

Accurate and Efficient Modeling Approach for Phase- Locked Loops for Mixed-Signal Design and Verification

Dr. Amr Fahim

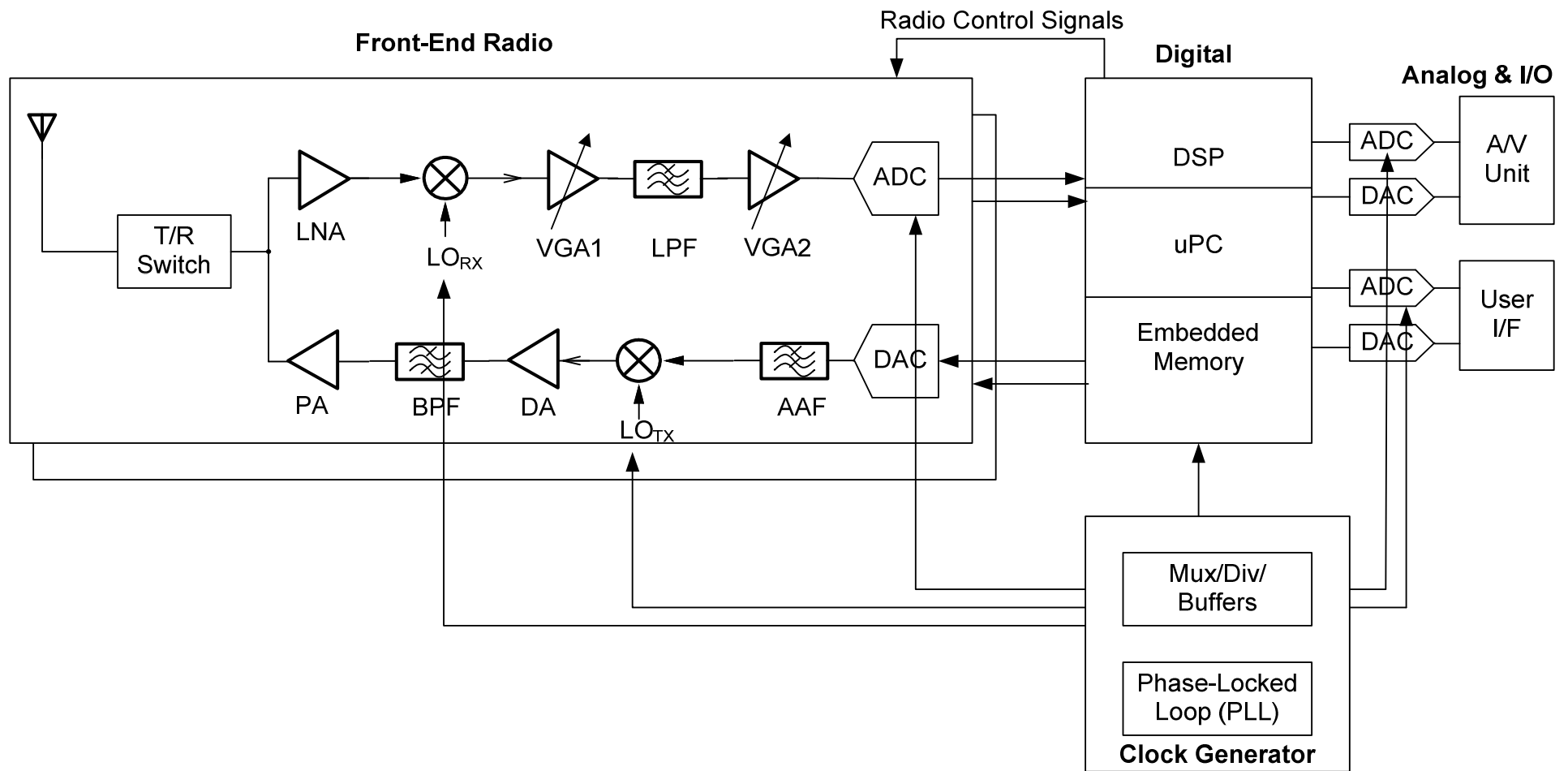
Semtech Corporation

Outline

- ❑ Target Mixed-Signal Designs
- ❑ Phase-Locked Loops in SoC Processors
- ❑ Conventional Modeling Approaches for PLLs
- ❑ Accurate and Time-Efficient Modeling Approach for PLLs

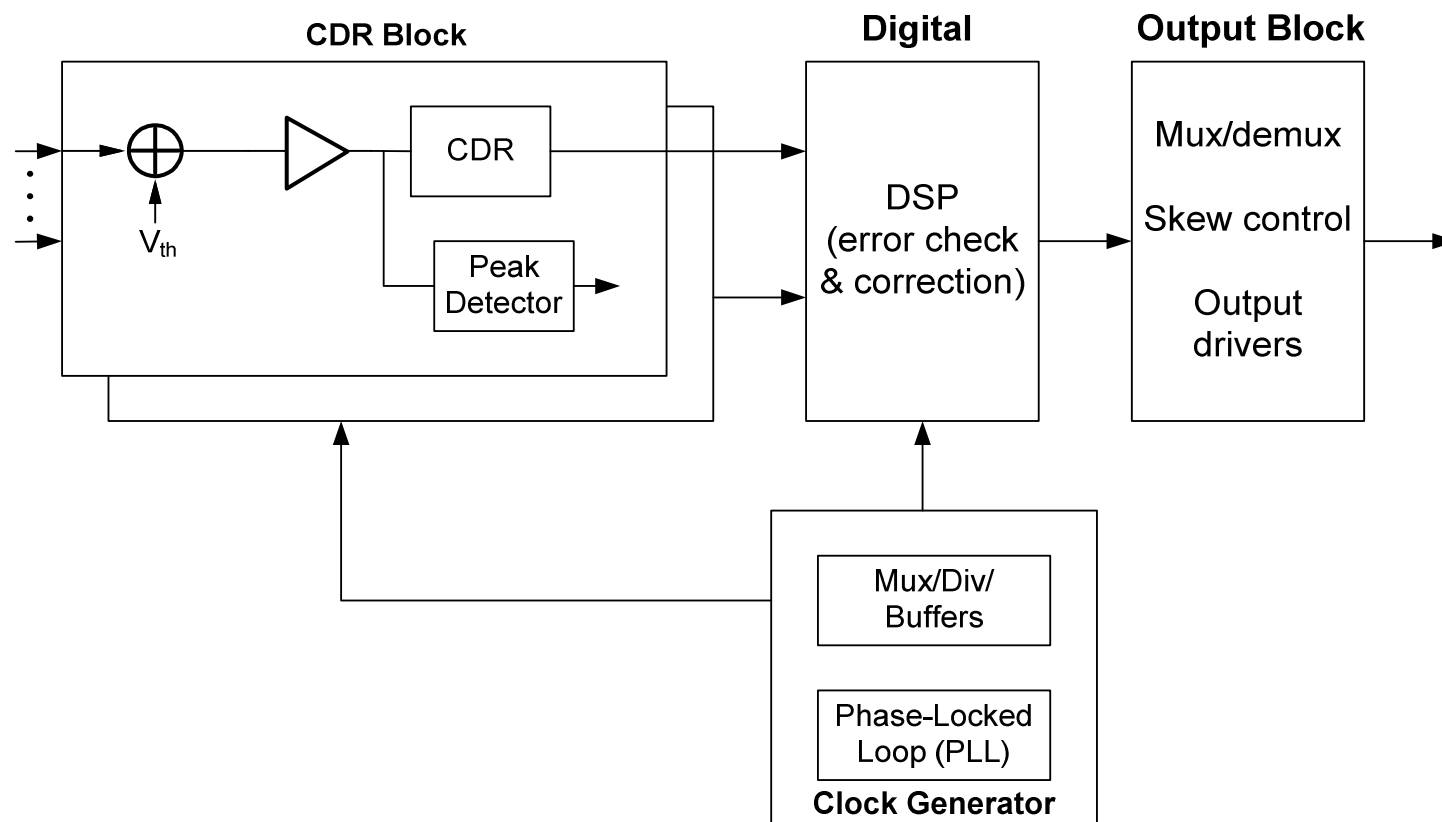
Target Mixed-Signal Designs

□ Typical RF SoC Processor:



Target Mixed-Signal Designs

- Typical Serdes SoC Processor:
 - Based on: SMI10031 (4:10 CDR Demux)

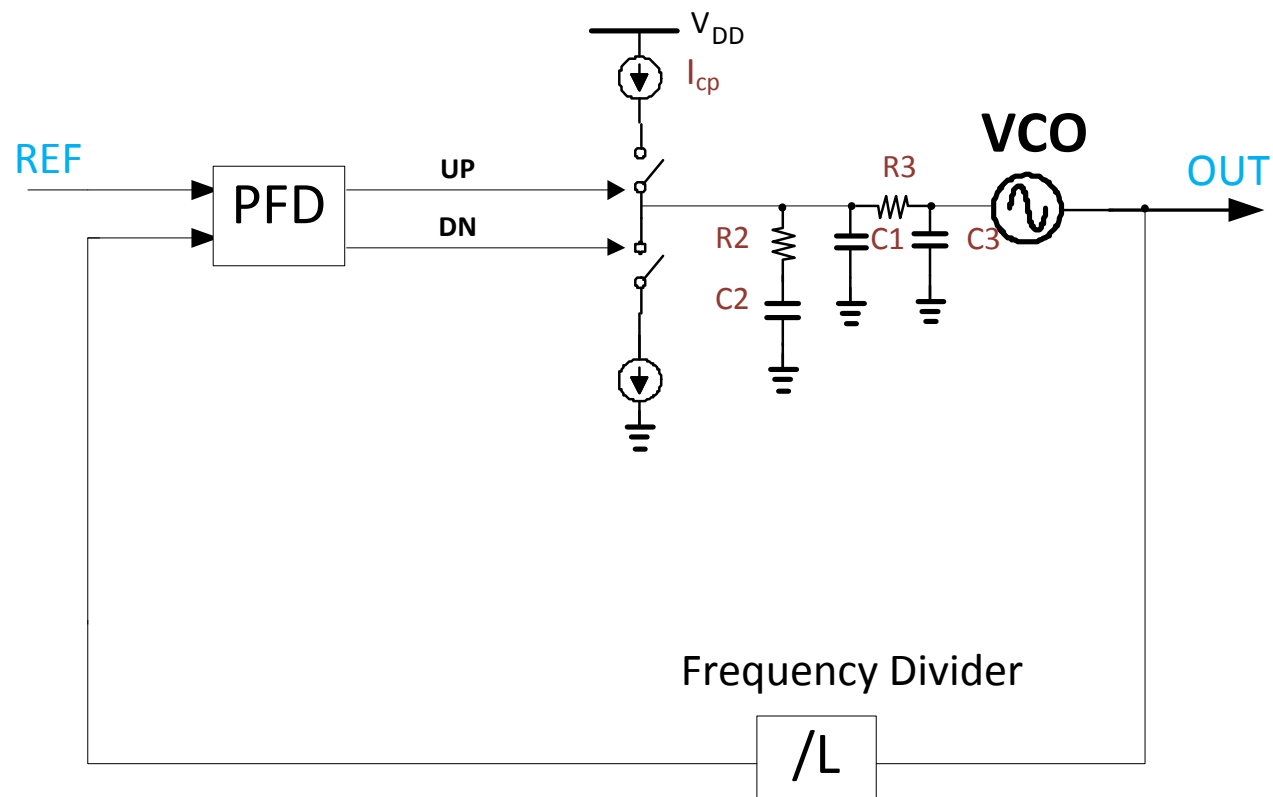


Outline

- ❑ Target Mixed-Signal Designs
- ❑ Phase-locked Loops in SoC Processors
- ❑ Conventional Modeling Approaches
- ❑ Accurate and Time-Efficient Modeling Approach

Phase-Locked Loops in SoC Processors

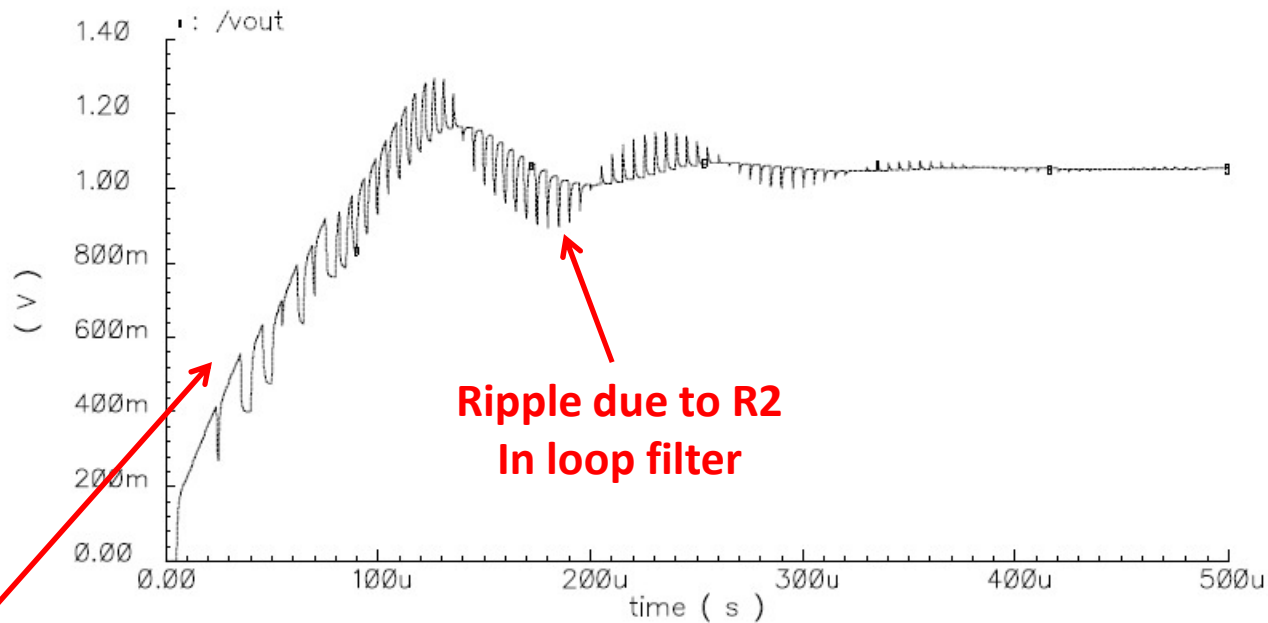
- Clock Generator \equiv charge pump-based PLL



Note: output frequency is integer multiple (L) of the REF frequency

Phase-Locked Loops in SoC Processors

- Typical locking behavior of a charge-pump PLL:
 - Loop filter voltage
 - Note ripples during the locking procedure



Limited PFD capture range

**Ripple due to R2
In loop filter**

Phase-Locked Loops in SoC Processors

□ Linear model of a charge-pump PLL

- Closed loop response:

