Phase Noise Simulation and Modeling of ADPLL by SystemVerilog

Tingjun Wen
Integrated Device Technology
Tad Kwasniewski
Carleton University

Sept 2008

Motivations

- Integrate the phase noise behavioral simulation with the circuit level design
- Find a better random number generator with good statistical properties to make the phase noise simulation more accurate
- Make the phase noise simulation faster
- Use the phase noise behavioral simulation to explore new ADPLL architectures with lower phase noise

Behavioral Simulation Languages

LANGUAGE	EVENT-DRIVEN
Matlab/Simulink	Function calls
Verilog/VHDL	Intrinsic
Verilog-AMS/VHDL-AMS	Intrinsic
SystemVerilog	Intrinsic
SystemC/AMS	C++ Class Library

SystemVerilog + C Language

- Direct programming interface (DPI)
  - Can call any standard C functions directly
  - Exporting Verilog tasks and functions to C
- Example

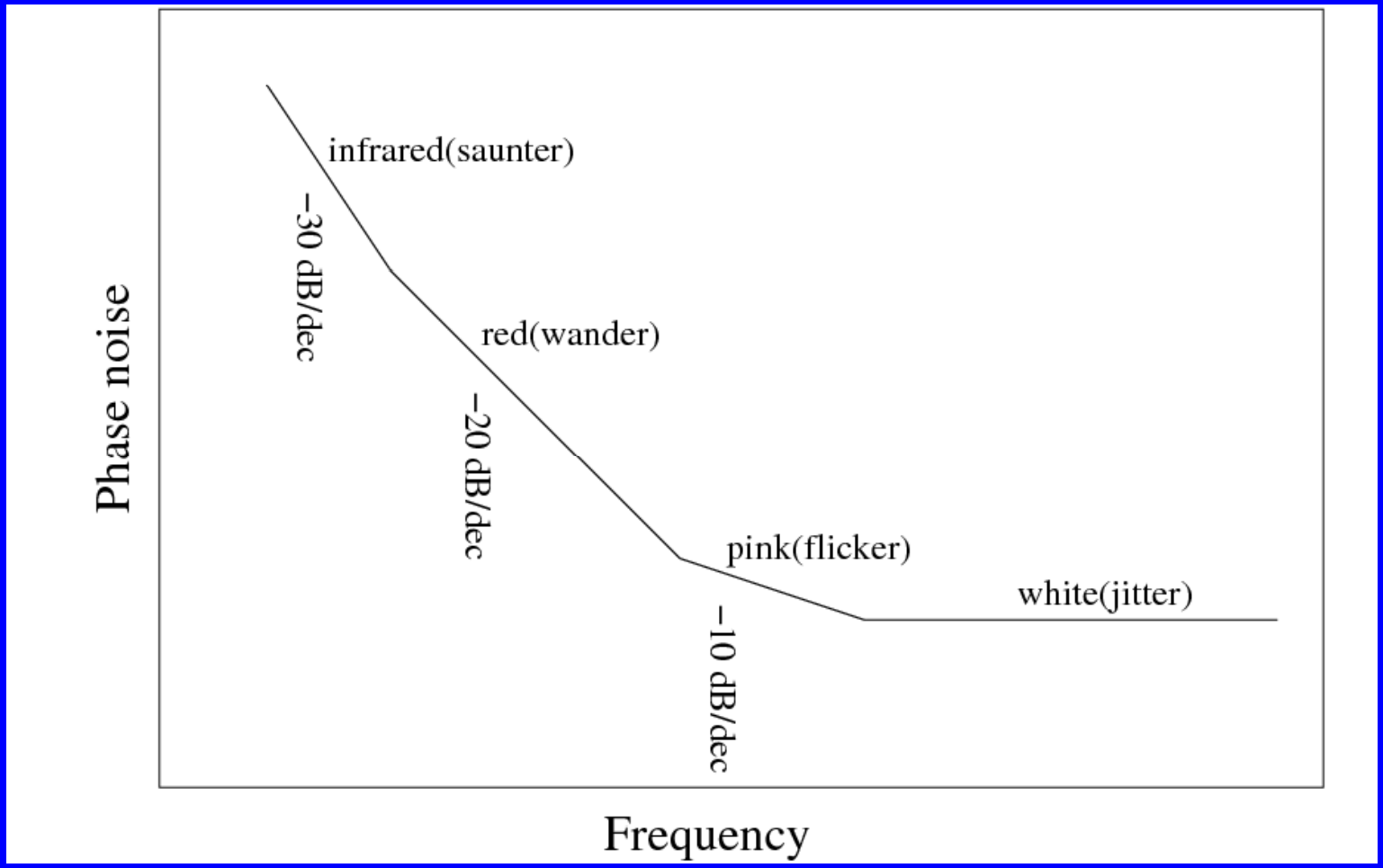
noise.c:

```
double flicker(double stddev, double *pflicker);
```

noise.v:

```
import "DPI-C" function real flicker(input real stddev,
                                     output real pflicker);
always @(posedge clock) begin
    period = period + flicker(flicker_stddev, pflicker);
end
```

