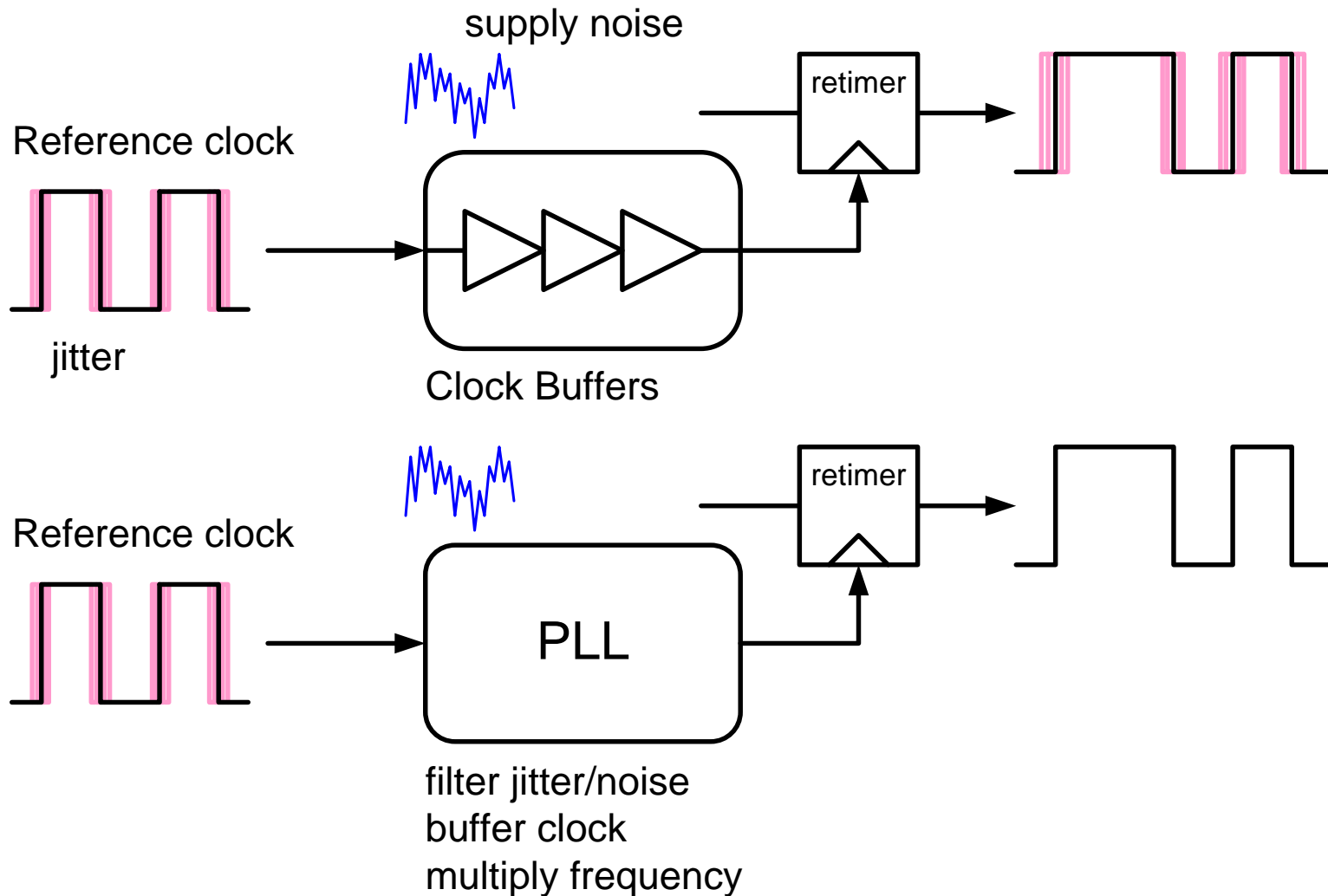


# Digital PLL Basics

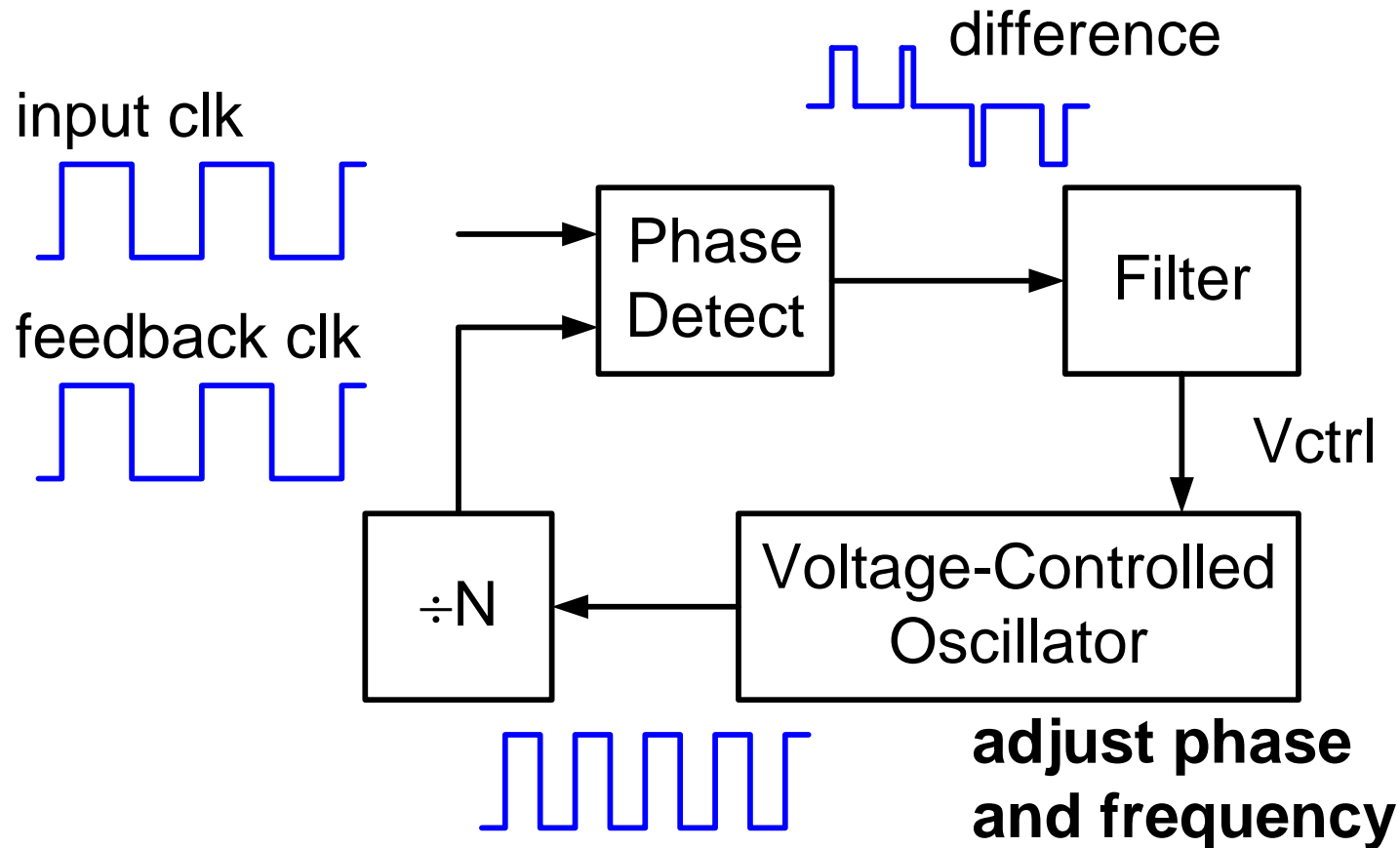
**Jaeha Kim, Sung-Joon Lee, and Eunseo Kim**  
**Mixed-Signal IC and System Group**  
**Seoul National University**  
**May 19. 2016**

# Need for Phase-Locked Loops



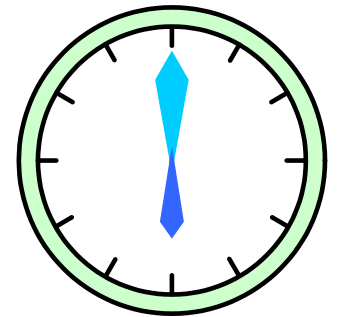
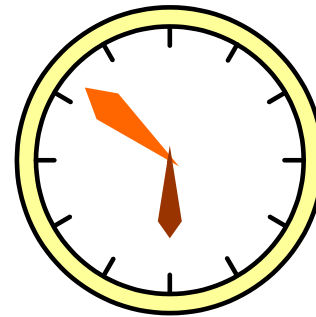