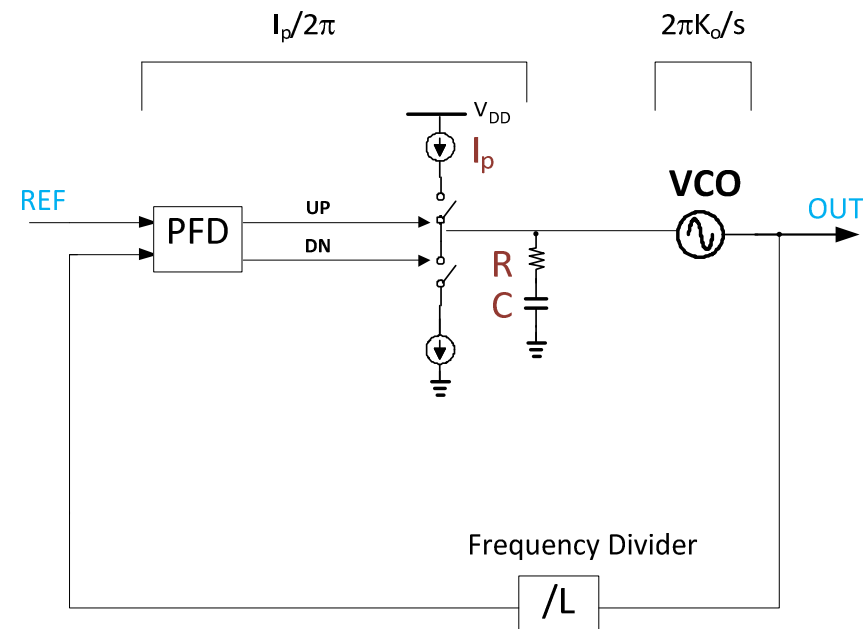
$$H_1(s) = \frac{I_p K_o (R + \frac{1}{sC}) / 2\pi}{s + \frac{I_p}{2\pi} K_o (R + \frac{1}{sC}) / L}$$

- Damping factor:

$$\zeta = \frac{\omega_n RC}{2}$$

- Natural frequency:

$$\omega_n = \sqrt{\frac{K_o I_p}{2\pi LC}}$$



Phase-Locked Loops in SoC Processors

□ Charge pump design considerations:

➤ Operation:

- Current steering charge pump
- I_{UP} & I_{DN} are always ON

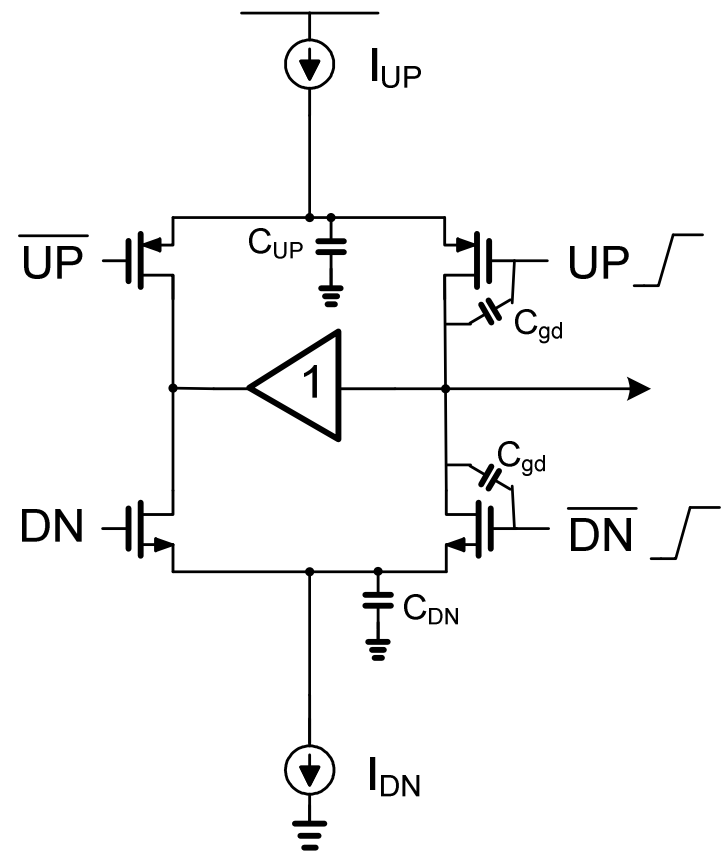
➤ Noise sources:

- I_{UP} , I_{DN} → Thermal/flicker noise
- Switches → flicker noise

➤ Glitches due to switching:

- CLK feedthrough
- Charge time of C_{UP} and C_{DN}
 - Minimized by unity gain buffer
- Effect:
 - ref spur (integer PLL)
 - nonlinearity ($\Sigma\Delta$ PLL)

➤ **Basic trade-off between noise & glitches in charge pump design**

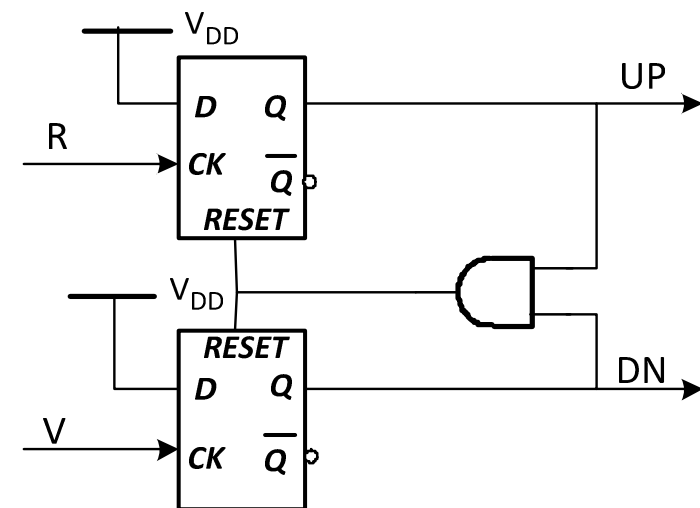


Phase-Locked Loops in SoC Processors

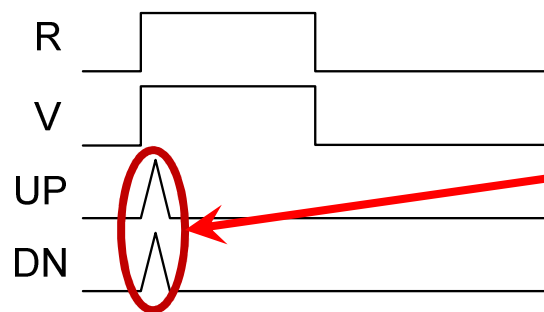
❑ Deadzone issue in PFD+CP

- Mention that if pulses are too small, can generate a large difference between UP and DN currents in the charge pump

State	State Condition	Operation
-1	DN=1,UP=0	VCO freq too high
0	DN=0,UP=0	PLL is in phase lock
1	DN=0,UP=1	VCO freq too low



Locked state:



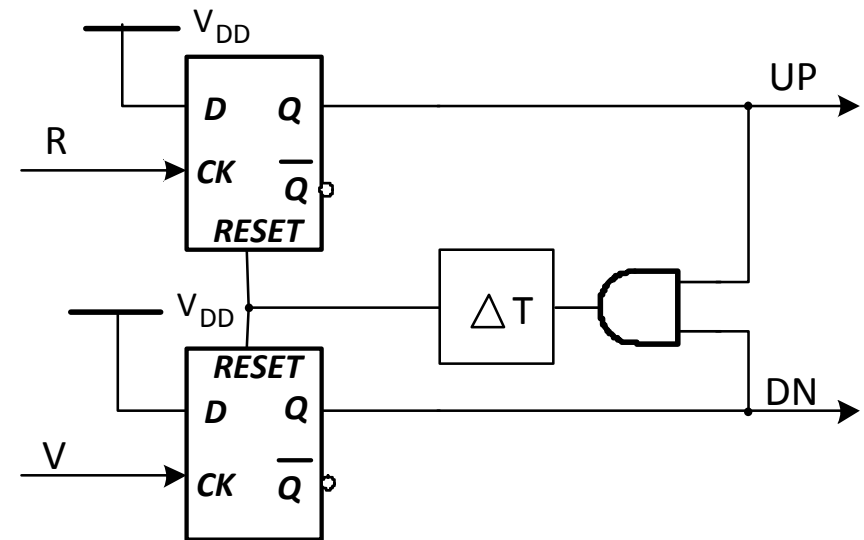
Pulses too narrow!! Can cause:

- 1. Significant reference spurs (integer PLL)**
- 2. Nonlinearity ($\Sigma\Delta$ PLL)**
- 3. Change in loop dynamics**

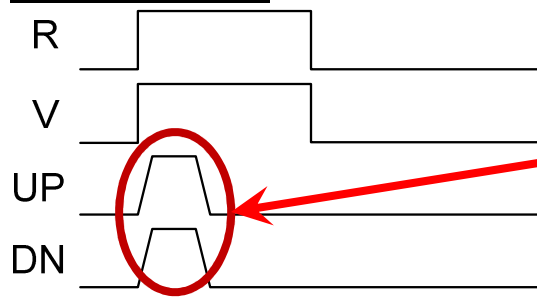
Phase-Locked Loops in SoC Processors

- ❑ Deadzone mitigation in PFD
 - Race condition in reset path
 - Add a 4th state:

State	State Condition	Operation
-1	DN=1,UP=0	VCO freq too high
0	DN=0,UP=0	PLL is in phase lock
1	DN=0,UP=1	VCO freq too low
Z	DN=1,UP=1	PLL frequency held



Locked state:



Delay in reset path widens UP & DN pulses:

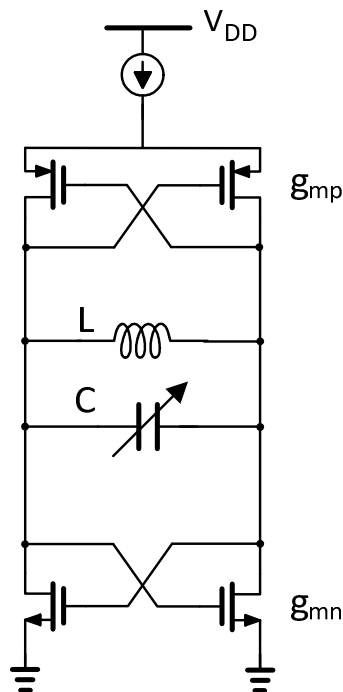
1. Less reference spurs (integer PLL)
2. More linear charge pump ($\Sigma\Delta$ PLL)
3. More stable loop dynamics

Phase-Locked Loops in SoC Processors

□ VCO design considerations:

➤ Typical VCO is an LC VCO with cross-coupled inverters

- Cross-coupled FETs to provide $-R$ to cancel LC parasitic resistance
- Capacitor value range determines frequency range of VCO



Optimization Eqns:

1.
$$-R = \frac{2}{g_{mp} + g_{mn}}$$
2.
$$g_m = \sqrt{\mu C_{ox} \frac{W}{L} I_D}$$
3.
$$R_p = Q^2 R_s$$
4.
$$f_{vco} = \frac{1}{2\pi\sqrt{LC}}$$

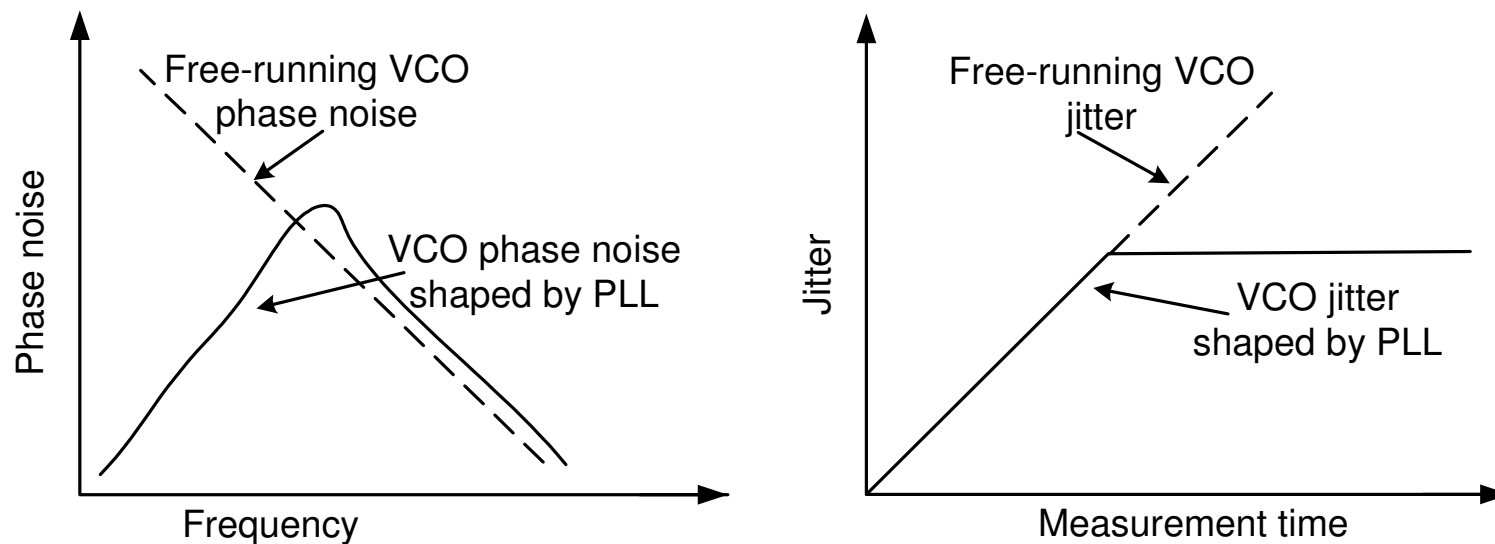
Design Iteration Steps:

1. Determine tuning range (initial L, C values)
2. Determine required $K_{vco} \rightarrow$ tuning bits is fixed
3. Increase I_D until increase in swing diminishes
4. Extract R,L,C parasitics
5. Readjust g_m , C until phase noise – frequency range trade-off is reached (steps 1-4).

