Analysis and Design Optimisation of Electronic Circuits using Oscad and OpenModelica

OpenModelica Annual Workshop 2015

Rakhi R and Kannan M. Moudgalya
Indian Institute of Technology Bombay, India

February 2, 2015

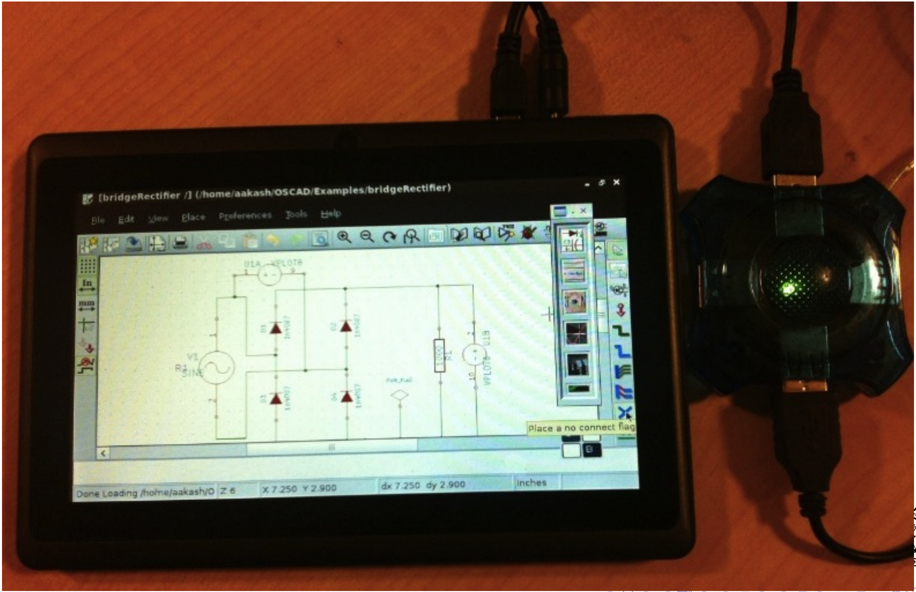
Outline

- What is Oscad?
- Enhancing explanation feature using OM in Oscad v2.0
- Circuit design optimisation examples
- Integrating Ngspice to Modelica converter in OM suite
- Conclusions and future work

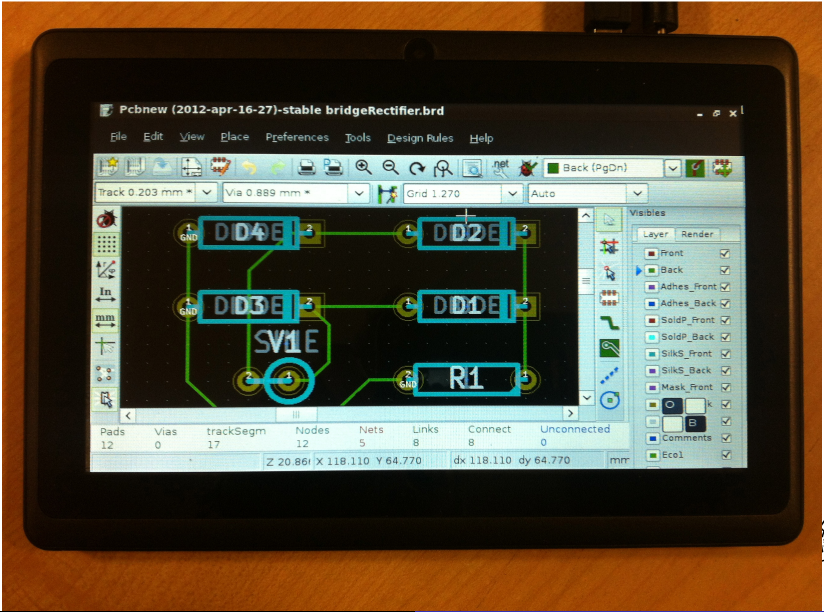
What is Oscad?

