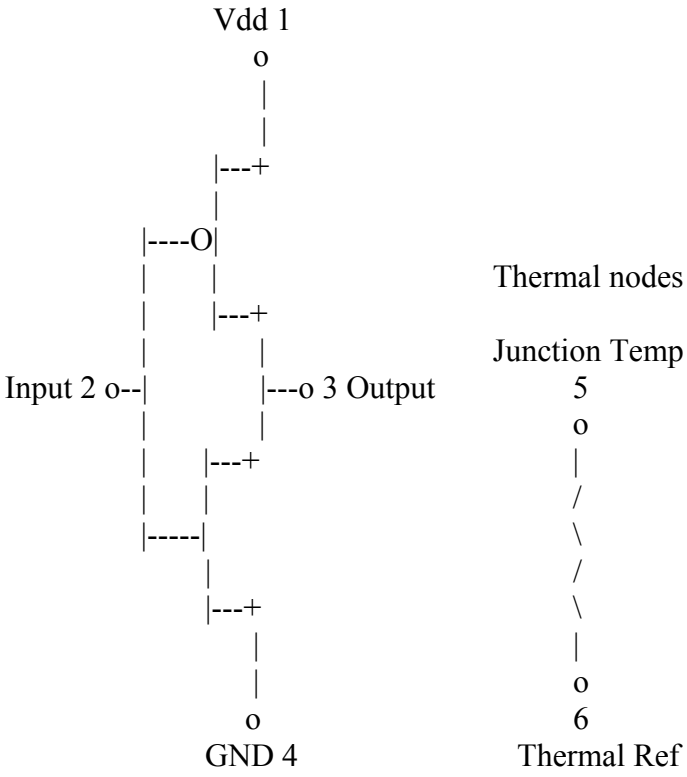
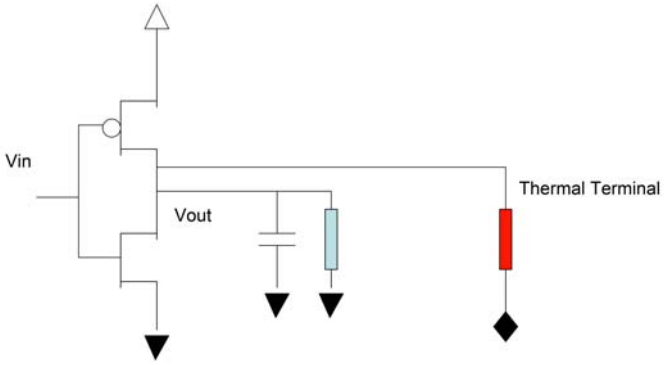
CMOS Inverter with Junction temperature cmosinvt

CMOS Inverter with Junction temperature





Authors: Tony Mulder, Travis Lentz

Description:
This element implements a generic CMOS inverter and calculates junction temperature.

Form: cmosinvt: <instance name> n_1 n_2 n_3 n_4 n_5 n_6 <parameter list>

instance name is the model name

n_1 is the inverter element Vdd node,

n_2 is the inverter element input node,

n_3 is the inverter element output node,

n_4 is the inverter element Ground node,

n_5 is the inverter element Junction temperature node,

n_6 is the inverter element Thermal Reference node,

Parameters:

Parameter	Type	Default value	Required?
vtn: NMOS threshold voltage (V)	DOUBLE	1	no
vtp: PMOS threshold voltage (V)	DOUBLE	-1	no
un: effective mobility of electrons in NMOS (cm^2)/(V-sec)	DOUBLE	500	no
up: effective mobility of holes in PMOS (cm^2)/(V-sec)	DOUBLE	200	no
en: permittivity of gate insulator in NMOS (F/cm)	DOUBLE	34.515e-14	no
ep: permittivity of gate insulator in PMOS (F/cm)	DOUBLE	34.515e-14	no
tox: thickness of gate insulator (cm)	DOUBLE	2e-6	no
wn: channel width of NMOS (cm)	DOUBLE	50e-6	no
ln: channel length of NMOS (cm)	DOUBLE	2e-6	no
wp: channel width of PMOS (cm)	DOUBLE	100e-6	no
lp: channel length of PMOS (cm)	DOUBLE	2e-6	no
td: response delay time (sec)	DOUBLE	0	no
thermal: thermal element flag	BOOLEAN	False	no
tnom: nominal temperature (K)	DOUBLE	300	no

zt: thermal impedance summation (K/W)	DOUBLE	310	no
c1: PMOS GS capacitance (F)	DOUBLE	1e-12	no
c2: NMOS GS capacitance (F)	DOUBLE	1e-12	no
c3: output GS (Miller) capacitance (F)	DOUBLE	1e-12	no
c4: parasitic diode capacitance from output to Vdd (F)	DOUBLE	1e-12	no
C5: parasitic diode capacitance from output to Ground (F)	DOUBLE	1e-12	no
freq: operating frequency (Hz)	DOUBLE	1e6	no
lk: leakage current (A)	DOUBLE	8e-6	no

Example:
cmosinvt:Q1 1 2 3 0 1000 "tref"

Model Documentation: (Start on a new page, Include English and make sure font sizes are consistent. Use figures if necessary. Please avoid using scanned image.)