Phase-Locked Loops in SoC Processors

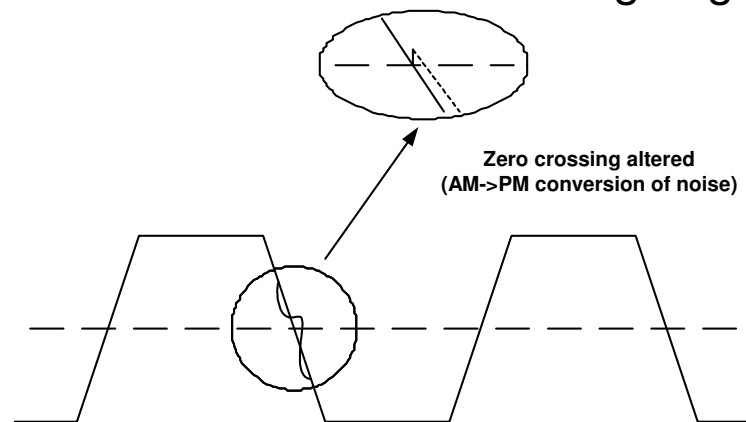
- Phase noise accumulation in VCOs:
 - Noise transfer function of VCO is a HPF:

$$H_{VCO}(s) = \frac{s}{s + K_d K_o F(s) / L}$$

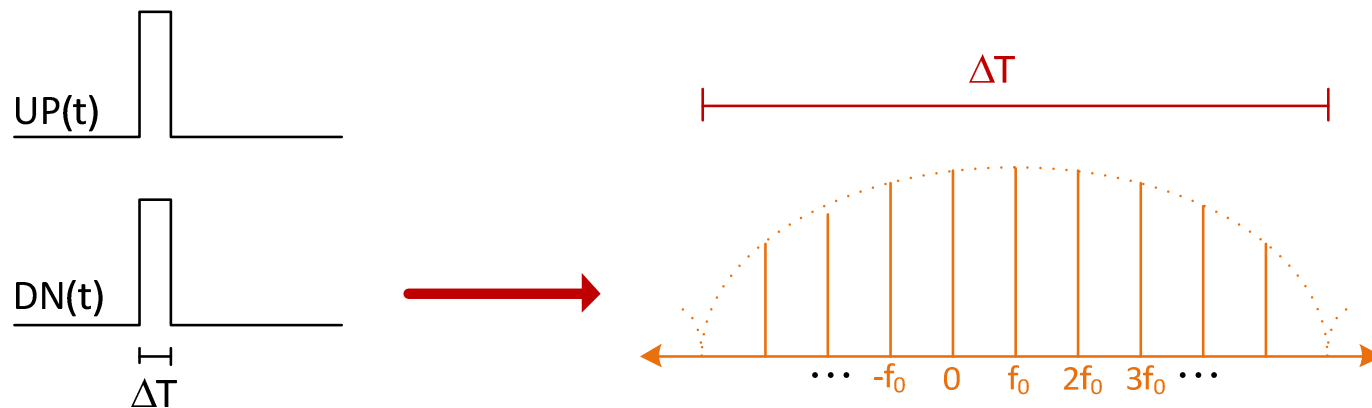


Phase-Locked Loops in SoC Processors

- Device noise in digital circuits: (digital dividers & PFD)
 - AM→PM conversion of noise occurs during edge transitions

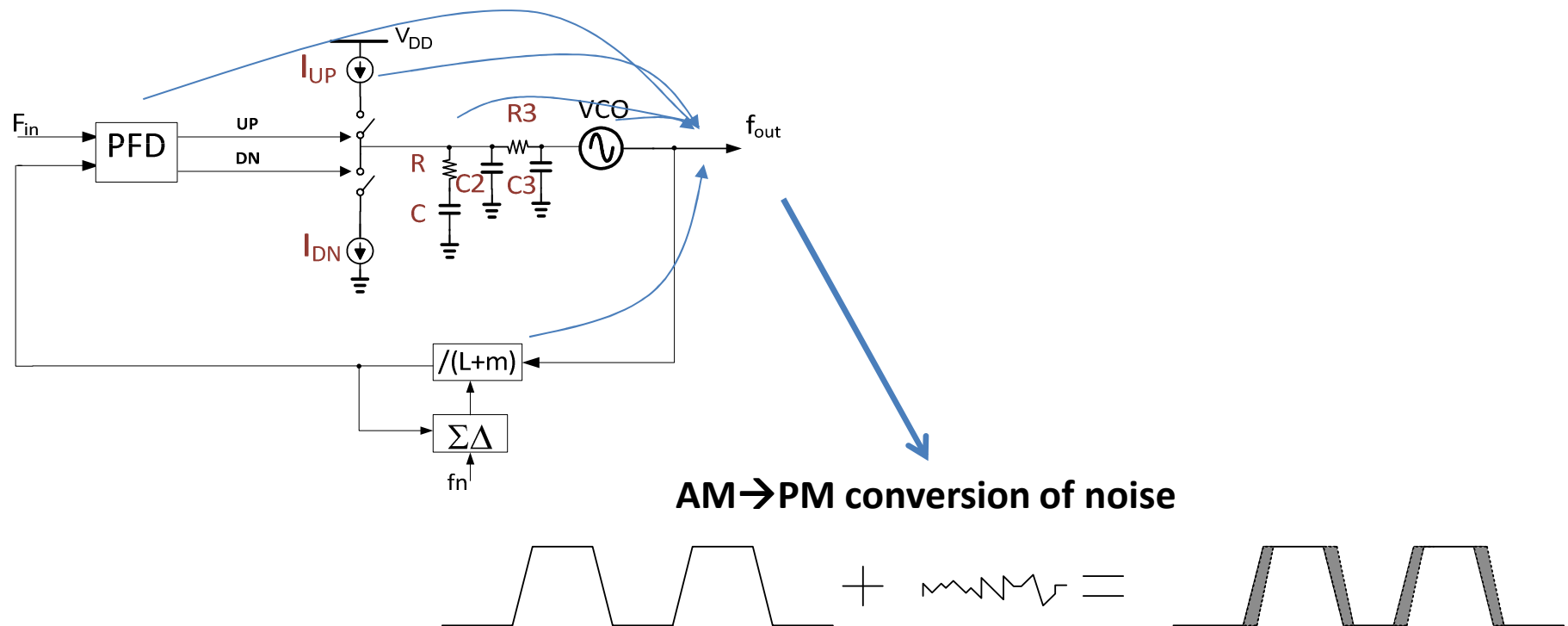


- Digital circuits in the PLL have periodic outputs (digital dividers & PFD):



Phase-Locked Loops in SoC Processors

- ❑ PLL intrinsic noise is sum of noise sources of its components
- ❑ Jitter is the phase variation resulting from amplitude noise (AM→PM conversion of noise).



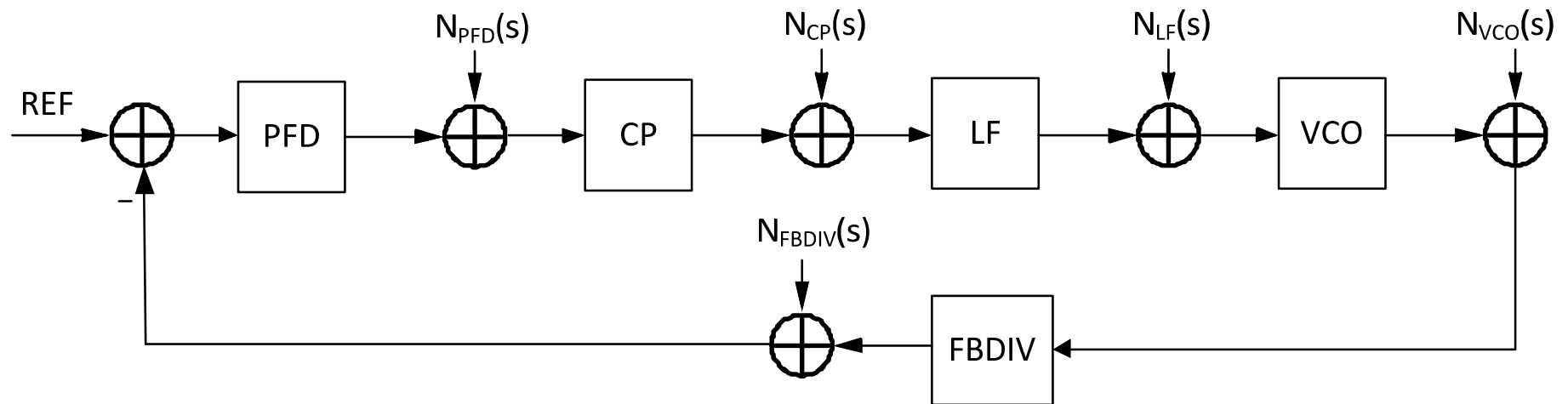
Phase-Locked Loops in SoC Processors

- Summary of PLL noise transfer functions (NTFs):

Block	NTF	TYPE
PFD	$H_{PFD}(s) = \frac{K_d K_o F(s)}{s + K_d K_o F(s)/L}$	LPF
CP	$H_{CP}(s) = \frac{K_o F(s)}{s + K_d K_o F(s)/L}$	LPF
LF	$H_{LF}(s) = \frac{K_o}{s + K_d K_o F(s)/L}$	BPF
VCO	$H_{LF}(s) = \frac{s}{s + K_d K_o F(s)/L}$	HPF
FDBK DIV	$H_{FBDIV}(s) = \frac{K_d K_o F(s)}{s + K_d K_o F(s)/L}$	LPF

Phase-Locked Loops in SoC Processors

- Linear model for noise analysis:
 - $N(f)$ noise sources obtained from PSS/PNOISE transient simulations



$$N(f) = N_{PFD}(f) \cdot \|H_{PFD}(f)\|^2 + N_{CP}(f) \cdot \|H_{CP}(f)\|^2 + N_{LF}(f) \cdot \|H_{LF}(f)\|^2 + N_{VCO}(f) \cdot \|H_{VCO}(f)\|^2 + N_{FBDIV}(f) \cdot \|H_{FBDIV}(f)\|^2$$

Objective function:

minimize $N(f)$ given area constraints

→ convex optimization routine

Phase-Locked Loops in SoC Processors

- ❑ Issues with linear model:
 - Does not take into account any non-linearities
 - *Will see shortly!*
 - Does not take into account any transient effects
 - One example: rippling during locking behavior

Phase-Locked Loops in SoC Processors

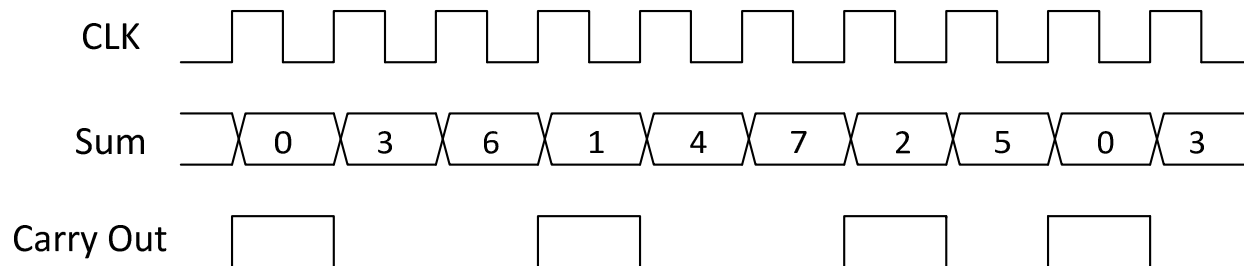
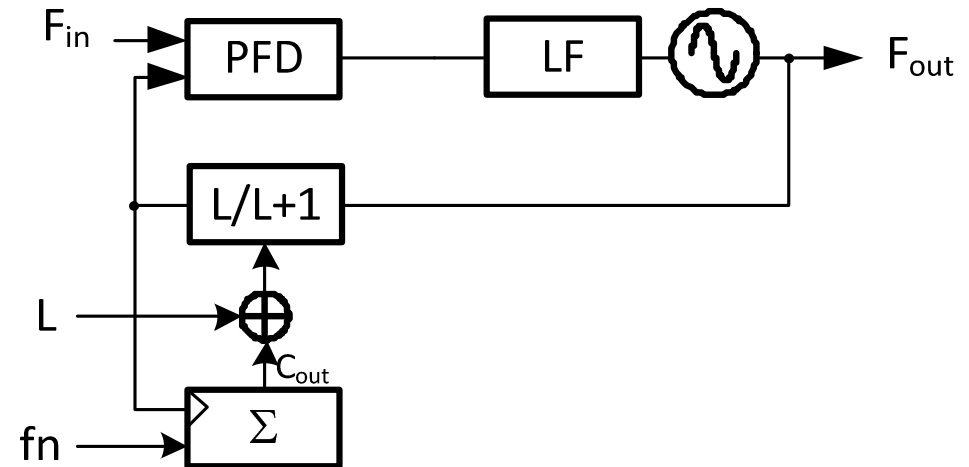
□ Fractional-N PLL Basics:

➤ Why fractional-N ?

- Obtain arbitrary output frequency – not restricted by the REF frequency

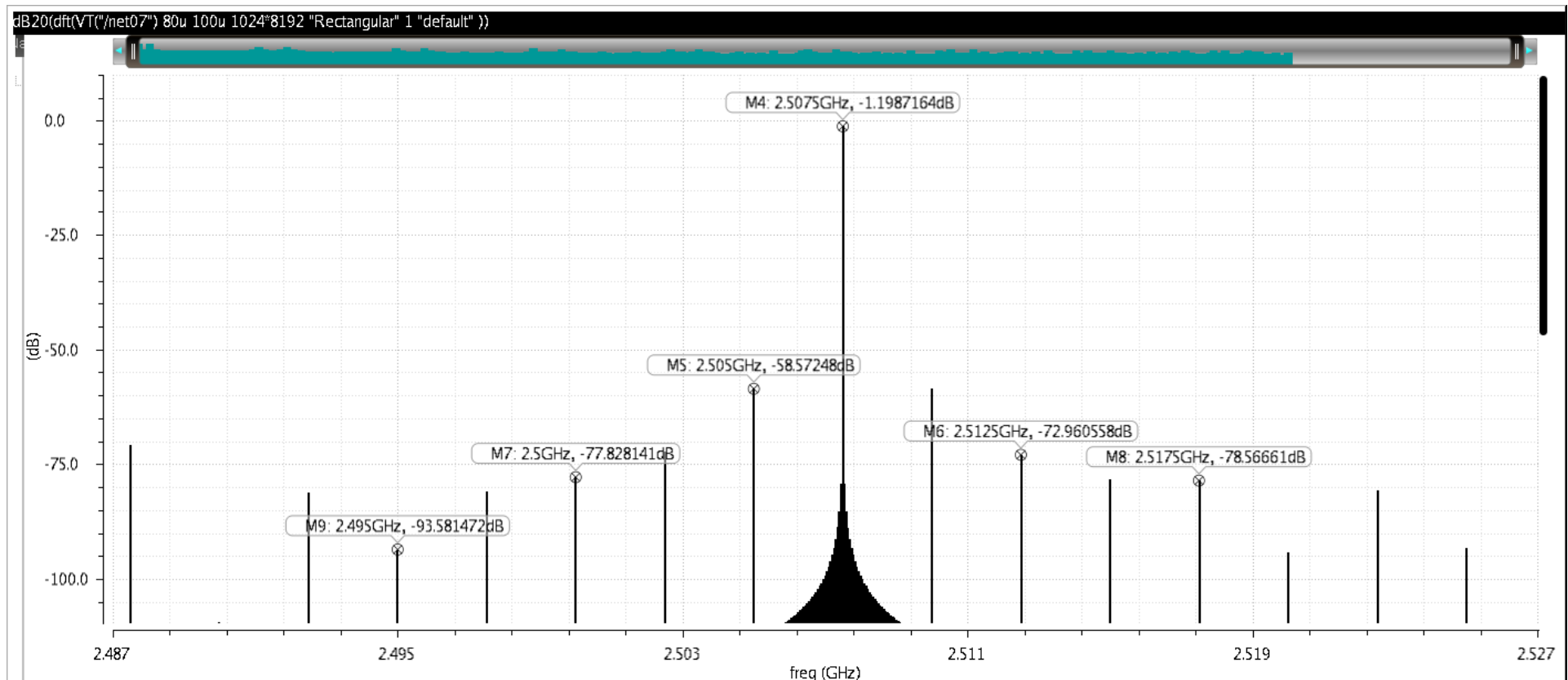
➤ Fractional-N operation:

