MITOPENCOURSEWARE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.976
High Speed Communication Circuits and Systems
Lecture 18
Design and Simulation of Frequency Synthesizers

Michael Perrott

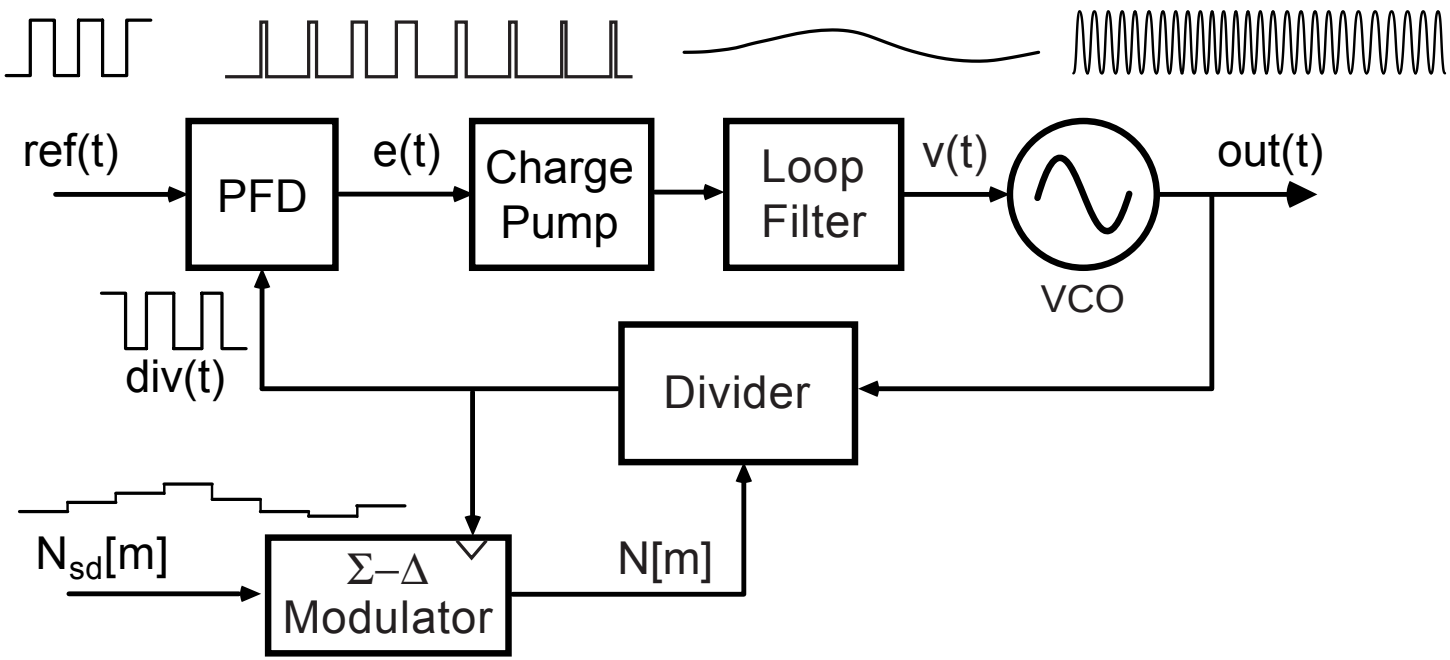
Massachusetts Institute of Technology

Outline

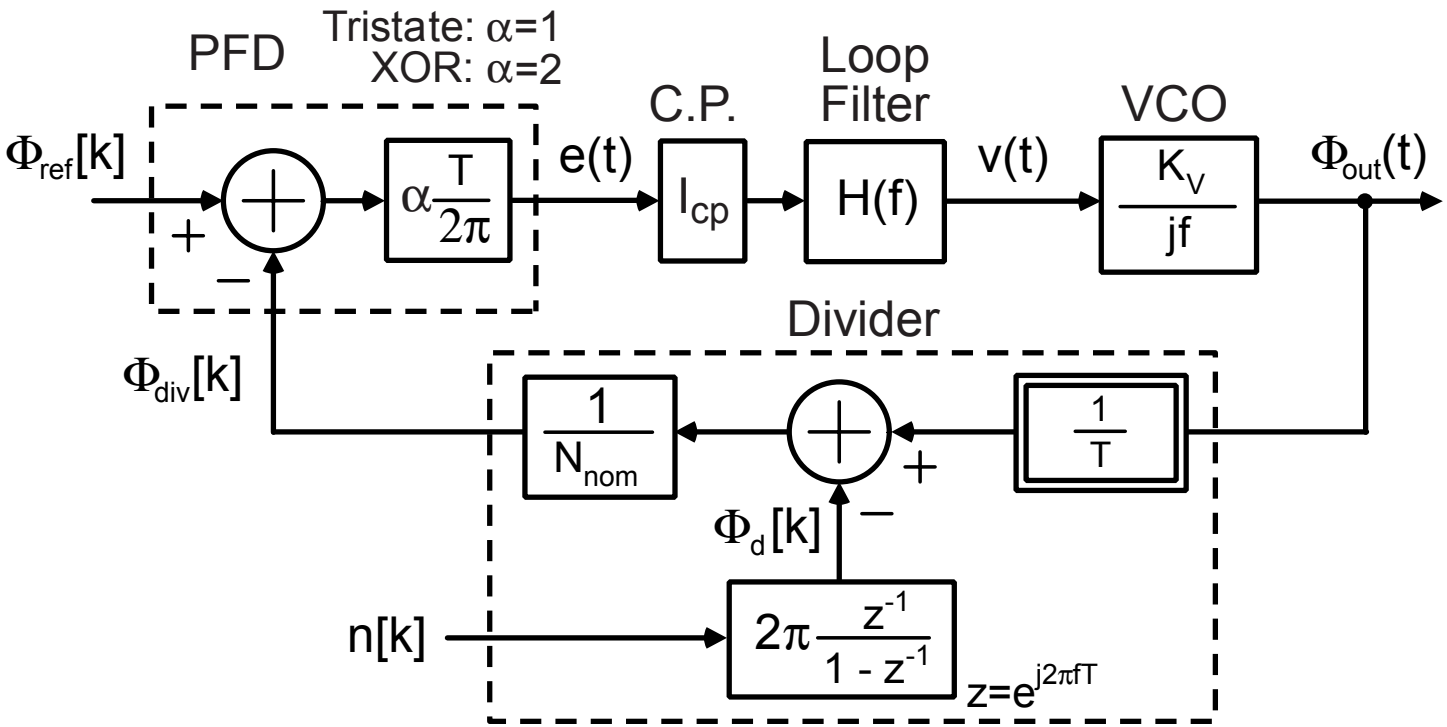
- **Closed-Loop Design of Frequency Synthesizers**
  - **Introduction**
  - **Background on Classical Open Loop Design Approach**
  - **Closed Loop Design Approach**
  - **Example and Verification**
  - **Conclusion**
- **Fast and Accurate Simulation of Frequency Synthesizers**
  - **Introduction**
  - **Difficulties of Traditional Approaches**
  - **Proposed Method**
  - **Example and Verification**
  - **Conclusion**

Σ–Δ *Fractional-N Frequency Synthesizer*

- **Focus on this architecture since it is essentially a "super set" of other synthesizers, including integer-N and fractional-N**
  - **If we can design and simulate this structure, we can also do so for classical integer-N designs**



Frequency-domain Model



See Perrott et. al. *JSSC*, Aug. 2002 for details

- **Closed loop dynamics parameterized by**

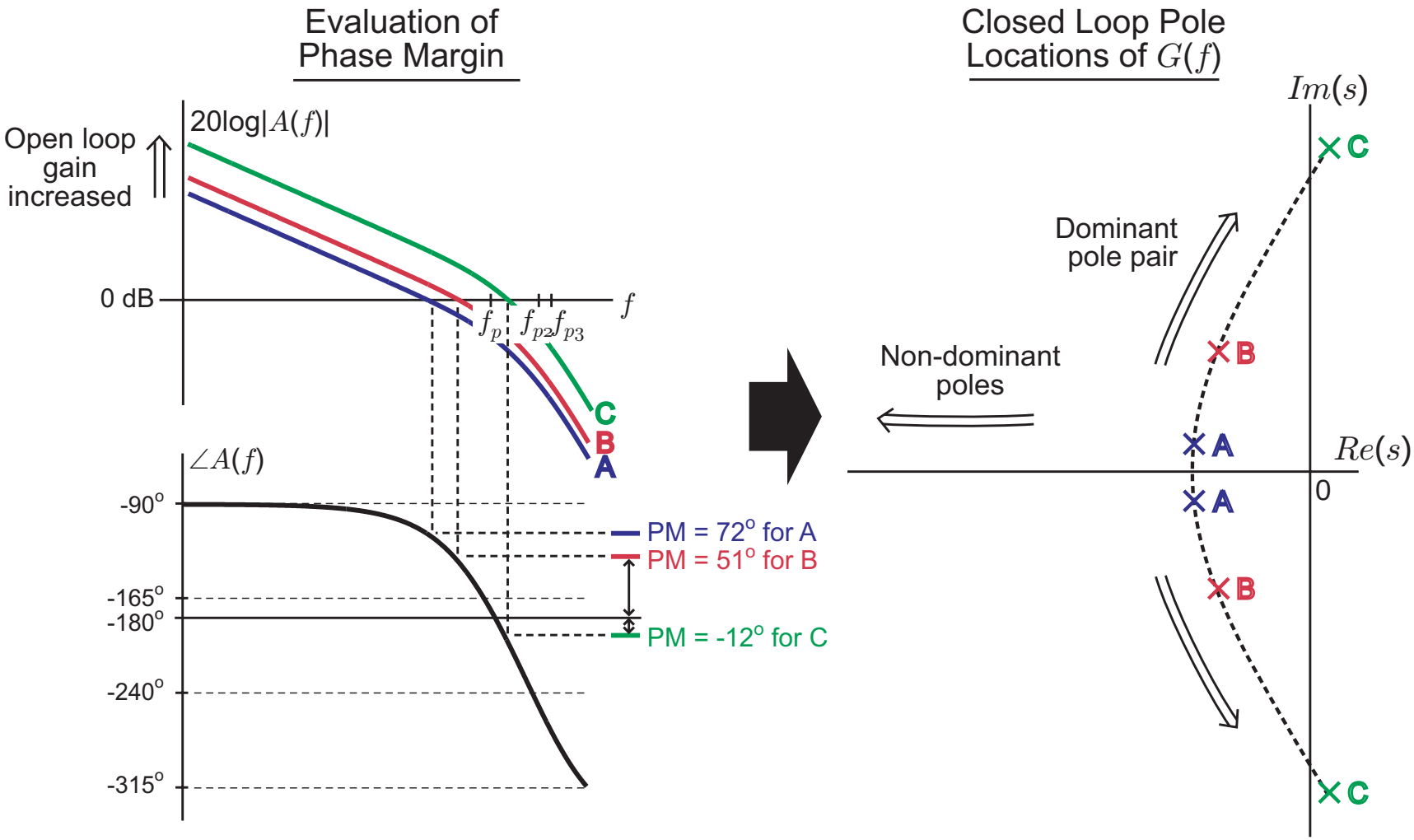
$$G(f) = \frac{A(f)}{1 + A(f)} \quad \text{where} \quad A(f) = \frac{\alpha I_{cp} H(f) K_V}{N_{nom} 2\pi j f}$$

Review of Classical Design Approach

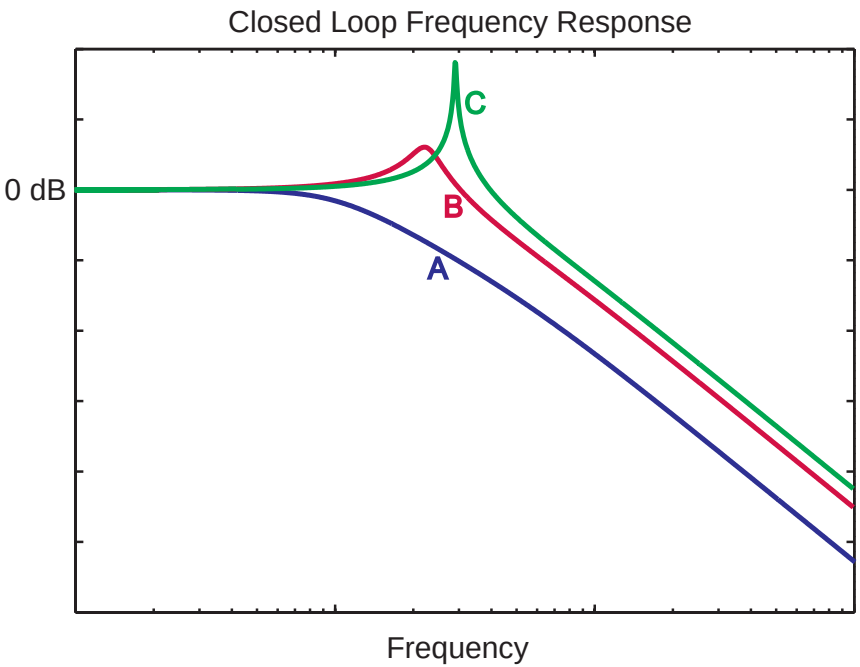
Given the desired closed-loop bandwidth, order, and system type:

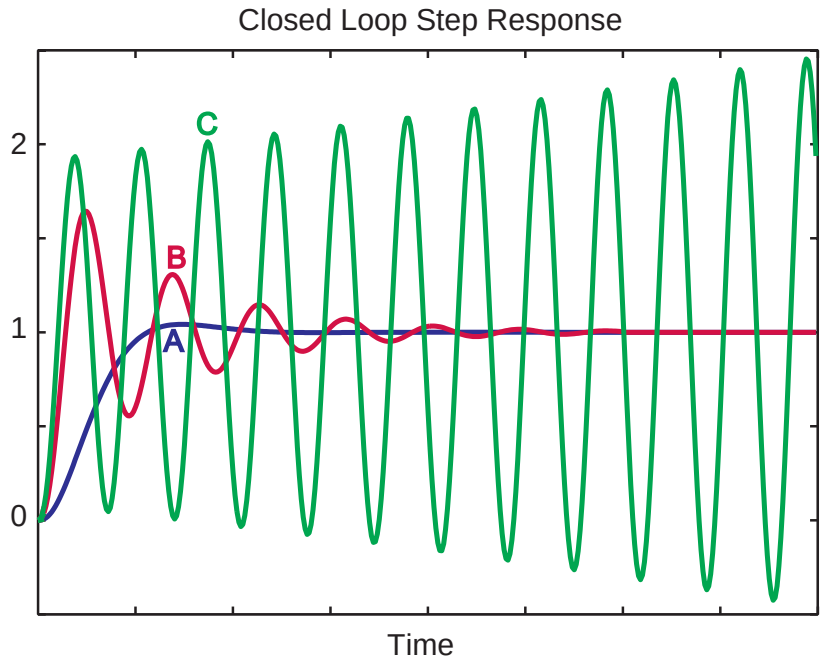
1. **Choose an appropriate topology for $H(f)$**
   - **Depends on order, type**
2. **Choose pole/zero values for $H(f)$ as appropriate for the required bandwidth**
3. **Adjust the open-loop gain to achieve the required bandwidth while maintaining stability**
   - **Plot gain and phase bode plots of $A(f)$**
   - **Use phase (or gain) margin criterion to infer stability**

Example: First Order, Type I with Parasitic Poles



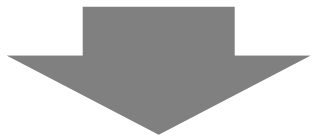
First Order, Type I: Frequency and Step Responses





Limitations of Open Loop Design Approach

- Constrained for applications which require precise filter response
- Complicated once parasitic poles are taken into account
- Poor control over filter shape
- Inadequate for systems with third order rolloff
  - Phase margin criterion based on second order systems

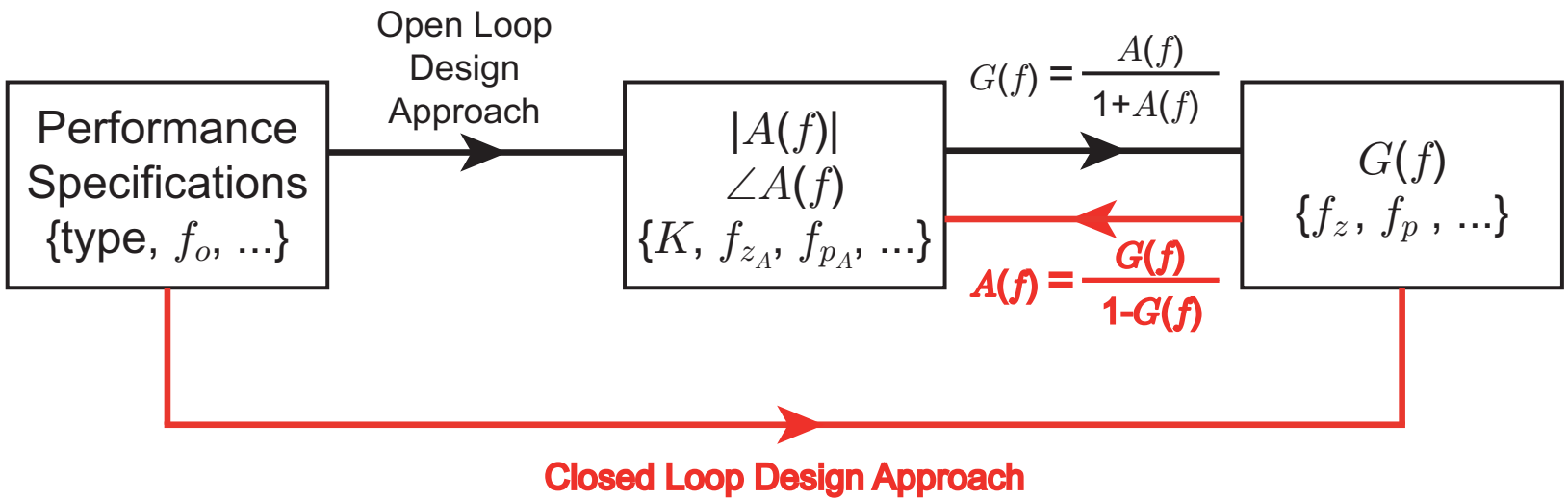


Closed loop design approach:

Directly design G(f) by specifying dominant pole and zero locations on the s-plane (pole-zero diagram)

Closed Loop Design Approach: Overview

- $G(f)$ completely describes the closed loop dynamics
  - Design of this function is the ultimate goal

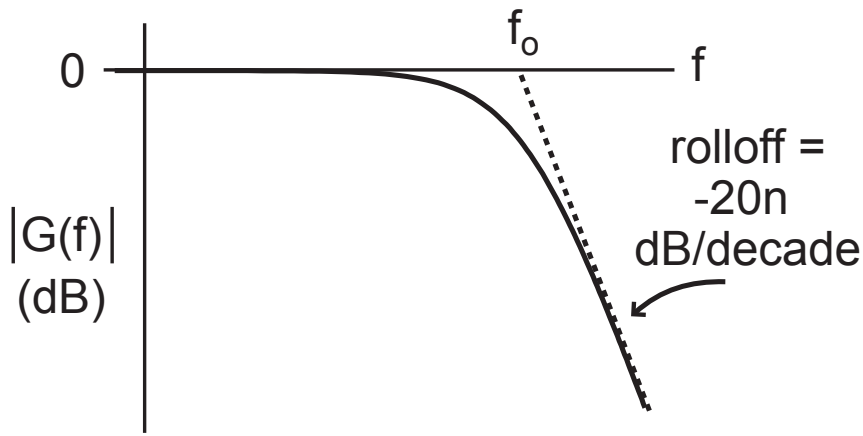


- Instead of indirectly designing $G(f)$ using plots of $A(f)$, solve for $G(f)$ directly as a function of specification parameters
- Solve for $A(f)$ that will achieve desired $G(f)$
- Account for the impact of parasitic poles/zeros

Closed Loop Design Approach: Implementation

- **Download PLL Design Assistant Software at http://www-mtl.mit.edu/research/perrottgroup/tools.html**
- **Read accompanying manual**
- **Algorithm described by C.Y. Lau et. al. in "Fractional-N Frequency Synthesizer Design at the Transfer Function Level Using a Direct Closed Loop Realization Algorithm", Design Automation Conference, 2003**

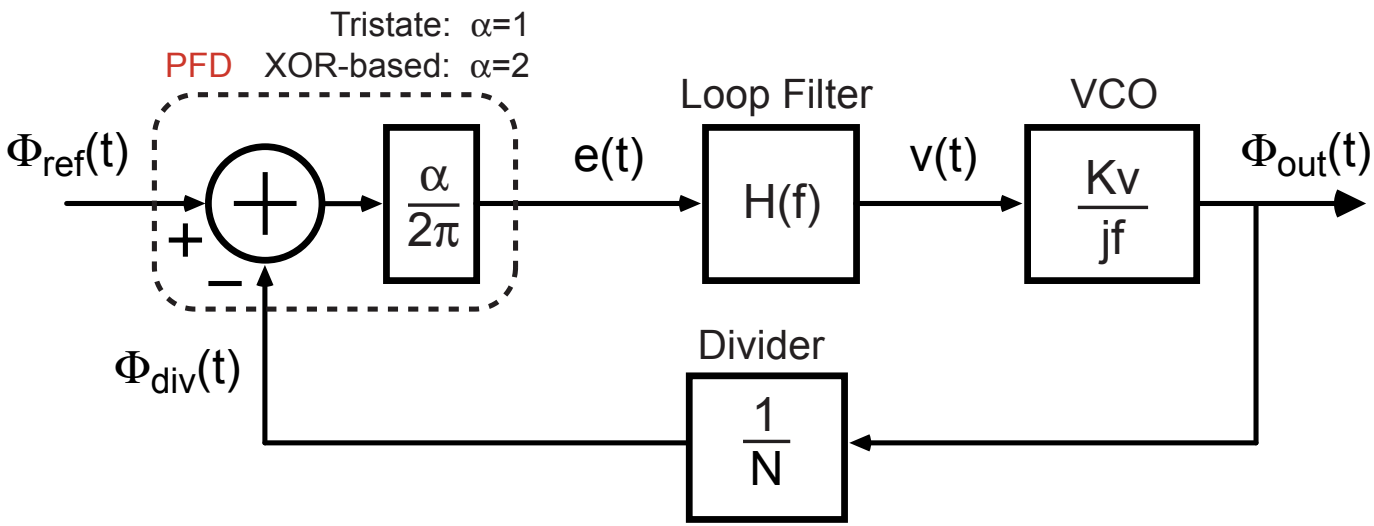
Definition of Bandwidth, Order, and Shape for G(f)



- **Bandwidth – f_o**
  - **Defined in asymptotic manner as shown**
- **Order – n**
  - **Defined according to the rolloff characteristic of G(f)**
- **Shape**
  - **Defined according to standard filter design methodologies**
    - Butterworth, Bessel, Chebyshev, etc.

