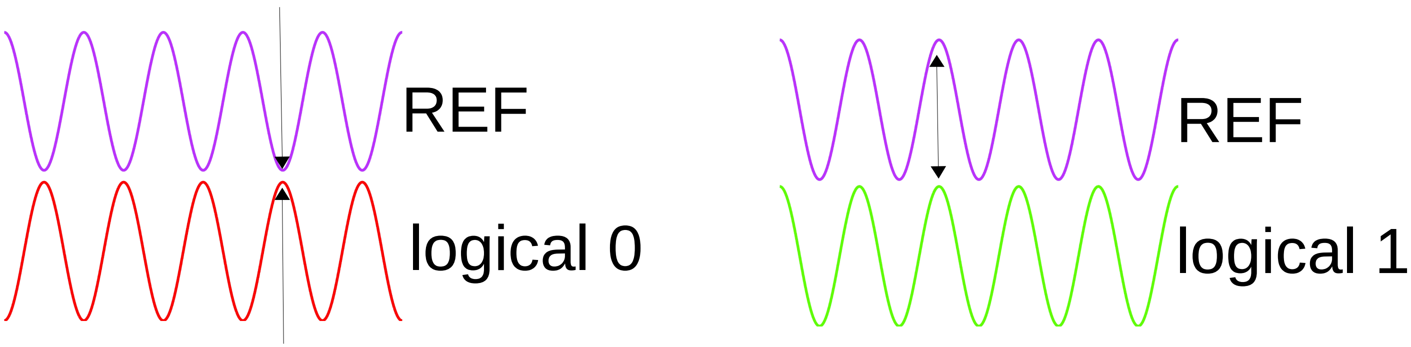
Design Tools for Oscillator-Based Computing Systems

Tianshi Wang, Jaijeet Roychowdhury

University of California, Berkeley

Encoding Bits Using Phase



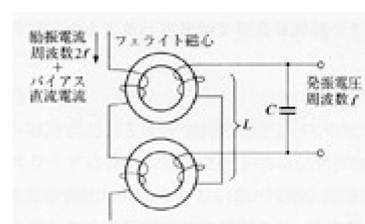
- Can you use this for computing?
- Even if you can: what is the advantage?

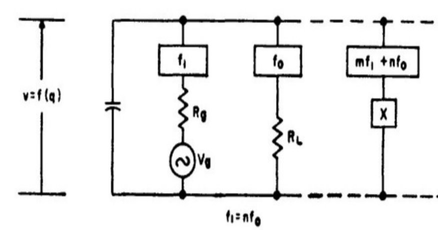
noise immunity ← loose analogy: PM/FM vs. AM in radio

Phase Logic Computer: Eiichi Goto, John von Neumann, 1950s and 60s

- “cheap and reliable”
  - “widely used in Japan”
- not easy to miniaturise
  - inductors, iron cores
  - transistors/ICs dominated
    - level-based logic

Oi Electric
Parametron X-א-01, 1964
Ferro-Electronic Calculator





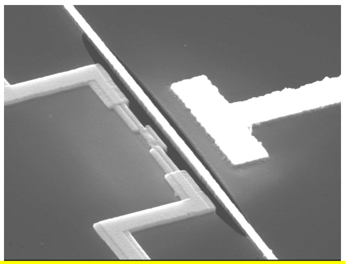
Phase Based Logic:
underlying circuitry/components
have been **difficult to miniaturise**
or **impractical for integration**

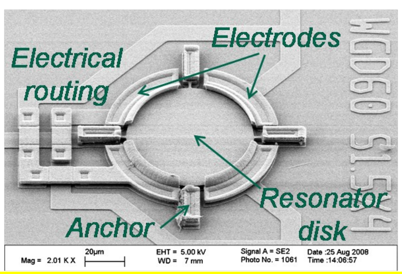
New Result: (almost) Any Oscillator will Do

details: Roychowdhury, “Boolean Computation Using Self-sustaining Nonlinear Oscillators”, arXiv:1410.5016 [cs.ET], Oct, 2014.

Wang/Roychowdhury, “PHLOGON: Phase-based LOGic using Oscillatory Nano-systems”. UCNC, 2014.

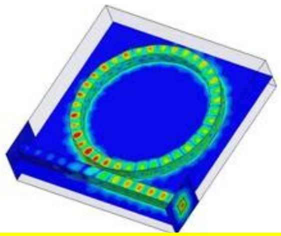
MEMS/NEMS


nanoswitch relaxation osc.



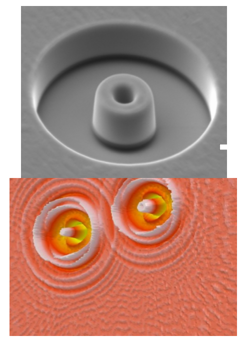
MEMS resonator osc.

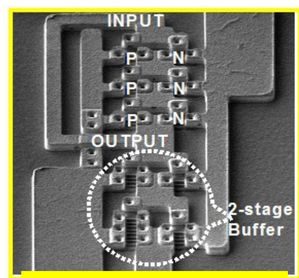
opto-electronic


opto-resonator laser


VCSELs

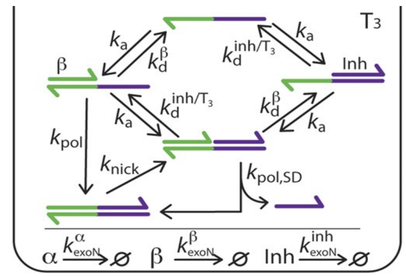
novel nanodevices


spin-torque

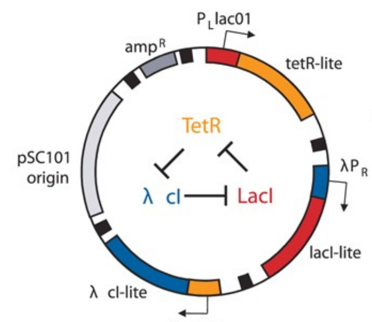


nanowire ring osc.