- Example: $f_n=3/8$



Phase-Locked Loops in SoC Processors

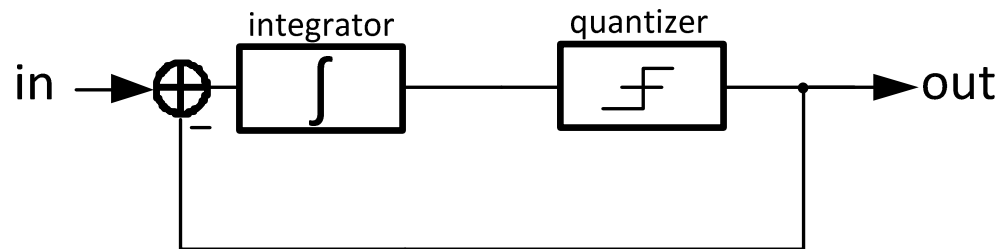
- Sample simulation of a fractional-N PLL:
 - $F_{out}=2.5\text{GHz}$, $F_{in}=20\text{MHz}$, fractional ratio= $3/8$ (3-bit accumulator)



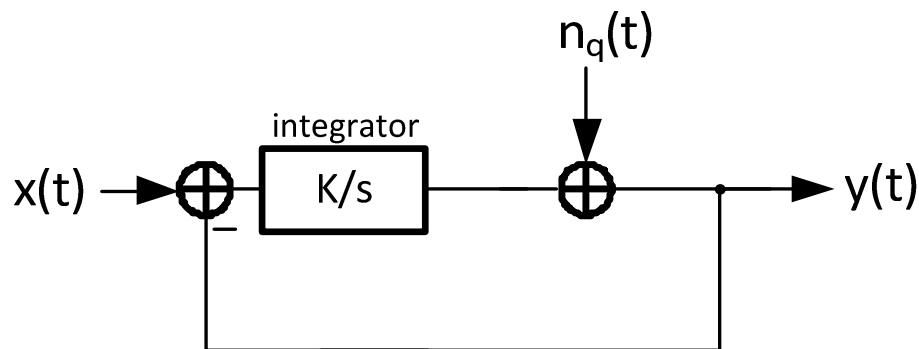
Phase-Locked Loops in SoC Processors

□ How to randomize the periodic jitter? → $\Sigma\Delta$ modulator

➤ Basic $\Sigma\Delta$ modulator:



➤ Linear model of $\Sigma\Delta$ modulator:



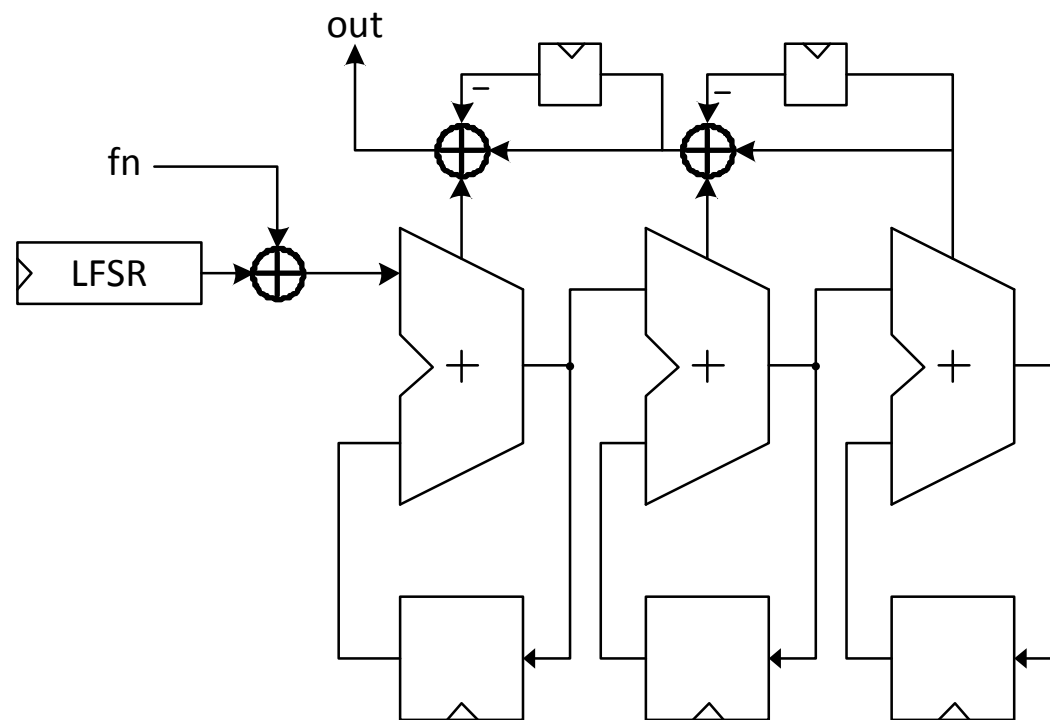
$$STF = \frac{K}{s + K}$$

$$NTF = \frac{s}{s + K}$$

Basic assumption: There is “sufficient signal activity” at the input of the integrator to make it appear random.

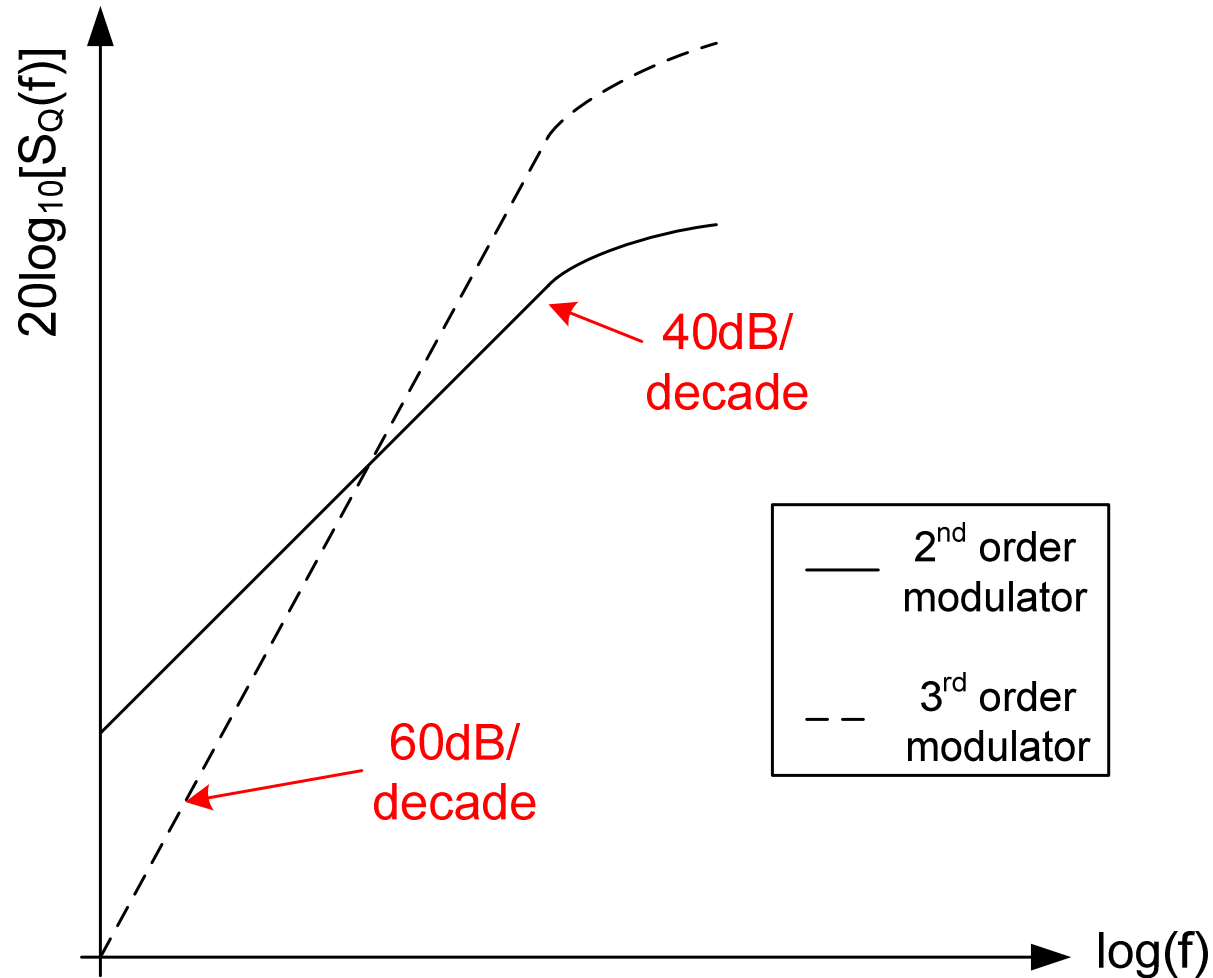
Phase-Locked Loops in SoC Processors

- A digital sigma-delta modulator for use in a PLL:
 - MASH1-1-1 sigma-delta modulator
 - LFSR included to add sufficient activity at the input
 - All flops are clocked by the output of the PLL feedback divider



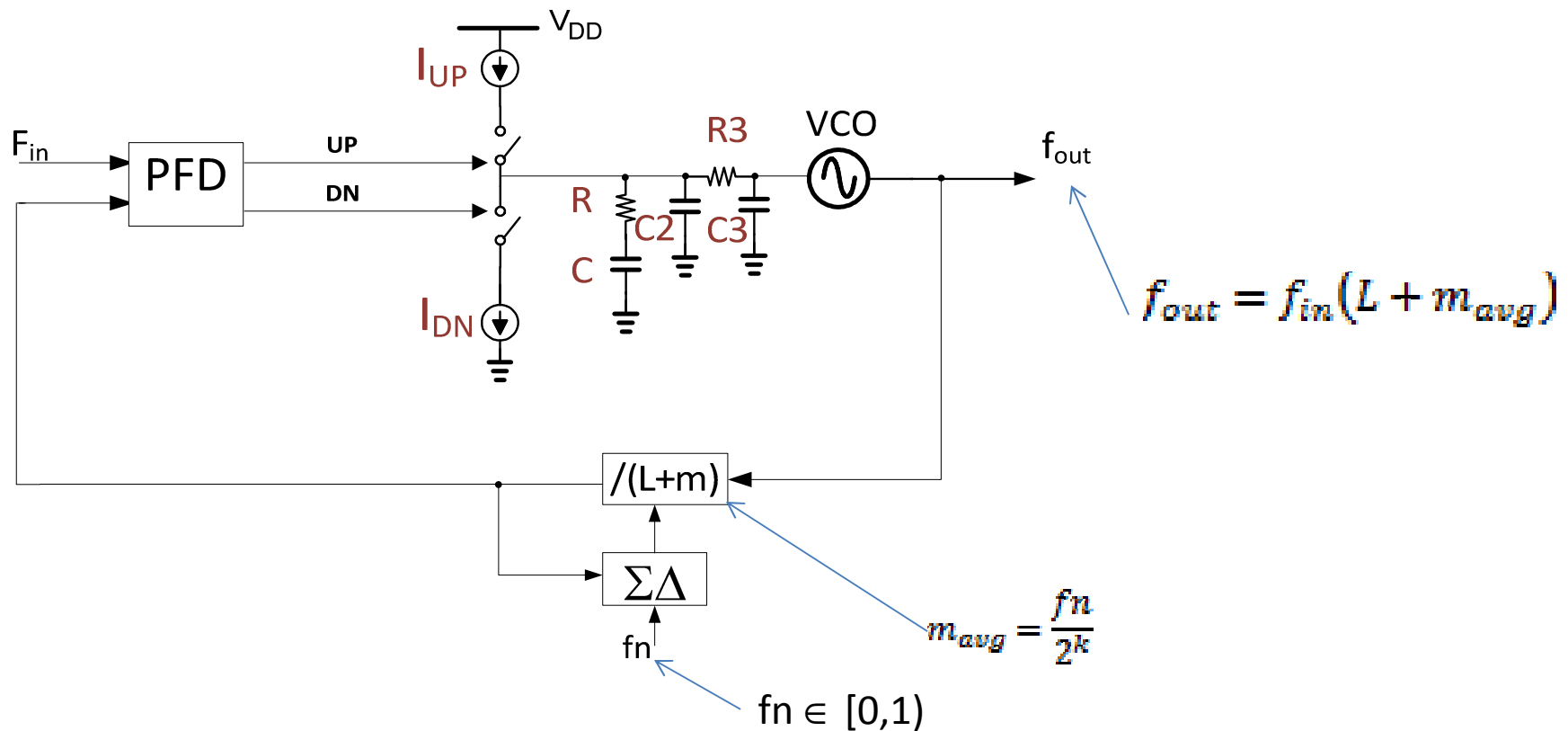
Phase-Locked Loops in SoC Processors

- Simulation of a sigma-delta modulator



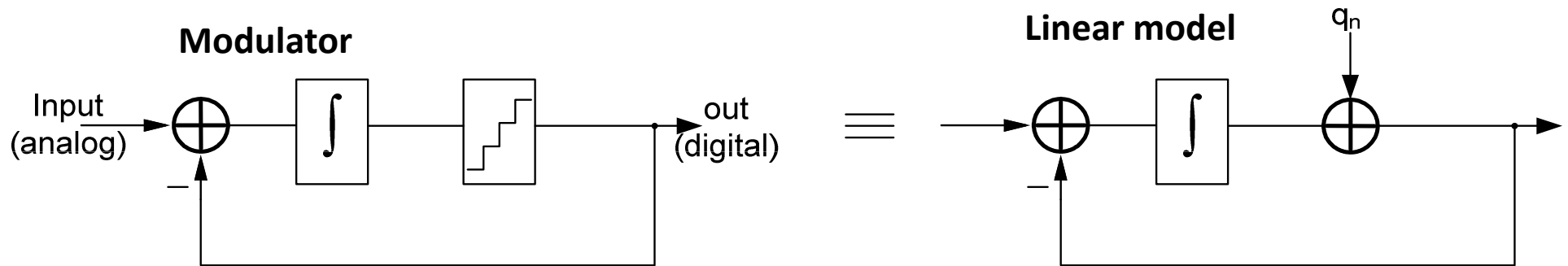
Phase-Locked Loops in SoC Processors

□ $\Sigma\Delta$ Fractional-N Phase-Locked Loop (PLL)



Phase-Locked Loops in SoC Processors

- SD modulation basics:



STF \rightarrow Low-pass filter
 NTF \rightarrow High pass filter

Quantization noise \rightarrow additive white Gaussian noise only if step size is uniform