Definition of Type

- **Type I: one integrator in PLL open loop transfer function**
  - **VCO adds on integrator**
  - **Loop filter, H(f), has no integrators**
- **Type II: two integrators in PLL open loop transfer function**
  - **Loop filter, H(f), has one integrator**



Loop Filter Transfer Function Vs Type and Order of G(f)

H(s) Topology For Different Type and Orders of G(f)

	Type I	Type II
Order 1	K_{LP}	$K_{LP}\frac{1+s/w_z}{s}$
Order 2	$\frac{K_{LP}}{1+s/w_p}$	$K_{LP}\frac{1+s/w_z}{s(1+s/w_p)}$
Order 3	$\frac{K_{LP}}{1+s/(w_pQ_p)+(s/w_p)^2}$	$\frac{K_{LP}(1+s/w_z)}{s(1+s/(w_pQ_p)+(s/w_p)^2)}$

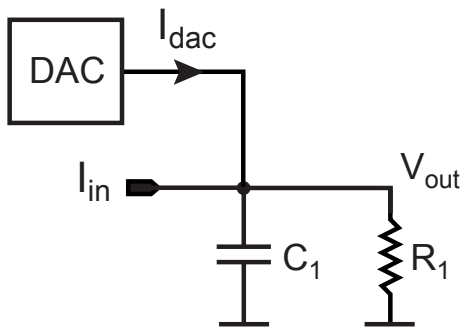
where $K_{LP} = K\frac{N_{nom}}{K_v I_{cp}\alpha}$

Calculated from software

- **Practical PLL implementations limited to above**
  - **Prohibitive analog complexity for higher order, type**
- **Open loop gain, K, will be calculated by algorithm**
  - **Loop filter gain related to open loop gain as shown above**

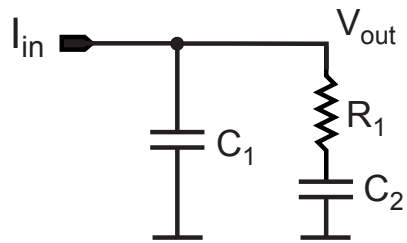
Passive Topologies to Realize a Second Order PLL

Type I, Order 2



$$\frac{V_{out}}{I_{in}} = \frac{R_1}{1+sR_1C_1}$$

Type II, Order 2

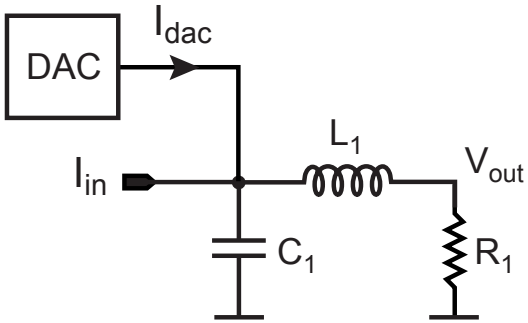


$$\frac{V_{out}}{I_{in}} = \frac{1}{s(C_1+C_2)} \frac{1+sR_1C_2}{1+sR_1C_{||}}$$

- **DAC is used for Type I implementation to coarsely tune VCO**
  - **Allows full range of VCO to be achieved**

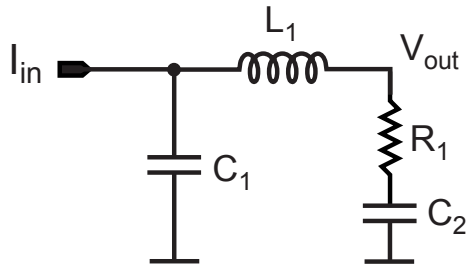
Passive Topologies to Realize a Third Order PLL

Type I, Order 3



$$\frac{V_{out}}{I_{in}} = \frac{R_1}{1+sR_1C_1+s^2L_1C_1}$$

Type II, Order 3



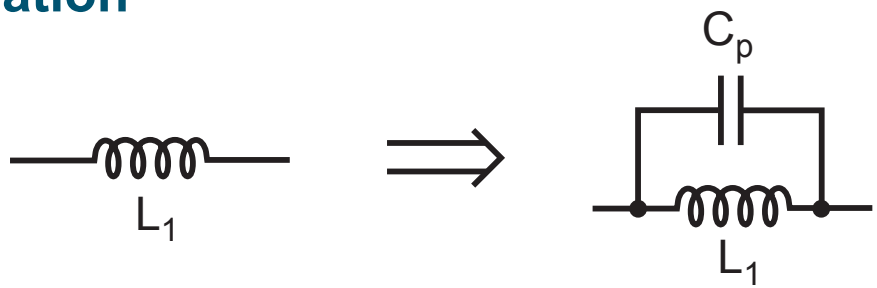
$$\frac{V_{out}}{I_{in}} = \frac{1}{s(C_1+C_2)} \frac{1+sR_1C_2}{1+sR_1C_{||}+s^2L_1C_{||}}$$

where $C_{||} = C_1C_2/(C_1+C_2)$

- **Inductor is necessary to create a complex pole pair**
  - **Must be implemented off-chip due to its large value**

Problem with Passive Loop Filter Implementations

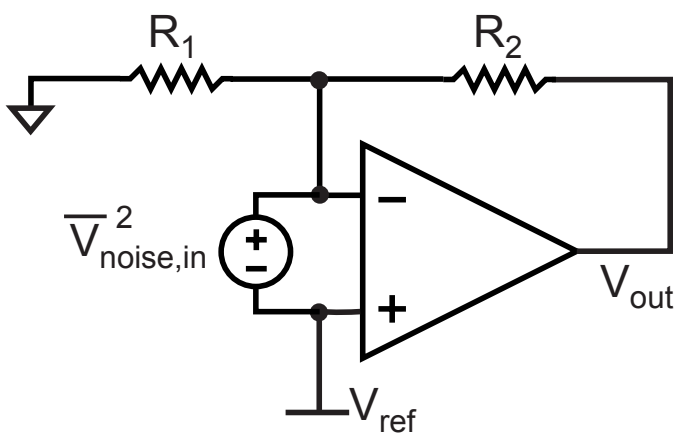
- Large voltage swing required at charge pump output
  - Must support full range of VCO input
- Non-ideal behavior of inductors (for third order G(f) implementations)
  - Hard to realize large inductor values
  - Self resonance of inductor reduces high frequency attenuation



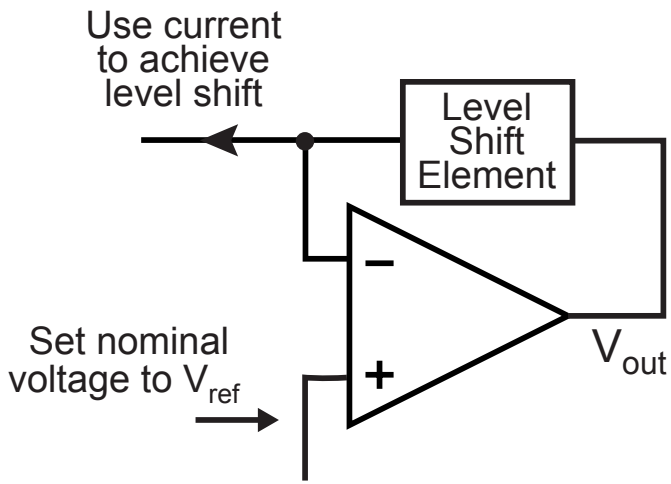
Alternative: active loop filter implementation

Guidelines for Active Loop Filter Design

- Use topologies with unity gain feedback in the opamp
  - Minimizes influence of opamp noise



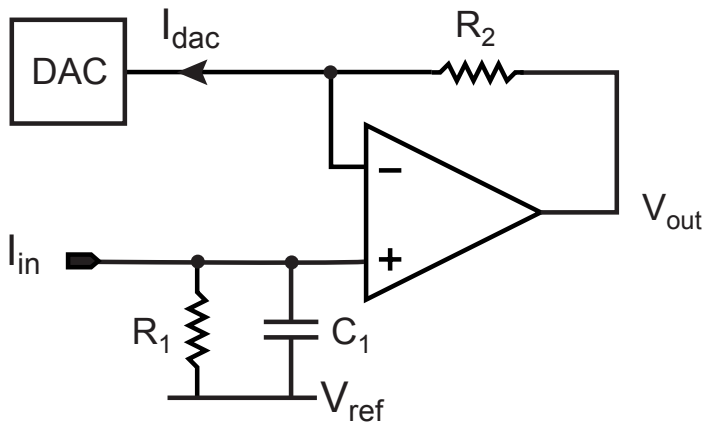
- Perform level shifting in feedback of opamp
  - Fixes voltage at charge pump output



- Prevent fast edges from directly reaching opamp inputs
  - Will otherwise cause opamp to be driven into nonlinear region of operation

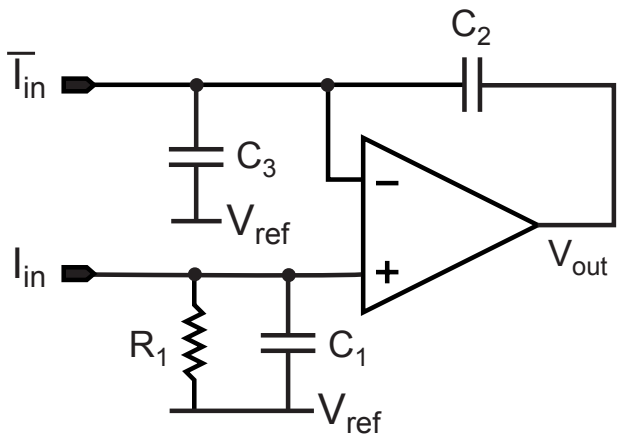
Active Topologies To Realize a Second Order PLL

Type I, Order 2



$$\frac{V_{out}}{I_{in}} = \frac{R_1}{1+sR_1C_1}$$

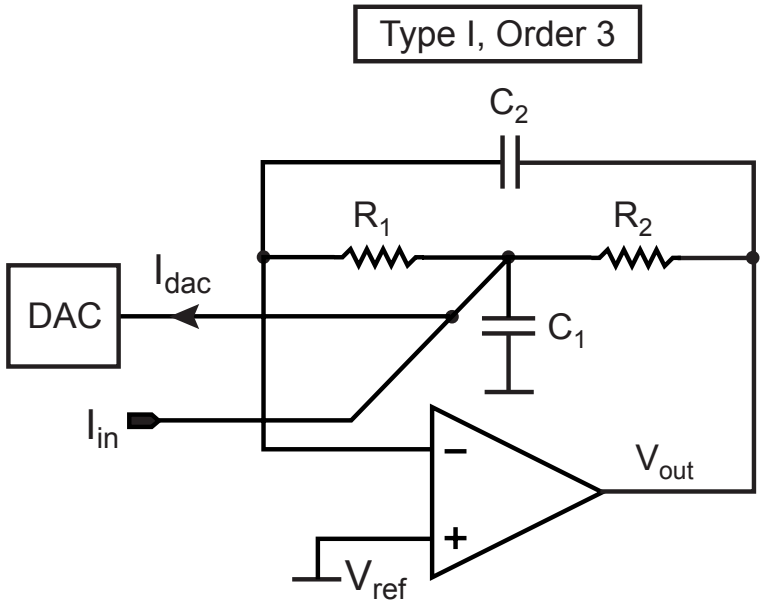
Type II, Order 2

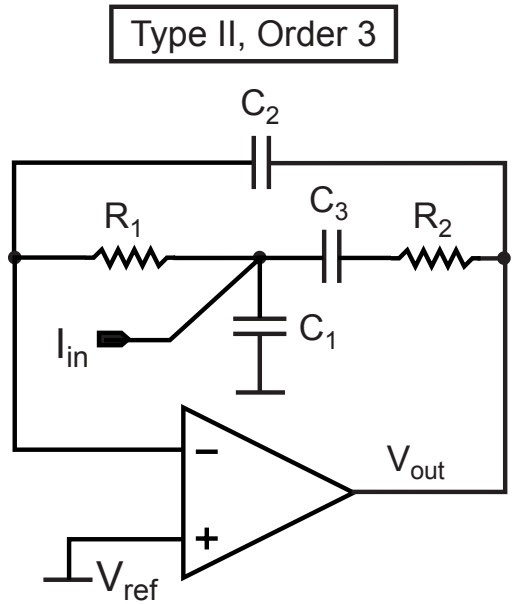


$$\frac{V_{out}}{I_{in}} = \frac{1+sR_1(C_1+C_2+C_3)}{sC_2(1+sR_1C_1)}$$

- Follows guidelines from previous slide
- Charge pump output is terminated directly with a high Q capacitor
  - Smooths fast edges from charge pump before they reach the opamp input(s)

Active Topologies To Realize a Third Order PLL





$$\frac{V_{out}}{I_{in}} = \frac{-R_2}{1+s(R_1+R_2)C_2+s^2R_1R_2C_1C_2}$$

$$\frac{V_{out}}{I_{in}} = \frac{-1}{s(C_1+C_2)} \cdot \frac{1+sR_2C_3}{1+sC_{||}(R_1(1+C_1/C_3)+R_2)+s^2R_1R_2C_1C_{||}}$$

where $C_{||} = C_2C_3/(C_2+C_3)$

- **Follows active implementation guidelines from a few slides ago**

Example Design

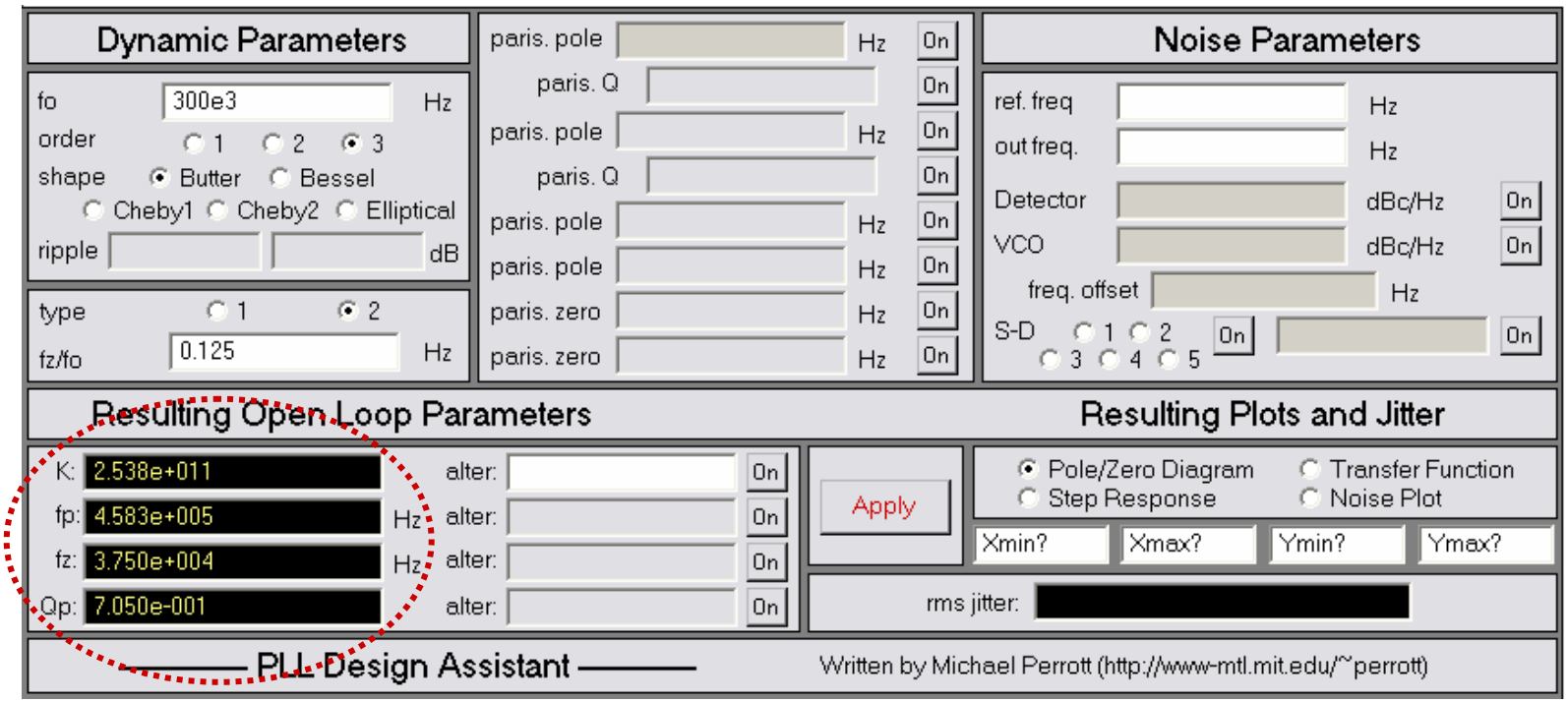
- Type II, 3rd order, Butterworth, f_o = 300kHz, f_z/f_o = 0.125
  - No parasitic poles
- Required loop filter transfer function can be found from table:

$$\Rightarrow H(s) = \frac{K_{LF}\left(1+\frac{s}{w_z}\right)}{s\left(1+\frac{s}{w_p Q_p}+\left(\frac{s}{w_p}\right)^2\right)} \quad \text{where}$$

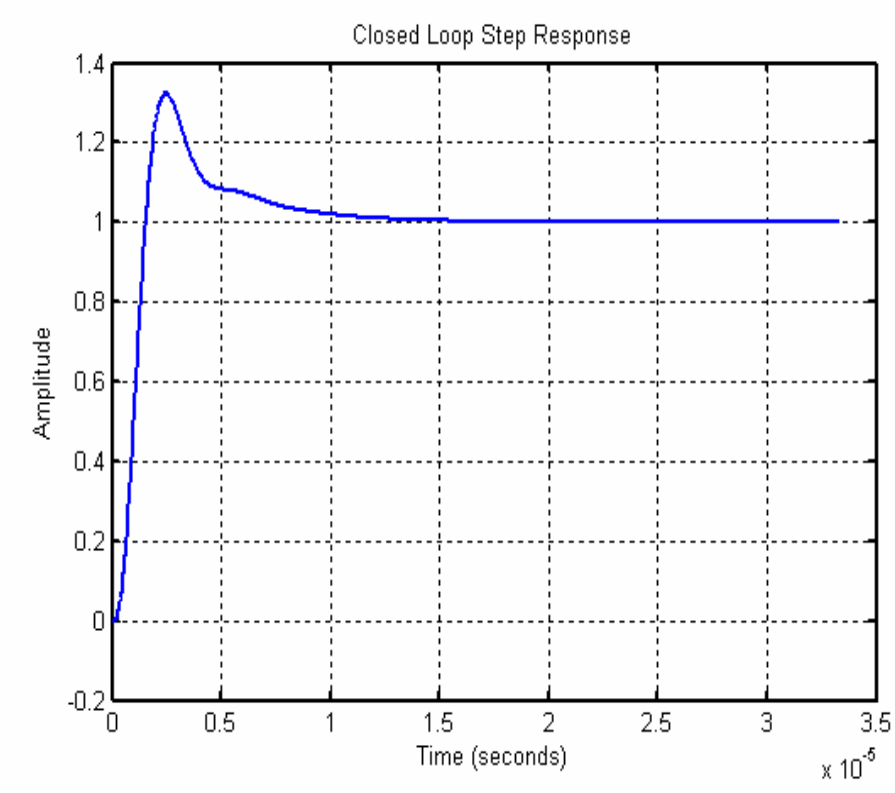
$$K_{LF} = \frac{N_{nom} K}{\alpha I_{cp} K_v}$$

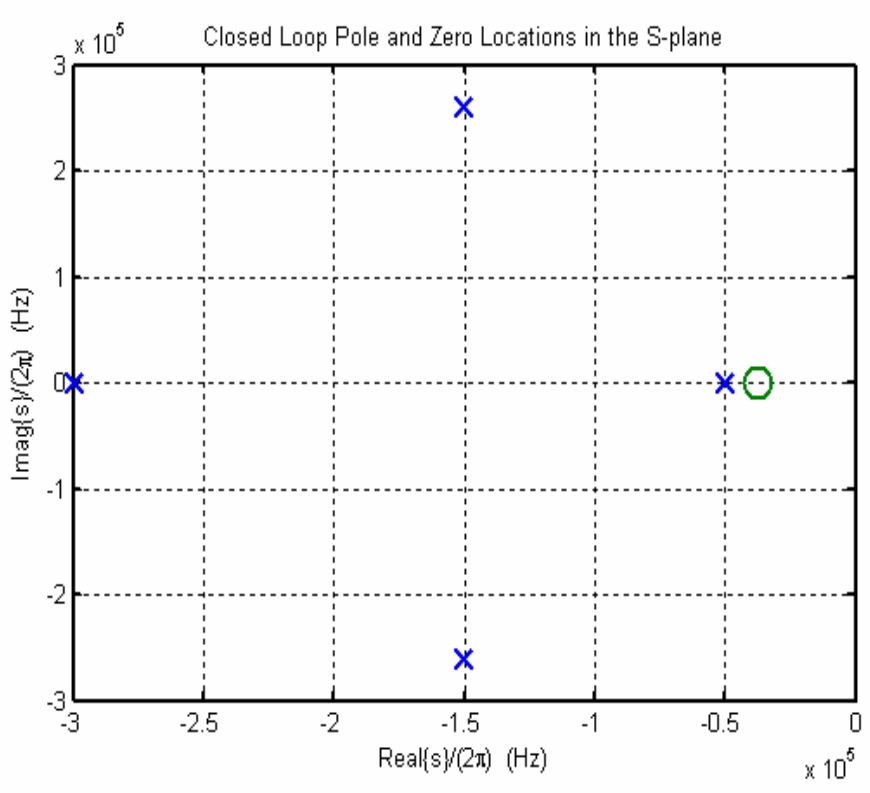
Use PLL Design Assistant to Calculate Parameters

$$H(s) = \frac{K_{LF}\left(1 + \frac{s}{w_z}\right)}{s\left(1 + \frac{s}{w_p Q_p} + \left(\frac{s}{w_p}\right)^2\right)} \quad \text{where} \quad K_{LF} = \frac{N_{nom} K}{\alpha I_{cp} K_v}$$