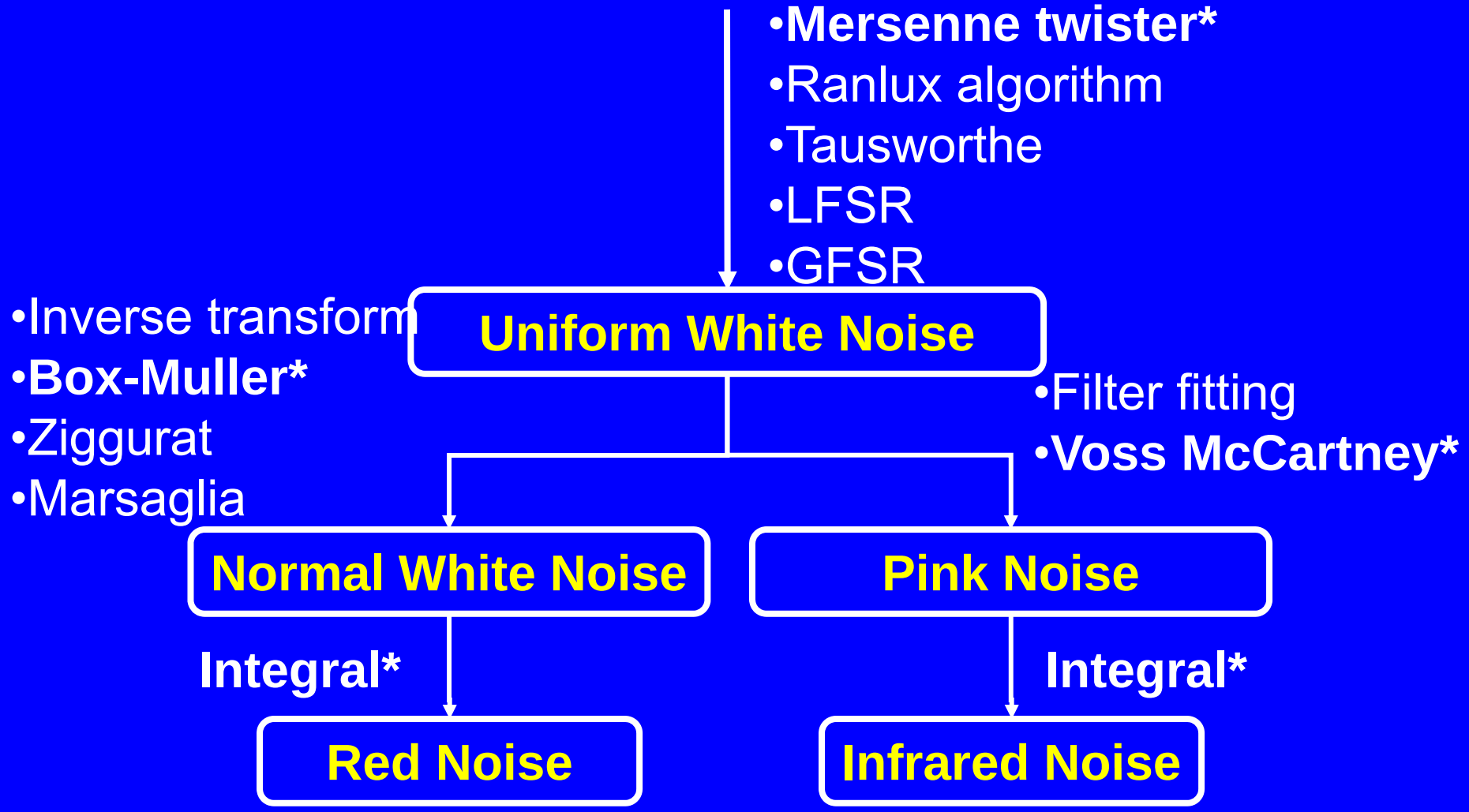
Noise Terminologies



Spectrum vs Distribution

FFT Spectrum	Distribution
White (0dB/dec)	Uniform
White (0dB/dec)	Normal
Pink (-10dB/dec)	?
Red (-20dB/dec)	?
Infrared (-30dB/dec)	?

