Intusoft Newsletter

Personal Computer Circuit & System Design Tools

Newsletter Issue #86, Nov 2016

In This Issue	
page	**Story**
1	Modeling SiC Schottky diodes
3	Controlling the Internet of Things
5	A Generic Power Factor Correction, PFC, model

Modeling SiC Schottky diodes

Recently, analog behavioral models (ABM)-for Silicon Carbide, SiC, devices have been published [1],[2],[3]. The main reason cited for using ABM instead of built-in SPICE 3 diode model is to include dynamic thermal modeling. Moreover, none of these models are described in sufficient detail to reproduce the Authors data. The approach taken in [4] is reproducible, albeit, values for EG and the Rs temperature coefficient don't agree with our measurements. We, at Intusoft, back in 1988 have shown how to do this in newsletters 10 and 11 (search "thermal node" in our PDF VIEWER). The downside of doing this is increased complexity and poor convergence properties. The built-in SPICE models handle the problem, as long as the EG and XTI parameters are included in the SPICE diode model and the steady state temperature, "Temp" is set as the model parameter ("Temp" is a SPICE3 enhancement). Here is why problems occur using the ABM thermal model for every-day use:

1. Behavioral models use abrupt switching to change operating regions. If handled improperly, this state switching can oscillate between transient operating points and never properly converge. The SPICE built-in models have been proven to work with experience of millions of successful circuits simulated. These models include heuristic mode switching algorithms not used in the cited ABM's.

2. The device temperature requires many seconds, perhaps several minutes, to come to steady state. Generally, circuit response only needs a fraction of a second of simulation time to give correct answers. So its best to solve the thermal problems separately and then include device temperatures in the individual device models. (It is important for simulations to take less than several seconds in real time so that human interaction can efficiently occur in an iterative design process. It is not all right to claim that it's OK to take "forever" to get the "right" answer when a "very close" result is possible in several seconds)

There are cases requiring a thermal model and some of the problems can be mitigated by reducing the thermal time constants, for example, turning hours into seconds. This has been successfully done in our Battery library and the Power Integrations, TNY device models, back in 2007.

Making a SiC model using SPICE 3 built-in model: The model parameters for SPICE 3 are set up so that XTI and EG have no affect at the model temperature, Tnom, which defaults to "room" temperature, about 300 Deg K or 27 Deg C. So you can go ahead and use your favorite model makers; we use our own SpiceMod. Then you need to add the "correct" values for EG and XTI. Neither of these values is readily available in the literature and many reported values are completely incorrect, ignoring the fact that the commercial SiC devices are metal barrier; that is, Schottky devices. The barrier voltage or EG is significantly lower for metal junction than for pn junctions. The lower forward drop allows for significantly higher power supply efficiency, and in the case of Silicon, sharply reduce reverse recovery time. Reverse or minority carrier recovery is not an Issue for a Silicon Carbide because it is a majority carrier device. Some pay-for references claim to have the correct answer [5]; however it's less expensive and sure fire to buy some devices and measure the parameters.

To find EG, you measure the forward drop at low current at 2 widely separated temperatures. Room temperature and 175 DegC are easily measured using an oven. Here's what we get using a CREE C3D02060F

```
I(mA)          T(Deg. C.)          Vf(Volts)
Device 1
1              23.1                .776
1              160.8               .548
Device 2
1              23.                 .766
1              153.4               .551
Device 3
1              23.1                .776
1              159.4               .540
Device 4
1              24.3                .766
1              157.3               .549
EG(Average)=1.22
```

Plugging EG into our simulator, we find it should be corrected to 1.16 because it's not a straight-line extrapolation to 0 Deg K, and SPICE3 accounts for bending of the EG vs. Temp curve.

IKF is a bit more difficult; we will use the default, 3, for now and see if any adjustment is needed.

SPICE 3 treats bulk resistance, RS, in the diode model [6] as a constant (no temperature coefficient), resulting in incorrect results above 25% of device rated current for SiC devices. The bulk resistance temperature coefficient can be added using an external resistor. But, here's the rub: There is no single behavior for this resistance temperature coefficient. It will vary depending on device construction, material and impurity levels. It generally runs around 6% per Deg C for Silicon devices. Extrapolating the resistance to 0 Deg K results in a negative resistance. The simplest way to avoid the problem is to introduce a second order term that has a value of TC1/273. Then the resistance is given by:

R(T) = R(To=27 Deg C) * (1+TC1*(T-To) + TC1/273*(T-To)^2)

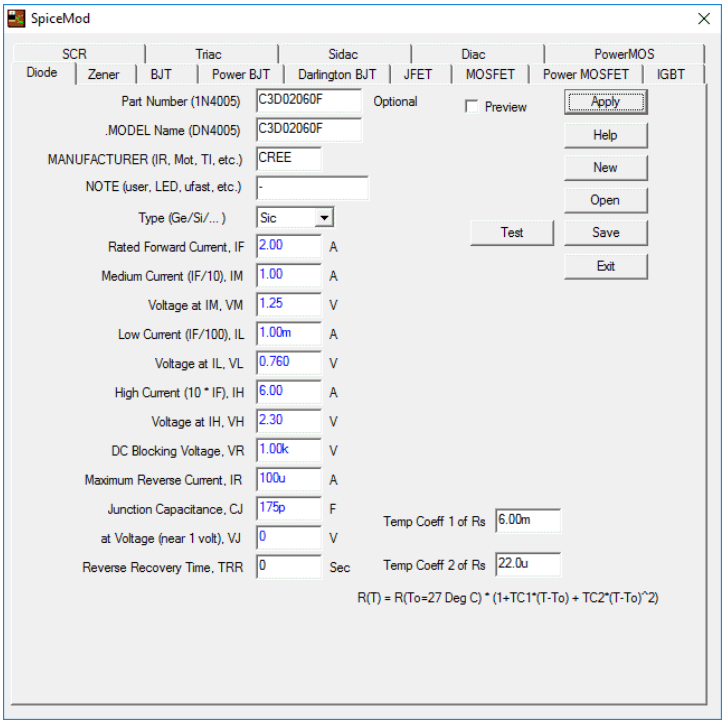
So at least we can make believe it's accurate all the way down to absolute 0. At any rate, it should be accurate over the range of –55 Deg. C to +175 Deg. C for which the devices are specified to work. For CREE devices, we use the following:

R(T) = R(To) + 6m*(T-To) + 22u*(T-To)^2)

Putting it altogether, we can compare the results against the datasheet specification. Some have reported XTI=2 should be used for Si Schottky diodes [7]; however, varying XTI from 1 to 3 has very little affect, so we will just leave it at 3.

Unlike the other models referenced previously, this model relies on first principles to extrapolate thermal behavior. All of the others rely on matching the TC1 and TC2 "constants" to correct bad values for EG, IS and N.

New SpiceMod does it all for you: Now that you see how it's done, we've made it even simpler by incorporating the results into SpiceMod. Just fill out the data sheet parameters and select "SiC" from the "Type" drop-list and the models will be wrapped into a subcircuit that can be immediately included in your library. Here is an example using the CREE C3D02060F data sheet. Note: the datasheet low current voltage level isn't very readable, so we used measured data from the test results of the devices we used to get EG. You can extrapolate it to other SiC diodes by increasing the Area multiplier in our model in proportion to the rated current, then using the simulated VF in the new SpiceMod spread sheet.



We will post a You Tube video showing how this works for adding models to our products and to PSpice. SpiceMod connects to our simulator and library manager when run with our products. For other SPICE 3 compatible simulators, we produce a .lib file that you must change into the library format used by your simulator.

[1] SiC Schottky diode electrothermal macromodel
Francesc N. Masana
Departament d'Enginyeria Electrónica (DEE), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
Mixed Design of Integrated Circuits and Systems (MIXDES), 2010 Proceedings of the 17th International Conference

[2] Parameter extraction sequence for silicon carbide schottky, merged PiN Schottky, and PiN power diode models T.R. McNutt Arkansas Univ., Fayetteville, AR, USA
Power Electronics Specialists Conference, 2002. pesc 02. 2002 IEEE 33rd Annual

[3] SiC Merged PiN and Schottky (MPS) Power Diodes Electrothermal Modeling in SPICE Authors: A. Lakrim, D. Tahri
World Academy of Science and Technology, International Journal of Electrical, Computer, Electronic and Communication Engineering Vol. 8 No. 8, 2014

[4] DC characteristics of the SiC Schottky diodes
W. Janke1 / A. Hapka1 / M. Oleksy1
Department of Electronics and Informatics, Koszalin University of Technology, 2 Sniadeckich St., 75-453 Koszalin, Poland1
Bulletin of the Polish Academy of Sciences Technical Sciences
The Journal of Polish Academy of Sciences Volume 59, Issue 2 (Jun 2011)

[5] Barrier height determination of SiC Schottky diodes by capacitance and current–voltage measurements C. Raynaud1, K. Isoird1, M. Lazar1, C. M. Johnson2 and N. Wright2 . Appl. Phys. 91, 9841 (2002); http://dx.doi.org/10.1063/1.1477256

[6] http://bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE/UserGuide/overview_fr.html

[7]http://www.acsu.buffalo.edu/~wie/applet/spice_pndiode/spice_diode_table.html

Controlling the Internet of Things

Using a web browser to control IoT devices:

You may have used a WiFi router that used an IP address to access and program the router. So, how does that work and can I do the same for my personal devices? That's what we will explain in the following articles.