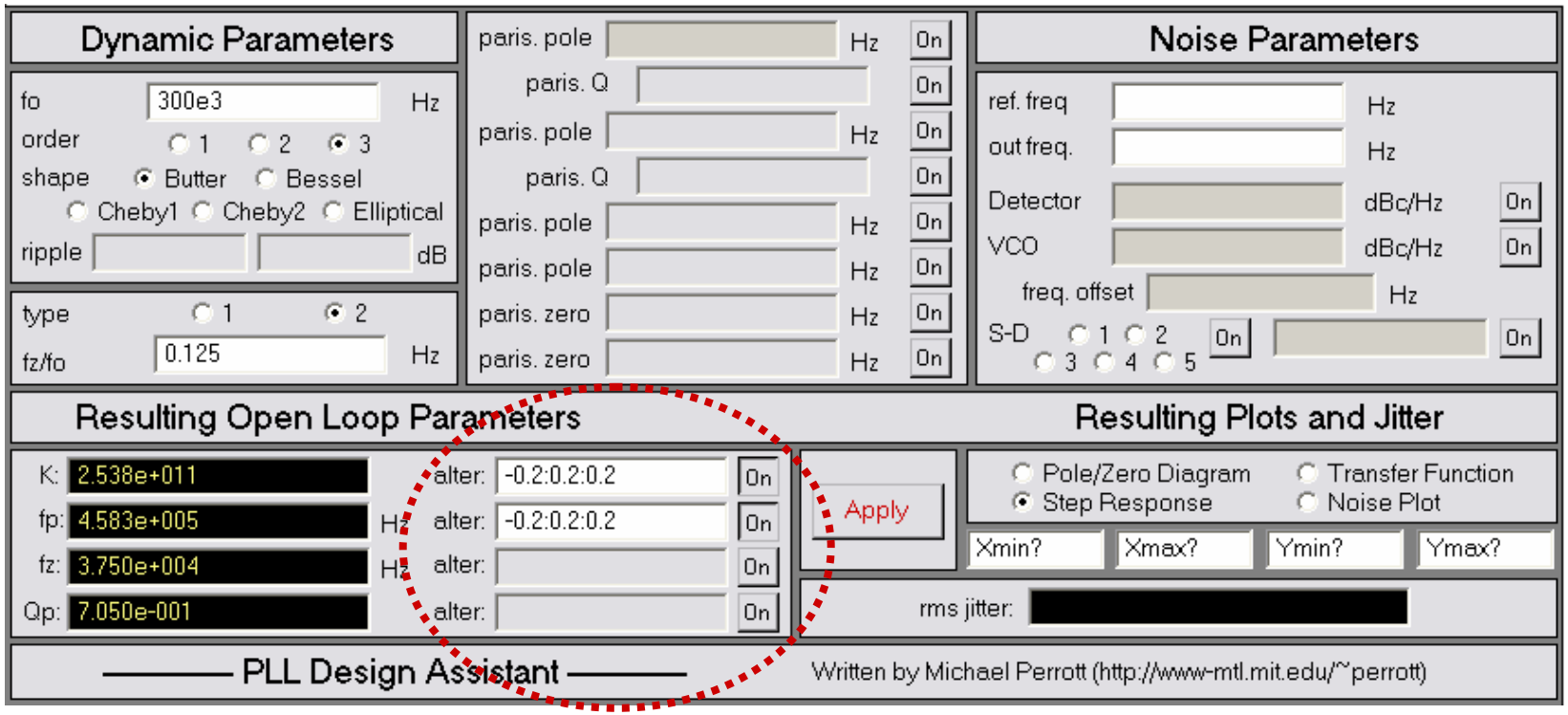


Resulting Step Response and Pole/Zero Diagram



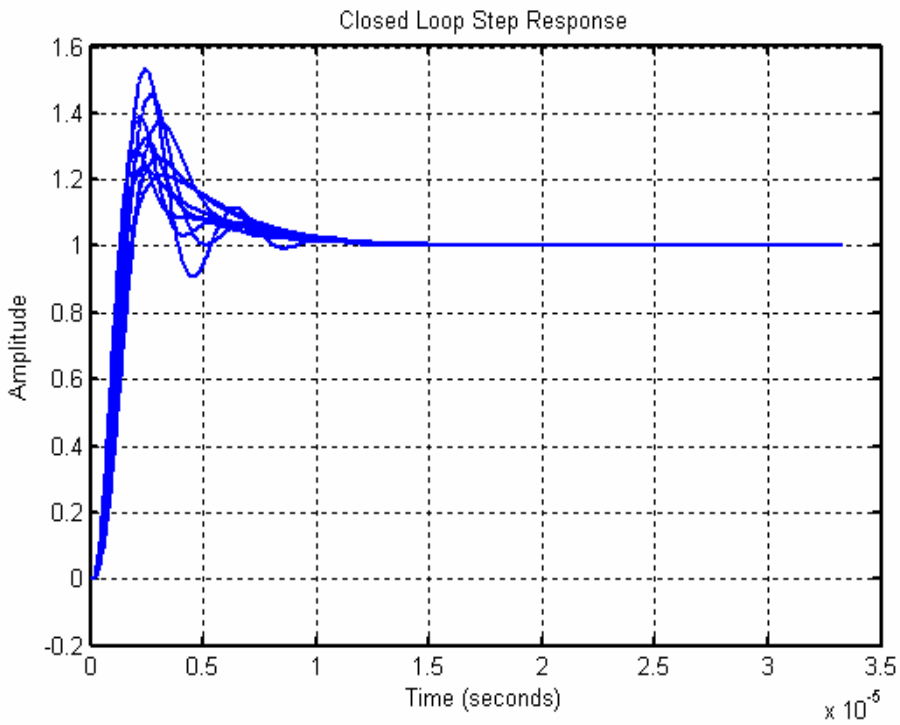


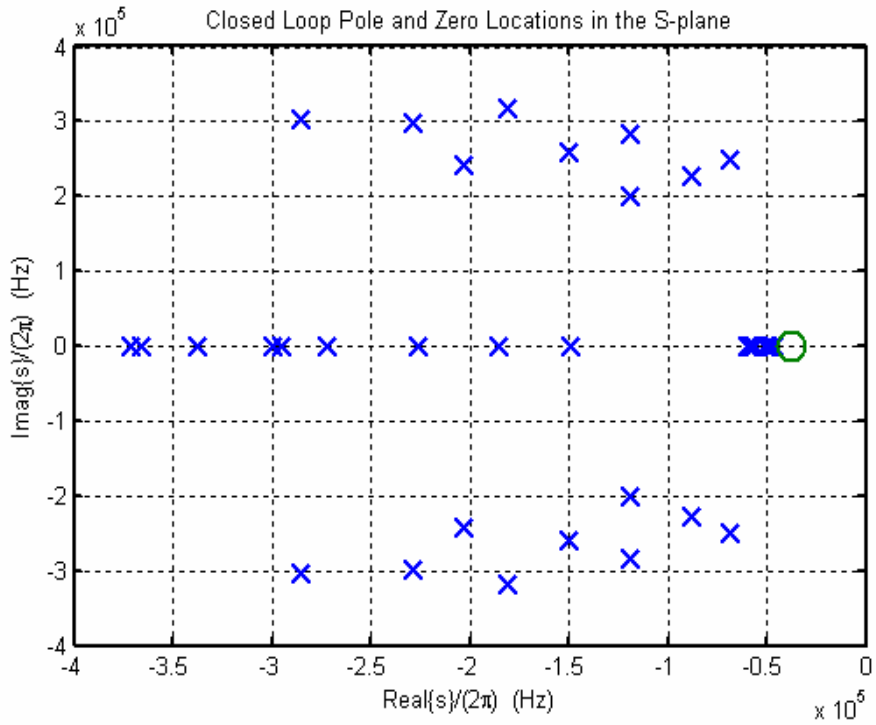
Impact of Open Loop Parameter Variations



- Open loop parameter variations can be directly entered into tool

Resulting Step Responses and Pole/Zero Diagrams



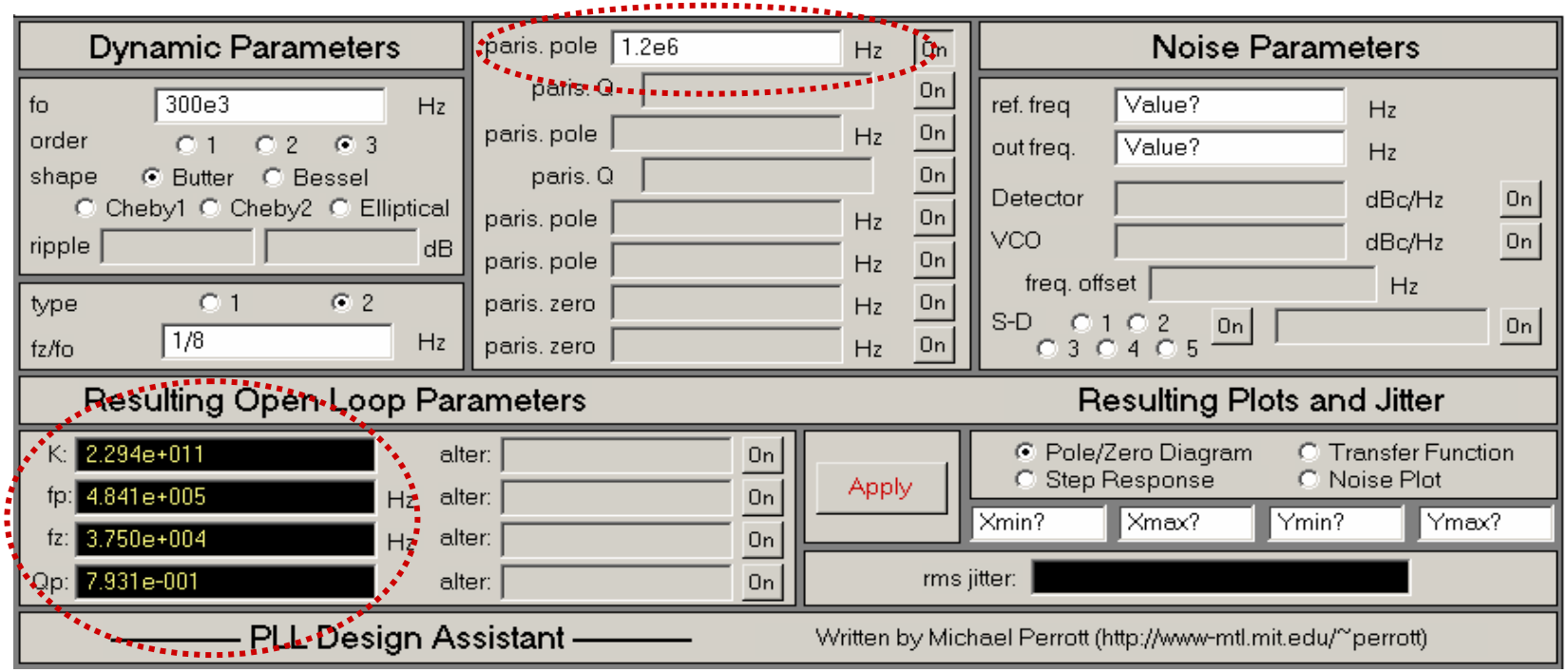


- **Impact of variations on the loop dynamics can be visualized instantly and taken into account at early stage of design**

Design with Parasitic Pole

- Include a parasitic pole at nominal value f_{p1} = 1.2MHz

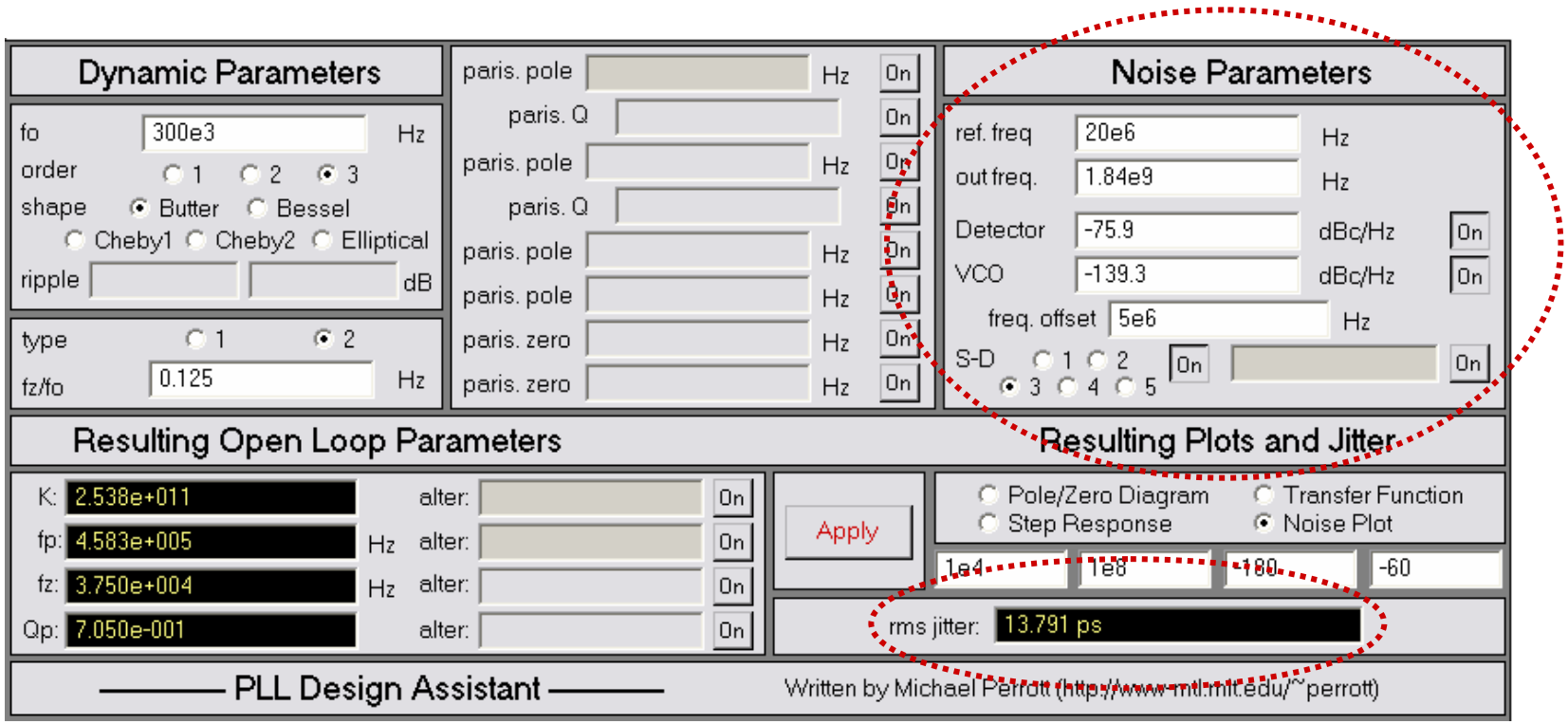
$$H(s) = \frac{K_{LF}\left(1+\frac{s}{w_z}\right)}{s\left(1+\frac{s}{w_p Q_p}+\left(\frac{s}{w_p}\right)^2\right)\left(1+\frac{s}{w_{p1}}\right)}$$



⇨ K, f_p and Q_p are adjusted to obtain the same dominant pole locations

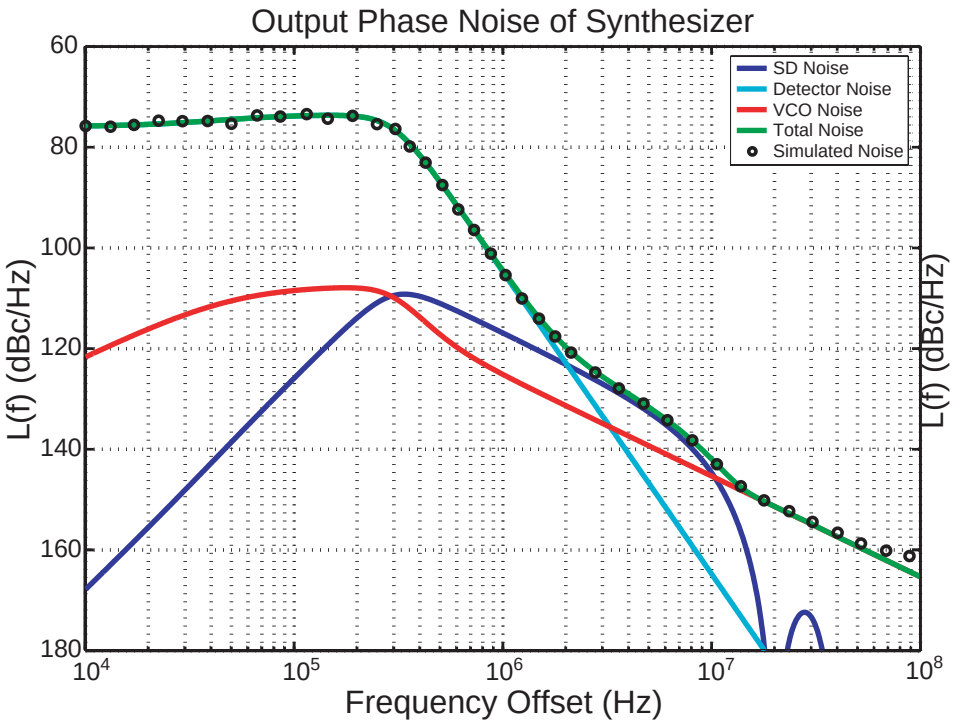
Noise Estimation

- Phase noise plots can be easily obtained
  - Jitter calculated by integrating over frequency range

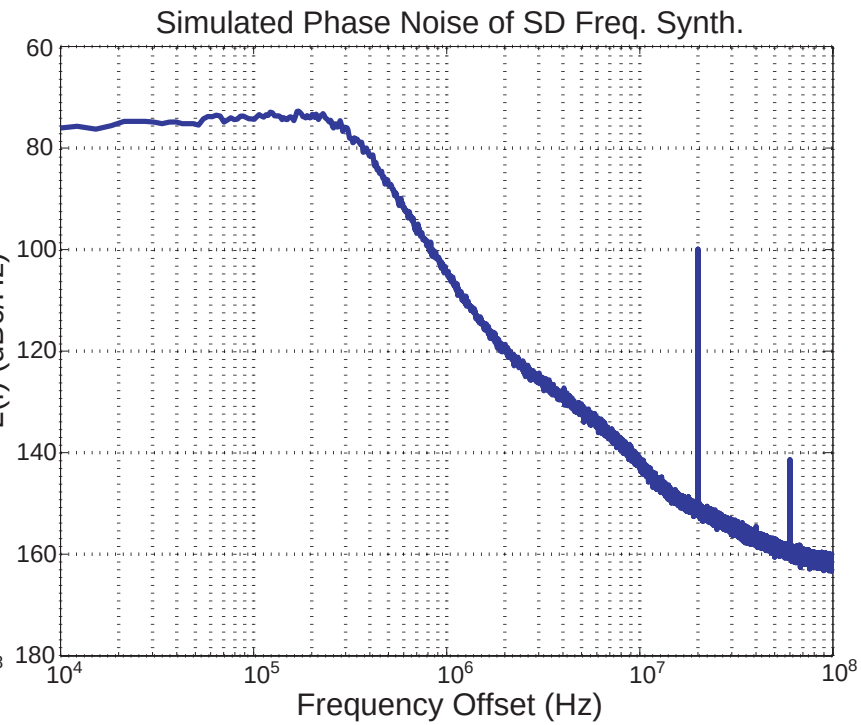


Calculated Versus Simulated Phase Noise Spectrum

Without parasitic pole:

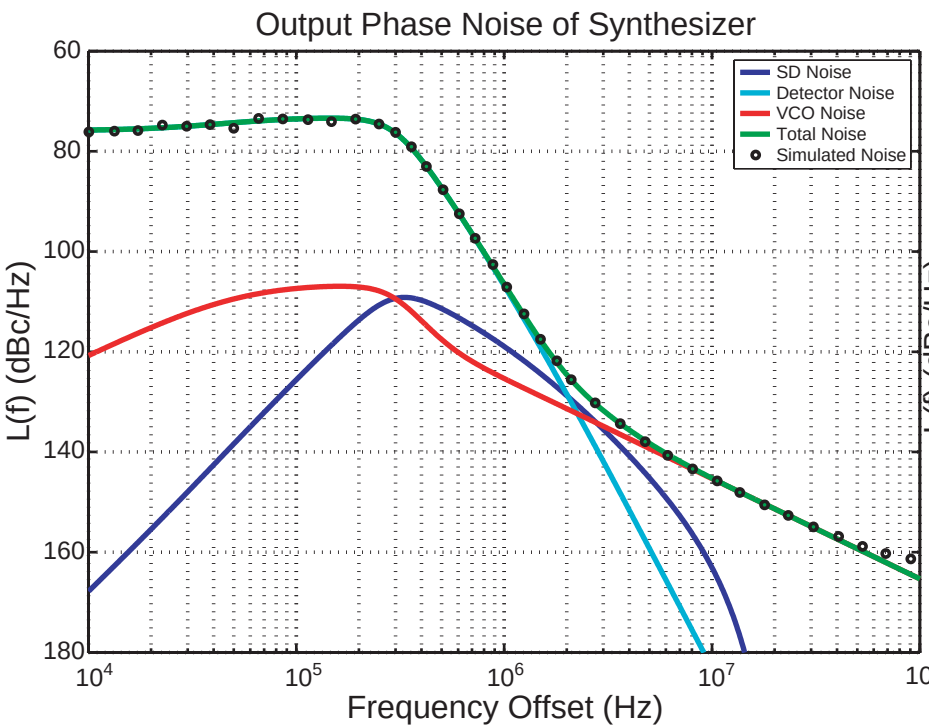


RMS jitter = 13.791ps

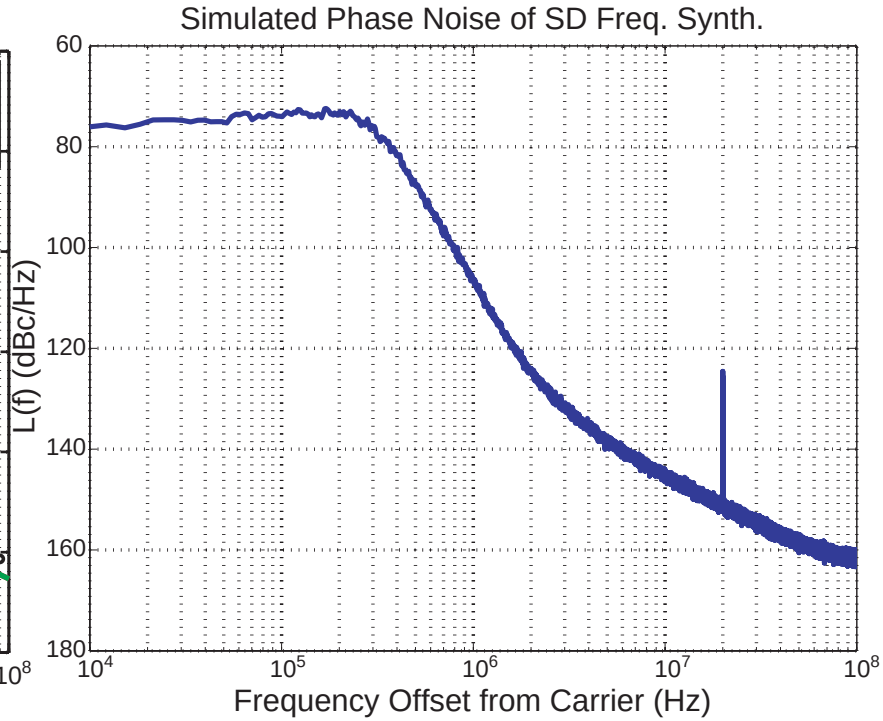


Calculated Versus Simulated Phase Noise Spectrum

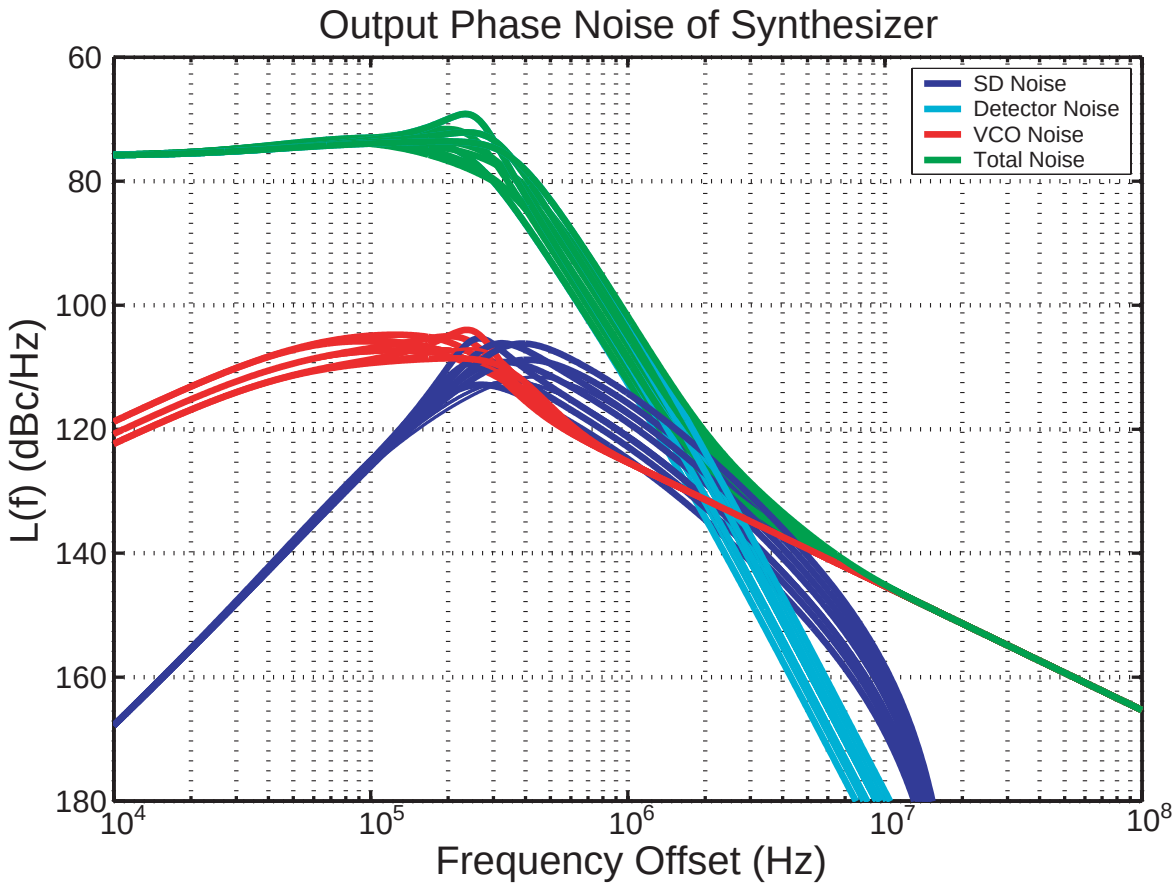
With parasitic pole at 1.2 MHz:



