Event Driven Analog Modeling for the Verification of PLL Frequency Synthesizers

Yifan Wang, Christoph Van Meersbergen,
Hans-Werner Groh* and Prof. Dr. Stefan Heinen

Chair of Integrated Analog Circuits, RWTH Aachen University, Germany

http://www.ias.rwth-aachen.de

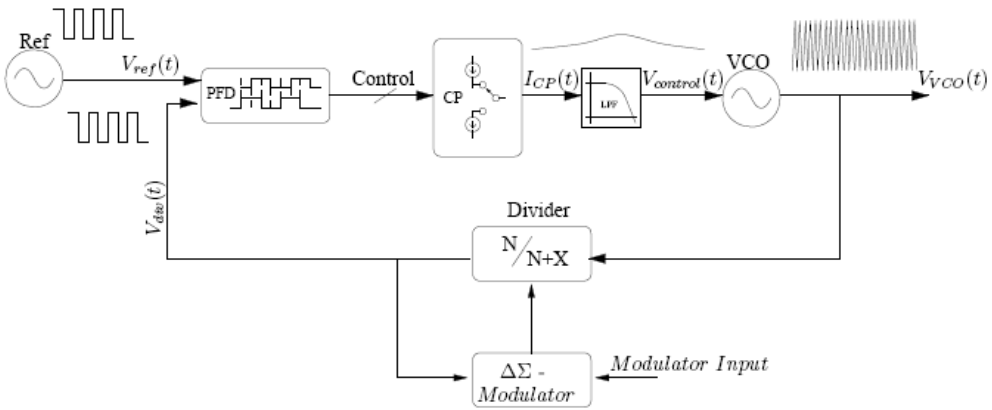
* Atmel Automotive GmbH

Outline

- Motivation & Target
- Proposed PLL models and comparison with other modeling approaches considering
  - Performance
  - Accuracy
  - Mixed-Level Simulation Compatibility
- Application example
- Conclusions

Motivation & Target

- Apply wreal data type to provide a efficient modeling approach for PLL System
- Map analog specification to event driven domain
- Allow rough performance analysis
- Ensure Mixed-Level simulation capability



Event driven loop filters

- Map the analog specification in frequency domain to event driven time domain
- Non uniform sampled filters applying forward Euler Methods

Non uniform sampled implementation:

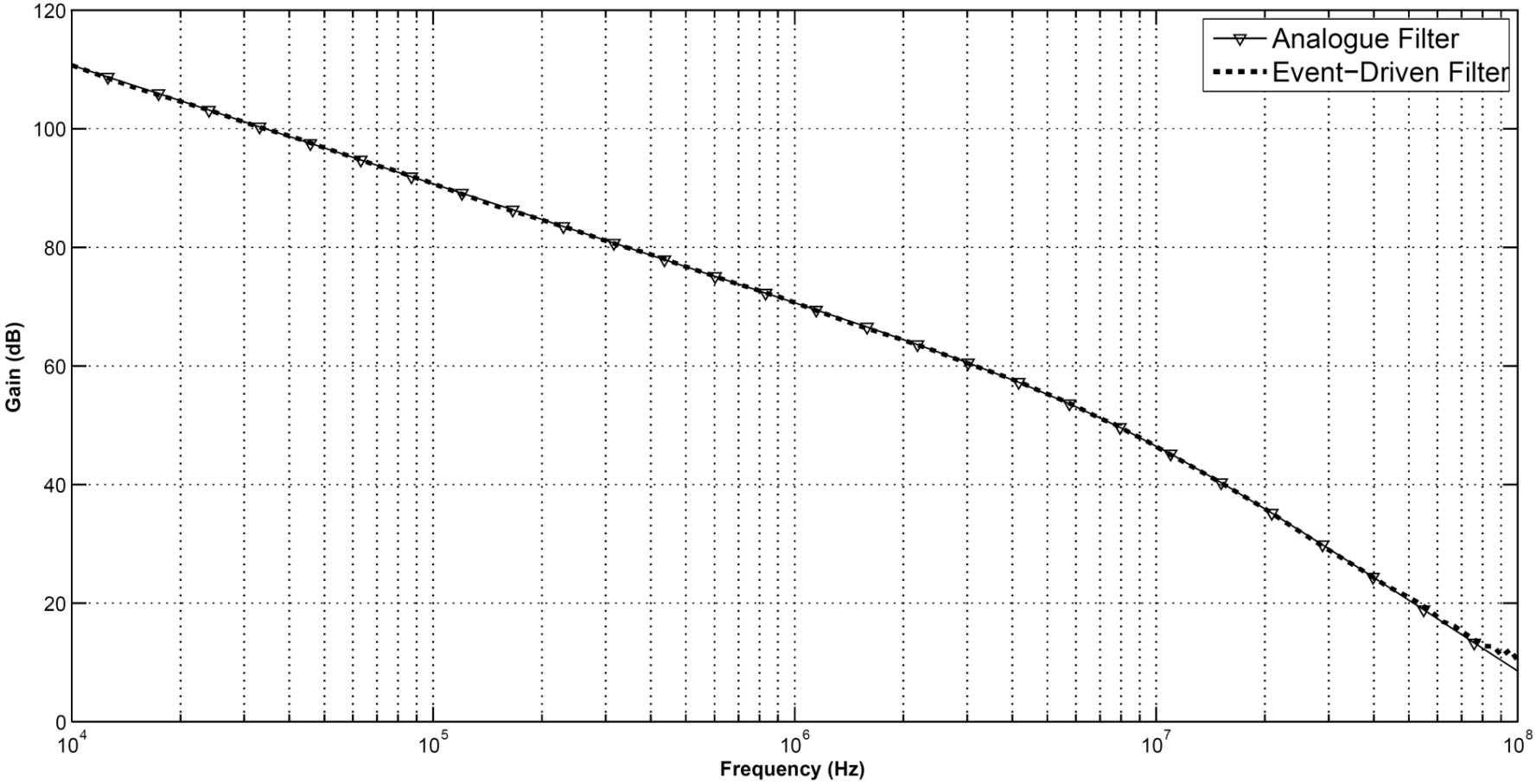
```
assign out=yn ;
always@( in ) begin
xn=in; act_time=$realtime;
hn=1.0f*(act_time-prev_time);
dxn=(xn-xn1 )/hn;
vn=vn1+hn*xn1; un=un1+hn*yn1;
yn=(( z0*vn+z1*xn+z2*dxn-
p0*un)/p1+p2/p1*yn1/hn)
/(1.0+p2 /( p1*hn ) ) ;
```