Why use a web browser as a controller: The "smart phone" or mobile device is a powerful hand held computer available to everyone. Using these devices eliminates the need for specialized controllers, thereby reducing the cost and making certain applications viable. There are many different device operating systems requiring many application if the devices are programmed in their native language. However, a single application using a web browser can operate on ALL hand held devices, desktops, tablets and laptops that have an Internet browser. And that's all of them!

Anatomy of a web server: A web server responds to a header request (issued by a web browser) by sending a response header and an HTML page back to the requester. You can see what these header requests and responses look like by using Charles Proxy, a low cost proxy server. Two kinds of information concerning the request are sent and they are POST and GET. Below is the Charles Proxy view of a login. Notice that this man-in-the-middle proxy was placed there voluntarily for debugging. The actual login credentials are sent as a POST and are available using the <raw> view. Scary to think that malware can do the same!

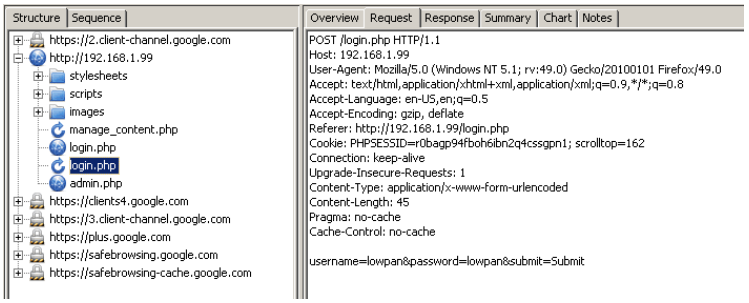


Figure 1, Charles Proxy display of raw request header

The GET information is sent in the URL bar after the page name using name=value syntax as follows:
…login.php?name1=value1&name2=value2…
where[=value] is optional.

The POST syntax is similar but is only seen at the end of the request header. The web browser cannot view the POST information, but it is transmitted over the Internet. The Internet connection can be encrypted using https (Charles can decode it!) or by sending it over a secure shell (ssh). For our purposes, we will not be concerned with Internet security issues because we will be using an

Intranet connection that is ordinarily not available outside of your Local Area Network, LAN.

The basic HTML is stateless; that is, given the same GET and POST requests, the HTML will always produce the same result. But web pages need to preserve a state; for example, when a form is submitted and found to be incorrect, it is undesirable to have the user re-enter all of the form data. The state must change to include the previously entered date that was correct.

A server-side language such as PHP can save the state permanently in its database/memory, or temporarily in its $_SESSION variable or regenerate the form by sending back the POST fields in the HTML response. The client can save data using Java Script, including AJAX and JASON or saving state in its cookie or in client memory.

Using a web server as an IoT controller has unusual properties:

1. There are very few clients, frequently only one.
2. The client issues commands using HTML forms
3. The server controls IoT devices based on client commands

The disadvantage of client side programming is that code is public. But it offloads the server and makes GUI based interfaces work without sending mouse clicks and/or motion over the network.

For IoT control, it makes sense to do most of the work in the server because it will ultimately control the IoT devices. We will use PHP as the server-side language. PHP is open source (free) and can be programmed with a text editor. You can download an Apache2 web server with PHP and MySQL database support for a variety of operating systems. For MAC and Windows, XAMPP is a good choice. For Linux, you can fetch the code from the repository that connects with your flavor of Linux. Using Debian for example, enter at the command line “sudo apt-get install apache2 –y”.

PHP is similar to the C programming language without requiring strong type casting. Variables are preceded with a $ sign and are coerced into the correct type. For example $x=1.2e3 or $s = “myname” or $n=5 will automatically make these floating point, string and integer respectively. Types can change on the fly and be assigned based of function relationships, as in $y=$n*$x. The suite of PHP functions is extreme! Just browse for “ string copy php syntax” or whatever else you might need. PHP can be either procedural of object oriented. We’ll stick to procedural because there are to many things PHP can’t do or does differently than a complete oops language like C++. The PHP array is amazing. It’s used to hold information similar to that found in C program structs. Browse for “array php syntax” to learn how they work.