RMS jitter = 14.057ps



Noise under Open Loop Parameter Variations



RMS jitter = 11.678ps (min), 18.211ps (max)

- **Impact of open loop parameter variations on phase noise and jitter can be visualized immediately**

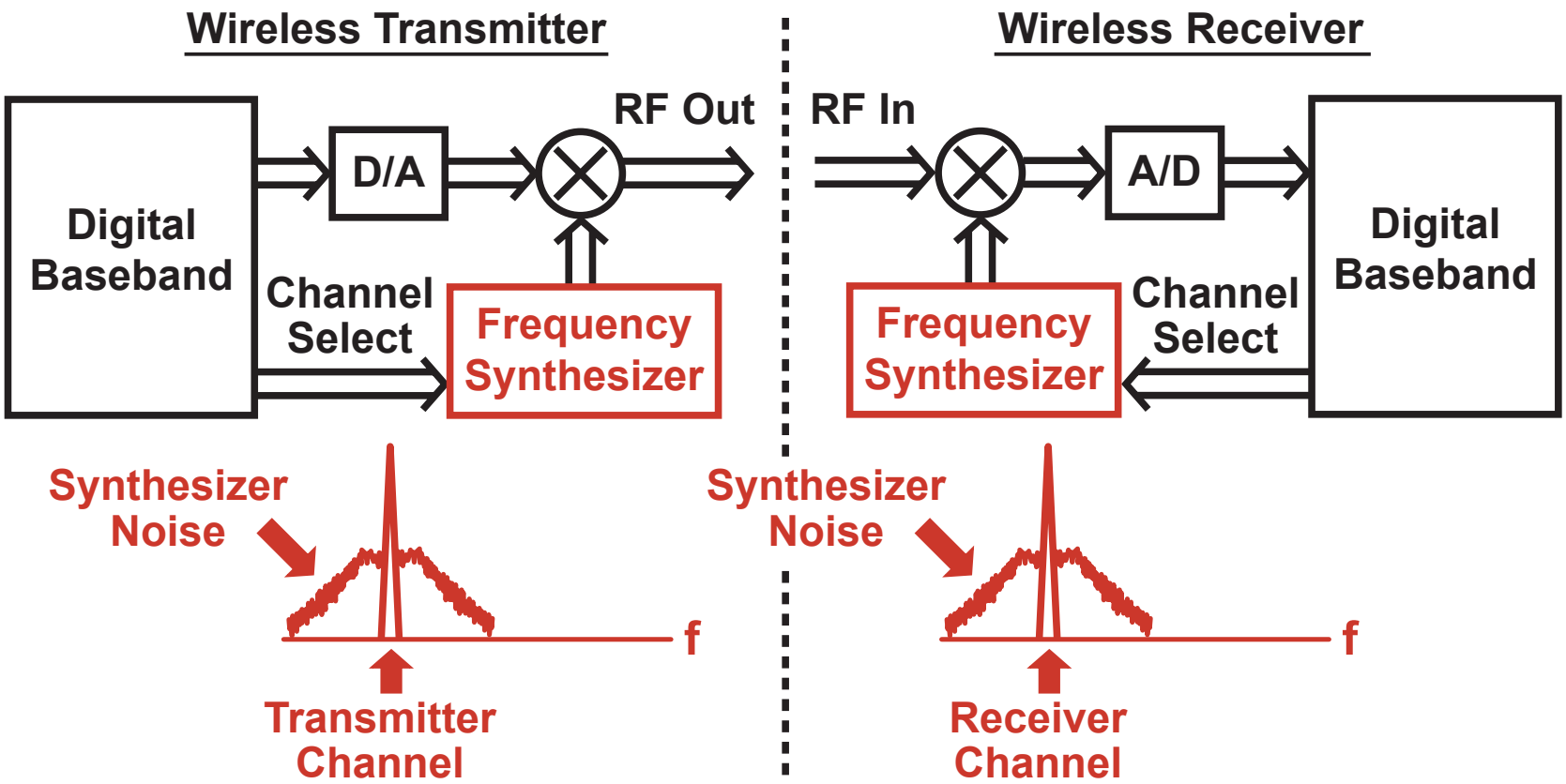
Conclusion

- **New closed loop design approach facilitates:**
  - **Accurate control of closed loop dynamics**
    - Bandwidth, Order, Shape, Type
  - **Straightforward design of higher order PLL's**
  - **Direct assessment of impact of parasitic poles/zeros**
- **Techniques implemented in a GUI-based CAD tool**

- **Beginners can quickly come up to speed in designing PLL's**
- **Experienced designers can quickly evaluate the performance of different PLL configurations**

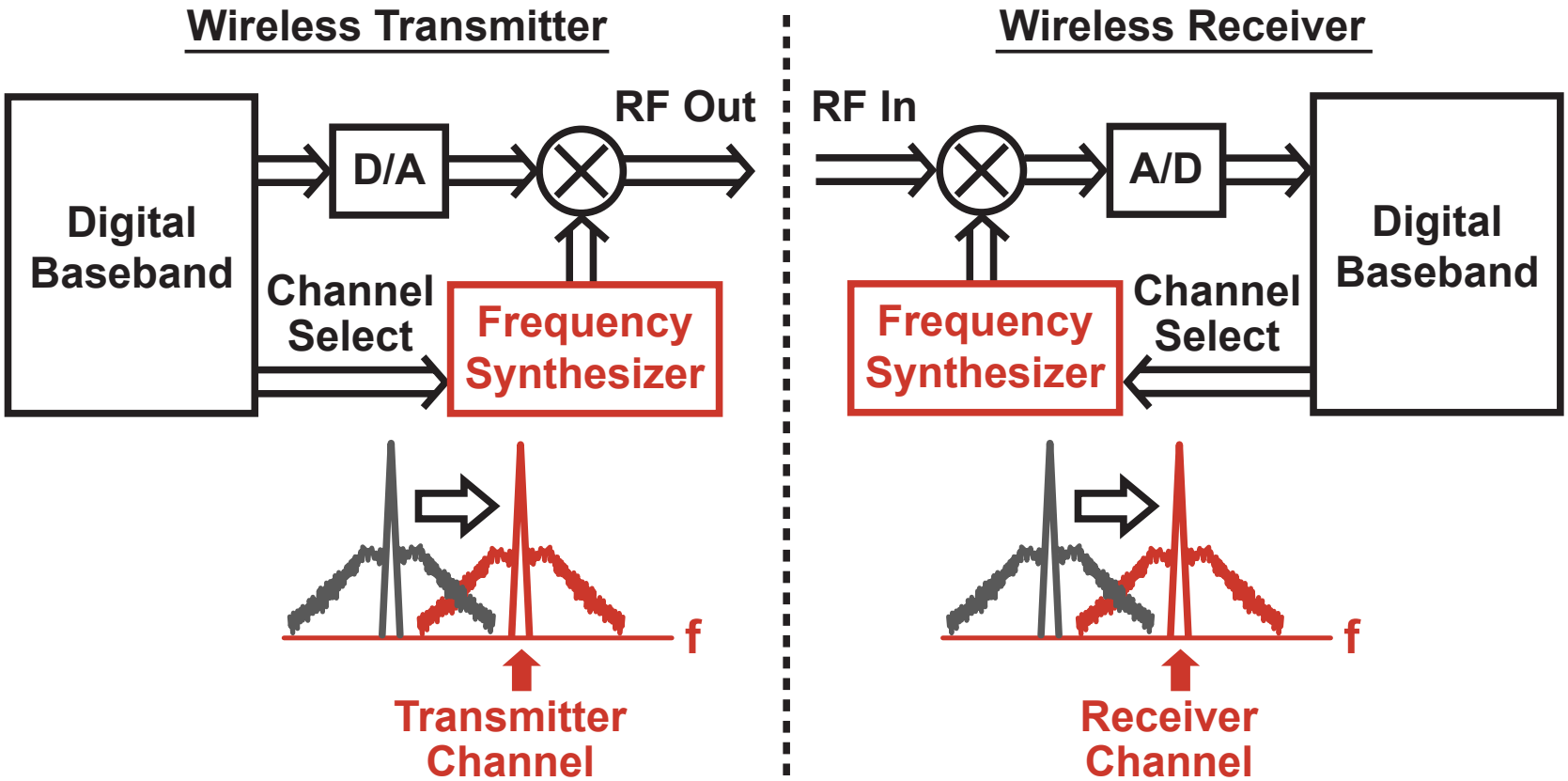
Simulation of Frequency Synthesizers

Impact of Synthesizer Noise



- Noise must be low to meet transmit mask requirement
- Noise must be low to meet receiver SNR and blocking requirements

Impact of Synthesizer Dynamic Behavior



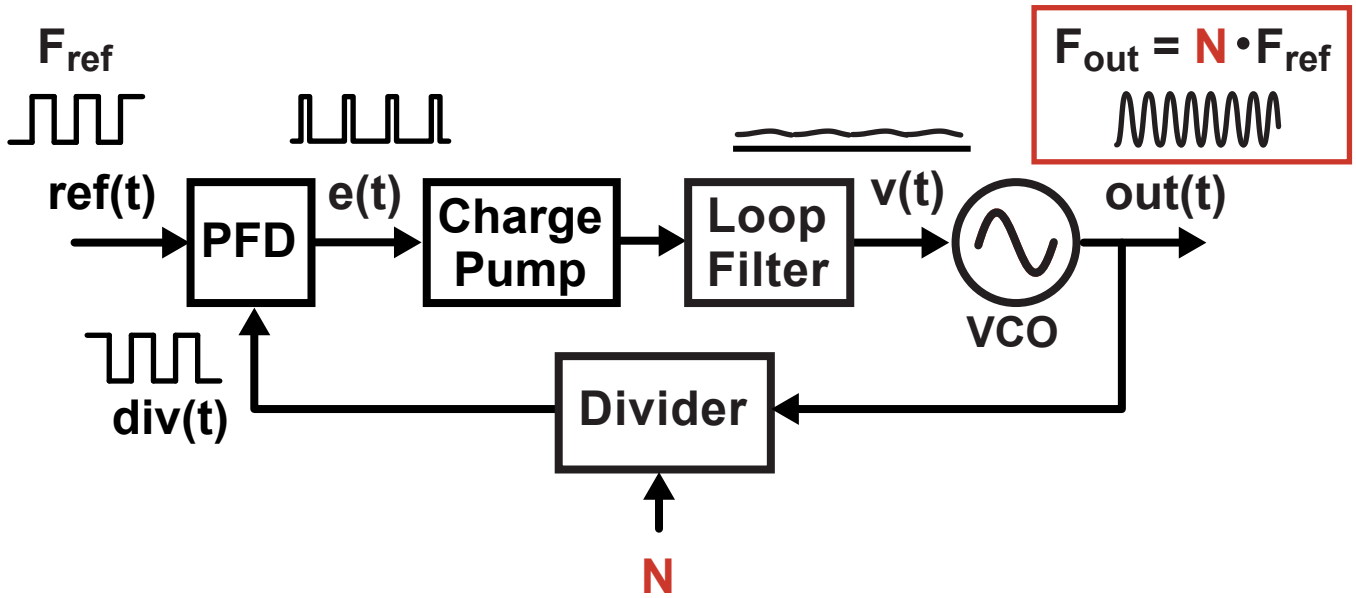
- Settling time must be fast to support channel hopping requirements

What Do We Want From a Simulator?

- Accurate estimation of synthesizer performance
  - Noise spectral density
  - Dynamic behavior
- Fast computation to allow use in IC design flow
- Simple to use
  - C++, Verilog, Matlab

Background Information

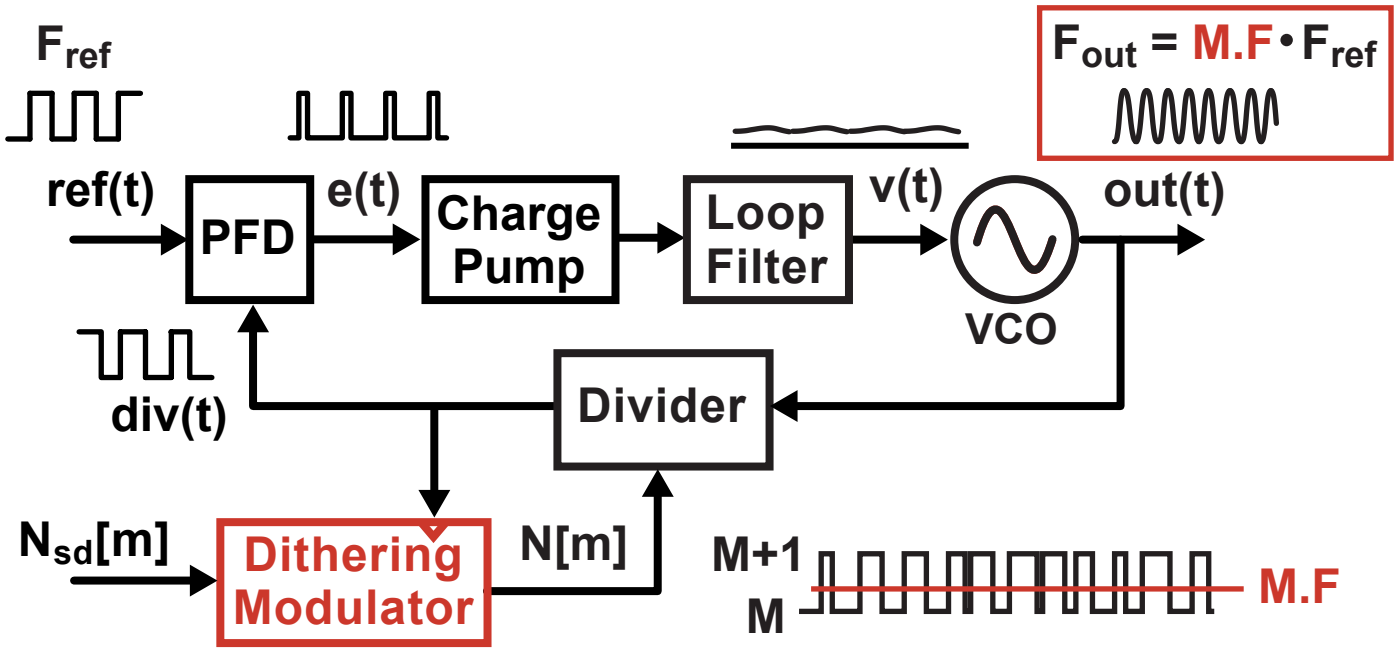
Integer-N Frequency Synthesizer



- VCO ⇨ produces high frequency sine wave
- Divider ⇨ divides down VCO frequency
- PFD ⇨ compares phase of ref and div
- Loop filter ⇨ extracts phase error information

Poor frequency resolution

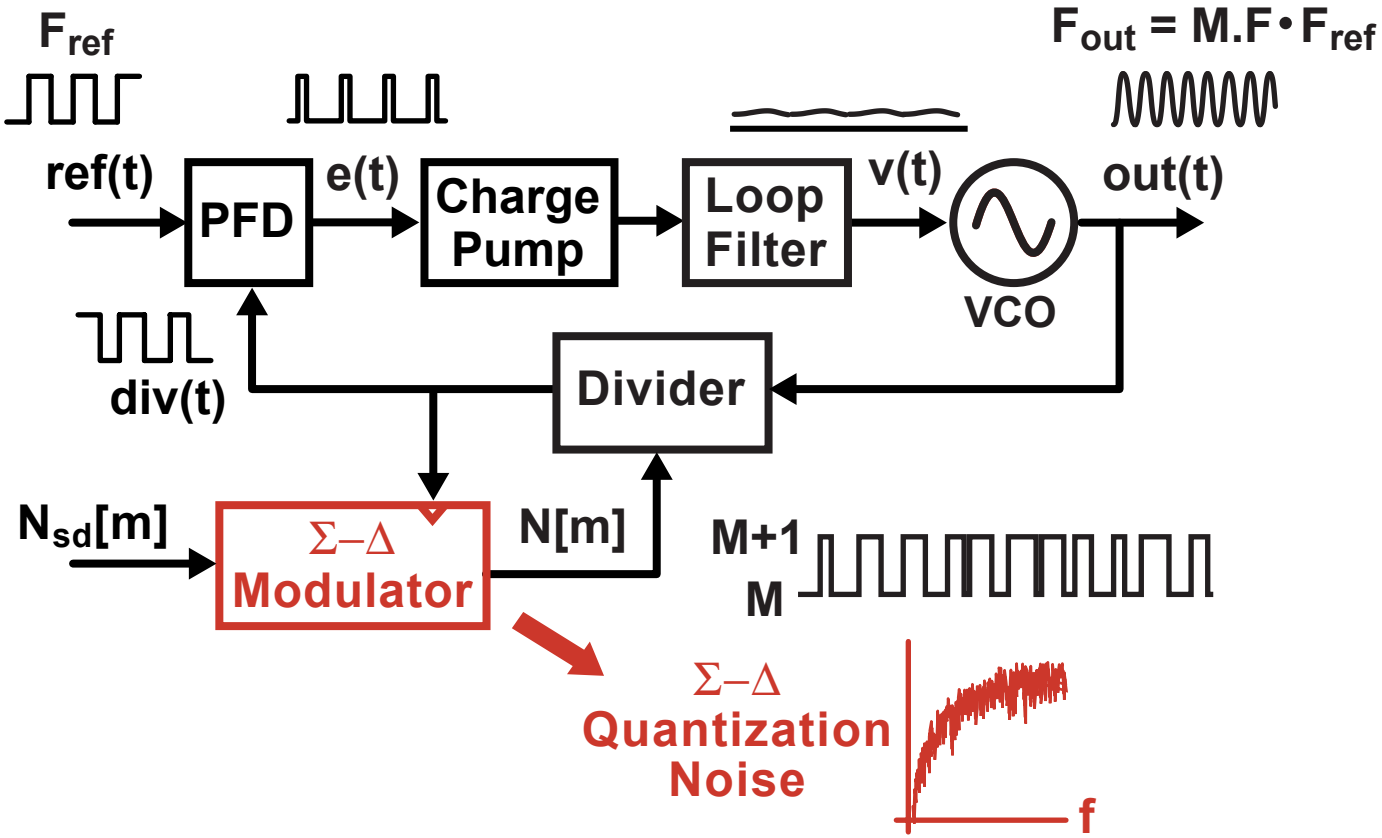
Fractional-N Frequency Synthesis



- Divide value is dithered between integer values
- Fractional divide values can be realized!

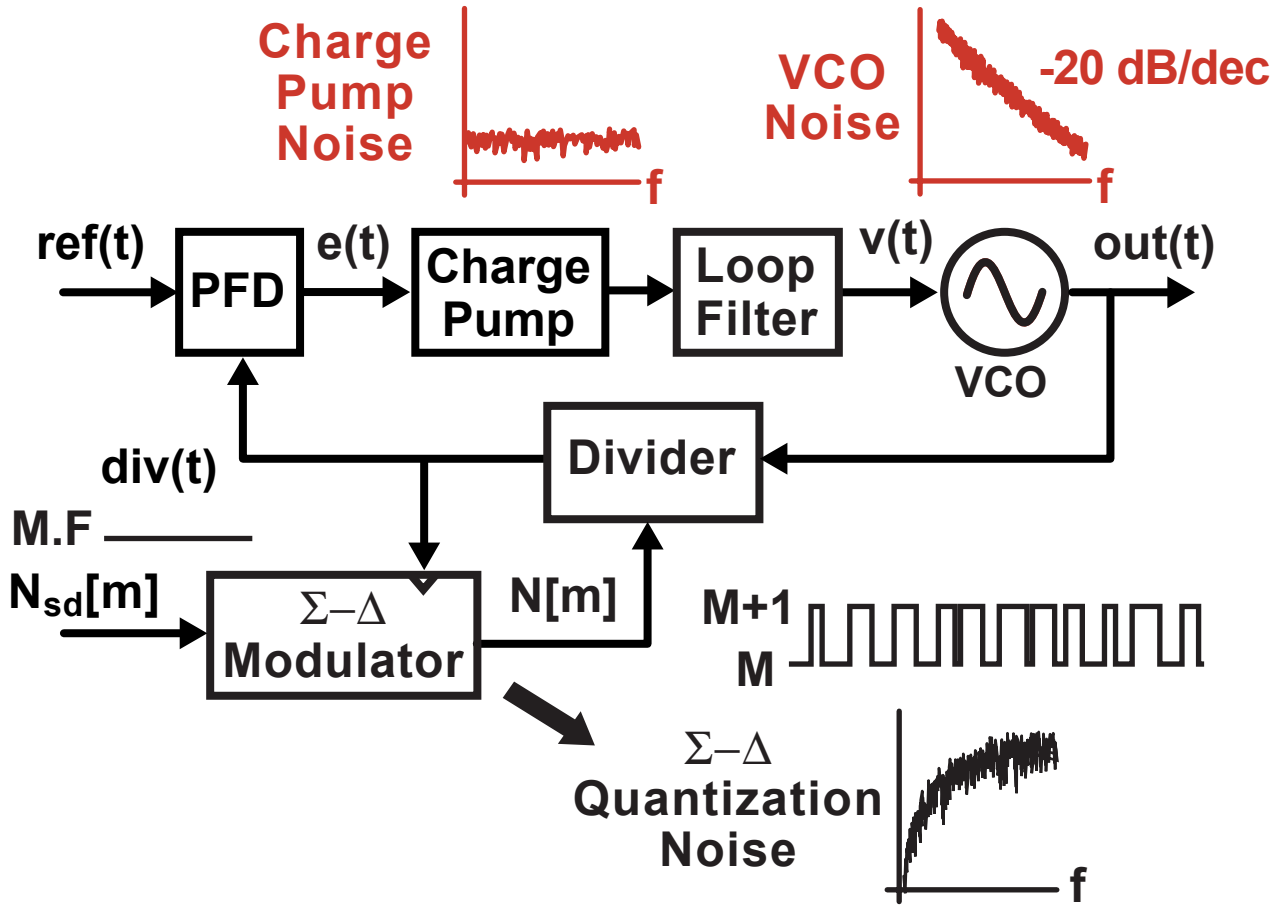
Very high frequency resolution

Σ–Δ Fractional-N Frequency Synthesis



- Dither using a Σ–Δ modulator
- Quantization noise is shaped to high frequencies