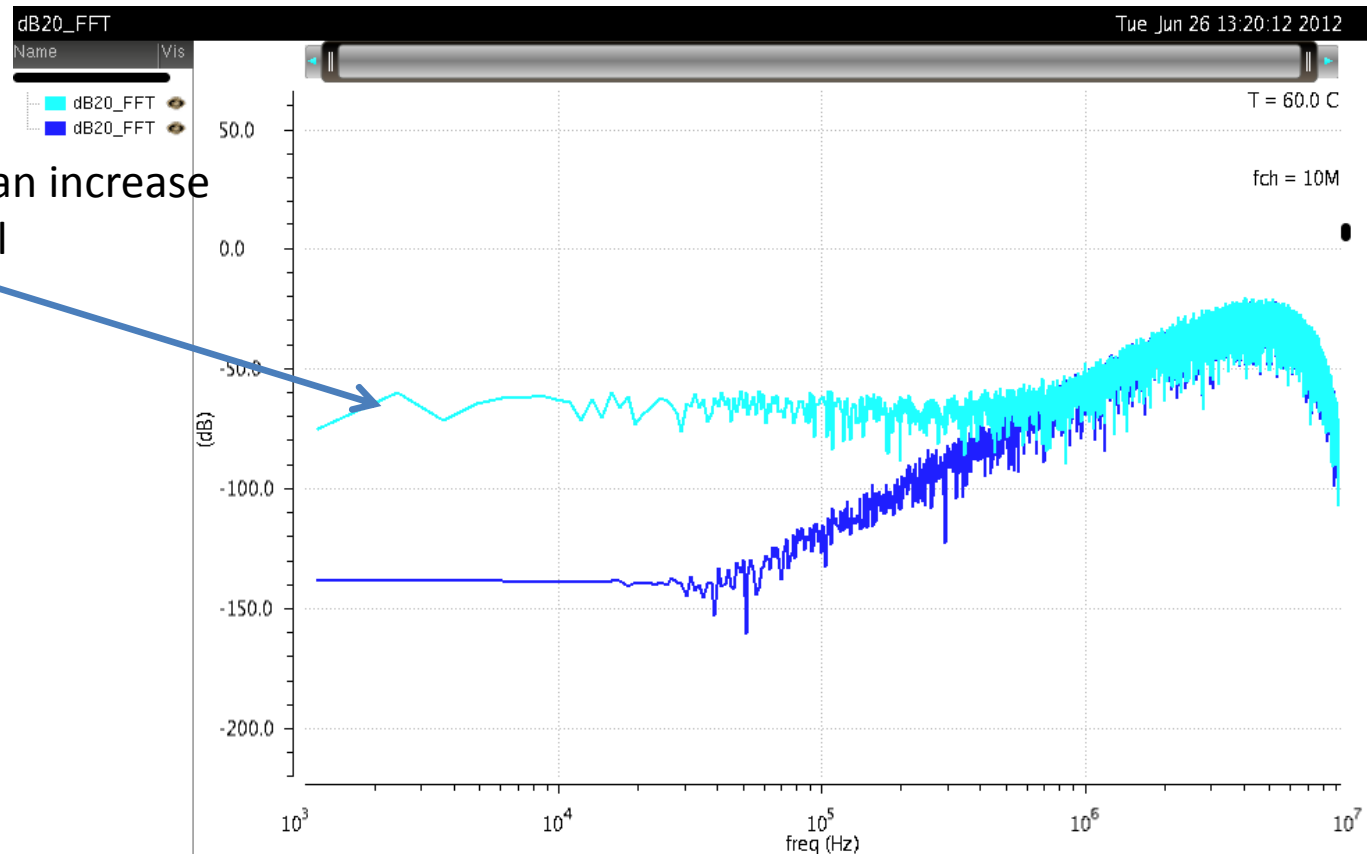
If step size is non-uniform, NTF shape breaks down.

Phase-Locked Loops in SoC Processors

□ Sigma-Delta Noise Folding:

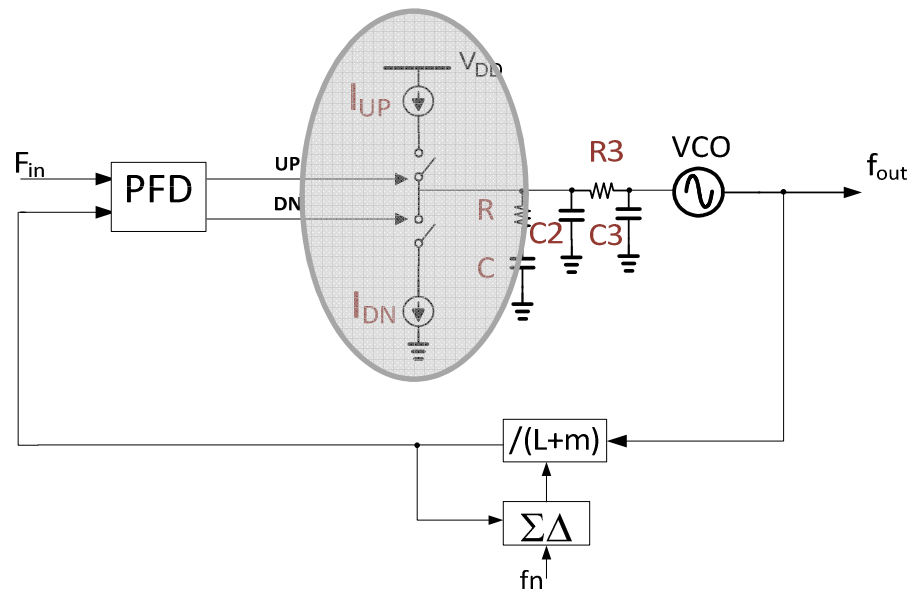
- Plot shows ideal (0%) and 2% mismatch
- Nonlinearity causes high frequency quantization noise to fold back in-band

Noise folding causes an increase in in-band noise level

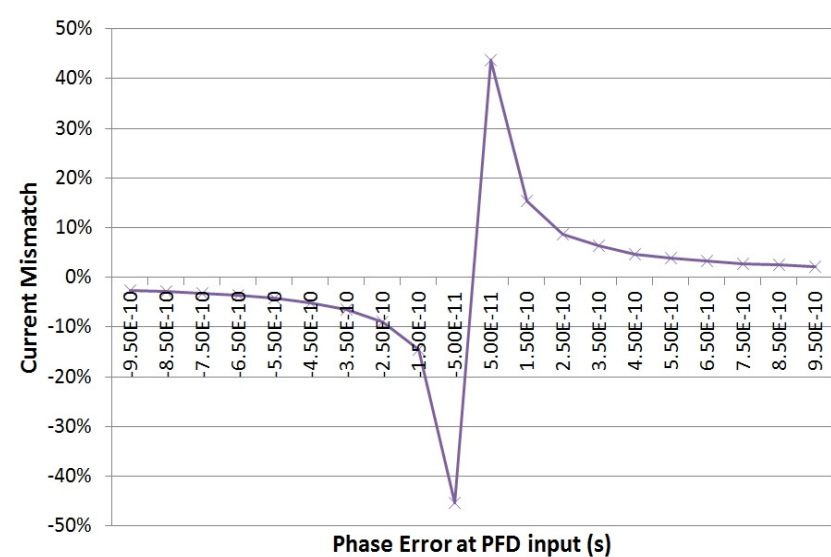


Phase-Locked Loops in SoC Processors

- ❑ Where does non-linear quantization noise steps arise in $\Sigma\Delta$ PLLs?
 - Charge pump current mismatch

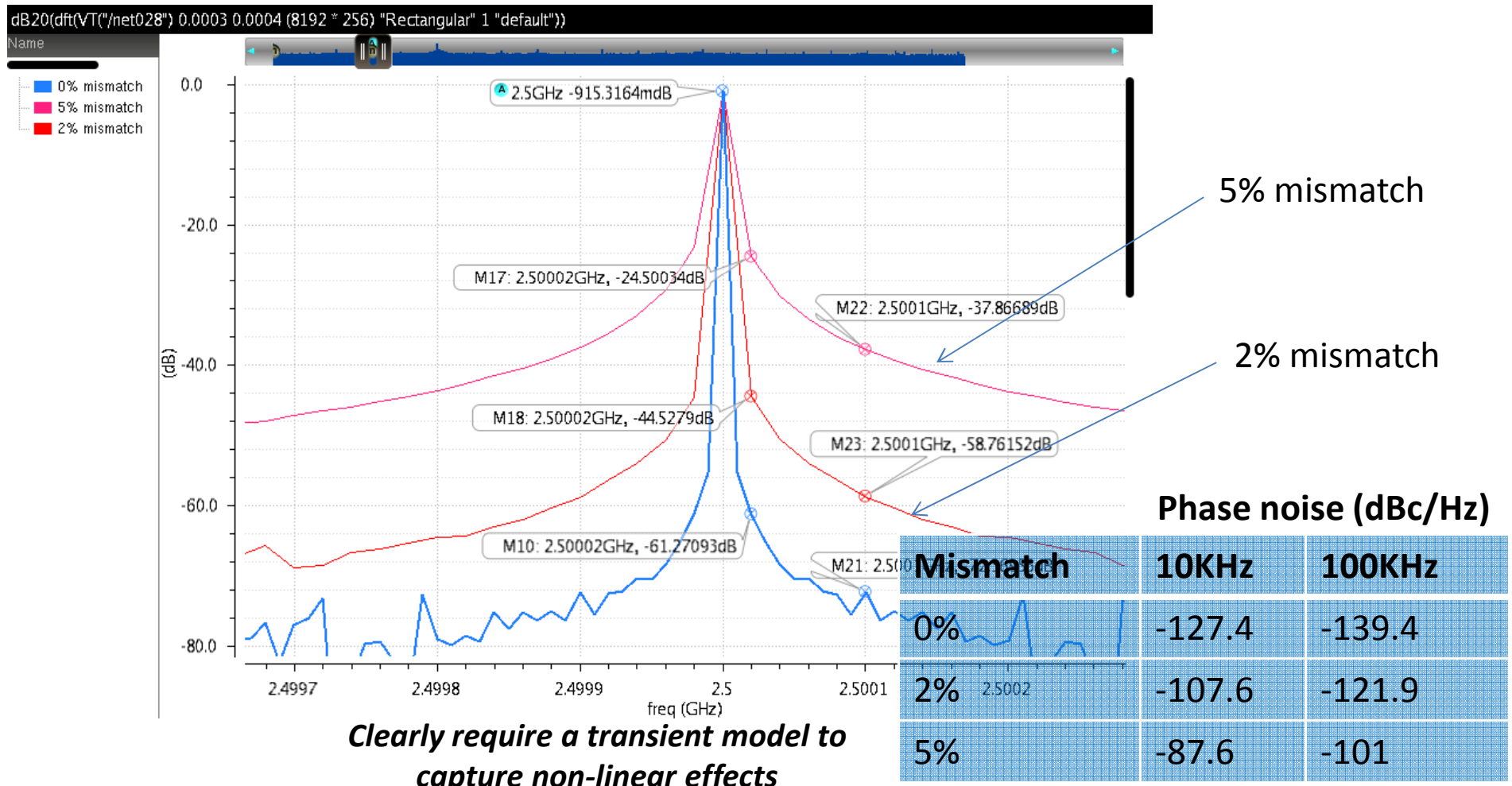


Charge pump dynamic current mismatch



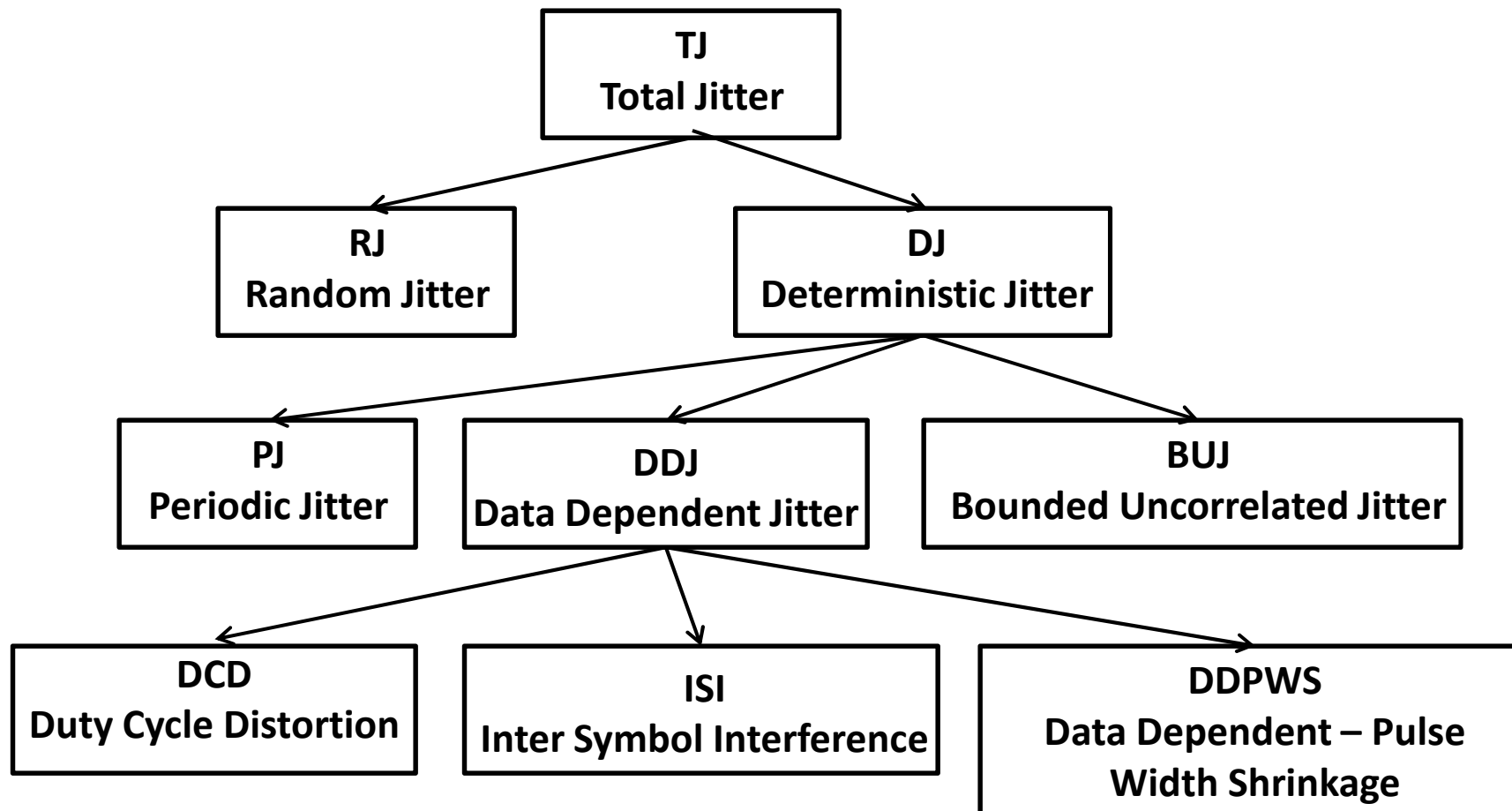
Phase-Locked Loops in SoC Processors

- Time-domain Verilog-A model with static current mismatch (UP/DN)



Phase-Locked Loops in SoC Processors

- Not all jitter created equal !