# Phase-Locked Loop (PLL)

- ▶ PLL aligns its output clock to the input clock via feedback



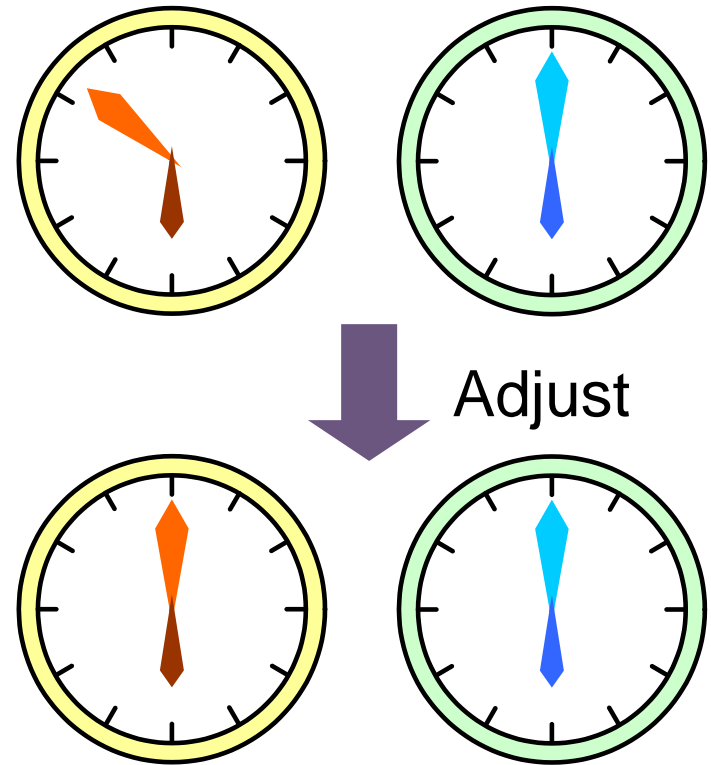
# Synchronize Two Clocks

- ▶ Charlie and Lucy want to meet at 6 o' clock sharp everyday
- ▶ But their watches have
  - ▶ phase offset
  - ▶ frequency offset
- ▶ Only Lucy can adjust Charlie's watch when they meet