synth. bio. (DNA)

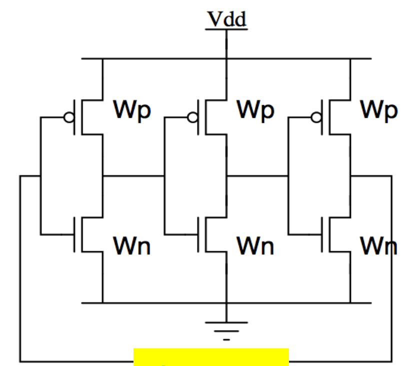


oligator

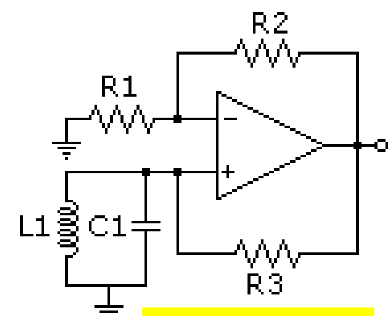


repressilator

CMOS/electronic



ring osc.

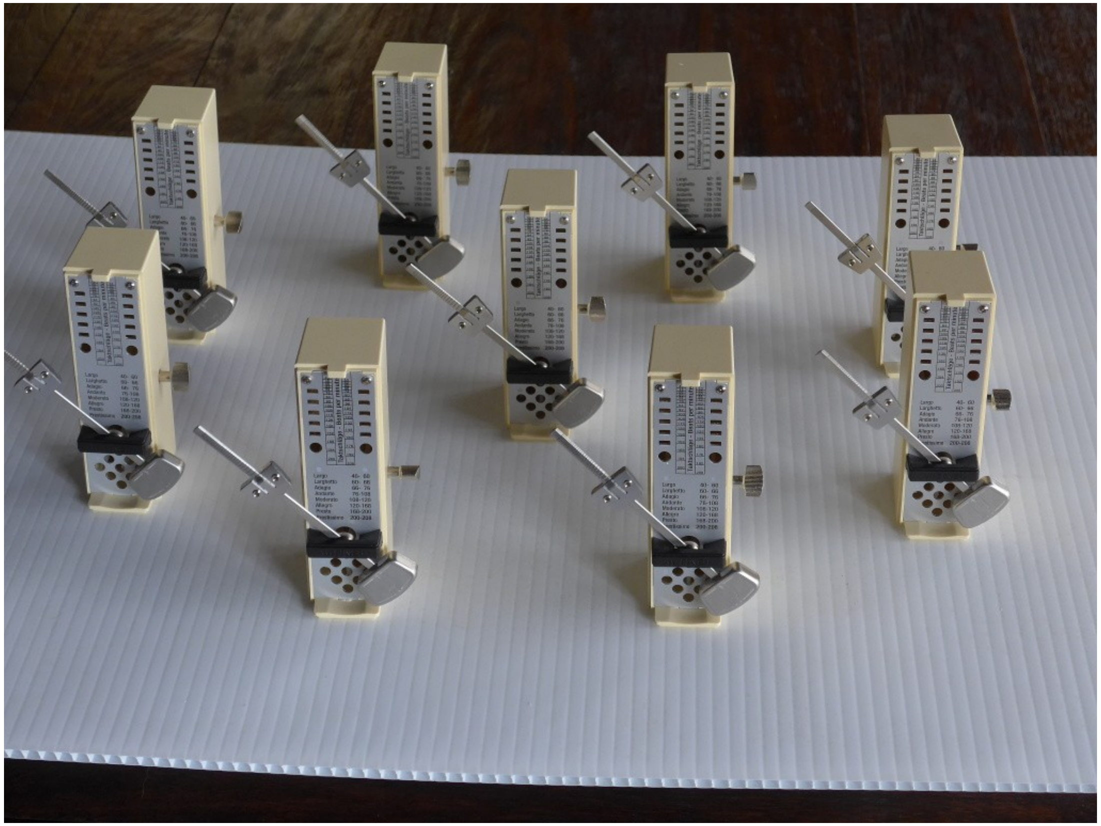


LC oscillator

many are integrable and nano-scale

Underlying Mechanism: Injection Locking

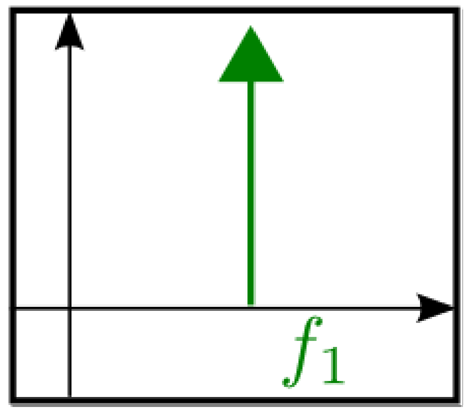
- Oscillators can synchronize in phase/frequency

