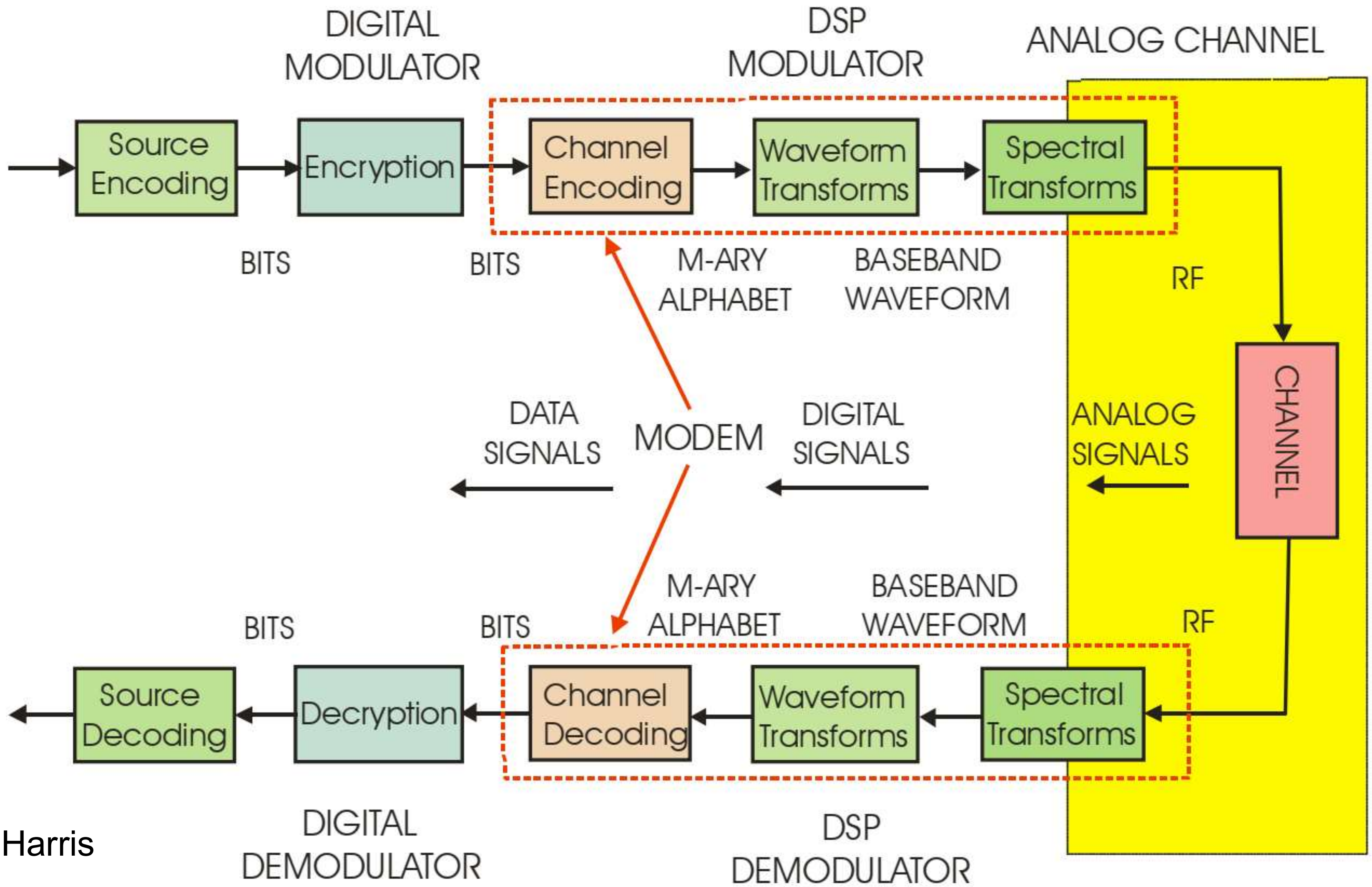
❑ From Shannon to the digital QPSK modem



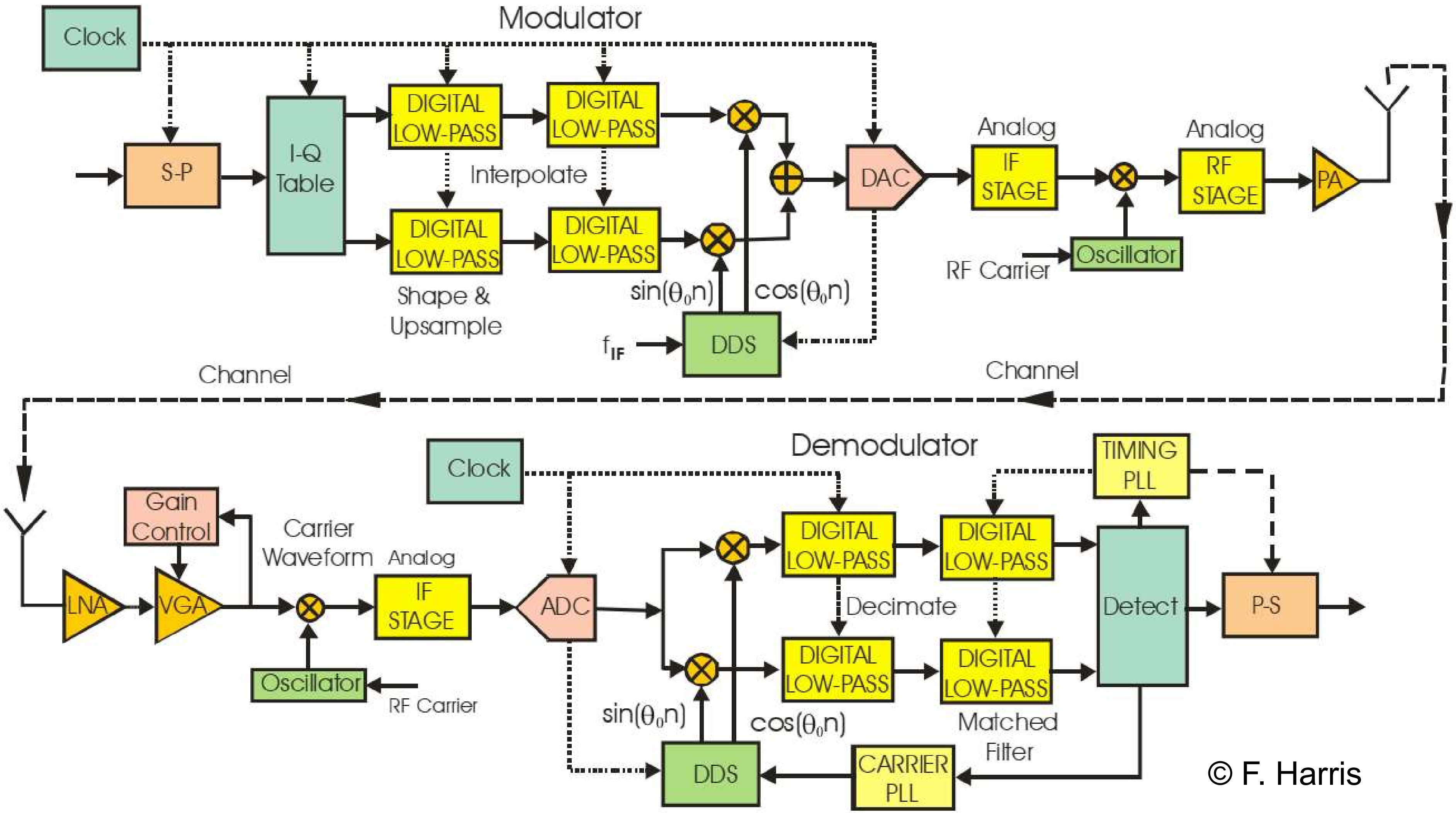
❑ From Shannon to the digital QPSK modem



❑ From Shannon to the digital QPSK modem



❑ From Shannon to the digital QPSK modem



- **Pulse shaping and matched filtering needed for:**

→ Limiting the BW
→ Maximizing SNR at the decision points
→ Reduce ISI

- **Use of the well known Root Raised Cosine (RRC) pulse**

➔ One at the transmitter and one at the receiver = Raised Cosine response, i.e. Nyquist ISI criterion.

- **In practice: what are the important parameters?**

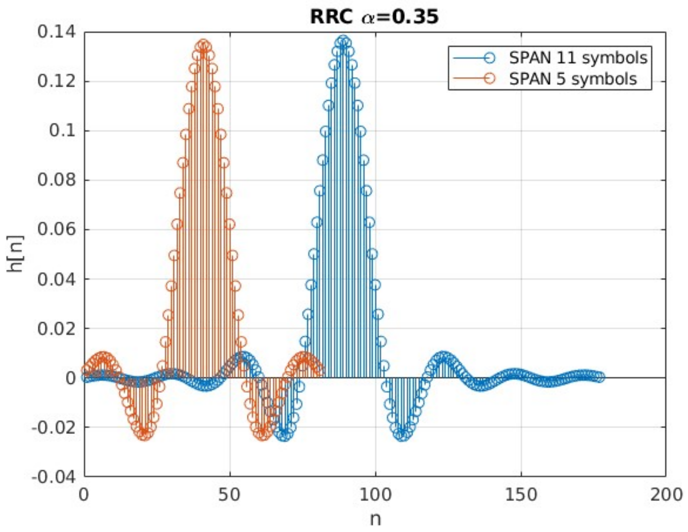
➔ Upsampling factor N (SPS): (4 to 16)
➔ Filter span (number of symbols).
➔ Roll-off α (excess bandwith/cardinal sine): between 0.25 and 0.6.