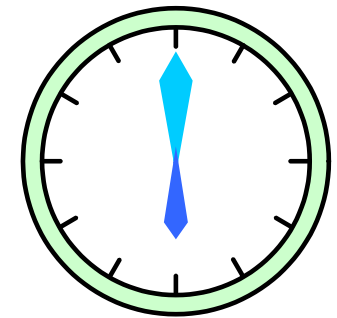
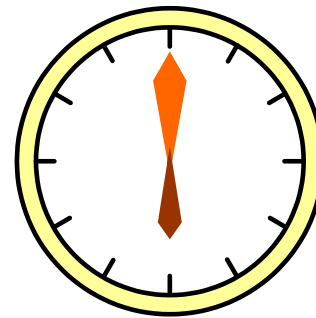
# Lucy's 1st Idea

- ▶ Adjust Charlie's watch so that it points the same time with Lucy's
- ▶ Hoping that they will stay synchronized till tomorrow
- ▶ Will it work?

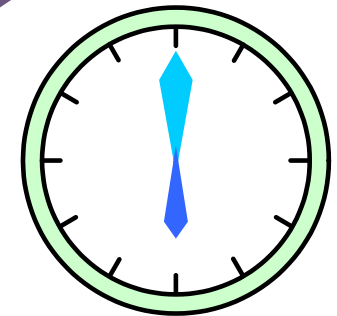
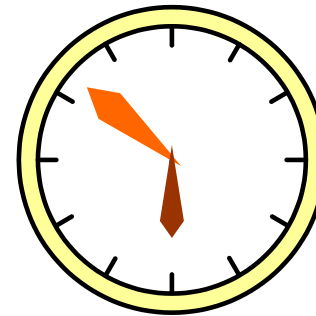


# Frequency Offset Causes Phase Drift

- ▶ If Charlie's clock doesn't tick at the same rate with Lucy's, his clock will always be off after a day
- ▶ Lucy realizes time error has two components
  - ▶ phase error
  - ▶ frequency error