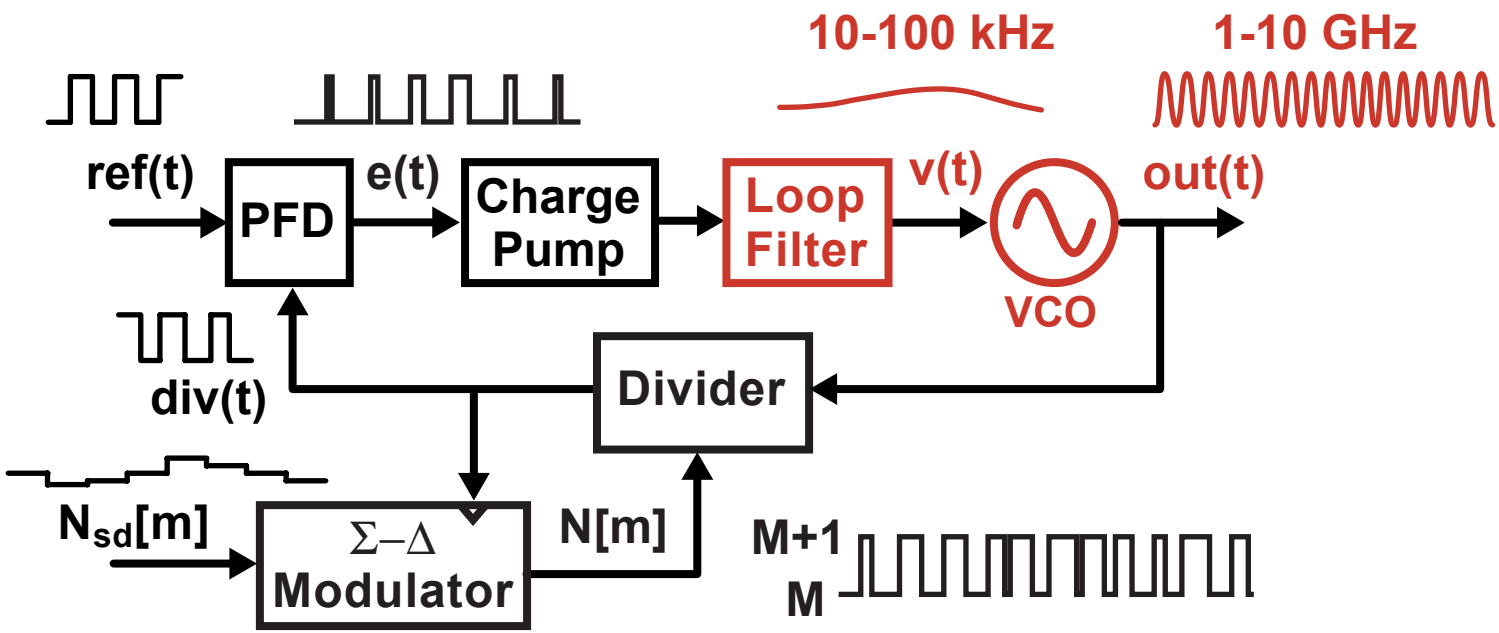
Other Noise Sources



- Charge pump noise
- VCO noise

Problems with Current Simulators

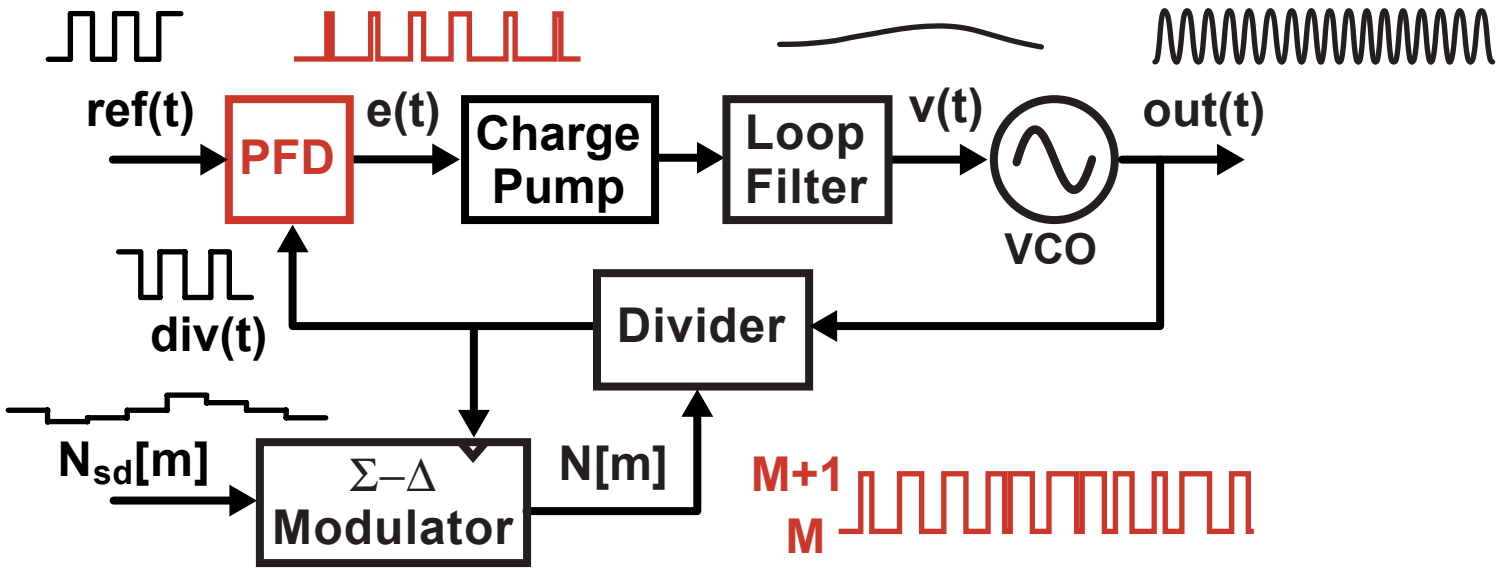
Problem 1: Classical Simulators are Slow



- High output frequency ⇨ High sample rate
- Long time constants ⇨ Long time span for transients

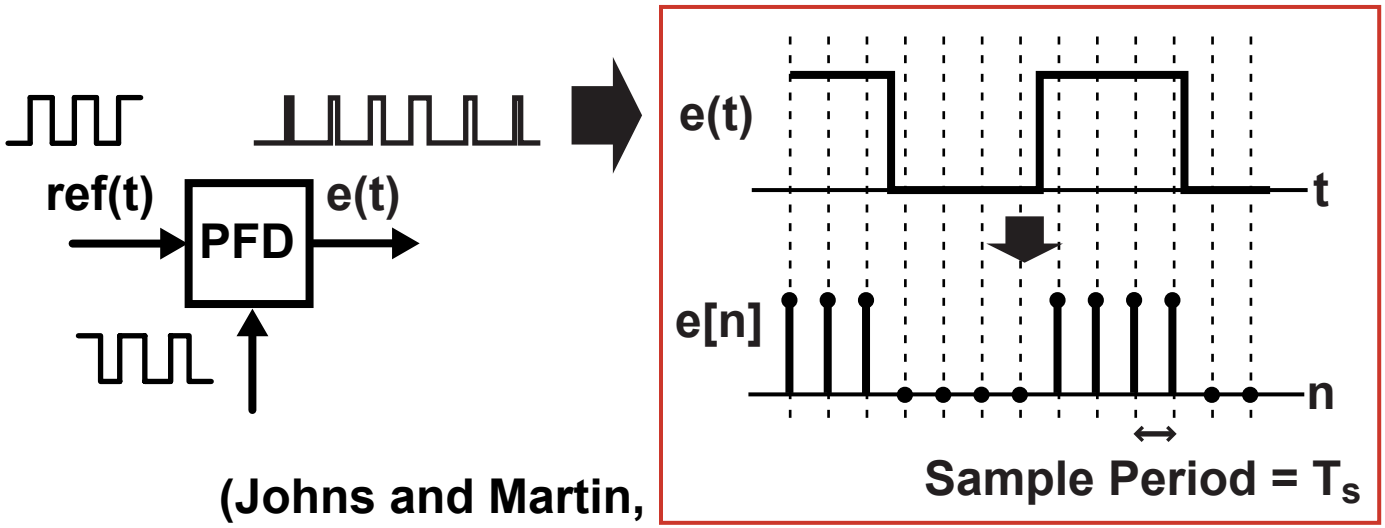
Large number of simulation time steps required

Problem 2: Classical Simulators Are Inaccurate



- PFD output is not bandlimited
  - PFD output must be simulated in discrete-time
- Phase error is inaccurately simulated
- Non-periodic dithering of divider complicates matters

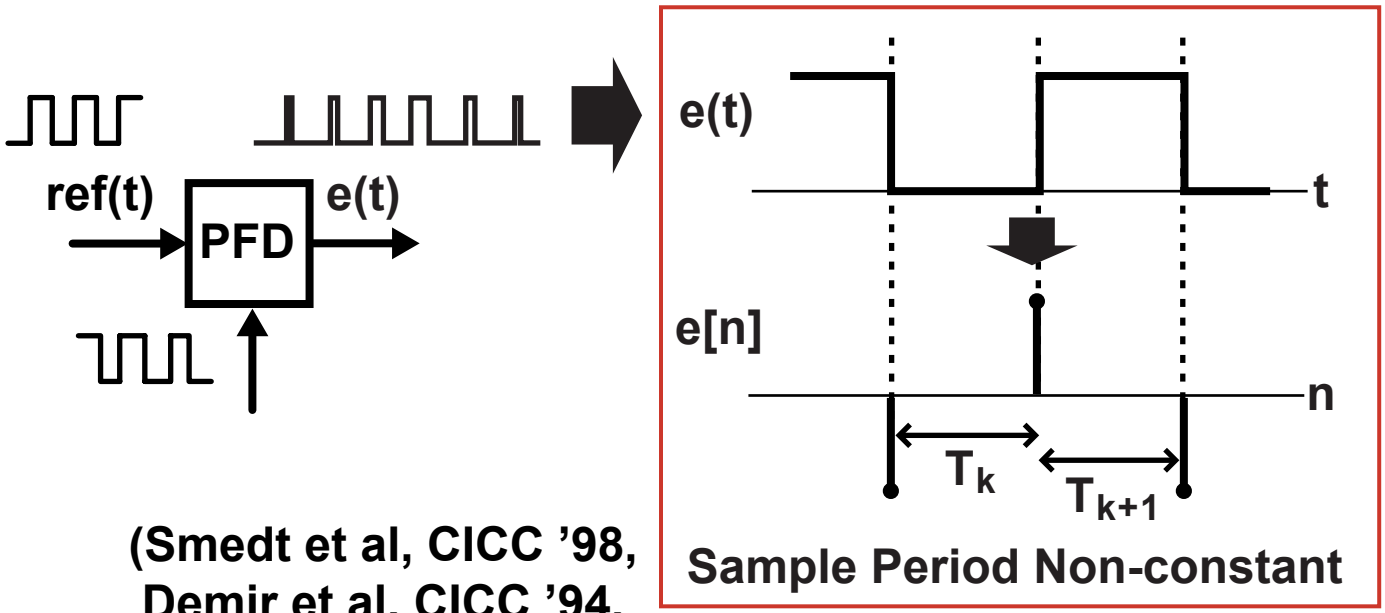
Example: Classical Constant-Time Step Method



(Johns and Martin, Analog Integrated Circuit Design)

- Directly sample the PFD output according to the simulation sample period
  - Simple, fast, readily implemented in Matlab, Verilog, C++
- Issue – quantization noise is introduced
  - This noise overwhelms the PLL noise sources we are trying to simulate

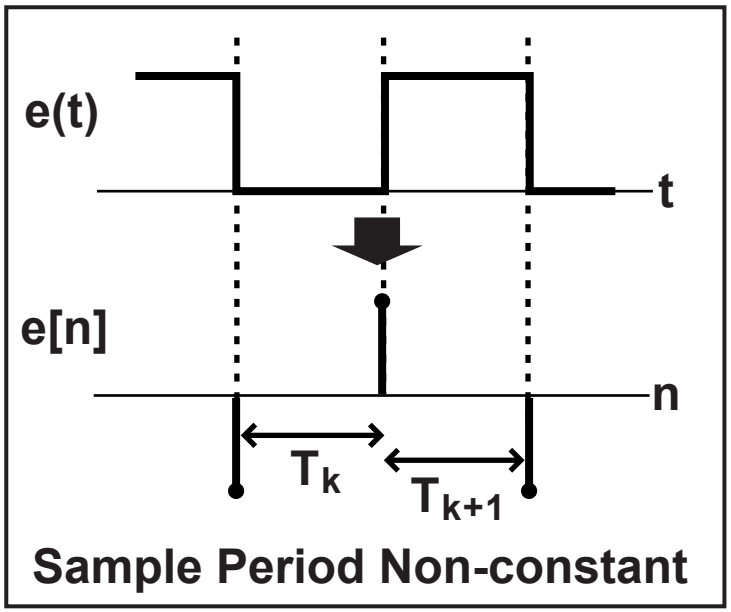
Alternative: Event Driven Simulation

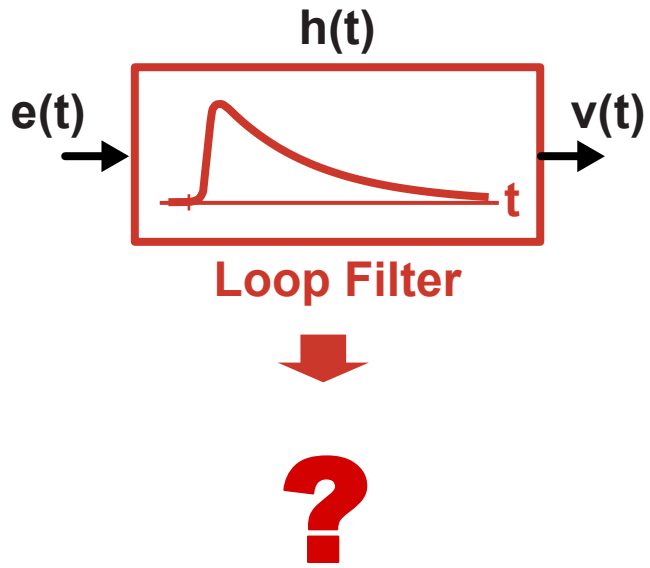


(Smedt et al, CICC '98,
Demir et al, CICC '94,
Hinz et al, Circuits and Systems '00)

- Set simulation time samples at PFD edges
  - Sample rate can be lowered to edge rate!

Issue: Simulation of Filter Blocks is Complicated

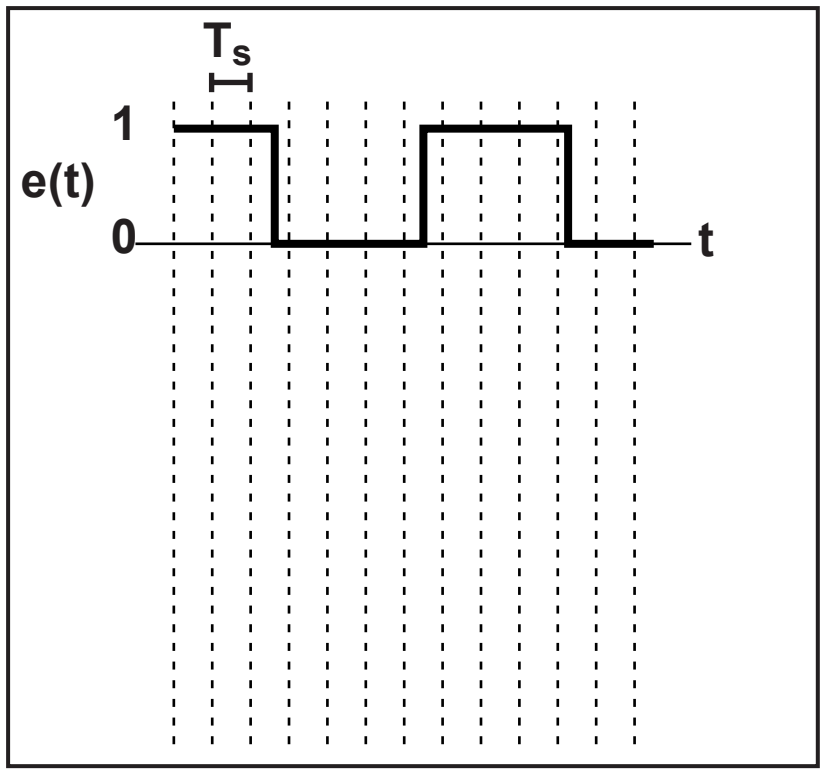


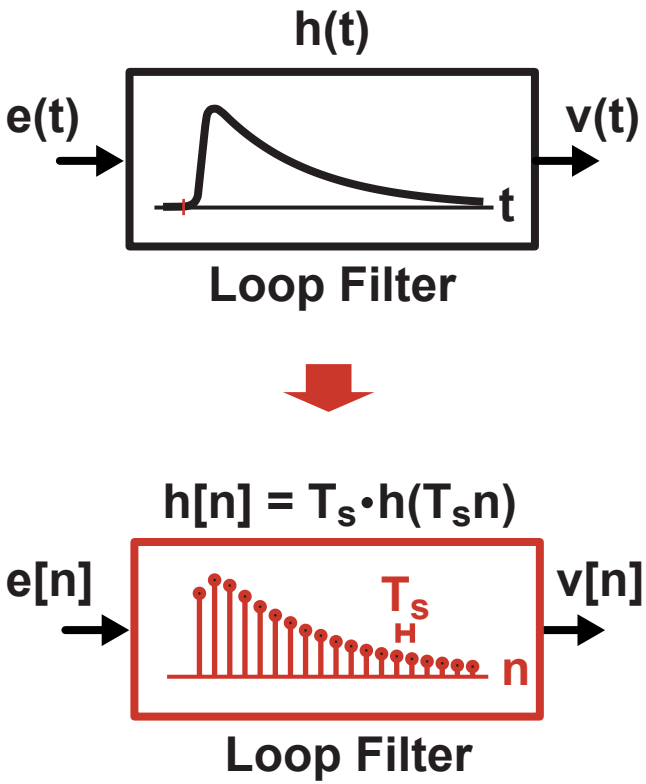


- Filtering computation must deal with non-constant time step
  - Closed-form calculation is tedious
  - Iterative computation is time-consuming
- Complicates Verilog, Matlab, or C++ implementation

Is there a better way?

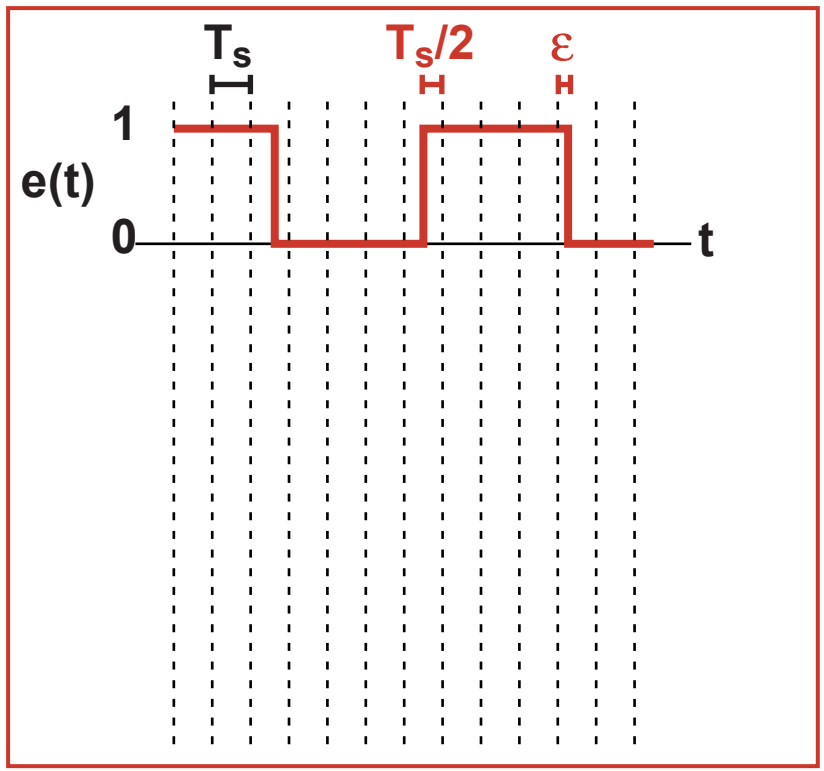
Proposed Approach: Use Constant Time Step

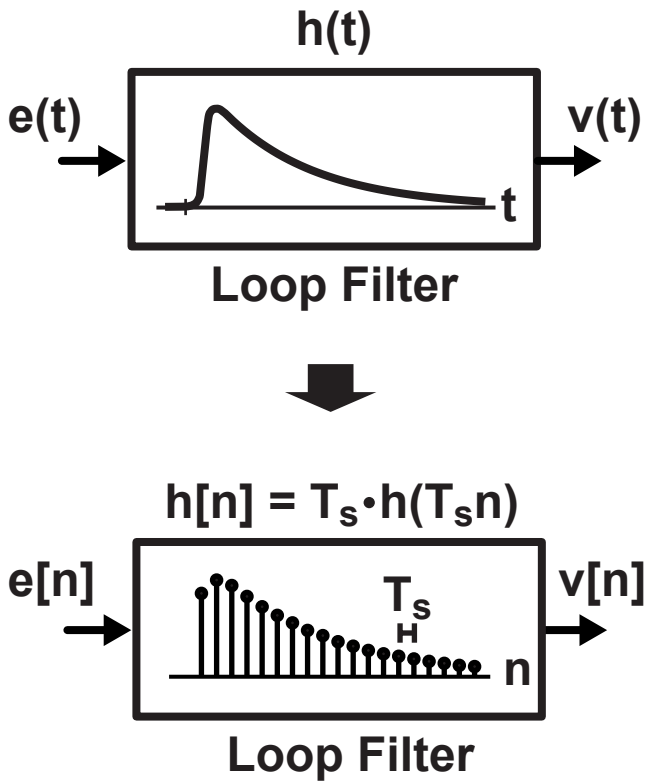




- Straightforward CT to DT transformation of filter blocks
  - Use bilinear transform or impulse invariance methods
- Overall computation framework is fast and simple
  - Simulator can be based on Verilog, Matlab, C++

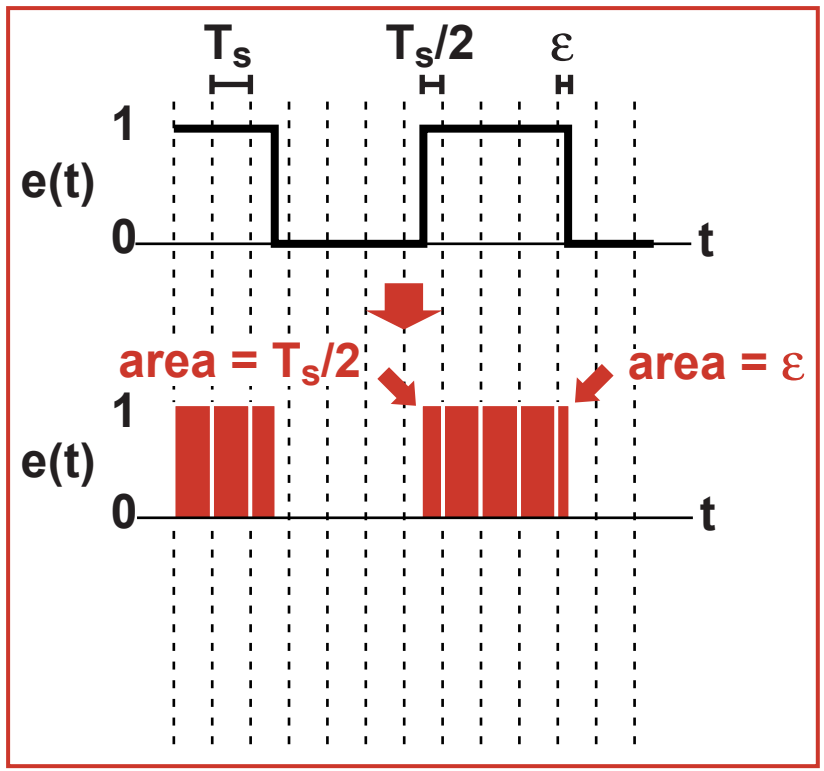
Problem: Quantization Noise at PFD Output

