Implementing the Nonlinear Oscillator Macromodel using Verilog-AMS Language

Ben Gu, Kiran Gullapalli, Steve Hamm, Brian Mulvaney,

Circuit Simulation Group (Mica)

Freescale Semiconductors, Austin, TX

Xiaolue Lai, and Jaijeet Roychowdhury

ECE Department University of Minnesota, Twin City, MN

IEEE Internaltional Behavioral Modeling and Simulation Conference, 2005

freescale™ semiconductor

Outline

- Introduction
- The nonlinear oscillator macromodel
- Implementation of the macromodel
- Simulation results
- Conclusions

Introduction

- Why we need oscillator macromodels?
  - Oscillators are used widely – transmitters, microprocessors
  - Simulations of oscillators are usually difficult and expensive
  - Phase characteristics of oscillators under perturbations are of significant interests, but difficult to simulate using conventional methods
- Advantages of oscillator macromodels
  - Fast without compromise of accuracy
  - Plugged into simulators – SPICE
  - Capture the phase characteristics under perturbations

Introduction (continued)

- Why we need nonlinear oscillator macromodels?
  - Oscillator is inherently a nonlinear system
  - Linear perturbation anaylsis doesn't apply to oscillators
  - Linear models often fail to accruately predict phase deviations under perturbations
- Nonlinear oscillator macromodels
  - Based on the oscillator phase noise analysis by Demir, Mehotra, and Roychowdhury
  - Successfully capture nonlinear dynamics of oscillators – injection locking, cycle slipping, …

Linear Perturbation Analysis

Unperturbed oscillator

$$\dot{x}=f(x) \longrightarrow x_s(t)$$

Perturbed oscillator

$$\dot{x}=f(x)+b(t) \longrightarrow x_p(t)=?$$

Linear perturbation analysis

$$\dot{x}=f(x)+b(t) \longrightarrow x_s(t)+w(t)$$

If *w(t)* always small

$$\dot{w}(t)\approx\frac{\partial f}{\partial x}w(t)+b(t)$$

w(t) can be made to grow UNBOUNDED despite *b(t)* remaining small

"Phase noise in oscillators: a unifying theory and numerical methods for characterization" *Demir, A. Mehrotra, A. Roychowdhury, J.,* IEEE Trans on Circuits and Systems, pp.655-674 (47), 1999

Nonlinear Perturbation Analysis

$$\dot{x} - f(x) = b_1(t) + \tilde{b}(t)$$

neglect the amplitude perturbation in our discussions

$$\dot{x} - f(x) = b_1(t)$$

$$b_1(t) = v_1^T(t + \alpha(t))\, b(t)\, u_1(t + \alpha(t))$$

Perturbation Projection Vector (PPV)

$\alpha(t)$ ⟶ Phase deviations $\qquad \dot{\alpha}(t) = v_1^T(t + \alpha(t))\, b(t)$

Solution of perturbed oscillator ⟶ $x_p(t) = x_s(t + \alpha(t))$

"Phase noise in oscillators: a unifying theory and numerical methods for characterization" *Demir, A. Mehrotra, A. Roychowdhury, J.,* IEEE Trans on Circuits and Systems, pp.655-674 (47), 1999

Nonlinear Oscillator Macromodel

Nonlinear macromodel based on those equations
(*X. Lai and J. Roychowdhury, ICCAD 2004*):

$$\dot{\alpha}(t) = v_1^T(t + \alpha(t))\, b(t)$$

$$x_p(t) = x_s(t + \alpha(t))$$

- Steps of building the macromodel:
  - Simulate the full-sized unperturbed oscillator to get unperturbed solution and PPV
  - Solve the nonlinear differential equation for α(t)
  - Get the perturbed solution
- Advantages of the macromodel
  - Reduction of system dimension
  - Speed-up of simulations
  - Information of phase deviations

Implementing the macromodel using Verilog- AMS

Implement an oscillator under perturbation as a Verilog- AMS module

```
module oscillator(in, out);
inout in, out;
electrical in, out, alpha;
......
```

$$\dot{\alpha}(t)=v_1^T(t+\alpha(t))\,b(t)$$

```
ddt(V(alpha)) = PPV($abstime+V(alpha))*V(in);
```

Time derivative operator in Verilog- AMS

Implementing the macromodel using Verilog- AMS (continued)

```
ddt(V(alpha)) = PPV($abstime+V(alpha))*V(in);
```

periodic waveform, no analytical form

$table_model is perfect for the job!

Usage:

- Store PPV waveform into a file, ppv.table
- $table_model($abstime+V(alpha) % period, “ppv.table”, “L”)

Implementing the macromodel using Verilog- AMS (continued)

Set up two branches:

```
branch (alpha) alpha1;
branch (alpha) alpha2;
...
perturbed_time=($abstime+V(alpha))%period;
ppv=$table_model(perturbed_time,“ppv.table”,”L”);

I(alpha1) <+ -ppv*V(in);
I(alpha2) <+ ddt(V(alpha));
```

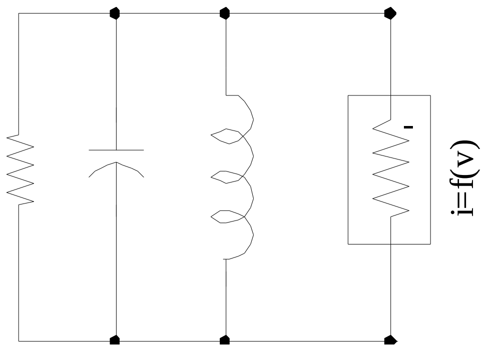
$$\dot{\alpha}(t) = v_1^T(t+\alpha(t))\,b(t)$$

KCL at node alpha will force I(alpha1) + I(alpha2) = 0, so as $abstime proceeds, this nonlinear differential equation is solved at each time step

Implementing the macromodel using Verilog- AMS (continued)

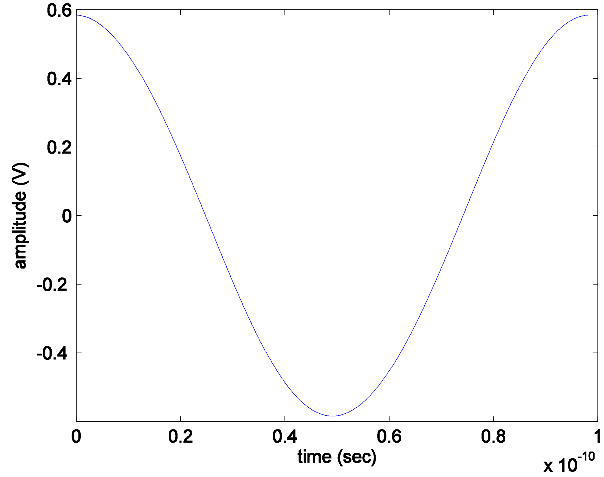
```
`include "discipline.h"
`include "constants.h"

// nonlinear macromodel implemented in Verilog-AMS
module oscillator(in, out);
// define nodes
inout      in, out;
electrical in, out, alpha;

// define variables and parameter
real phase, ppv, period, omega,
     perturbed_time;

parameter real freq=1.0e9 from(0:inf);

// define branches
branch (alpha) alpha1;
branch (alpha) alpha2;

analog
begin
    @(initial_step)
    begin
    //set up initial condition for eq(9)
    V(alpha) <+ 0;
    omega =  2.0*`M_PI*freq;
    period = 1.0/freq;
    end

    // real perturbed time is given by
    // $abstime + V(alpha)
    // but our ppv table has only one period
    // so take the modulus of period
perturbed_time = ($abstime+V(alpha)) % period;

    // look up table to get ppv value
    // $table_model(arg1, arg2, arg3)
    // arg1: input (independent) variable
    // arg2: name of the file storing the table
    // arg3: interpolation method, L-linear


   ppv =  $table_model(perturbed_time,
           "osc_ppv.table", "L");

    // right hand side of eq(9)
    // ppv:  v1(t+alpha(t))
    // V(in):b(t)
    I(alpha1) <+ -ppv*V(in);

    // left hand side of eq(9)
    I(alpha2) <+  ddt(V(alpha));

    // KCL at alpha forces
    //       I(alpha1) + I(alpha2) = 0
    // so that eq(9) is solved at each time point

    phase = V(alpha)*omega; // output only

    // look up table to generate output
    V(out) <+ $table_model(perturbed_time,
              "osc_output.table", "L");
end
endmodule
```

