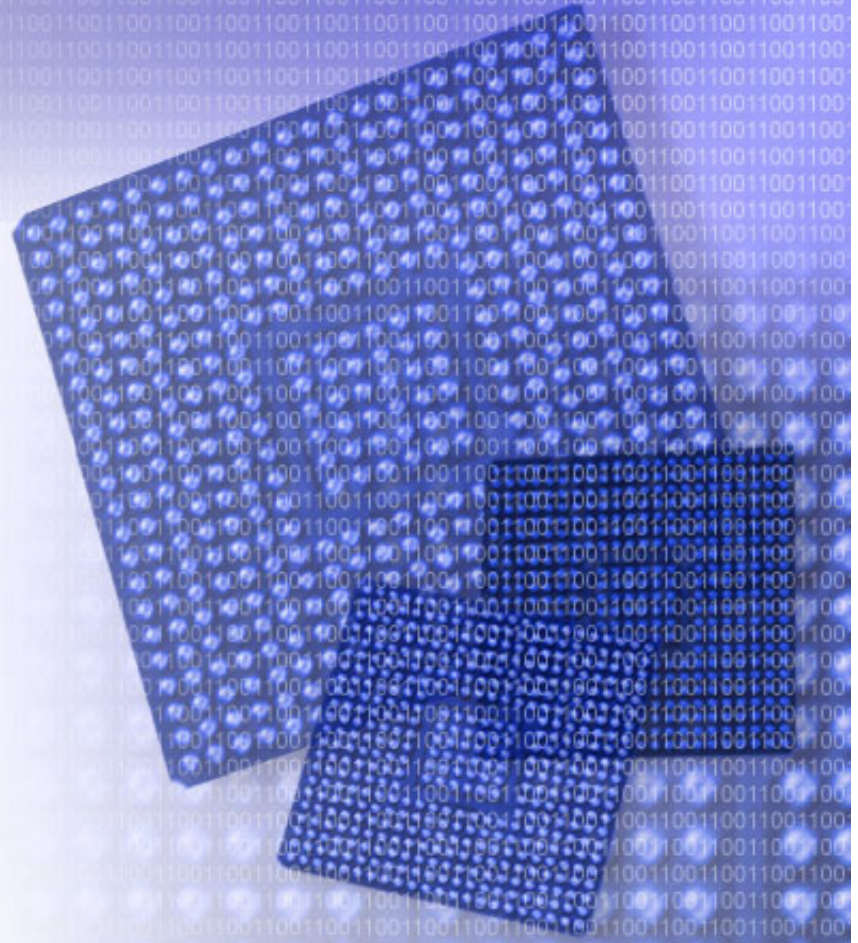
XILINX®

The **DSP** Primer15

Timing Issues



Return
DSP*primer* Home

Return
DSP*primer* Notes

THIS SLIDE IS BLANK

Contents

- The synchronization problem;
- The effect synchronization of errors;
- Carrier recovery loops;
- Numerically controlled oscillators;
- Timing recovery loops;
- Loop filter implementation;
- Sampling rate conversion and downconversion;
- BPSK receiver example;
- How do we address the synchronisation problem with FPGAs.

Notes:

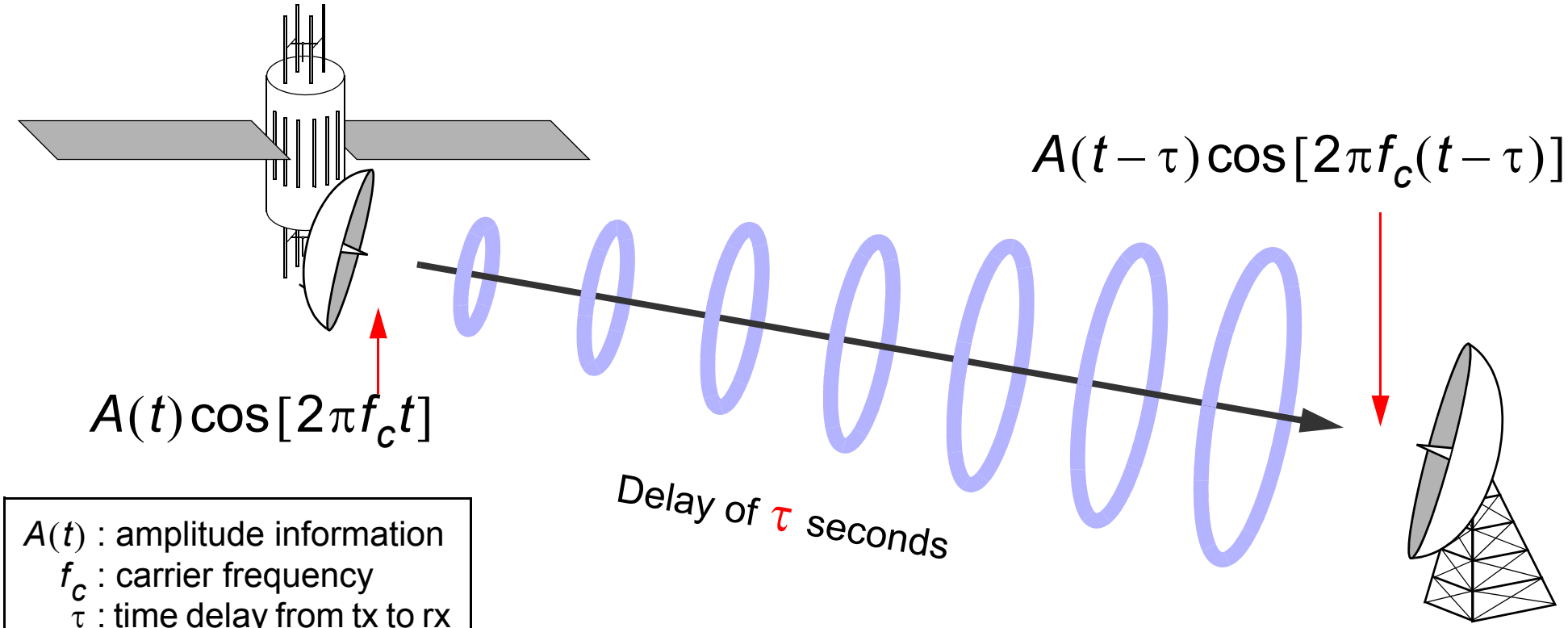
Timing recovery and synchronisation is one of the most important aspects of digital communications - if you do not get this right then you will not be successful in recovering the transmitted data.

Key DPS components to be developed in this section includes DLL (delay locked loops), PLL (phase locked loops), Early-Late gate detection, polyphase filters.

In any FPGA design we must take care of the timing recovery with suitable DSP circuitry.

The Synchronization Problem

- Propagation delay between transmitter and receiver is unknown;

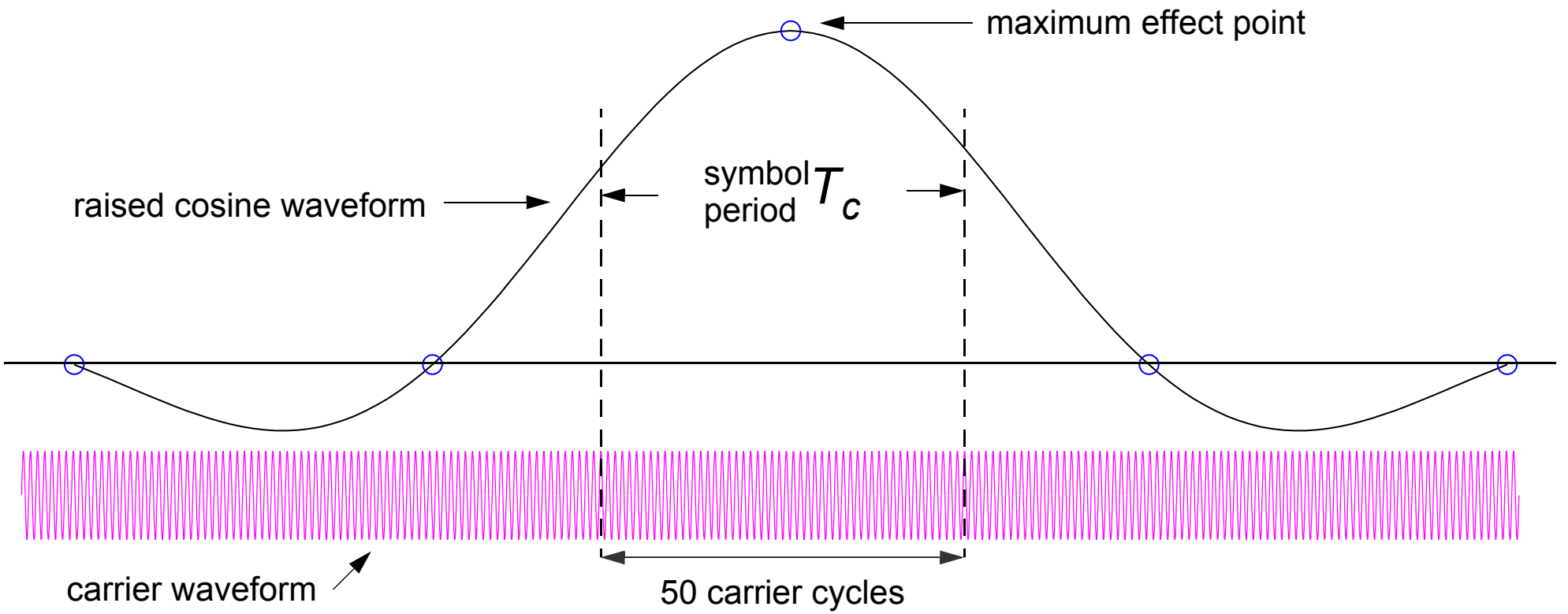


$A(t)$: amplitude information
f_c : carrier frequency
τ : time delay from tx to rx

- Must estimate two important parameters:
  - Symbol timing - when to sample;
  - Carrier phase offset.

Notes:

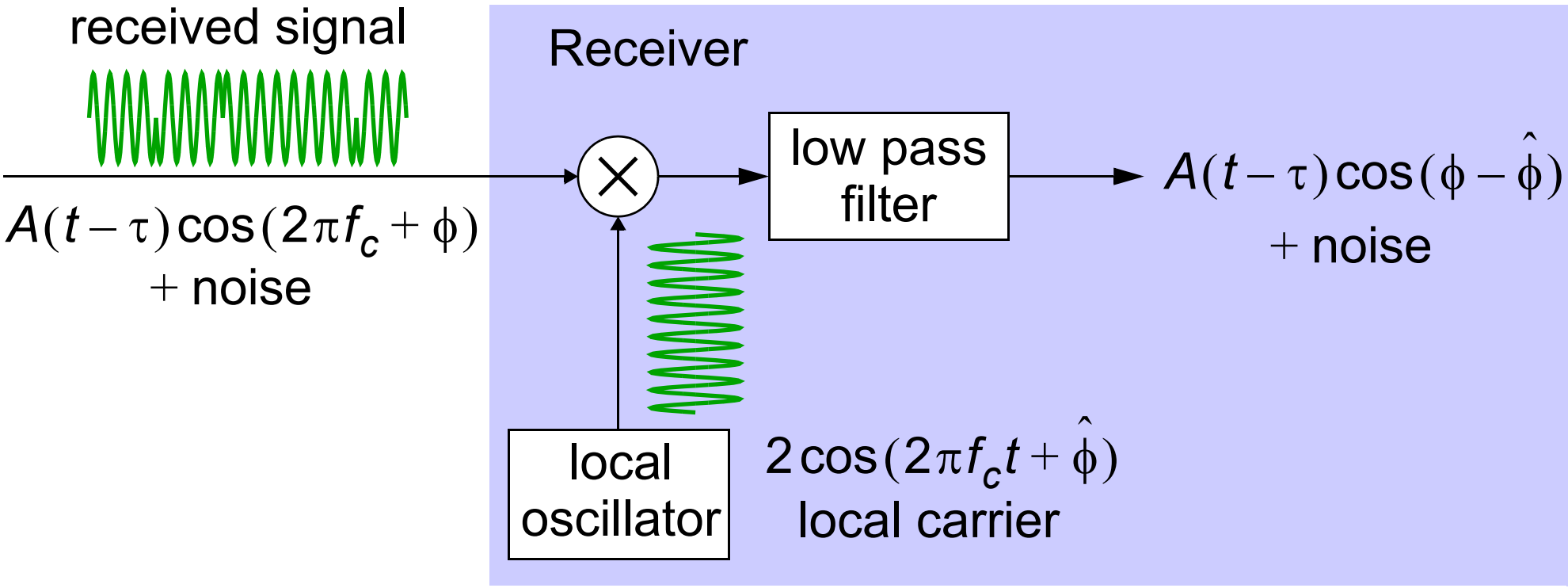
So why do we have to estimate both the symbol timing and the carrier phase? Surely if we can estimate the propagation delay then this can be used to synchronize both the local carrier replica as well the symbol timing! The reason that both require to be estimated and tracked is that the carrier period is typically much smaller than the symbol period. Consider an example where the carrier frequency is 100kHz and the symbol rate is 20k symbols per second. This is illustrated below. Note that the carrier frequency is 50 times the symbol rate.



Clearly any change in delay between the transmitter and receiver which is small relative to the symbol period will have a much greater effect on the carrier phase error than it would on the symbol timing error. It is therefore useful to estimate the optimal symbol timing and the carrier phase as separate parameters. These are effectively two different **levels** of synchronization.

Effect of Carrier Phase Offset

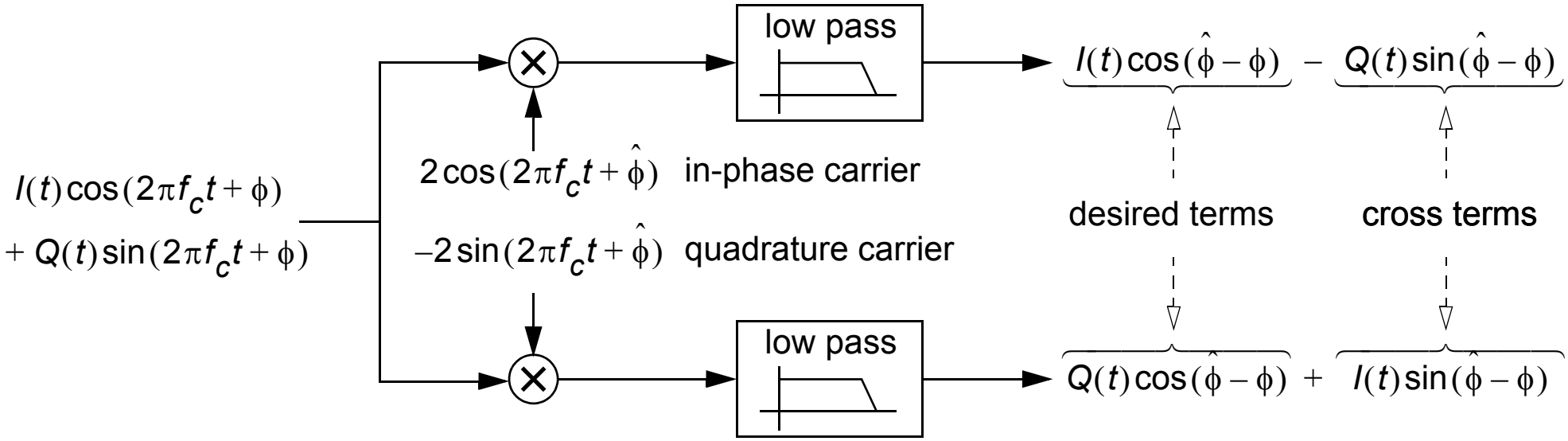
- Consider receiving a modulated signal as shown below;
- The signal strength is reduced due to a phase error in the local carrier compared to the received signal;



- Hence there is a loss in the signal-to-noise ratio.

Notes:

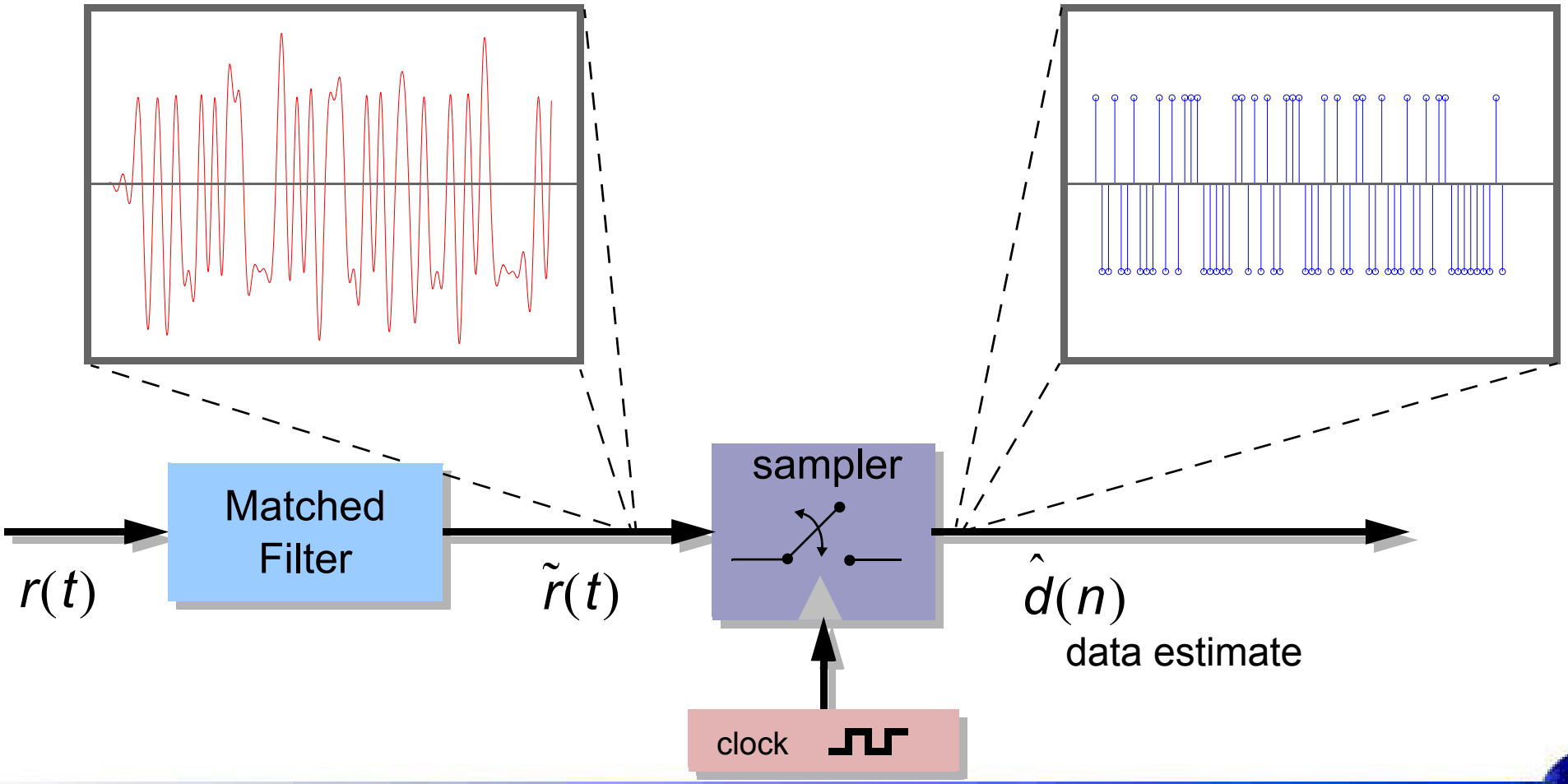
The effect of carrier phase errors on QAM signals, which use both in-phase and quadrature components, is even worse. In a QAM receiver both in-phase and quadrature versions of the carrier are required to demodulate the incoming signal. If the local carrier is perfectly synchronized to the received signal then the in-phase carrier should be orthogonal to the quadrature component of the received signal. The same goes for the locally generated quadrature carrier and the received in-phase component. In the presence of a phase error this orthogonality is destroyed and the in-phase and quadrature components interfere with each other, degrading the performance of the system even further.



$I(t)$: in-phase component
$Q(t)$: quadrature component

Symbol Timing

- Receiver filters are matched to optimise the signal to noise ratio of their output;
- In addition the output of the Matched Receiver Filter must be sampled on the Maximum Effect Points to prevent inter-symbol interference.