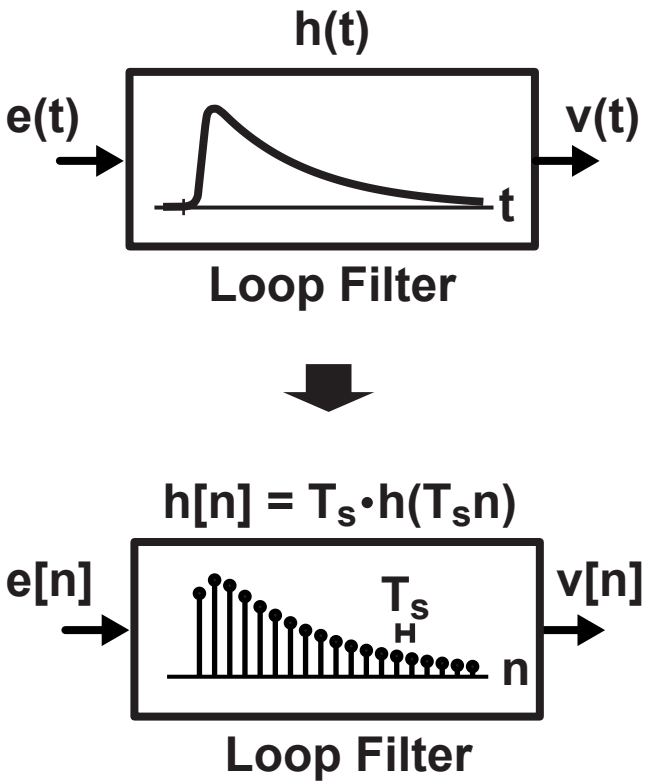




- Edge locations of PFD output are quantized
  - Resolution set by time step: T_s
- Reduction of T_s leads to long simulation times

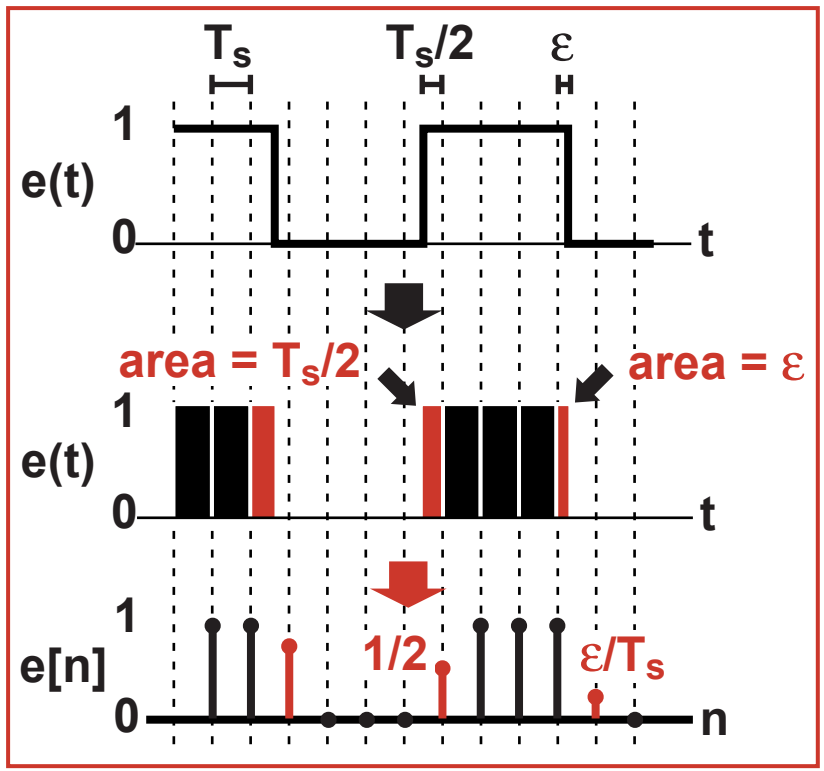
Proposed Approach: View as Series of Pulses

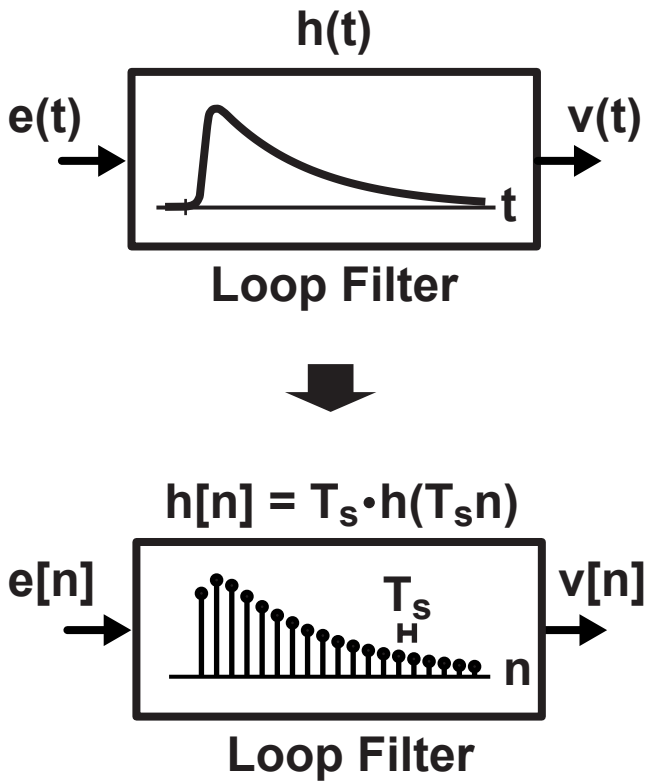




- Area of each pulse set by edge locations
- Key observations:
  - Pulses look like impulses to loop filter
  - Impulses are parameterized by their area and time offset

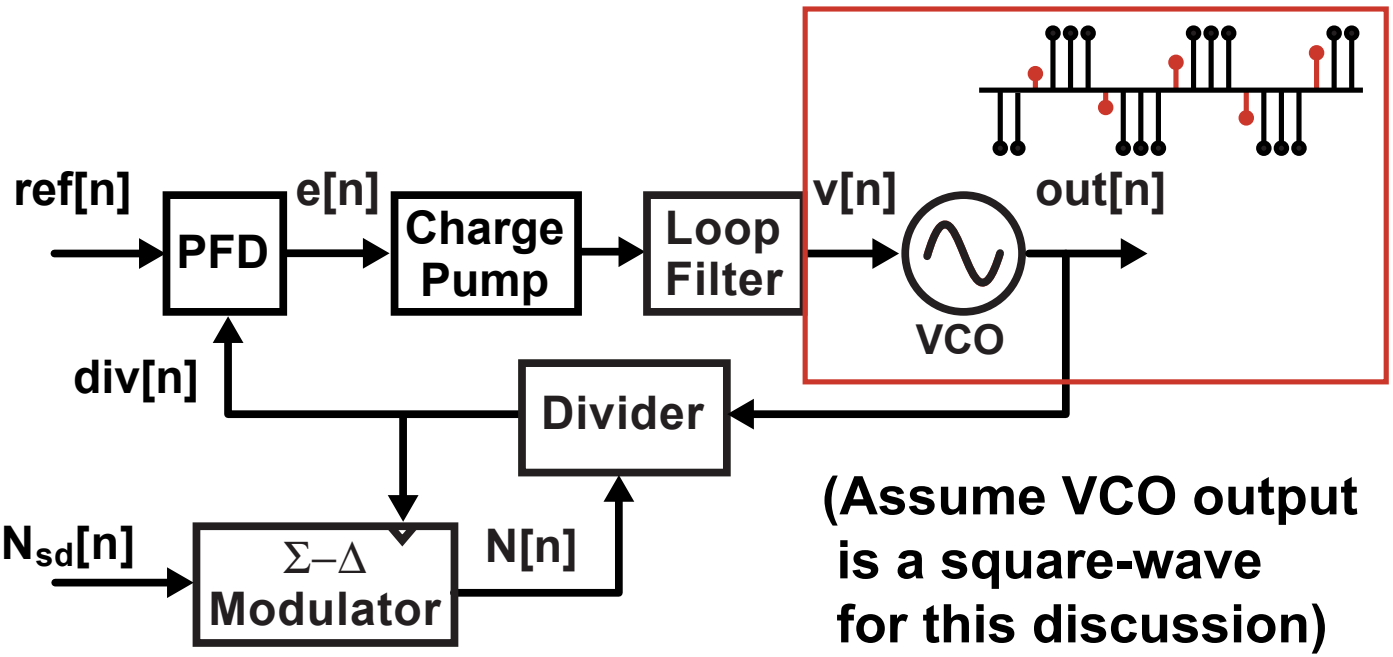
Proposed Method





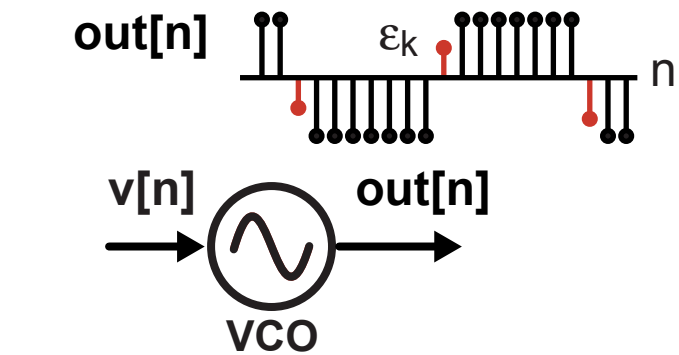
- Set e[n] samples according to pulse areas
  - Leads to very accurate results
    - Mathematical analysis given in paper
  - Fast computation

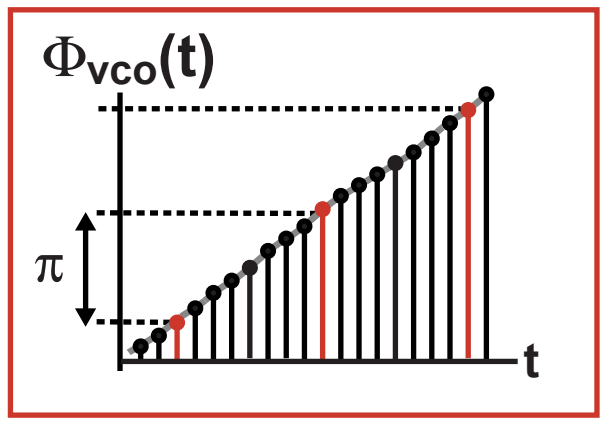
Implementation Overview



- Compute transition values in VCO block

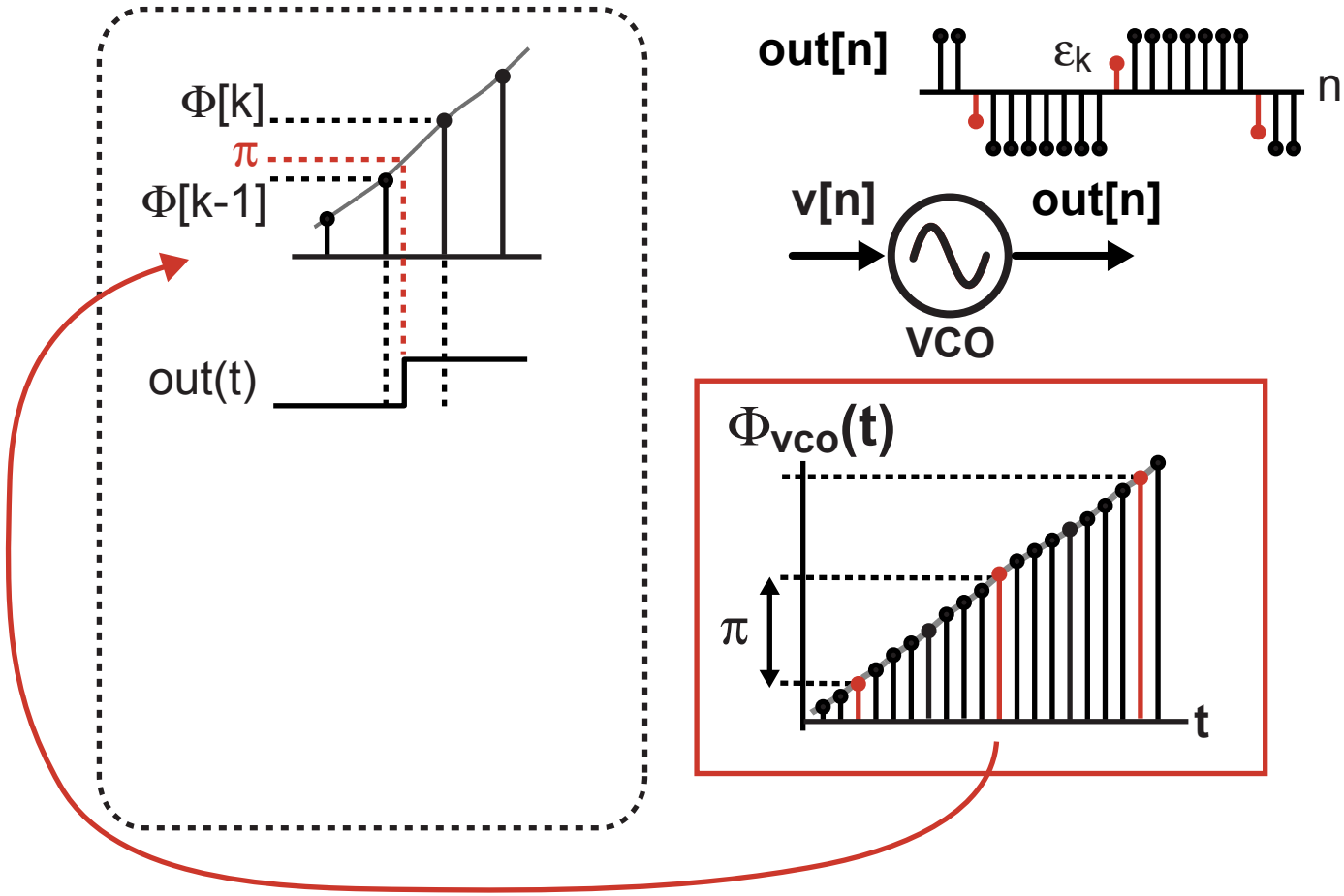
Calculation of Transition Values





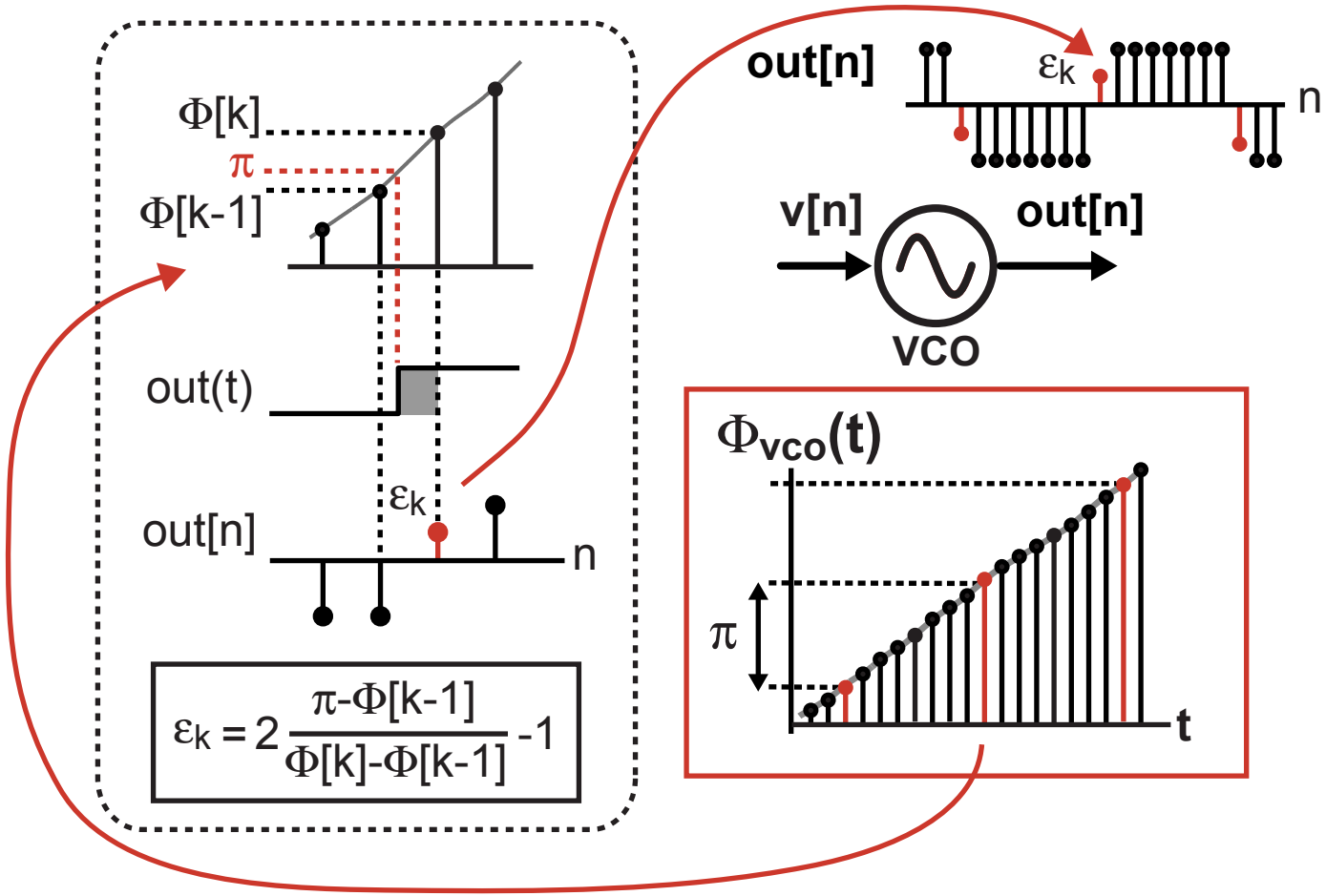
- Model VCO based on its phase

Calculation of Transition Values



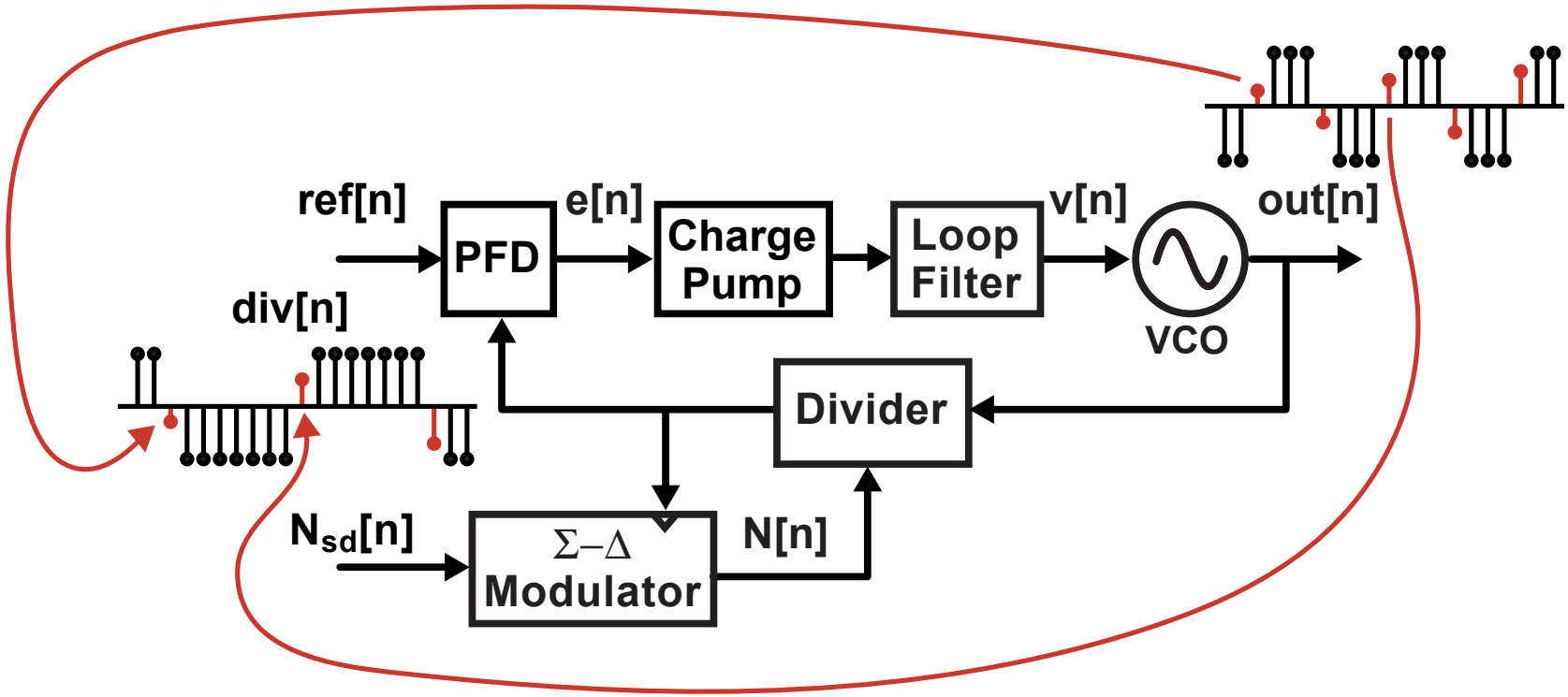
- Determine output transition time according to phase

Calculation of Transition Values



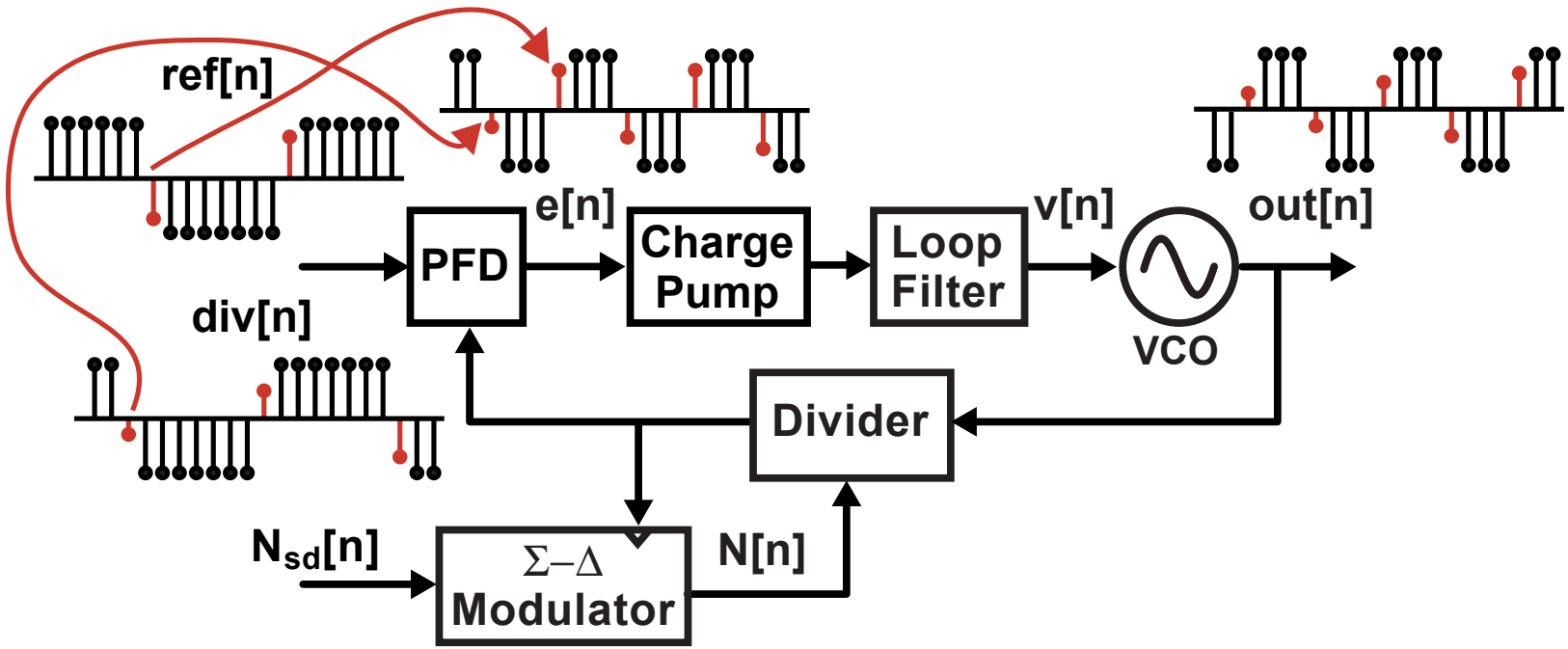
- Use first order interpolation to determine transition value