- Open Source Electronic Design Automation (EDA) tool
- Developed at IIT Bombay
- Capable of circuit design, simulation, analysis (explanation feature) and PCB layout design
- Runs on Linux, Windows, Aakash
- About half a million students need such a tool every year in India
- India is also a centre for circuit designs

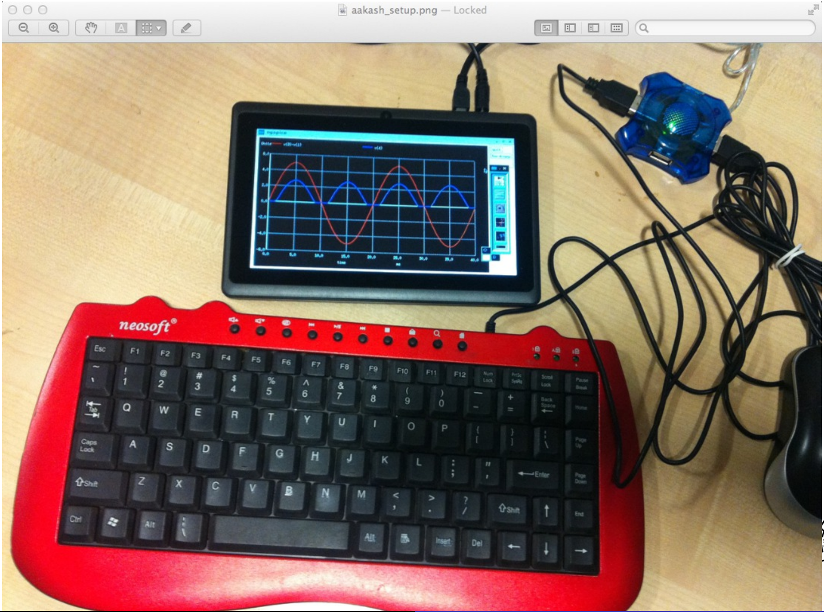
Schematics: Oscad on Aakash



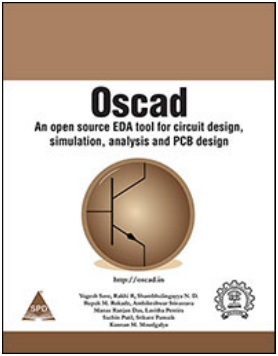
PCB layout: Oscad on Aakash



Method to use: Oscad on Aakash



Released Oscad Book as Open Source



Oscad: Electronic Design Automation

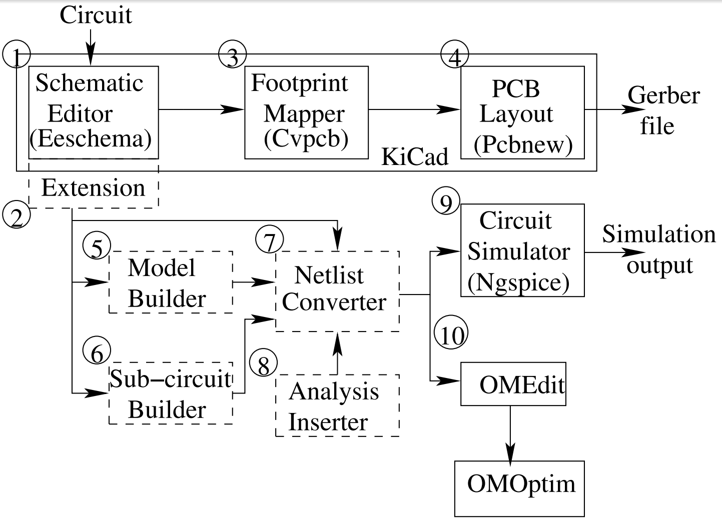
Built by putting together

- KiCAD for schematics
- Ngspice for simulation
- Fritzing for bread boarding
- Scilab to see mathematical equations
- GHDL for digital simulation

Explanation Facility

- Wanted to extract mathematical equations
- Can explain (oscillations, correctness, etc.) using them
- Previous method: manual generation
- Needed to maintain the interface to solvers
- Needed different interfaces for different types of problems
- Developed an OpenModelica interface

Design Flow in Oscad v2.0



Create circuit schematic → Ngspice netlist → use OM tools for optimisation and circuit equations

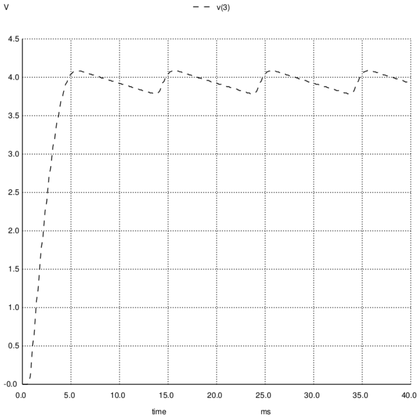
Integrating OpenModelica and Oscad

1. Convert Ngspice netlist to Modelica code
2. LaTeX interface to display equations in standard mathematical notations
3. Open OMOptim for circuit design optimisation