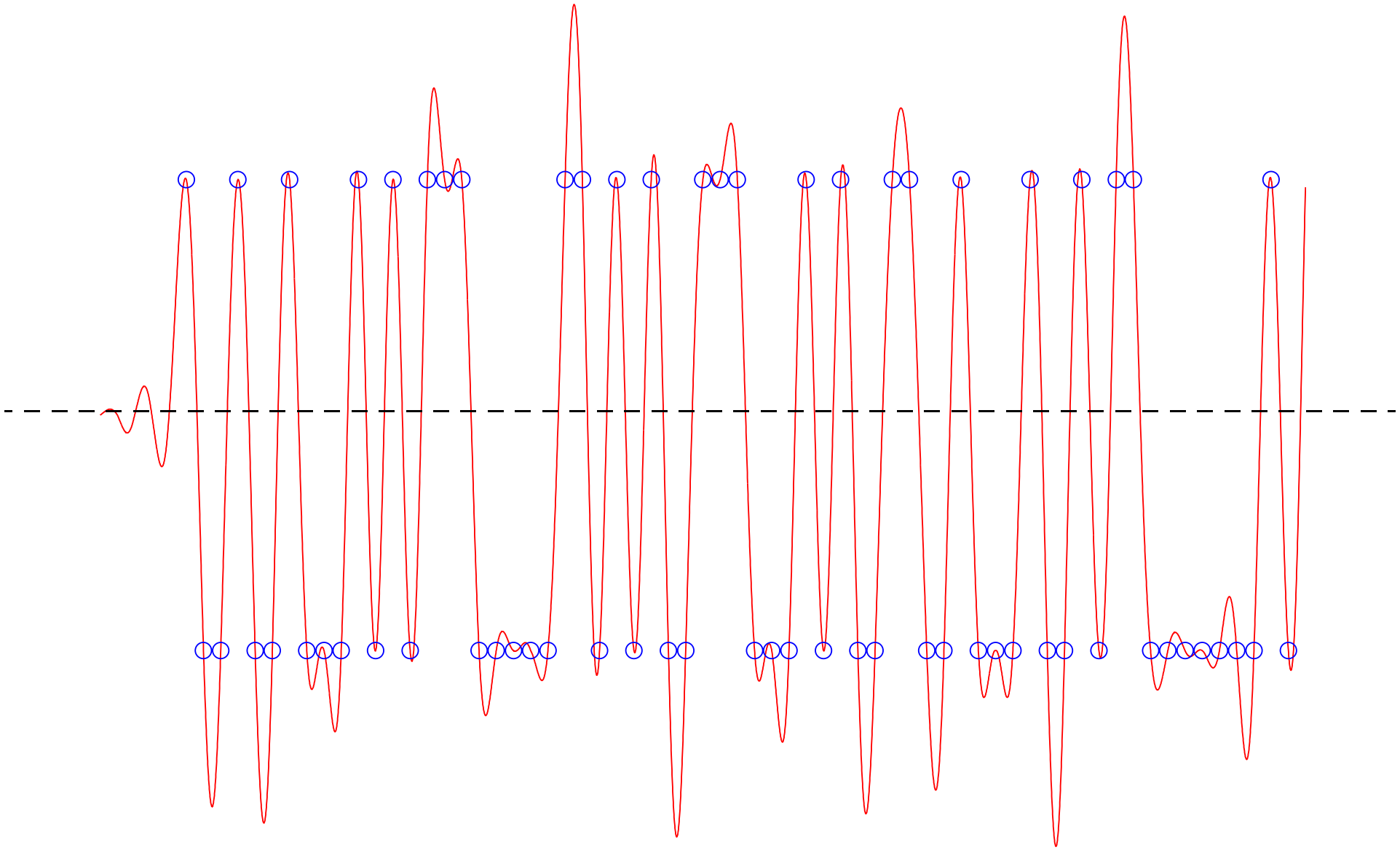


Notes:

Pulse shapes (such as the root-raised cosine pulse) are usually designed so that when a filter matched to the pulse shape is used at the receiver then intersymbol interference is be avoided. This however depends on whether the signal is sampled at the correct time sampling instants. The optimal points to sample the signal which result in zero intersymbol interference are called the **maximum effect points**.

Effect of Symbol Timing Errors (I)

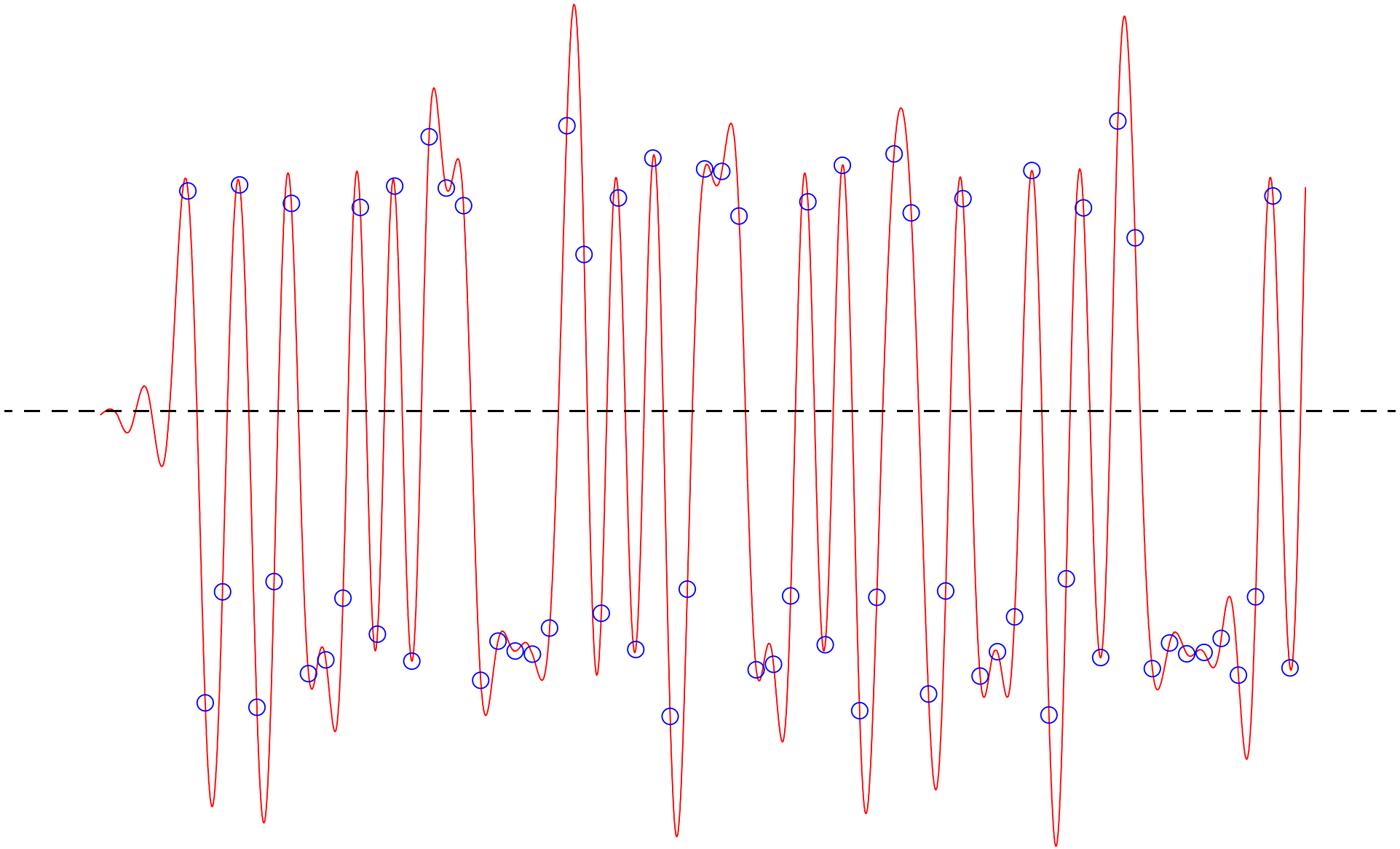
- No symbol timing error:



Notes:

Effect of Symbol Timing Errors (II)

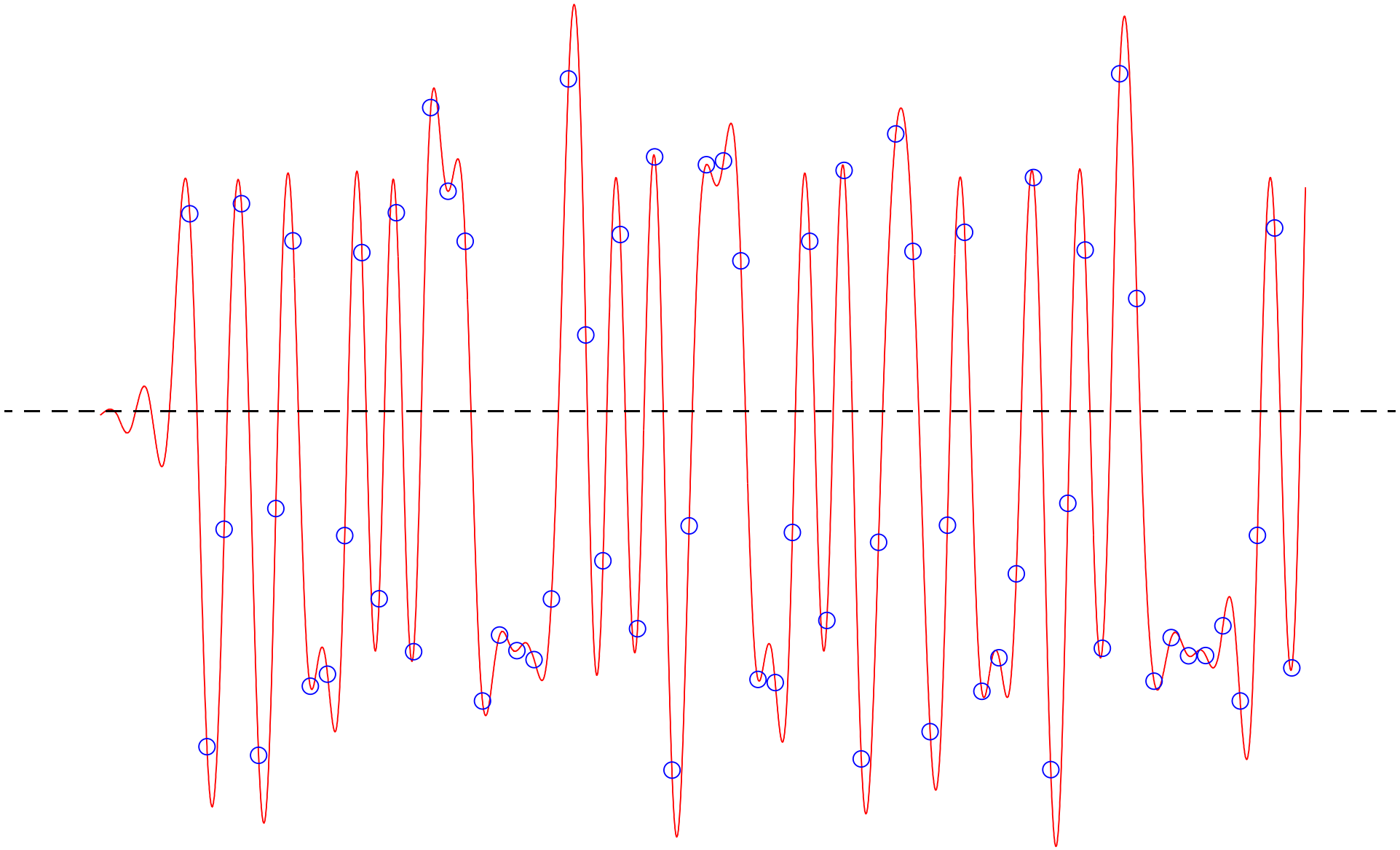
- Timing error = 10% of symbol period:



Notes:

Effect of Symbol Timing Errors (III)

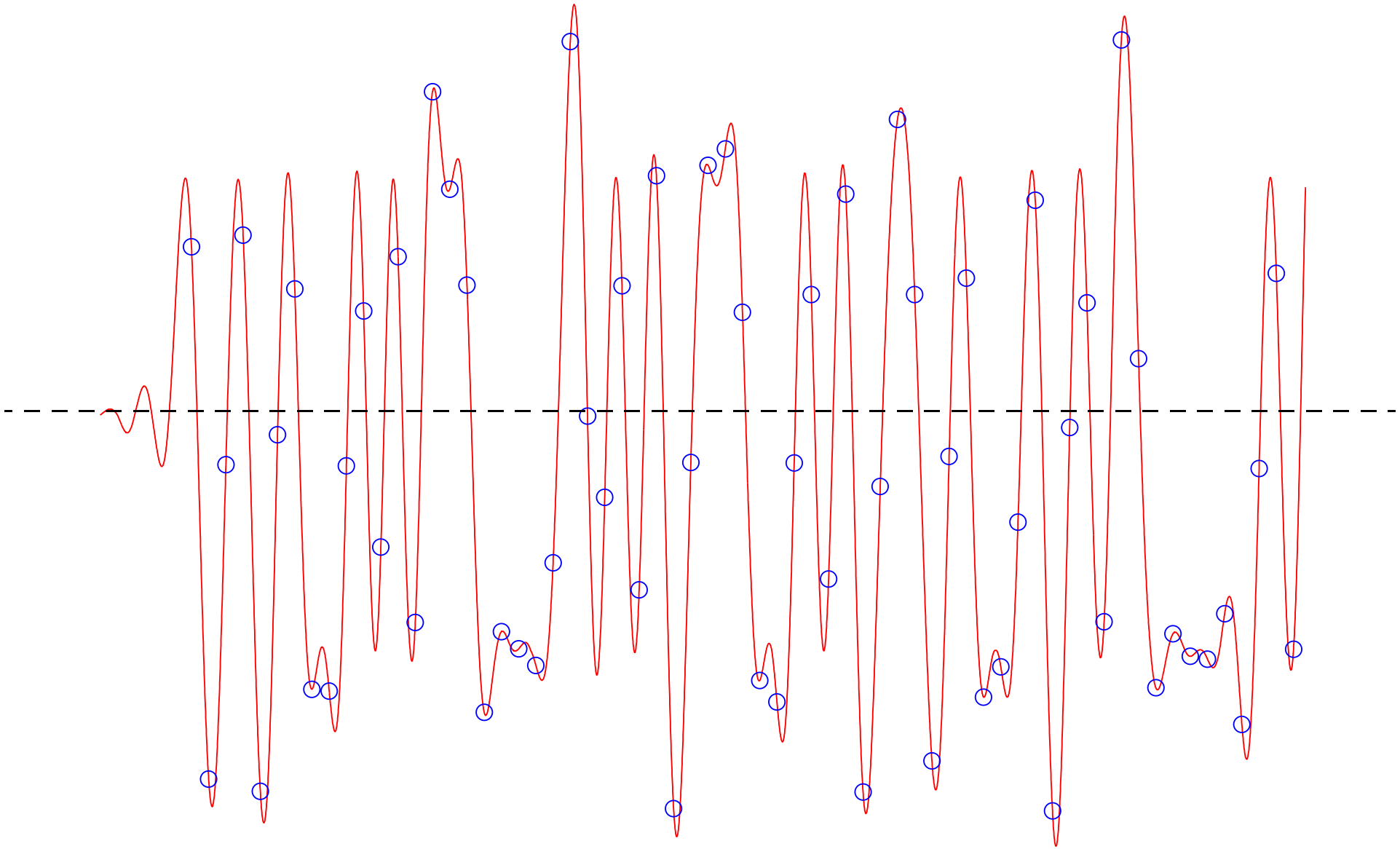
- Timing error = 20% of symbol period:



Notes:

Effect of Symbol Timing Errors (IV)

- Timing error = 30% of symbol period:



Notes:

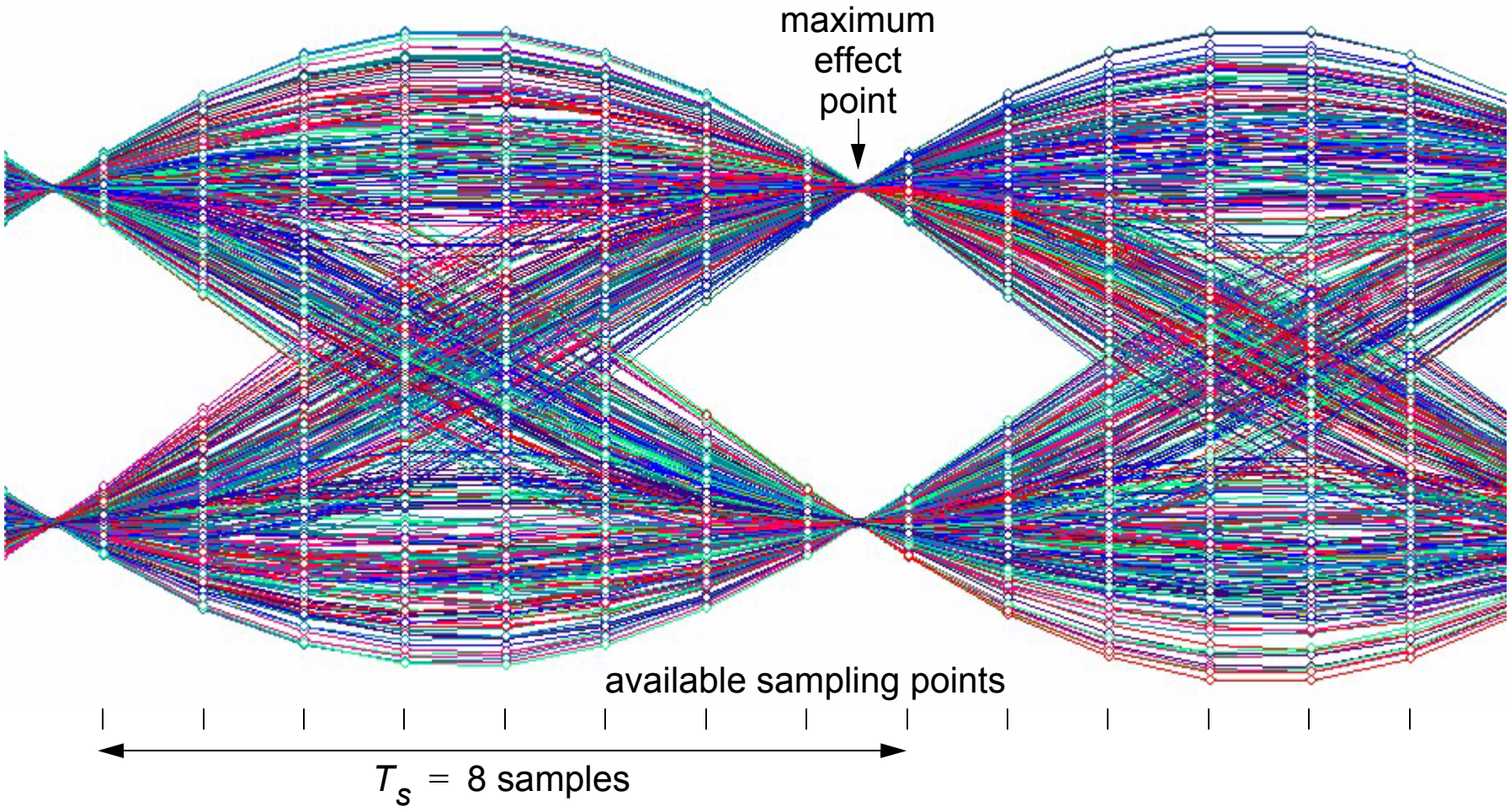
An error in symbol timing effectively introduces an additional noise term to the output of the sampler. As a result of this incorrect symbol timing may cause bit errors to occur even if the channel and thermal noise components are insignificant.

Timing Resolution

- Digital receivers typically have limited timing resolution.
- This is a direct result of the method by which timing adjustments are made.
- Two timing adjustment techniques are...
    - Oversampling by a large factor and selecting the best set of samples;
    - Oversampling by a small factor (e.g. 2x symbol rate) and using interpolation techniques.
- Both of these techniques may yield finite timing resolution;
- High sampling rates should be avoided if possible;
- High sampling rate = high complexity in signal processing functions.