Implementation Overview



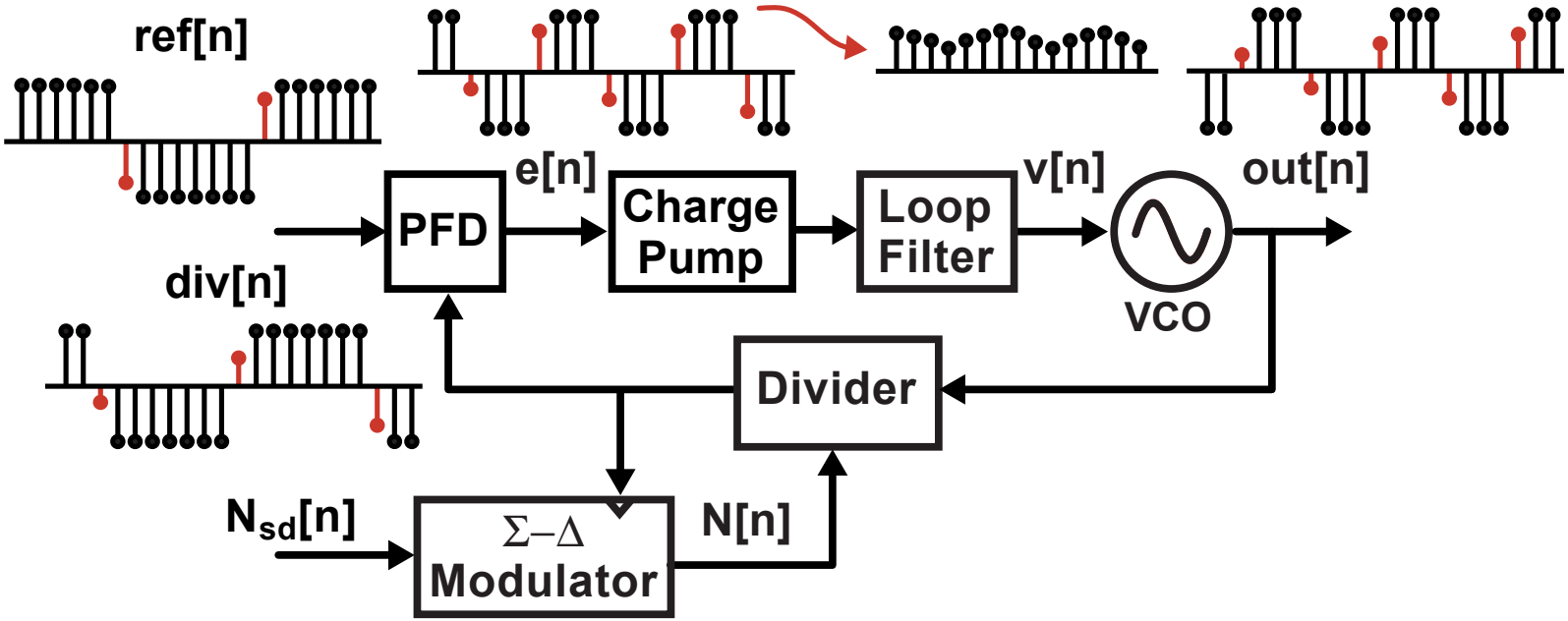
- Compute transition values in VCO block
- Pass transition information in Divider block

Implementation Overview



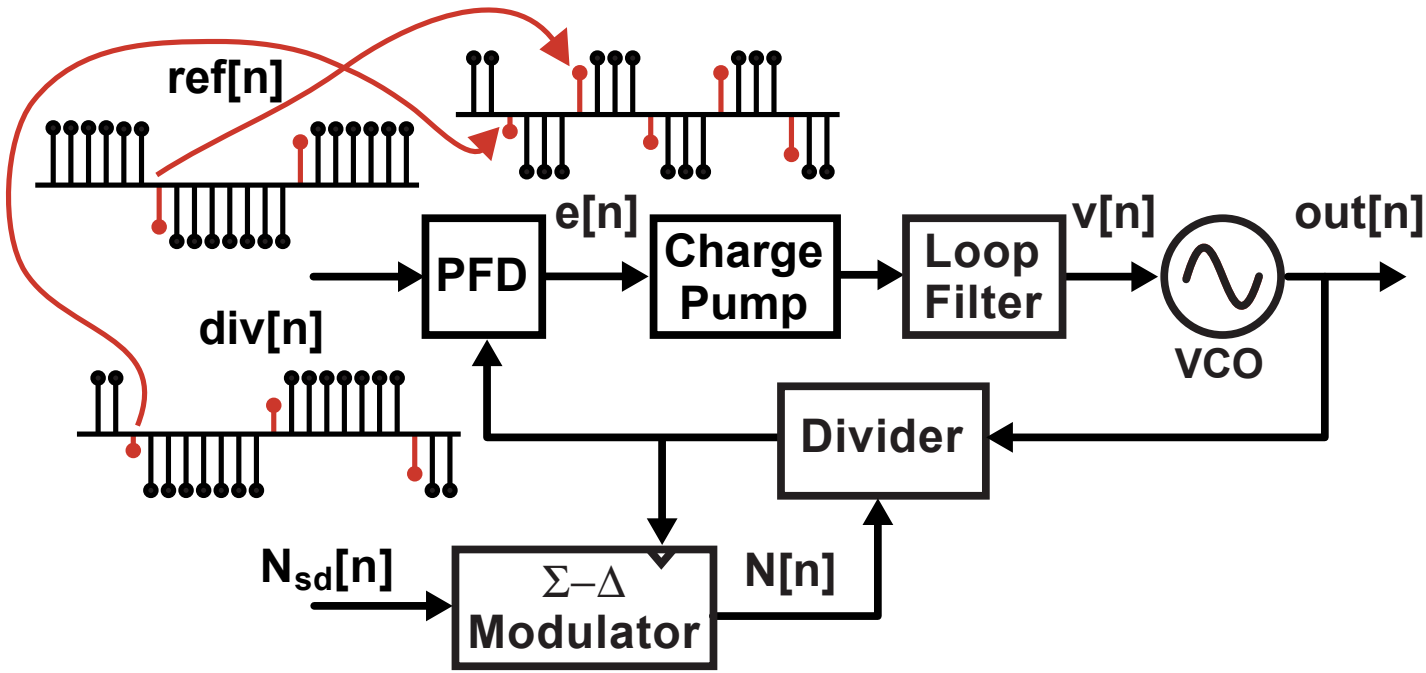
- Compute transition values in VCO block
- Pass transition information in Divider block
- Compute transition values for PFD output

Implementation Overview



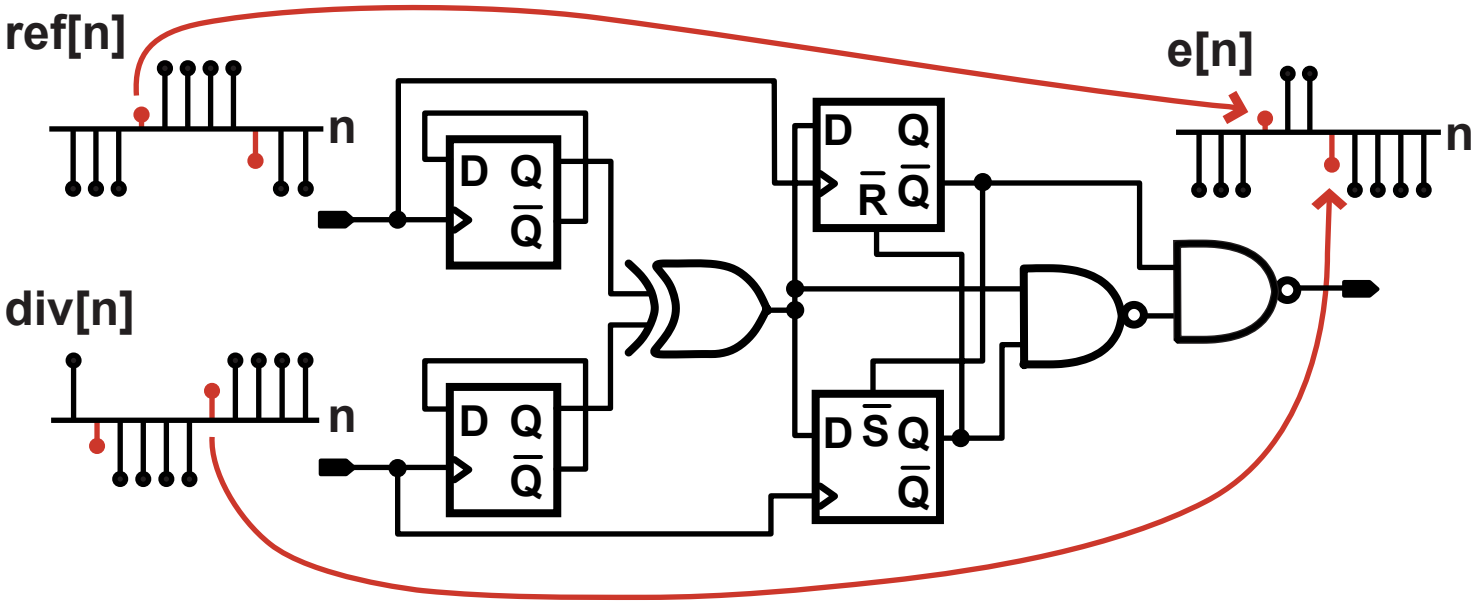
- Compute transition values in VCO block
- Pass transition information in Divider block
- Compute transition values for PFD output
- Compute Filter output

Implementation Overview



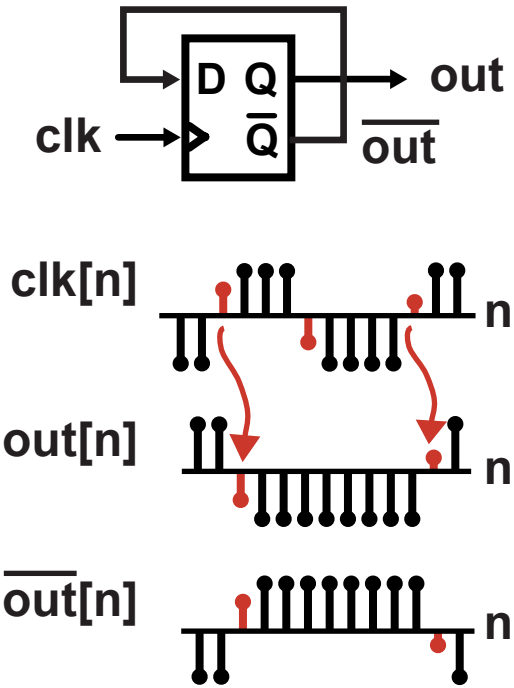
- Compute transition values in VCO block
- Pass transition information in Divider block
- Compute transition values for PFD output
- Compute Filter output

Computation of PFD Output



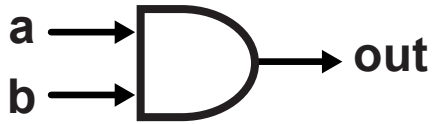
- Goal: compute transition information in terms of primitive blocks (registers, XOR gates, etc.)
  - Allows straightforward implementation in simulator
  - Accommodates a rich variety of PFD structures

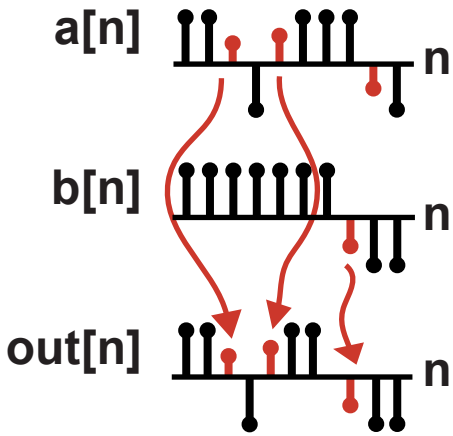
Implementation of Primitives - Registers



- Relevant timing information is contained in the clock signal
  - Transfer transition information from the clock to the register output
  - Complement output using a sign change

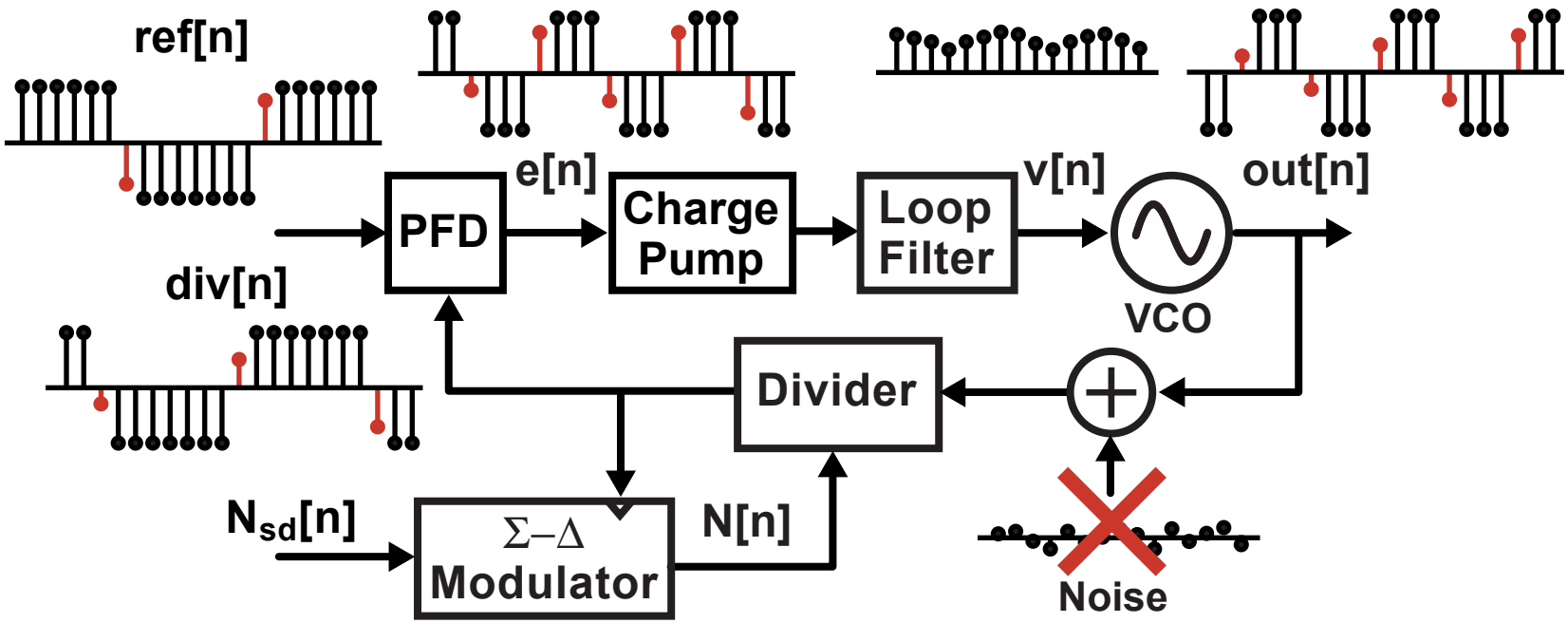
Implementation of Primitives – Logic Gates





- Relevant timing information contained in the input that causes the output to transition
  - Determine which input causes the transition, then pass its transition value to the output

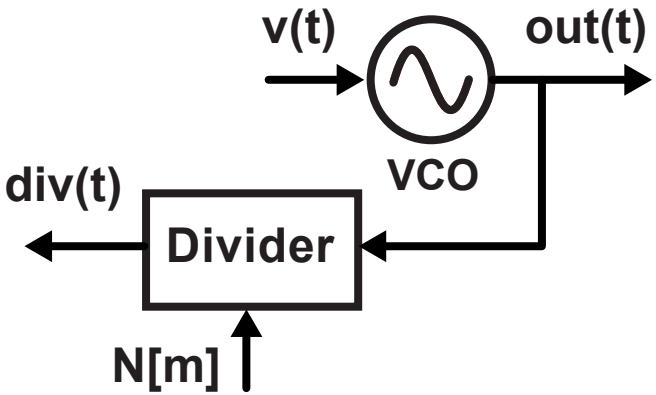
Issue: Must Observe Protocol When Adding Noise



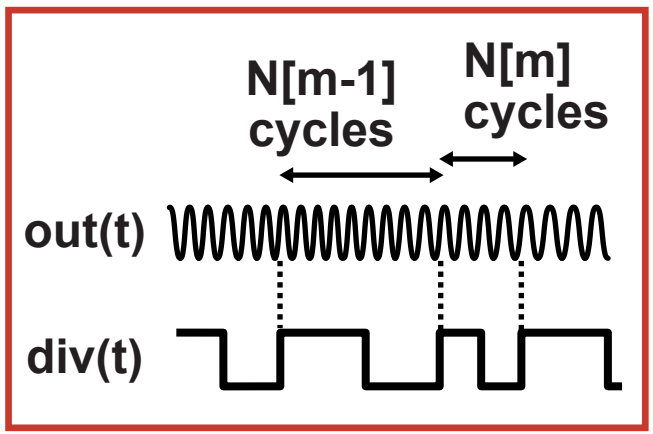
- Divider and PFD blocks operate on a strict protocol for their incoming signals
  - Values other than 1 or -1 are interpreted as edges
  - Example: inputting noise at divider input breaks protocol!
- Add noise only at places where signal is "analog"
  - PFD, charge pump, and loop filter outputs are fine

Can we speed the simulation up further?

Sample Rate Set by Highest Frequency Signal

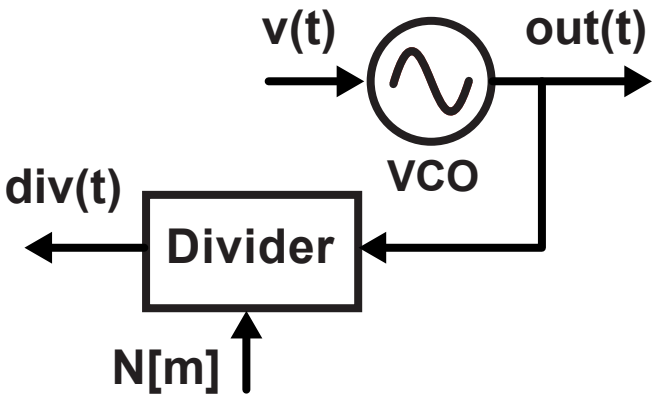


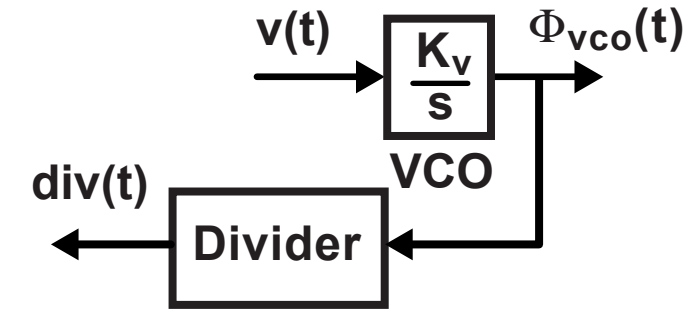
- Time step of simulation typically set by VCO output
- Small time steps means long simulation runs
- Divider output often 100 times lower in frequency

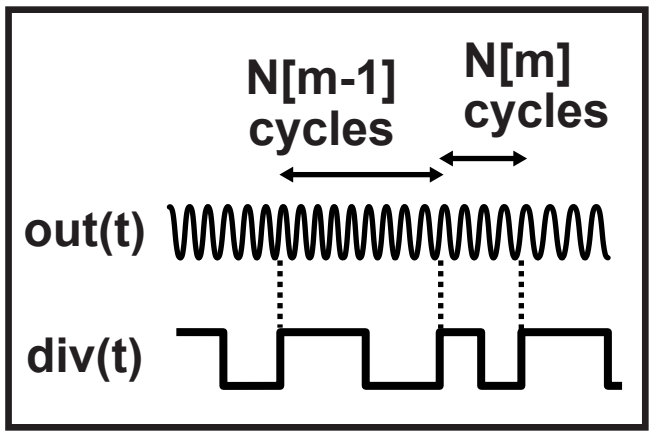


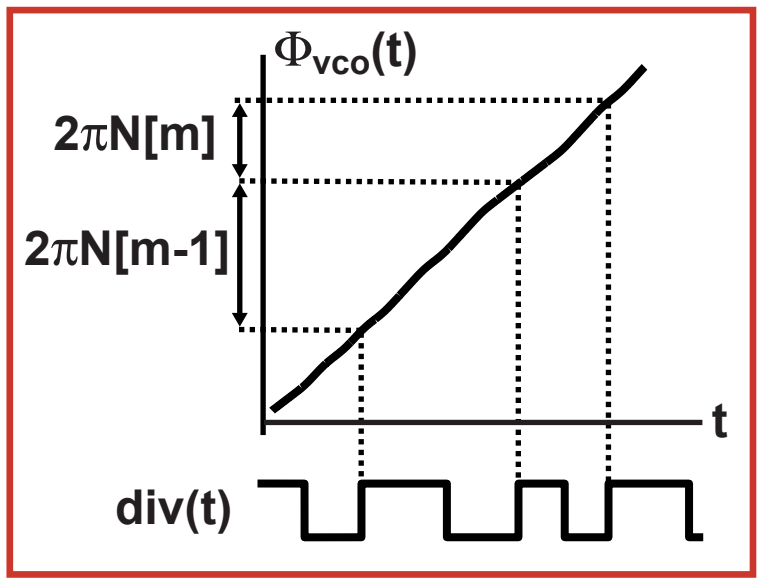
Can we sample according to divider output?