Uniform sampled implementation:

```
assign out=y0
always #ts begin
x2 = x1; x1 = x0; x0 = in;
y2 = y1;  y1 = y0;
y0 = ( (n0*x0) + (n1*x1) + (n2*x2)
– (d1*y1 + d2*y2) ) / d0;
```

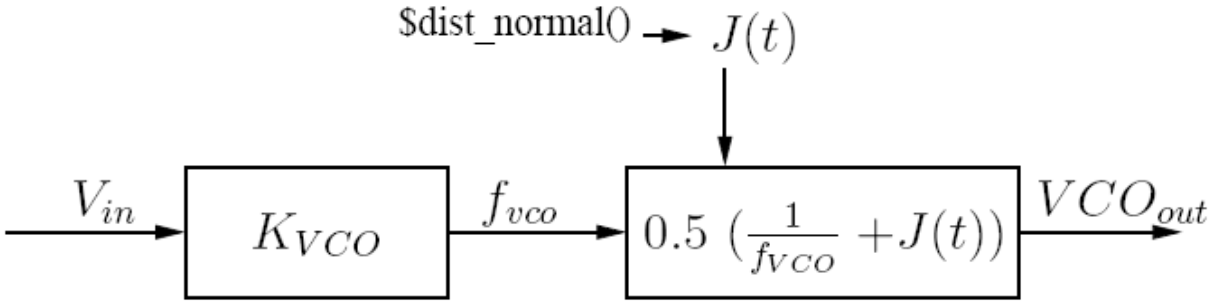
Loop filter: Accuracy comparison

AC characteristics

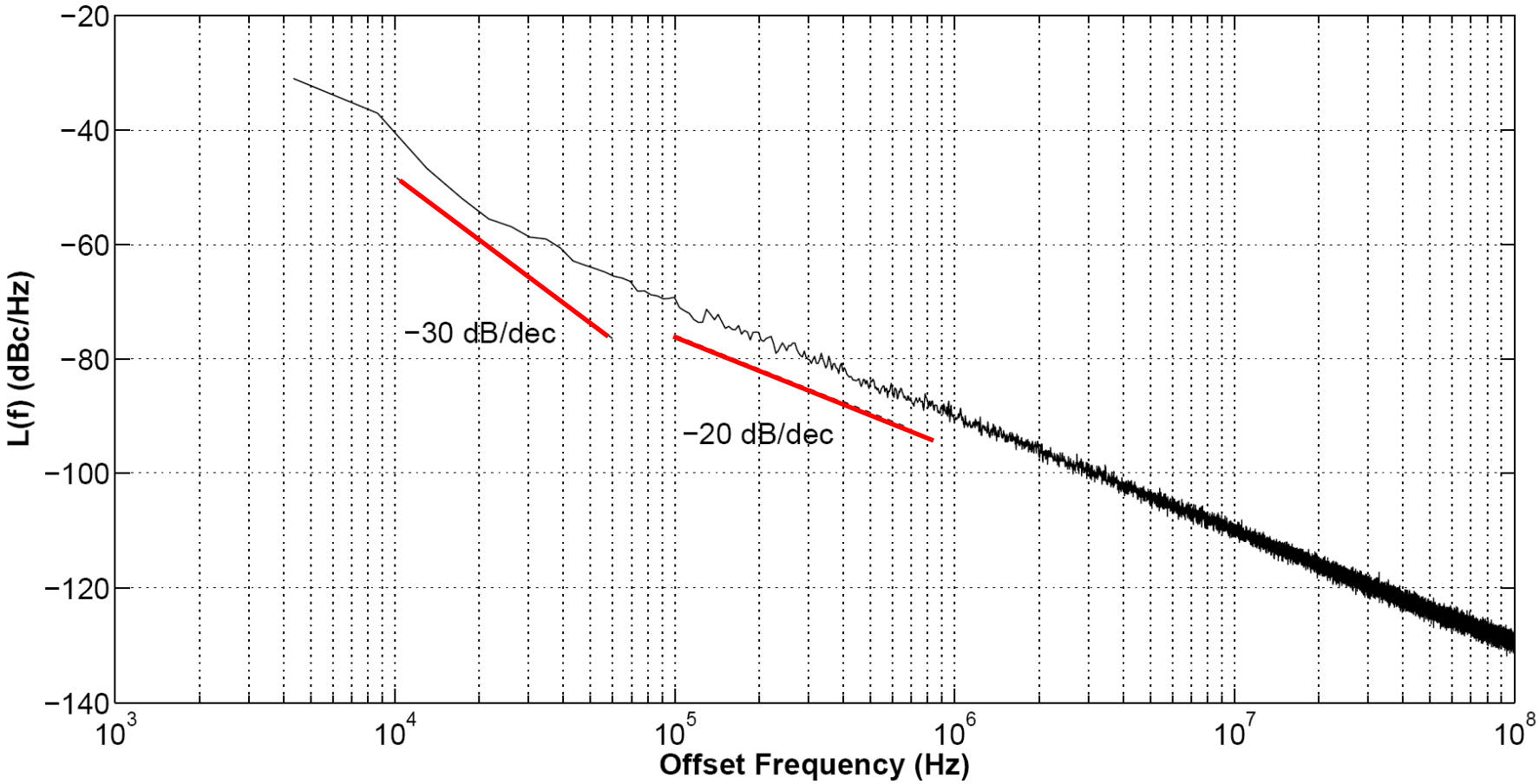


VCO: Model implementation

- Frequency & tuning range
- Phase noise
  - Equivalent accumulating jitter in time domain
  - Covers -30dB/dec & -20dB/dec regions



VCO: Phase noise simulation result



Free running VCO @ 1GHz, -90 dBc/Hz @1 MHz offset