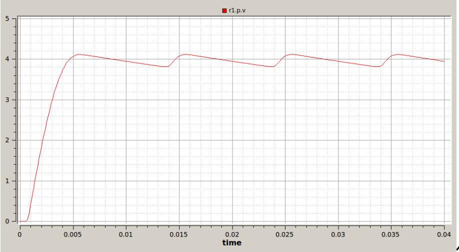
Ngspice to Modelica Conversion

- OM requires circuit description in Modelica language
- Oscad provides circuit description as Ngspice netlist
- Netlist converters available in Dymola, do not work in OM
- A tool for Ngspice to Modelica conversion developed for OM
- Takes care of Spice device parameters, can handle subcircuits
- Opens the converted Modelica code in OMEdit

Comparison of Ngspice and Modelica for Bridge Rectifier with Filter



Ngspice Simulation



OM Simulation

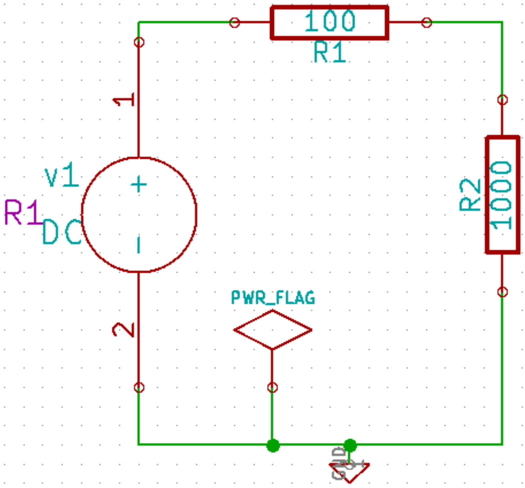
LaTeX Interface for Equation Generator

LaTeX Interface for Equation Generator

1. Ngspice code converter generates Modelica code
2. OMC generates equations from Modelica code
3. Parser extracts the equations and writes to a text file
4. Python code converts text to LaTeX
5. Compile LaTeX code to produce pdf

Shell script to perform all the above steps.
Let us see the output of some of these steps using a voltage divider circuit

Equation Generator Example: Circuit and Ngspice Netlist



r2 2 0 100
r1 2 1 100
v1 1 0 5
.dc v1 0e-00 5e-00 1e-00
** Control Statements*
.control
run
.endc
.end

OpenModelica Description

```
model max_power_transfer
Modelica.Electrical.Analog.Basic.Ground
ground1;
Modelica.Electrical.Analog.Basic.Resistor R1(R
= 100);
Modelica.Electrical.Analog.Basic.Resistor R2(R
= 100);
Modelica.Electrical.Analog.Sources.ConstantVolta
VDC(V = 5);
equation
connect(VDC.n, ground1.p);
connect(R2.n, ground1.p);
connect(R1.n, R2.p);
connect(VDC.p, R1.p);
end max_power_transfer;
```

Output of Parser

```
Equations (9, 9)
========================================
1/1 (1): R2.LossPower = (-R2.v) * VDC.i
2/2 (1): R2.v = (-R2.R_actual) * VDC.i
3/3 (1): R1.v = VDC.V - R2.v
4/4 (1): R1.LossPower = (-R1.v) * VDC.i
5/5 (1): R1.v = (-R1.R_actual) * VDC.i
6/6 (1): R2.R_actual = R2.R * (1.0 + R2.alpha *
7/7 (1): R1.R_actual = R1.R * (1.0 + R1.alpha *
8/8 (1): R1.T = R1.T_ref
9/9 (1): R2.T = R2.T_ref


State Sets
```

LaTeX Code

```
\begin{align}
R2.LossPower &= (-R2.v) \times VDC.i\\
R2.v &= (-R2.R\_act) \times VDC.i\\
R1.v &= VDC.V - R2.v\\
R1.LossPower &= (-R1.v) \times VDC.i\\
R1.v &= (-R1.R\_act) \times VDC.i\\
R2.R\_act &= R2.R \times (1.0 + R2.\alpha \
    times (R2.T - R2.T\_ref))\\
R1.R\_act &= R1.R \times (1.0 + R1.\alpha \
    times (R1.T - R1.T\_ref))\\
R1.T &= R1.T\_ref\\
R2.T &= R2.T\_ref\\
\end{align}
```