Divider Output Can Be Computed from VCO Phase







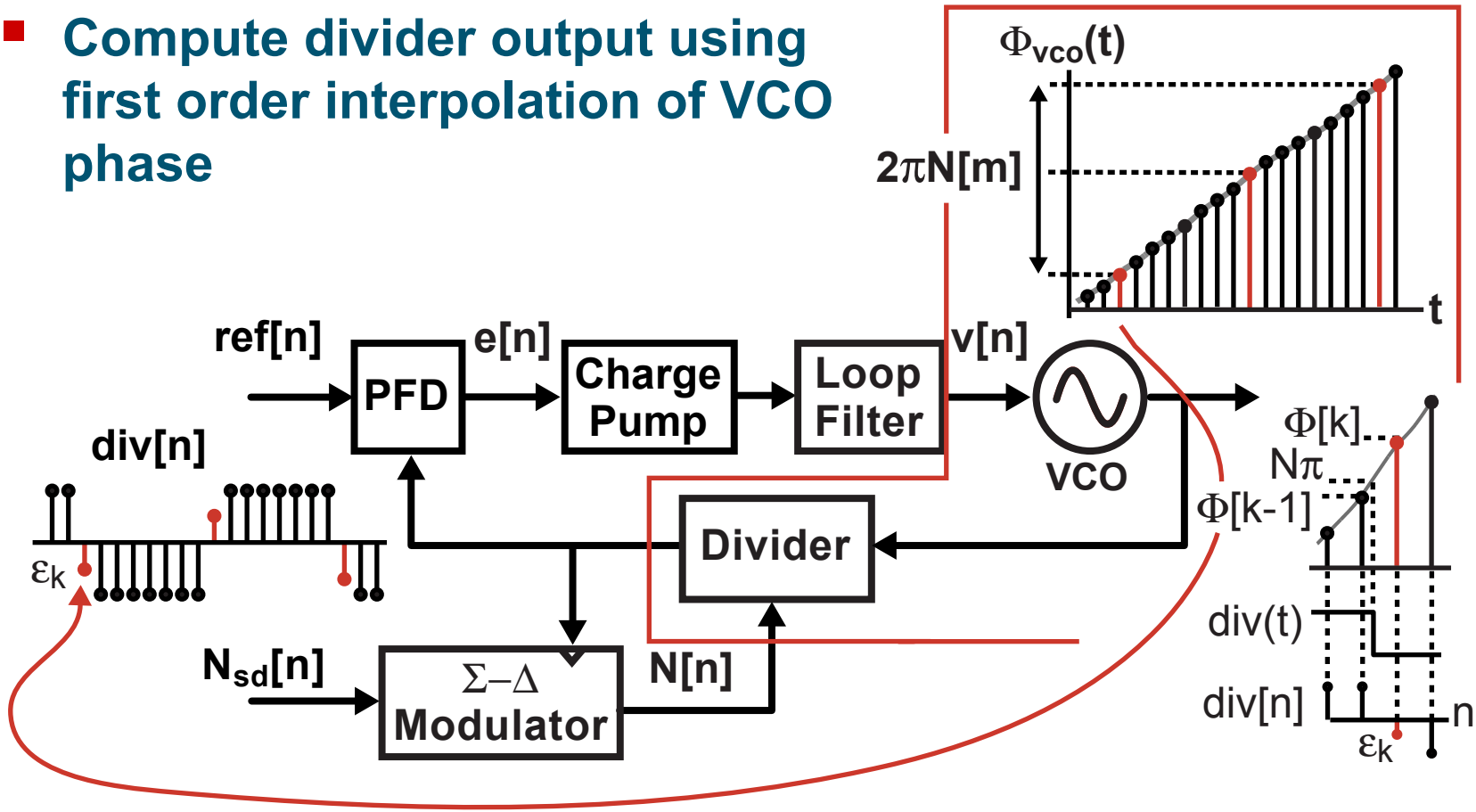


(Van Halen et al, Circuits and Systems '96)

Key Idea: Model VCO and Divider using Phase

Combine VCO and Divider Blocks

- Compute divider output using first order interpolation of VCO phase



Transient simulations run 2 orders of magnitude faster!

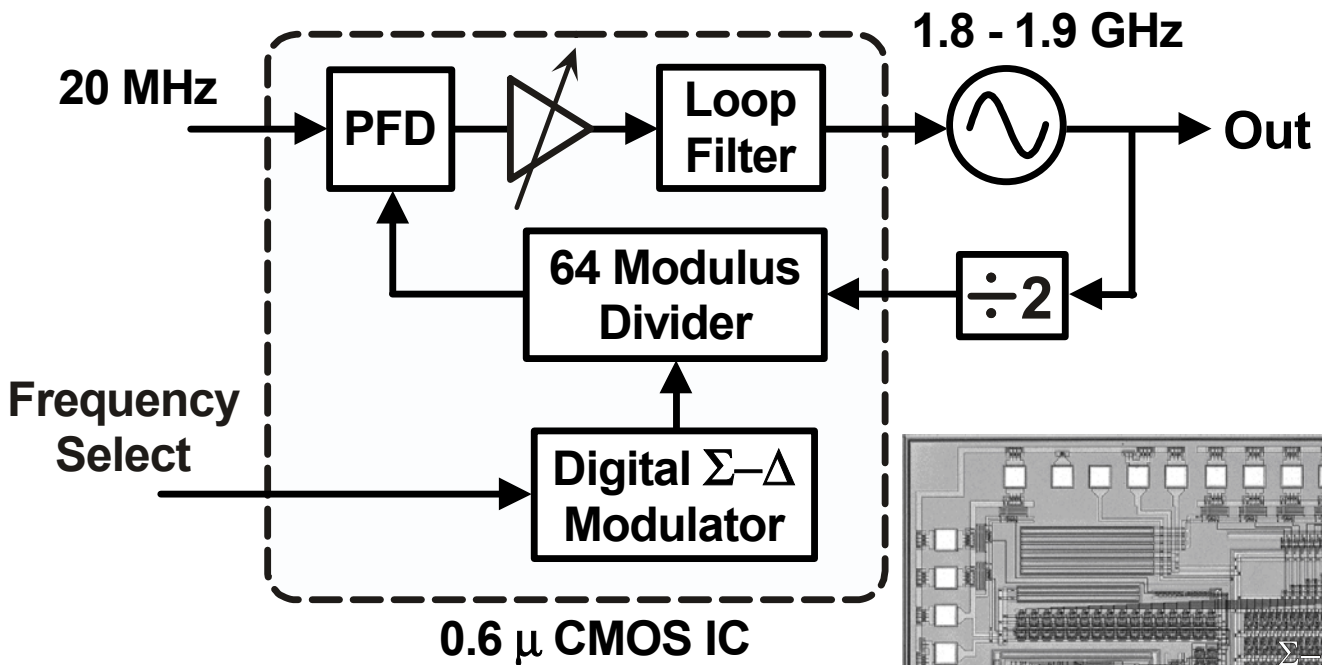
Does it really work?

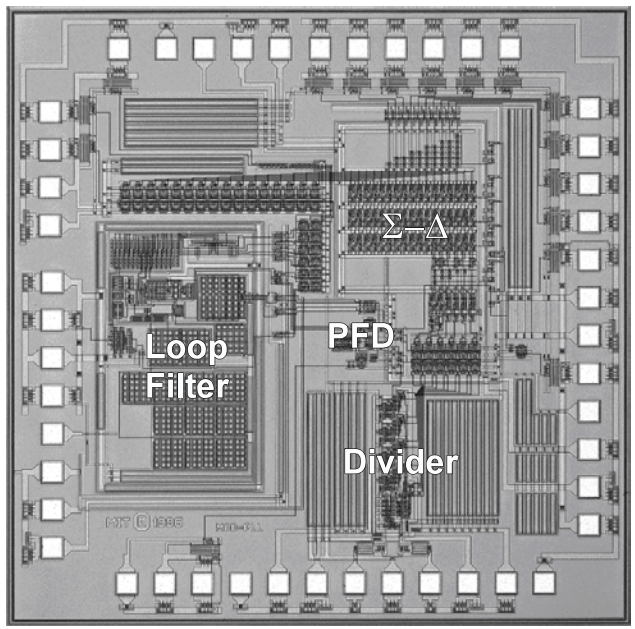
The CppSim Simulator

- Blocks are implemented with C/C++ code
  - High computation speed
  - Complex block descriptions
- Users enter designs in graphical form using Cadence schematic capture
  - System analysis and transistor level analysis in the same CAD framework
- Resulting signals are viewed in Matlab
  - Powerful post-processing and viewing capability

Simulation package available on Athena and freely downloadable at http://www-mtl.mit.edu/~perrott

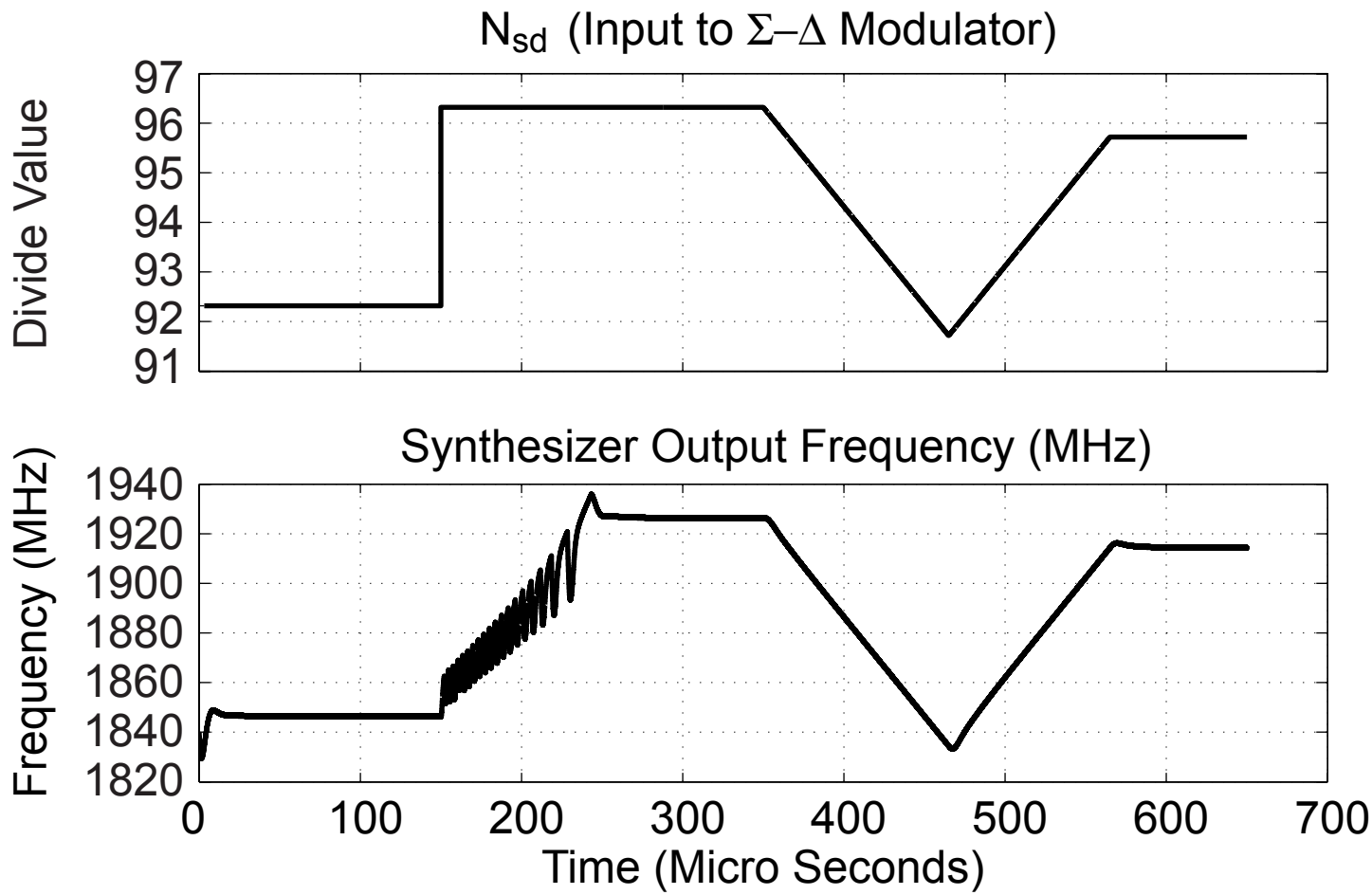
Experimental Prototype to Verify Approach





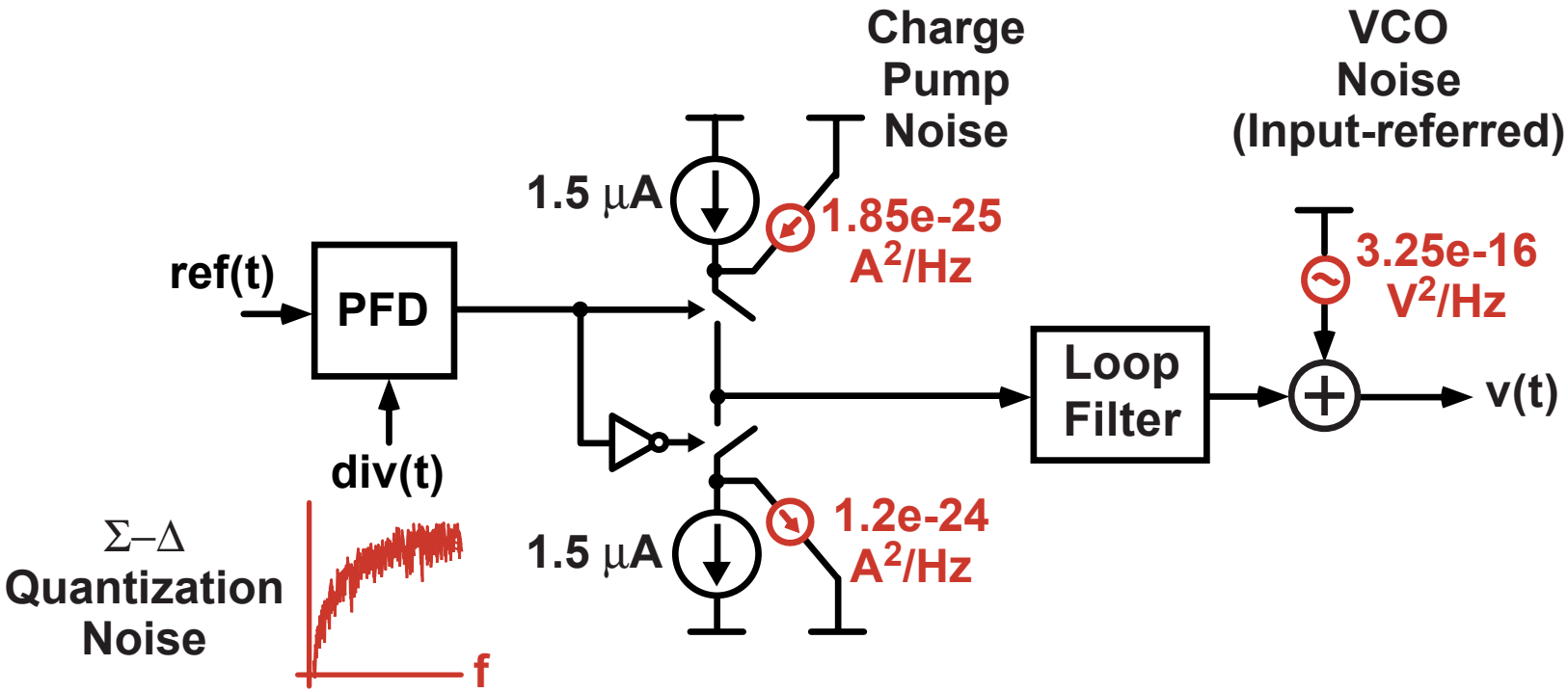
Perrott et al
JSSC, Dec 97

Simulation Results - Dynamic Behavior



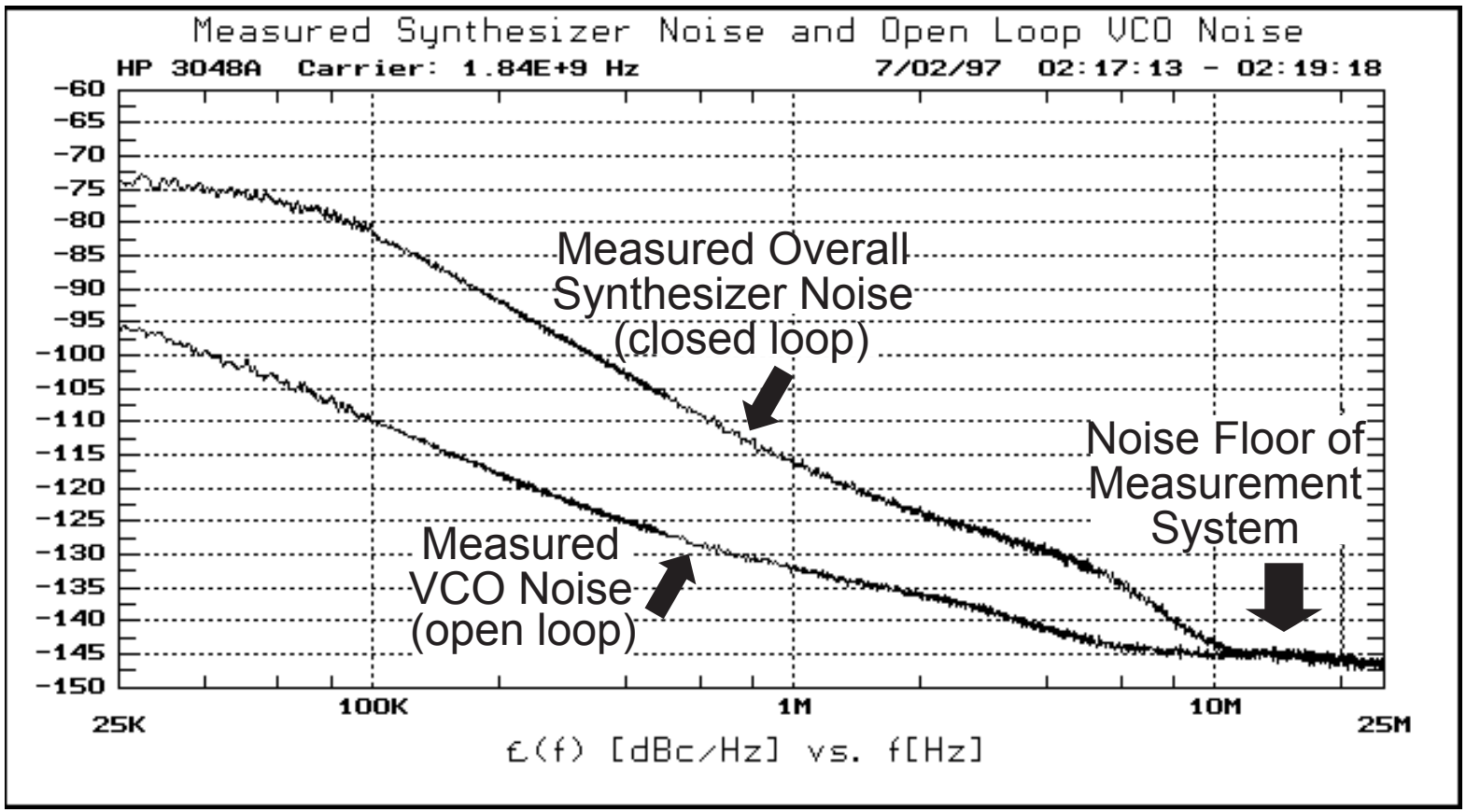
- **Simulation time: 260 thousand time steps in 5 seconds on a 650 MHz Pentium III Laptop (custom C++ simulator)**

Noise Sources Included in Simulation

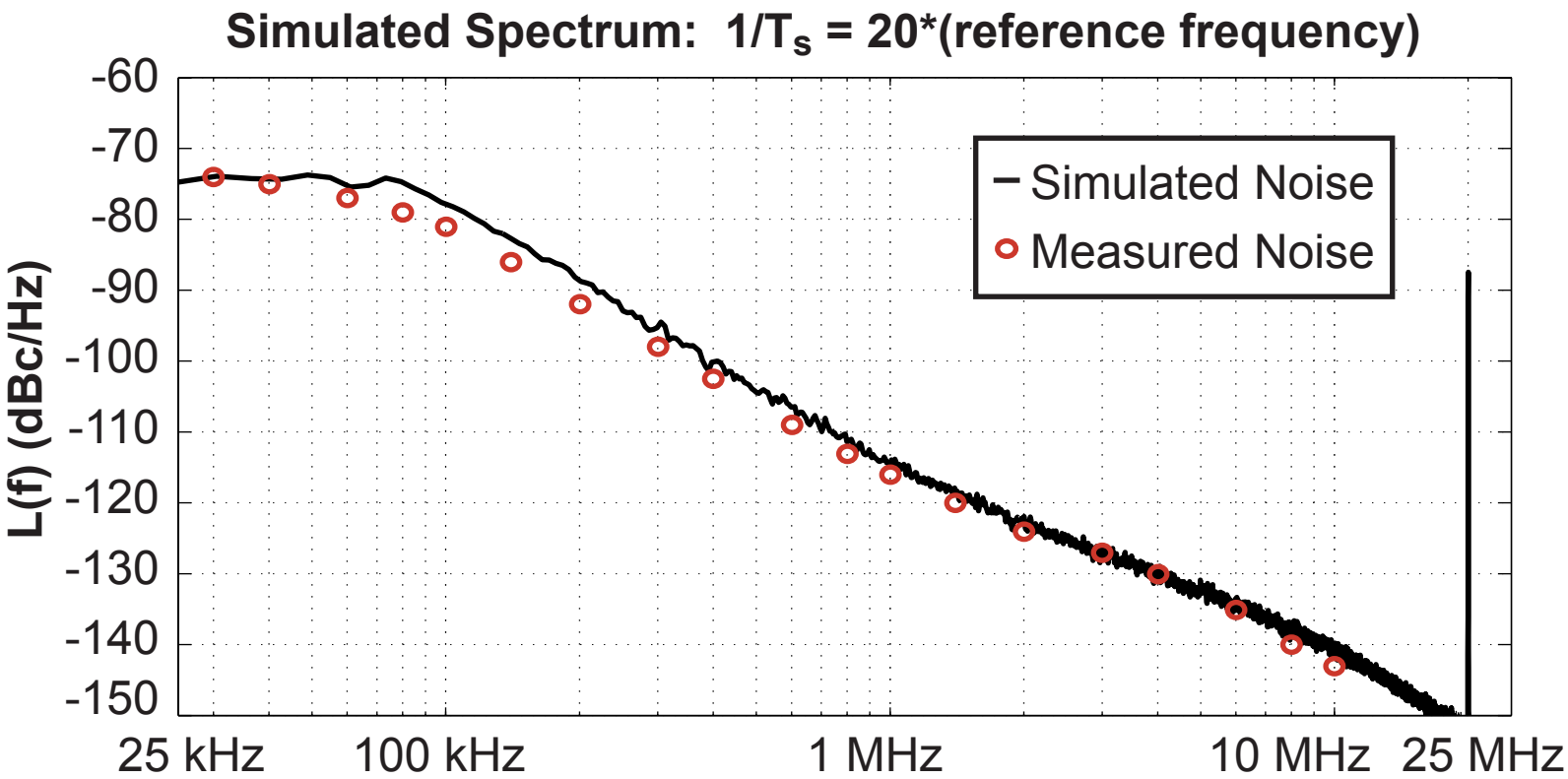


- Dominant noise sources in synthesizer
  - Quantization noise of Σ–Δ (produced by Σ–Δ block)
  - Charge pump noise (calculated from Hspice)
  - VCO noise (input-referred – calculated from measurement)

Measured Synthesizer Noise Performance



Simulated Synthesizer Noise Performance



- Simulated results compare quite well to measured!
- Simulation time: 5 million time steps in 80 seconds

Conclusion

- **Phase locked loop circuits can be quickly and accurately simulated**
  - **Accuracy achieved with area conservation principle**
  - **Fast computation by combining VCO and Divider blocks**

- **A variety of simulation frameworks can be used**
  - **C++, Matlab, Verilog**
  - **Circuit primitives are supported**

Noise and dynamic performance of fractional-N frequency synthesizers can be investigated at system level