MOSFET cutoff current I_{ds}, for $V_{gs} < V_t$:

$$I_{ds} = 0$$

MOSFET current factor β:

$$\beta = \mu * C_{ox} * (W / L)$$

MOSFET linear current I_{ds}, for $V_{gs} - V_t > V_{ds} > 0$:

$$I_{ds} = \beta * (V_{gs} - V_t - (V_{ds} / 2)) * V_{ds}$$

MOSFET saturation current I_{ds}, for $V_{ds} > V_{gs} - V_t > 0$:

$$I_{ds} = (\beta / 2) * (V_{gs} - V_t)^2$$

Power dissipation P_D:

$$P_D = (V_{sd,pmos} * I_{sd,pmos}) + (V_{ds,nmos} * I_{ds,nmos}) + (2 * freq * C * V_{dd}) + leakage$$

Junction temperature T_j:

$$T_j = T_{ambient} + (P_D * \theta_{jA})$$

References:

1. Mazen M Kharbutli. fREEDA element moscinv. DEFAULT_ADDRESS"elements/Moscinv.h.html"

Sample Netlist:

```
* DC CMOS Thermal Inverter Test
* DC sweep input voltage from 0 to 5V

.dc sweep="vsource:Vin" start=0 stop=5 step=0.1

.options iniTmp=300

.ref "tref"
.ref 0

vsource:Vdd 1 0 vdc=5
vsource:Vin 2 0

cmosinvt:Q1 1 2 3 0 1000 "tref" thermal=1
res:R2 3 0 r=1000000

***thermal circuit
res:R1 1000 1001 r=6e3
vsource:t1 1001 "tref" vdc=iniTmp
```

```
cap:c1 1000 1001 c=1e-12

.out plot term 3 vt in "CMOS_InvT2_DC.out"
.out plot element "cmosinvt:Q1" 0 it in "CMOS_InvT2_current.out"
.out plot element "cmosinvt:Q1" 3 ut in "temperature.out"

.end
```

Validation:

This thermal CMOS inverter model adds thermal junction temperature calculation to the existing electrical CMOS inverter model (moscinv). Simulations were performed using the existing moscinv model in a netlist with a DC sweep from 0 to 5 Volts at the input terminal. The resulting output voltage and current was recorded. The thermal CMOS inverter model (cmosinvt2) was validated by making a new netlist with the cmosinvt2 model. The resulting cmosinvt2 voltage output and current output was compared to those of the original moscinv model and identical electrical results were obtained. The resulting cmosinvt2 junction temperature data was empirically verified by correlating junction temperature data points with manually calculated results.

The original moscinv model uses a default parameter of wp = 50e-6 cm (channel width of PMOS). If this default parameter is used an erroneous element current output results. The cmosinvt2 model uses a default parameter of wp = 100e-6 cm, resulting in an expected element current output.

Known Bugs:

None.

Credits:

Name	Affiliation	Date	Links
Travis Lentz	NC State University	April 2005	tlentz@hotmail.com
Tony Mulder	NC State University	April 2005	mulder_la@yahoo.com

Introduction:

The purpose of this paper is to describe the implementation and operation of three fundamental digital logic building blocks being modeled with thermal data and as a macro model. These are the initial steps of imbedding thermal performance information in electrical models, and reducing simulation time/complexity with the use of macro models at the same time. Using these ideas, much more complex blocks, entire shift registers or adders for example, could be modeled as macro models with thermal performance.

Moore's Law is threatened by the thermal performance of today's microprocessors. The frequency of commercial processors is limited because it is now becoming difficult to cool them adequately in a cost-effective way for the system-level designers. Thermal information of the circuits, as they are being designed (as opposed to after), is approaching a state of necessity.

The issue comes with the fact that there are no links between thermal and electrical information in today's commercially available simulators. There are no device models that will function in both SPICE, as an example of an electrical simulator, and FLOWTHERM, as an example of a thermal simulator. With the advent of fREEDA™, there is now a simulator that has the ability to imbed any information, electrical, thermal, mechanical, etc. though the use of state variables, which can be defined by the modeler. This opens up virtually limitless possibilities in modeling and delivers a platform in which one is able to capture accurate, instantaneous thermal information along side the voltage, current, and charge.

Another concern involves the time it takes to simulate accurate models in today's multi-million transistor chips as it would involve multi-million model files if using discrete devices. A macro model, capturing the essential performance and function of a block of transistors, hastens the simulation process by decreasing the number of terminals/nodes in the circuit. Even a model for a push-pull inverter, as opposed to modeling two discrete MOS transistors, reduces the number of terminals to keep track of, by 50%. A two-input NAND gate can be modeled with six terminals vs. the sixteen that would require bookkeeping with the use of discrete devices.