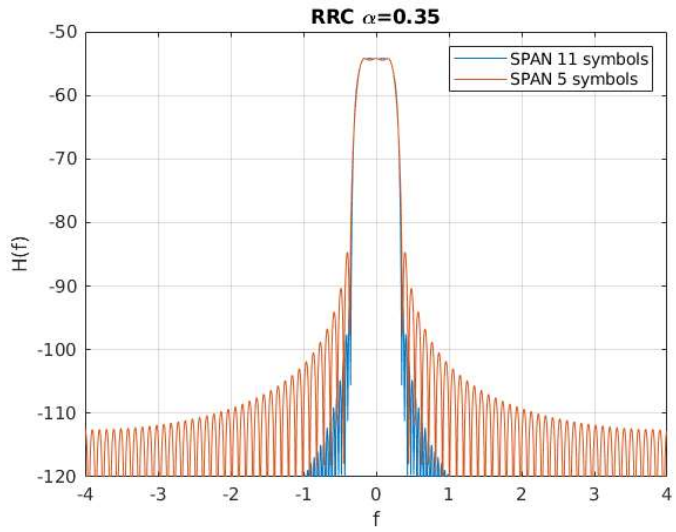
❑ RRC pulse filter coefficients generation

➔ Matlab h = rcosdesign(α, SPAN, SPS)
➔ Octave h = rcosfir(α,nT,SPS,T,’sqrt’)
➔ GNU Radio firdes.root_raised_cosine(gain, SPS,T, α,ntaps (SPAN*SPS))

- RRC pulse filter example (α = 0.35, SPS = 16, SPAN = 11)

firdes.root_raised_cosine(2, 2*8, 1.0, 0.35, (11 or 5)*8*2)





❑ **How many samples per symbol (N)?**

N = Ts/T with Ts = sample rate and T = symbol rate

In practice: $4 \leq N \leq 16$

❑ **Be careful about the signal level at the reception side!**

- **At baseband: system with symbol of unit energy.**
- **Amplitude A is then given by:**

→ A = sqrt((3*Es)/(2*(M-1)) with Es = symbol energy in J and M = number of points of the constellation.

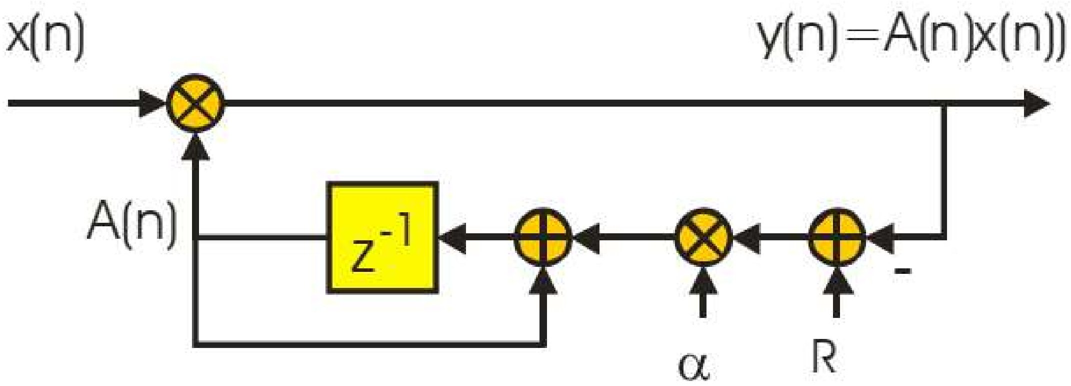
→ For QPSK A = sqrt(2.0) = 0.707.

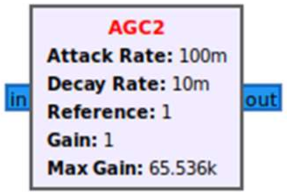
Receiver functions designed to handle this value which has to remain constant → need for an AGC

❑ Signal level at the reception side: AGC

- **Need to ensure a constant signal level for the receiver function to work**
- **Signal cannot be constant because propagation channel varies**

➔Use of an Automatic Gain Control (AGC) structure





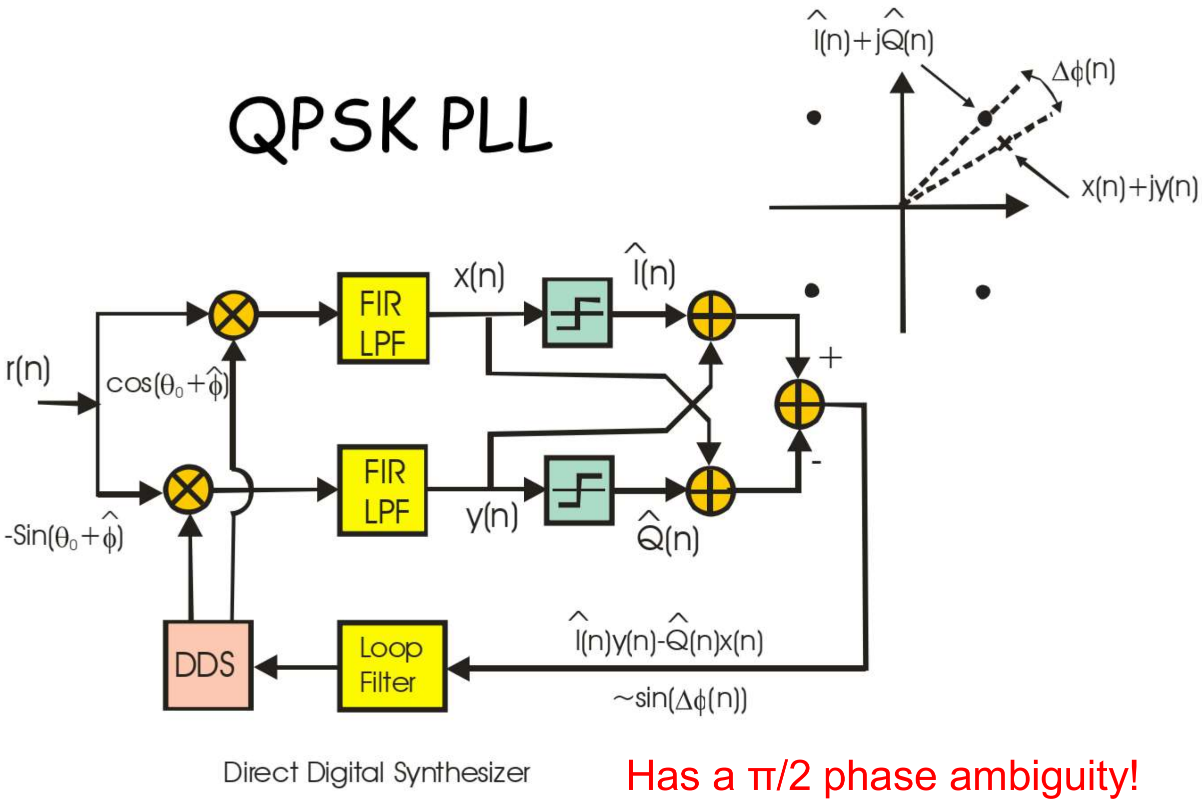
$A[n+1] = A[n](1-\alpha c) + \alpha R$
R = reference, α = gain c = constant and $\alpha c < 2.0$.
Loop time constant is $1/\alpha c$ samples.
If c small long transient. If c large short transient.

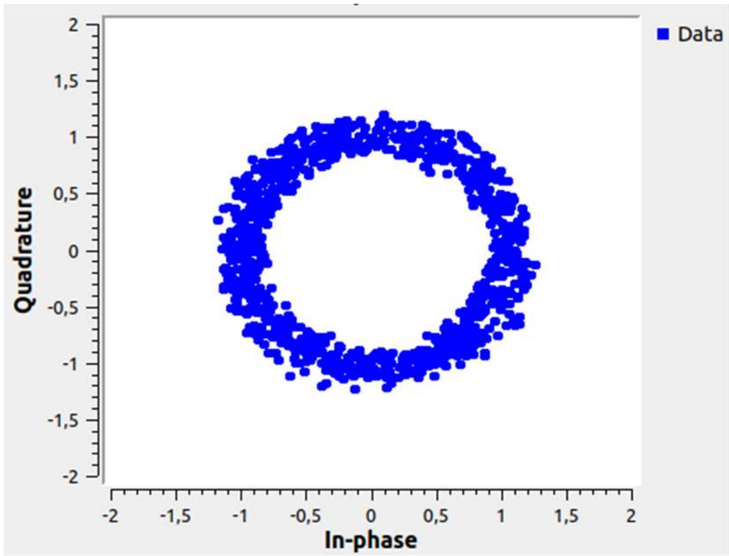
❑ **Some more loops: carrier and timing synchronization PLLs**