Outline

- ❑ Target Mixed-Signal Designs
- ❑ Phase-Locked Loops in SoC Processors
- ❑ **Conventional Modeling Approaches**
- ❑ Accurate and Time-Efficient Modeling Approach

Conventional Modeling Approaches

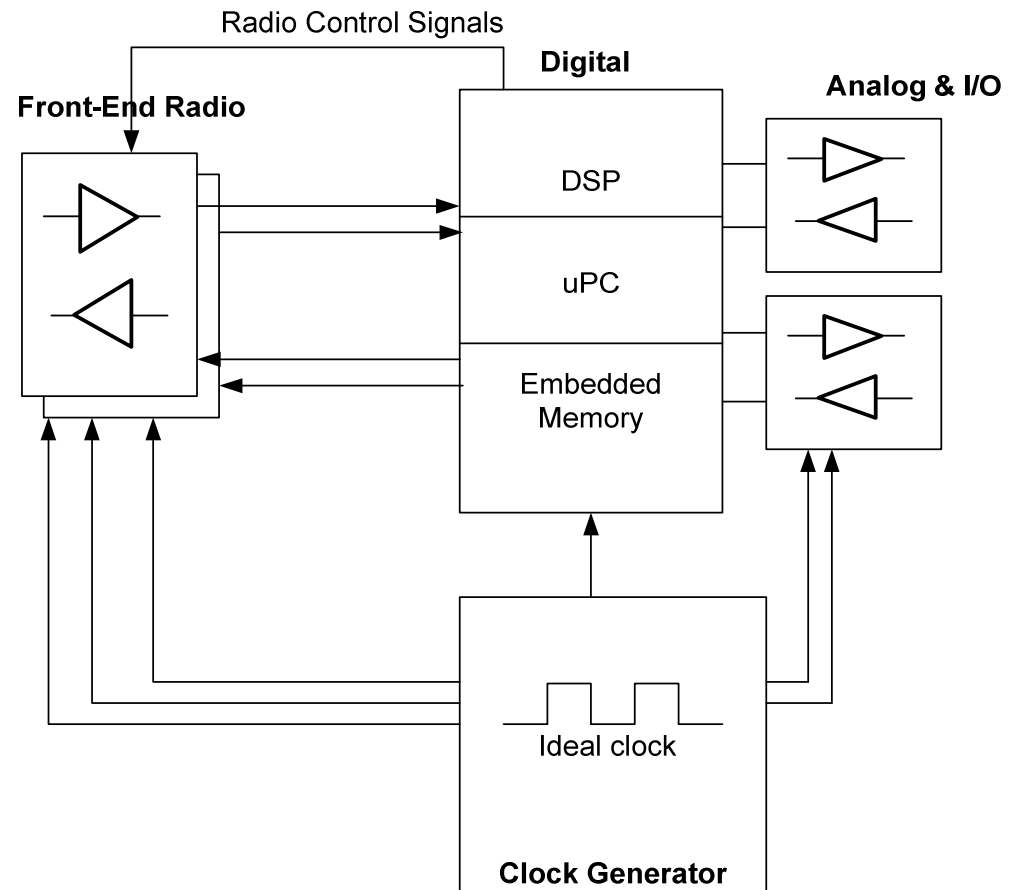
❑ Level 0: Classical SoC

Verification flow

- Limited functional modeling of analog blocks
- Ideal clock for Phase-Locked Loop

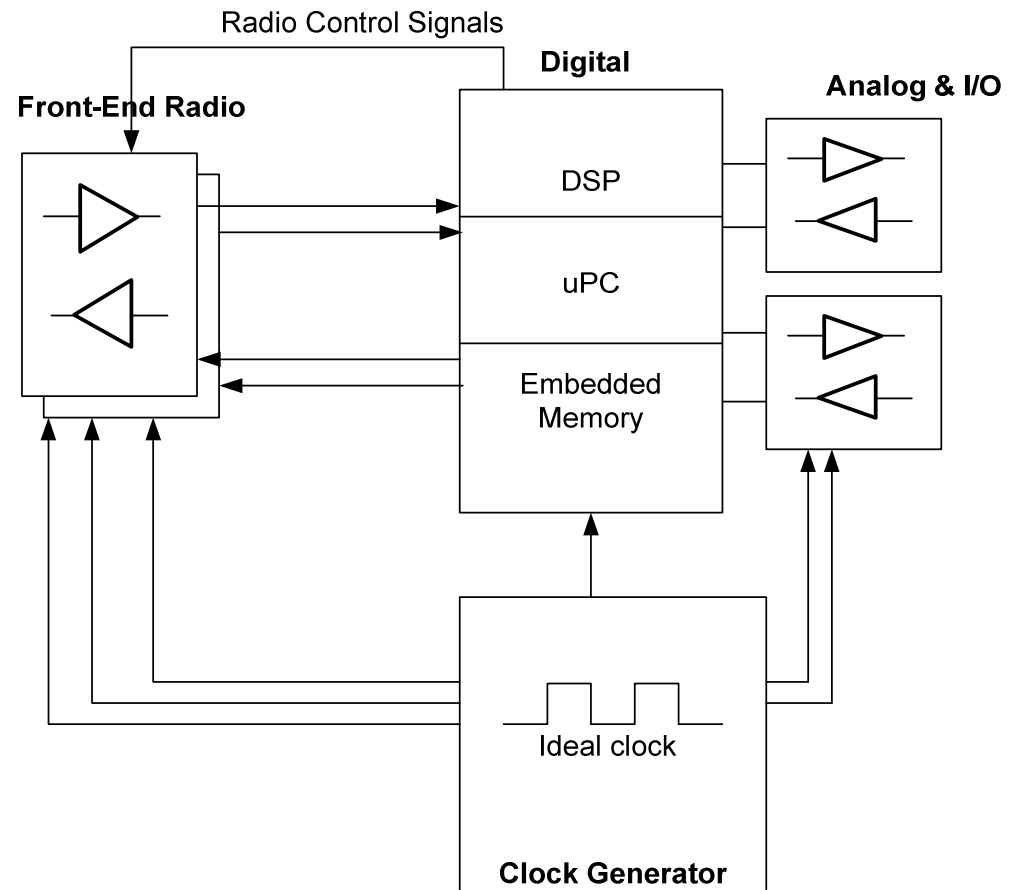
❑ Issues:

- Maintenance and verification of analog models
- Non-ideal analog effects ignored



Conventional Modeling Approaches

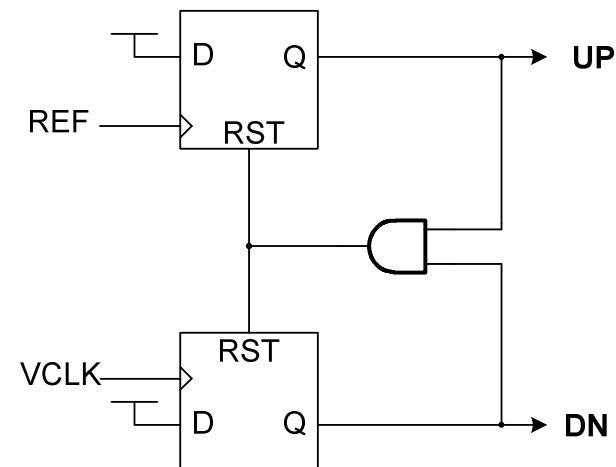
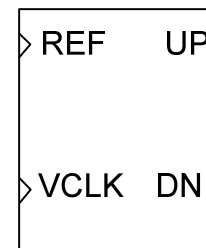
- ❑ Level 1: Analog Functional Verification Flow
 - Model analog blocks using Verilog-A /Verilog-AMS
 - Phase-Locked Loop ideal
- ❑ **Issues:**
 - Analog-centric flow – limited digital verification
 - How to simulate end-to-end verification in reasonable time?
 - Maintenance and verification of analog models



Conventional Modeling Approaches

PFD Verilog-A code

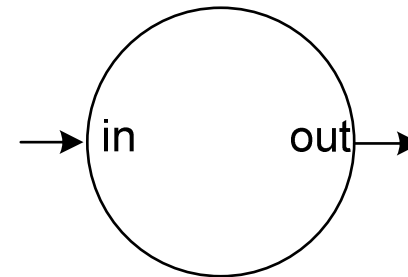
```
module pll_pfd (REF, VCLK, UP, DN);  
  inout REF, VCLK, UP, DN;  
  electrical REF, VCLK, UP, DN;  
  parameter real vdd=3.3, ttol=10f, ttime=0.2n ;  
  integer state; // state=1 for down, -1 for up  
  real td_up, td_down ;  
  Initial begin td_up = 1n; td_down=1n; end  
  analog begin  
    @(cross( V(REF) - vdd/2 , 1 , ttol )) begin  
      If (state > -1) state = state - 1;  
    end  
    @(cross( V(VCLK) - vdd/2 , 1 , ttol )) begin  
      If (state < 1) state = state + 1;  
    end  
    V(DN) <+ transition( (state + 1)/2*vdd , td_down , ttime );  
    V(UP) <+ transition( (state - 1)/2*vdd+vdd , td_up , ttime );  
  end  
endmodule
```



Conventional Modeling Approaches

VCO Verilog-A code

```
module pll_vco ( in, out ) ;  
  inout in, out ;  
  electrical in, out ;  
  parameter real vdd = 1.8,  
    Kvco = 60e6, // gain [Hz/V]  
    vnom = vdd/2,  
    fc =2.5e9;  
  real freq ;  
  analog begin  
    freq = fc + Kvco*(V(in) - vnom) ;  
    V(out) <+  
      ((sin(2*`M_PI*idt(freq)) > 0) ?  
       vdd : 0);  
  end  
endmodule
```



Conventional Modeling Approaches

PFD Verilog-AMS

```
`timescale 10ps / 1ps
module pfd (UP, DN, VCLK, REF);
  output UP, DN;
  input VCLK, REF;
  wire fv_rst, fr_rst;
  reg q0, q1;

  assign fr_rst = q0 & q1;
  assign fv_rst = q0 & q1;

  always @(posedge VCLK or posedge
    fv_rst) begin
    if (fv_rst) q0 <= 0; else q0 <= 1;
  end
  always @(posedge REF or posedge
    fr_rst) begin
    if (fr_rst) q1 <= 0; else q1 <= 1;
  end
  assign UP = q1;
  assign DN = q0;
endmodule
```

CP Verilog-AMS

```
`timescale 10ps / 1ps
module cp (lout, gnd, UP, DN);
  parameter real cur = 1m;//
  output current (A)
  input UP, DN;
  electrical lout, gnd;
  real out;

  analog begin
    @(initial_step) out = 0.0;

    if (DN && !UP) out = -cur;
    else if (!DN&& UP) out = cur;
    else out = 0;

    I(lout, gnd) <+ -transition(out,
    0.0, 10n, 10n);
  end
endmodule
```

VCO Verilog-AMS

```
`timescale 1ns / 1ps
module vco (in, out);
  parameter real fc= 2.5e9;
  parameter real Kvco = 60e6;

  output out;
  electrical in;
  reg out; logic out;
  initial out = 0;
  always begin
    #(0.5e9 / (fc + kvco * V(in)))
    out = ~out;
  end
endmodule
```

Comparison of Conventional Modeling Approaches

- Summary comparison table:

Model Level	Speed	Accuracy	SoC sim friendly?
Linear model	++	0	N
Level 0	++	--	Y
Level 1	-	++	N
Level 1+	0	++	N

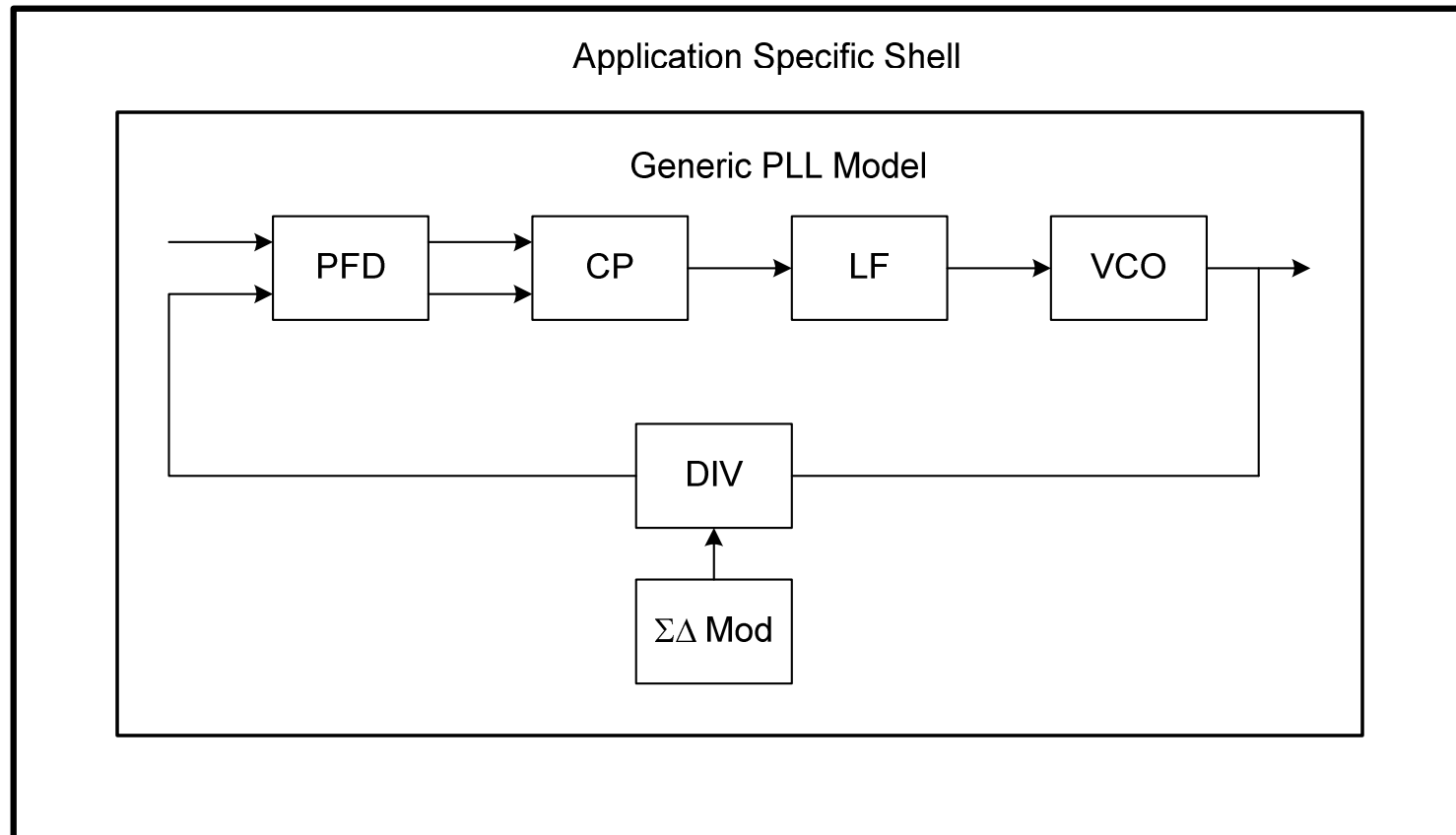
Outline

- ❑ Target Mixed-Signal Designs
- ❑ Phase-Locked Loops in SoC Processors
- ❑ Conventional Modeling Approaches
- ❑ **Accurate and Time-Efficient Modeling Approach**

Accurate and Time-Efficient Modeling Approach

- ❑ ***Pure Verilog-D model of a PLL***
- ❑ Why Verilog-D model?
 - Event driven simulator → large speedup!!
 - Compatibility with SoC simulation environment
- ❑ Challenges to Verilog-D model:
 - How to model analog charge pump current, analog loop filter voltage?
 - How to deal with any other shortcoming of a pure Verilog-D simulator?
 - How to model noise effects in a pure Verilog-D simulator?