VCO: Phase model noise implementation

```
accJitter = sqrt(pow(10,phase_noise/10.0)*pow(Deltaf,2.0)/pow(freq,3));
freq_n= freq/(1+noise*freq);
phi = 2*`M_PI*idt(freq_n,0);
V(phase) <+ 2*`M_PI*(idtmod(freq_n,0.0,1.0,-0.5));
@(cross(V(phase)-0.5*`M_PI,1,ttol) or cross(V(phase)+0.5*`M_PI,1,ttol))
begin
   noise = accJitter*sqrt(2)*$rdist_normal(seed,0,1);
end
V(out) <+ phi ;
```

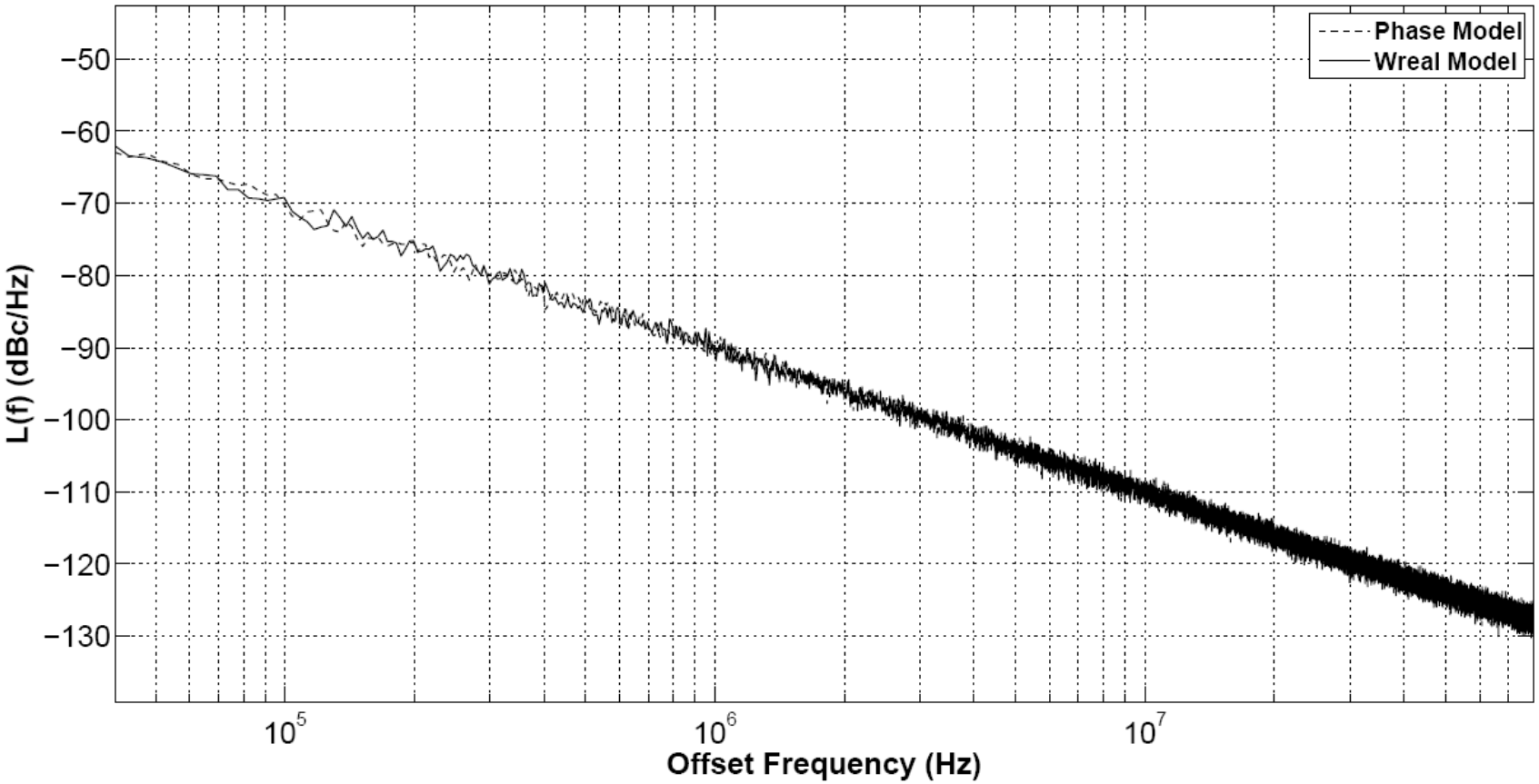
- *Cross()* function is most time critical during simulation
- Worst case: *bound_step* has to be used to ensure correct *cross()* detection