Noise Generator Hierarchy



* Used by this paper

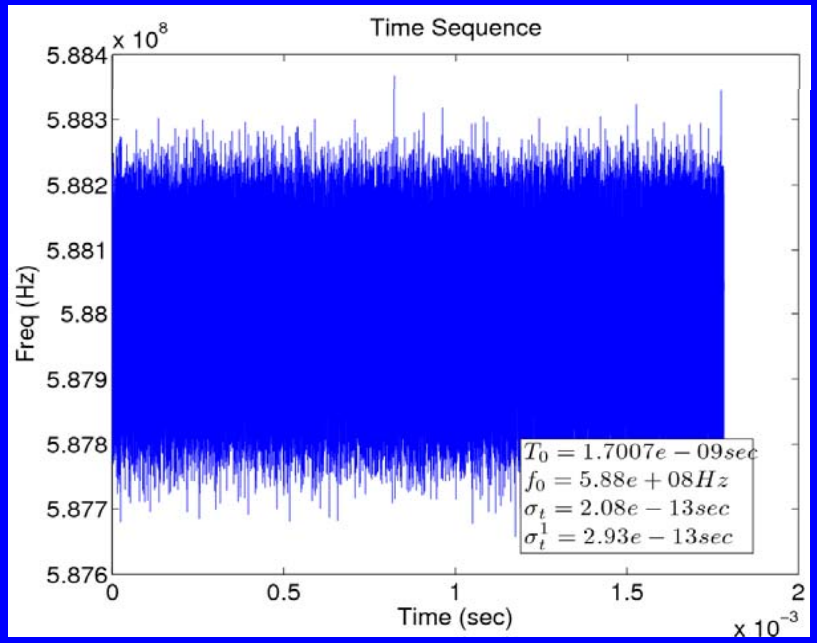
Uniform White Noise Generator

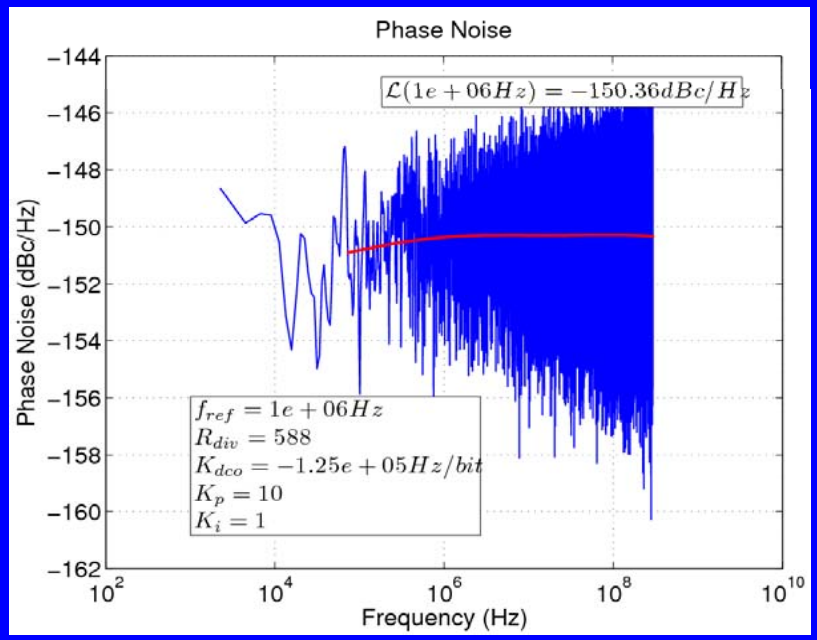
- Mersenne twister generator
- have a period of (2^19937−1)
- there is negligible serial correlation
- Have good statistical randomness
- Source code available

Normal White Noise Generator

- Central limit theorem
- Box-Muller algorithm
- Generate Gaussian random variable from uniform distribution
- Simple to implement

Normal White Noise Generator



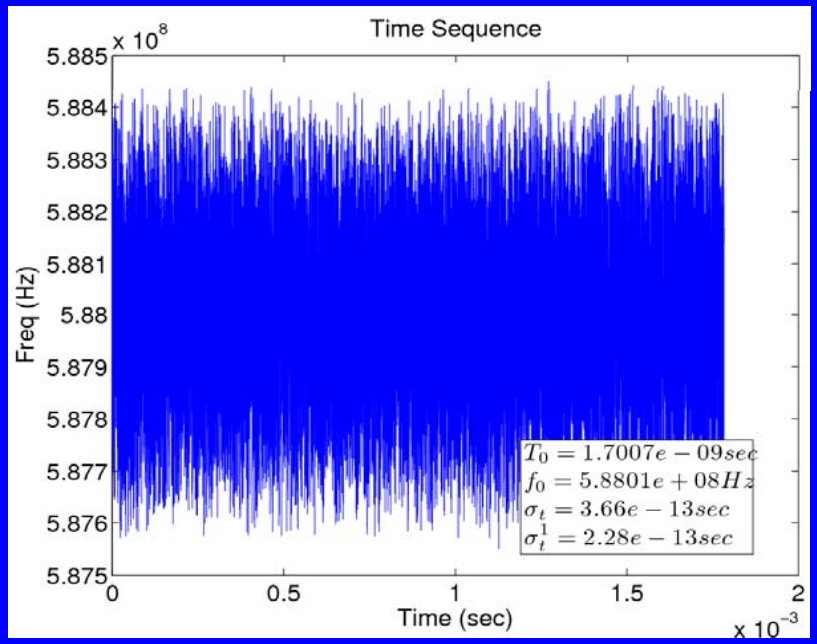


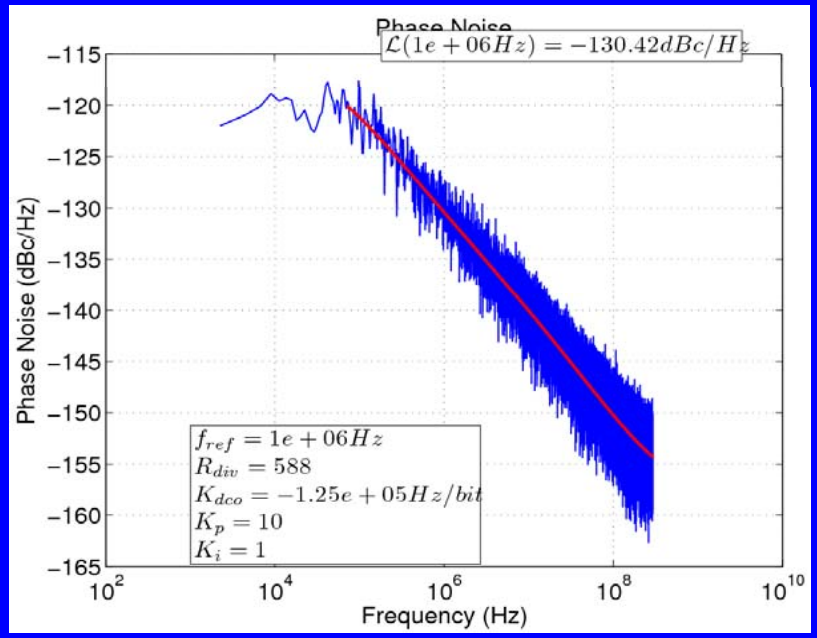
Correlated Pink Noise Generator*

- Stochastic Voss-McCartney variant
- Popular in the music industry
- Each data needs no more than 2 random numbers generation operations
- It is very efficient