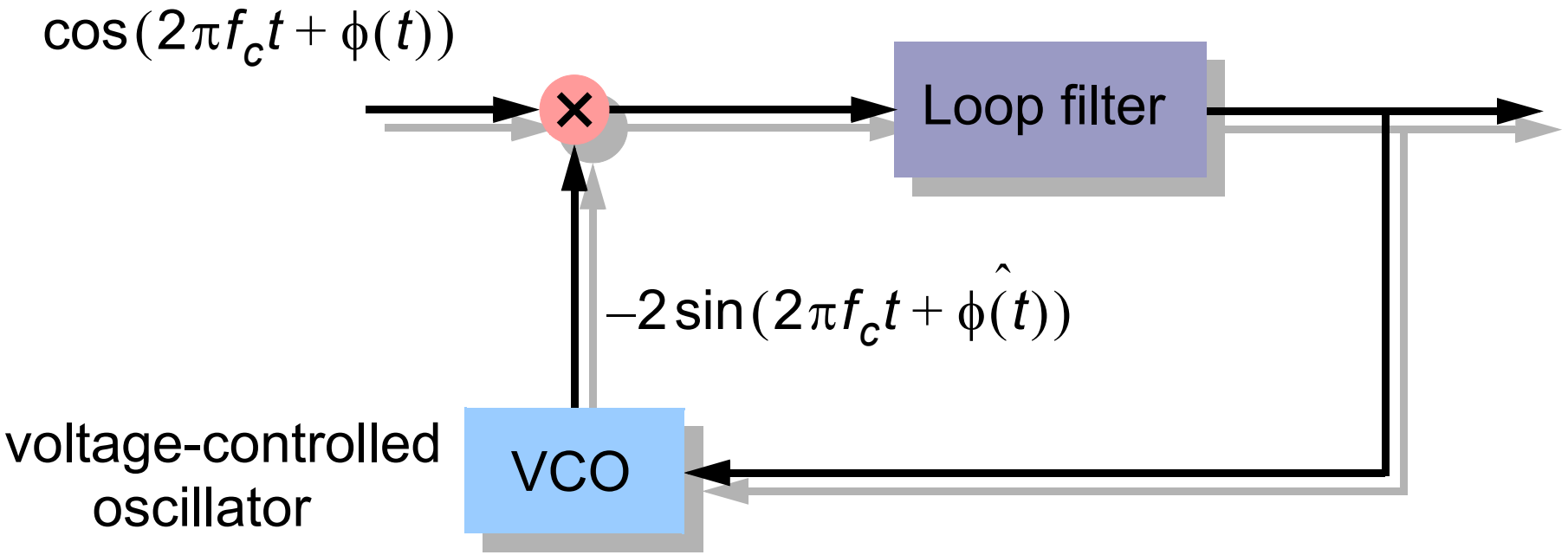
Notes:

The diagram shows an example of an eye diagram for a raised cosine filtered BPSK signal. The signal is 8 times oversampled and the available sampling points at this oversampling ratio are shown. Note that none of the sampling points fall on the maximum effect point. Therefore some intersymbol interference will always be present in this case. The timing resolution must be fine enough to yield an acceptable amount of intersymbol interference.



Phase-Locked Loops (PLLs)

- Phase modulated systems may use a phase-locked loop (PLL) to track the carrier phase;
- In such systems the PLL can actually perform the demodulation;



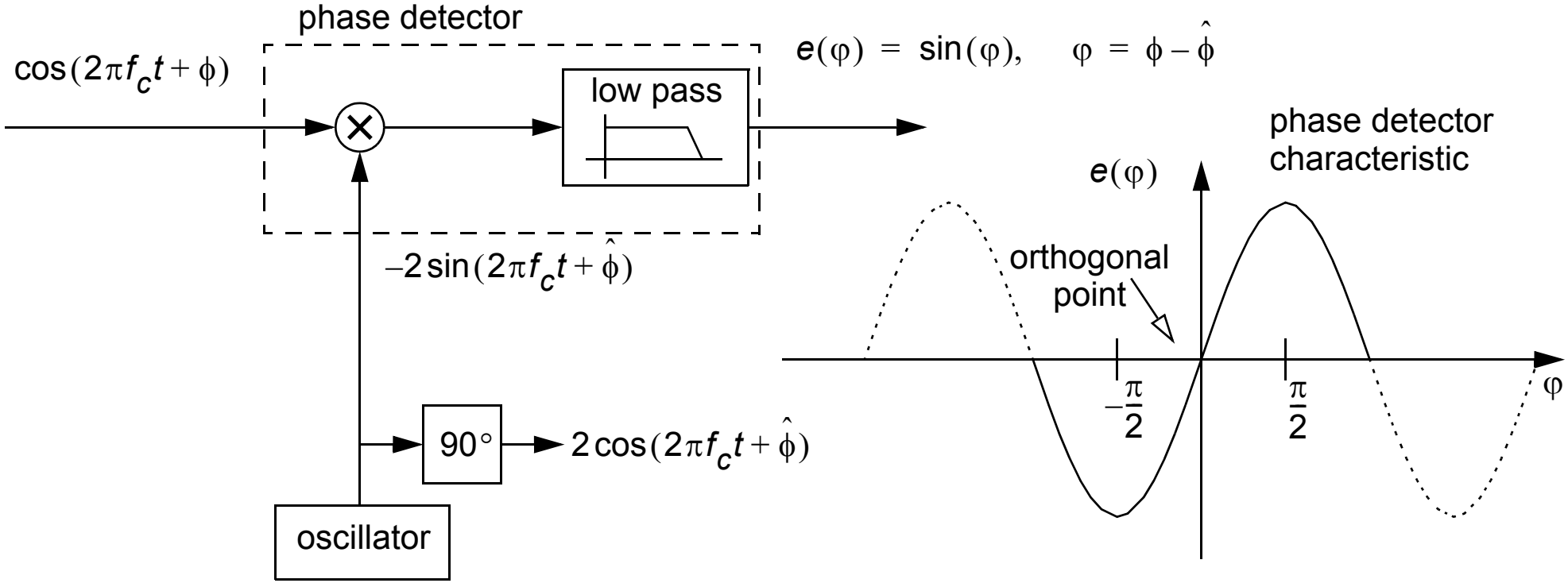
- A PLL can also be used to identify the nominal carrier phase and frequency.

Notes:

The operation of a PLL is based on the fact that $\sin[\theta(t)]$ and $\cos[\theta(t)]$ are orthogonal to one another. This means that the following equation holds

$$\int_{-\infty}^{\infty} \sin[\theta(t)]\cos[\theta(t)]dt = 0$$

If the two signals are not exactly 90 degrees out of phase then the orthogonality is destroyed. We can therefore use this property to design a device which is able to estimate the phase error between two carrier signals. This is shown in the diagram below. Note that the factor of 2 in the local oscillator is introduced to simplify the maths.

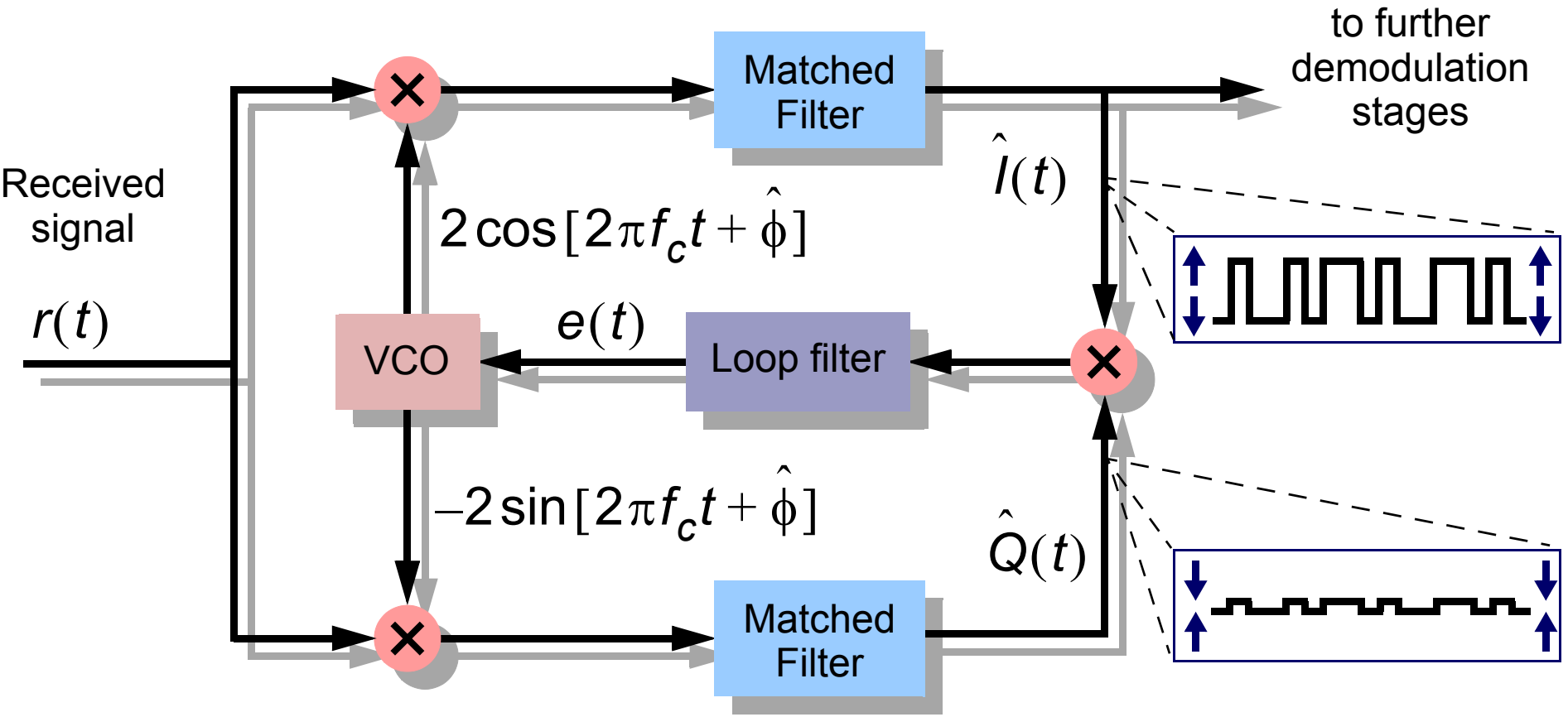


For small phase errors small angle approximation can be used so that $e(\varphi) \approx \varphi$ In a PLL this phase error estimate is then fed back to control the frequency of the VCO in an attempt to correct the phase.

Costas Loops

- Costas loops may be used to track suppressed-carrier amplitude modulated signals of the form:

$$r(t) = A(t)\cos(2\pi f_c t + \phi)$$



- Both in-phase and quadrature components are demodulated.

Notes:

Costas loops are also based on the orthogonality of the $\sin[\theta(t)]$ and $\cos[\theta(t)]$ functions. A signal of the form $A(t)\cos(2\pi f_c t + \phi)$ only contains power in the in-phase component. The received signal $r(t)$ can therefore be described as:

$$r(t) = I(t)\cos(2\pi f_c t + \phi) + Q(t)\sin(2\pi f_c t + \phi)$$

where $I(t) = A(t)$ and $Q(t) = 0$. In practice the VCO is never perfectly synchronized with the carrier therefore both *arms* of the costas loop receive some of the signal power as is demonstrated by the following equations:

$$\hat{I}(t) = A(t)\cos(\phi - \hat{\phi})$$

$$\hat{Q}(t) = A(t)\sin(\phi - \hat{\phi})$$

In a Costas loop the input to the loop filter is the product of these two signals:

$$\begin{aligned} e(t) &= A^2(t)\cos(\phi - \hat{\phi})\sin(\phi - \hat{\phi}) \\ &= \frac{1}{2}A^2(t)\sin(2(\phi - \hat{\phi})) \end{aligned}$$

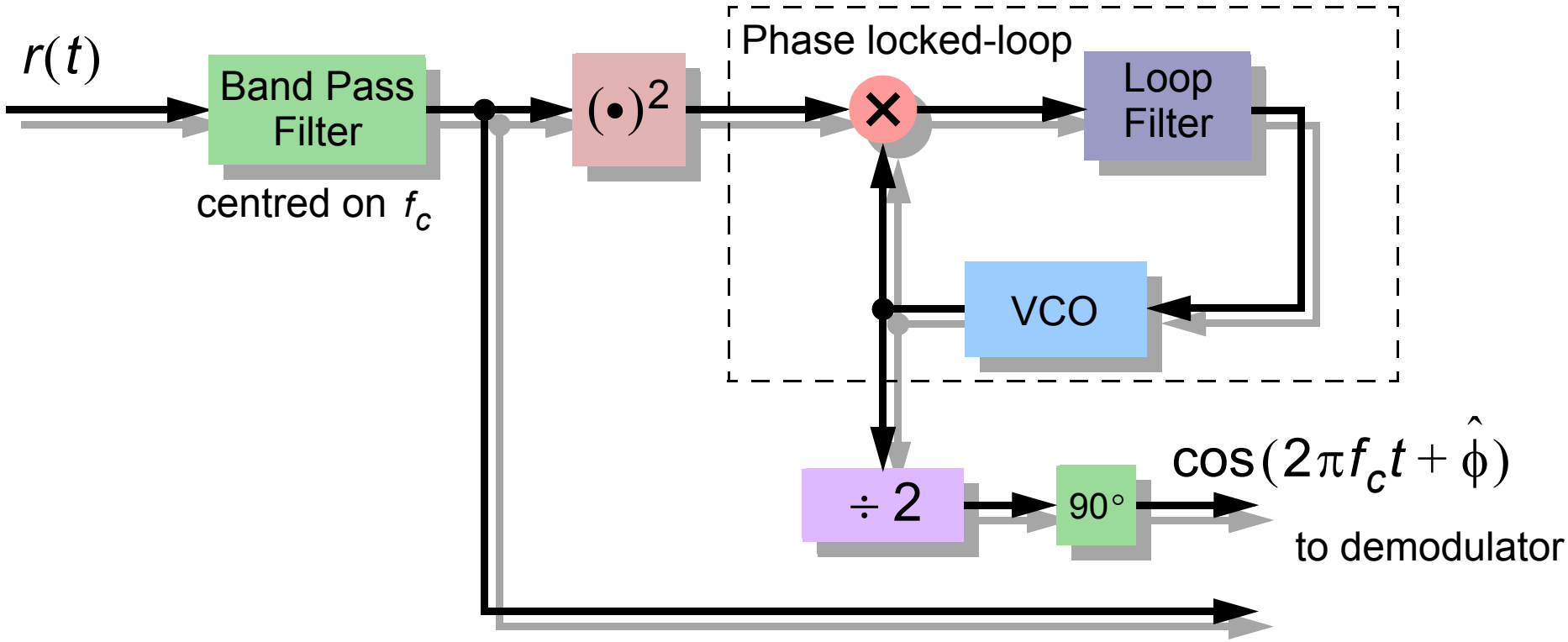
The loop filter then removes the any noise which comes from the term $A^2(t)$ so that

$$e(t) \approx \frac{1}{2}\overline{A^2(t)}\sin(2(\phi - \hat{\phi}))$$

It is also possible to sample the output of the matched filters at symbol rate (on the maximum effect points) and use these to derive phase error estimate in a similar way to that just described.

Squaring Loops

- Squaring loops are an alternative method of carrier recovery for suppressed-carrier amplitude modulated signals.
- Squaring the signal generates a signal component at twice the carrier frequency.



- This component can be tracked and used to derive an estimate of the carrier phase.

Notes:

Consider squaring the amplitude modulated signal $r(t)$ as follows:

$$r(t) = A(t)\cos(2\pi f_c t + \phi)$$

$$r^2(t) = \frac{A^2(t)}{2}[1 - \cos(4\pi f_c t + 2\phi)]$$

There is clearly a signal component at twice the carrier frequency. In a squaring loop a square law device is used to square the received signal. The $4\pi f_c$ component is then tracked by a standard PLL circuit. A replica of the true carrier frequency can then be derived.

The phase synchronizer circuits which have been shown here only represent the basic techniques which are used for closed loop carrier recovery. There are many other techniques which may be used to improve the performance of these devices.

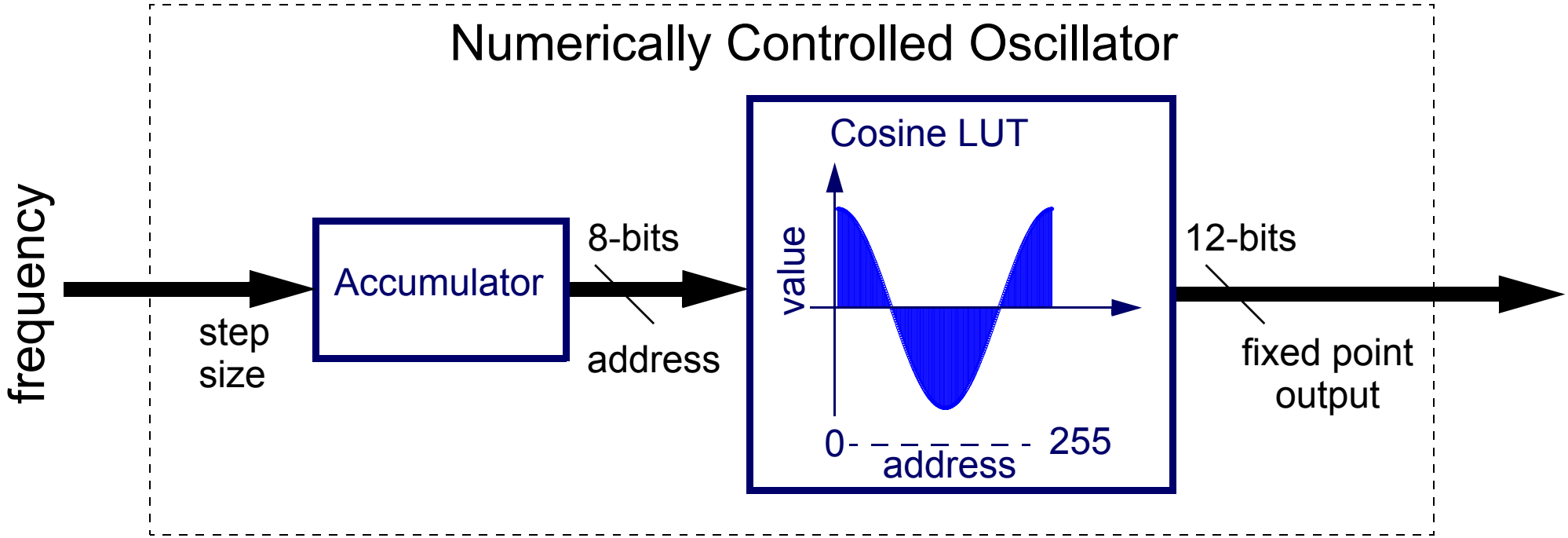
Digital PLL Implementations

- PLLs may be implemented digitally on ASICs, FPGAs and DSPs;
- A phase detector can be constructed using.....
  - Multipliers;
  - Filters (multipliers, adders, delays...).
- But how do we efficiently generate a sine or cosine signal with variable frequency?
- A digital device which can produce such a signal is known as a numerically controlled oscillator (NCO);
- We shall now describe how an NCO can be designed for an FPGA.