VCO: Wreal model noise implementation

```
always #next begin
c=pow(10,phase_noise/10.0)*pow(Deltaf,2.0)/(pow(freq,2.0));
accJitter=sqrt(c/(freq*2.0));
dT0=accJitter*$dist_normal(dSeed,0,1);
dT1= dT*fc*next+dT1_last;
next= 0.5/freq+dT0+dT1;
dT1_last = dT1;
end
```

- No *timer()* or *cross()* functions
- Ensure the appropriate timescale is selected in order to map the jitter specification correctly

VCO: Comparison of wreal and phase model



- Free running VCOs @1 GHz, -90 dBc/Hz @ 1MHz offset
- 10M oscillations are simulated
- Phase model requires 8 min. with wreal model reduced to 31sec.

Divider

- Synchronous jitter implementation based on transistor level spec
- Suitable for hierarchical switching between transistor and model level

  ➔Divider is not embedded in the VCO model

  ➔Capable to explorer different divider architectures

Divider

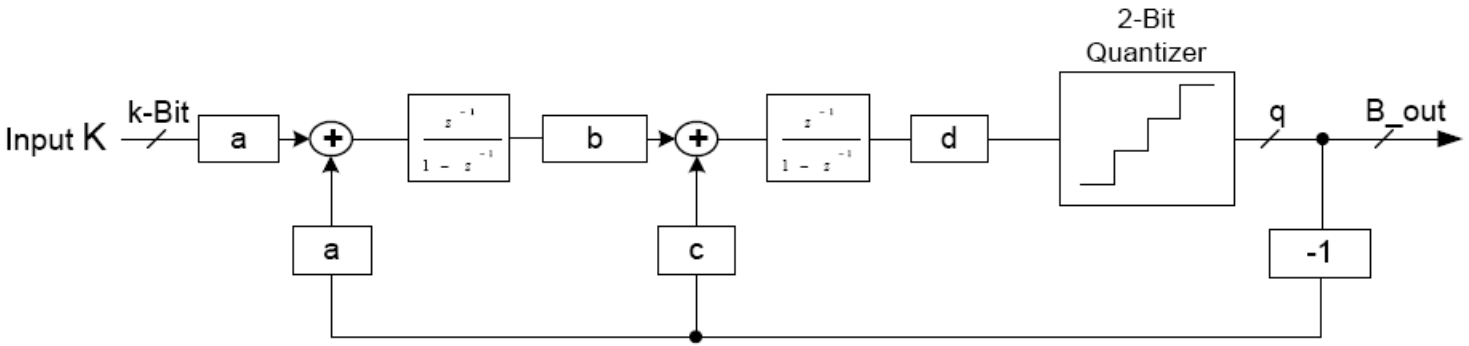
- There are no wreal/logic connect modules
- 2 possibilities for edge detection:
  - Use internal digital wire

```
Module divider(in, out, ctrl)
Wreal in, ctrl;
wire dig_intern;
assign dig_intern=(in>Vth? 1'b1:1'b0);
always@(posedge dig_intern or negedge dig_intern) begin
//noise & couter implementation
```

  - Pure wreal implementation

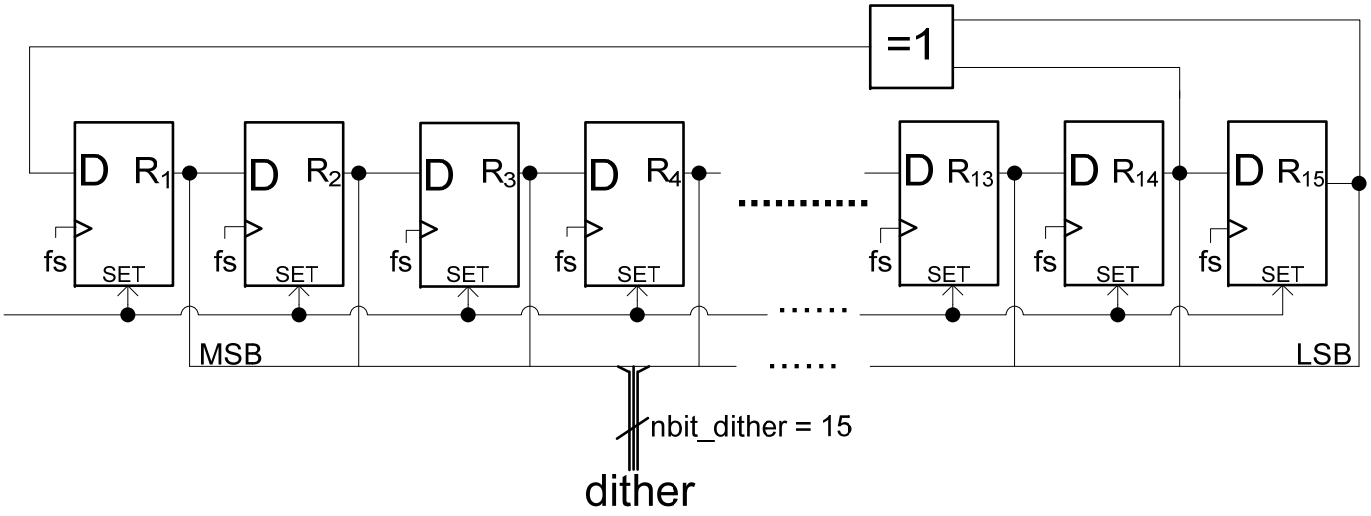
```
Module divider(in, out, ctrl)
Wreal in, out, ctrl
always@ (in) being
If((in_last < vth && in>vth ) || (in_last>vth && in<vth))begin
In_last=in;
…
```