Simulation Results

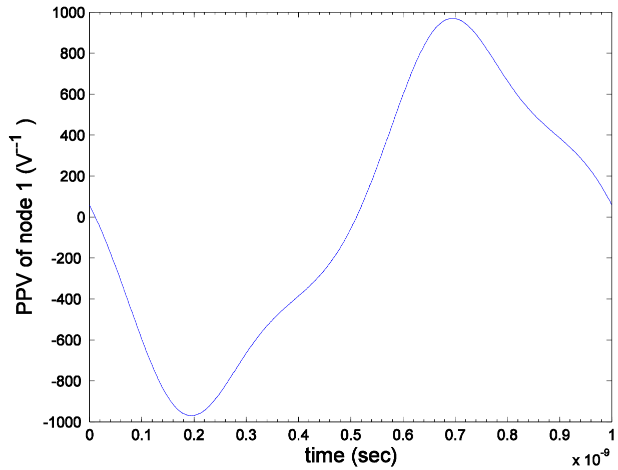


$$-C\frac{dv(t)}{dt}=\frac{v(t)}{R}+i(t)+f(v(t))+b(t)$$

$$L\frac{di(t)}{dt}=v(t)$$



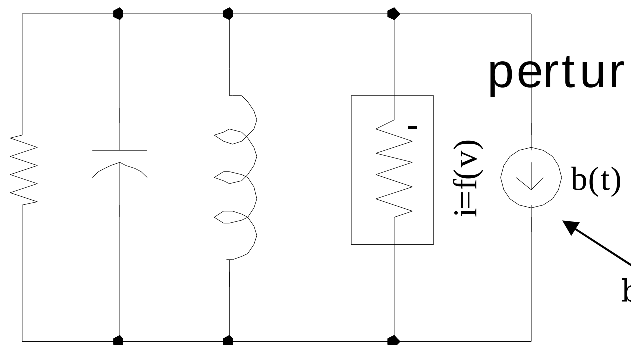
Oscillating freq: 1.0GHz, mag: 600mV



Simulate using Freescale's in-house circuit simulator, Mica, to get steady state solution and PPV waveform

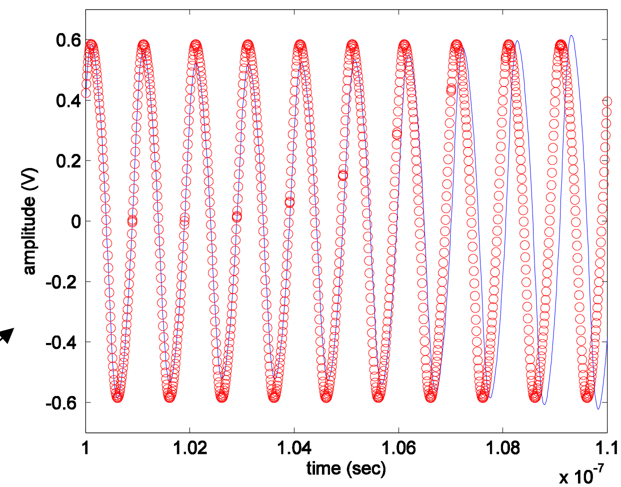
Simulation Results (continued)

Full SPICE simulation (time domain)



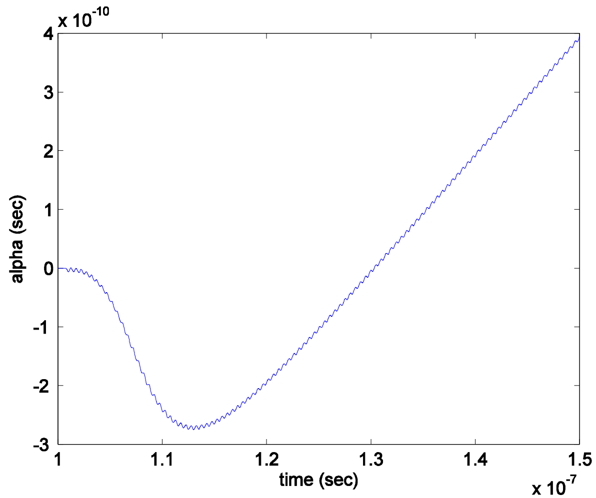
perturbation

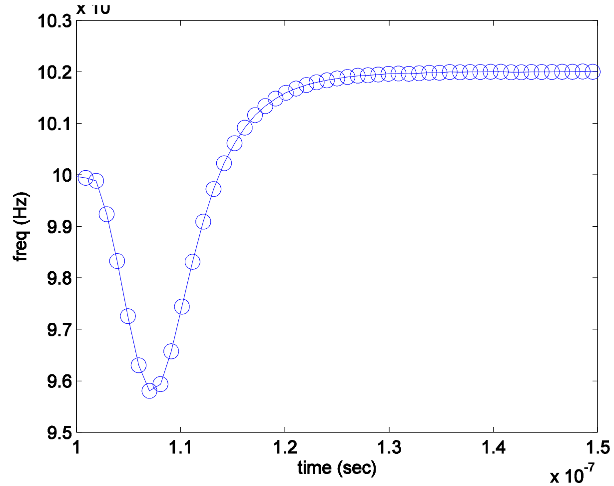
$b_0 \sin(\omega t)$



Macromodel (phase domain)