Notes:

Numerically Controlled Oscillators (I)

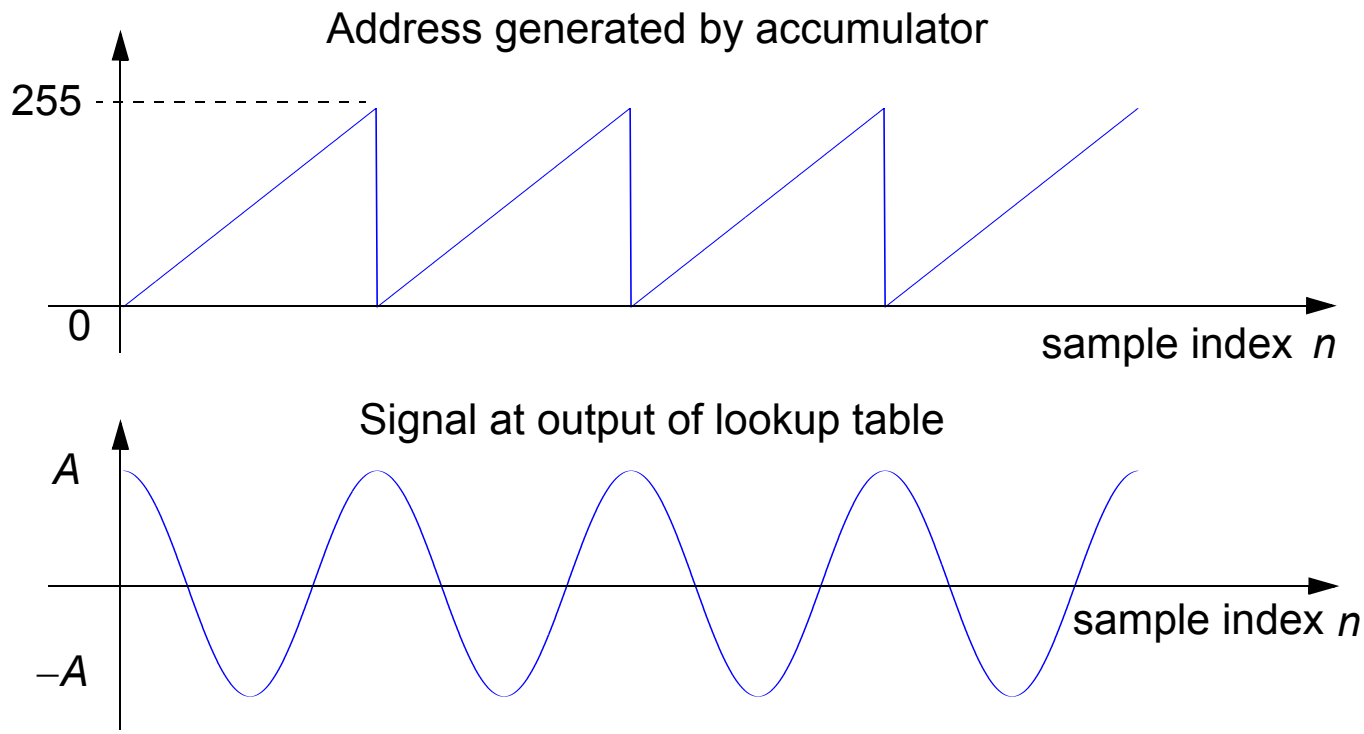
- A numerically controlled oscillator may be constructed using.....
  - A sine lookup table;
  - An accumulator to generate the address.



- The step size determines the frequency of the oscillator.

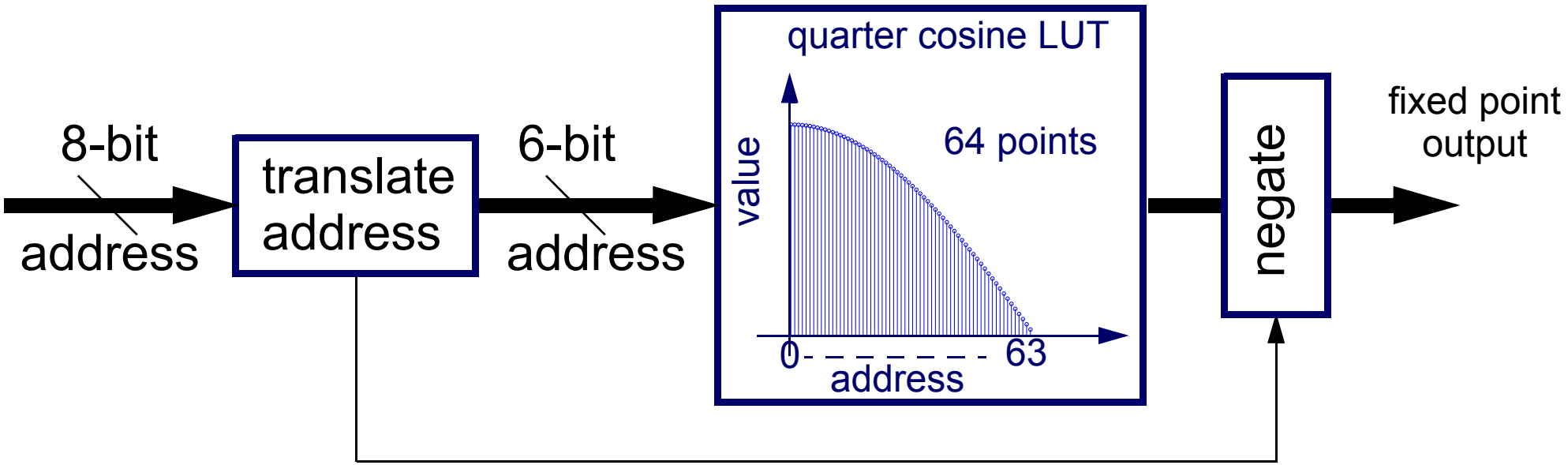
Notes:

The diagram below demonstrates how a sinewave is generated by applying a ramp signal to the input of the LUT. The ramp signal is produced by the accumulator (the digital equivalent of an integrator) which is essentially a counter with a variable step size. Since the counter is only 8-bit it wraps around to zero after passing address 255 thereby generating a periodic sine wave.



Numerically Controlled Oscillators (II)

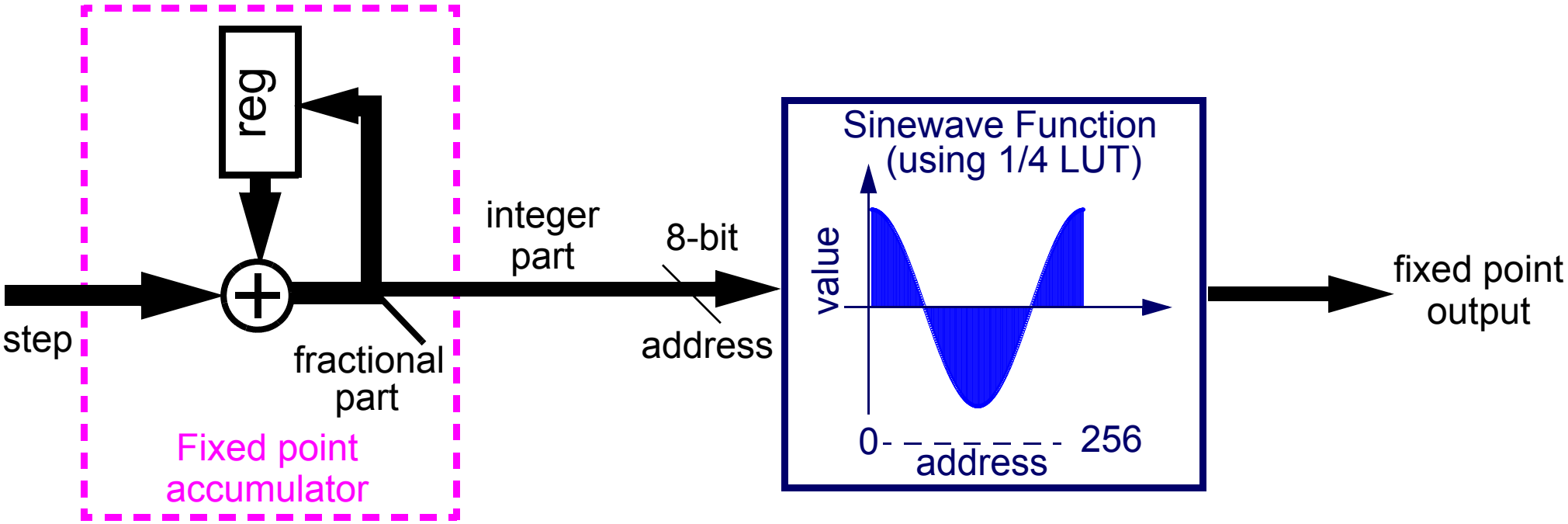
- The LUT can be reduced in size;
- Only the first quarter of the cosine wave is actually needed;
- Some straight forward logic is required to...
  - Translate the 8-bit address to a 6-bit address;
  - Determine whether or not to negate the output.



Notes:

Numerically Controlled Oscillators (III)

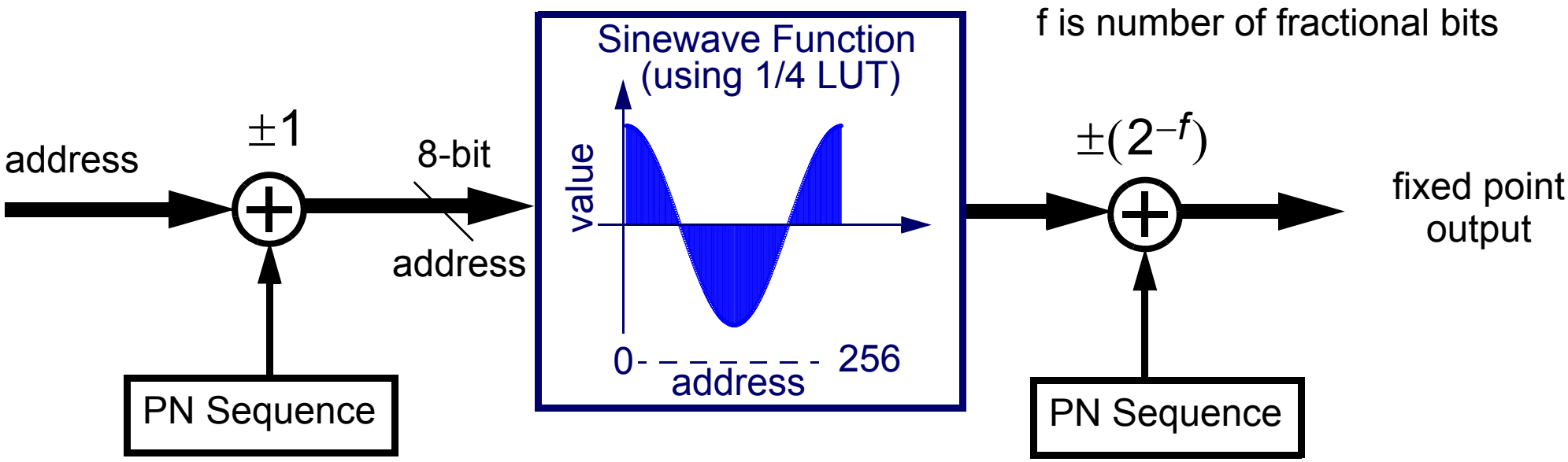
- The frequency resolution is improved with a fixed point accumulator;
- Therefore the step size can have a fractional part;
- The fractional part is discarded when accessing the LUT;



- However, the fractional digits are still accumulated so that the period of the resulting NCO signal corresponds to the step size.

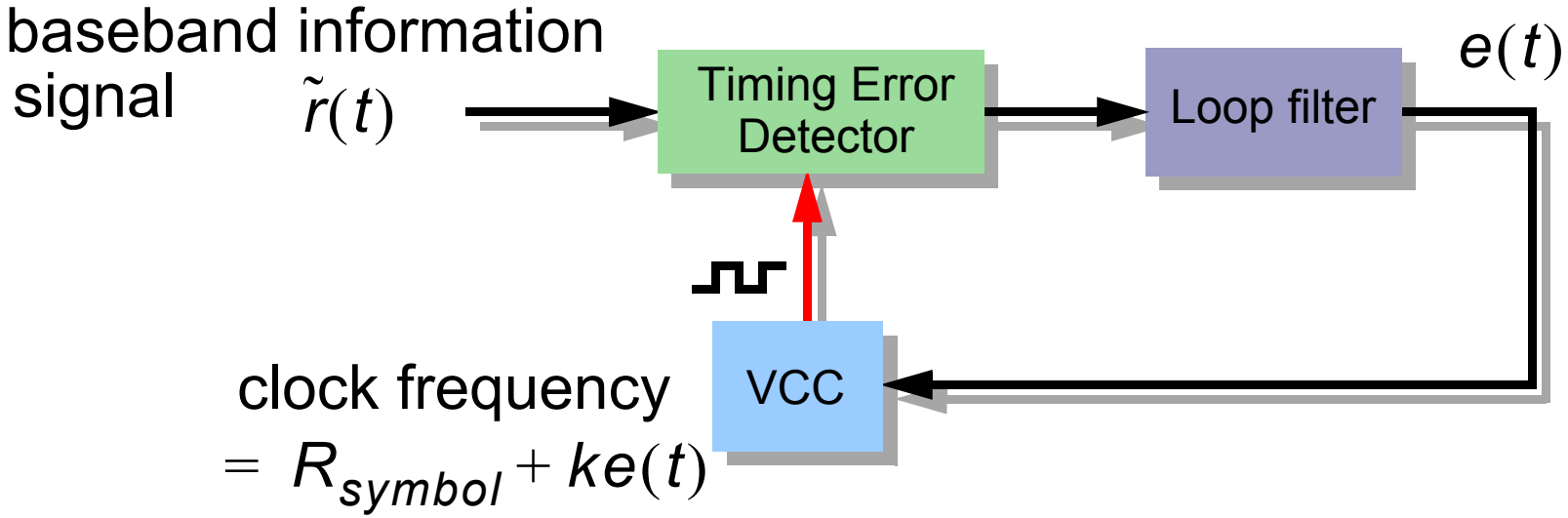
Notes:

In the above NCO implementation only the integer part of the accumulator output is used to select values from the lookup table. Because the phase angle of the sinewave also has finite resolution the resulting signal at the output may have unwanted harmonics. There are a number of ways to improve the performance of the oscillator. One is to use a PN sequence to dither both the phase angle (address) before applying it to the lookup table and also dithering the least significant bit of the function output. Alternatively it may be possible to use a comb filter to remove the unwanted harmonics.



Symbol Timing Recovery Loops

- Symbol timing recovery is achieved using the correlation statistics of the received signal;
- A timing error detector (TED) is used to derive an estimate of the timing error just as a phase error detector is used in a PLL;
- The timing error estimate may be filtered and used to control a Voltage-Controlled Clock (VCC).

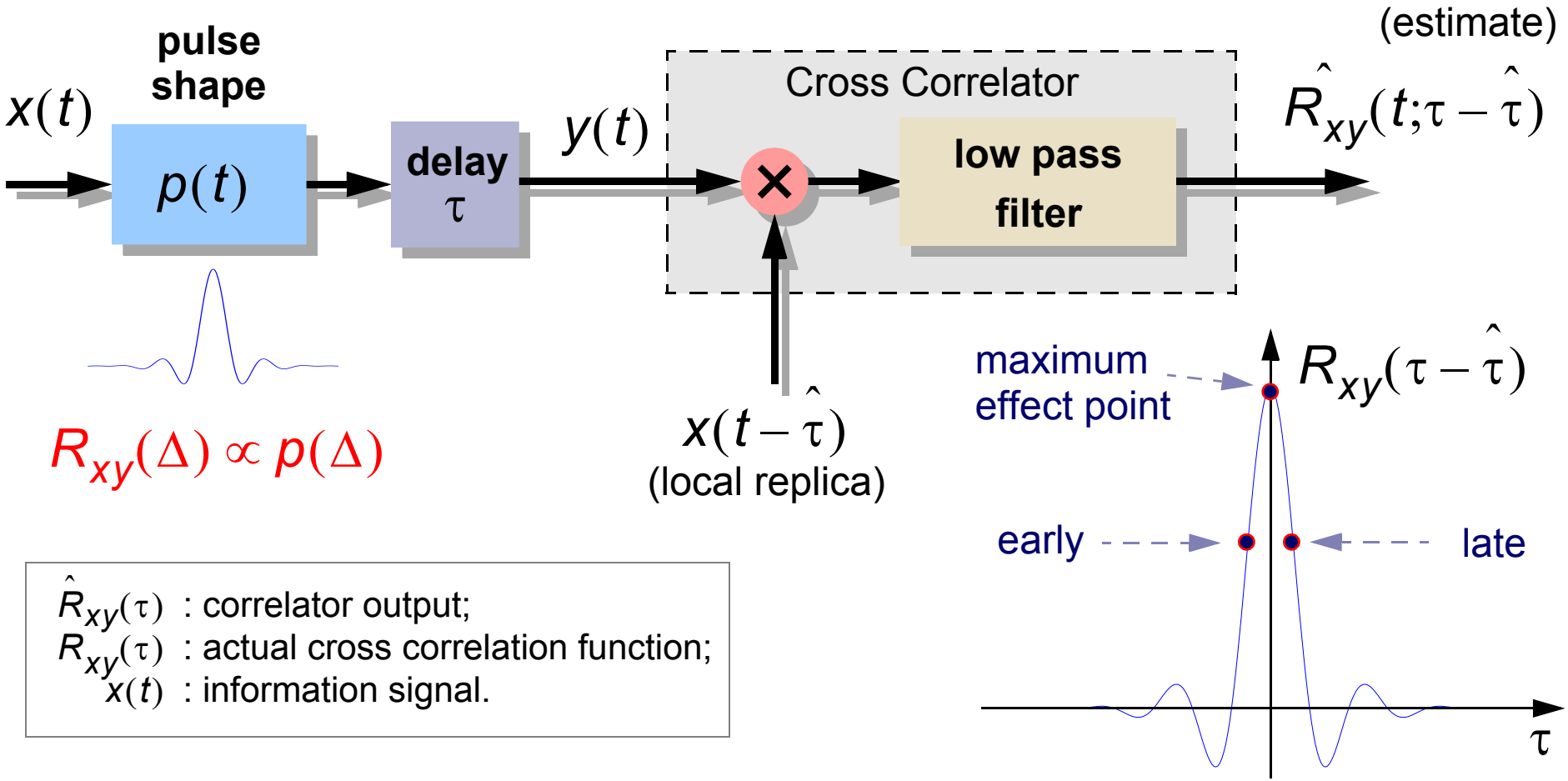


- Notice how similar this structure is to a PLL!

Notes:

Timing Error Detector (I)

- Use the effect of pulse shaping on statistics of the received signal to estimate the symbol timing error;
- Correlating the information signal (before pulse shaping) with the received signal reveals the pulse shape!

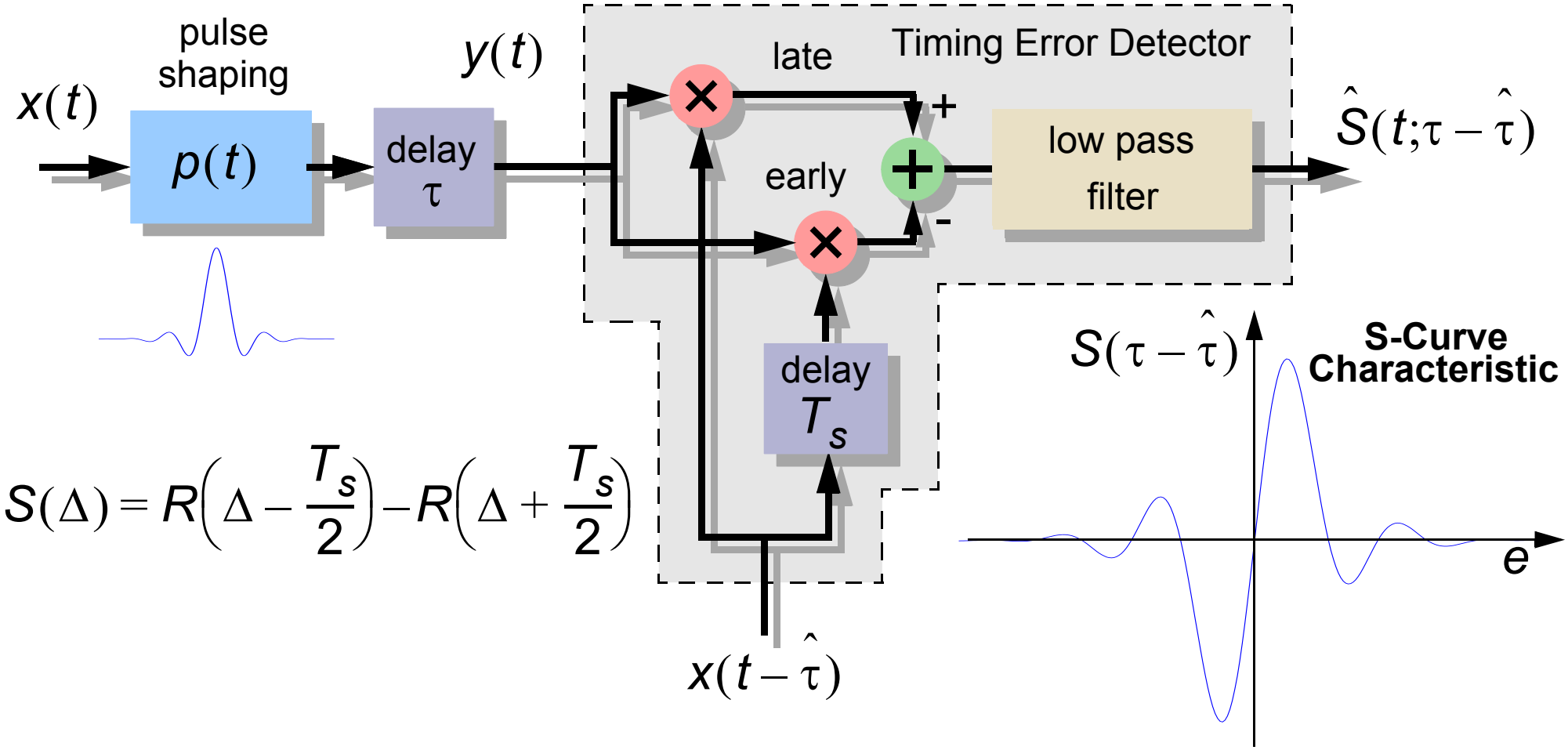


$\hat{R}_{xy}(\tau)$: correlator output;
$R_{xy}(\tau)$: actual cross correlation function;
$x(t)$: information signal.