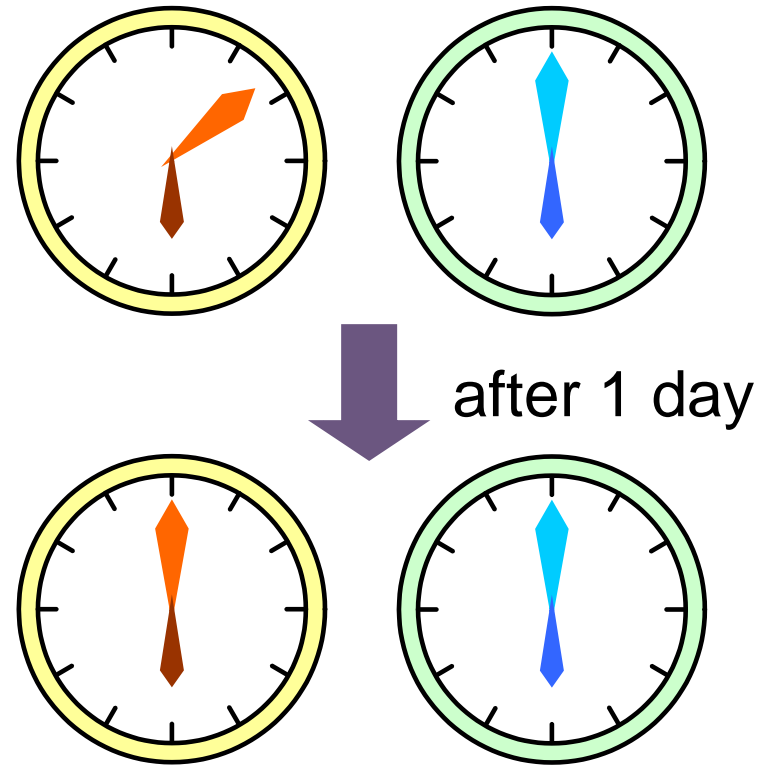


after 1 day

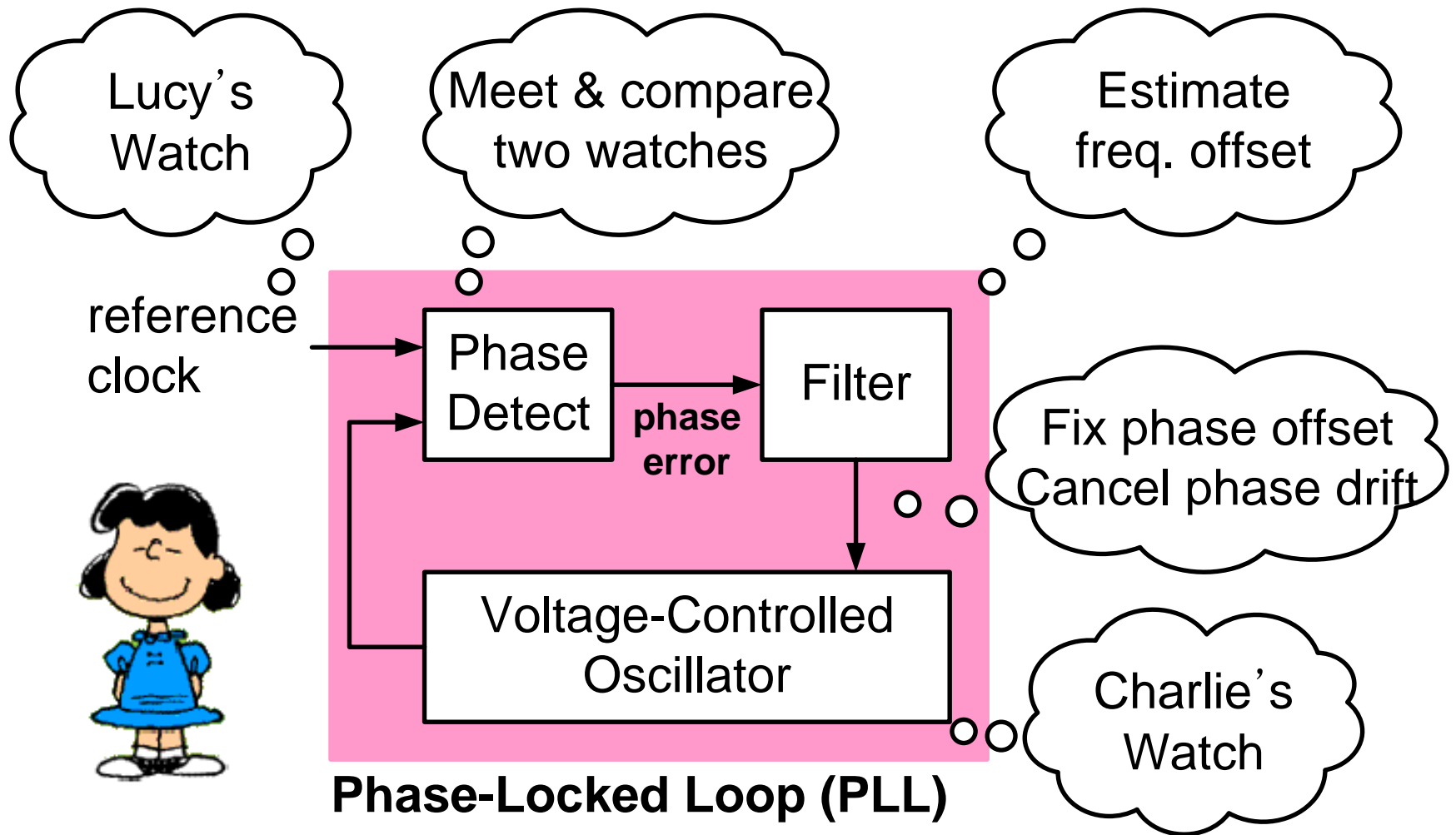


# Lucy's 2nd Idea

- ▶ Cancel the phase drift in advance
  - ▶ so the clocks will be synchronized the next day
- ▶ But Lucy must know the drifting rate (freq. offset)
  - ▶ Estimate it based on the past time differences
  - ▶ while fixing phase offsets