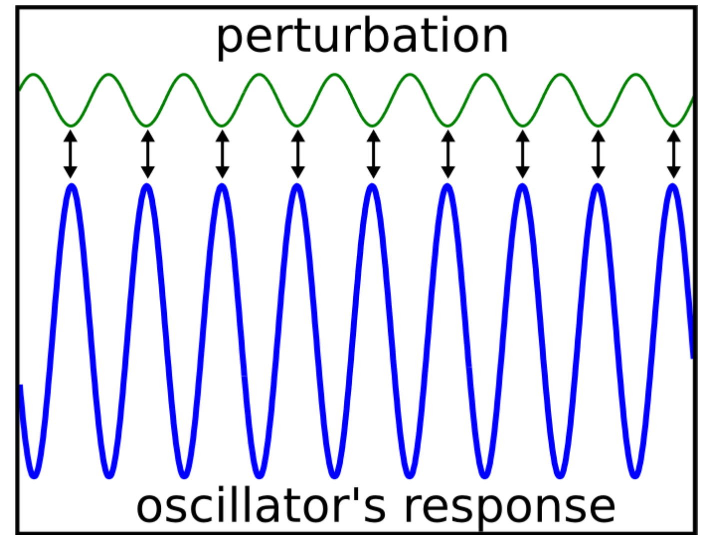


Underlying Mechanism: Injection Locking

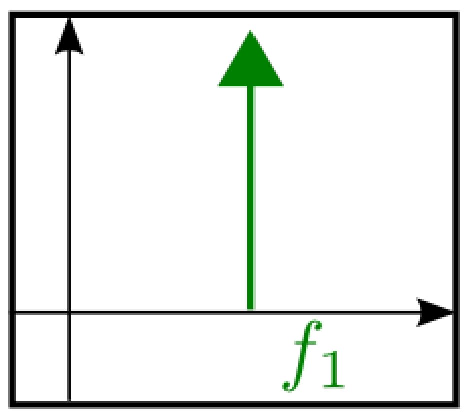
Injection Locking

phase lock





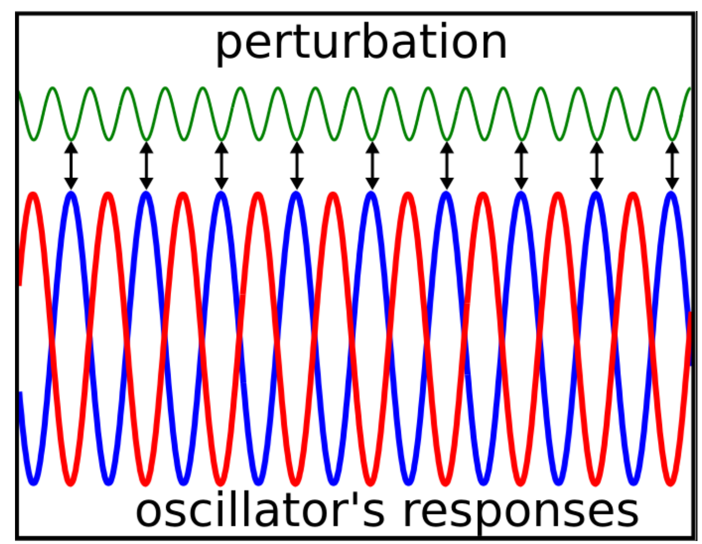
Sub-harmonic Injection Locking (SHIL)



lock 1

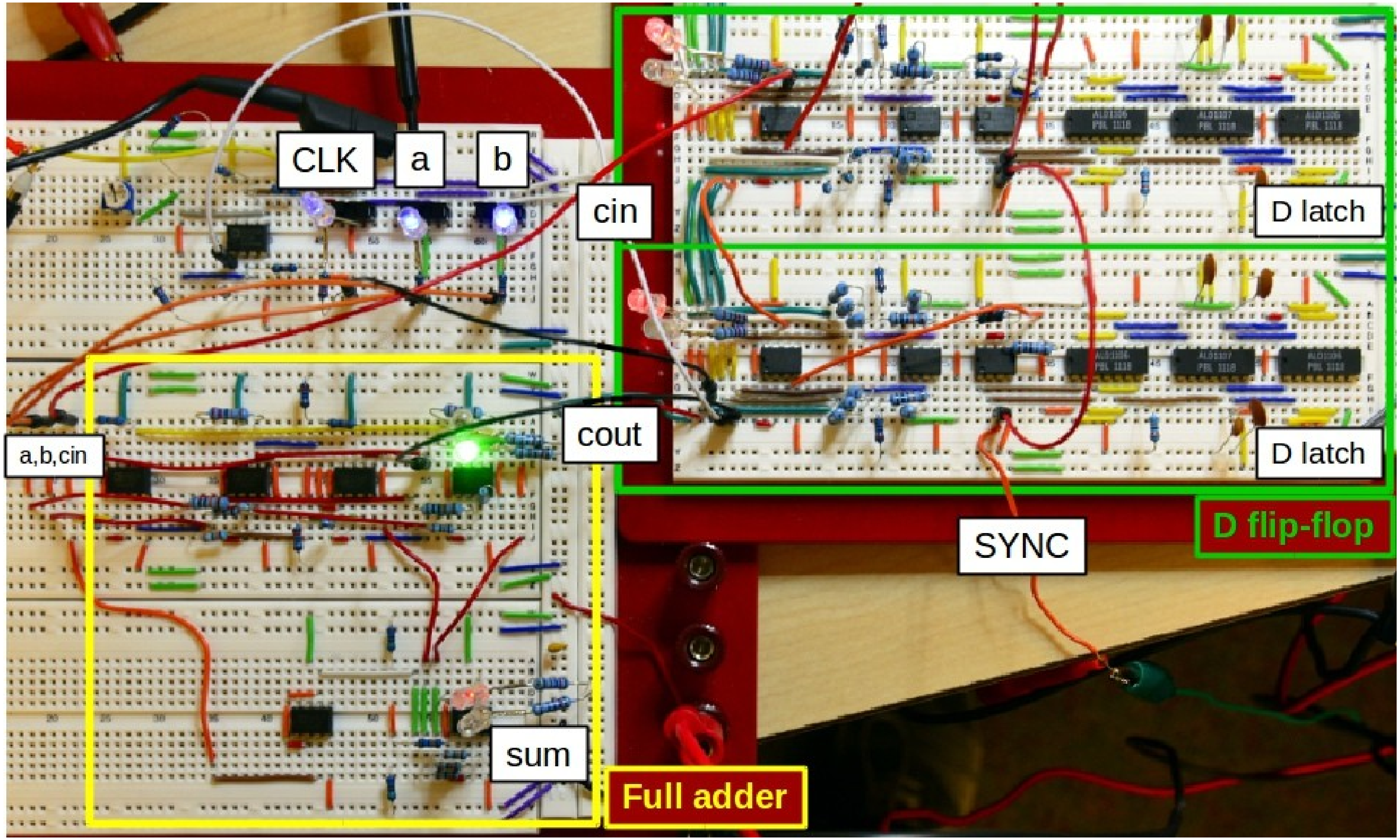
180° phase shift

lock 2

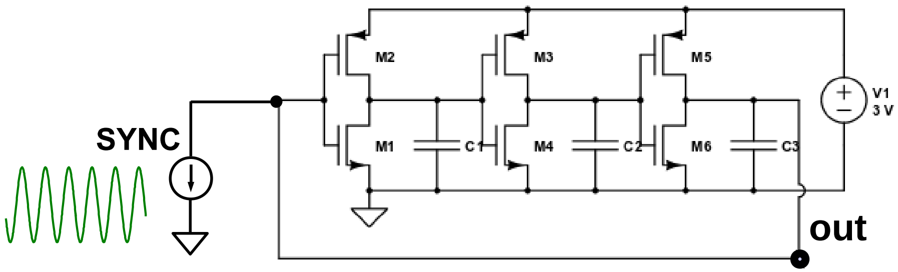


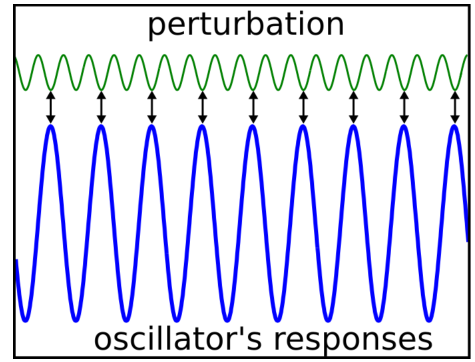
First Phase Logic FSM with Oscillators

- **PHLOGON: PHase LOGic using Oscillatory Nanosystems** using **<u>CMOS ring oscillators</u>**



Simulating SHIL of Oscillators

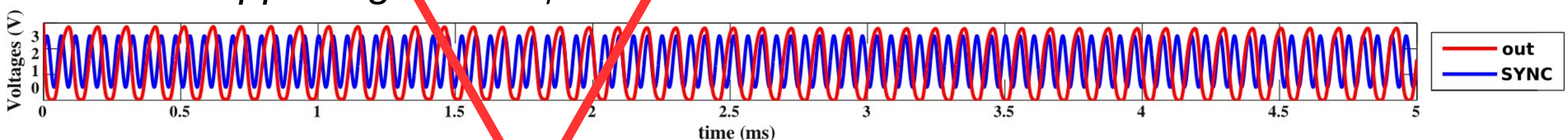




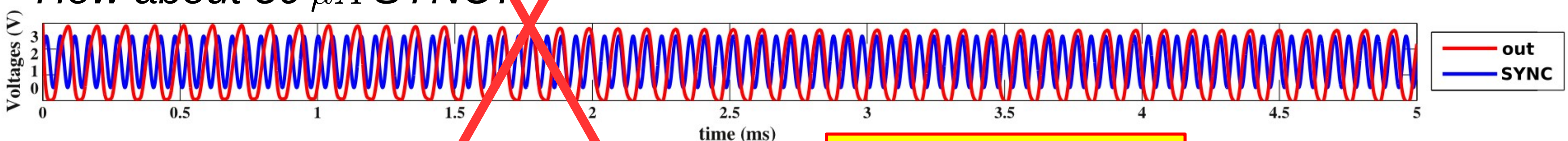
Sub-harmonic Injection Locking (SHIL)

Standard SPICE transient simulation

Is SHIL happening with 20 μA SYNC?

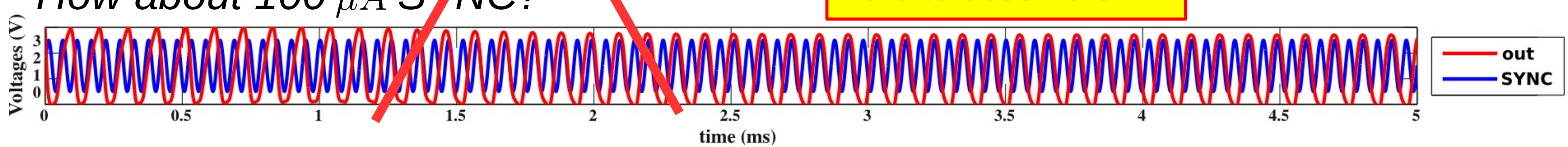


How about 50 μA SYNC?



hard to observe SHIL

How about 100 μA SYNC?



inefficient

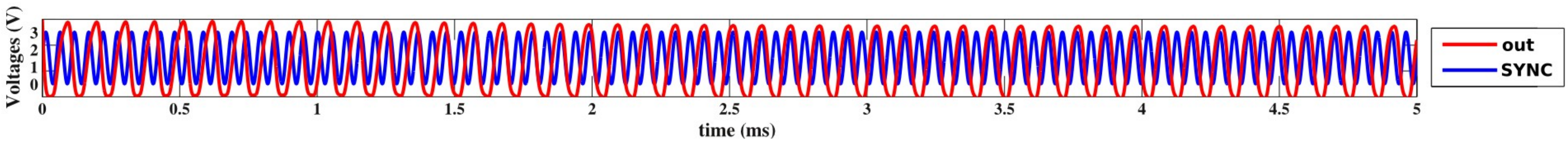
unbounded error in phase

not much insight into design

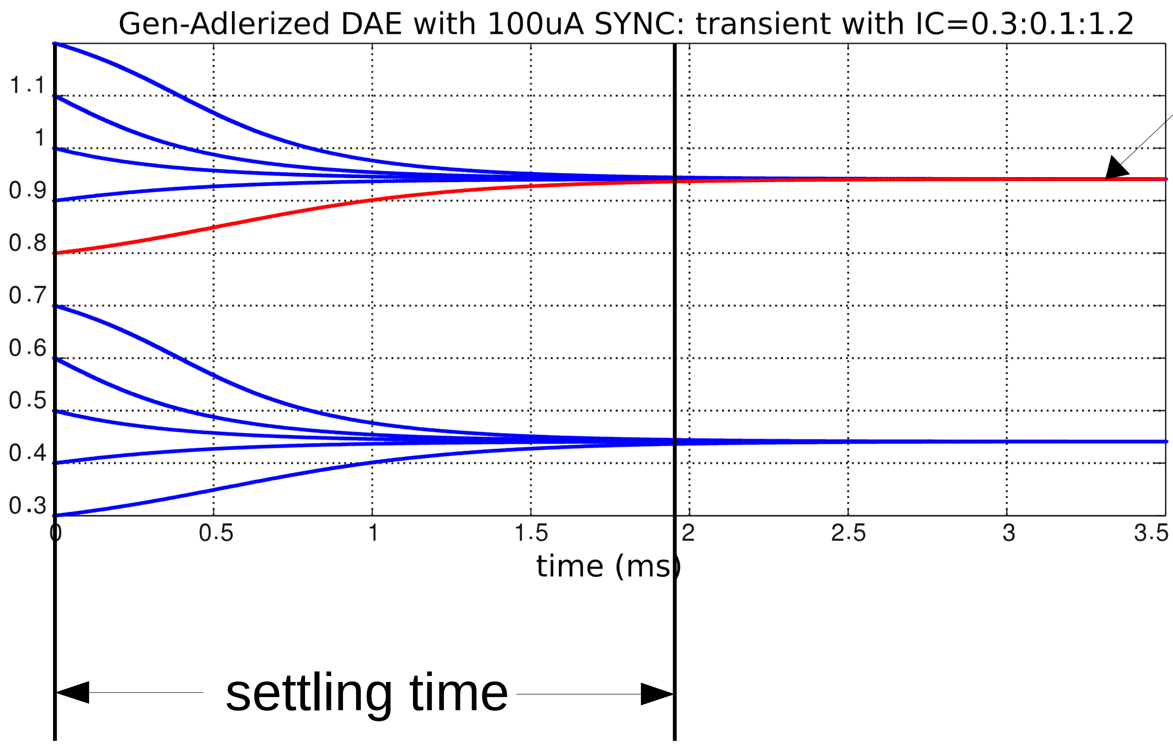
Design tools with phase macromodel analyses

Phase-macromodel-based Analyses

Standard SPICE transient simulation



Phase-based simulation



SHIL occurs: curve “flattens”

“locked phase error”

$\Delta\phi$

Generalized Adler's Equation

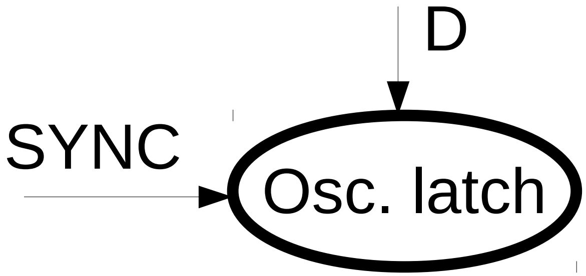
$$\frac{d}{dt}\Delta\phi(t) = f_0 - f_1 + f_0 \cdot g(\Delta\phi(t))$$

$$g(\Delta\phi(t)) = \int_0^1 \vec{v}_1^T(\tau + \Delta\phi(t)) \cdot \vec{b}_1(\tau) d\tau$$

Perturbation Projection Vector (PPV)

details: Bhansali/Roychowdhury, “Gen-Adler: the Generalized Adler's equation for injection locking analysis in oscillators”. Proc. ASPDAC, 2009.

Phase-macromodel-based Analyses



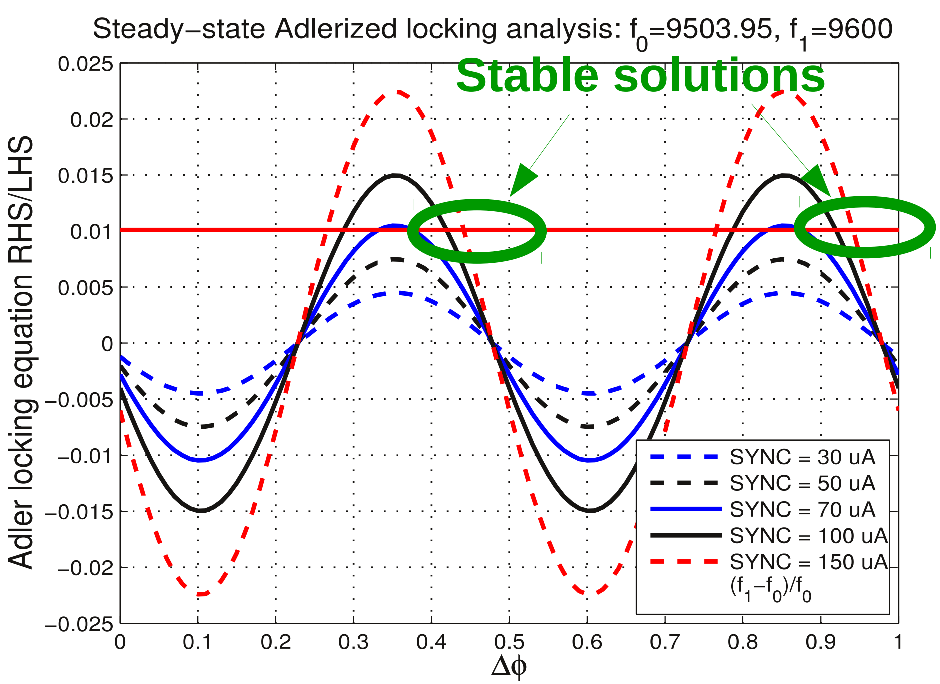
Generalized Adler's Equation (GAE)

$$\frac{d}{dt}\Delta\phi(t) = f_0 - f_1 + f_0 \cdot g(\Delta\phi(t))$$

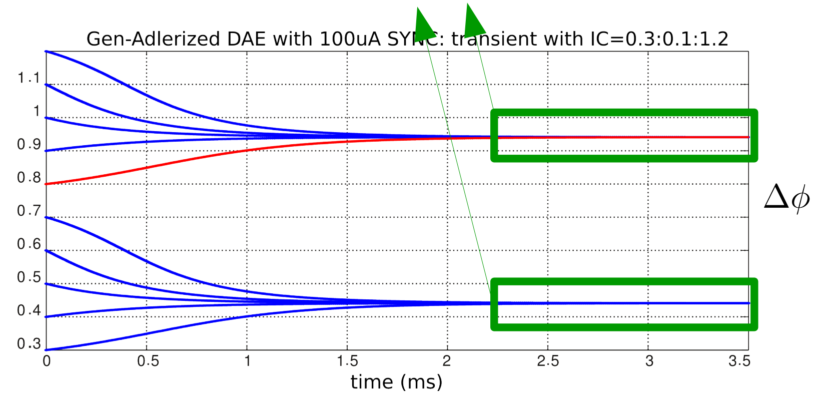
$$\frac{d}{dt}\Delta\phi(t) = 0$$

$$\frac{f_1 - f_0}{f_0} = g(\Delta\phi(t))$$

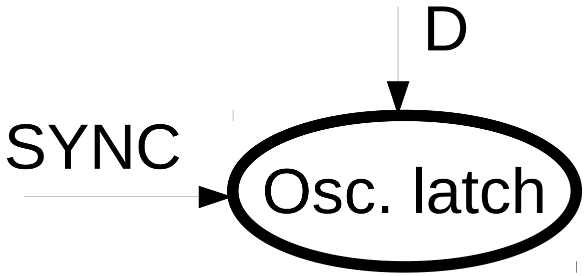
visualize LHS and RHS using MAPP



DC solutions of GAE



Phase-macromodel-based Analyses

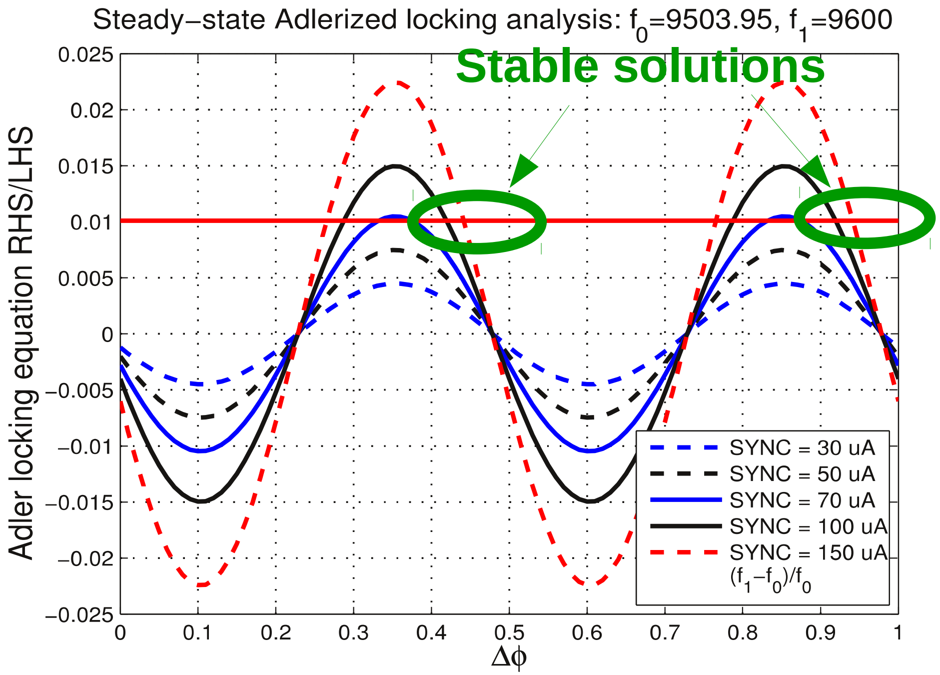


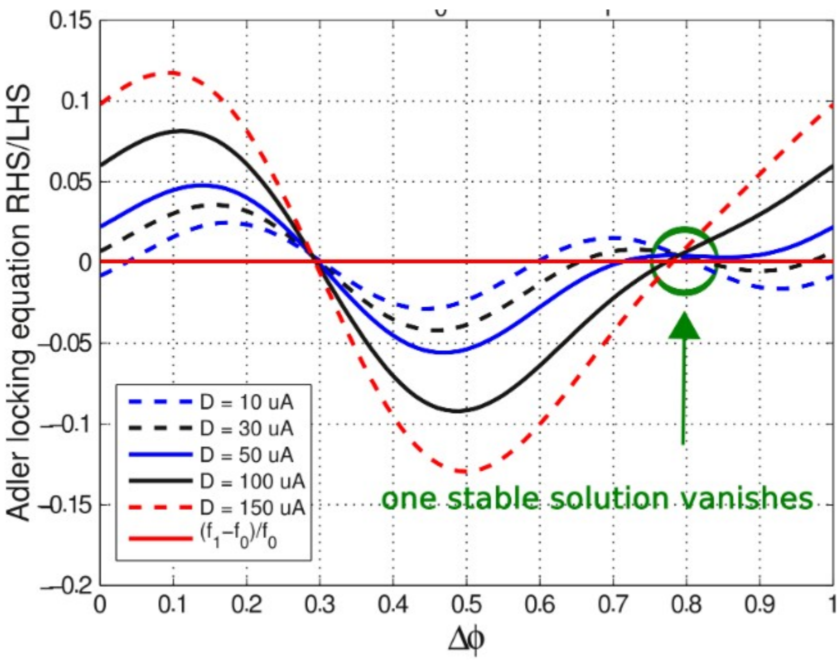
Generalized Adler's Equation (GAE)