- **Carrier and phase recovery loop**

➔ TX oscillator unknown phase

➔ Generated carrier frequency f_{Rx} at receiver is not exactly f_{Tx}

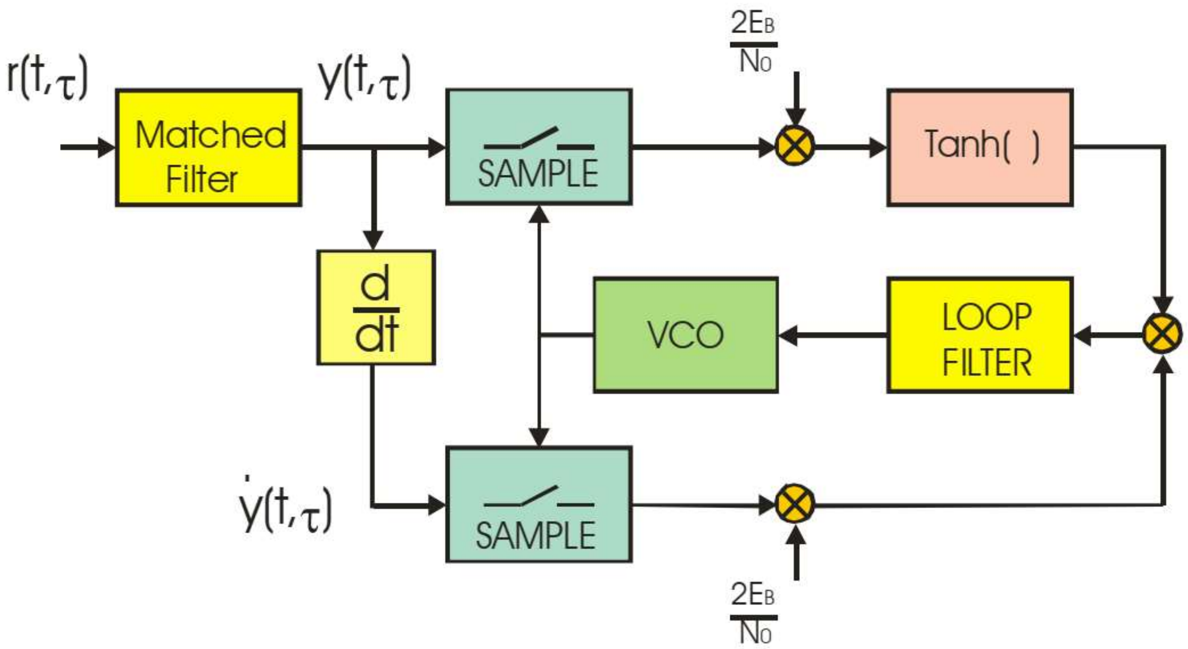


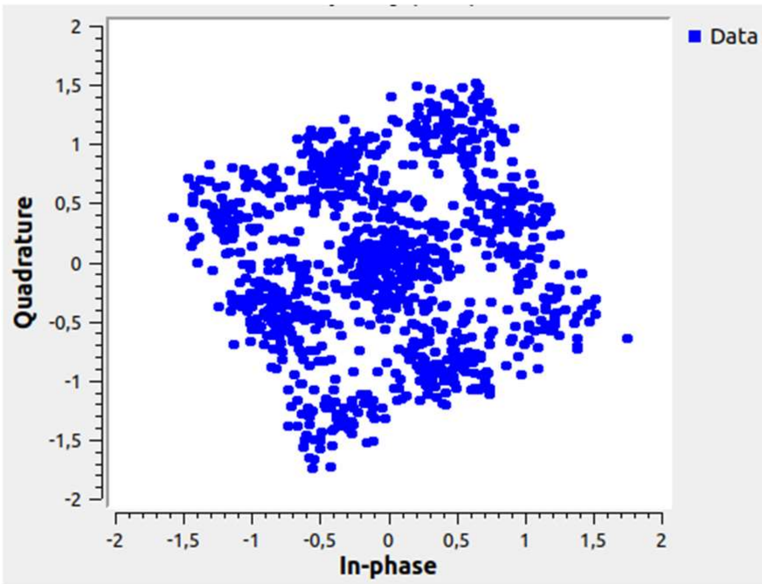


Has a π/2 phase ambiguity!

- **Some more loops: carrier and timing synchronization PLLs**
- **Timing recovery loop**

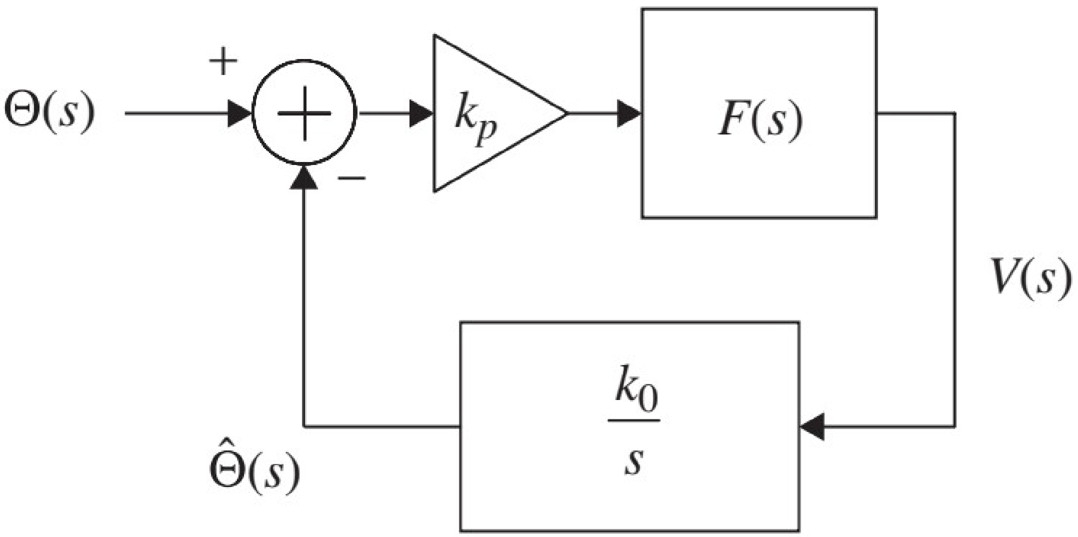
➔Optimum sampling instants have to be extracted from the data





❑ From analog PLL to digital PLL and its main parameters

- We will restrict to a second order loop (the most used).
- The analog PLL



Proportional-plus-integrator filter:

$$F(s) = k_1 + \frac{k_2}{s}$$

Transfer function:

$$H_a(s) = \frac{k_0 k_p k_1 s + k_0 k_p k_2}{s^2 + k_0 k_p k_1 s + k_0 k_p k_2}$$

Can be rewritten:

$$H_a(s) = \frac{2\varsigma\omega_n s + \omega_n^2}{s^2 + 2\varsigma\omega_n s + \omega_n^2}$$

With:

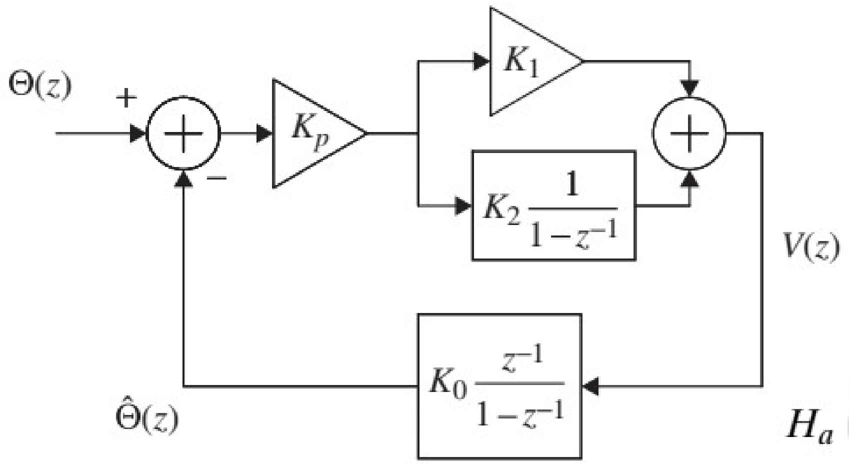
$$\varsigma = \frac{k_1}{2}\sqrt{\frac{k_0 k_p}{k_2}} \qquad \omega_n = \sqrt{k_0 k_p k_2}$$

PLL BW: $$\omega_{3dB} = \omega_n\sqrt{1 + 2\varsigma^2 + \sqrt{(1+2\varsigma^2)^2 + 1}}$$

PLL noise BW: $$B_n = \frac{\omega_n}{2}\left(\varsigma + \frac{1}{4\varsigma}\right)$$

❑ From analog PLL to digital PLL and its parameters

- **We will restrict to a second order loop (the most used).**
- **The digital PLL (apply Tustin's equation)**



$$H_d(z) = \frac{K_pK_0(K_1+K_2)z^{-1} - K_pK_0K_1z^{-2}}{1 - 2\left(1 - \frac{1}{2}K_pK_0(K_1+K_2)\right)z^{-1} + \left(1 - K_pK_0K_1\right)z^{-2}}$$

$$H_a\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right) = \frac{\frac{2\zeta\theta_n+\theta_n^2}{1+2\zeta\theta_n+\theta_n^2} + 2\frac{\theta_n^2-\zeta\theta_n}{1+2\zeta\theta_n+\theta_n^2}z^{-1} + \frac{\theta_n^2}{1+2\zeta\theta_n+\theta_n^2}z^{-2}}{1 - 2\frac{\theta_n^2-1}{1+2\zeta\theta_n+\theta_n^2}z^{-1} + \frac{1-2\zeta\theta_n+\theta_n^2}{1+2\zeta\theta_n+\theta_n^2}z^{-2}}$$