# That's What a PLL Does



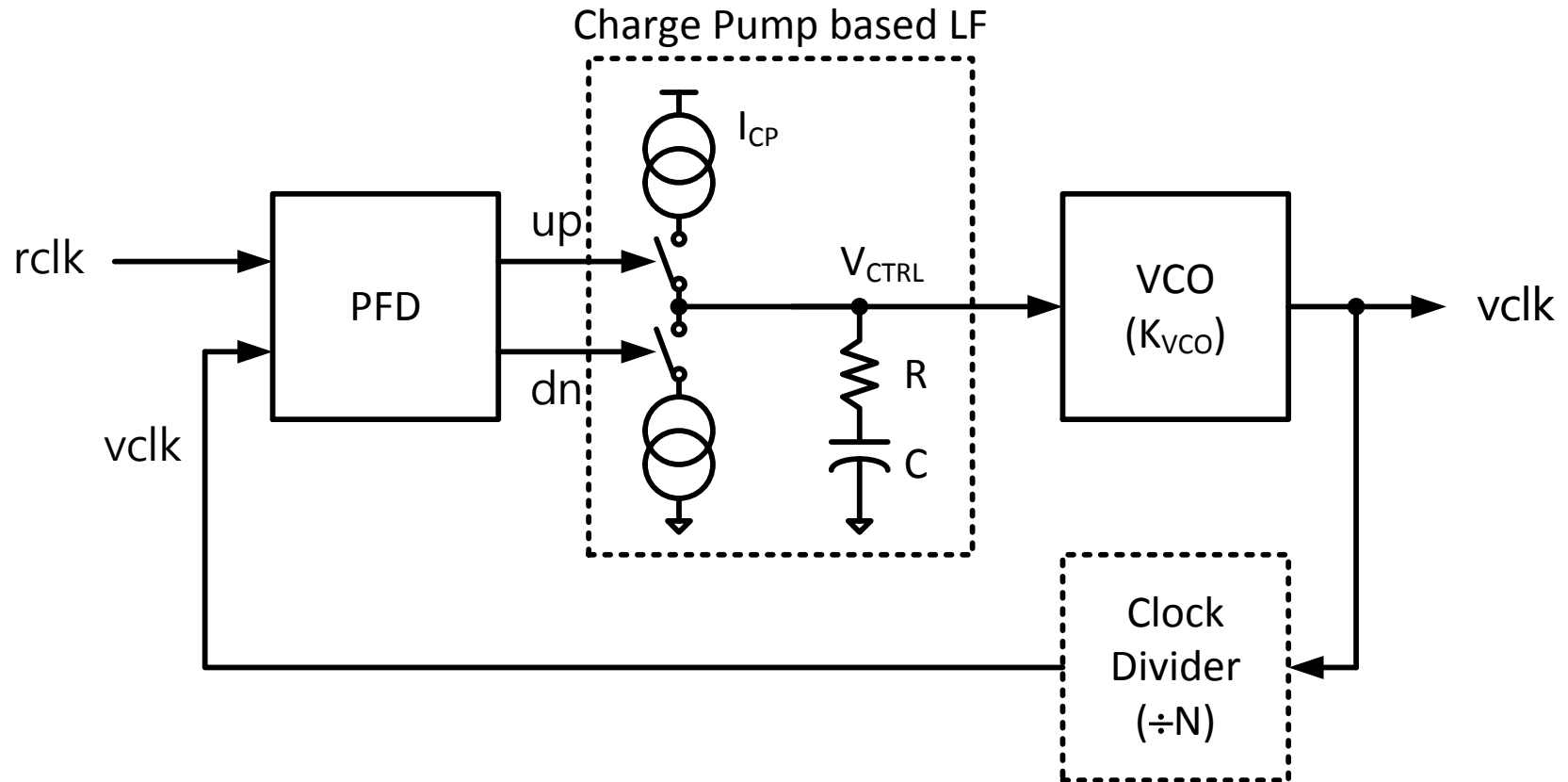


# PLL Control Basics

- ▶ Goal is to make the reference phase ( $\phi_{\text{ref}}$ ) and feedback phase ( $\phi_{\text{fb}}$ ) the same
  - ▶ Make Lucy and Charlie arrive at the same time
- ▶ Upon the detection of the phase error ( $\phi_{\text{err}}$ ), the PLL adjusts two things from the output clock:
  - ▶ The phase:  $\phi_{\text{out}} \leftarrow \phi_{\text{out}} + K_1 \cdot \phi_{\text{err}}$   
(The current time of Charlie's watch)
  - ▶ The frequency:  $\omega_{\text{out}} \leftarrow \omega_{\text{out}} + K_2 \cdot \phi_{\text{err}}$   
(The current estimate of Charlie's watch tick rate)
- ▶ There are two variables that Lucy needs to control – a second-order system!



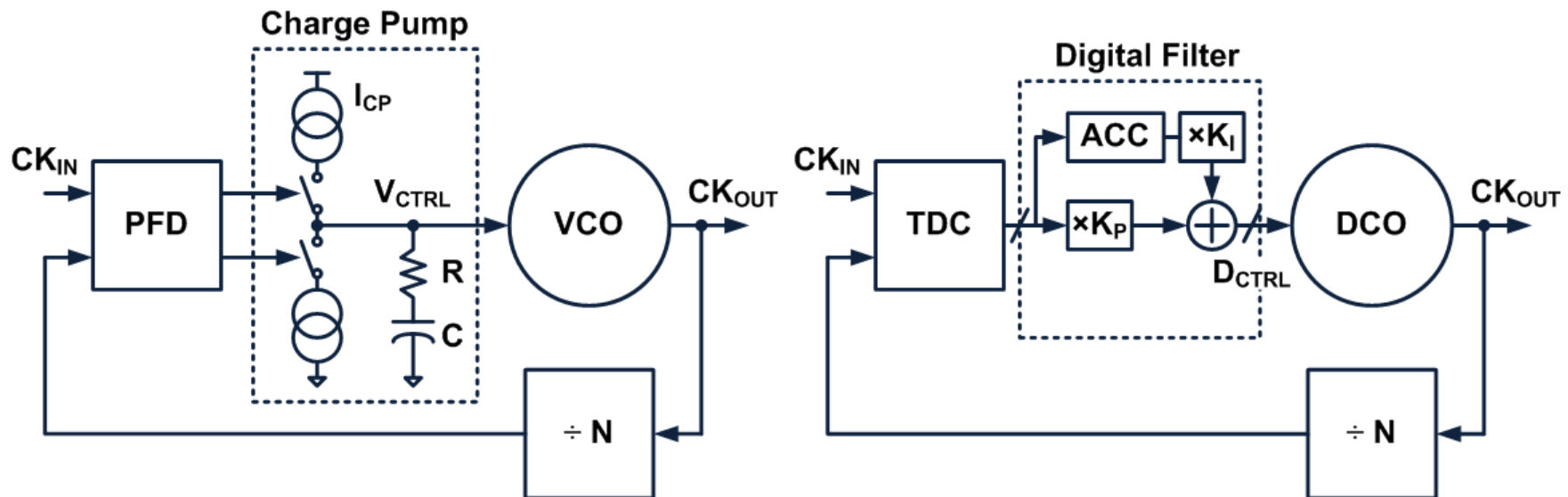
# Charge-Pump PLL



- Uses a charge pump followed by an RC filter as the loop filter (controller)

# Digital PLLs

- ▶ Use accumulators instead of charge pumps to:
  - ▶ Overcome technology limitations such as CP current mismatch, capacitor leakage, design difficulty, etc.
  - ▶ Implement transfer functions, calibrations, fast-settling algorithms that are difficult with analog



# PLL Components

- ▶ Just with any feedback loop, a PLL comprises of:
- ▶ **Producer (DCO):** generates the output clock with desired phase/frequency
- ▶ **Sensor (TDC):** measures the error between the output and the reference input
- ▶ **Loop filter or Controller:** decides how to adjust the control input of the producer based on the measured error

# Performance Metrics of PLL

# PLL Evaluation

**Q. How to evaluate whether it is a good PLL?**

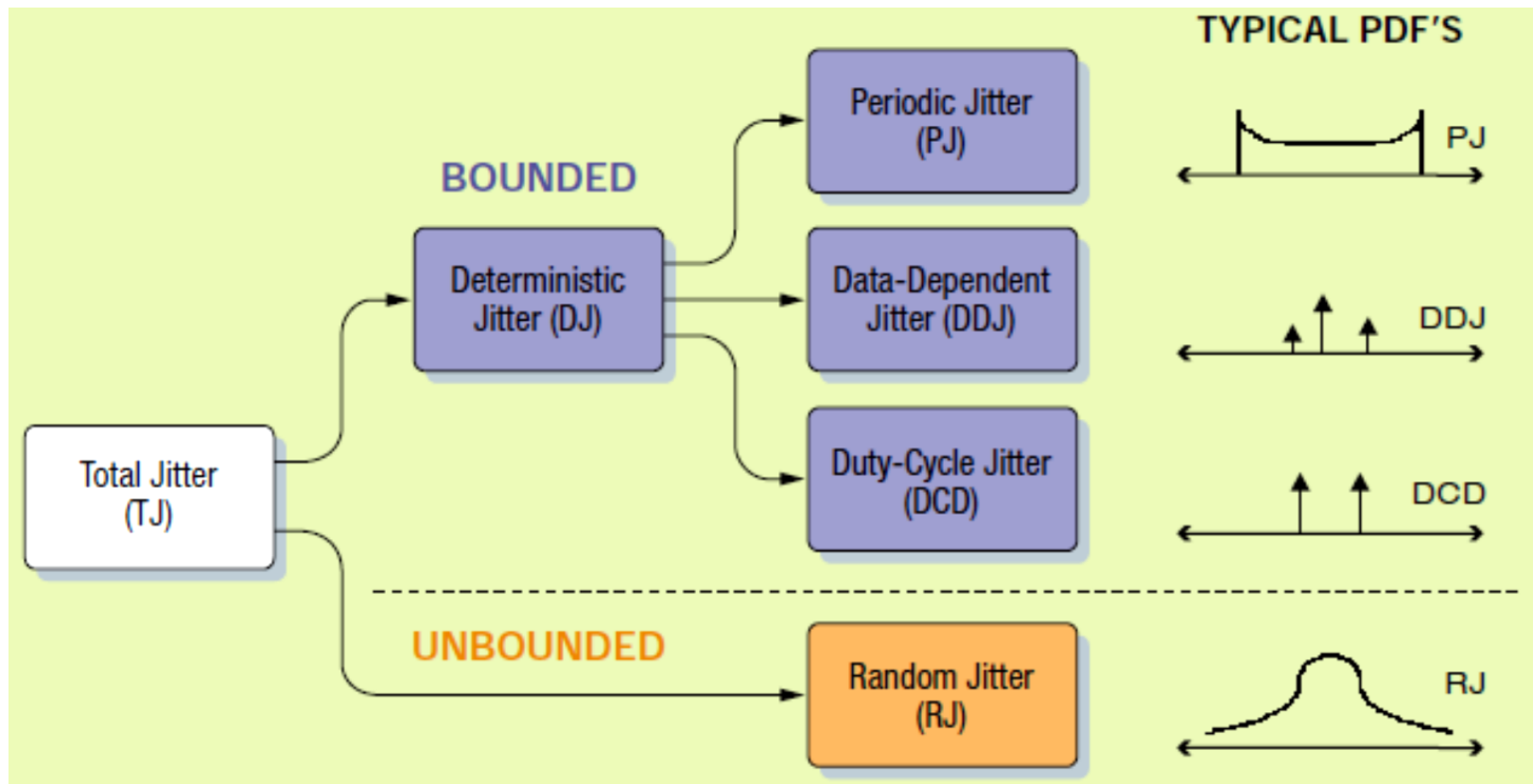
- ▶ **Jitter & phase noise**
- ▶ Lock-in range
- ▶ Locking time
- ▶ (reference) spur, supply noise rejection
- ▶ Power, chip area, ...

# Definition of Jitter

- ▶ The short term variations of digital signal's significant instants from their ideal positions in time
  - ▶ Jitter : timing variations that occurs rapidly (over 10Hz)
  - ▶ Wander : those that occur slowly (under 10Hz)
  - ▶ The exact moments when transitional signal crosses a threshold (zero-crossing time)
- ▶  $v_n(t) = v(t + j(t))$ 
  - ▶ A noise-free signal  $v(t)$
  - ▶ Stochastic process  $j(t)$

# Jitter Decomposition

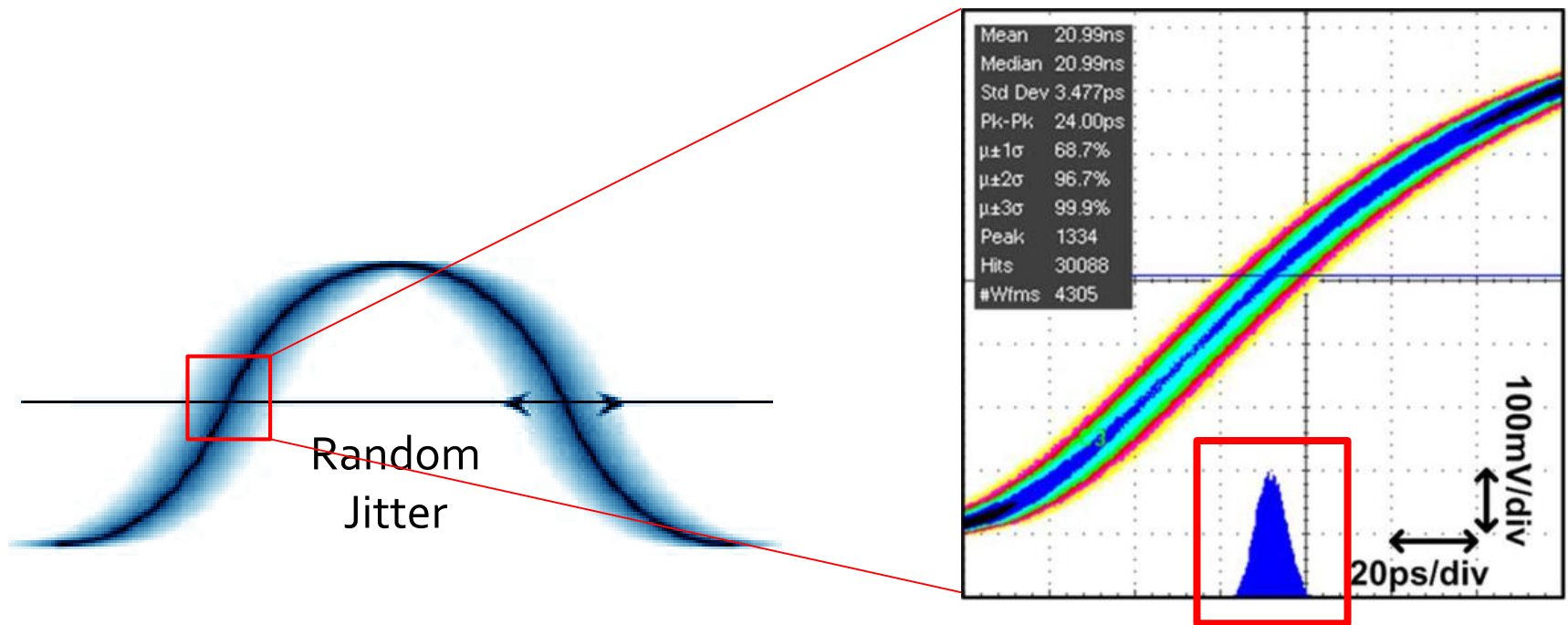
- ▶ Random jitter: unpredictable electronic timing noise
- ▶ Deterministic jitter: predictable and reproducible





# Random Jitter

- ▶ Gaussian distribution (central limit theorem)
- ▶ Unbounded peak-to-peak value



# RMS Jitter and Peak-to-Peak Jitter

- ▶ RMS jitter : root-mean-square value of Gaussian noise distribution
- ▶ Peak-to-peak jitter : the magnitude that the jitter exceeds only for a specified “error rate”

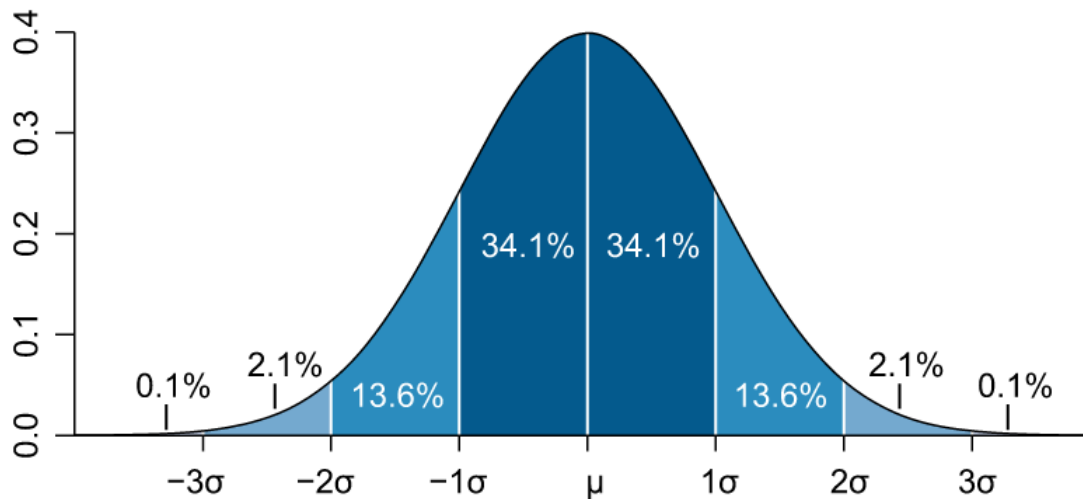
$$\underline{J_{PP}} = \alpha \underline{J_{RMS}}$$

Peak-to-peak jitter

RMS jitter

# Error Rate vs. $\alpha$

$$J_{PP} = \alpha J_{RMS}$$



- ▶ With the given timing constraint and desired BER (bit-error-rate), a designer can determine the RMS jitter spec of clock source

Error Rate	$\alpha$
$10^{-3}$	6.180
$10^{-4}$	7.438
$10^{-5}$	8.530
$10^{-6}$	9.507
$10^{-7}$	10.399
$10^{-8}$	11.224
$10^{-9}$	11.996
$10^{-10}$	12.723
$10^{-11}$	13.412
$10^{-12}$	14.069
$10^{-13}$	14.698
$10^{-14}$	15.301
$10^{-15}$	15.883
$10^{-16}$	16.444

# Definition of Phase Noise

- ▶ Frequency domain representation of rapid, short-term ( $> 10\text{Hz}$ ), random fluctuations in the phase of a periodic wave
- ▶ 
$$v_n(t) = v(t + \frac{\varphi(t)}{2\pi f_0})$$
  - ▶ A noise-free signal  $v(t)$
  - ▶ Stochastic process  $\varphi(t)$
- ▶ Cannot be directly measured
- ▶ **Instead, use power spectral density (PSD)**