$$\frac{d}{dt}\Delta\phi(t) = f_0 - f_1 + f_0 \cdot g(\Delta\phi(t))$$

$$\frac{d}{dt}\Delta\phi(t) = 0$$

$$\frac{f_1 - f_0}{f_0} = g(\Delta\phi(t))$$

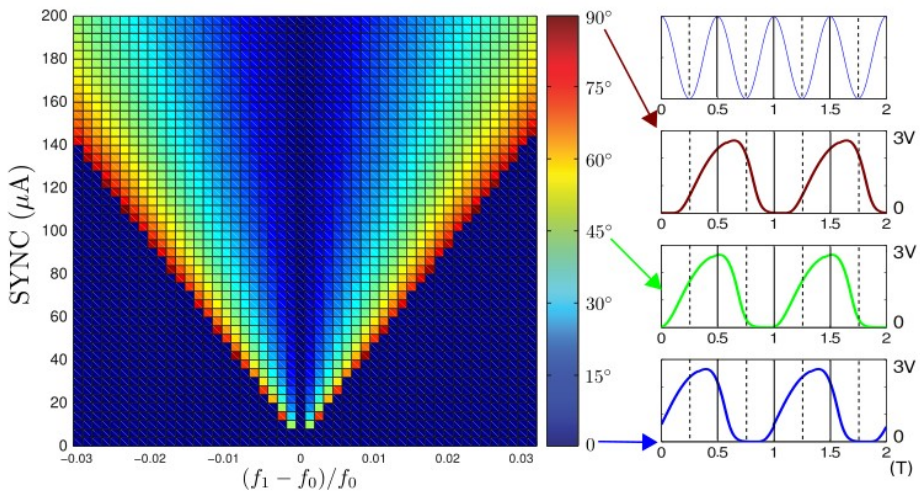




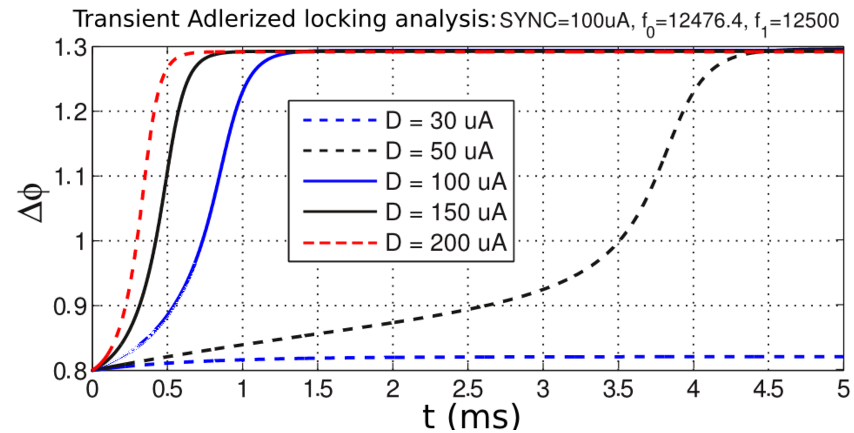
osc. latch with bit storage and bit flip

More Capabilities of the Design Tools

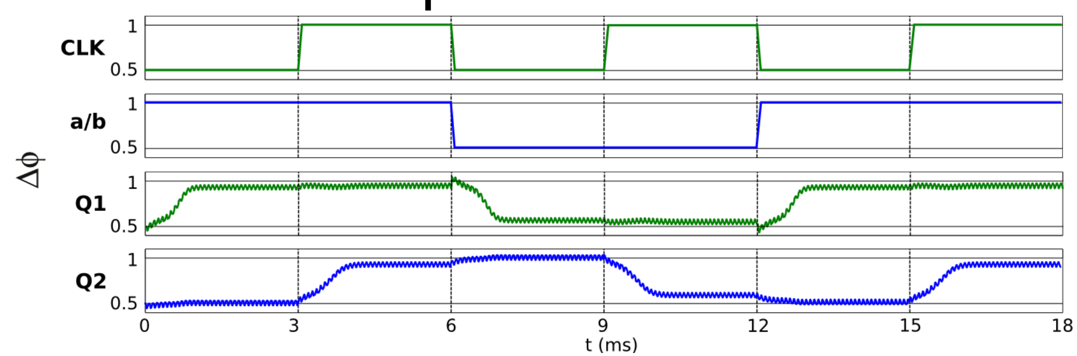
Locked phase error vs. variations in oscillator natural frequency