$$x_p(t) = x_s(t + \alpha(t))$$

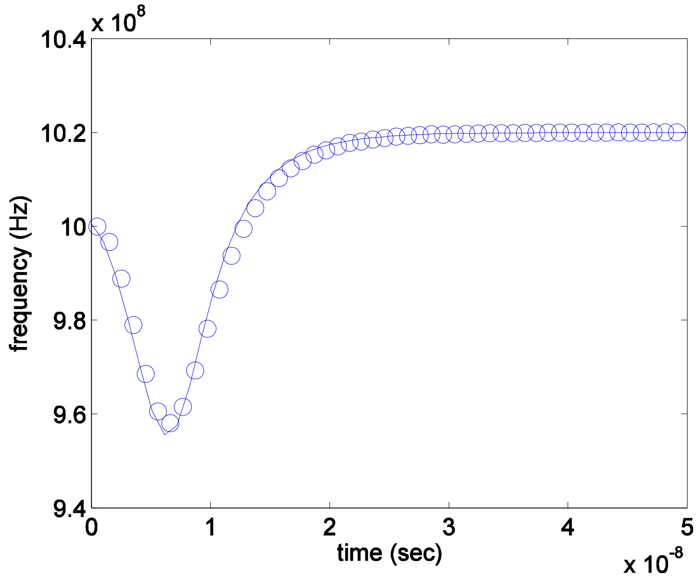


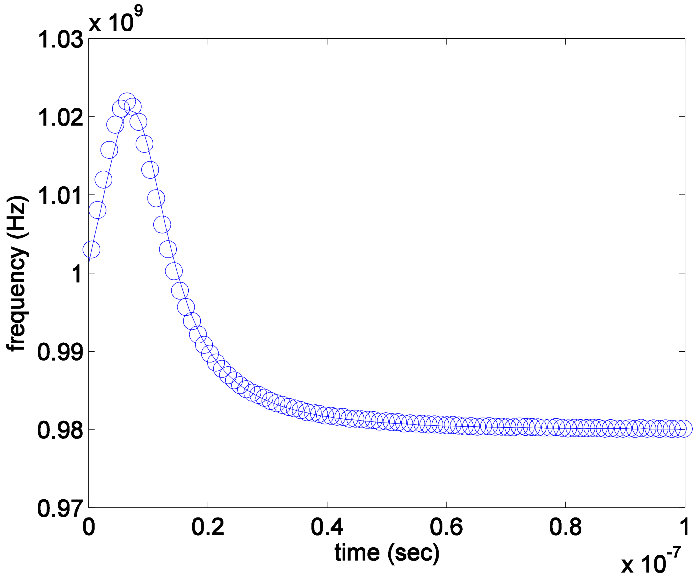


Simulation Results (continued)

Injection locking

(lines) macromodel (symbols) full SPICE simulation





Perturbation frequency: 1.02 GHz

Perturbation amplitude: 100 uA

Perturbation frequency: 0.98 GHz

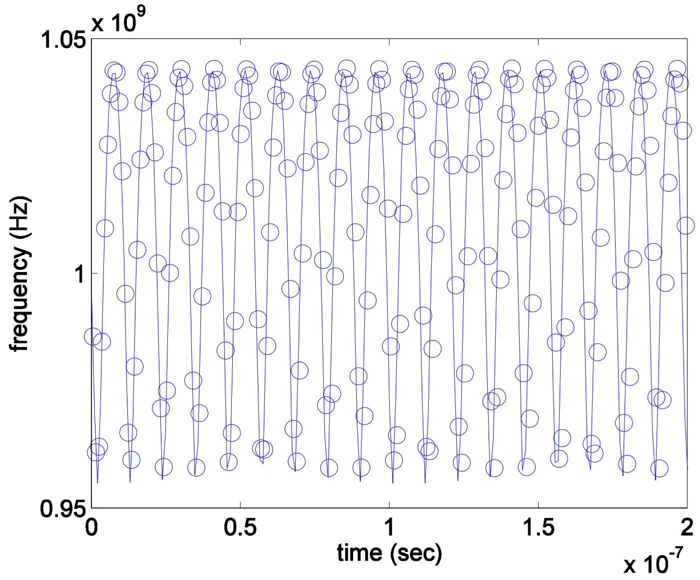
Perturbation amplitude: 50 uA

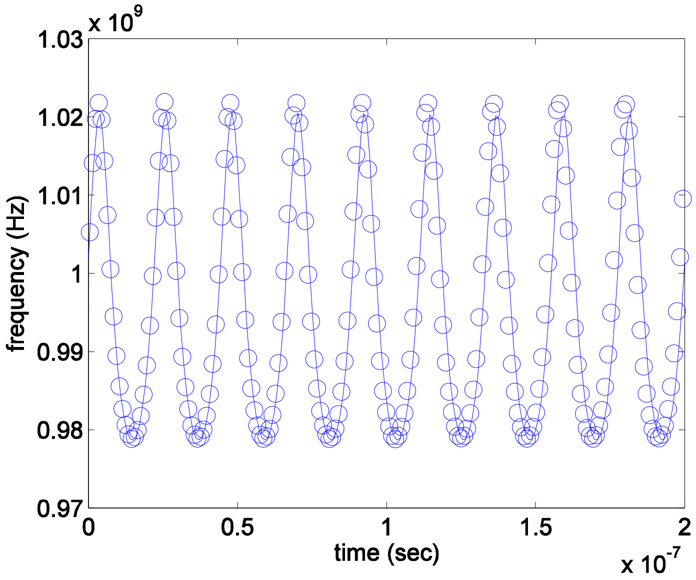
See Lai and Roychowdhury, ICCAD 2004 for detailed discussions about injection locking

Simulation Results (continued)