Delta Sigma Modulator



- The signal flow is modeled based on the transfer function of the modulator
- Dithering using 15bit Linear-Feedback-Shift-Register (LFSR), in order to reduce Spurious emission

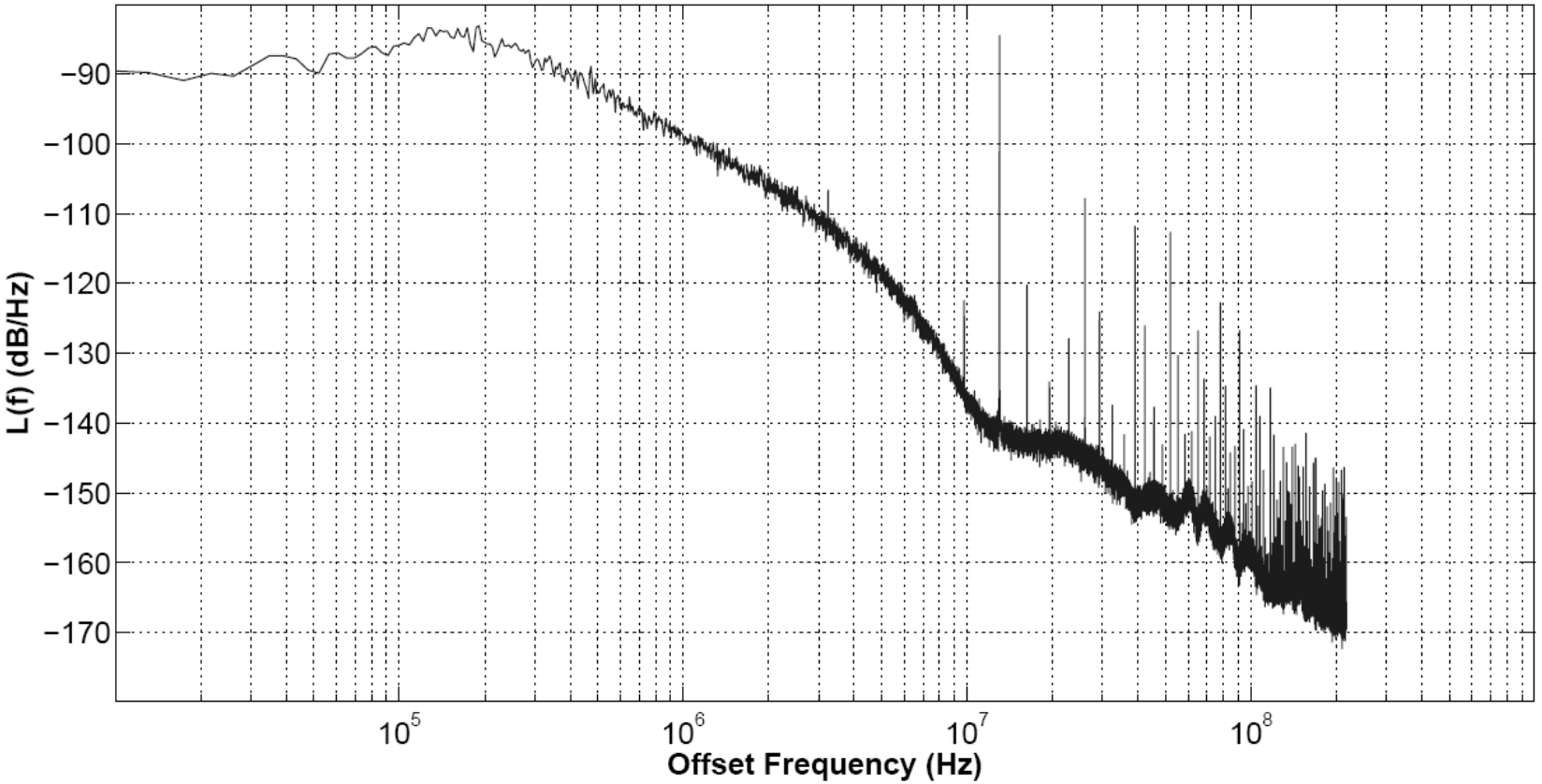
Delta Sigma Modulator: Dithering structure



- The LSFR can be implemented pure digitally or using wreal
- Output of LSFR is applied into the first accumulator of the Delta Sigma Modulator