Notes:

Timing Error Detector (II)

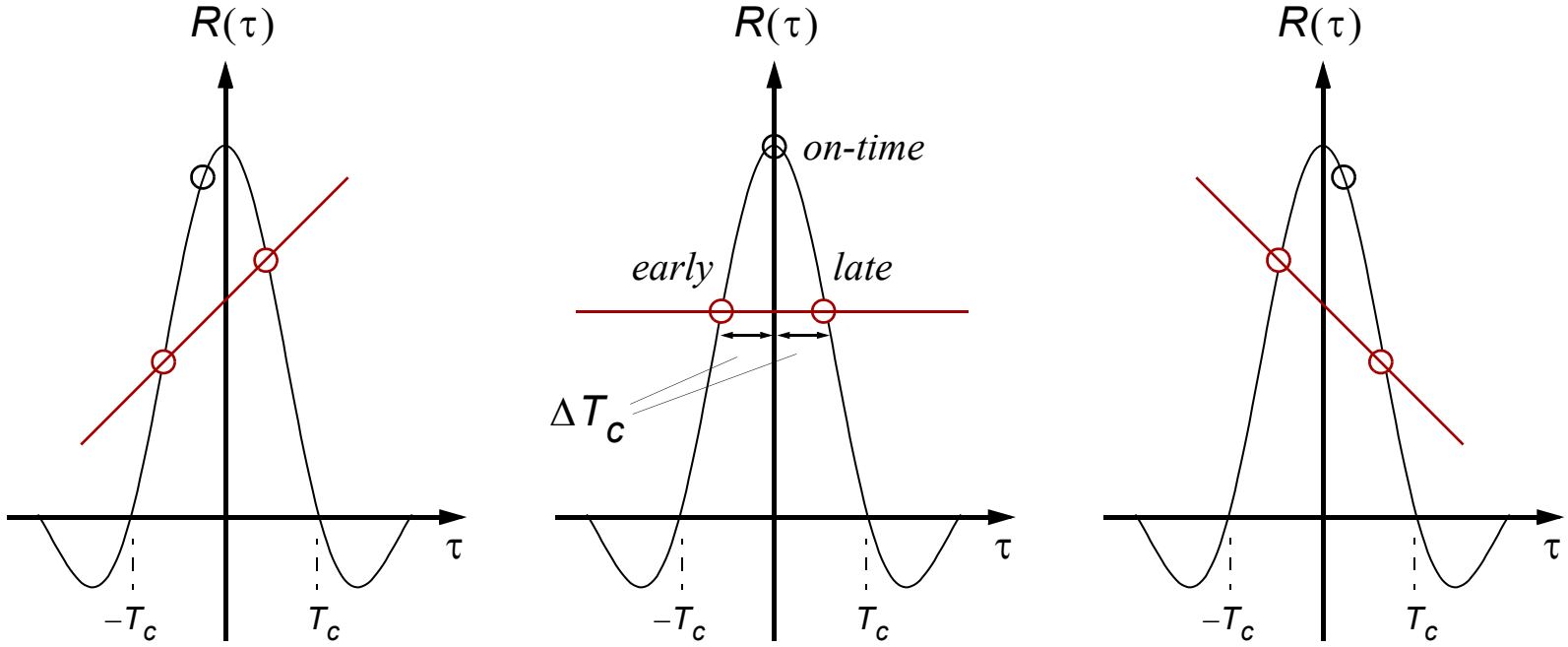
- Perform two correlation operations: one early and one late;
- Perfect synchronisation → early and late correlations balanced;



- Output of TED can then be used to control the symbol clock frequency.

Notes:

The diagram below demonstrates how a timing error is detected by measuring the gradient between the early and late sampling points on the autocorrelation function. Note how the sign of the gradient indicates in which direction the timing adjustment should be made to reduce the timing error.



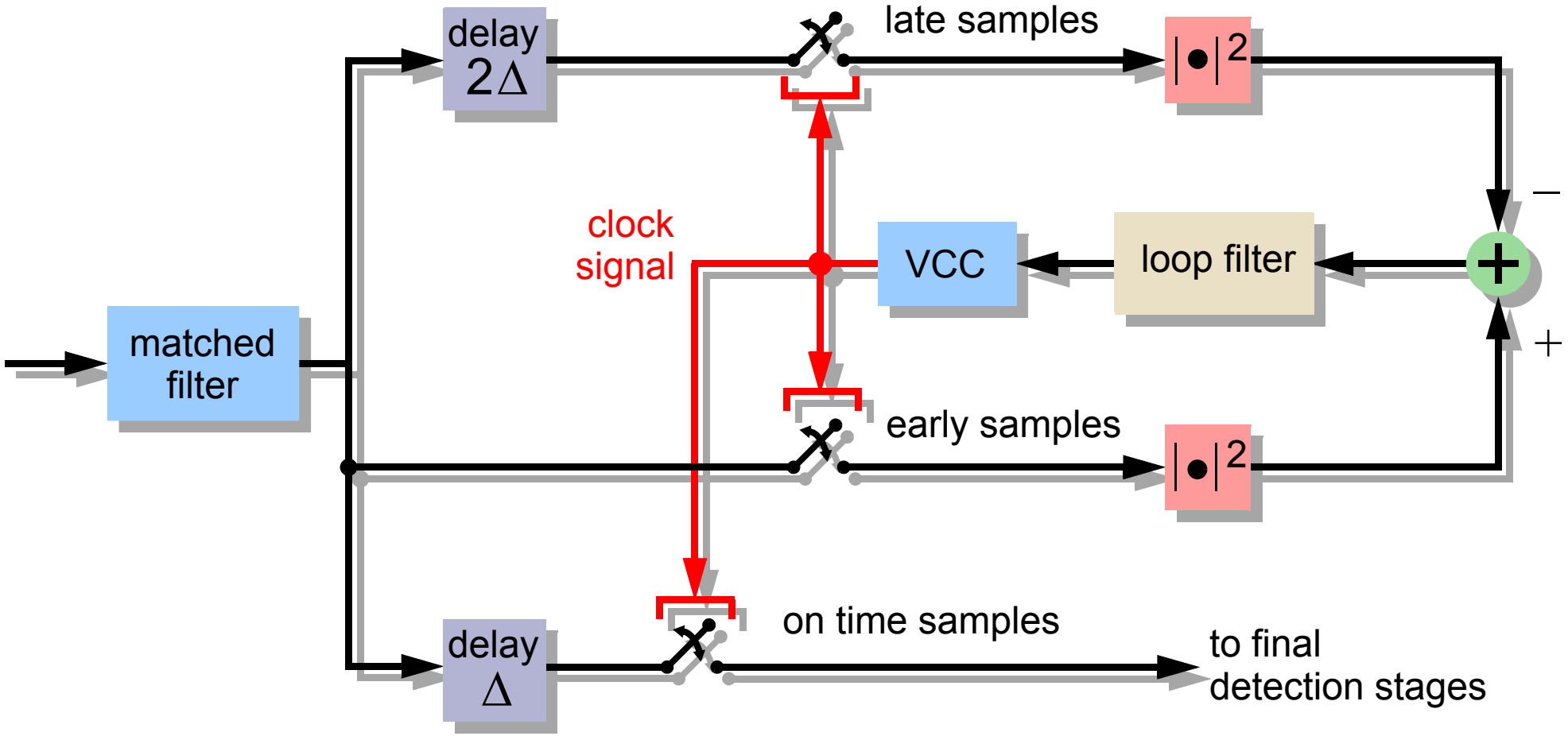
(a) negative timing error

(b) perfect timing

(c) positive timing error

Early Late Symbol Synchronizer

- In general the transmitted symbol sequence is unknown;

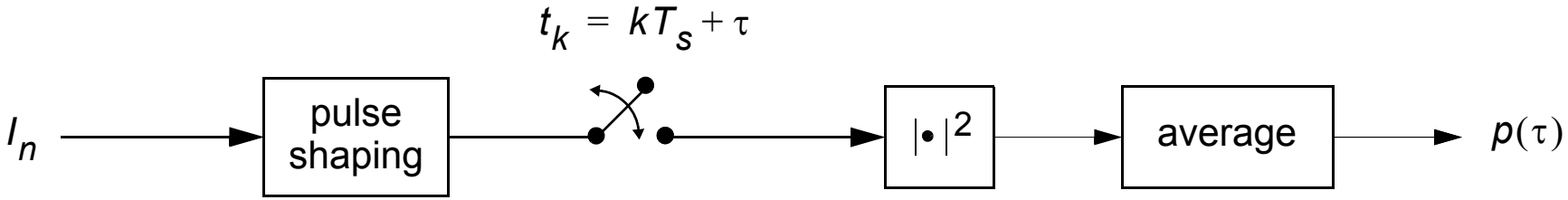


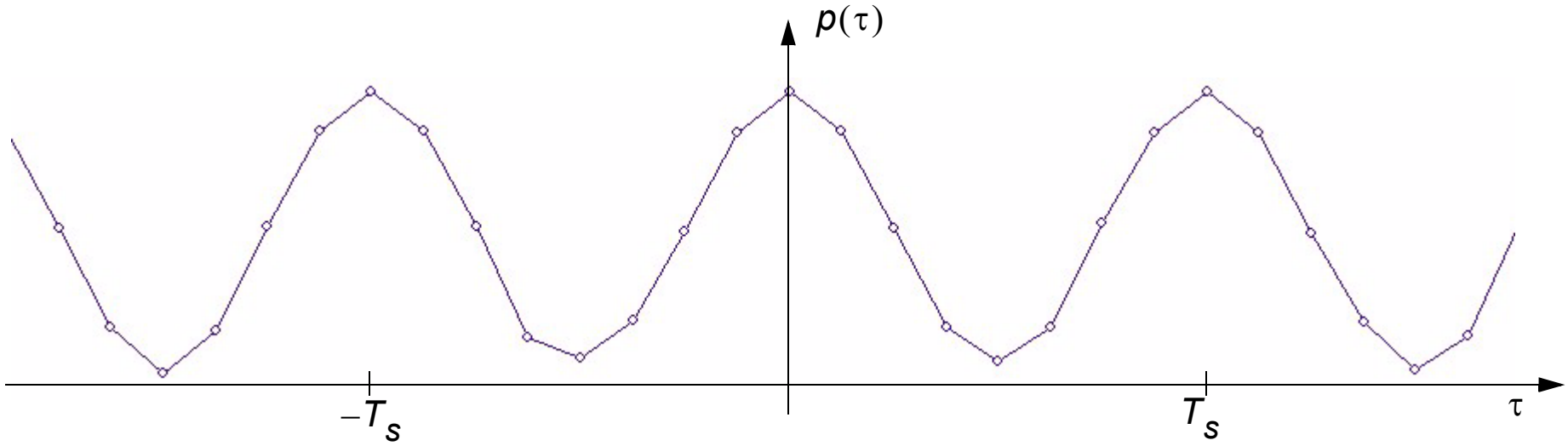
- Solution: sample signal Δ seconds early and Δ seconds late;
- Try to balance the mean square of these samples.

Notes:

The early late symbol synchronizer shown in the previous slide is an example of a non-decision directed device. This means that it does not assume any knowledge of the transmitted symbols. It simply relies on known properties of the signal statistics.

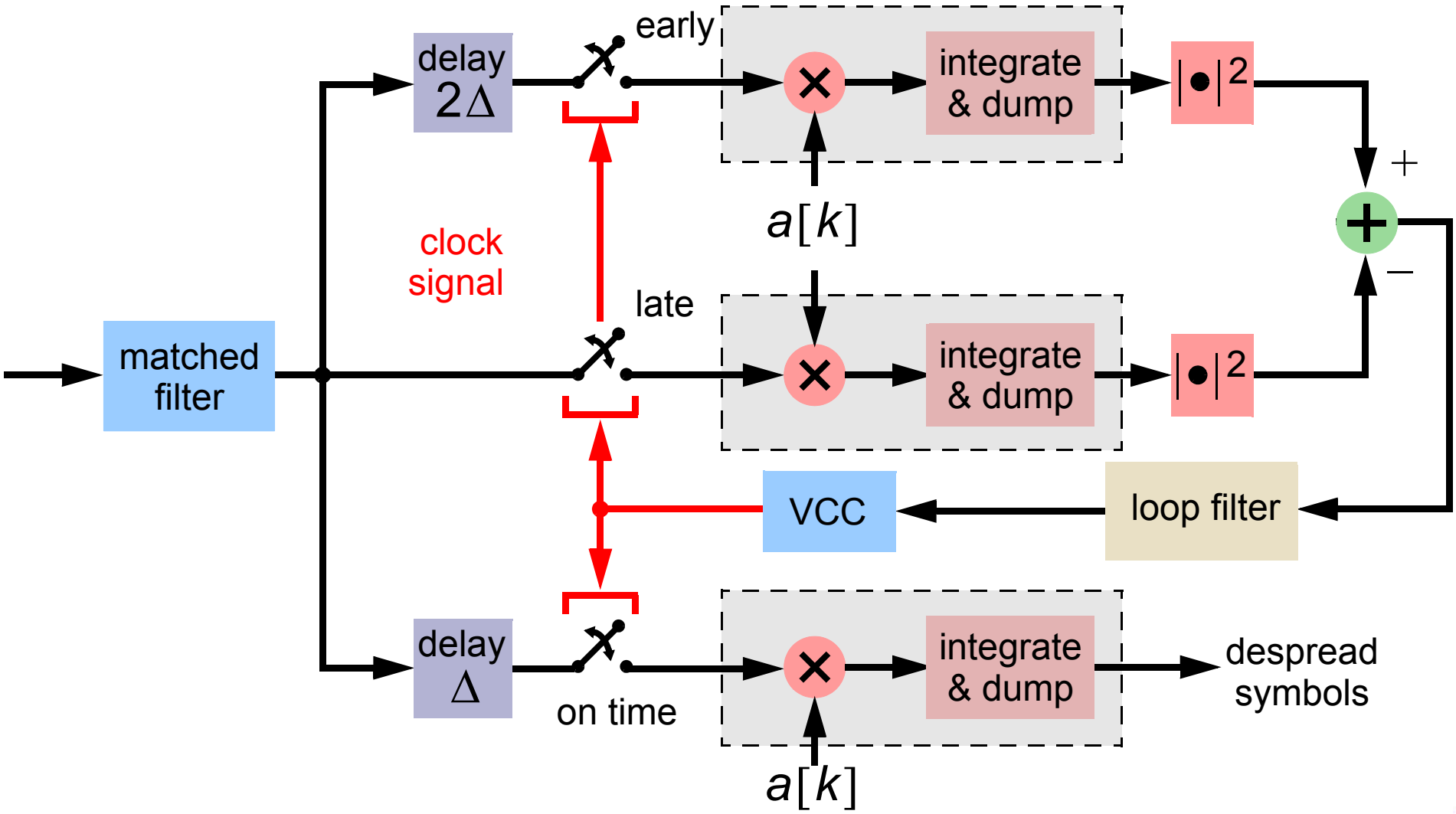
A pulse shaped sequence of data symbols has cyclostationary statistics. This essentially means that the signal statistics are the same at time instants nT_s, $(n+1)T_s$, $(n+2)T_s$, …. However the statistics of the signal at time instants nT_s and $nT_s+\tau$ (where τ is not an integer multiple of T_s) are not necessarily the same. Take the following system as an example..... The statistics allow the maximum effect points to be identified without knowledge of the data symbols!





Spread Spectrum Delay-Lock Loop

- Direct-sequence spread-spectrum systems employ delay-lock loops to track the chip timing parameters;



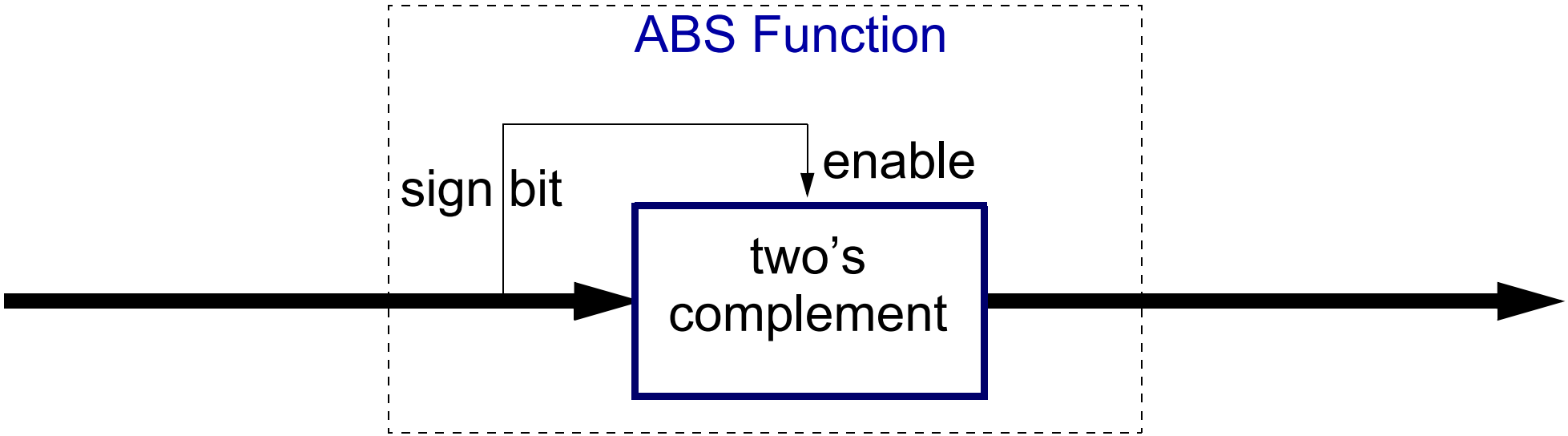
Notes:

Delay-Lock Loop Optimisation

The delay-lock loop can be efficiently implemented on an FPGA. Firstly one should note that the spreading sequence consists of 1s and -1s therefore no explicit multiplication is actually required in the despreader. The despreader only requires addition. Secondly the same despreader can be used to decode the early late and on-time signal components thereby reducing the size on the device. Thirdly a spread spectrum receiver typically requires a large number of despreaders to decode multiple channels or users and decode several multipath components in a RAKE receiver. Depending on the speed of the FPGA relative to the chip rate the same despreader hardware could also be used to decode multiple channels or be distributed across multiple DLLs.