Equations Generated

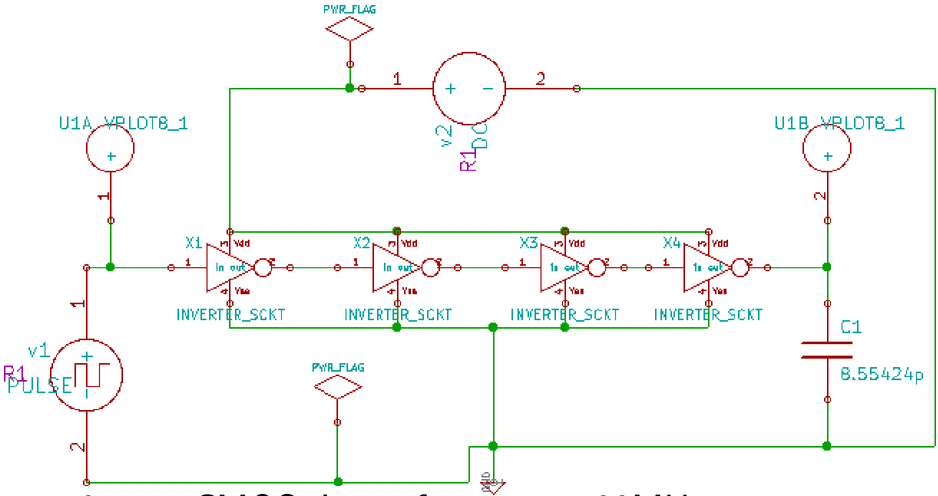
$$R2.LossPower = (-R2.v) \times VDC.i \quad (1)$$
$$R2.v = (-R2.R\_act) \times VDC.i \quad (2)$$
$$R1.v = VDC.V - R2.v \quad (3)$$
$$R1.LossPower = (-R1.v) \times VDC.i \quad (4)$$
$$R1.v = (-R1.R\_act) \times VDC.i \quad (5)$$
$$R2.R\_act = R2.R \times (1.0 + R2.\alpha \times (R2.T - R2.T\_ref)) \quad (6)$$
$$R1.R\_act = R1.R \times (1.0 + R1.\alpha \times (R1.T - R1.T\_ref)) \quad (7)$$
$$R1.T = R1.T\_ref \quad (8)$$
$$R2.T = R2.T\_ref \quad (9)$$

Circuit design optimisation examples

Circuit design optimisation examples

- Inverter chain delay minimisation
- Low noise amplifier design

Min. prop. delay in CMOS inv. chain



- $0.5\mu m$ CMOS, Input frequency 500MHz
- In each inverter, NMOS, width $W_n = w_{inv} \times \lambda$, PMOS, $W_p = 3W_n$, $\lambda = 0.25\mu m$

Parameter optimisation

- Find widths $w_{inv2}, w_{inv3}, w_{inv4}$ such that propagation delay is minimum
- Delay min. when width of every inverter in the chain approx. 4 times that of previous one

Algo.	**Delay (s)**	w_inv2, w_inv3, w_inv4	**Time (m)**
PSO	2.73e-10	34.95, 119.21, 463.40	15:47
NSGA2	2.71e-10	36.87, 137.84, 508.16	15:40
SPEA2	2.86e-10	33.44, 149.63, 345.18	17:23

- Widths multiplied by about 3.6 (on an average) in subsequent stages

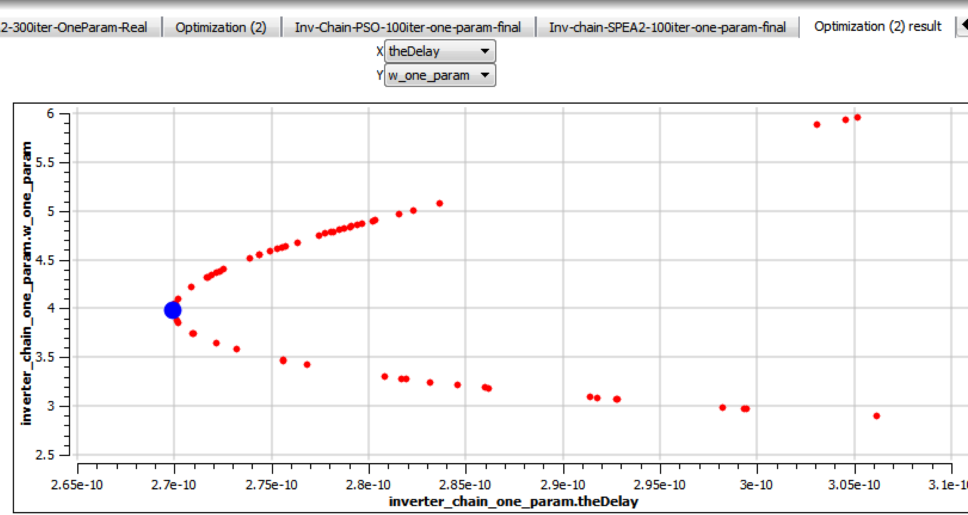
Parameter optimisation

- Above problem converted to 1 parameter optimisation problem
- Defined $W_{inv2} = h \times W_{inv1}$, $W_{inv3} = h^2 \times W_{inv1}$ so on
- h optimised for minimum delay