Delta Sigma Modulator: Noise contribution

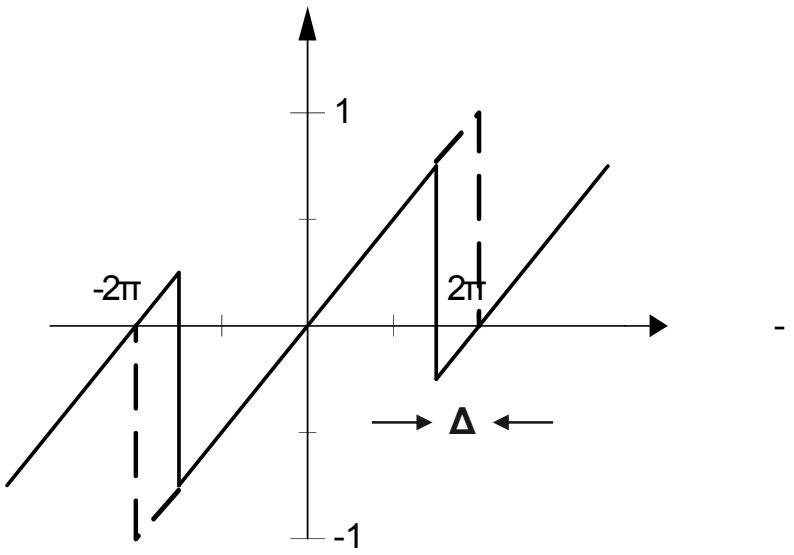
Phase noise and spurious emission evaluation



- Clock rate: 13 MHz
- Further noise sources in the PLL are not taken into account

PFD

- Pure digital implementation
- Synchronous jitter
- Delay time based on spec, in order to map the correct linear region of PFD
- Reduced linear region of PFD slows down the locking process



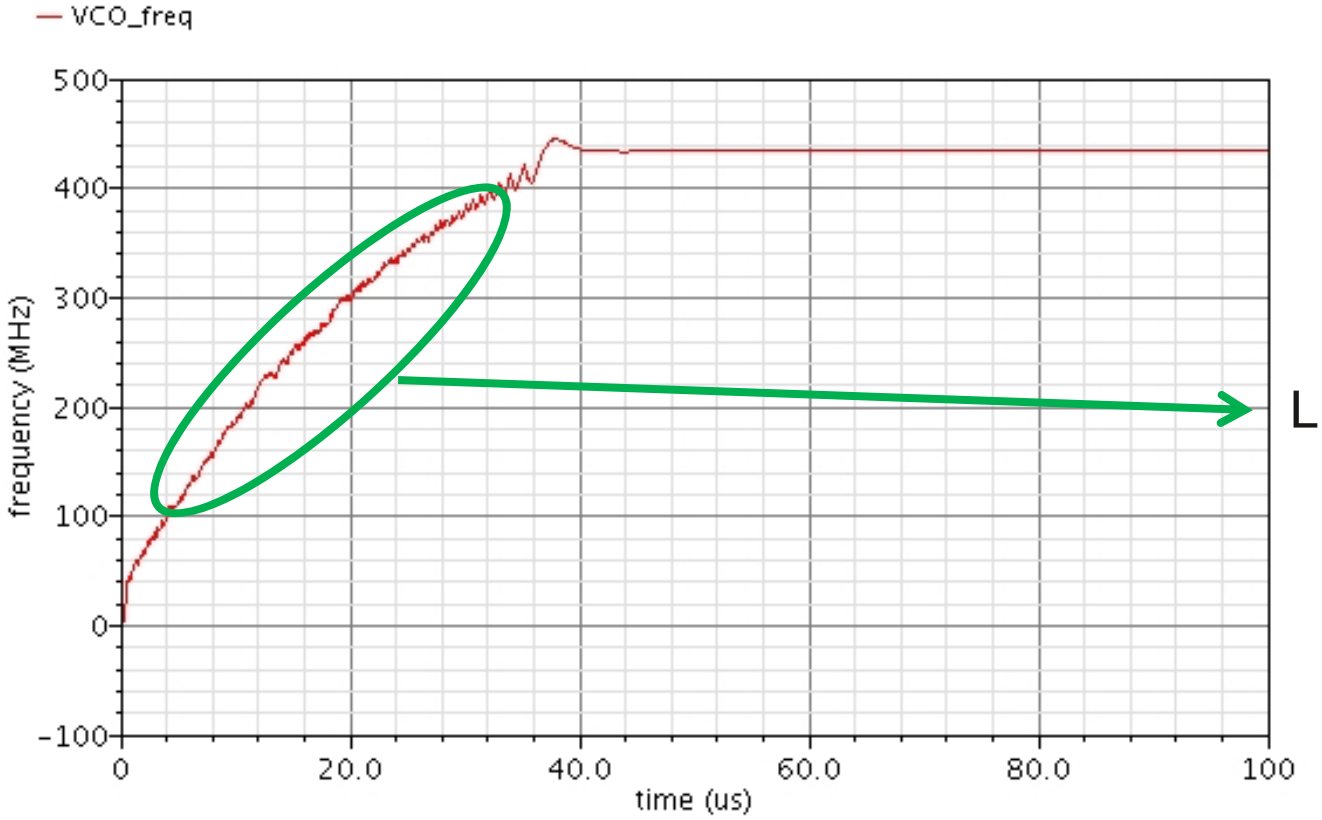
$$t_{delay} = \frac{\Delta}{2\pi f_{ref}}$$

Application example

- Atmel Tire Pressure Monitoring (TPM) transmitter
- Dual path, Fractional-N structure
- All parameter extractions are based on transistor level SPICE simulation
- PLL output: 433 MHz, Ref: 13 MHz
- Phase noise @ 1MHz offset: -90 dBc/Hz
- Lock in time < 100 μs
- Refined models are used for mixed-level simulations