Injection unlocking

(lines) macromodel (symbols) full SPICE simulation





Perturbation frequency: 1.1 GHz

Perturbation amplitude: 100uA

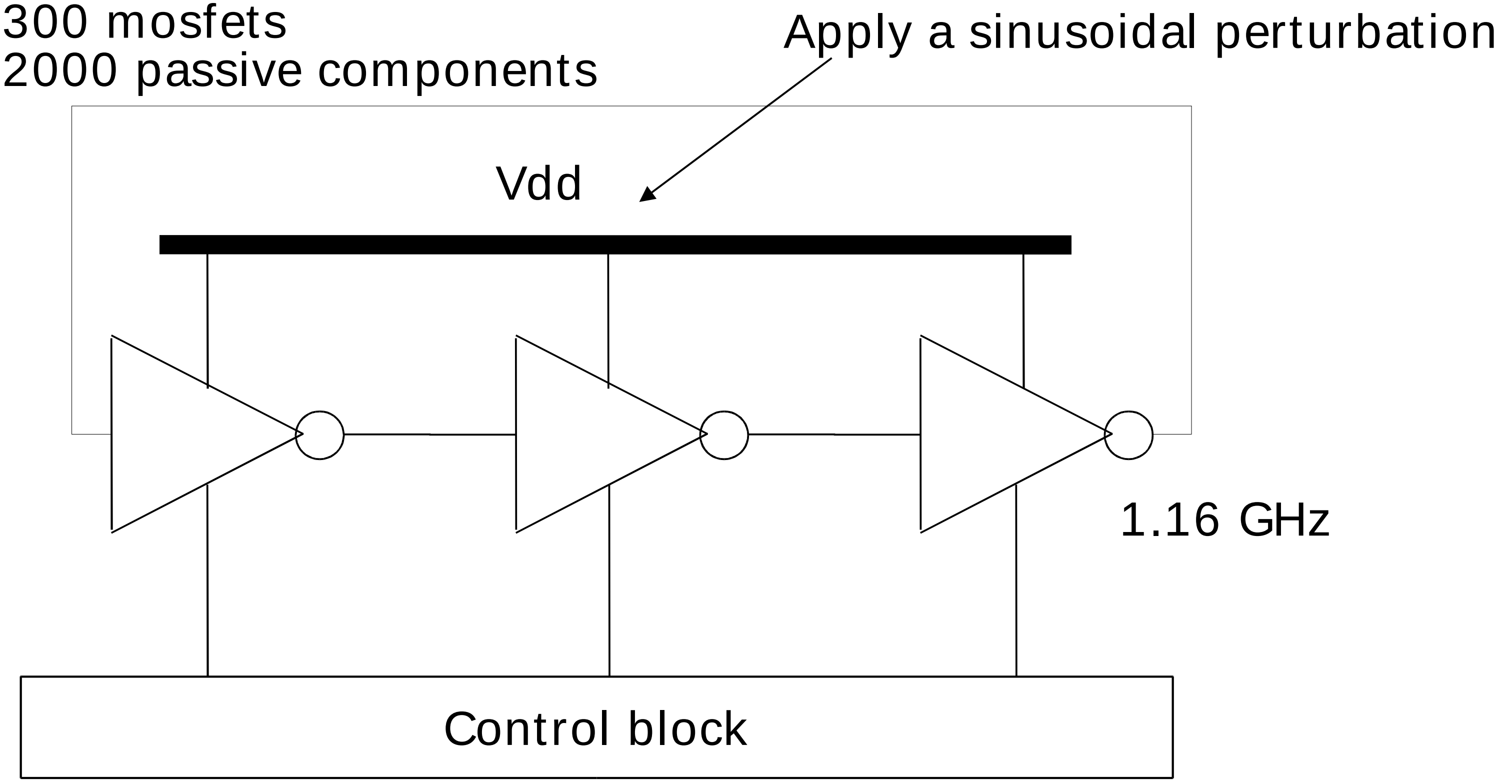
Perturbation frequency: 0.95 GHz

Perturbation amplitude: 50uA

Simulation Results (continued)