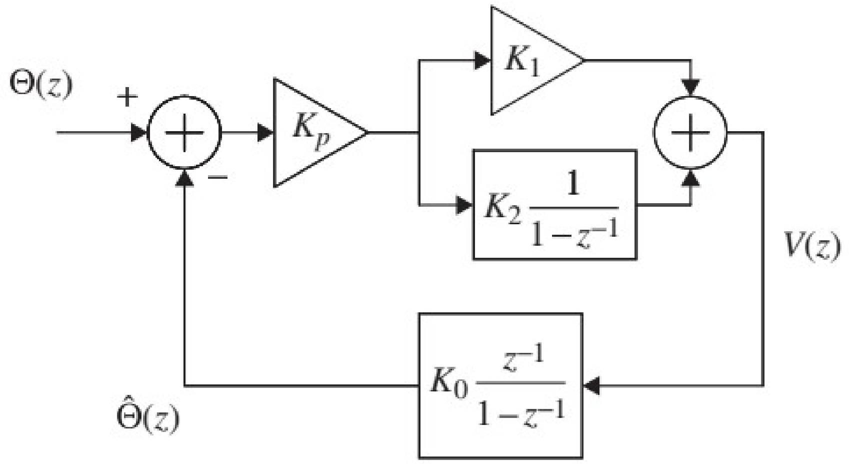
Equating the denominator polynomials in Hd and Ha gives :

$$K_pK_0K_1 = \frac{4\zeta\theta_n}{1+2\zeta\theta_n+\theta_n^2}$$

$$K_pK_0K_2 = \frac{4\theta_n^2}{1+2\zeta\theta_n+\theta_n^2}.$$

❑ From analog PLL to digital PLL and its parameters

- **We will restrict to a second order loop (the most used).**
- **The digital PLL (apply Tustin's equation)**



With N = Ts/T we finally get:

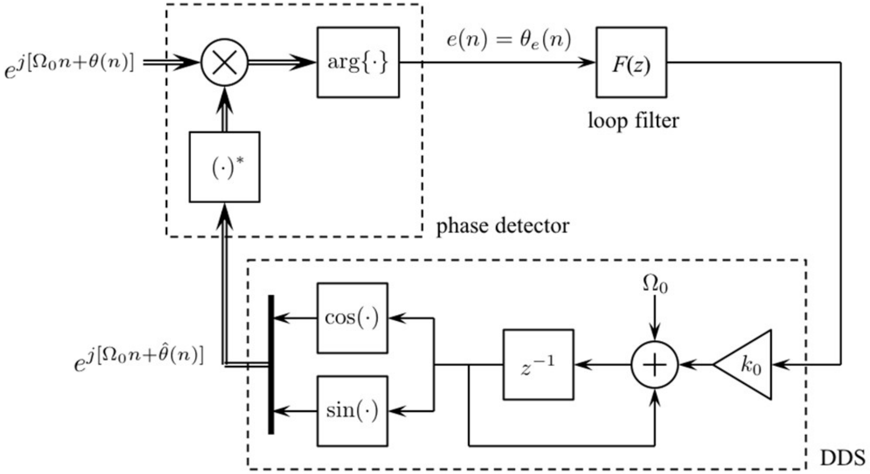
$$K_0 K_p K_1 = \frac{\frac{4\zeta}{N}\left(\frac{B_n T_s}{\zeta + \frac{1}{4\zeta}}\right)}{1 + \frac{2\zeta}{N}\left(\frac{B_n T_s}{\zeta + \frac{1}{4\zeta}}\right) + \left(\frac{B_n T_s}{N\left(\zeta + \frac{1}{4\zeta}\right)}\right)^2}$$

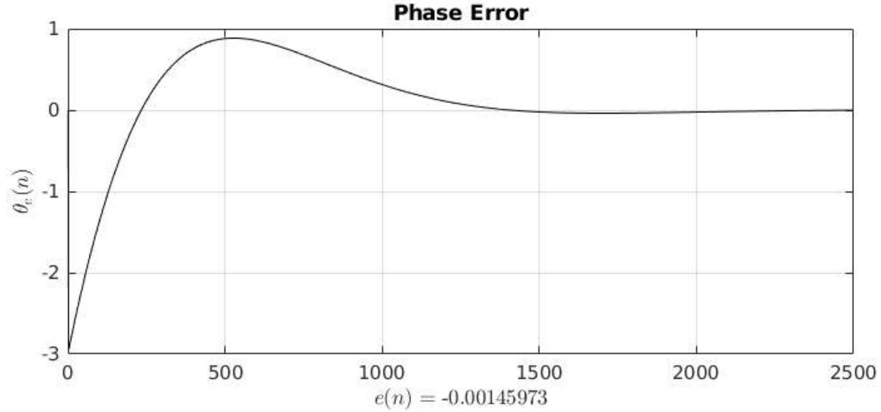
$$\theta_n = \frac{B_n T_s}{N\left(\zeta + \frac{1}{4\zeta}\right)}.$$

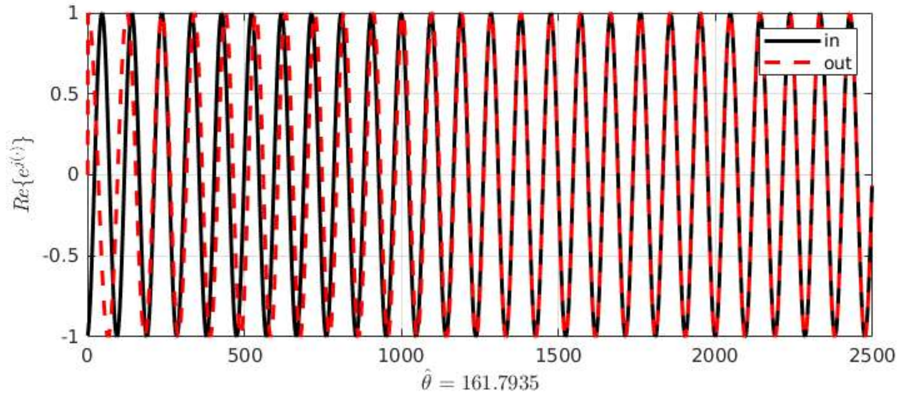
$$K_0 K_p K_2 = \frac{\frac{4}{N^2}\left(\frac{B_n T_s}{\zeta + \frac{1}{4\zeta}}\right)^2}{1 + \frac{2\zeta}{N}\left(\frac{B_n T_s}{\zeta + \frac{1}{4\zeta}}\right) + \left(\frac{B_n T_s}{N\left(\zeta + \frac{1}{4\zeta}\right)}\right)^2}.$$

❑ From analog PLL to digital PLL and its parameters

- **Digital PLL example**







f_{DDS} = 0.01 cycles/sample, f_{in} = 0.0105 cycles/sample

ζ = 0.707, B_nT = 0.2%

We get: K1 = 0.0053
and K2 = 1.4184e-05

Outline

1 –

2 –

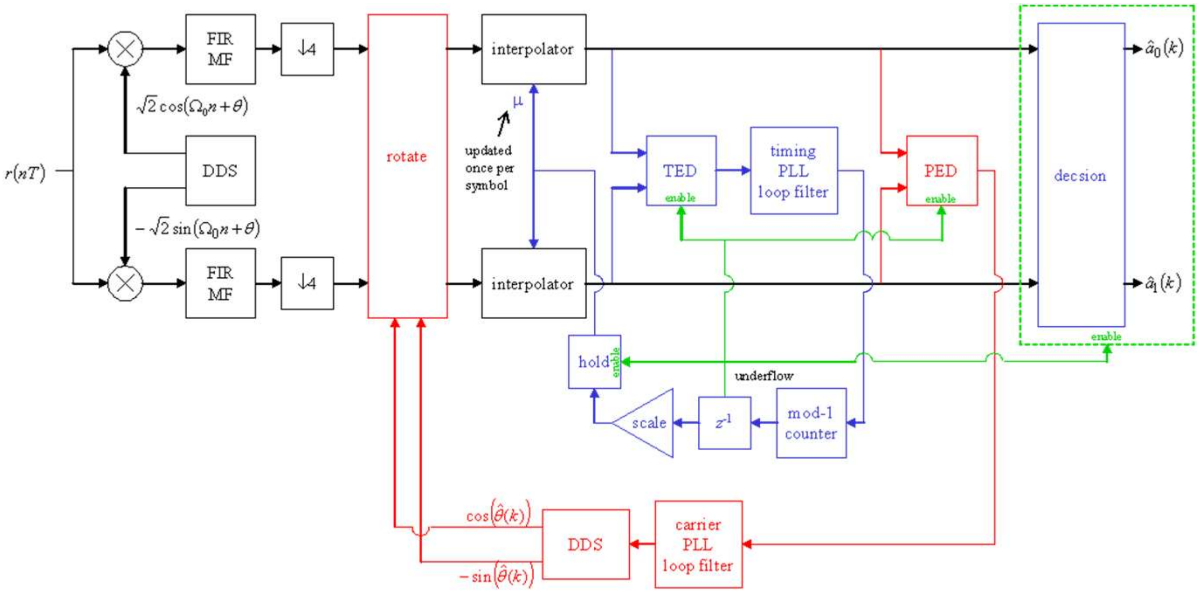
3 – The fully digital QPSK synchronizer.