Timing of phase-based D latch

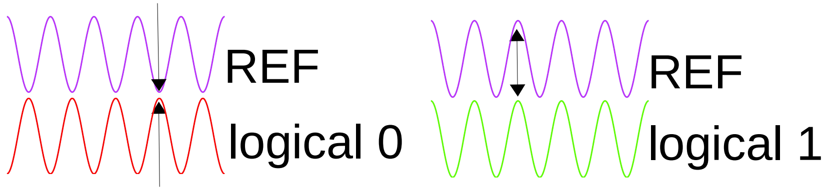


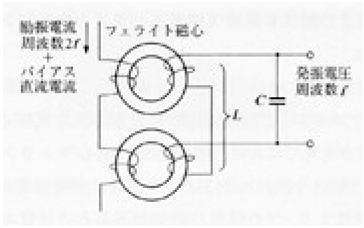
Full system transient in phase domain

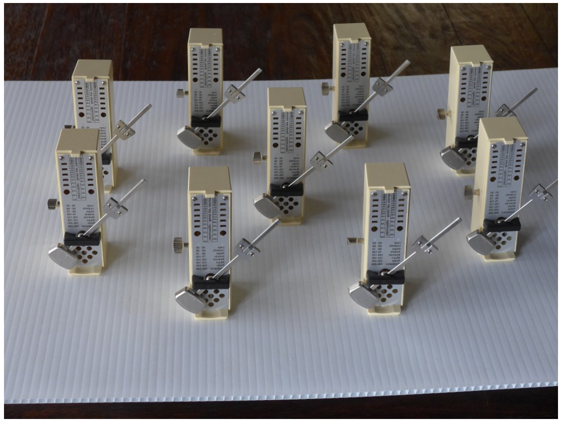


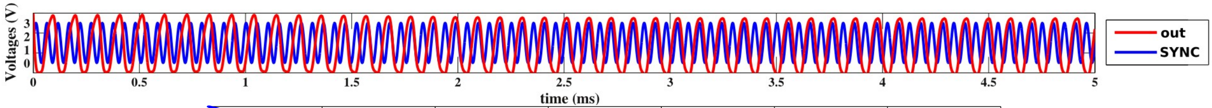
open-source release this summer

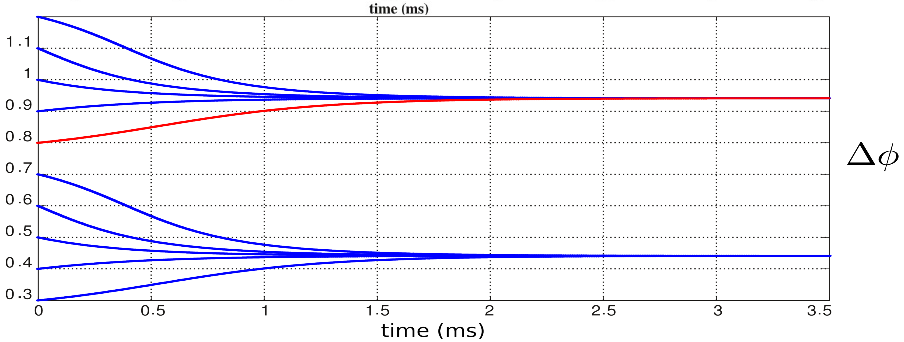
Summary

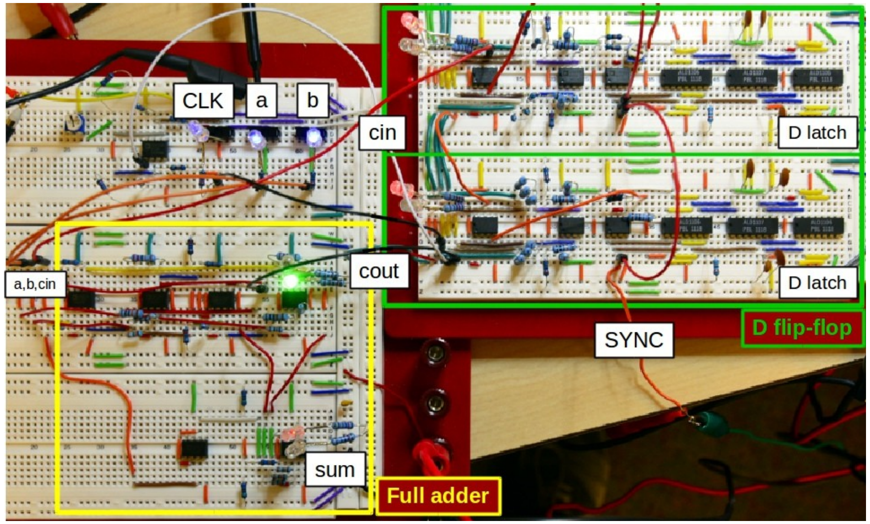

REF
logical 0
REF
logical 1

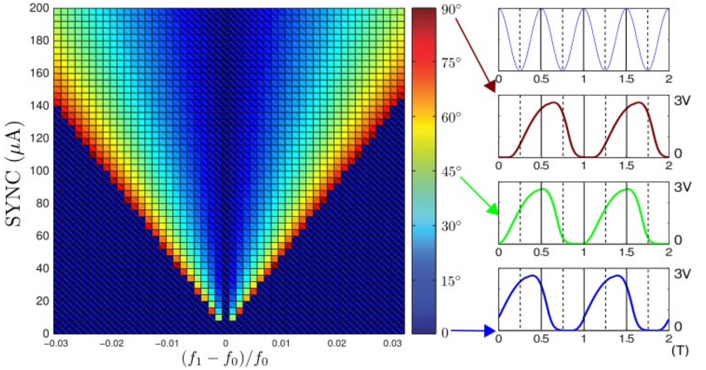

励振電流
周波数2f
+
バイアス
直流電流
フェライト磁心
L
C
発振電圧
周波数f




Voltages (V)
time (ms)
out
SYNC


time (ms)
$\Delta\phi$


CLK
a
b
cin
D latch
a,b,cin
cout
D latch
SYNC
D flip-flop
sum
Full adder


SYNC (μA)
$(f_1 - f_0)/f_0$
3V
(T)