Simulation result

Lock-in process

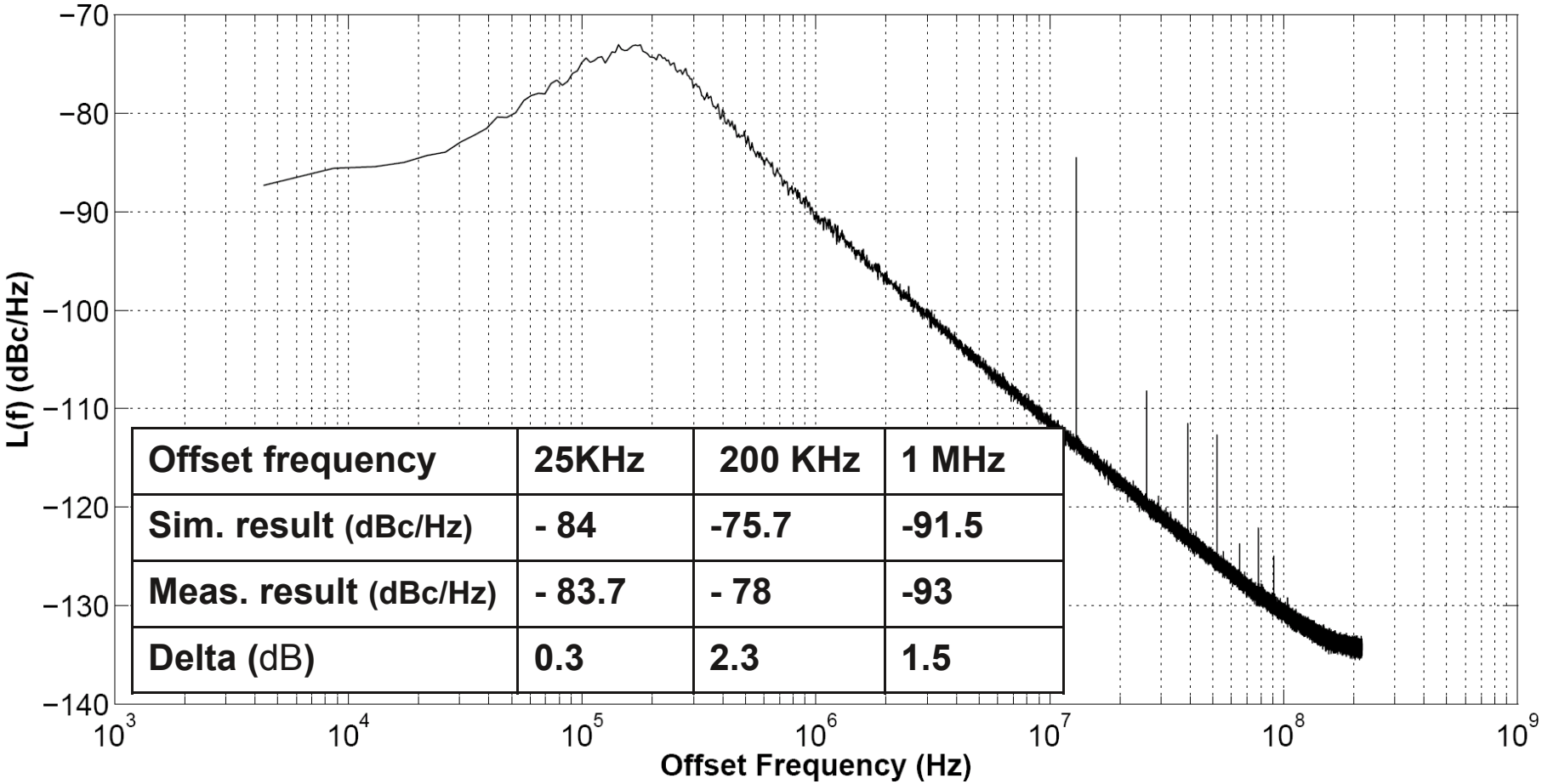


Longer locking time due to PFD delay

- For 100 μs, the simulation requires less than 1 min.

Simulation result

Overall phase noise and spurious emission evaluation



Offset frequency	25KHz	200 KHz	1 MHz
Sim. result (dBc/Hz)	- 84	-75.7	-91.5
Meas. result (dBc/Hz)	- 83.7	- 78	-93
Delta (dB)	0.3	2.3	1.5

- Simulation and evaluation take 2h for 26M oscillations

Conclusions

- Successfully modeled the PLL using wreal data type
- The models are more efficient than phase model for noise and spurious emission estimation
- Transistor level block can be verified using mixed level simulation
- The application example shows feasibility of integrating the method into industrial design flow