Taking the Magnitude

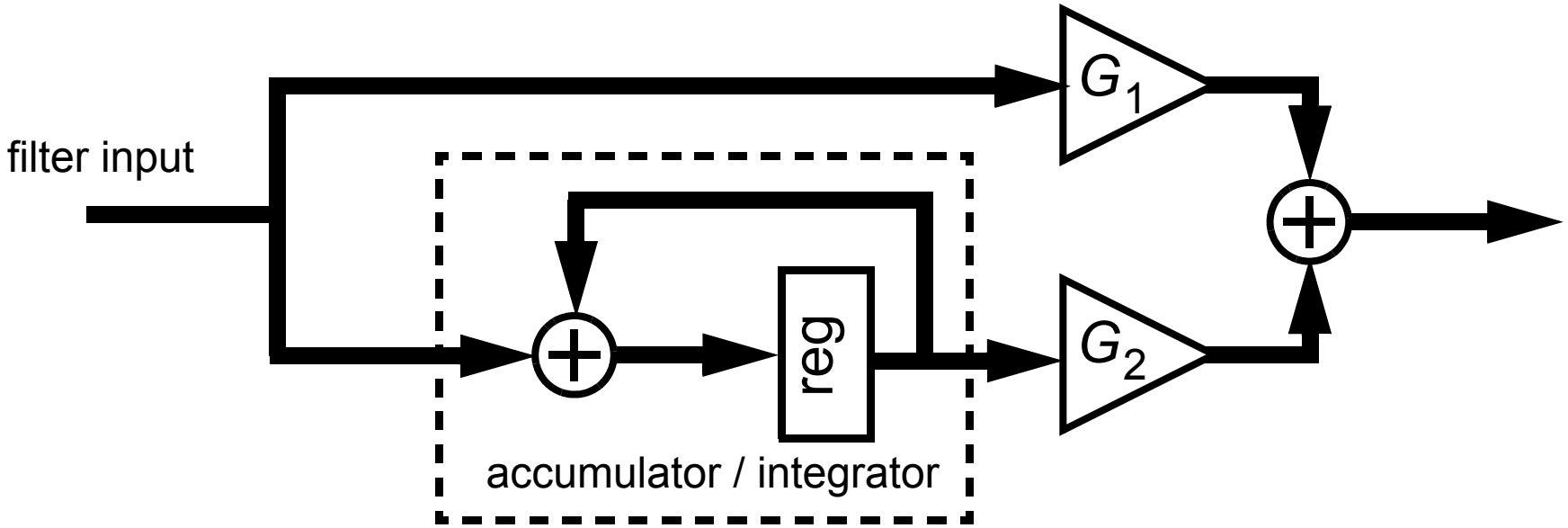
- For many applications the square law detector can be replaced with an absolute value function;
- In hardware a square law detector requires multiplication;
- Taking the absolute value of a real number involves taking the two's complement if the sign bit is set.
- Two's complement requires only a full adder and a bitwise negation.



Notes:

Loop Filters

- Loop filters are used in carrier phase and symbol timing recovery loops;
- The loop filter determines the order of the loop;
- A first order loop uses the filter below with $G_2 = 0$ and G_1 set to an appropriate non-zero value:

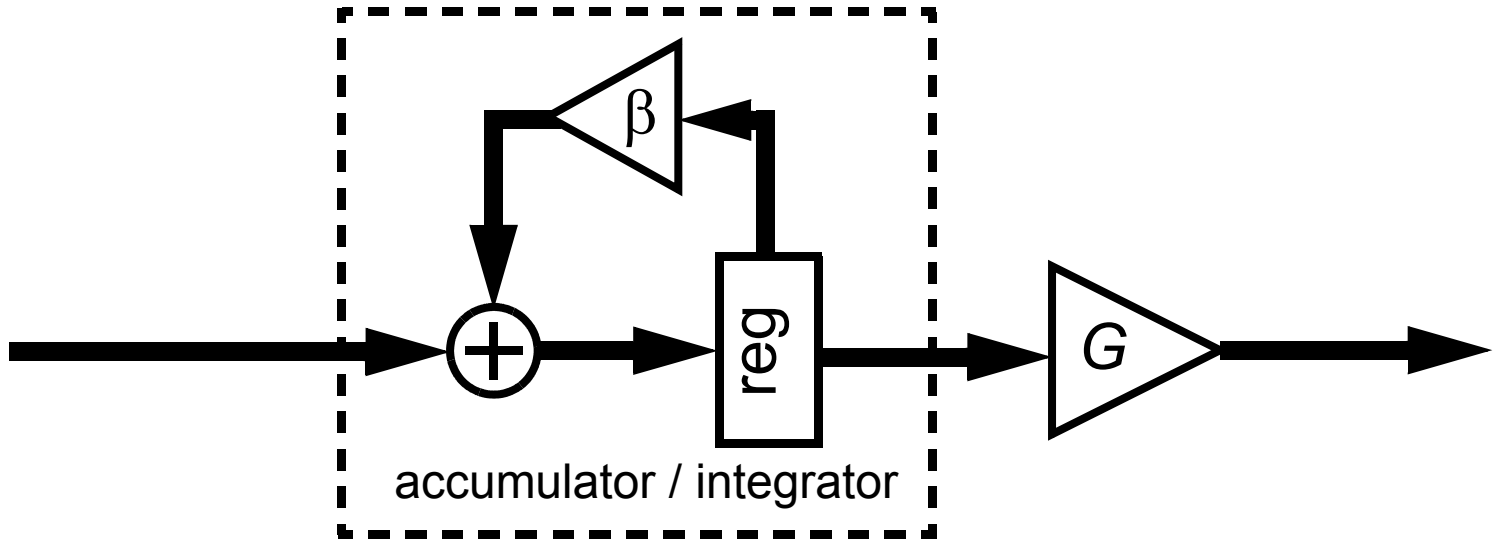


- For a second order loop G_2 should also be set to a non-zero value.

Notes:

Second order loops are frequently used in tracking devices. A second order carrier-phase tracking loop can track a constant change in phase (i.e. a frequency offset) with no error (when there is no noise). In practice there is always a slight frequency offset between the transmitter and the receiver due to clock inaccuracies. In addition a doppler shift may occur, for example, in a mobile communication system where a handset is an a car moving at a constant speed. This will cause an additional frequency offset which will vary with the speed of the vehicle.

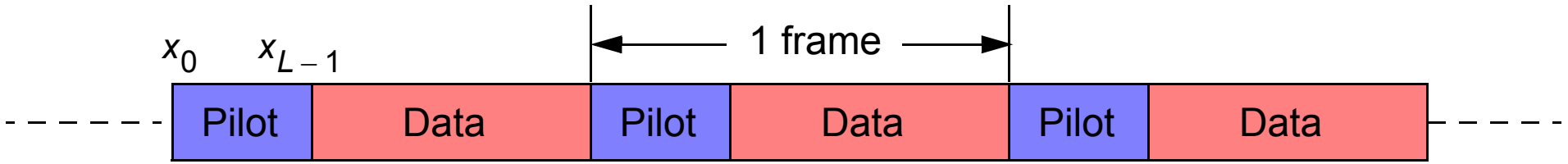
Another possibility is to use a one-pole filter, which is also known as a leaky integrator.The coefficient β is



normally set to a value between 0 and 1 otherwise the filter becomes unstable. Values of β which are close to (but not quite equal to 1) are quite common. In such cases the multiplier can be efficiently implemented by setting the gain to be a number such as $1-2^{-n}$. For example setting $n = 4$ results in $\beta = 0.9375$. This can then be implemented very easily using fixed point shift and add operations.

Pilot Symbol Provision

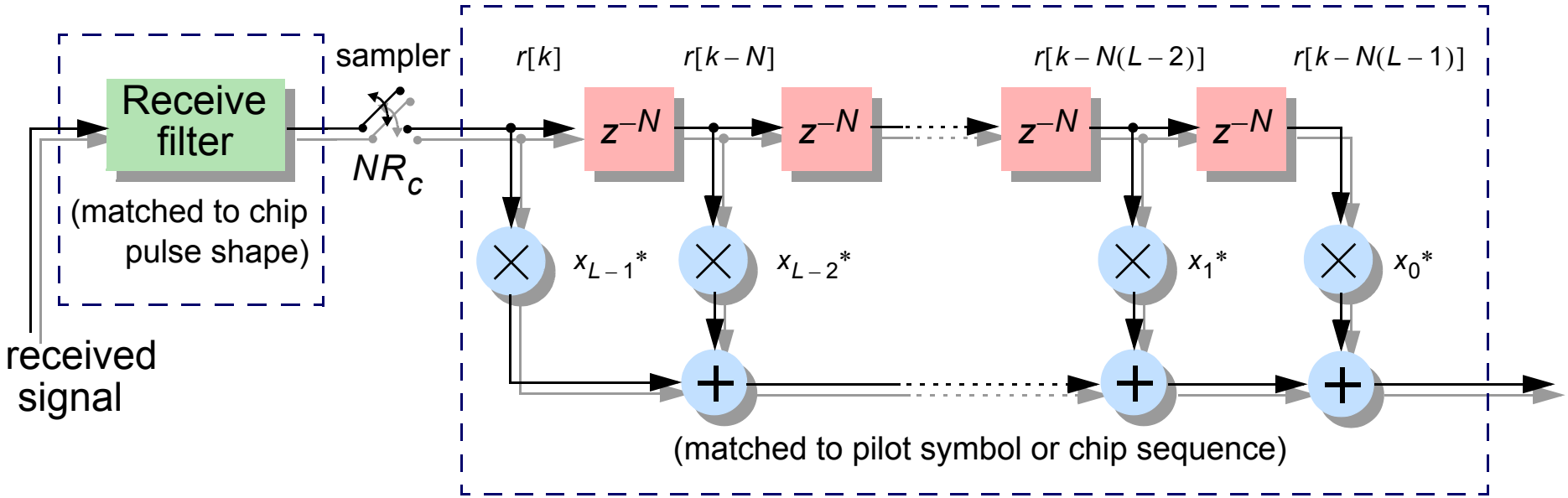
- Communication system designs usually provide pilot symbols;
- These are used for a variety of purposes including
    - Carrier phase estimation and correction;
    - Active gain control;
    - Initial frame synchronization;
    - Channel estimation.
- For example pilot symbols may be periodically inserted in a frame structure as shown below.....



Notes:

Frame Synchronization

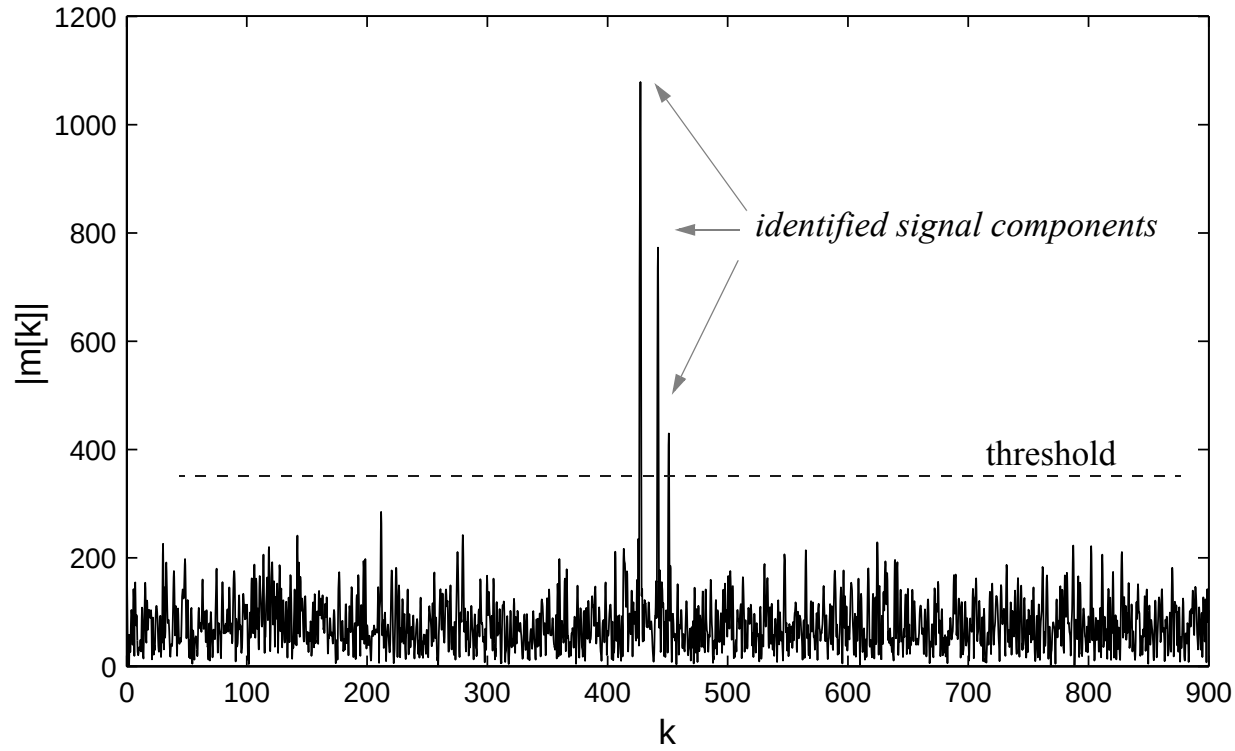
- Frame synchronization can be achieved using a special matched filter such as the one shown below;
- This filter measures the level of correlation between a portion of the received signal and the pilot sequence;



- Each output sample represents the correlation level at a different time instant.

Notes:

As the pilot symbols pass through the matched filter an estimate of the channel is generated at the output of the filter. Spikes are produced when there are significant levels of correlation between the filter coefficients (pilot symbols) and the received signal. An example of the signal magnitude at the filter output is shown below. A threshold can be applied to decide which spikes actually correspond to signal components. The receiver can then synchronize to these paths and track changes in their delays. When a spike occurs the receiver knows that it is at the start of a frame.



Again we should note that since the filter coefficients may be taken from a relatively small finite set of values. For example in BPSK the coefficients are either 1 or -1. This allows for substantial optimization.

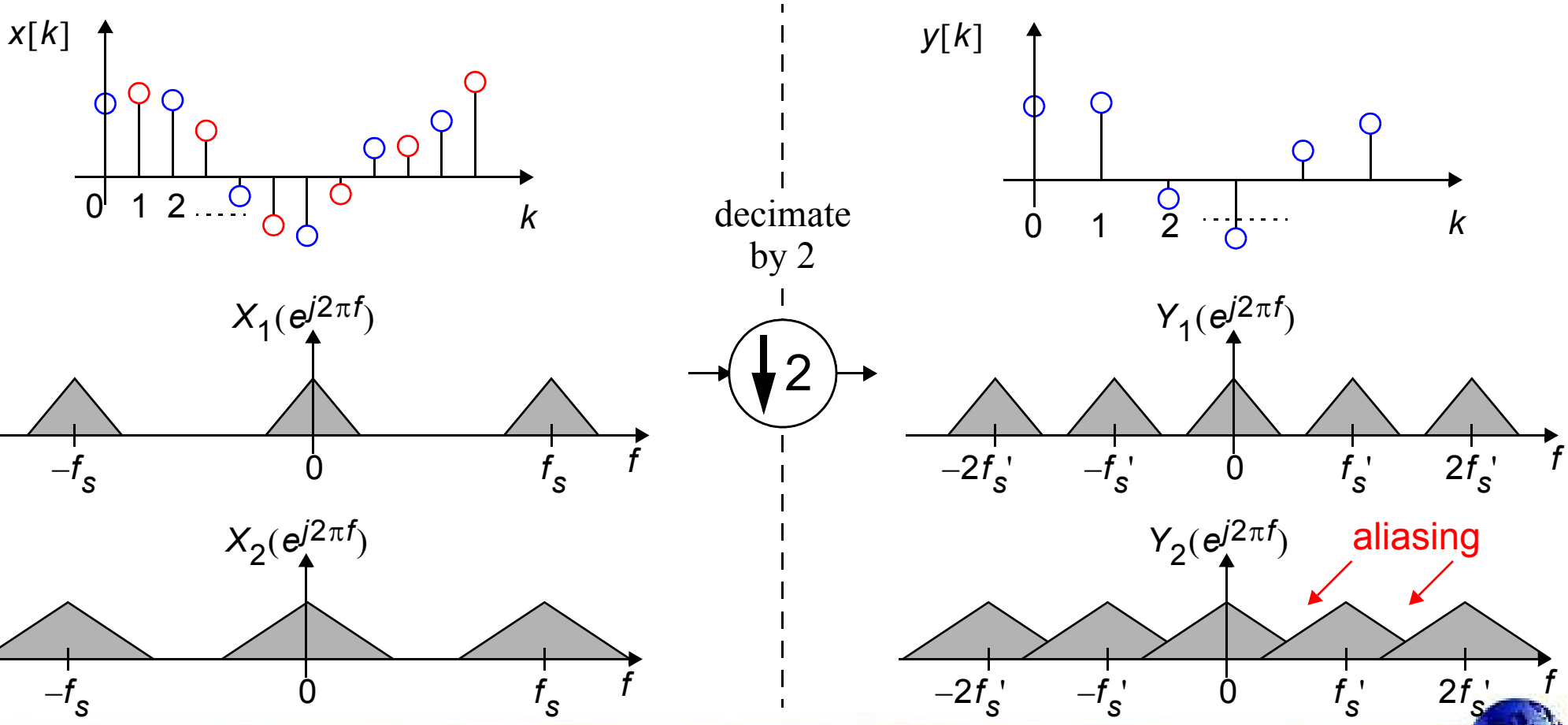
Sampling Rate Conversion

- Changing the sampling rate of a signal requires careful consideration for what happens as a result in the frequency domain;
- In most cases some type of filtering is required to constrain the frequency content of the signal;
- Sampling rate changes by an integer factor are relatively straight forward:
  - **Increasing** the sampling rate by an integer factor requires expansion followed by image rejection filtering;
  - **Decreasing** the sampling rate by an integer factor requires bandlimiting followed by decimation;
- Non-integer sampling rate conversions are more difficult to achieve.

Notes:

Reducing the Sampling Rate (I)

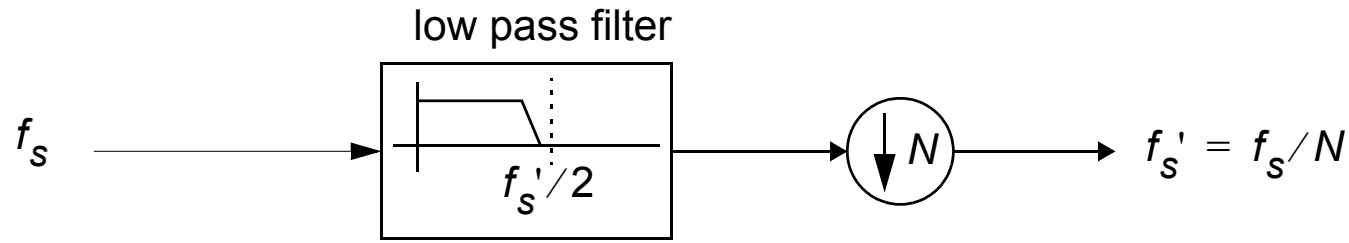
- The frequency content of a sampled signal is periodic with the sampling frequency - denote original sampling frequency as f_s;
- Decimating the signal by a factor of N introduces new images at multiples of the new sampling frequency $f_s' = f_s / N$:



Notes:

Decimating a signal by a factor of N causes new images of the signal's spectrum to appear at multiples of the new sampling frequency. If the signal is not sufficiently bandlimited then the aliasing can occur. When this happens the frequency components which contribute to the aliasing cannot be separated and in many cases this is undesirable. In general a real valued signal must be bandlimited to $f_s'/2$ before either sampling or downsampling where f_s' is the new sampling frequency.