4 –

5 –

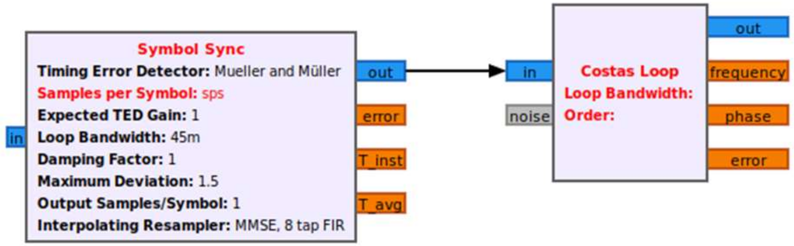
❑ The all digital phase/frequency and timing synchronizer

What we want to build



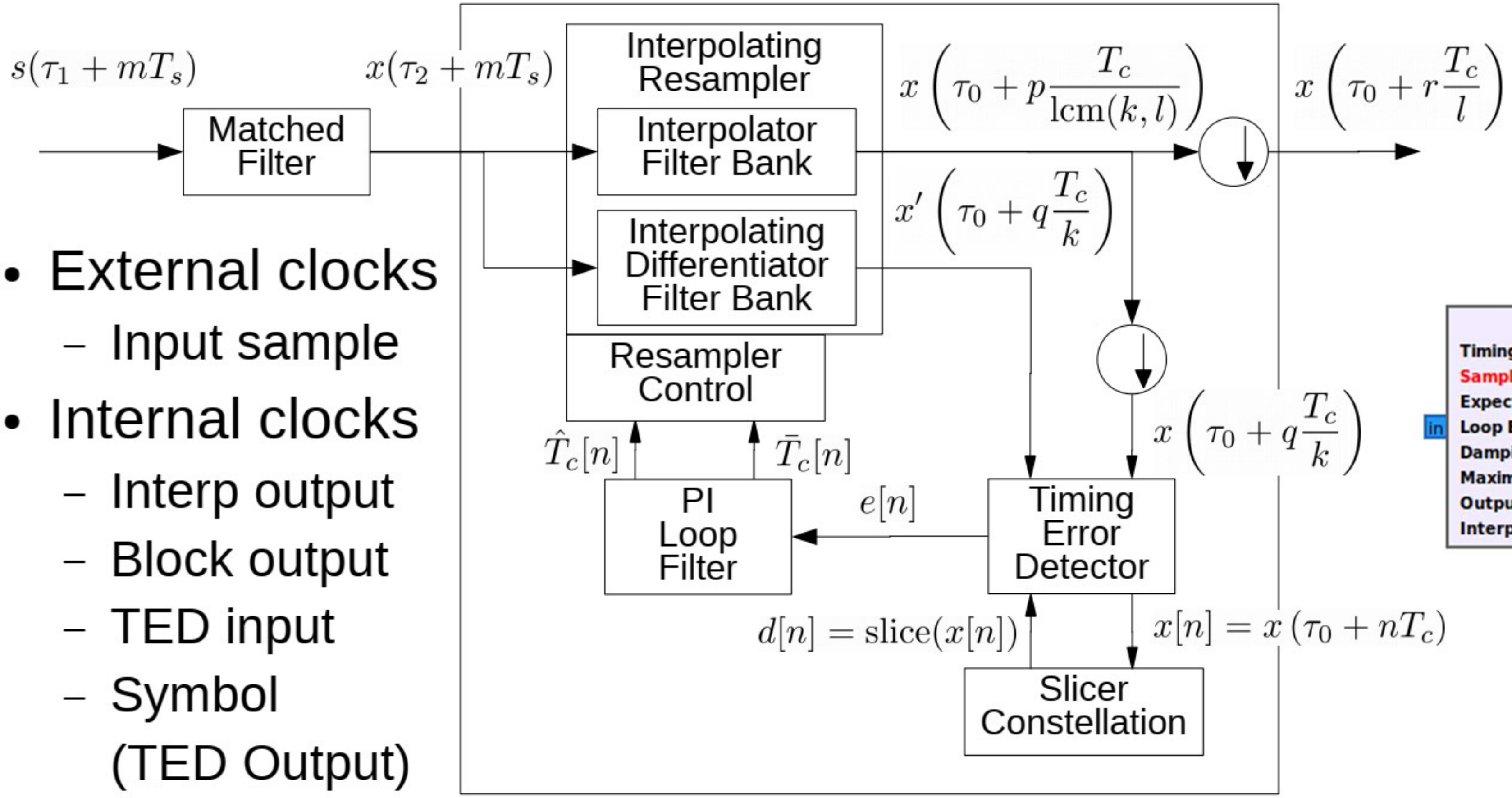
The red parts = GNU Radio Costas Loop block

Has it something to do with this?

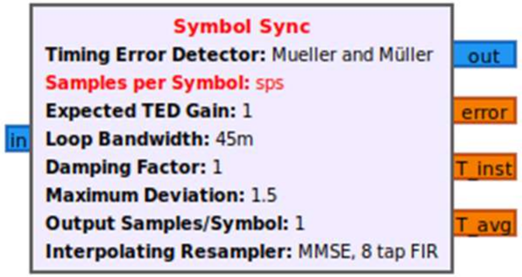


❑ Let's concentrate on the symbol timing synchronization

❑ The timing synchronizer and A. Wall's GNU Radio Symbol Sync block



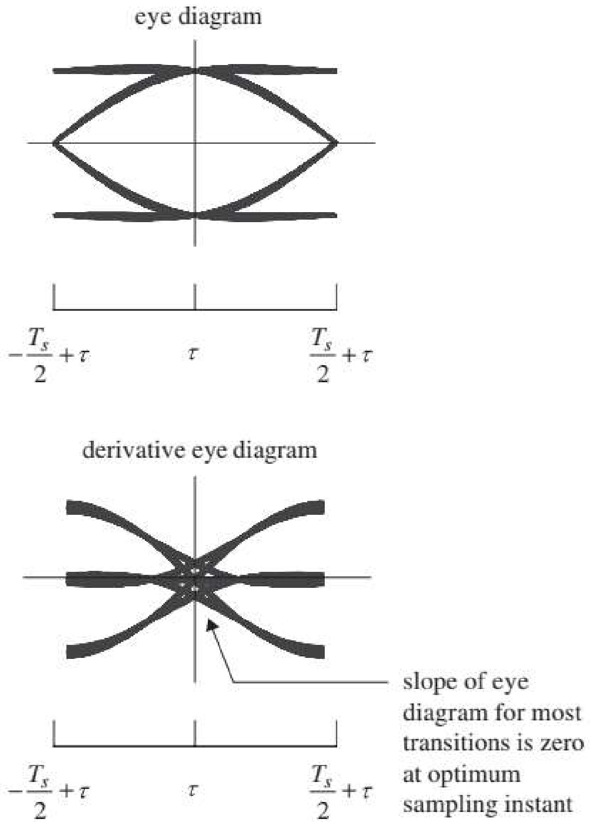
- External clocks
  - Input sample
- Internal clocks
  - Interp output
  - Block output
  - TED input
  - Symbol (TED Output)

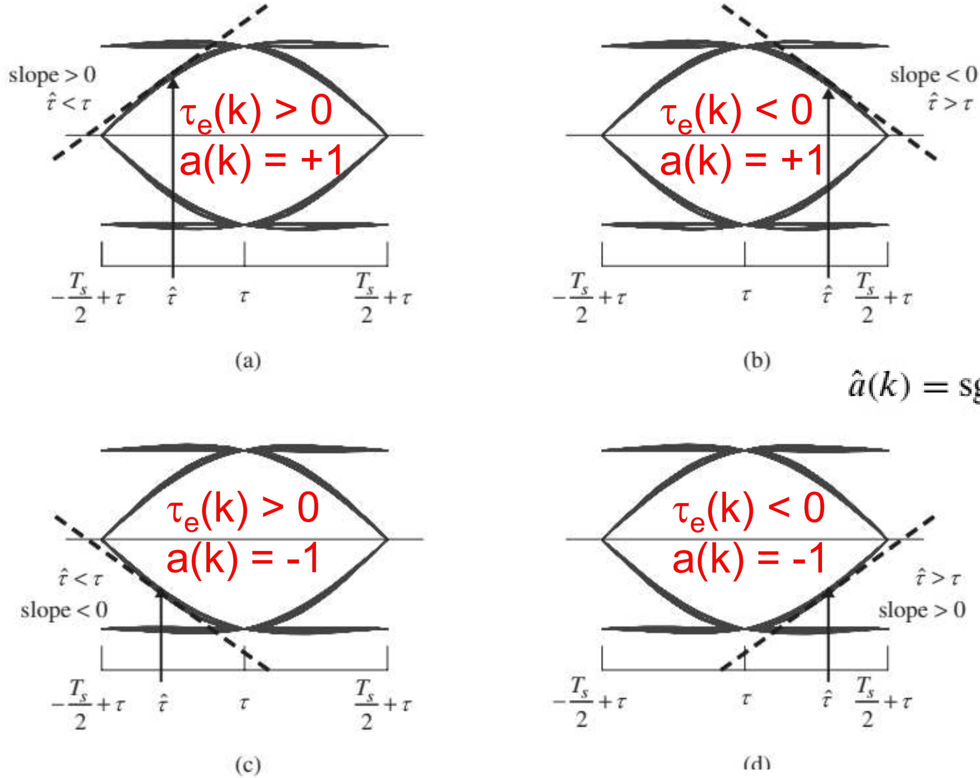


➔ Two new elements: Interpolator and Timing Error Detector (TED)