Presented here are the electro-thermal Inverter, NAND, and NOR digital building blocks. By keeping track of the power dissipated by the device and understanding the sum of thermal impedances found in the device, instantaneous junction temperatures can be derived. These temperatures can be used as a large chip is being designed to redistribute functional blocks as part of better distributing the heat of a chip, for example. Designers can now start to think in terms of thermal supply in the way that current and charge is though of in circuits today. By dispensing the hot spots of a large circuit more intelligently, not only can higher frequencies be realized, more reliable circuits can be developed earlier in the design process.

Initially accurate CMOS macro models had to be built for push-pull the Inverter, two-input NAND and two-input NOR gates. Within the fREEDA^{TM} directories set up for NCSU students taking ECE718, these models existed, but were not accurate. PMOS currents were not sufficient over the full range of operation, and with investigation it was found that device dimensions between the NMOS and PMOS were not properly radioed. After device dimensions were attuned to adjust for the difference in carrier velocities, the macro models produced accurate behavior.
A DC sweep from 0V to 5V of the inverter produced the output seen in Fig. 1 along with the original output from the incorrect file. The output currents were drastically impaired because of this inaccuracy. The original and corrected current outputs are illustrated in Fig. 2.
(See Appendix A for example net list file)

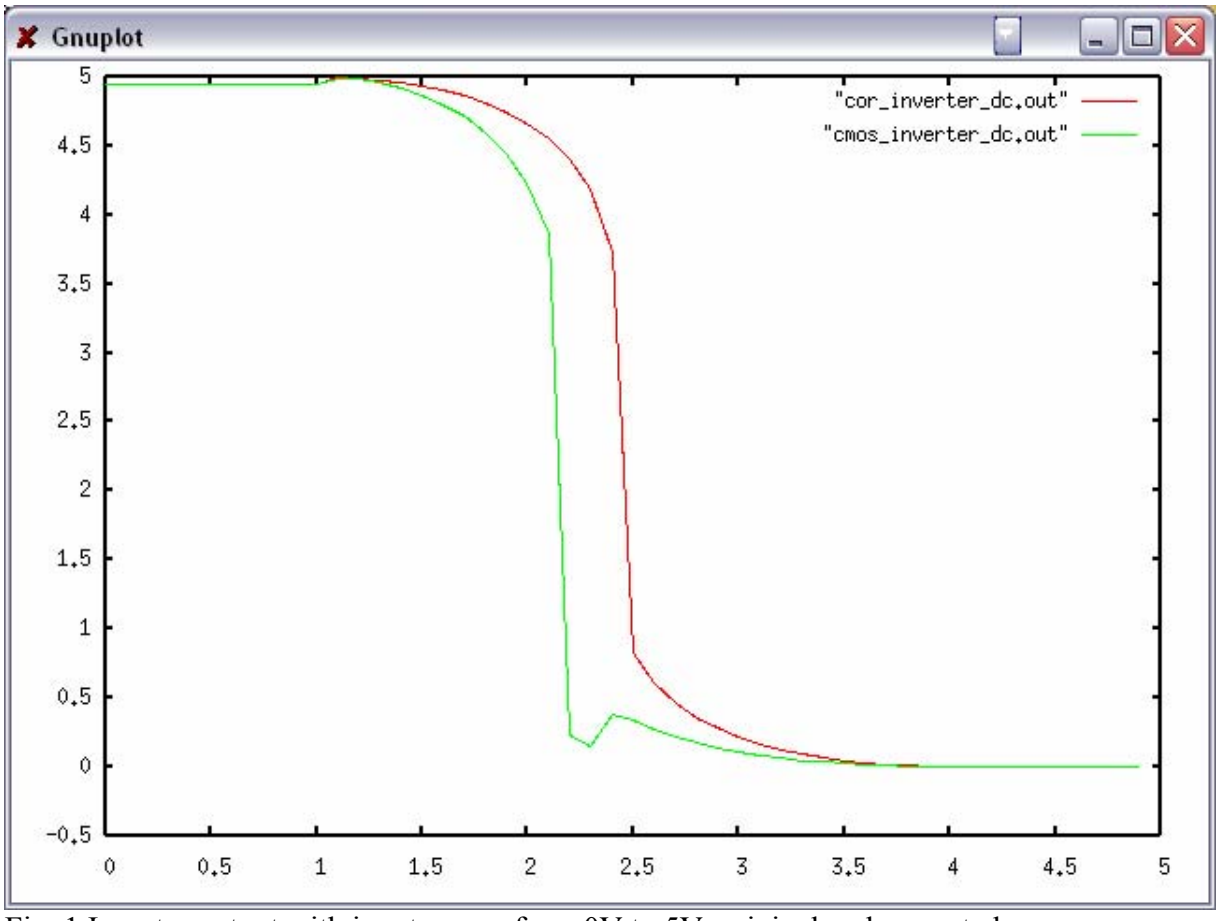


Fig. 1 Inverter output with input sweep from 0V to 5V; original and corrected responses.