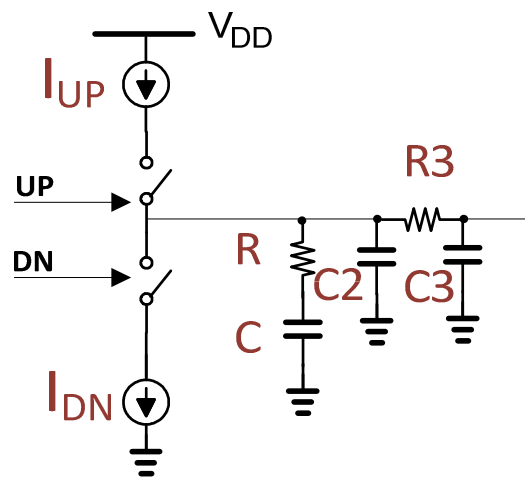
Accurate and Time-Efficient Modeling Approach



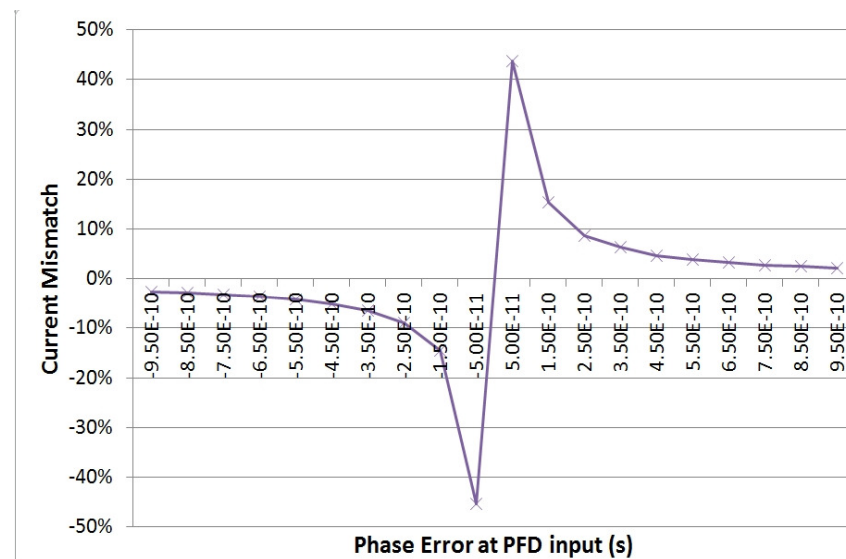
- Generic parameterizable PLL model that can be used in multiple applications
- Application specific shell to interface with rest of the chip
- Time-domain noise models embedded in each block
- Charge pump nonlinearity modeled in time-domain

Accurate and Time-Efficient Modeling Approach

- PLL Loop Filter and charge pump nonlinearity combined



Charge pump dynamic current mismatch



Steps:

1. Generate a lookup table to capture the charge pump nonlinearity ($I - \Delta T$ curve)
2. Compute the transfer function of the loop filter (time-domain)
3. Apply Taylor Series expansion of the composite transfer function of each exponential term and limit number of terms fast simulation time).

Accurate and Time-Efficient Modeling Approach

- ❑ Lookup table for $I - \Delta T$ curve of charge pump:
 - Generated from transistor level simulation

- ❑ Model for loop filter (time-domain expression):

$$y_{filt}(\Delta t) = \frac{A_0}{C_2} \Delta t + A_1 \exp\left[-\frac{1 + C/C_2}{RC} \Delta t\right] + A_2 \exp\left[-\frac{\Delta t}{R_3 C_3}\right]$$

- where A_0, A_1, A_2 are coefficients dependent on loop parameters

Accurate and Time-Efficient Modeling Approach

- ❑ Model used to compute the gradient for a time-domain simulation
- ❑ Most terms in the form of decaying exponentials:

$$f(x) = 1 - \alpha_0 \exp[-\beta_0 t] - \dots - \alpha_i \exp[-\beta_i t]$$

- ❑ Gradient is, therefore, in the form of:

$$f'(x) = \alpha_0 \exp[-\beta_0 t] + \dots + \alpha_i \exp[-\beta_i t]$$

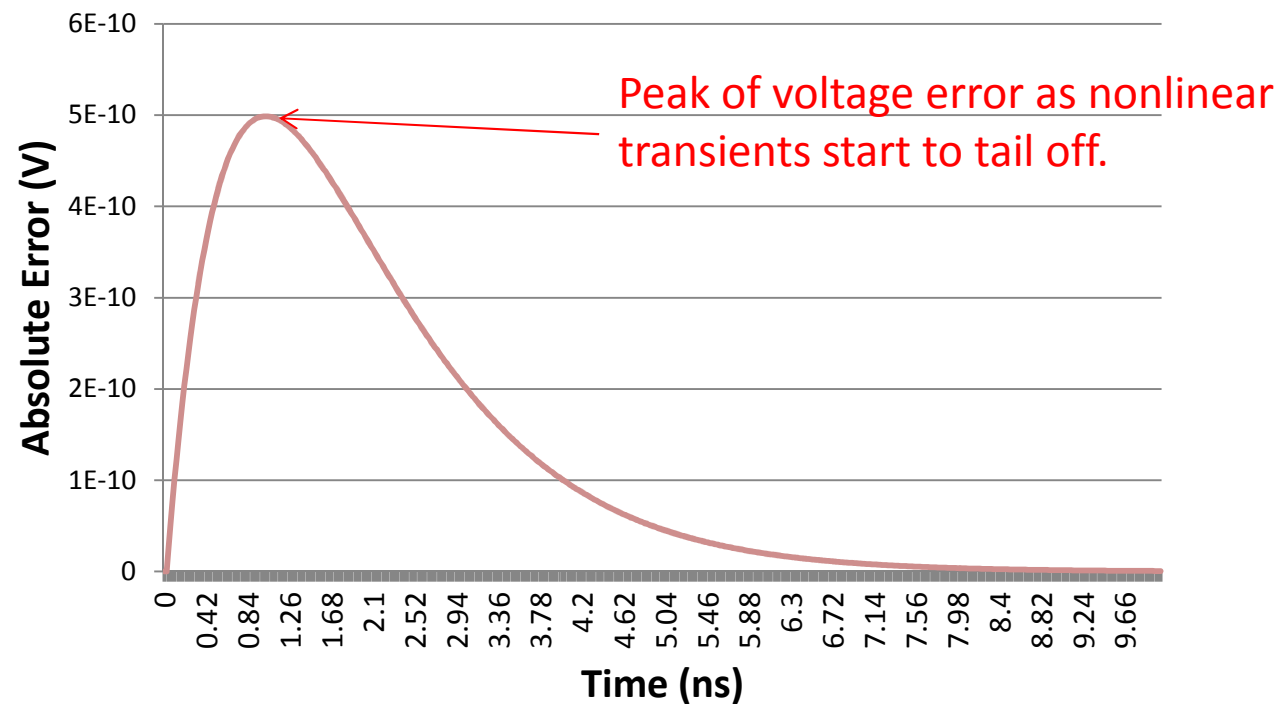
or

$$f'(x) = 1 - f(x)$$

- ❑ To obtain a numerical solution at each iteration, a Runge-Kutta algorithm (RK4) with variable time step was used. This provides < 0.5nV voltage error with linear computations.

Accurate and Time-Efficient Modeling Approach

- Error analysis between proposed approach and transient simulation:
 - Simulation involves PFD+CP+LF over 10ns (with 100MHz reference)
 - Absolute error < 0.5nV
 - This error curve is periodic



Accurate and Time-Efficient Modeling Approach

□ Standard Verilog-D VCO model:

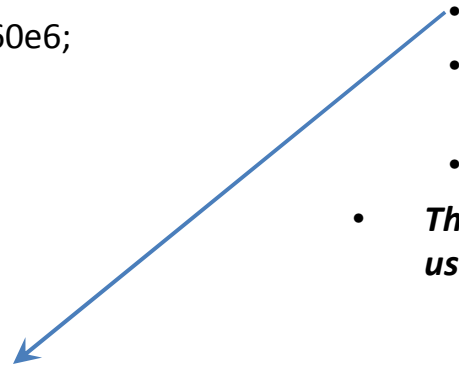
```

`timescale 1ns / 1ps
module vco (in, out);
    parameter real fc= 2.5e9;
    parameter real Kvco = 60e6;

    output out;
    electrical in;
    reg out; logic out;
    initial out = 0;
    always begin
        #(0.5e9 / (fc + kvco * V(in)))
        out = ~out;
    end
endmodule

```

- Minimum time-step of Verilog-D simulator is 1fs → sufficient for most, if not all, digital applications
- This is problematic in VCO model:
 - **Min resolution in computation is 1fs**
 - **If VCO period = 250ps (4GHz), maximum error is bounded to 1fs → $1/(250\text{ps}+1\text{fs}) \cong 16\text{kHz}$**
 - **Error is too large for most applications!**
- ***This frequency error must be resolved for practical use of a VCO model in Verilog-D!***



Accurate and Time-Efficient Modeling Approach

□ Modified VCO Verilog-D model:

➤ Make use of the Verilog-D built-in “real” datatype to store the actual VCO period

➤ Steps:

1. Compute the desired VCO period: `VCO_period_desired`

1. This will be stored in a “real” datatype for maximum precision

2. Truncate the `VCO_period_desired` to 1fs: `VCO_period_actual`

3. Store the error in an accumulated variable:

1. **`err_accum = err_accum + (VCO_period_desired - VCO_period_actual)`**

2. If the **`err_accum`** > 1fs, then increase the `VCO_period_actual` by 1fs and subtract 1fs from the **`err_accum`** variable.

4. Repeat steps 1-3 at the end of every VCO period

➤ ***Result is a zero average frequency error!***

- Operation is similar to a fractional-N divider operation

- The “fractional-N spurs” produced are very low.

- Spur level_{max} (dB) = $20 \log \left[\frac{1fs}{VCO_period_desired} \right]$ (fvco=1GHz → spur = -120dBc)

Accurate and Time-Efficient Modeling Approach

- Sample VCO code with zero frequency error:

```


`timescale 1ps/1fs
module PLL_vco(VCO_clk, V_ctr);
    output VCO_clk;
    input V_ctr;

    reg VCO_clk;
    real V_ctr, VCO_semiperiod, VCO_semiperiod_act, err_acc, fvco, diff,
    Fvco_center;

    always # (VCO_semiperiod_act) ) begin
        VCO_clk = ~VCO_clk;
        fvco = Fvco_center + V_ctr * `Kv; // in MHz
        VCO_semiperiod = 1 / (2 * f_run) * 1e6; // in psec
        diff = VCO_semiperiod - VCO_semiperiod_act; // difference in psec
        err_acc = err_acc + diff; // accumulated diff in psec
        if (err_acc > 1e-15) begin
            VCO_semiperiod_act = VCO_semiperiod + 1f;
            err_acc = err_acc - 1e-15;
        end
    end
endmodule

```

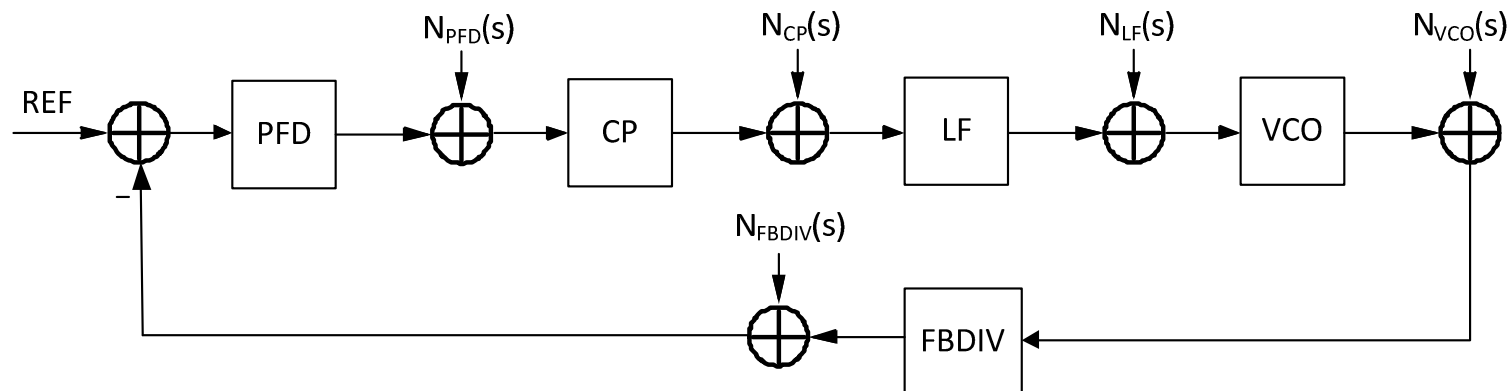
Code that enables
sub-fs resolution



Accurate and Time-Efficient Modeling Approach

□ Noise modeling strategy:

- Obtain noise numbers from transistor level periodic steady-state (pss) noise simulation of each PLL sub-block
- Noise will be a composite of flicker and thermal noise

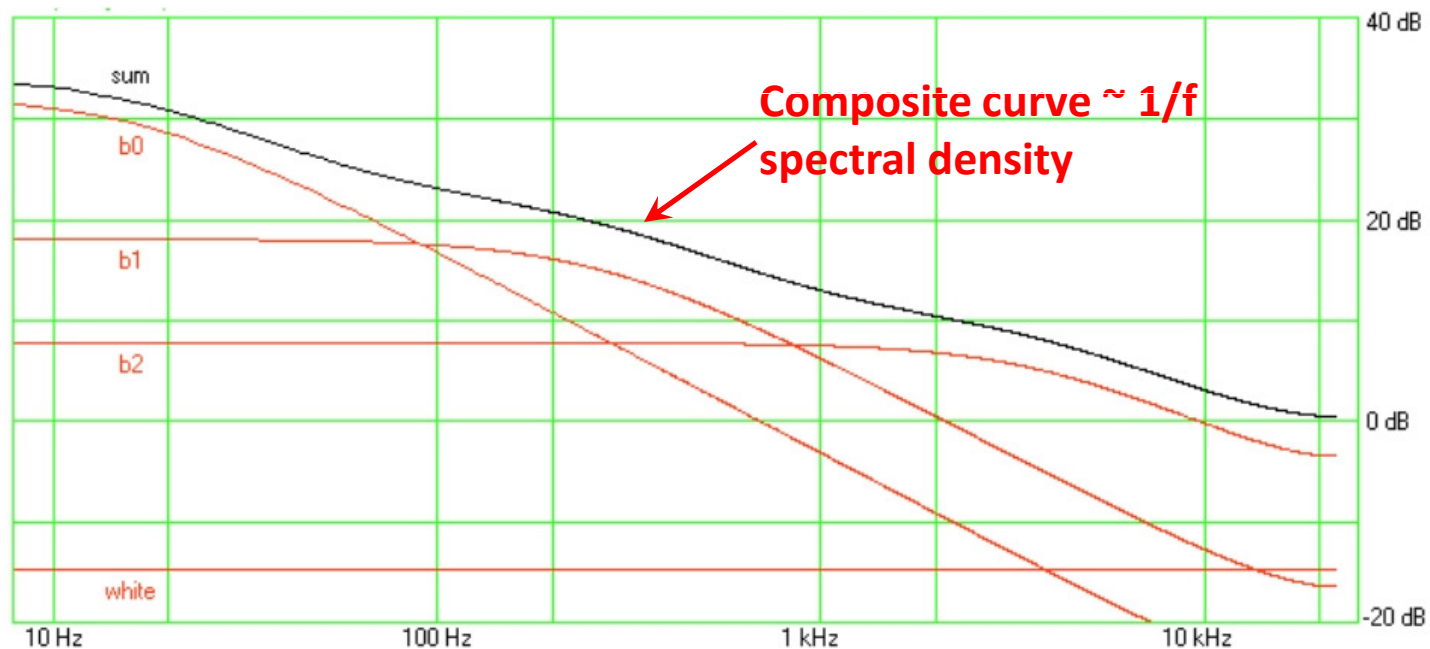


$$N(f) = N_{PFD}(f) \cdot \|H_{PFD}(f)\|^2 + N_{CP}(f) \cdot \|H_{CP}(f)\|^2 + N_{LF}(f) \cdot \|H_{LF}(f)\|^2 + N_{VCO}(f) \cdot \|H_{VCO}(f)\|^2 + N_{FBDIV}(f) \cdot \|H_{FBDIV}(f)\|^2$$