❑ The timing synchronizer: Timing Error Detector (TED)

- **Plays the role of phase detector in the DPLL**
- **Best understood with the eye diagram whose slope can be used to generate a timing error e(k) (binary PAM)**





$$\hat{a}(k) = \text{sgn}\left\{x(kT_s + \hat{\tau})\right\}$$

➔ **Error signal:** $e(k) = a(k)\dot{x}\left(kT_s + \hat{\tau}(k)\right)$ **DA**

ML estimate $e(k) = \hat{a}(k)\dot{x}\left(kT_s + \hat{\tau}(k)\right)$ **DD**

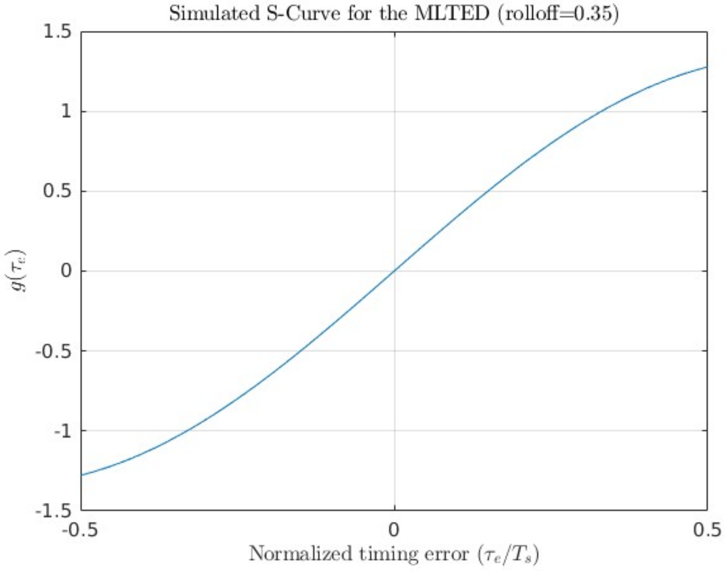
❑ The timing synchronizer: Timing Error Detector (TED)

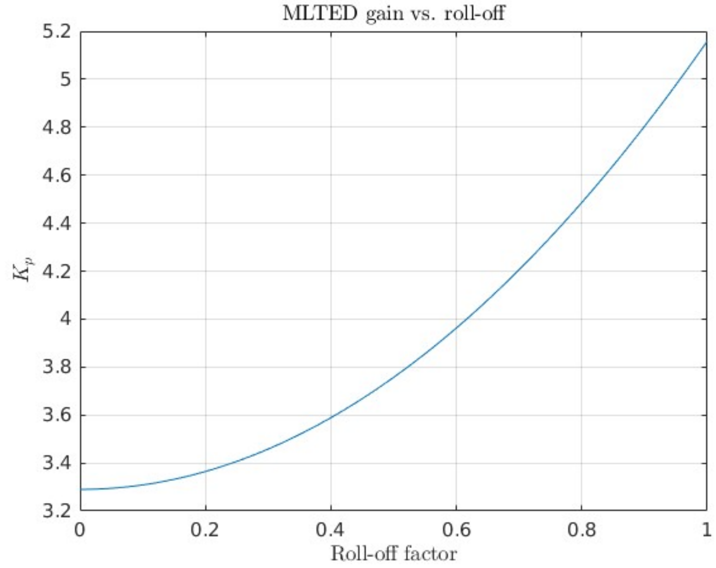
- **There exists quite a lot of TED structures (ZC, M&M, Gardner etc.).**
- **Characteristics of TED described by the S-curve $g(\tau_e)$**
- **S-curve obtained by computing expected value of error signal:**

$$g(\tau_e) = \mathrm{E}\{e(k)\}$$

- **For example for the MLTED we get:**

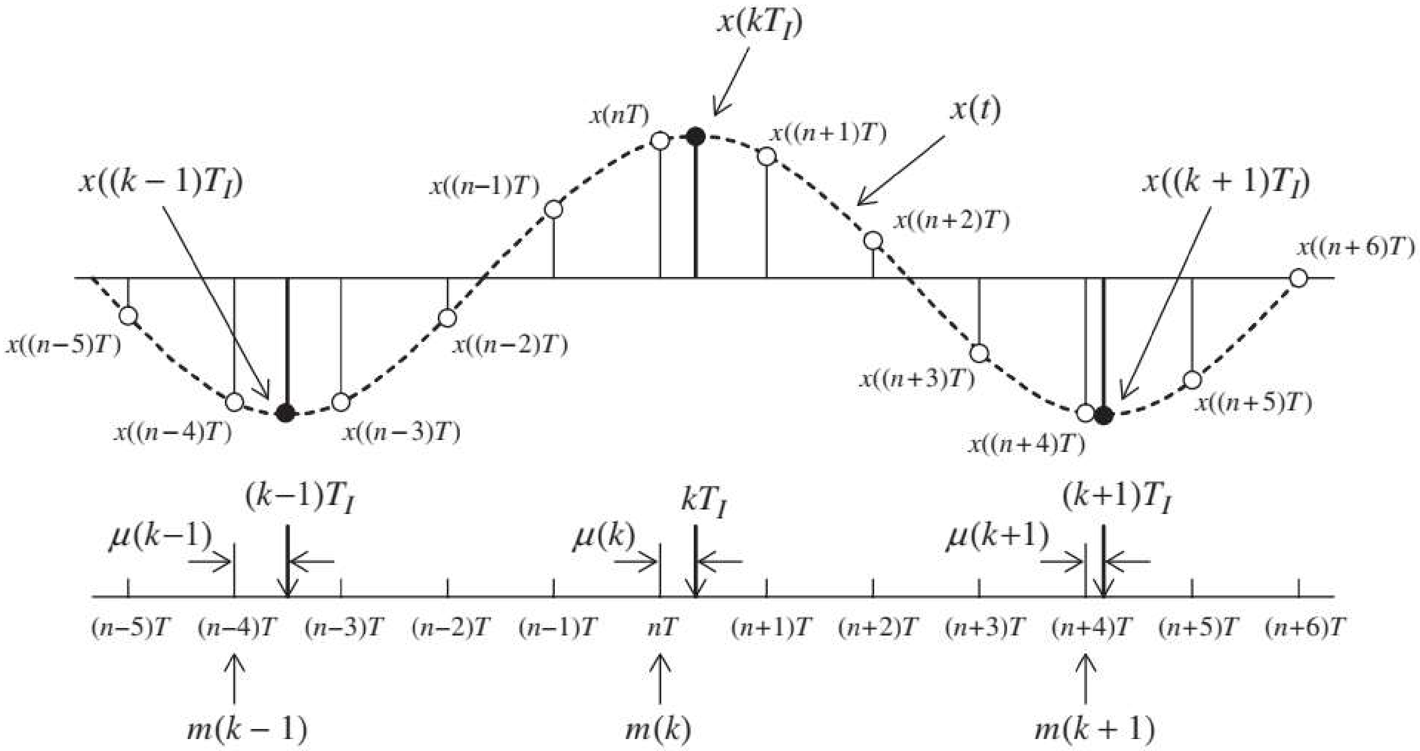
Needed by Symbol Sync block!