*http://home.earthlink.net/%7Eltrammell/tech/newpink.htm

Correlated Pink Noise Generator*





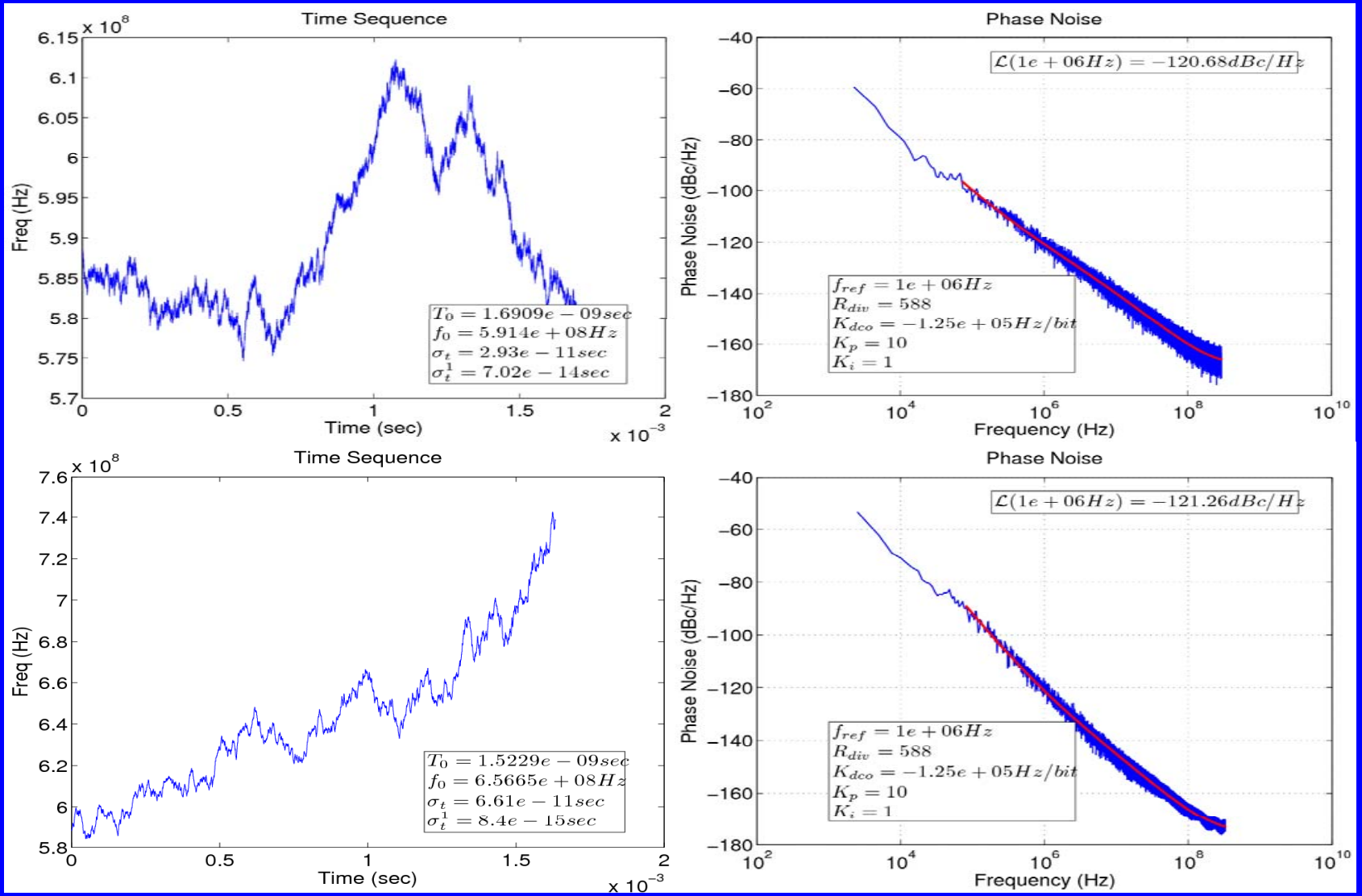
Higher Order Noise Generators

- Similar to the noise shaping theory developed from Sigma-Delta modulator

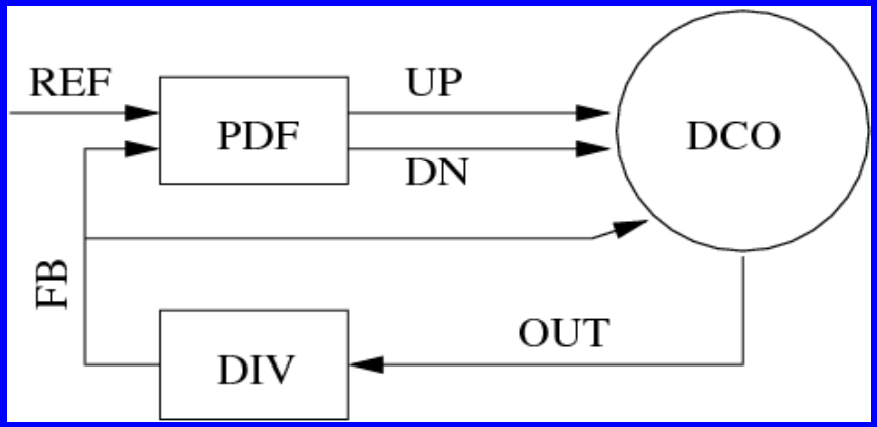
  Δ Differentiation → (+20dB/dec)
  Σ Integration → (- 20dB/dec)

- Σ White noise → Red (-20dB/dec)
- Σ Pink noise → Infrared (-30dB/dec)

Higher Order Noise Generators

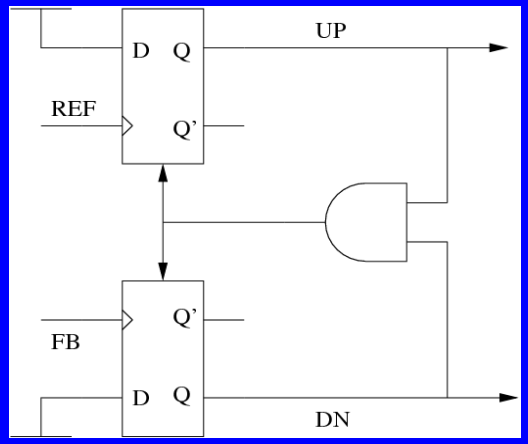


ADPLL Architecture



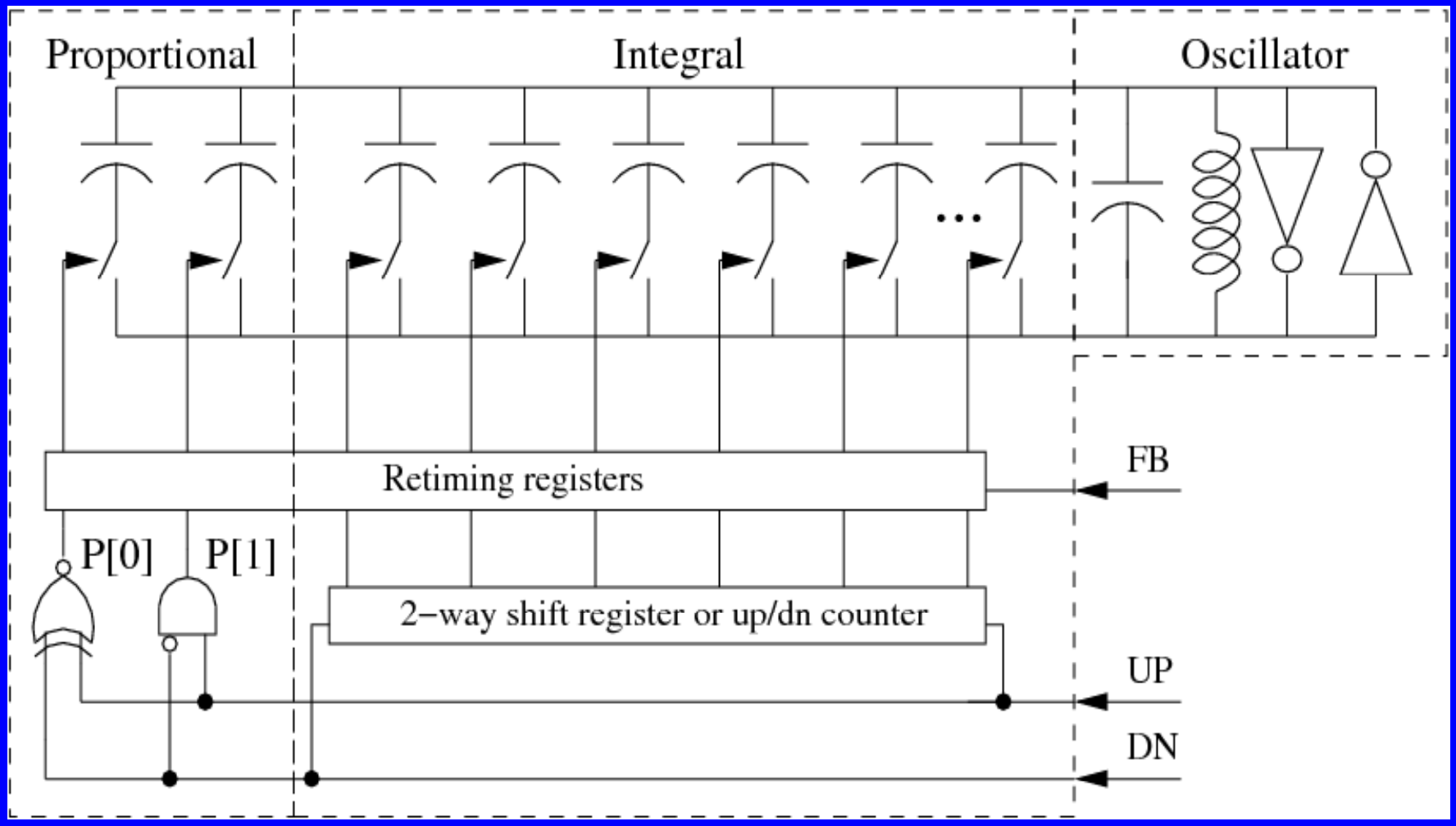

```
module adpll (fref, M, fdiv, fout, rst_n);
  input fref, rst_n, fdiv;
  input [`div_width-1:0] M;
  output fout;
  pfd  PFD (.fref(fref), .fdiv(fdiv), .up(up), .dn(dn), .rst_n(rst_n));
  dco DCO (.up(up), .dn(dn), .fdiv(fdiv), .fout(fout));
  div  DIV (.fin(fout1), .M(M), .fdiv(fdiv), .rst_n(rst_n));
endmodule
```

Phase Frequency Detector




```
`include "adpll.vh"
module pfd (fref, fdiv, up, dn, rstn);
  output up, dn;
  input fref, fdiv, rstn;
  reg up, dn;
  wire pfdrst;
  always @( posedge fref or posedge pfdrst) begin
    if (pfdrst) up <= 0; else up <= 1;
  end
  always @( posedge fdiv or posedge pfdrst) begin
    if (pfdrst) dn <= 0; else dn <= 1;
  end
  assign pfdrst = ~rstn | (up & dn);
endmodule
```

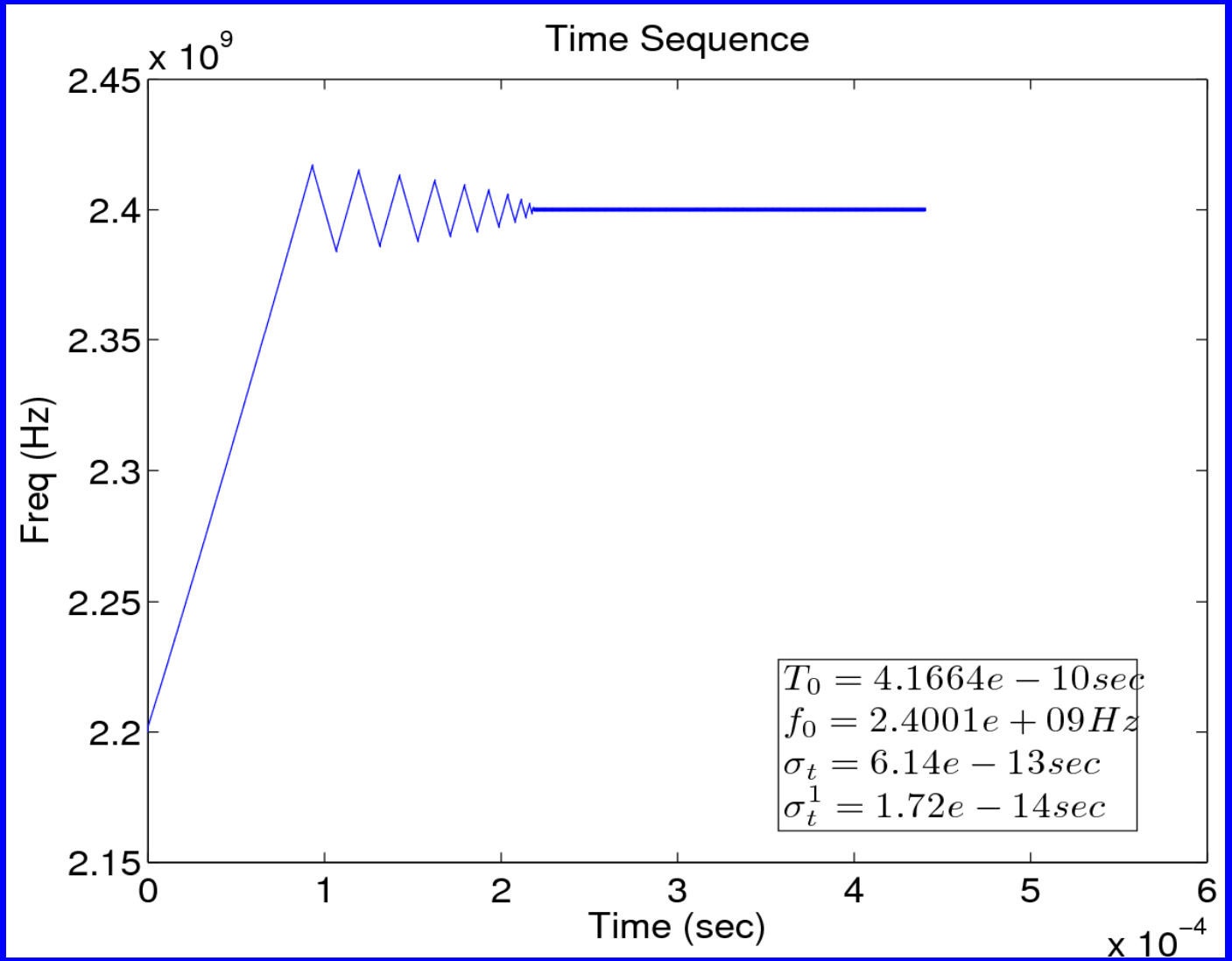
DCO with Integrated LPF



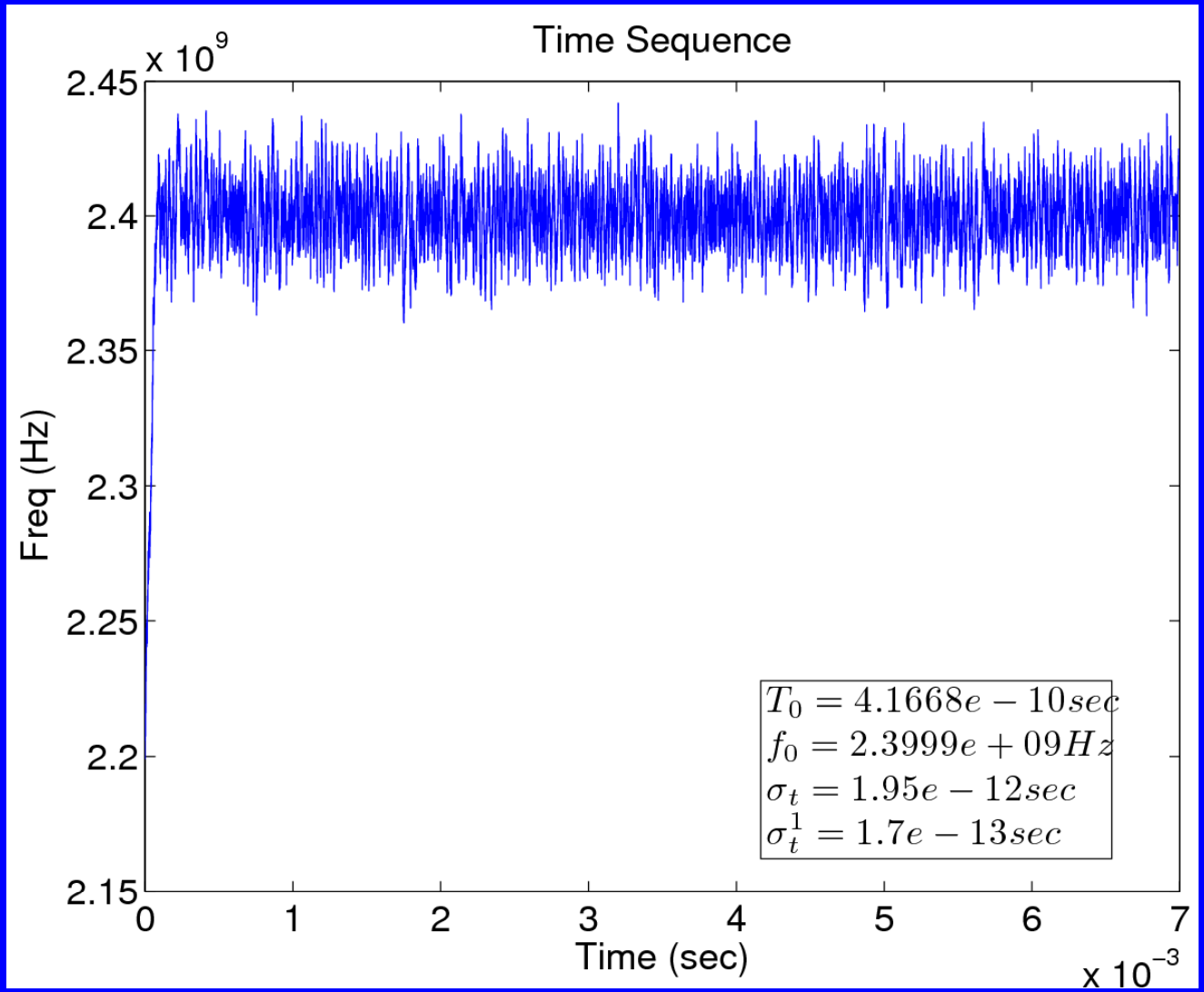
DCO with Integrated LPF

```
always @( up or dn ) begin
  case ({ up , dn })
   2 'b01 : I = I − 1 ;  2 'b10 : I = I + 1 ; default : ;
  endcase
  P [ 0 ] = up ^ ~ dn ;  P [ 1 ] = up & ~ dn ;
end
always @( posedge fdiv )  Ctrl <= ( `Kp  P ) + ( `Ki  I ) ;
always begin
  period = 1.0 / ( ` f0dco + `Kdco  Ctrl );
  period = period + flicker (stddev , pflicker ) ;
  #( period / 1e −12/2) fout = ~ fout ;
end
```

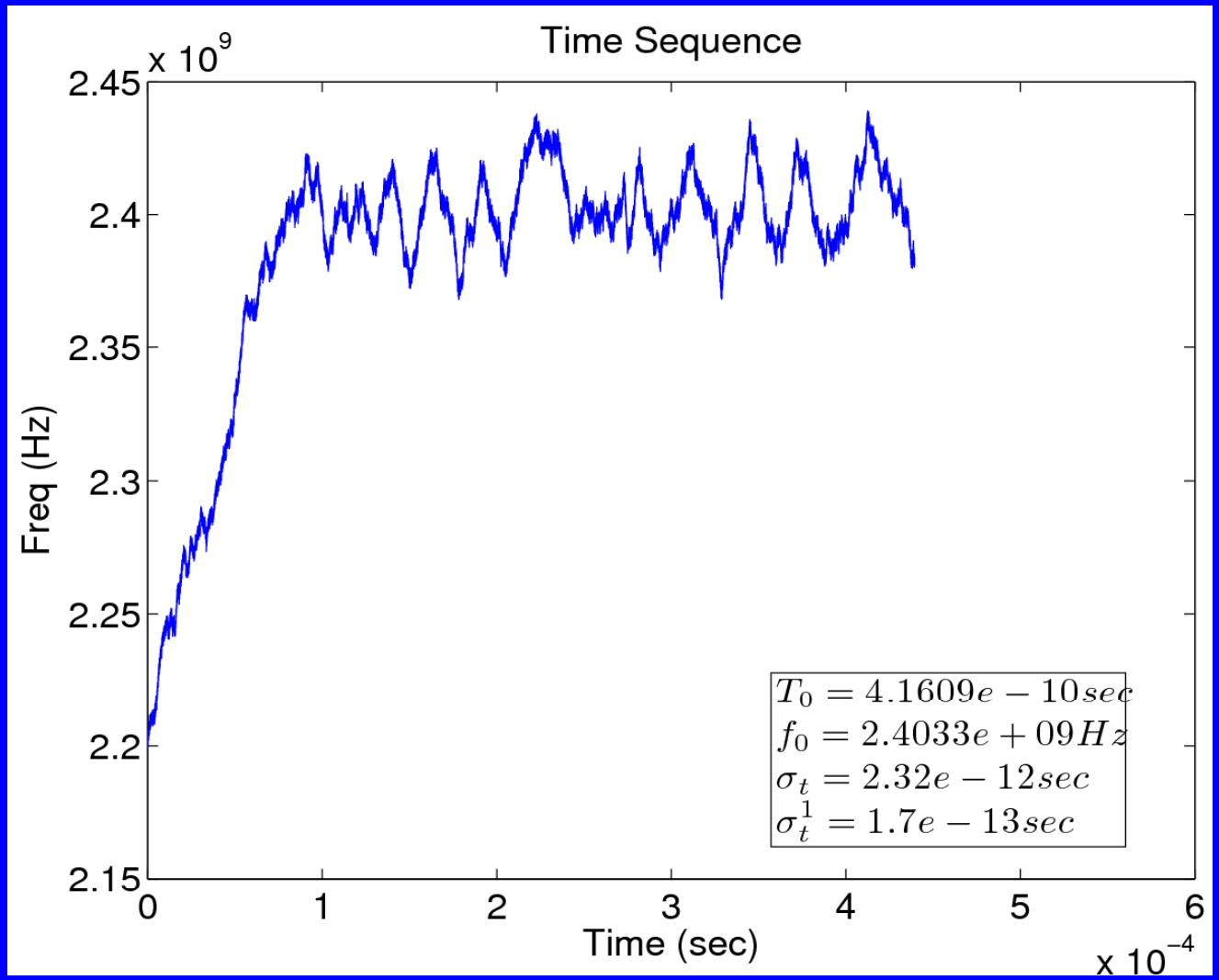
Step Response without Noise



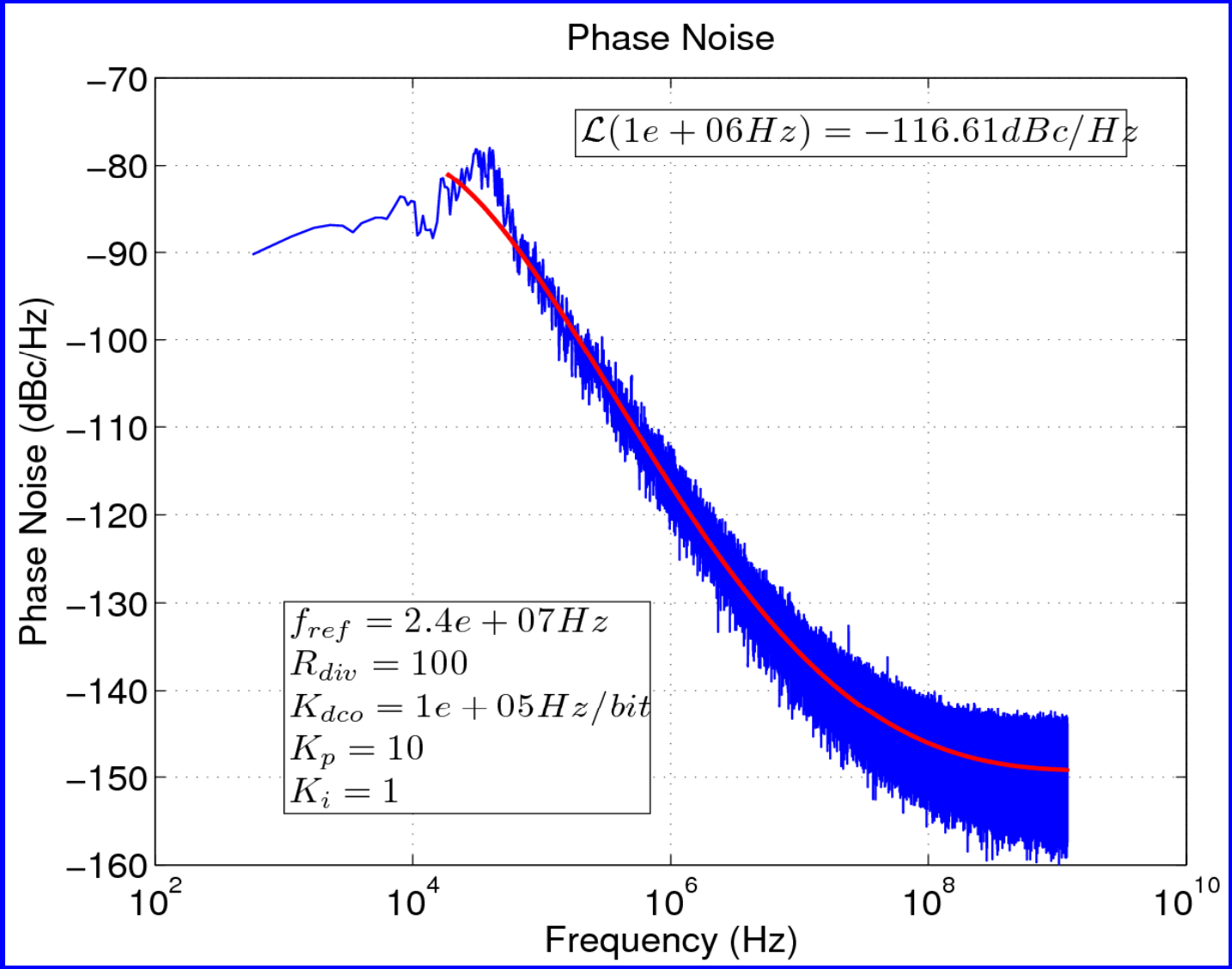
Step Response with Noise



Step Response with Noise



Simulated Phase Noise



Conclusions

- Mersenne-Twister → Uniform white noise
- Box-Muller → Normal white noise
- Stochastic V-M → 1/f pink noise
- Integral operation → higher order noises
- SystemVerilog DPI to integrate with the noise generator functions written in C language
- Phase noise simulation techniques successfully applied to a new ADPLL architecture
- Event-driven runs less than 1 minute while Spice runs by hours or by days
- Design parameters extracted for future transistor level circuit design

References

- [1] K. Kundert. (2006, Aug.) Predicting the phase noise and jitter of pll based frequency synthesizers. [Online]. Available:http://www.designersguide.org/Analysis/PLLnoise+jitter.pdf
- [2] J. Zhuang, Q. Du, and T. Kwasniewski, “Event-driven modeling and simulation of an digital PLL,” *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International*, pp. 67–72, Sept. 2006.
- [3] S. Huang, H. Ma, and Z. Wang, “Modeling and simulation to the design of fractional-n frequency synthesizer,” in *DATE ’07: Proceedings of the conference on Design, automation and test in Europe*. San Jose, CA, USA: EDA Consortium, 2007, pp. 291–296.
- [4] R. Staszewski, C. Fernando, and P. Balsara, “Event-driven simulation and modeling of phase noise of an rf oscillator,” *Circuits and Systems I: Regular Papers, IEEE Transactions on [Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on]*, vol. 52, no. 4, pp. 723–733, April 2005.
- [5] M. Matsumoto and T. Nishimura, “Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator,” *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, 1998.
- [6] L. Tammell. (2006, Jan.) Improvements in the correlated pink noise generator evaluation. [Online]. Available: http://home.earthlink.net/%7Eltrammell/tech/newpink.htm
- [7] J. Zhuang, Q. Du, and T. Kwasniewski, “A 3.3 ghz lc-based digitally controlled oscillator with 5khz frequency resolution,” *Solid-State Circuits Conference, 2007. ASSCC ’07. IEEE Asian*, pp. 428–431, Nov. 2007.