Web Server as a Device Controller: Low cost single board computers; for example, the Raspberry PI (rpi), costing less that $30 can host an Apache2 server along with PHP and MySQL. Configured as a “headless” computer, it can be controlled over a network using a tightVNC viewer. Adding a case, power supply and RG45 adapter cable and possibly a USB WiFi brings the hardware cost to around $50 for the DIY enthusiast. TightVNC is a free Java application allowing you to view the remote desktop on your workstation. The rpi can control IoT device directly or it can be a gateway device for controlling a low power personal area network, lowpan. Lowpan networks include Zigbee, MiWi, Z-Wave and many others. These networks are characterized by lower bit rate transmissions, thereby, requiring less standby power. For an automated household with hundreds of IoT devices, a lowpan network is necessary to bring the standby power down to a reasonable level (10 to 100mW per device). A USB lowpan transceiver can cost as little as $10 with an even lower cost transceiver operating on each IoT device.

Getting started with your own IoT controller: Besides the rpi, you will need a serious toolset. There is no getting around having a integrated development environment (IDE). The complete toolset we recommend is:

1. phpStorm, IDE with Xdebug
2. GIT version control system
3. TightVNC viewer
4. Apache2 server with PHP and MySQL
5. Charles Proxy
6. Lynda.com

Items 1 and 5 are pay-for items. Everything else is open source. Using an rpi has great documentation advantages. The rpi community has already answered most questions you may have. PHP and MySQL offer the same level of online expertise, giving you the power to create your applications.

You can use Lynda.com to learn how to use PHP and MySQL to create a web page, and moreover create a web interface that can grow dynamically as you add content.
Modern web pages are easily created using phpStorm, largely because the styles have been separated from content using cascading style sheets, CSS. Our next article will concentrate on building a content management system, CMS, style web application. CMS allow the web page to grow automatically as you add more IoT devices.

The GIT version control system is unique in that it is a distributed system. You don’t need to checkout a file. Just make your changes and use the GitHub website to archive your changes. If others work with you, then GIT can easily merge changes, almost always without any interaction as long as you keep it current with frequent “git push” commands. If a colleague made changes, using “git pull” merges the changes into you archive. Followed by a “git push” to make the archive current. Even if you work alone, GIT makes sure all of your history is saved so that you can rollback if you screwed up your working copy. GitHub is free if you make your archive public. You still control users ability to push changes into your archive.

A Generic Power Factor Correction, PFC, model

Theory: A PFC is used to produce an apparent resistive load (Req) to AC mains (Vmains), and convert the pulsating AC input power to a constant DC output. This is accomplished by forcing the primary side current to be Vmains/Req. A single stage controller filters the rectified output using a capacitor. The output voltage becomes a low ripple DC as the capacitor value is increased.

Some of control laws that can be used are:

1. Constant input power
2. Constant output voltage
3. Constant output current

The model: PFC's use a number of different switch mode topologies. The result is a different switching current signature at the input and output; however, the average values all obey the same theory.

Imains = Vmains*Geq
Iload = abs(Imains)* vmains/v(vload))

Constant input resistance control is accomplished with

Geq = Pavg/ average(vmains^2) == constant value

This, simplest of all control laws, where Geq is constant is illustrated below in Figure 2.

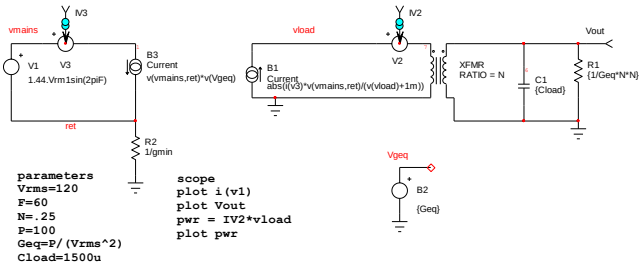


Figure 2, IsSpice4 PFC model for input current control

The transformer model is a "DC" transformer with the ratio controlling the output voltage level.

Other control laws require a feedback loop to force control of another voltage or current. Analog controllers will be devices the us a P-I controller, whose gain is given by:

GAIN = (P + I/s)

And applying the output to B2. These analog controllers and a deadbeat controller, applicable to digital control, will be modeled in our next newsletter.

The mains and capacitor currents contain a switching signature that depends on the circuit topology. This signature can be superimposed with the average model to allow evaluation of snubbers and EMI filters. Modeling these affects will be discussed in our next newsletter.