Thank you

Silicon measurement results

PNOISE_FC_25K_433_2_CC

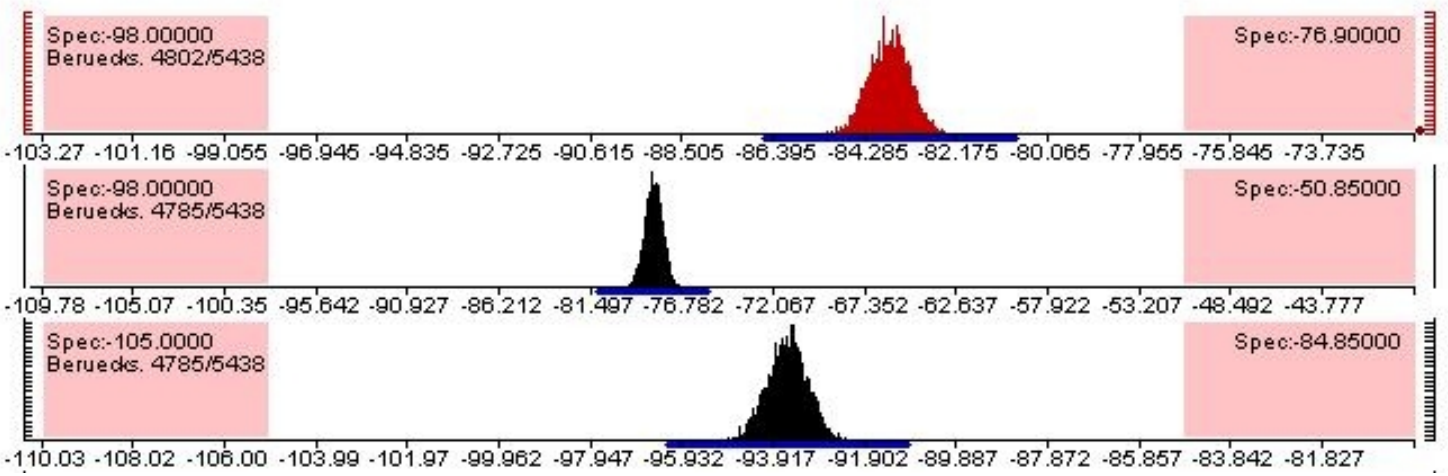
Mittelwert: -83.6763
Standardabw.: 1.55544
cpk: 1.45217
6 Sigma: -86.6289 -80.8511

PNOISE_FC_200K_433_2_CC

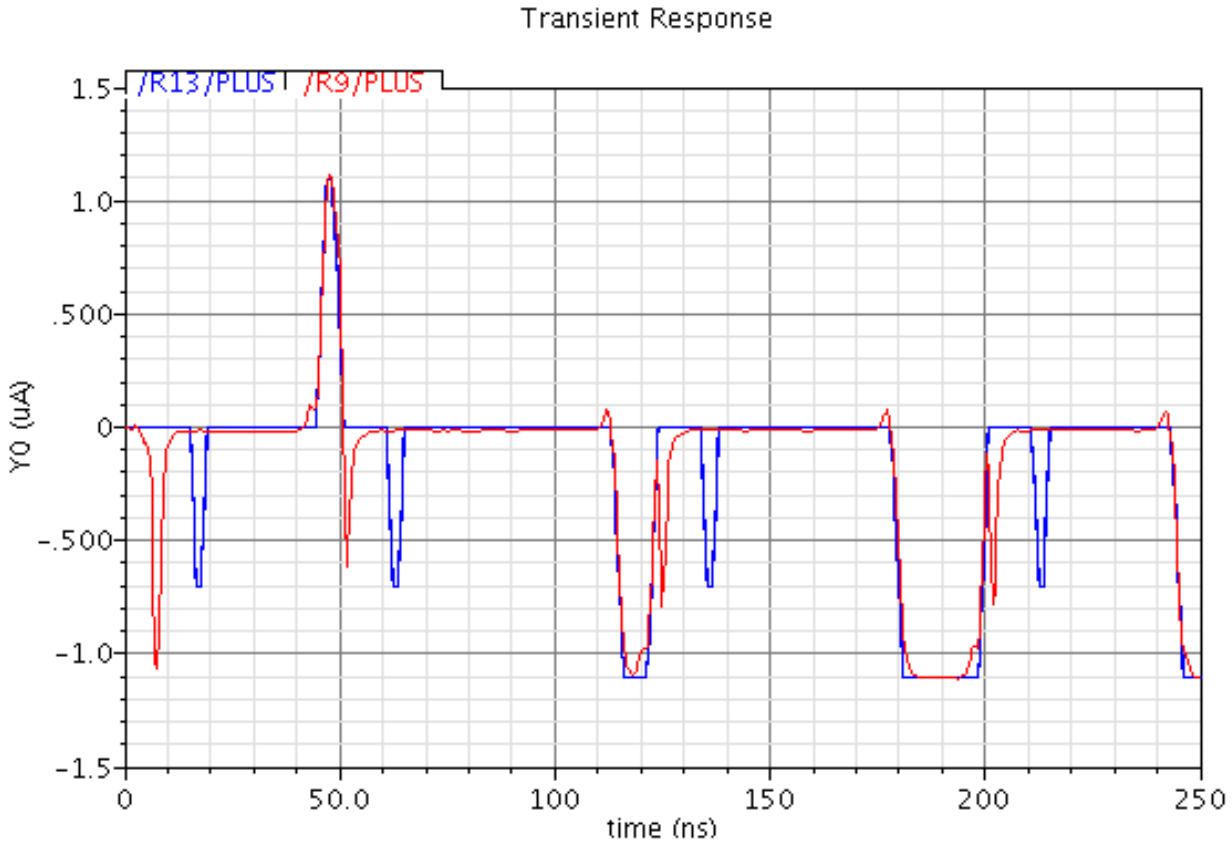
Mittelwert: -78.3158
Standardabw.: 0.486043
cpk: 13.4997
6 Sigma: -81.1644 -75.4756

PNOISE_FC_1M_433_2_CC

Mittelwert: -93.6526
Standardabw.: 0.455353
cpk: 6.44379
6 Sigma: -96.3067 -90.9733



Charge pump current



Timescale

Timescale:

100ps:

10ps:

1ps:

100fs:

10fs:

1fs: