Python Framework for Design and Analysis of Integer-N ADPLLs

Specialization Project Progress - 11th Week

Cole Nielsen
Department of Electronic Systems, NTNU

8 November 2019 (Calendar week 45)

Timeline

Week	Dates	Tasks	Outcomes
36	2.9 - 8.9	Review PLL Design	Refreshed Knowledge
37	9.9 - 15.9	Modeling/simulation (set up)	–
38	16.9 - 22.9	Modeling/simulation	TDC/DCO Requirements
39	23.9 - 29.9	Modeling/simulation	Loop Filter/Digital Algorithms
40	30.9 - 6.10	Modeling/simulation	Loop filter, DCO, TDC, calibration
41	7.10 - 13.10	Circuit Research	DCO/Divider topologies
42	14.10 - 20.10	Circuit Research	TDC/other topologies
43	21.10 - 27.10	Spur analysis, filter automation	
44	28.10 - 3.11	Filter automation, SNR estimation	
45	4.11 - 10.11	Variation analysis, flicker noise	Histograms/yield estimates
46	11.11 - 17.11	Real DCO sensitivity, TDC/divider jitter	Simlate ring-DCO in Virtuoso
47	18.11 - 24.11	PLL + Radio simulation	BER estimate
48	25.11 - 1.12	Agglomerate into cohesive framework	(I have an Exam on 30.11)
49	2.12 - 8.12	Finish framework, report writing	
50	9.12 - 15.12	Report writing	Complete before 15.12

Legend: **Done** **Current** **Revised**

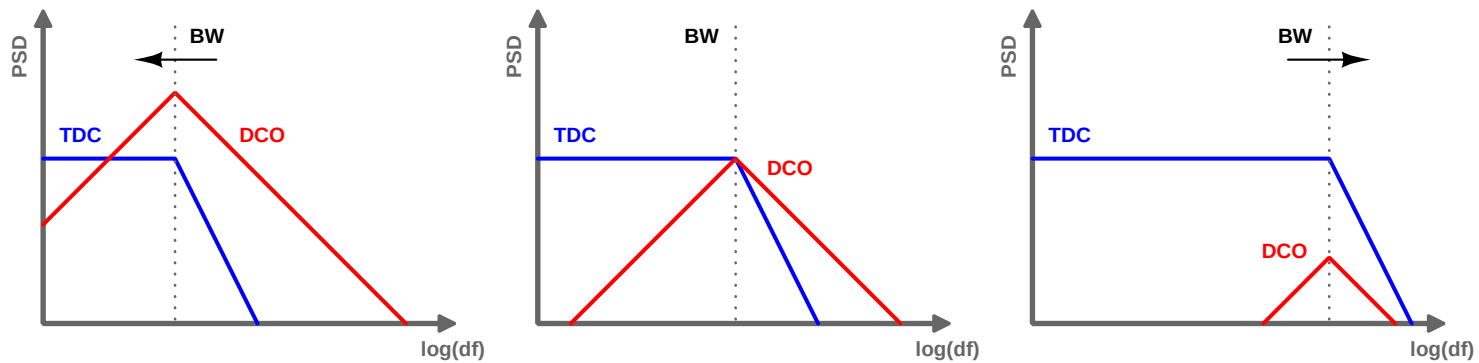
Timeline Tasks

This week

- **Loop filter optimization:**
  - Devised new method to optimize loop filter based on lock-time and total phase noise power estimates of loop filter.
- **Variation/sweep analysis**:
  - Implemented engine for parametric sweep and variation analysis through Monte-Carlo sampling.
    - Can easily vary an arbitrary number of parameters.
    - Utilize parallelization across CPU cores for speed.
  - Yield estimate based on statistical fits of data, can also emulate corner cases.

Filter automation

So, what is truly optimal???



— Optimal phase noise occurs when TDC/DCO contributions are approximately equal.
  - However this may not occur at a wide enough bandwidth for desired lock time.

— Therefore optimum is when:
  - Total phase noise is minimized *while* meeting constraint for maximum lock time.
  - May have to accept worse phase noise to achieve lock time.

Filter automation

New optimization approach

— Constrained by a maximum settling time $t_{s,max}$, try to minimize phase noise.

- Utilize quick estimates in optimizer of **(a)** integrated phase noise power and **(b)** settling time of PLL for an arbitrary closed loop response G(f).

— **Phase noise estimate**

- Use same models as before for phase noise of PLL, where S_{TDC} and S_{DCO} are the intrinsic phase noise PSD respectively for the TDC and DCO. N is the PLL divider modulus. The total PLL phase noise $S_{\Sigma}(f)$ is:

$$S_{\Sigma}(f) = f_{clk} \cdot |2\pi N \cdot G(f)|^2 S_{TDC} + |1 - G(f)|^2 S_{DCO} \tag{1}$$

- Given a bandwidth of interest Δf (i.e. baseband bandwidth), the total integrated phase noise power is:

$$P_{\phi noise} = 2 \int_0^{\Delta f} S_{\Sigma}(f) df \tag{2}$$

Filter automation

New optimization approach

— Settling time (PLL lock time) estimate

- Easy to translate G(f) to state space representation with Python (`scipy.special`):

$$G(f) = \frac{\Sigma_0^M b_m s^m}{\Sigma_0^N a_n s^n} \quad \rightarrow \quad s\mathbf{Y}(s) = \mathbf{A}\mathbf{Y}(s) + \mathbf{B}X(s) \tag{3}$$

- The state transition matrix is then $\mathbf{\Phi} = (s\mathbf{I} - A)^{-1}$. The eigenvalues and eignvectors of $\mathbf{\Phi}$ are then $\{\lambda_1, \ldots, \lambda_N\}$ and $\{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ respectively. The step response of G(f) takes the form:

$$\mathbf{y}(t) = \mathbf{v_1} e^{\lambda_1 t} + \ldots + \mathbf{v_k} e^{\lambda_k t}, \quad \mathbf{y}(t) = [y(t) \;\; y^{'}(t) \;\; \ldots \;\; y^{(k)}(t)]^T \tag{4}$$

- The λ_k with the smallest real component then governs the long term settling of y(t). Assuming one dominant pole, the time constant of G(f) can be estimated as $\Re(\lambda_k)^{-1}$. Settling time t_s can be considered as the interval required for the signal to drop within a tolerance band $\pm\delta_{tol}$y(∞) about the final value y(∞). Thus:

$$t_s = \tau \ln(\delta_{tol}) = \frac{\ln(\delta_{tol})}{\min(|\Re(\{\lambda_1, \ldots, \lambda_k\})|)} \tag{5}$$

- This turns out to be fairly accurate versus simulation.

Filter automation

New optimization approach

— Based on the open loop PLL transfer function A(f), optimization for the closed loop G(f) is done using two levels of optimization.

$$A(f) = \frac{K}{s^2} \frac{(s/\omega_z + 1)}{(s/\omega_p + 1)} \tag{6}$$

— **Level A - Minimize phase noise for fixed settling time.**

- Minimize phase noise while maintaining fixed settling time = t_s.
- **Solution:** use two steps per interation until satisfactorily converged:
  1. Minimize phase noise using pole/zeros locations (gradient descent).
  2. Tune K such that settling time = t_s (golden section search).

— **Level B - Optimize settling time given constraints.**

- Use golden section search to find optimal constrained settling time that has minimal phase noise using method from level A as cost function.

Variation analysis

Monte-Carlo sampling

— Implemented analysis of variaton using Monte-Carlo (MC) sampling.

— Simulation configured by dictionary with PLL/simulation parameters.
  - Implemented MC engine allows for variation any of these parameters, or any combination of the parameters.

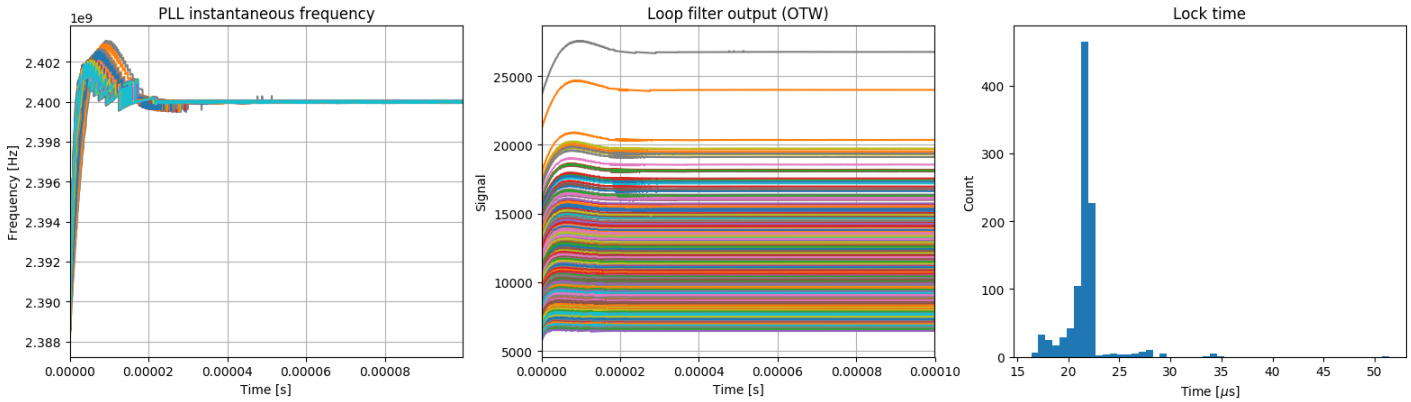
— Implemented concurent execution of simulations with `multiprocessing` package.
  - Can run 8 simulations simultaneously on my machine
  - Run of 1000 simulations for Monte-Carlo analysis of settling time takes ca. 20s.

Variation analysis

Monte-Carlo sampling

— **Example 1 - KDCO variation**

- f=2.4 GHz, f_{ref}=16 MHz, 150 TDC steps, 20ps BBPD resolution, KDCO = 10 kHz/LSB, σ_{KDCO} = 2 kHz, initial f_{error} = 12 MHz.
- Settling/lock tolerance = $\pm 100kHz$, samples = 1000



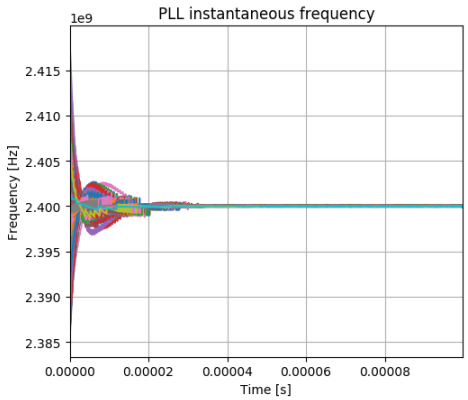
- Mean = 34.2 μs
- Non-gaussian distribution

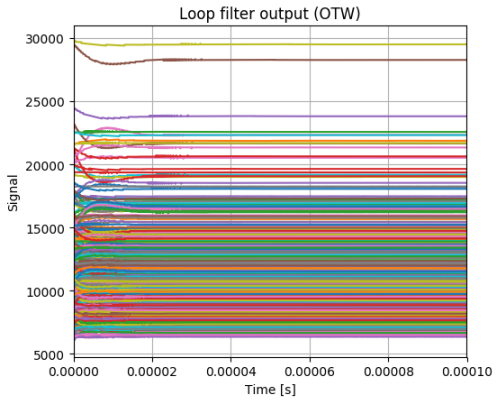
Variation analysis

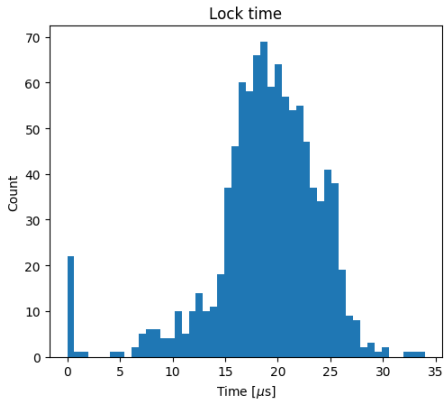
Monte-Carlo sampling

— **Example 1 - KDCO variation + large initial frequency variation**

- f=2.4 GHz, f_{ref}=16 MHz, 150 TDC steps, 20ps BBPD resolution, KDCO = 10 kHz/LSB, σ_{KDCO} = 2 kHz, initial frequency error $\sigma_{f-error}$ = 5 MHz.
- Settling/lock tolerance = $\pm 100kHz$, samples = 1000







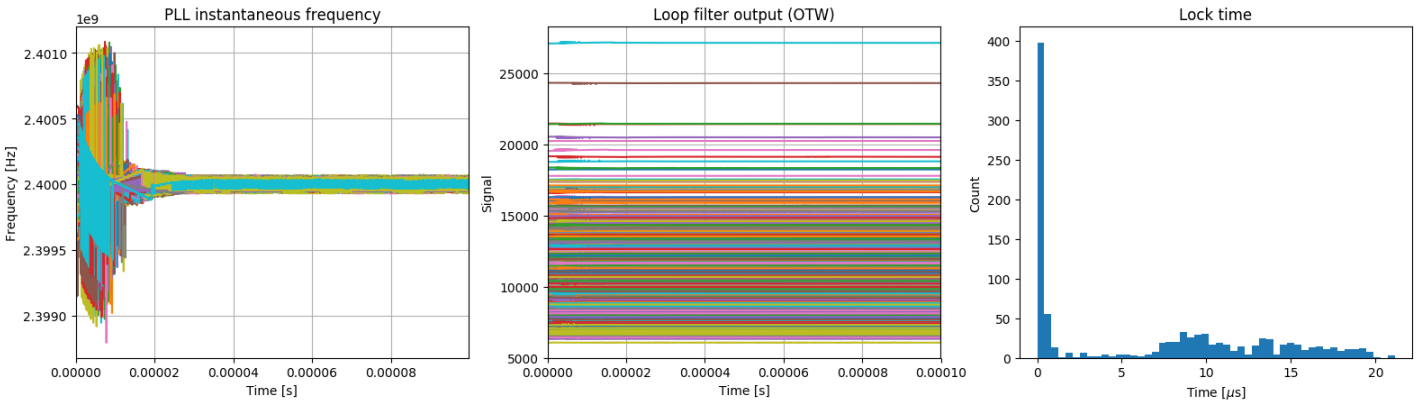

```
Monte Carlo simulation, Samples = 1000
        Mean = 1.905169E-05
        Stdev = 5.043830E-06
        Tsettle, 99% CI = 2.825000E-05
```

Variation analysis

Monte-Carlo sampling

— **Example 1 - KDCO variation + small initial frequency variation**

- f=2.4 GHz, f_{ref}=16 MHz, 150 TDC steps, 20ps BBPD resolution, KDCO = 10 kHz/LSB, σ_{KDCO} = 2 kHz, initial frequency error $\sigma_{f-error}$ = 0.25 MHz.
- Settling/lock tolerance = $\pm 100kHz$, samples = 1000




```
Monte Carlo simulation, Samples = 1000
        Mean = 6.436750E-06
        Stdev = 6.549602E-06
        Tsettle, 99% CI = 1.943750E-05
```

Variation analysis

Monte-Carlo sampling

— It should be noted that the mean lock time from the large initial frequency deviation simulation (19.1 μs) to the small frequency deviation simulation (6.4 μs) decreased.

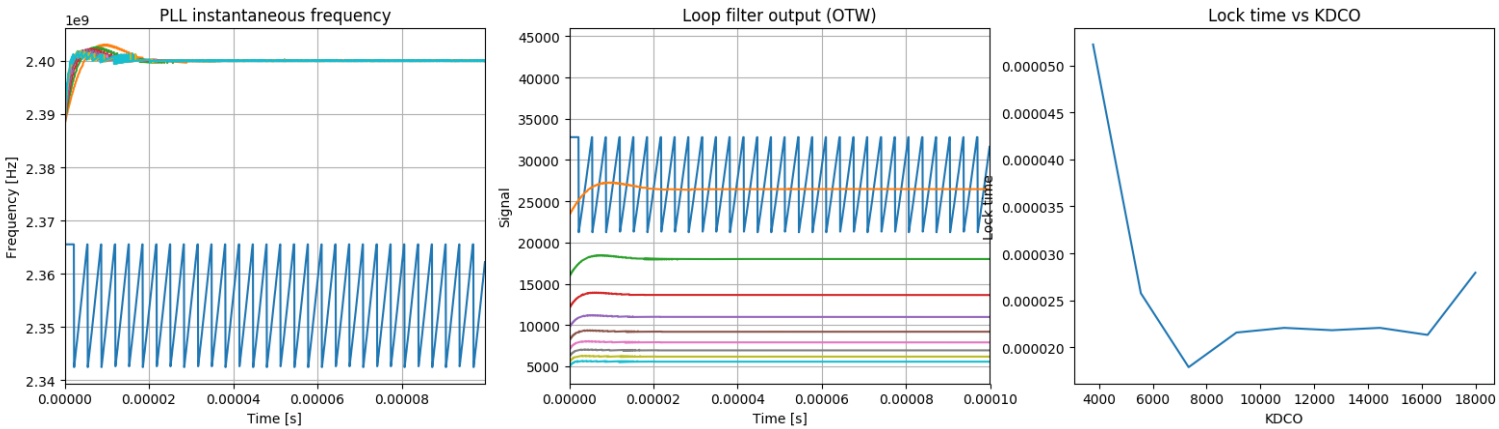
— This implies the lock-time (i.e. re-lock time) improvement with the ADPLL for starting with a small frequency error desired for use with a WuRx is realized.

- Further improvements in re-lock time should be realizable through optimization.
- Should enable power reduction in WuRx versus analog PLL due to this lock time improvement.

Variation analysis

Parameter sweeps

- Can sweep arbitiary PLL/simulation parameters in similar fashion to MC simulation.
- f=2.4 GHz, f_{ref}=16 MHz, 150 TDC steps, 20ps BBPD resolution, KDCO = [2,18] kHz/LSB, initial frequency error = 12 MHz.
- Settling/lock tolerance = $\pm 100kHz$, samples = 11



Specification (unchanged)

System Performance Targets

Parameter	Value	Unit	Notes
Frequency	2.4-2.4835	GHz	2.4G ISM Band
Ref. frequency	16	MHz	Yields 6 channels
Power	$\leq$ 100	μW	
FSK BER	$\leq$ 1e-2		2FSK with f_{dev}=$\pm$250 KHz
Initial Lock Time	$\leq$ 50	μs	Upon cold start
Re-lock Time	$\leq$ 5	μs	Coming out of standby
Bandwidth	50	kHz	(nominally), tunable

Additionally: PLL output should support IQ sampling at LO frequency.

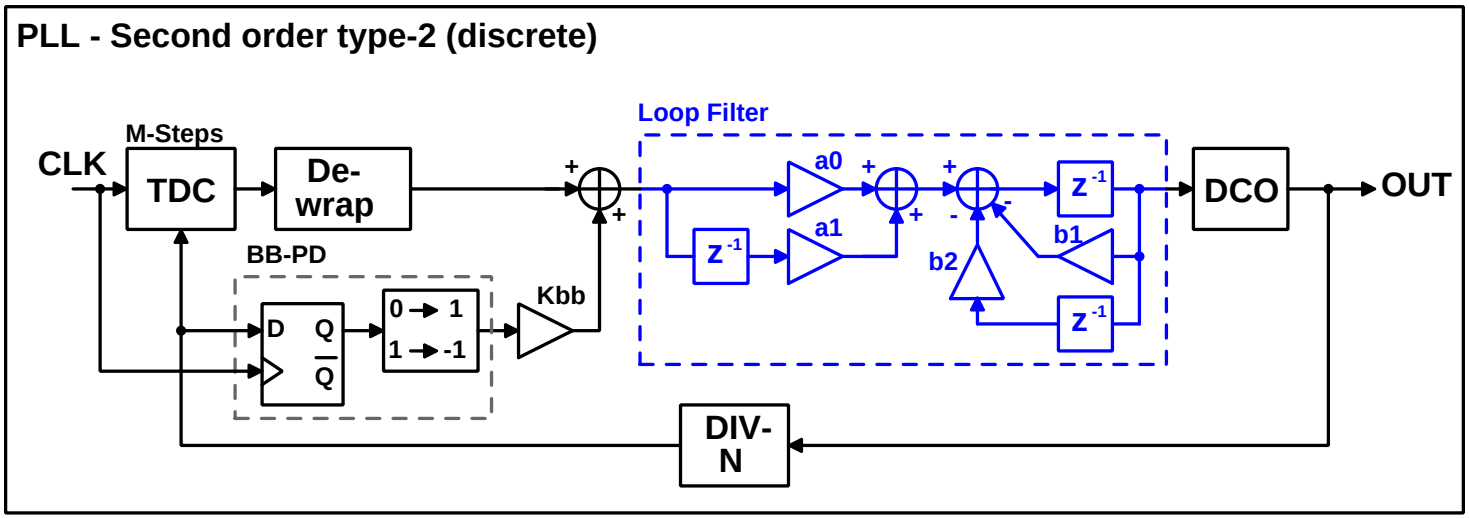
Specification (unchanged)

PLL Component Performance Targets

Parameter	Value	Unit	Notes
DCO LSB Resolution	≤ 50	kHz	Determined from quantization noise.
DCO DNL	< 1	LSB	Ensures monotonicity
TDC Resolution	0.95	ns	
TDC Resolution (bits)	6	bits	

Architecture (updated)

Block Diagram



Power Targets

DCO	TDC	Divider	Other	SUM
70 μW	20 μW	10 μW	$<<$ 1 μW	100 μW

Project Phases

Autumn 2019

- System modeling and simulation.
  - Learn PLL theory in detail
  - Evaluate feasability of PLL architectures (counter, TDC-based)
  - Determine requirements for TDC/DCO/Divider/logic (bits of resolution, accuracy etc) to meet PLL performance specifications.
  - Determine digital logic for loop filter, validate stability and lock time performance.
- Research ultra-low power circuit topologies to implement system components that will meet determined requirements.
- Translate component-level specifications into schematic-level circuit designs.
  - Try, fail, try again until functional at schematic level.
    - I expect the TDC to be difficult.

Project Phases (continued)

Spring 2020

- Finalize schematic-level design.
- Establish thorough tests for PLL performance (automated?) to help in layout.
- Layout of PLL.
  - Design iteration until design specs met.
  - Probably very time consuming.
- Full characterization/validation of design performance.
  - Comprehensive Corners/Monte-Carlo testing (time consuming??)
  - More design iteration if new issues crop up...
- Thesis paper writing.

References

[1] "Ultra-Low Power Wake-Up Receivers for Wireless Sensor Networks", N. Pletcher, J.M Rabaey, 2008.

`http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-59.html`