# Power Spectral Density (PSD)

- ▶ How much the power of a signal or time series is distributed with frequency
- ▶ For convenience with abstract signals, power is the squared value of the signal
- ▶ **The PSD of a wide-sense stationary random process is the Fourier transform of the corresponding autocorrelation function**

# Single Sideband (SSB) Noise Spectral Density

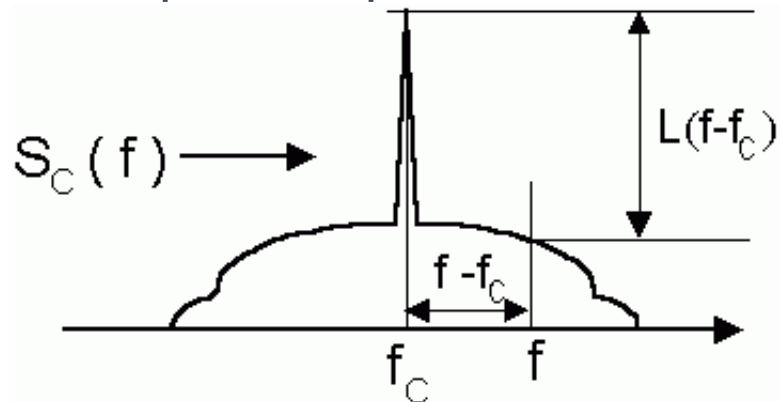
- ▶ Characterizes an oscillator's short term instabilities in the frequency domain

- ▶ 
$$L(\Delta f) = 10 \log \left[ \frac{P_{sideband}(f_0 + \Delta f, 1 \text{ Hz})}{P_{carrier}} \right] \quad (\text{dBc/Hz})$$

- ▶  $P_{sideband}(f_0 + \Delta f, 1 \text{ Hz})$ , the single sideband power at a frequency offset of  $\Delta f$  from the carrier in a measurement bandwidth of 1Hz (dB/Hz)

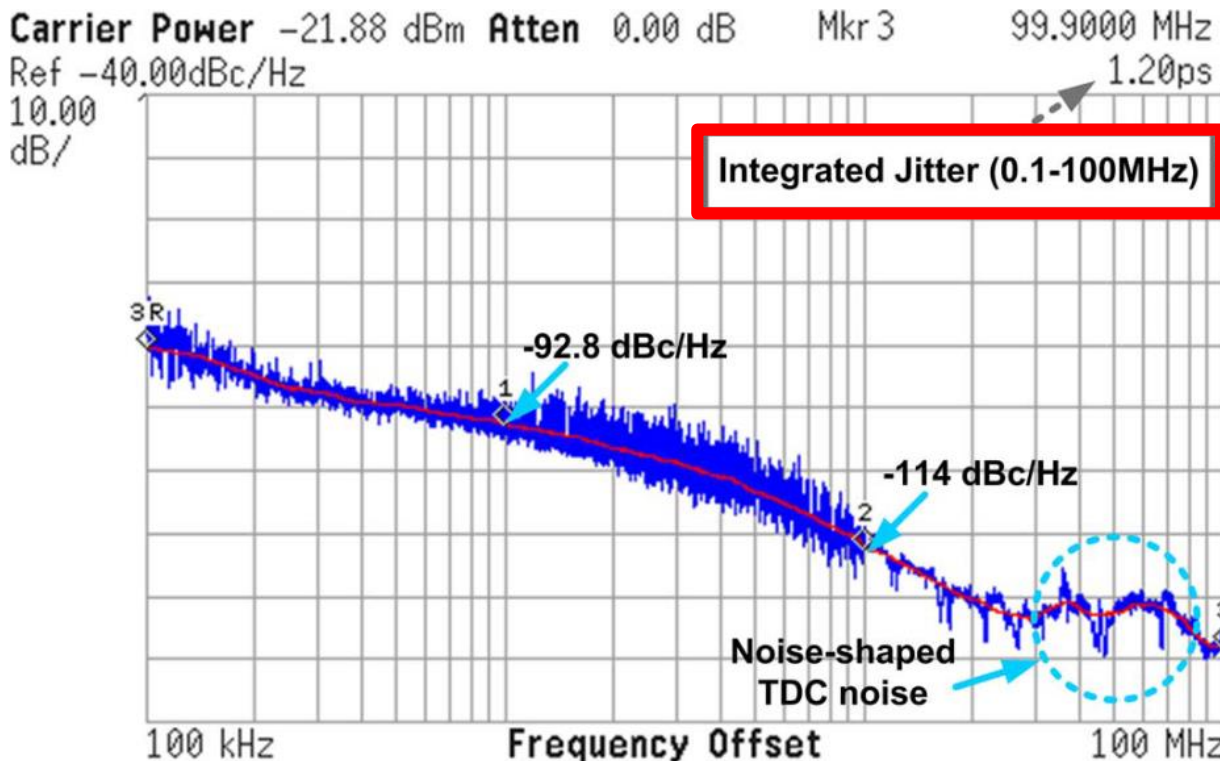
- ▶  $P_{carrier}$  = total power under the power spectrum

- ▶  $S_{\phi}(\Delta f) = 2 * L(\Delta f)$



# Jitter vs. Phase Noise

- ▶ Jitter and phase noise are equivalent
  - ▶ Jitter : time-domain representation
  - ▶ Phase noise : frequency-domain representation



[S. Ryu, 14]

Now, you are ready to design a PLL

*You have never seen  
any TDC or DCO circuits?*

*Don't worry  
Cell-Based Design Flow will help you*