❑ The timing synchronizer: Interpolator structures

- **Because we use asynchronous sampling (i.e. sample clock independent to the data clock used at the transmitter) we need an interpolator.**
- **Some vocabulary:**

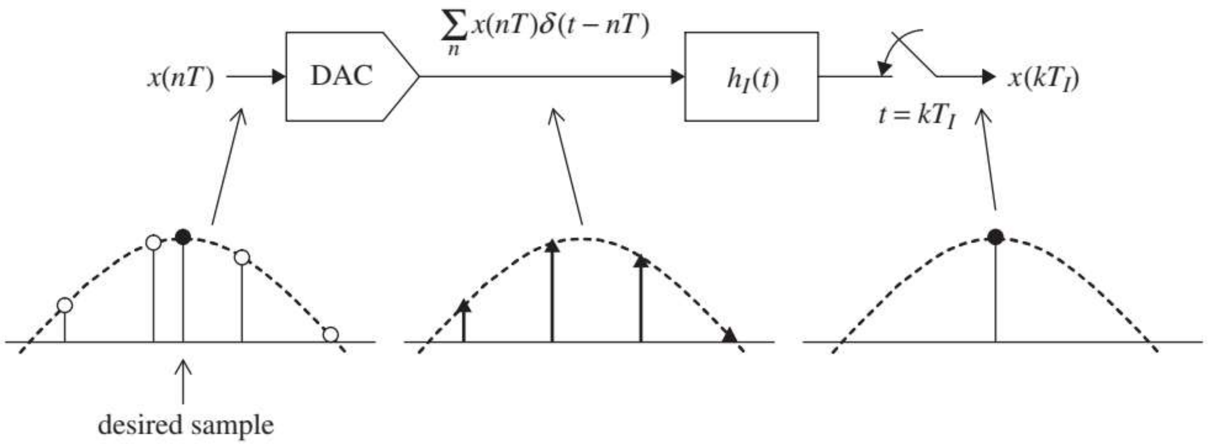


- **k-th interpolant = kT_I**
- **When k-th *interpolant* is between samples x(nT) and (x(n + 1)T) the sample index n is called the k-th *basepoint index* and denoted m(k).**
- **kT_I is some fraction of a sample time greater than m(k)T.**
- **This fraction is called the k-th fractional interval and is denoted $\mu(k)$. $0 < \mu(k) < 1$.**
- **It is defined by:**

$$\mu(k)T = kT_I - m(k)T$$

❑ The timing synchronizer: Interpolator structures

- **Conceptually, an interpolator can be seen as a filter.**



- **The k-th interpolant evaluated at t = kT$_I$ and may be expressed as:**

$$x(kT_I) = \sum_n x(nT) h_I(kT_i - nT).$$

- **If we reexpress with a filter index i, m(k) = int(kT$_I$/T) and μ(k) = kT$_I$ /T– m(k) the filter index is i = m(k) – n. Then:**

$$x(kT_I) = \sum_i x\left((m(k) - i)\, T\right) h_I\left((i + \mu(k))\, T\right).$$

❑ The timing synchronizer: Interpolator structures

- **From now to build the interpolator we can:**

  ➔ Use a FIR piecewise polynomial filter

  ➔ A polyphase-filterbank (massively upsample input of matched filter then downsample matched filter output at an appropriately chosen sample offset to get desired interpolant).

- **Let's have a look at the first possibility.**
- **The continuous underlying waveform is approximated by a polynomial of the form:**

$$x(t) \approx c_p t^p + c_{p-1} t^{p-1} + \cdots + c_1 t + c_0.$$

- **The polynomial coefficient are determined by the p + 1 sample values surrounding the basepoint index. Once the coefficients values are known, the interpolant at t = kTI = (m(k) + μ(k)T is obtained by:**

$$x(kT_I) \approx c_p (kT_I)^p + c_{p-1} (kT_I)^{p-1} + \cdots + c_1 (kT_I) + c_0.$$

❑ The timing synchronizer: Interpolator structures

- **Let's have a look at the first possibility.**
- **It can be shown that:**

$$x((m(k)+\mu(k))T) = \sum_{i=-2}^{1} h_2(i)x((m(k)-i)T)$$

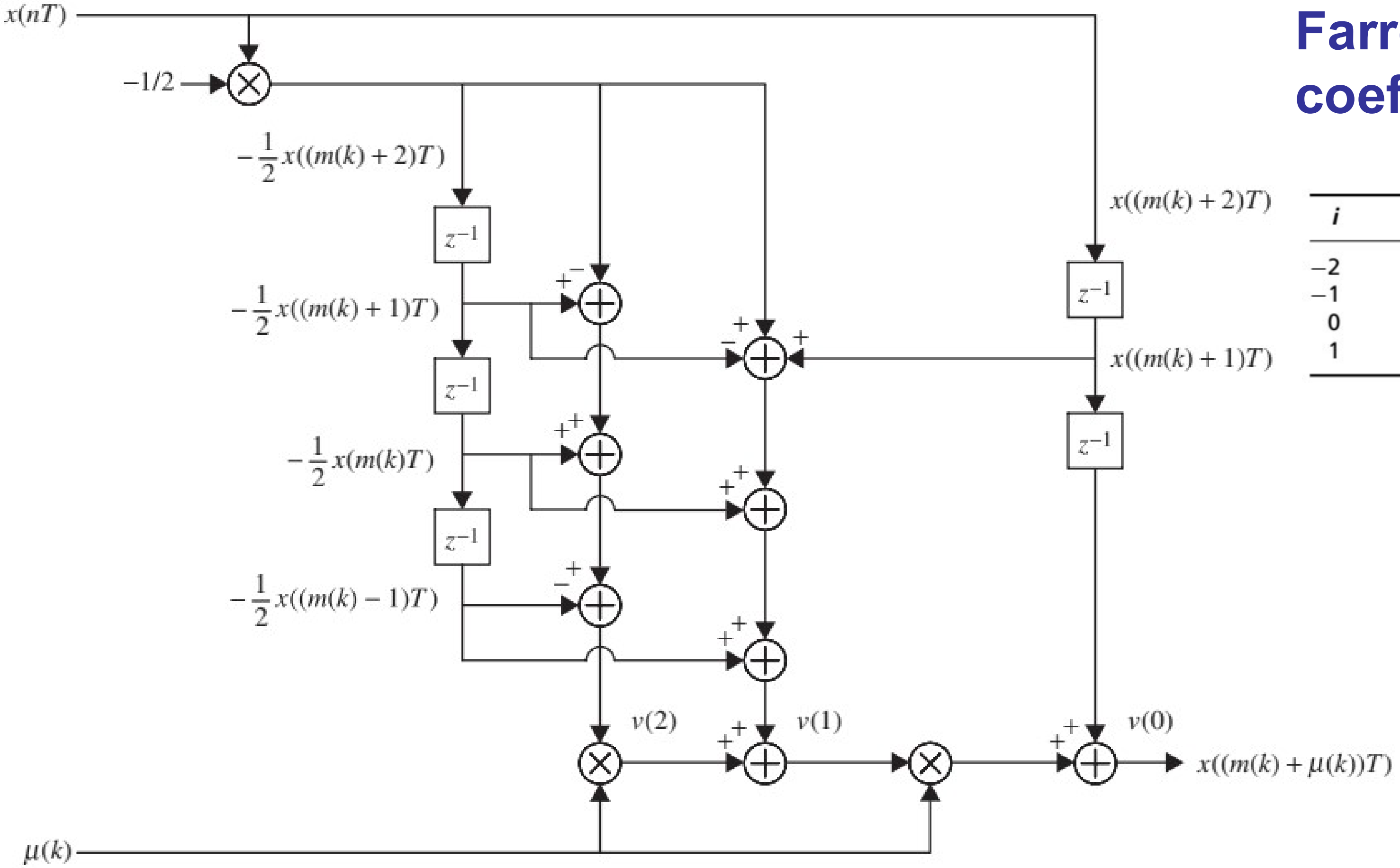
- **In the case of a parabolic interpolator this leads to four coefficients:**

$$\begin{cases} h_I[-2+\mu_k] = \alpha\mu_k^2 - \alpha\mu_k \\ h_I[-1+\mu_k] = -\alpha\mu_k^2 + (\alpha+1)\mu_k \\ h_I[\mu_k] = -\alpha\mu_k^2 + (\alpha-1)\mu_k + 1 \\ h_I[1+\mu_k] = \alpha\mu_k^2 - \alpha\mu_k \end{cases}$$

- **The interpolation equation becomes:**

$$\begin{aligned} y(\mu_k) &= \sum_{i=-2}^{1} x[(-i)]h_I[(i+\mu_k)] \\ &= x[2](\alpha\mu_k^2 - \alpha\mu_k) + x[1](-\alpha\mu_k^2 + (\alpha+1)\mu_k) \\ &\quad + x[0](-\alpha\mu_k^2 + (\alpha-1)\mu_k + 1) + x[-1](\alpha\mu_k^2 - \alpha\mu_k) \end{aligned}$$