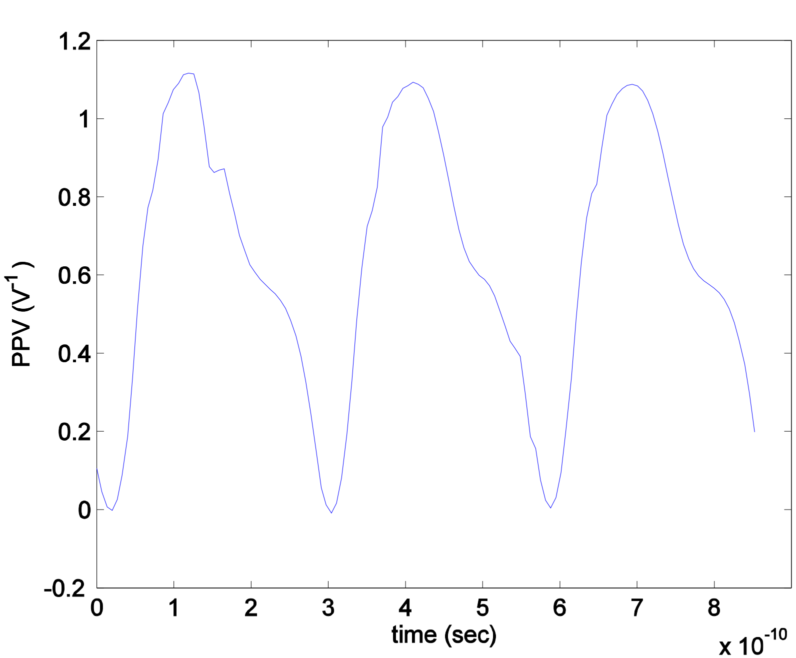
3 stage ring oscillator (used in Freescale products)

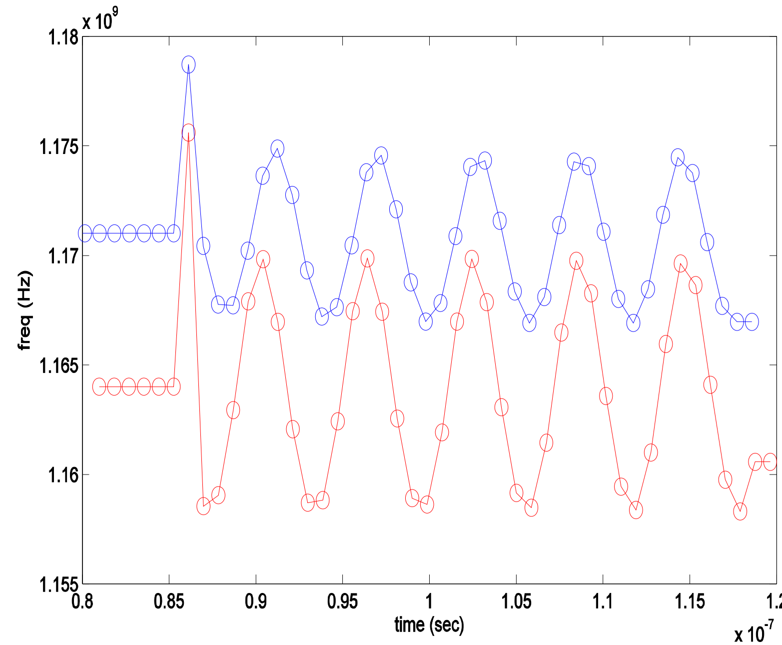
300 mosfets
2000 passive components



Nearly impossible to study the phase deivations using time domain methods (SOI floating body effect)

Simulation Results (continued)





qualitatively matches full SPICE simulation

SPICE simulation hasn't reached the real steady state yet!

System reduction: 1709 to 1
Speed-up: 969sec to 9sec

100X speed-up

Conclusions

- The nonlinear oscillator macromodel can be implemented in Verilog-AMS very compactly
- The macromodel implemented in Verilog-AMS can be readily plugged into SPICE to perform simulations with other blocks
- The macromodel is able to capture the nonlinear locking dynamics of oscillators
- The macromodel can provide significant system reduction and speed up over full SPICE representation without loss of accurary

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © Freescale Semiconductor, Inc. 2005.