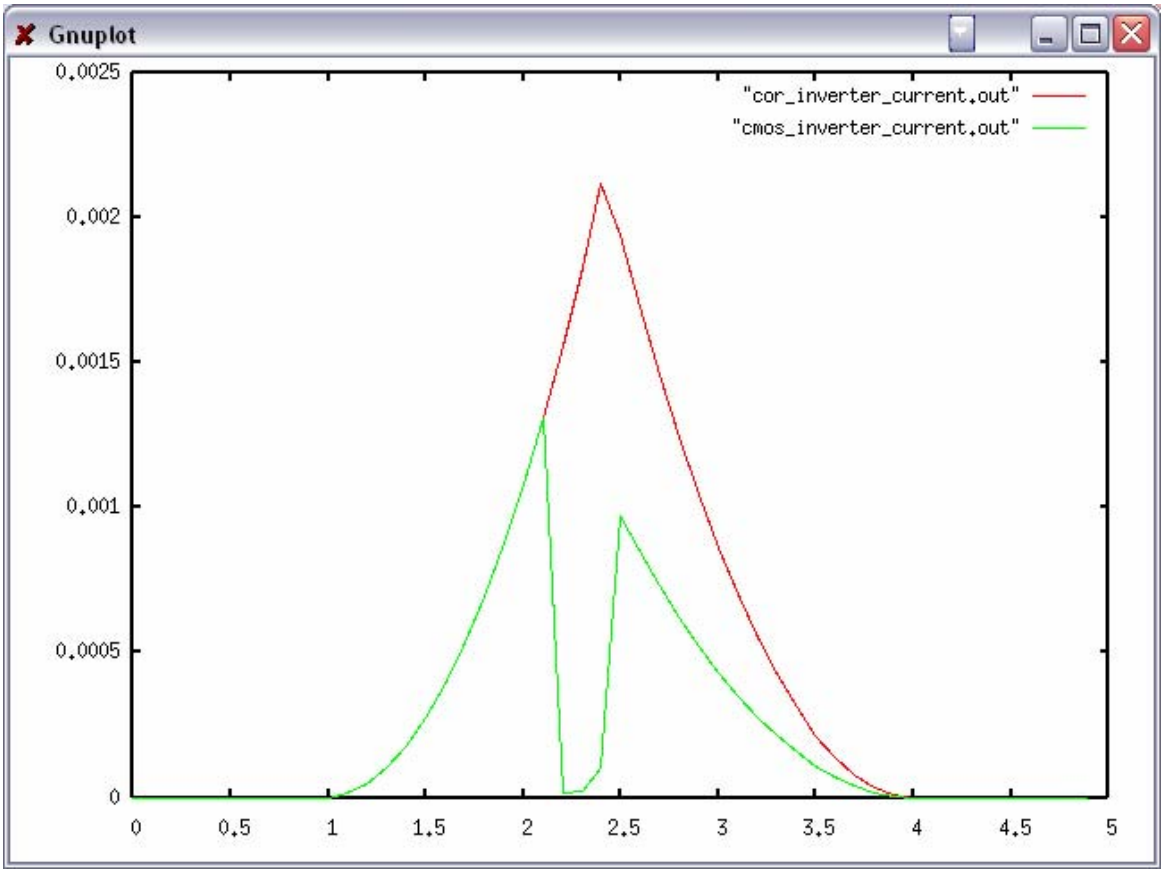


Fig.2 Inverter output currents during switching, before and after correction.

Thermal information was extracted by starting with the power dissipation of the logic block. There are two primary sources of power dissipation in this model: quiescent (when the circuit is not switching, but current is leaking) and dynamic power consumption (internal capacitance transient dissipation and current spiking during transition). The most significant contributor is the power dissipated from the current spikes during the transition wile both the nMOS and pMOS devices are in saturation at the output. This is given by, $V_{dsn}*I_{dsn} + V_{dsp}*I_{dsp.}$ The next dynamic power contributor comes from charging and discharging the internal (and external, though these are handled through the netlist, not in the model) capacitance and is frequency dependent. It is given by, $(C_{parasitic})V_{CC}^2 f$. These overlap and Miller capacitances, along with frequency are passed in the model through parameters. The user must know the frequency at which the device will be operated to take advantage of this level of accuracy. The leakage current is given by, $I_{CC,}$ and the quiescent power so given by, $I_{CC}V_{CC}$.

Equation 1: $$P_D = (V_{dsn}*I_{dsn} + V_{dsp}*I_{dsp}) + (C_{parasitic})V_{CC}^2 f + I_{CC}V_{CC}$$

The majority of the power consumption in a CMOS circuit comes from the current spiking during switching as will be seen shortly. The quiescent power and internal transient capacitance charge/discharge power dissipation is not seen given that the change in temperature due to current spikes is tens of Kelvin while the former is an order of

magnitude smaller until very high frequencies or very high capacitance is observed (which will be explored later).

The junction temperature is now found by including the sum of all the thermal impedances multiplied by the power dissipated which is then added to the ambient temperature. Thermal impedance depends on package types, size of the die, attachment of the die to the package, and organization of the high-power-dissipation elements of the die. These boundary conditions must be understood and the sum of the thermal impedance, zt, supplied to the model by the user or used as part of the load in the netlist (see Appendix A).

Equation 2: $T_j = T_{AMB} + (P_D * \theta j_A)$

Where $\theta j_A = \theta j_C + \theta j_{CA}$

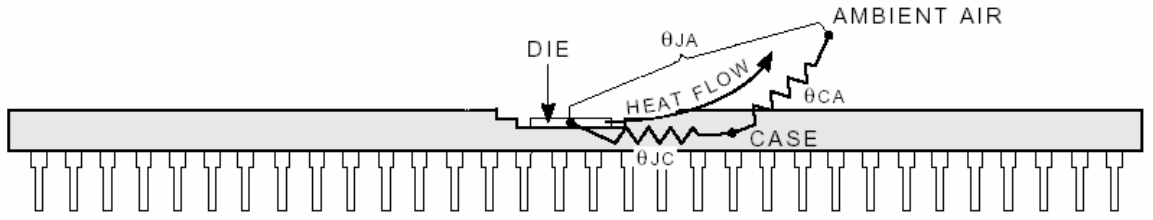


Fig. 3

From this, the instantaneous junction temperature during the switching of the circuit can be derived from the model as seen in Fig4. Here load resistance/capacitance is being varied illustrating the direct correlation between dissipated power and junction temperature. This can be does just as varying the model's thermal impedance can. Note that here only the dynamic power due to switching is being observed. The scale at which the plot can capture the entire rise and fall of the temperature is too great to differentiate between the power due to current spikes and the quiescent and transient capacitive power dissipation.

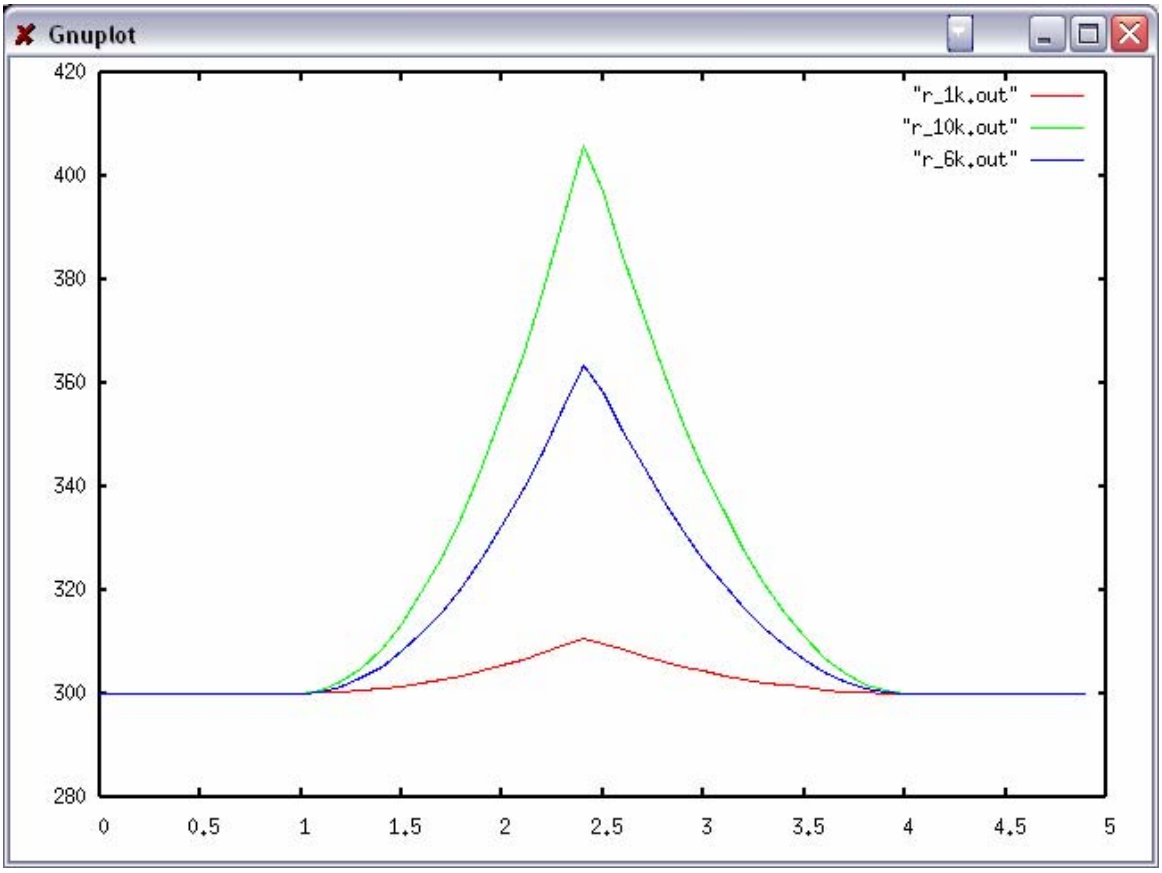


Fig. 4 Inverter Junction Temperature during switching with increasing resistance

Next, transient simulations were run to demonstrate the instantaneous junction temperature over time. As expected, it follows the current spikes during the phase of operation were both MOS devices are on at the same time during the switch. See Fig. 5

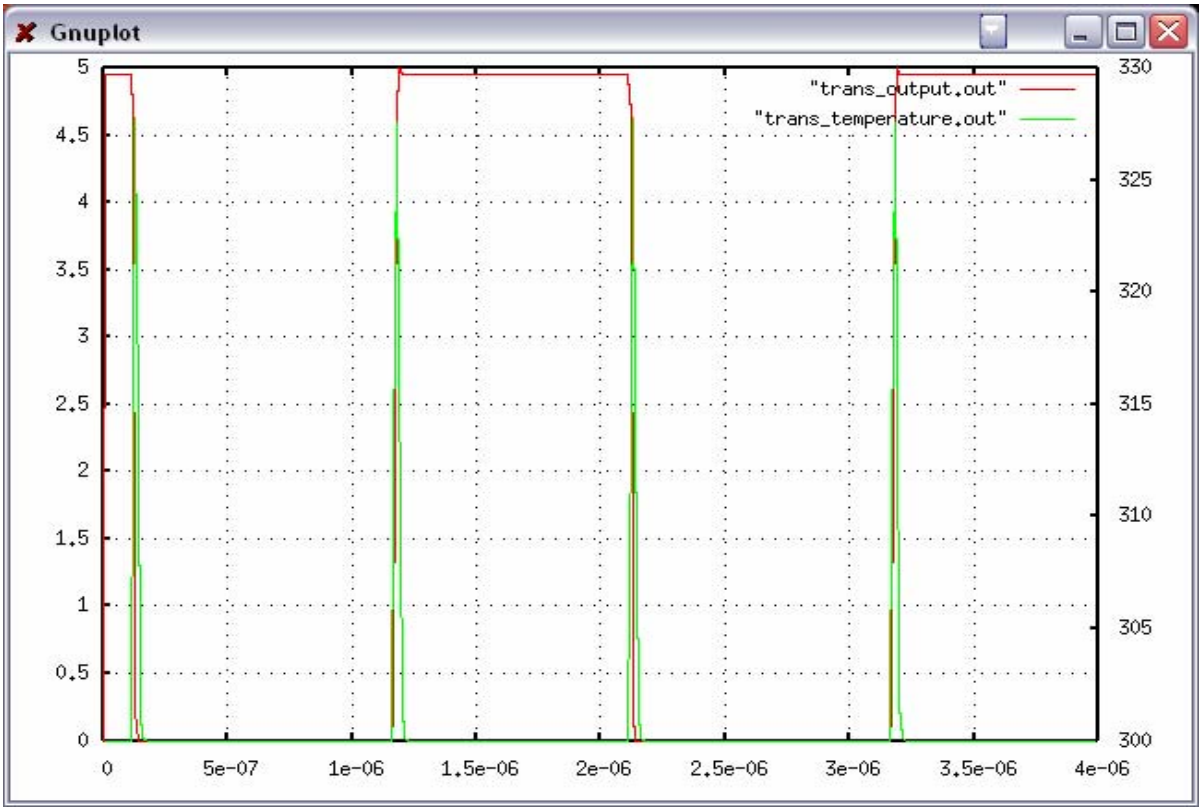


Fig. 5 Output with temperature spikes during the switching of the MOS inverter.

To illustrate the effects of the quiescent and transient capacitive power dissipation and quiescent power, the capacitive and leakage current values were exaggerated to illustrate their effects on the same scale. A small increase in temperature is seen in the quiescent state of the circuit along with a temperature rise due to the extra power required to charge and discharge the (exaggerated) internal capacitances. See fig. 6

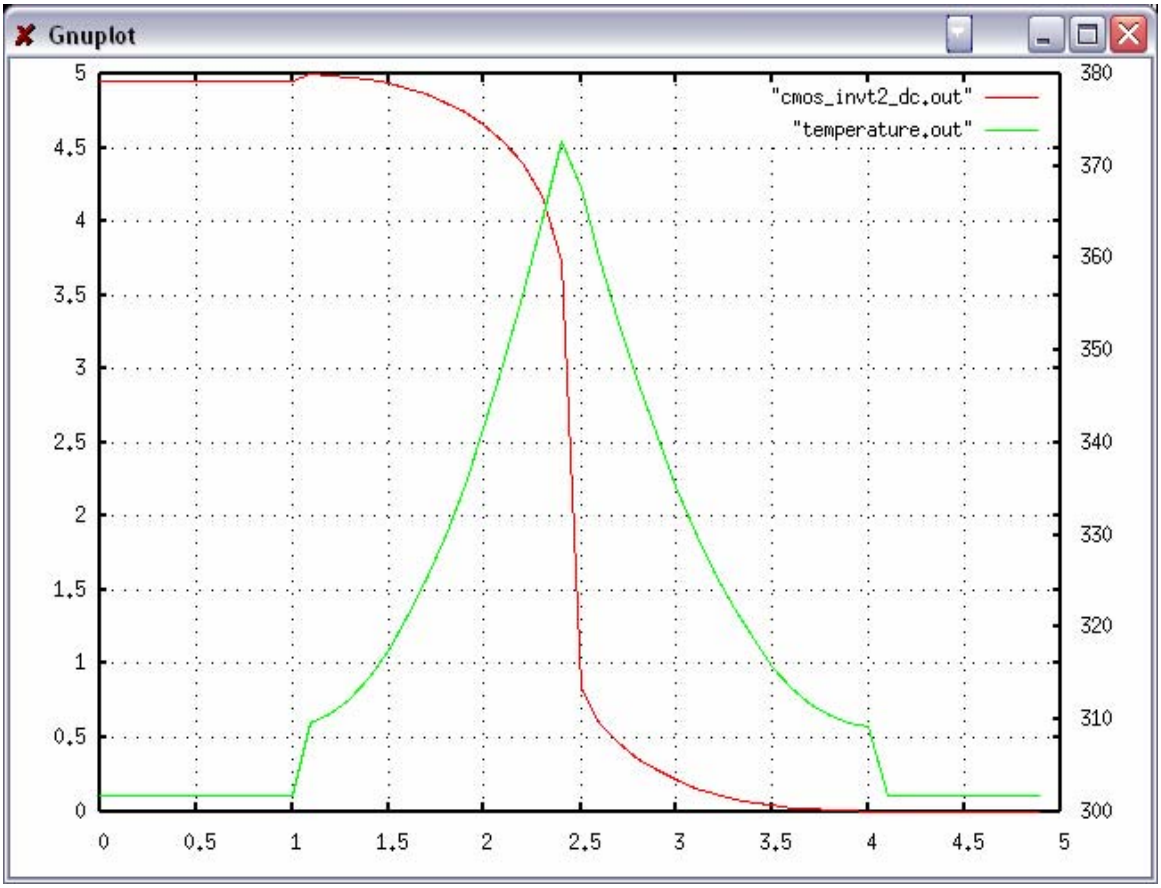


Fig. 6 Effects of internal capacitance and leakage current on temperature

Now we turn to implementation into fREEDA™.

This thermal information is added to the model by creating an additional state variable to represent the junction temperature and creating an additional set of terminals to monitor and provide the output of these temperatures. For details on creating models in fREEDA™, refer to chapter 3, 'Adding Nonlinear Elements to fREEDA™' in version 1.1.1 of the fREEDA™ Programmer's Guide.

Electrical models contain state variables representing voltages across and currents through the model's terminals relative to their corresponding reference terminal. fREEDA allows the use of any device parameter to be represented by state variables, including thermal information, as in this case. Depending on the version of the createTape() function used, a state variable can be added to any of the vectors created for state variables, their 1st and 2nd derivatives and delayed versions of these state variables. Each state variable is mapped in order to each non-reference node in the model. The assignment of the voltage variables vp[] and current variables ip[] in this case are:

x[0] : Vdd x[4] : d(Vdd)/dt
x[1] : Vin x[5] : d(Vin)/dt
x[2] : Vout x[6] : d(Vout)/dt

x[10] : Vout(t-td)

vp[0] : Vdd	ip[0] : Isdp
vp[1] : Vin	ip[1] : Ig
vp[2] : Vout	ip[2] : Iout
vp[3] : temp	ip[3] : power

Refer to the source code for more information on the model variable assignments.

With the addition of the thermal terminals, the electrical reference terminal (representing 0V, or ground potential) must be separated from the thermal reference terminal (representing 0 Kelvin). This is accomplished with the ability of fREEDA™ to recognize local reference terminals and is implemented in the getLocalRefIdx() function. A single node and reference terminal was required in this case. Again, refer to chapter 3 of the programmer's guide for more details on how local reference terminals are implemented in fREEDA™ .

The evaluation function pulls all this together and manipulates the input parameters and state variables for outputs that can be monitored though the nodes called out in a net list, much like SPICE. In this case, the standard 1st order current equations for a MOS transistor in all modes of operation were used.

- Cutoff

$$I_{ds} = 0 \qquad V_{gs} < V_t$$

- Linear

$$I_{ds} = \beta \left(V_{gs} - V_t - \frac{V_{ds}}{2} \right) V_{ds} \qquad V_{ds} < V_{dsat}$$

- Saturation

$$I_{ds} = \frac{1}{2} \beta \left(V_{gs} - V_t \right)^2 \qquad V_{ds} > V_{dsat}$$

$$\beta = \mu C_{ox} \frac{W}{L}$$

The currents were then used for the prior explained power and junction temperature derivations (see equations 1 and 2) for the compete electro-thermal description of the logic blocks.

Summary:

The goal here was to build a model that would derive the most fundamental thermal information in the most basic model for faster simulation as it would be considered a part

of a very large network of transistors. Continued work could include making the device more accurate. Non-ideal characteristics of MOS devices would be included even if as simple as channel-length modulation, body effect, and velocity saturation. Feeding the instantaneous junction temperature fluctuations back into the model to precisely vary the channel current through changes in both threshold voltage and carrier mobility due to temperature differences, would make this model even more useful. Finally, expanding the power consumption beyond current spiking during switching to include internal capacitance transient dissipation would give a more complete model to work with.

Appendix A Example DC sweep netlist files

Example 1

```
* DC CMOS Thermal Inverter Test
* DC sweep input voltage from 0 to 5 V

.dc sweep="vsource:Vin" start=0 stop=5 step=0.1

.options iniTmp=300

.ref "tref"
.ref 0

vsource:Vdd 1 0 vdc=5
vsource:Vin 2 0

moscinvt2:Q1 1 2 3 0 1000 "tref" thermal=1
res:R2 3 0 r=1000000

***thermal circuit
res:R1 1000 1001 r=6e3
vsource:t1 1001 "tref" vdc=iniTmp
cap:c1 1000 1001 c=1e-12

.out plot term 3 vt in "CMOS_InvT2_DC.out"
.out plot element "moscinvt2:Q1" 0 it in "CMOS_InvT2_current.out"
.out plot element "moscinvt2:Q1" 3 ut in "temperature.out"

.end
```

Example 2

```
* DC thermal CMOS NAND Test

.dc sweep="vsource:Vin" start=0 stop=5 step=0.1

.options iniTmp=300

.ref "tref"
.ref 0

vsource:Vdd 1 0 vdc=5
vsource:Vin 2 0
vsource:Vdd2 3 0 vdc=5

moscnandt:nand 1 2 3 4 0 1000 "tref" thermal=1
res:R 4 0 r=1000000

***thermal circuit
res:R1 1000 1001 r=6e3
vsource:t1 1001 "tref" vdc=iniTmp
cap:c1 1000 1001 c=1e-12

.out plot term 4 vt in "CMOS_NANDT_DC.out"
.out plot element "moscnandt:nand" 0 it in "CMOS_NANDT_current.out"
.out plot element "moscnandt:nand" 4 ut in "temperature.out"

.end
```

Example 3

```
* DC thermal CMOS NOR Test

.dc sweep="vsource:Vin" start=0 stop=5 step=0.1

.options iniTmp=300

.ref "tref"
.ref 0

vsource:Vdd 1 0 vdc=5
vsource:Vin 2 0
vsource:Vdd2 3 0 vdc=0

moscnort:nor 1 2 3 4 0 1000 "tref" thermal=1
res:R 4 0 r=1000000

***thermal circuit
res:R1 1000 1001 r=6e3
vsource:t1 1001 "tref" vdc=iniTmp
cap:c1 1000 1001 c=1e-12

.out plot term 4 vt in "CMOS_NORT_DC.out"
.out plot element "moscnort:nor" 0 it in "CMOS_NORT_current.out"
.out plot element "moscnort:nor" 4 ut in "temperature.out"

.end
```

Appendix B Example transient netlist file

Example 1

```
* CMOS Inverter Test with Thermal terminals
* Transient Response

.tran2 tstop=4e-6 tstep=10e-9 out_steps=1

.options iniTmp=300

.ref "tref"
.ref 0

vsource:Vdd 1 0 vdc=5
vpulse:Vin 2 0 v1=0 v2=5 per=2e-6 pw=1e-6 tr=0.05e-6 tf=0.05e-6
*vpulse:Vin 2 0 v1=0 v2=5 per=2e-7 pw=1e-7 tr=0.05e-7 tf=0.05e-7

moscinvt2:Q1 1 2 3 0 1000 "tref" thermal=1 td=0.1e-6
res:R2 3 0 r=1000000

***thermal circuit
res:R1 1000 1001 r=6e3
vsource:t1 1001 "tref" vdc=iniTmp
cap:c1 1000 1001 c=1e-12


.out plot term 2 vt in "trans_input.out"
.out plot element "moscinvt2:Q1" 0 it in "trans_current.out"
.out plot element "moscinvt2:Q1" 3 ut in "trans_temperature.out"

.end
```