❑ The timing synchronizer: Interpolator structures

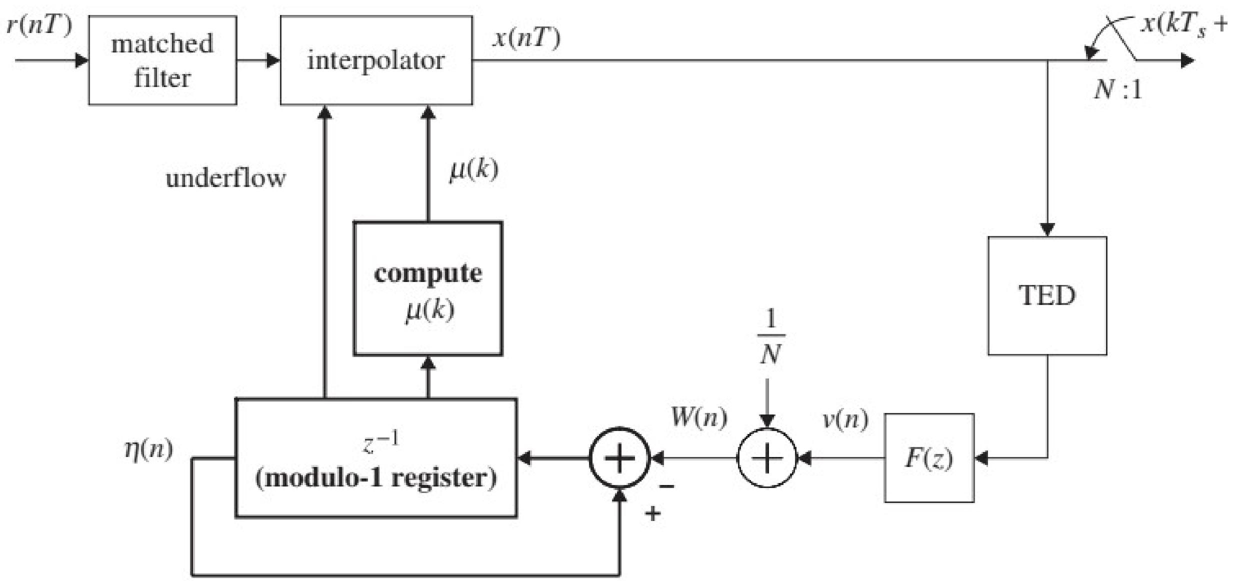


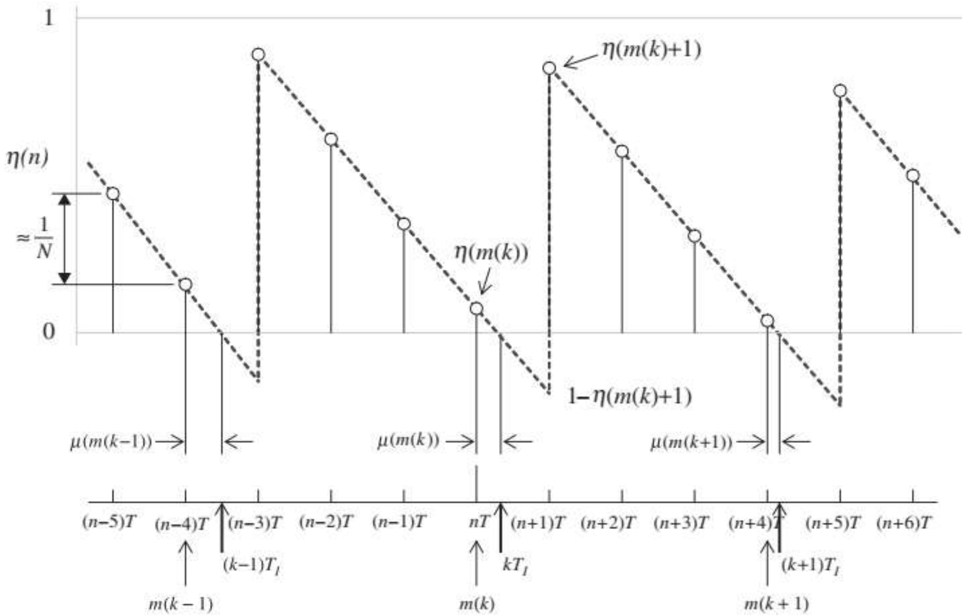
Farrow filter coefficients:

i	$b_2(i)$	$b_1(i)$	$b_0(i)$
−2	α	$-\alpha$	0
−1	$-\alpha$	$1+\alpha$	0
0	$-\alpha$	$\alpha-1$	1
1	α	$-\alpha$	0

❑ The timing synchronizer: Interpolator structures

- **The Farrow interpolator is controlled by a modulo 1 counter:**





Outline

1 –

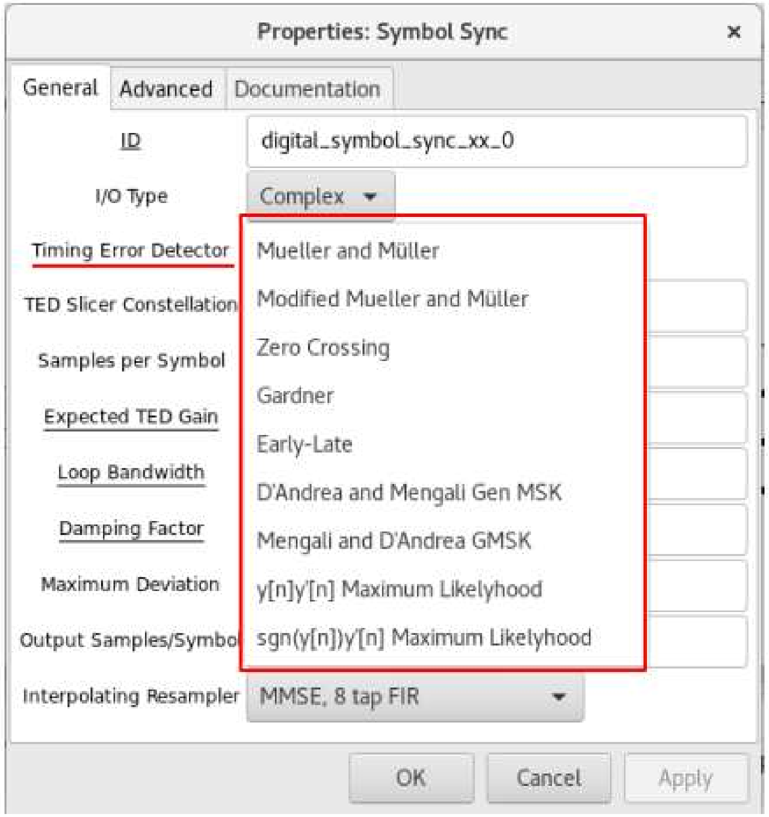
2 –

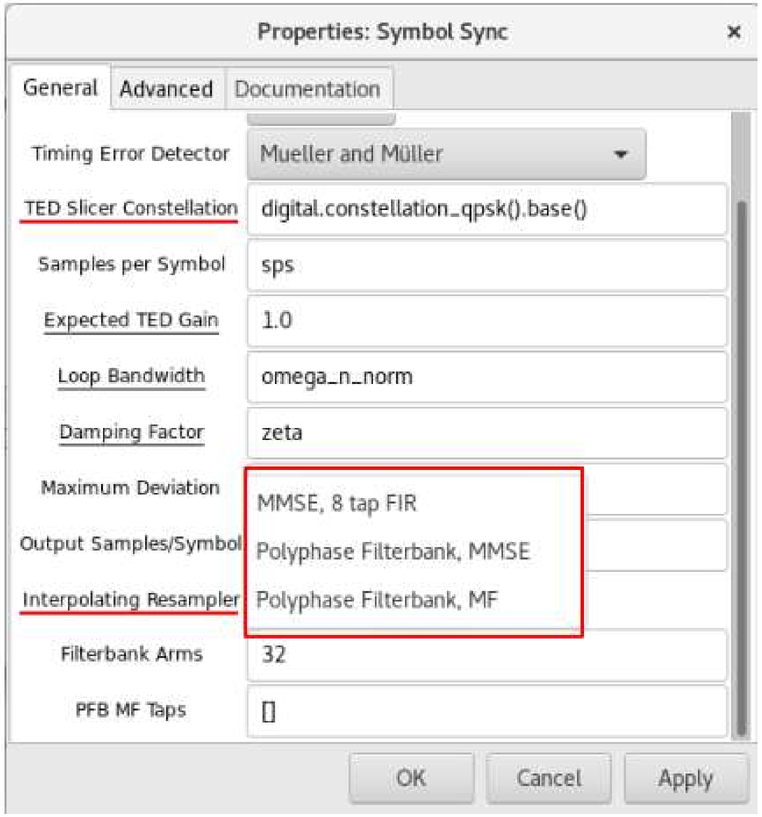
3 –

4 – GNU Radio demo with real hardware

5 –

❑ QPSK system with the Adalm-Pluto





Outline

1 –

2 –

3 –

4 –

5 – Conclusion

5. Conclusion

- **Going from simulation to implementation adds (a lot of) complexity.**

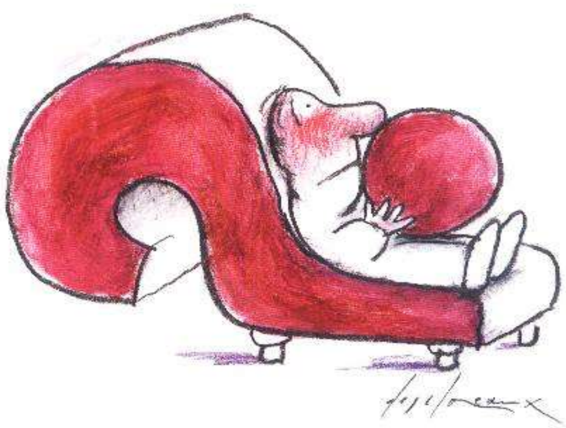
- **Digital synchronization functions understanding and implementation require a good level in digital signal processing.**

- **Practical implementations are requested by today's students.**

- **For this particular purpose, GNU Radio is the right tool!**

[1] M. Rice, “Digital Communications: a discrete time approach”, 2nd Edition, Author, 2020.
[2] T. P. Zielinski, “Starting Digital Signal Processing in Telecommunication Engineering”, Springer, 2021.
[3] https://igorfreire.com.br/2016/10/15/symbol-timing-synchronization-tutorial/
[4] http://ricesimulink.groups.et.byu.net/

Thank you for your attention!



Questions?