Algorithm	**Delay (s)**	h	**Comp. Time (min.)**
PSO	2.69e-10	3.9896	15:13
NSGA2	2.69e-10	3.9942	16:10
SPEA2	2.69e-10	4.0020	15:08

- The results clearly show that the delay is minimum when the stage effort is around 4

Parameter optimisation results

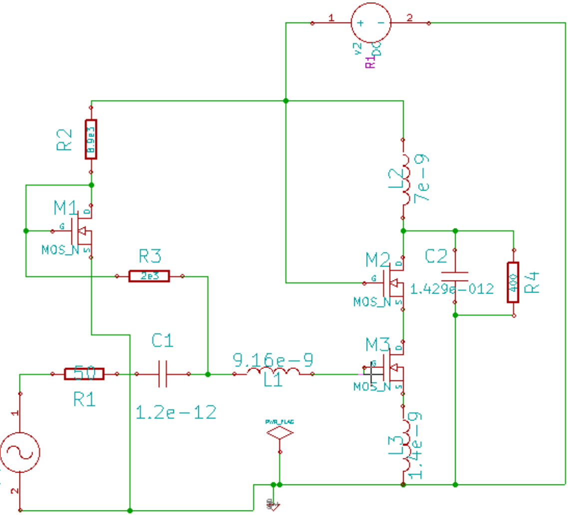


Clearly shows delay converges to a minimum when stage effort is 4.

Low Noise Amplifier Design for Maximum Gain and Minimum Power

- LNA circuit optimised for max gain, min power consumption, with sufficient linearity
- Designed for min Noise Factor
- Resonant frequency $\omega_0 \approx 10^{10} s^{-1}$
- Input impedance, $R_s = 50\Omega$
- Cascode Common Source (CS) LNA with inductive degeneration used
- $0.5\mu m$ CMOS technology

LNA Optimisation Problem



Max. Gain, Min. Power, subject to

$$
\begin{aligned}
Vgs_1 &> V_{th} \\
Vds_1 &> Vdsat \\
Vgs_2 &> V_{th} \\
Vds_2 &> Vdsat \\
Ids &\geq 0.0015A \\
0.98\omega_0 \leq f_{out} &\leq 1.02\omega_0
\end{aligned}
$$

Vgs_i, Vds_i: Voltages of gate-source, MOS Mn_i's drain-source
V_{th}: Threshold voltage, Ids: Drain-source current

Optimisation Results

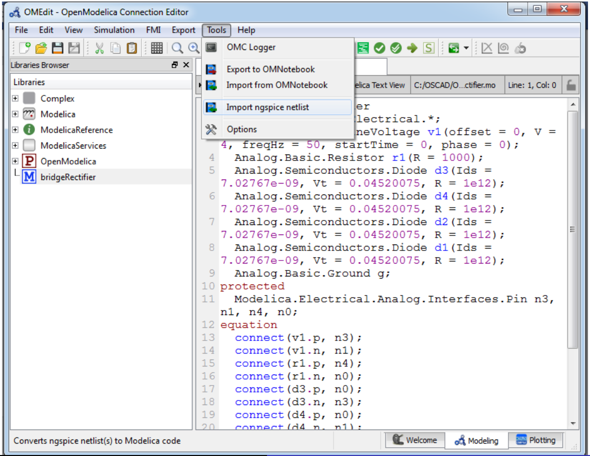
Parameter	Value
L2	7.136nH
C2	1.396pF
w_{nbias}	236λ
$w_{M2,M3}$	2360λ
R2	$9.383k\Omega$
R3	$2.63k\Omega$
NF	2.08
power	10.63mW
gain	9.3

- Performed multi objective optimisation using PSO algorithm
- Output freq: 9.96×10^{10} rad/s, within limits

Integrating Ngspice to Modelica Converter

Integrating Ngspice to Modelica Converter

- API ngspicetoModelica() added to omc
- Tool "Import Ngspice Netlist" added in OMEdit
- Generates Modelica code and opens in OMEdit



Conclusions

- Explanation feature in Oscad enhanced using OM tools
- Feature helps user gain more insights into the circuit
- Design optimisation facility helps design efficient circuits in a short time
- Implemented LaTeX interface to equation generator in OM
- A generic Ngspice netlist to Modelica code converter added in OM suite
- Helps obtain Modelica code from Ngspice netlists, without creating schematic in Oscad
- Several optimisation problems illustrated using enhanced explanation feature in Oscad 2.0

Thanks

kannan@iitb.ac.in