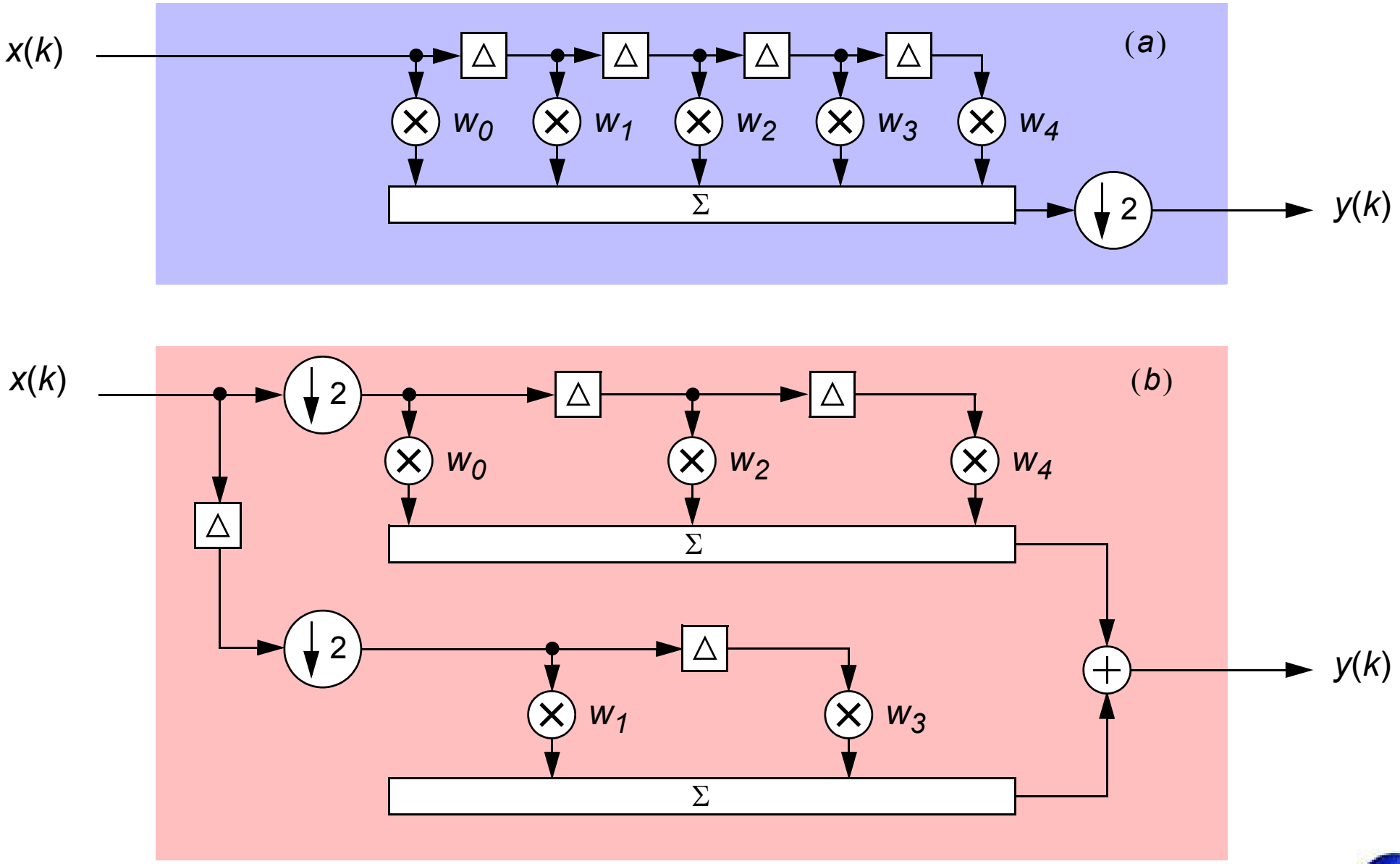
This can be achieved by filtering the signal with a low pass filter as shown in the diagram.



An important property of this filter structure which can be exploited is that only every N^{th} sample at the output is actually retained. For example when decimating by a factor of 8 only one in 8 samples at the filter output is actually kept by the decimator. This means that $7/8^{ths}$ of the actual processing is a complete waste!

Reducing the Sampling Rate (II)

- The following two downsampler implementations are equivalent:



Notes:

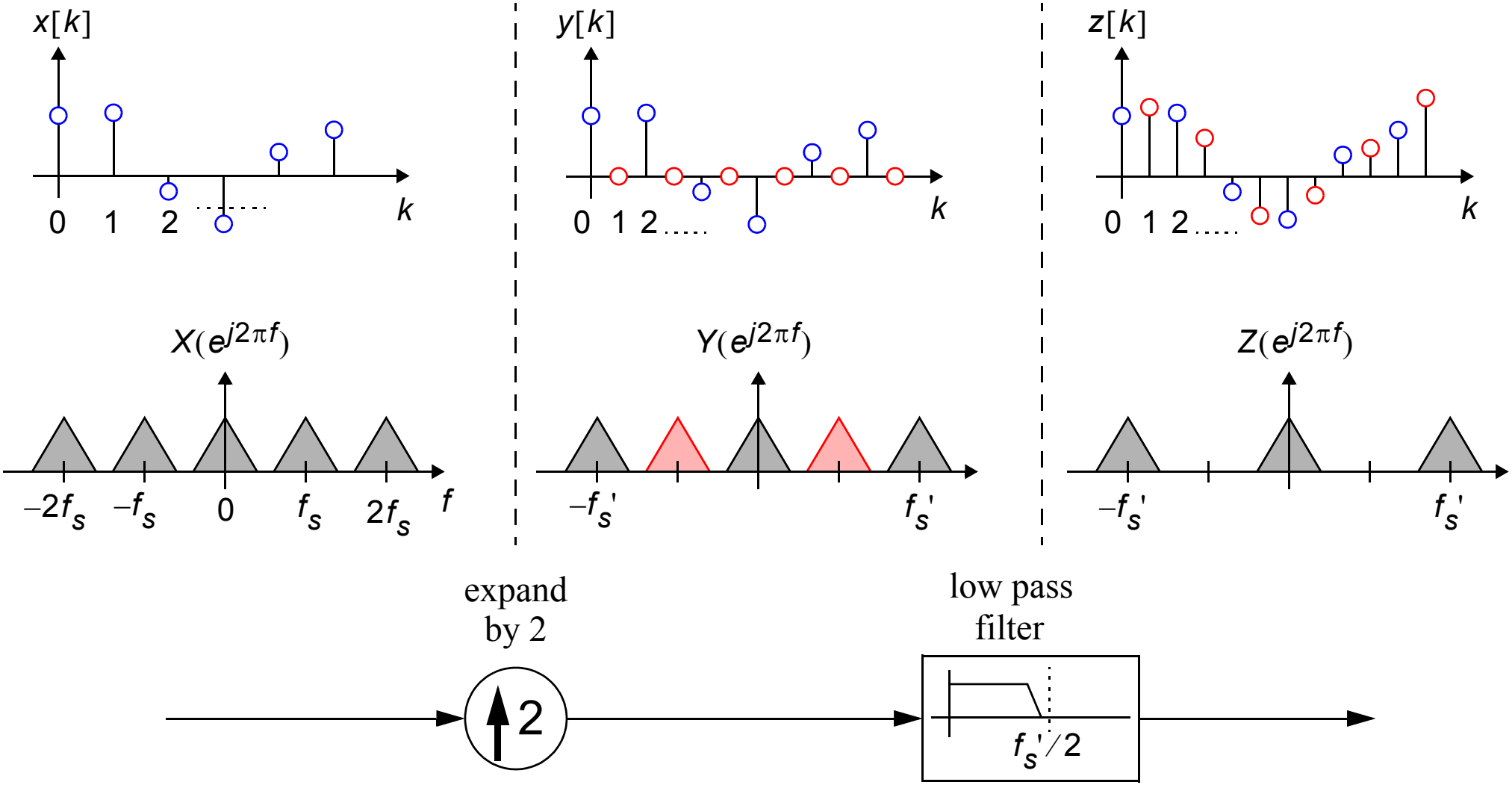
The downsampling structures in the previous slide are equivalent in terms of functionality however (b) is about twice as efficient as (a).

The first point to note is that in (b) the filter has been split into two parts. The top part contains the even coefficients and the bottom part contains the odd coefficients. This is known as a **polyphase** filter. In this case because we are decimating by a factor of 2 we split the filter into 2 polyphase components. In general for decimation by a factor of *N* we would split the filter into *N* polyphase components.

The second point to note is that samples are clocked in and out of structure (a) twice as fast as in the polyphase filters in structure (b) however the total number of multipliers in structure (b) is the same as in (a). This leads to a substantial reduction in complexity in particularly when the decimation factor is high.

Increasing the Sampling Rate (I)

- Expanding a signal by a factor of *N* leaves unwanted images of the signal at multiples of the old sampling frequency:

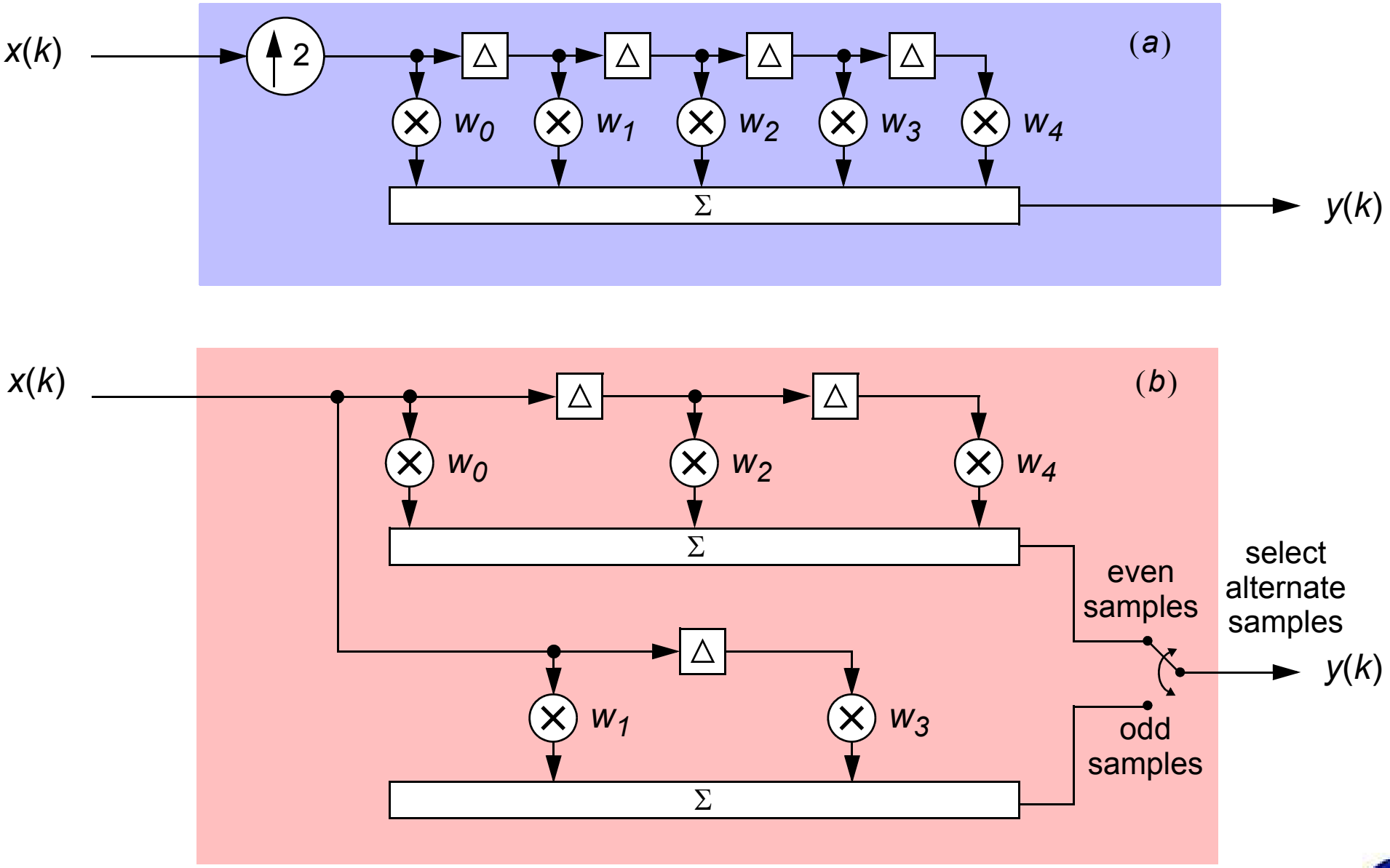


Notes:

The combination of expansion followed by digital filtering can also be exploited using a polyphase structure. In the example in the previous slide half of the samples applied to the filter are guaranteed to be zeros! As the zeros pass through the filter they waste half of the multiplication and addition operations since any number multiplied by zero is just zero.

Increasing the Sampling Rate (II)

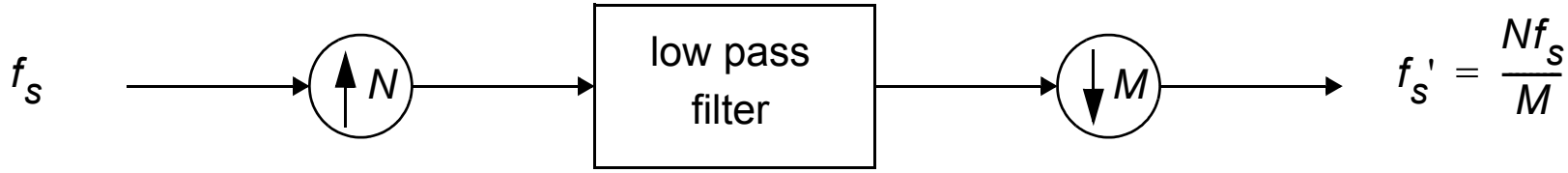
- The following two upsampler implementations are equivalent:



Notes:

So the sampling rate can be increased and decreased with some fancy filtering operations as shown above. What happens however when we wish to alter the sampling frequency by a non-integer factor?

Consider changing the sampling rate by a rational fraction $R = N/M$. This can be achieved using a combination of upsampling and downsampling operations. In order to avoid losing any frequency components which we wish to retain it is necessary to first upsample by N and then downsample by M.



In this case the low pass filter is used to reject the unwanted images which remain at multiples of the original sampling frequency and simultaneously bandlimit the signal to avoid aliasing in the decimation stage. Therefore the cut off frequency of the filter must be set to just under the lowest sampling rate (i.e. select the lowest of the in put rate f_s and the output rate f_s').

For sampling rate changes which cannot be represented as a (realizable) rational fraction it is possible to upsample to a relatively high sampling rate. *Zero order* interpolation (i.e. simply choosing the nearest value) can then be used to downsample. Alternatively more accurate linear or quadratic interpolation could be used to perform the subsequent downsampling operation.

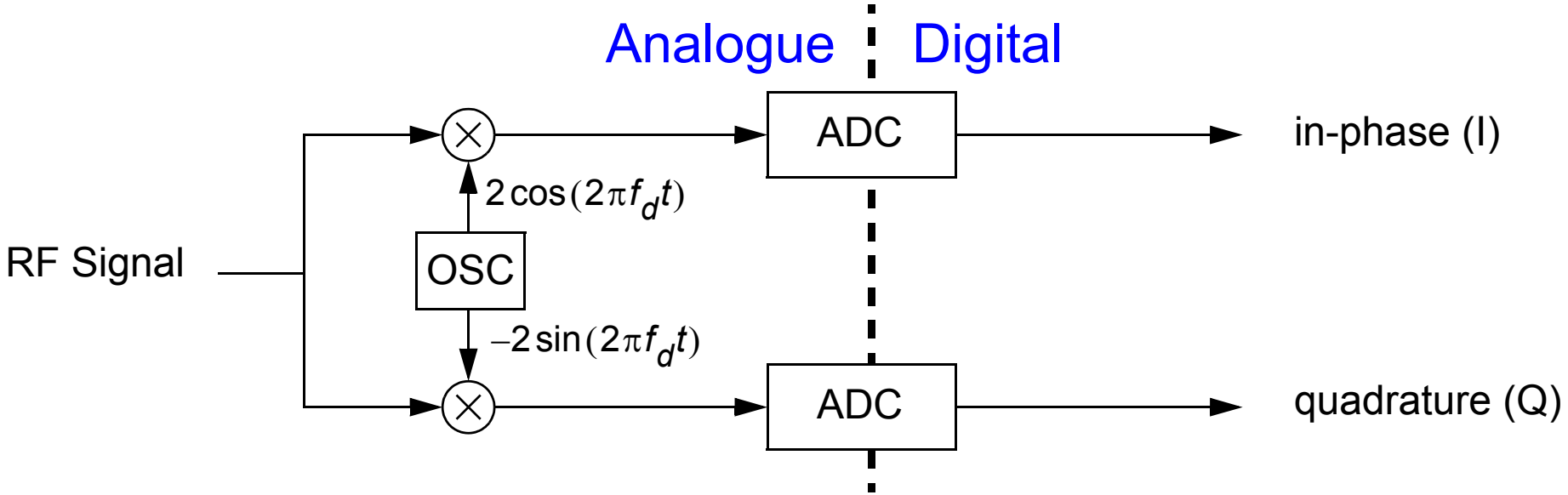
Downconversion (I)

- Many signal processing applications involve bandpass signals which reside at high frequencies;
- Representing high frequencies digitally requires high sampling rates;
- Solution: downconvert the bandpass signal to a lower frequency position;
- Cost of processing the signal digitally is substantially reduced;
- The two dominant downconversion techniques are:
  - mixing;
  - direct downconversion.

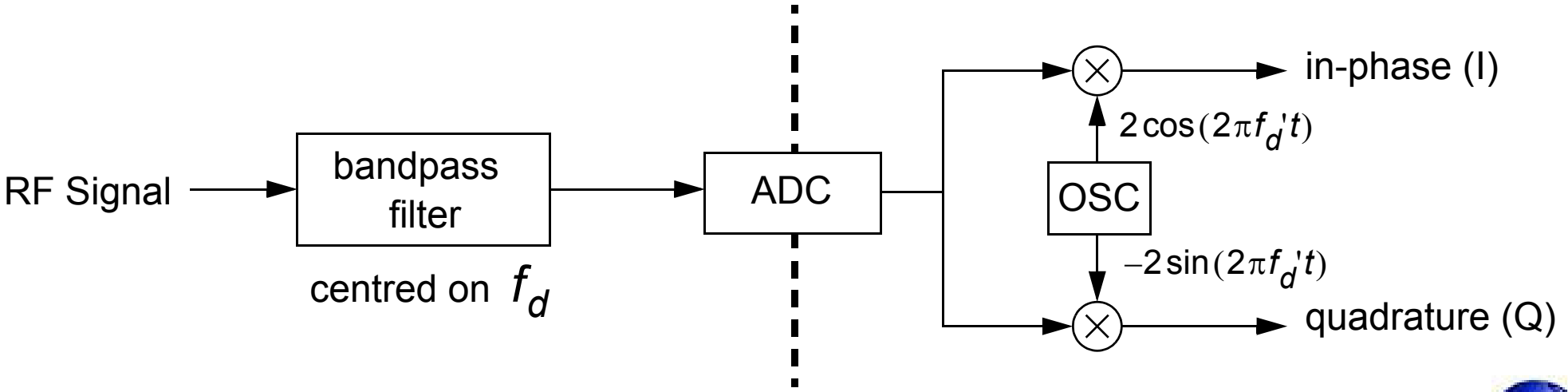
Notes:

Downconversion (II)

- Mixing involves multiplication by sinusoids followed by filtering:



- Direct downconversion: direct sampling of passband signal:



Notes:

In the mixing example in the previous slide multiplication of the signal by in-phase and quadrature frequency components causes the signal to shift in frequency so that it is centred on zero hertz. The resulting digital signal is called a baseband signal.