Accurate and Time-Efficient Modeling Approach

□ Flicker noise:

- Based on well-established and efficient Voss-McCartney algorithm to generate pink ($1/f$) noise densities (as well as $1/f^3$ for VCO up-converted flicker noise).
- Illustration of basic idea of Voss-McCartney algorithm:

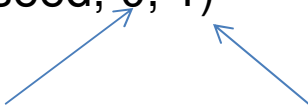


Accurate and Time-Efficient Modeling Approach

- ❑ Thermal Noise:
 - White noise, uniformly distributed
 - Verilog-D built-in function: \$rdist_normal()
- ❑ Relationship of phase noise & absolute jitter:

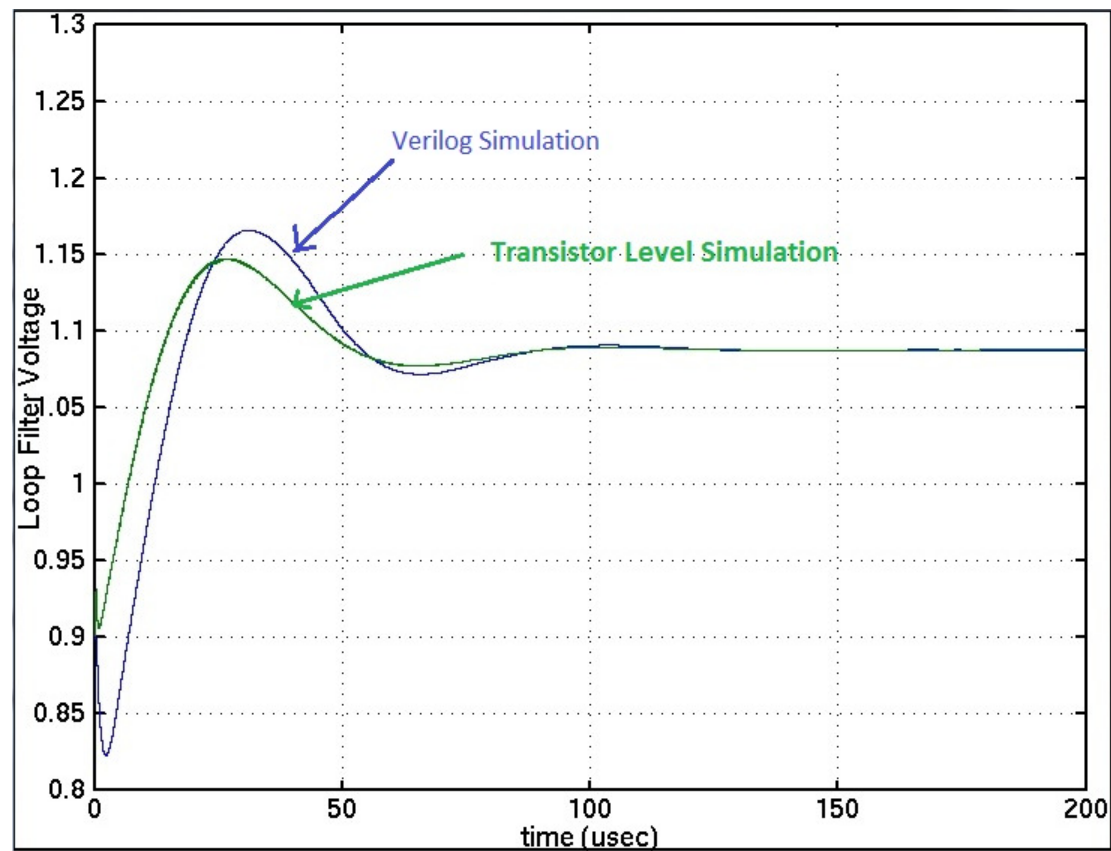
$$\sigma_{abs}^2(t) = 4 \int_{-\infty}^{\infty} S_{\phi}(f) \sin^2(\pi ft) df$$

- Noise bandwidth is usually bounded for most applications
- Thermal noise is both white and uniformly distributed
- Example: Phase noise = -160dBc/Hz @ 1GHz from 10MHz to 20MHz
 - Jitter : $\sigma_n^2 = 4 \times 10^{-9} \text{ nV}^2$
 - Jitter*\$rdist_normal(seed, 0, 1)


 Zero mean Std deviation

Simulation Results

- Transient simulation validating Verilog-D model
 - Difference in behavior due to nonlinear varactor characteristic not being modeled in Verilog-D



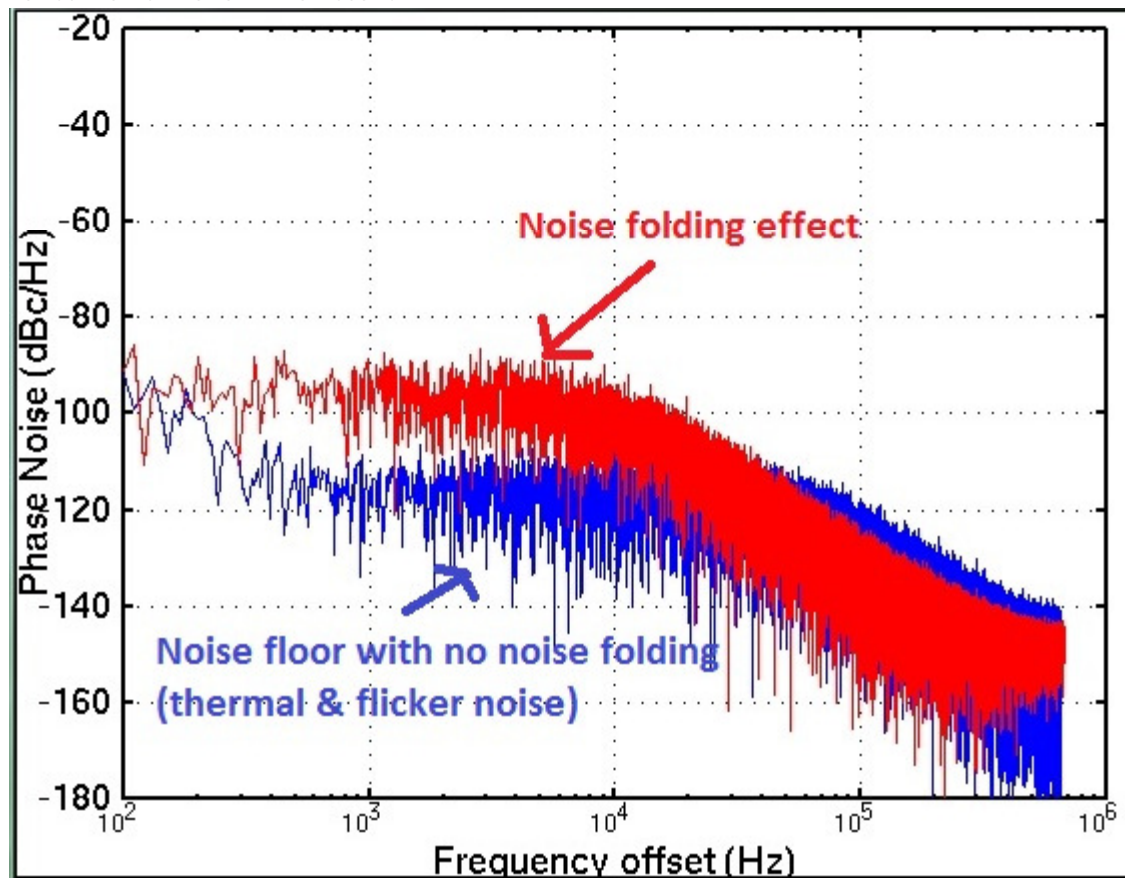
Simulation Results

- ❑ PLL Simulation Time:
 - 200usec transient simulation
 - Only Verilog-D model contains noise information
 - Using Icarus Verilog running on an Intel i7-2.4GHz machine

Model Type	Simulation Time
Transistor Level	1636 minutes
Verilog-A	36.3 minutes
Verilog-D	2.75 minutes

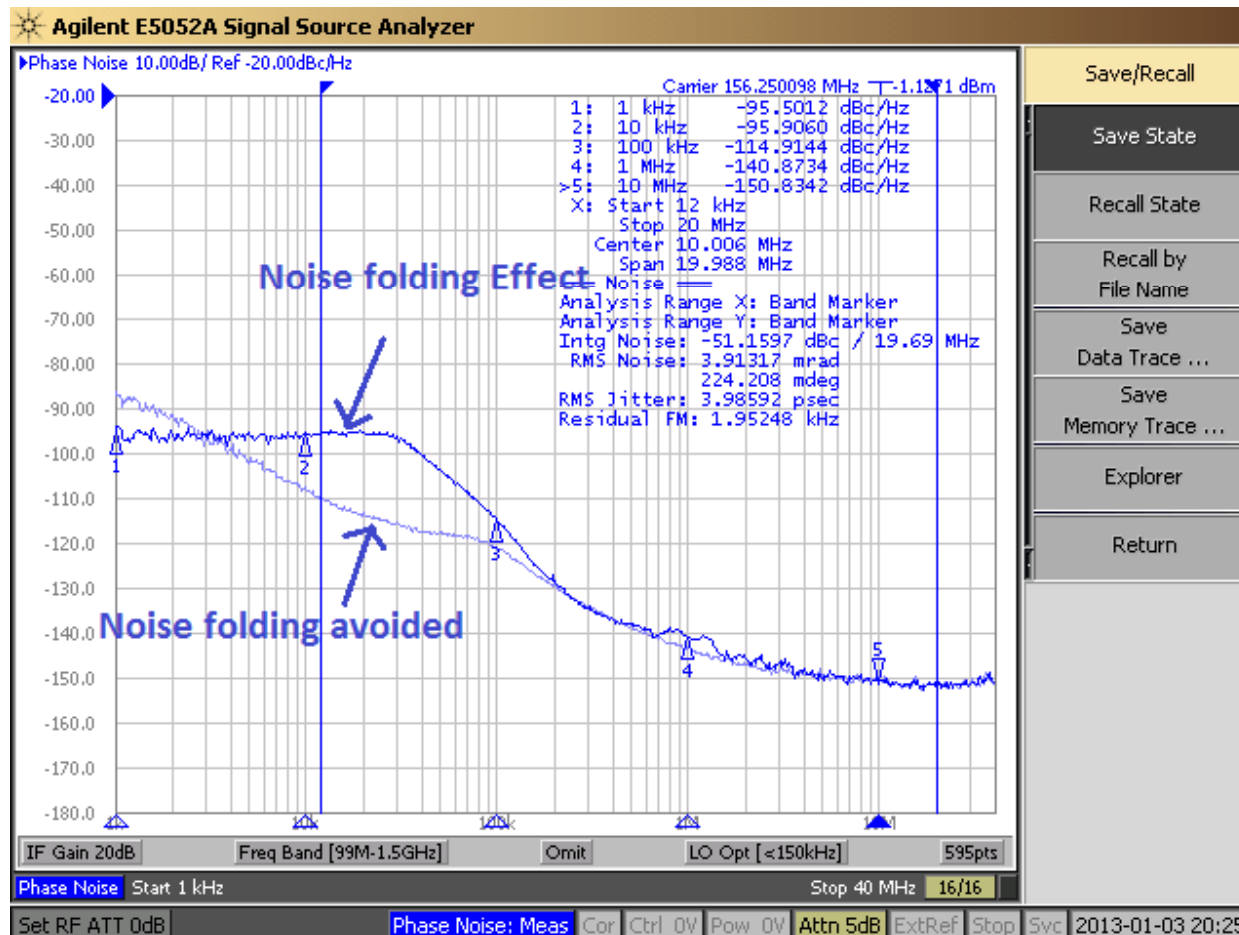
Simulation Results

- FFT of transient simulation with and without noise folding effect:
 - Linear & non-linear charge pump
 - 20ms transient simulation



Measured Results

- Semtech's ACS1790 fractional-N Phase-Locked Loop was used to validate model



Summary

- ❑ Reviewed PLL basics and sources of noise in PLLs
- ❑ Reviewed classical modeling techniques for PLLs
- ❑ Introduced a new model approach based on pure Verilog-D
 - Compatible with digital verification flows
 - Non-linear noise folding effect in $\Sigma\Delta$ PLL is well predicted
 - Noise models were also included to provide a full picture of total performance
 - Modeling methodology can be extended to other analog/RF circuits

References

1. J. Park, K. Muhammad, and K. Roy, "Efficient Modeling of 1/f Noise Using Multirate Process," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, No. 7, July 2006, pp. 1247-1256.
2. A. Kuo, T. Farahmand, N. Ou, S. Tabatabaei, and A. Ivanov, "Jitter Models and Measurement Methods for High-Speed Serial Interconnects," *ITC Int'l Test Conference*, pp. 1295-1302, 2004.
3. J. Nan, J. Ren, M. Cong, and L. Mao, "Design of PLL behavioral model based on the Verilog-A," *4th Int'l Symposium on MAPE*, pp. 380-383, 2011.
4. T. Wen and T. Kwasniewski, "Phase Noise Simulation and Modeling of ADPLL by SystemVerilog," *BMAS 2008*, pp. 29-34, 2008.
5. Y. Wang, C. Van-Meersbergen, H. Groh, and S. Heinen, "Event Driven Analog Modeling for the Verification of PLL Phase-Locked Loops," *BMAS 2009*, pp. 25-30, 2009.
6. A. Fahim, *Clock Generators for SoC Processors: Circuits and Systems*, Boston: Kluwer Academic Publishers, 2005.