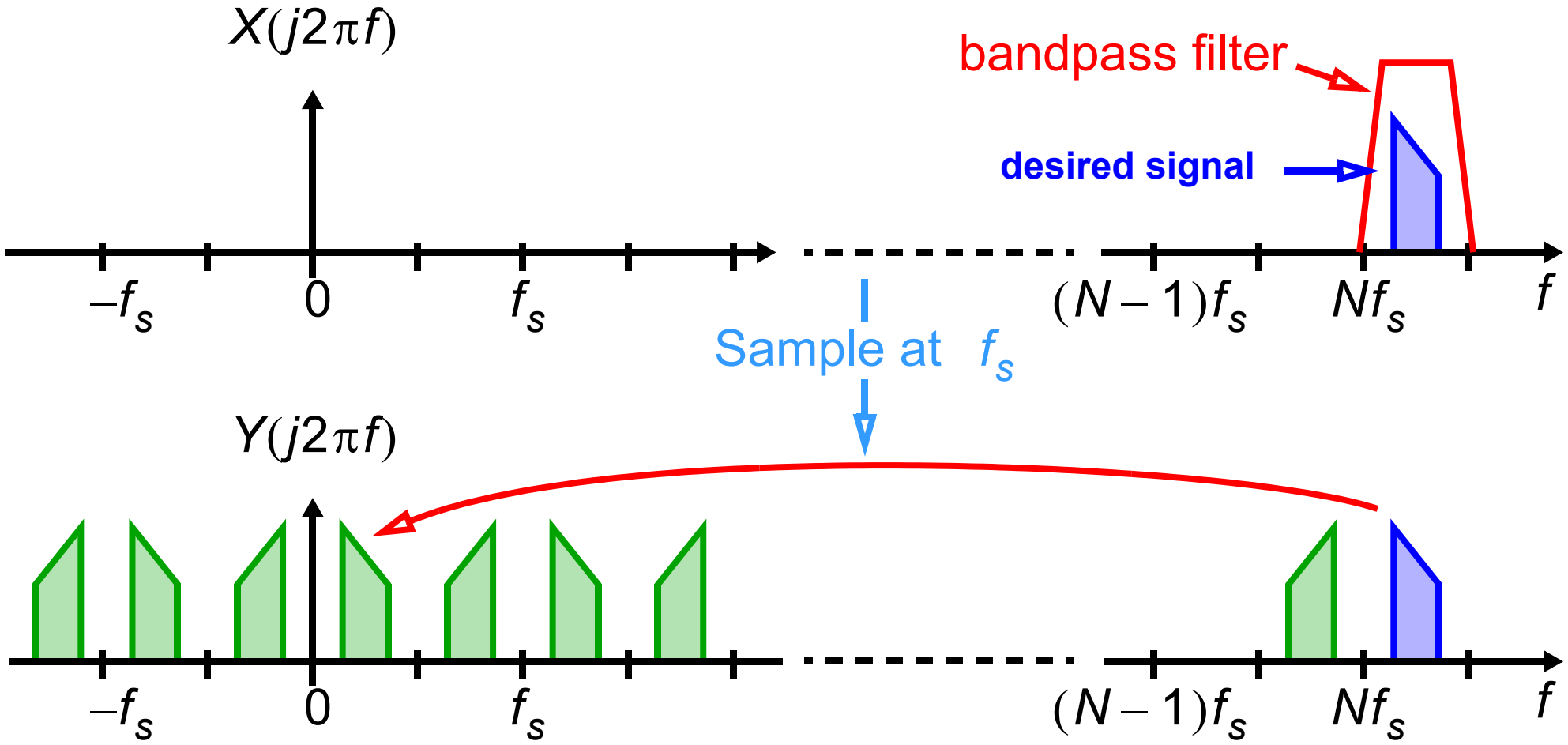
In the direct downconversion architecture a bandpass filter is used to extract the frequency range of interest. This resulting signal can then be sampled at a rate which is related to its bandwidth. Other constraints must also be placed on the signal however it is generally possible to sample the signal at a rate greater twice its bandwidth with no loss of information. This actually breaks the Nyquist sampling criteria which states that the sampling rate must be greater than twice the highest frequency in the signal. The signal is intentionally aliased however this is alright here because we have removed other frequency components which could potentially overlap as a result of the aliasing.

Another possible downconversion architecture is to demodulate the signal to an intermediate frequency. This can be achieved by mixing the signal with a cosine waveform and filtering unwanted components. The quadrature component is not required at this stage since we are not yet converting to baseband.

A major advantage of the direct downconversion and intermediate frequency techniques is that the final downconversion stages can be performed digitally. This means that, in a communications receiver for example, there is no need for feedback from the digital signal processing stages to the analogue circuitry. Everything can be done digitally!

Direct Downconversion

- Direct downconversion makes use of bandpass sampling theory:



- Desired signal is aliased down to a suitable frequency for processing.

Notes:

BPSK Receiver (I)

- Consider how a BPSK receiver might be implemented in an FPGA;
- First a few equations to clarify the problem!
- The transmitted signal can be expressed as:

$$s(t) = A(t)\cos(2\pi f_c t)$$

- $A(t)$ is the pulse shaped symbol sequence is:

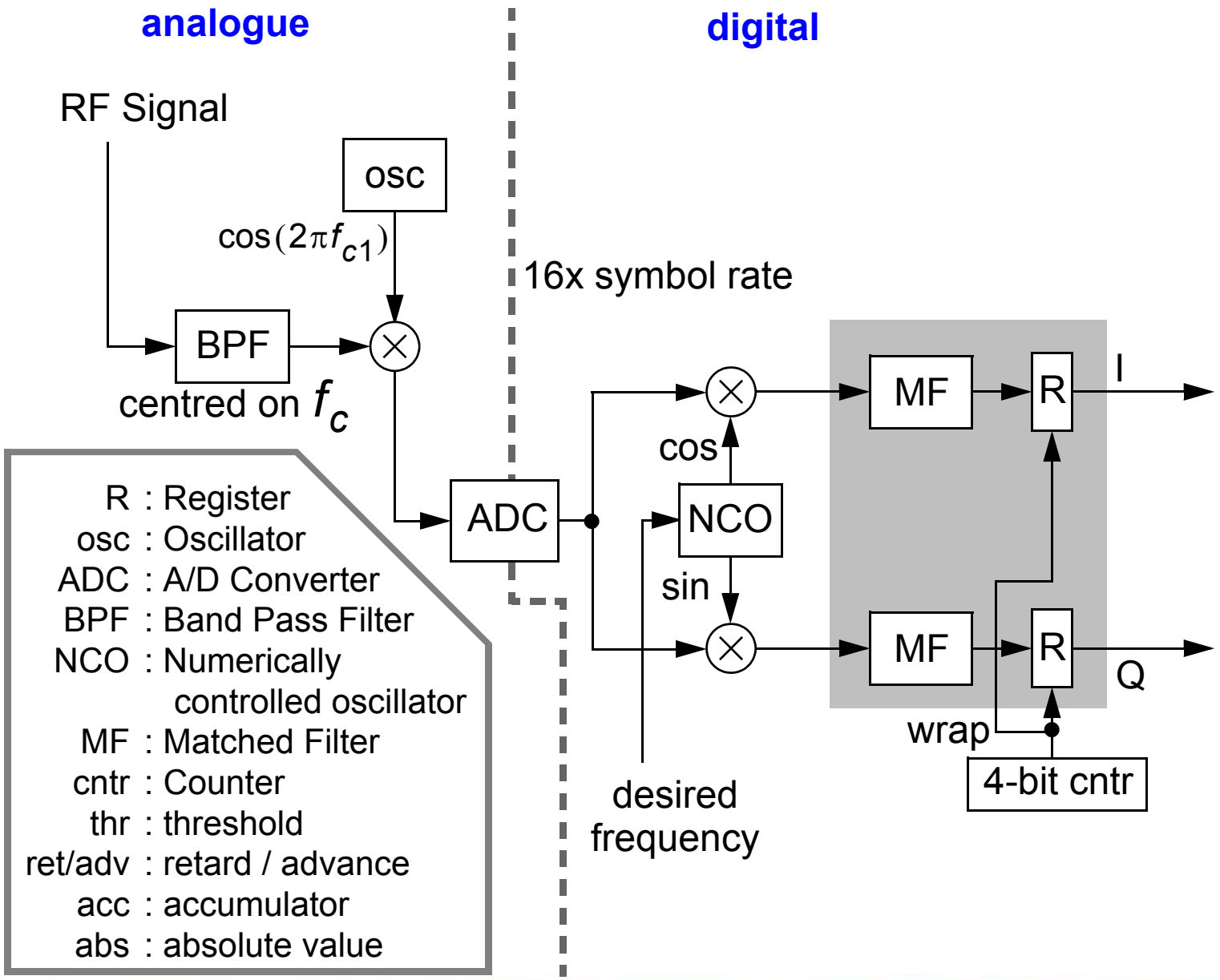
$$A(t) = \sum_k I_k p(t - kT_c)$$

- I_k is the k^{th} symbol and $p(t)$ is the pulse shaping function;
- The data symbols I_k are either -1 for a '0' bit and 1 for a '1' bit.
- Assume that root-raised-cosine (RRC) pulse shaping is used.

Notes:

BPSK Receiver Architecture (II)

- Downconvert to intermediate frequency and demodulate digitally:



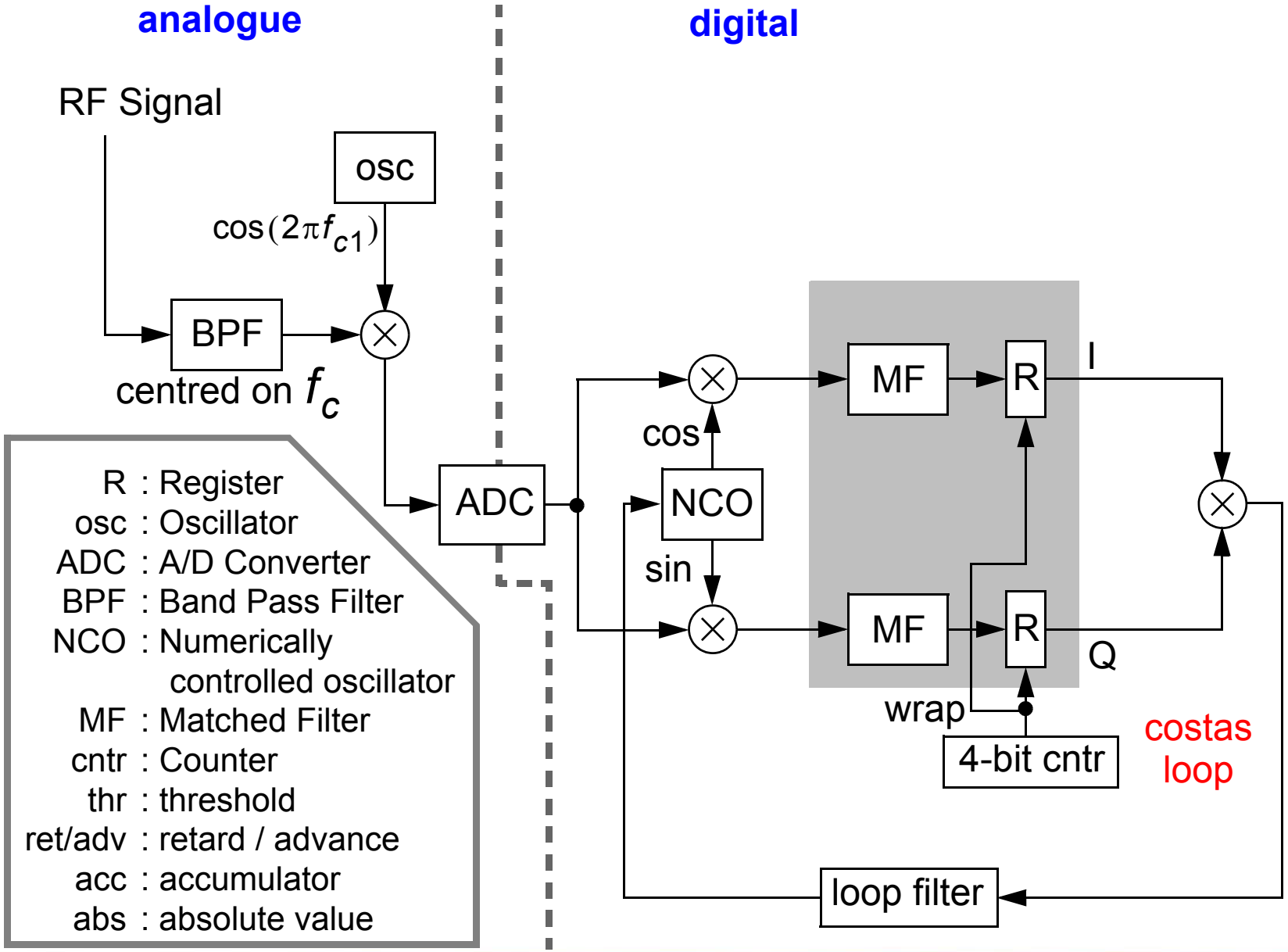
Notes:

This system first downconverts the RF (Radio Frequency) signal to an IF (Intermediate Frequency) signal. It is then sampled at 16x the symbol rate. Demodulation to baseband is performed using a Numerically Controlled Oscillator (NCO) followed by matched filtering operations. The output of these registers is effectively at symbol rate. The output of the each matched filter is then downsampled by latching every 16th output sample in a register. The timing of this is controlled using a 4-bit counter which triggers the latches every time it wraps around to zero.

Note that because the (Root-Raised-Cosine) matched filters are immediately followed by decimation stages they can be efficiently implemented using polyphase techniques. Further gains in efficiency can be realised because the two matched filter / decimator structures are identical. This means that they may be able to share hardware depending on the relative speed of the device compared to the filter input rate.

BPSK Receiver Architecture (III)

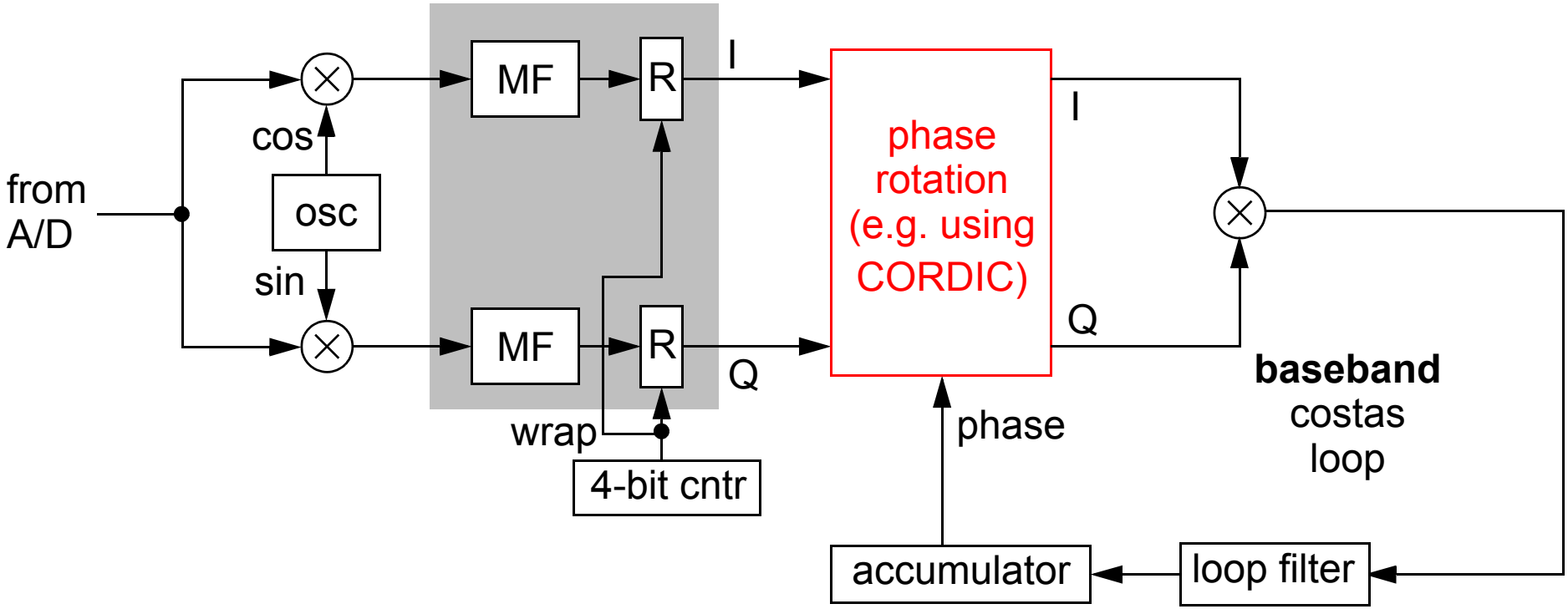
- Use a costas loop to control the Numerically Controlled Oscillator:



Notes:

A costas loop has been introduced to adjust the frequency of the NCO in order to track the carrier phase of the received signal. A multiplier is required to determine the product of the I and Q samples and the result is fed into a loop filter. The output of the loop filter then controls the NCO frequency.

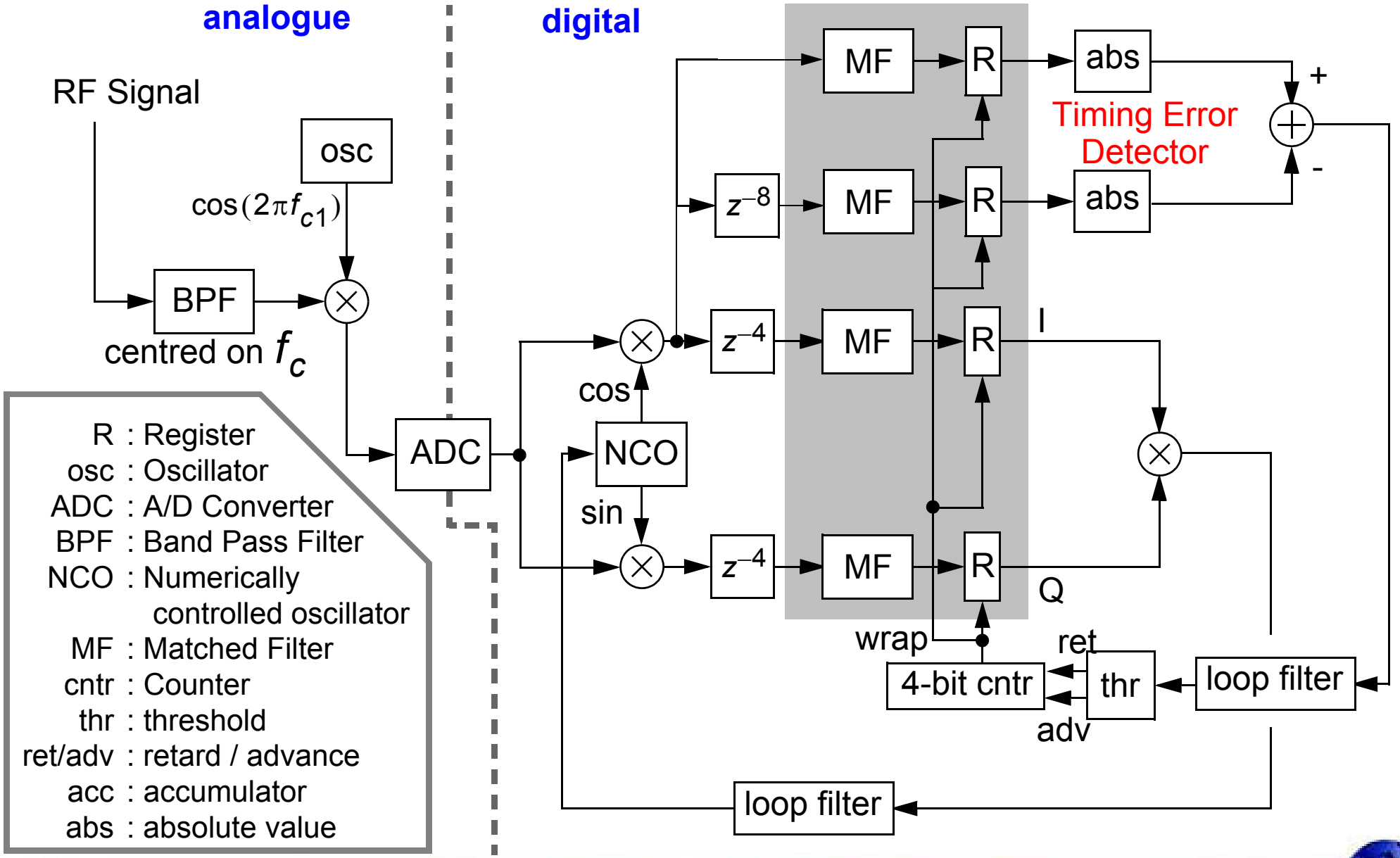
An alternative to this strategy would be to have an fixed frequency digital oscillator and correct the phase at the other side of the matched filter and decimation stages. This strategy is depicted below:



Previously the loop filter output controlled the frequency of the NCO however a phase rotation unit requires a phase input. Hence an accumulator is required to convert the frequency control to a phase control.

BPSK Receiver Architecture (I)

- Use early late timing error detection to control the symbol timing clock:



Notes:

Finally an early late symbol timing error detector has been introduced. The early / late symbol detector intentionally samples a quarter of a symbol earlier and a quarter of a symbol later than the maximum effect points. It then calculates the difference between the early and late samples and filters the result to estimate the timing error. A threshold is then applied to the magnitude of the loop filter output. If the threshold is exceeded then the 4 bit counter is either advanced (incremented by an extra step) or retarded (prevented from counting for a single step) depending on the sign of the filter output. The result is to briefly either speed up or slow down the count in order to track the timing of the received signal.

Conclusion

- The general synchronization problem was described;
- Standard methods for carrier and symbol timing recovery reviewed;
- Sampling rate changes and downconversion methods were introduced;
- A BPSK example was used to demonstrate how some well known techniques can lead to efficient implementation on an FPGA;
- Some of these techniques were:
    - The use of downconversion by bandpass sampling;
    - Lookup tables for numerically controlled oscillators;
    - Polyphase implementation of matched filters;
    - Use of simple but effective filter structures.

Notes: