HOT SPOT SIMULATOR MANUAL

OWNER : TECHNOLOGY DEVELOPMENT

LOCATION : N.A.

REVISION STATUS SUMMARY

Revision	Requestor	Date	Request Number	Pages	Description
1.0	Bart Desoete	02-06-2006			First release of the tool

This page is intentionally empty.

TABLE OF CONTENTS

1 INTRODUCTION

The thermal simulation tool *Hot Spot Simulator* has been developed as a fast and flexible alternative to commercial simulators, which are usually based on finite-element methods. One of these simulators, called *Flotherm*, has been used within the company, but requires a substantial effort by trained people, and usually has a quite long simulation time.

The *Hot Spot Simulator* tool is able to calculate temperature as a function of time and position on the die surface. Thereby the user can specify any number of rectangular power sources on the die surface, each having an arbitrary power waveform as a function of time. The tool is able to generate both temperature distributions across the die area, and temperature evolutions as a function of time.

The calculation of temperature is based on the Green's function solution of the heat diffusion equation. This approach enables a fast simulation. However, this approach inevitably is based on a number of assumptions:

- The rectangular power sources are infinitely thin and located at the surface of the die.

  This assumption is in most cases a good approximation of reality, as the depth and thickness of the power sources is generally small enough compared to the lateral dimensions.

- The power is homogeneously generated across the power source area.

  This assumption neglects the fact that power is usually generated in stripe structures. However, it does not introduce too large errors, especially for structures with a large number of stripes.

- The silicon area is bound at the top and at the side walls by perfectly isolating material (adiabatic surfaces). At the bottom the silicon is assumed to have an infinite extent.

  This assumption is based on the fact that the mould compound of the package around the silicon die is usually two orders of magnitude less thermally conductive than silicon itself. Therefore transient simulations will be quite accurate if the time interval is kept small enough compared to typical time constants of the package. *Time intervals larger than approximately 1 second will result in incorrect results!*

- The thermal conductivity of silicon is assumed to be independent of temperature.

  In reality this conductivity is dependent on temperature. In order to be able to use the Green's function approach, a constant value needs to be taken. It has been found that the introduced errors are acceptable. An important consequence of this approximation is that the calculated temperature rise is independent of the ambient temperature.

More information on physical background and validation by measurements can be found in the Engineering Forum 2006 paper entitled *'A Fast and Flexible Thermal Simulation Tool Validated on Smart Power Devices'*. This paper can be loaded by selecting *'Help'* and then *'Engineering Forum paper…'* in the menu.

2 SCOPE OF THE TOOL

- **The tool is only intended for transient thermal simulations not extending beyond approximately 1 second. Longer time-scale simulations and/or steady-state simulations need to be performed by software correctly taking into account the package and possibly the PCB. For these specific cases, please contact Eddy Blansaer, who has access to Flotherm.**
- **The tool is performing purely thermal simulations without feedback to the electrical behaviour of devices. The power is treated as an input, while temperature is calculated as an output (no electro-thermal simulations).**

3 MAIN APPLICATION WINDOW

The main application window contains a number of fields where the user can specify his inputs (see Figure 1). Following groups are defined:

- *Die size*
- *Power sources*
- *Sensors*
- *Simulation setup*

The interface also contains a number of buttons to simulate or plot:

- *Simulate / plot temperature*
- *Plot power*
- *Draw floorplan*

Finally, the menu contains a couple of fields:

- *File -> Load / Save state*
- *Help*

Following sections will describe each of the fields or buttons in more detail.

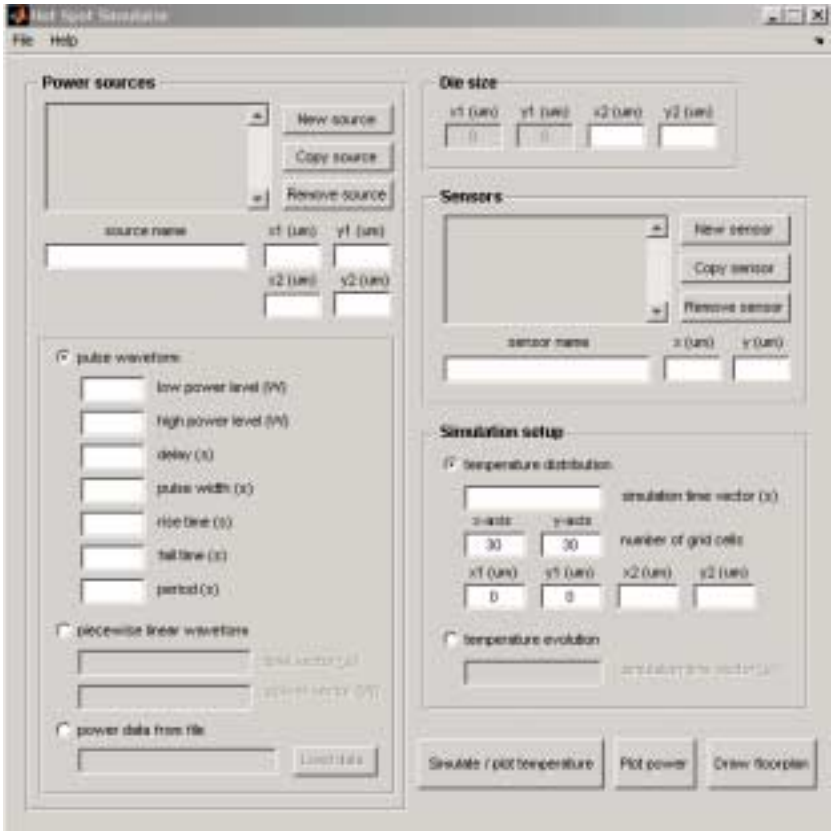

Figure 1: Main window.

4 INPUT FIELDS

4.1 Die size

The die is defined as the rectangular silicon area excluding the scribe, which will be sawn away later. Outside this area a material with a very low thermal conductivity is assumed, which is approximated by a perfect insulator (typically the moulding compound of a package is about 100 times less conductive than silicon). The lower-left corner is assumed to be at coordinates $(x_1,y_1) = (0,0)$. The coordinates of the upper-right corner (x_2,y_2) need to be specified by the user (in μm).

4.2 Power sources

The tool allows to specify an arbitrary number of rectangular power sources. Each of the power waveforms can be an arbitrary function of time. Following subsections describe how to define the power sources.

4.2.1 Creating, copying and removing sources

Three buttons for manipulation of sources are defined:

- *New source* : creation of a new power source
- *Copy source* : copy a previously defined source
- *Remove source* : remove a previously defined source

All sources are listed in the box next to the previously mentioned buttons, and can be selected in this box. After selection, the inputs of the corresponding source (name, coordinates and waveform information) appear in the fields below.

> ***Remark :*** Also for the first power source the button *'New source'* needs to be pressed, otherwise the inputs will not be taken into account !

4.2.2 Source name and position

The name and position of every power source can be specified in the following fields:

- *source name* : name which will be used on all plots
- *x1* : x-coordinate of lower-left corner (in μm)
- *y1* : y-coordinate of lower-left corner (in μm)
- *x2* : x-coordinate of upper-right corner (in μm), which needs to be larger than *x1*.
- *y2* : y-coordinate of upper-right corner (in μm), which needs to be larger than *y1*.

4.2.3 Source power waveform

Every power source can have an arbitrary waveform as a function of time (with the limitation that for t<0 no power is generated). However, in practice the number of waveform types is limited. Therefore, three different methods of specifying waveforms are provided.

> ***Important remark*** : The tool assumes that for t<0 no power is generated and that the whole die is at a steady-state uniform temperature. Starting from t=0 the tool performs a transient simulation.

4.2.3.1 Pulse waveform

A pulse waveform (see Figure 2) has a limited number of parameters, each of which can be specified in a separate field:

- *low power level* (in W)
- *high power level* (in W)
- *delay* (in s)
- *pulse width* (in s)
- *rise time* (in s)
- *fall time* (in s)
- *period* (in s)

This waveform also allows to specify simple step functions, periodic square waves, or periodic triangular waves.

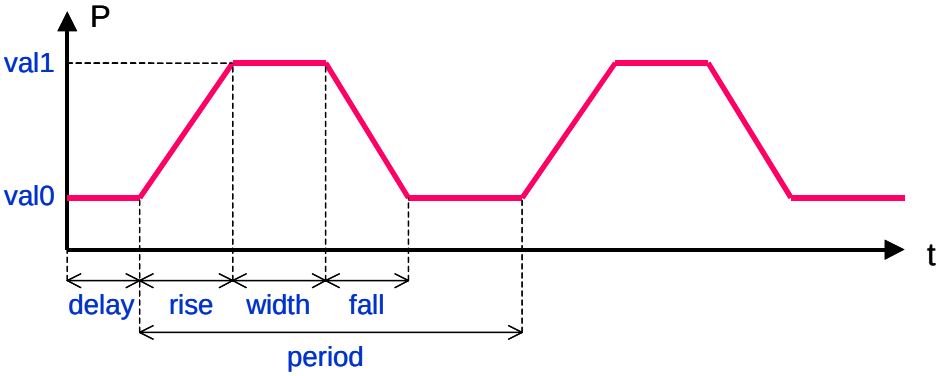

Figure 2: Pulse waveform.

4.2.3.2 Piecewise linear waveform

This option allows the user to approximate an arbitrary waveform by a piecewise linear waveform, which needs to be specified by a vector of points in time and a corresponding vector of power values. The simulator makes a linear interpolation between two successive points, and maintains a constant power after the last point.

The time and power vectors need to be specified in the following fields:

- *time vector* (in s)
- *power vector* (in W)

The vectors need to be specified in standard Matlab format. Also mathematical expressions are allowed. Some examples:

- ***Example 1*** (see Figure 3):

  time vector : `[0 40 40 60 60]*1e-6`

  power vector : `1 1 3 3 0.5`

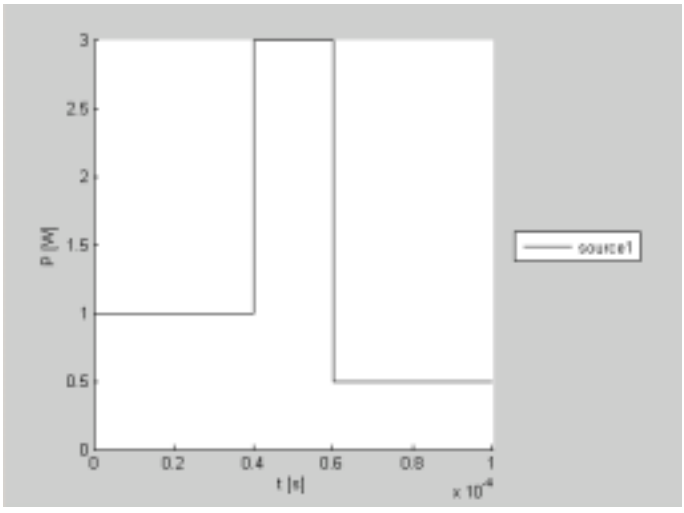

Figure 3: Piecewise linear waveform: example 1.

- ***Example 2*** (see Figure 4): quadratic dependence:

  time vector : `0:20e-6:100e-6` (syntax: `start:step:stop`)

  power vector : `0.1*(0:2:10).^2` (syntax: `start:step:stop`)

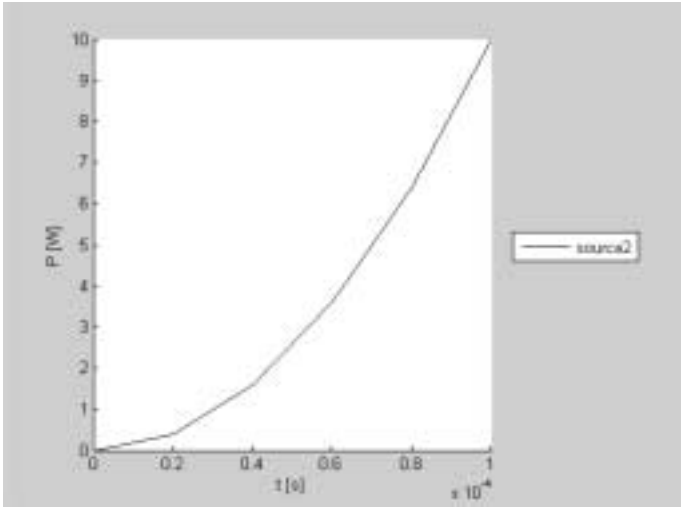

Figure 4: Piecewise linear waveform: example 2.

4.2.3.3 Power data from file

This option allows the user to load data from an external text file (ASCII format). The file should contain two columns of data, the first one containing the time points, the second one containing the corresponding power points. Just like for the piecewise linear waveform (described in Section 4.2.3.2), the simulator makes a linear interpolation between two successive points, and maintains a constant power after the last point.

The data file can be located by pushing the *'Load data'* button, after which a file browser appears. After selection, the file name appears in the field at the left of the button.

4.3 Sensors

The tool allows to specify an arbitrary number of sensors at the surface of the die. Following subsections describe how to define the sensors.

4.3.1 Creating, copying and removing sensors

Three buttons for manipulation of sensors are defined:

- *New sensor* : creation of a new sensor
- *Copy sensor* : copy a previously defined sensor
- *Remove sensor* : remove a previously defined sensor

All sensors are listed in the box next to the previously mentioned buttons, and can be selected in this box. After selection, the inputs of the corresponding sensor (name and coordinates) appear in the fields below.

4.3.2 Sensor name and position

The name and position of every sensor can be specified in the following fields:

- *sensor name* : name which will be used on all plots
- *x* : x-coordinate of sensor (in μm)
- *y* : y-coordinate of sensor (in μm)

  ***Remark*:** To measure the temperature of a heat source, place a sensor in the centre of this source.

4.4 Simulation setup

After the power sources and sensors have been defined on the die surface, the tool is able to calculate temperature rise for any combination of time t and coordinates (x,y). In practice the tool allows to simulate temperature for two specific cases:

- *temperature distribution* : temperature as a function of coordinates (x,y) for specified times t
- *temperature evolution* : temperature as a function of time t for specified sensor coordinates (x,y)

These two cases will be described in more detail in what follows.

4.4.1 Temperature distribution

The temperature distribution will be calculated for the times specified in the field *'simulation time vector (s)'*. The format of the time vector is defined in the standard Matlab format, examples of which are shown in Section 4.2.3.2 (for only one time point: simply enter the value itself). For every time point two graphs will be generated (a surface plot and a contour plot, showing the isotherms), as will be shown in Section 5.3.1.

The temperature will be calculated on a rectangular grid of evenly spaced points on the die surface. The location of the area in which the temperature needs to be calculated, and the number of grid cells in horizontal and vertical direction, can be specified by the user. Therefore following fields are provided:

- *number of grid cells (x-axis)* : number of equally sized cells in x-direction of simulation area
- *number of grid cells (y-axis)* : number of equally sized cells in y-direction of simulation area
- *x1* : x-coordinate of lower-left corner of simulation area (in μm)
- *y1* : y-coordinate of lower-left corner of simulation area (in μm)
- *x2* : x-coordinate of upper-right corner of simulation area (in μm), where *x2 > x1*
- *y2* : y-coordinate of upper-right corner of simulation area (in μm), where *y2 > y1*

By default the simulation area is set equal to the die area, while the number of grid cells in each direction is set to 30.

> ***Tip*** : In case of a very localised hot spot (compared to the die area) it is very useful to restrict the simulation area to a small region around the power source. For the same number of grid points this will allow to determine the hottest point with more accuracy (while the accuracy of the calculation at a given point is not influenced by the grid selection).

Next to the temperature at the grid points, also the temperature at the sensor locations is calculated. These temperatures are printed in the legend of the contour plot (see Section 5.3.1). However, also without specification of sensors a temperature distribution can be calculated.

4.4.2 Temperature evolution

The temperature evolution will be calculated for every sensor as a function of time. The time vector needs to be specified in the field *'simulation time vector (s)'*. The format of the time vector is defined in the standard Matlab format, examples of which are shown in Section 4.2.3.2. The temperature as a function of time at every sensor location will be plotted on one graph, as will be shown in Section 5.3.2.

> ***Tip***: Because for most applications the 'worst-case' situation where the maximum temperature occurs, is the most interesting one, it is advisable to first run a temperature evolution simulation for critical sensor positions. This type of simulation will yield the maximum temperature over time for every sensor.

5 SIMULATION AND PLOT BUTTONS

5.1 Draw floorplan

This button enables the user to visualise the location of power sources and sensors on the die before performing the actual simulations. The floorplan can be drawn at any intermediate step and can serve as a means to detect input errors as early as possible. The minimum requirement in order to draw the floorplan is to have the die coordinates available.

- ***Example*** (see Figure 5): two power sources and three sensors:

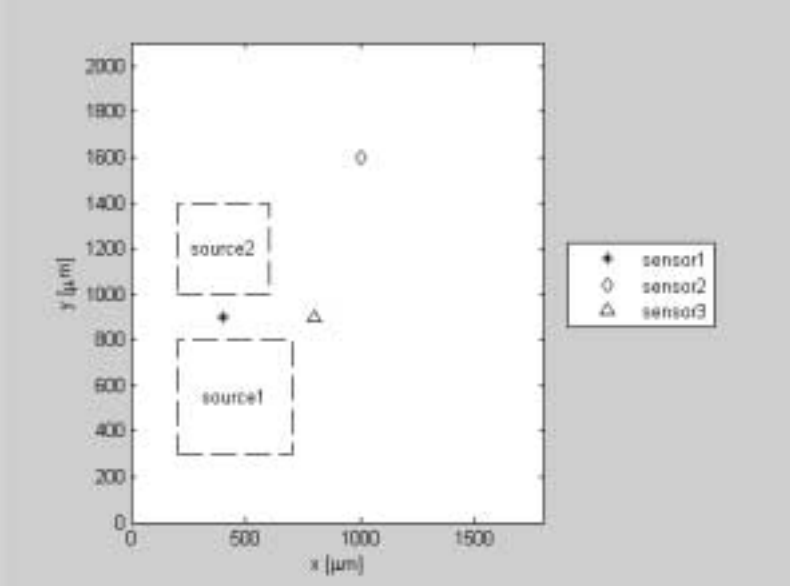

Figure 5: Draw floorplan: example with two power sources and three sensors.

5.2 Plot power

This button enables the user to plot the waveforms of all power sources on one graph. In order to be able to do this the waveforms of every power source need to be provided, as well as the simulation time vector (either in *'temperature distribution'* or in *'temperature evolution'* mode), from which the final point in time is taken.

- ***Example*** (see Figure 6): two power sources (both defined as a pulse waveform):

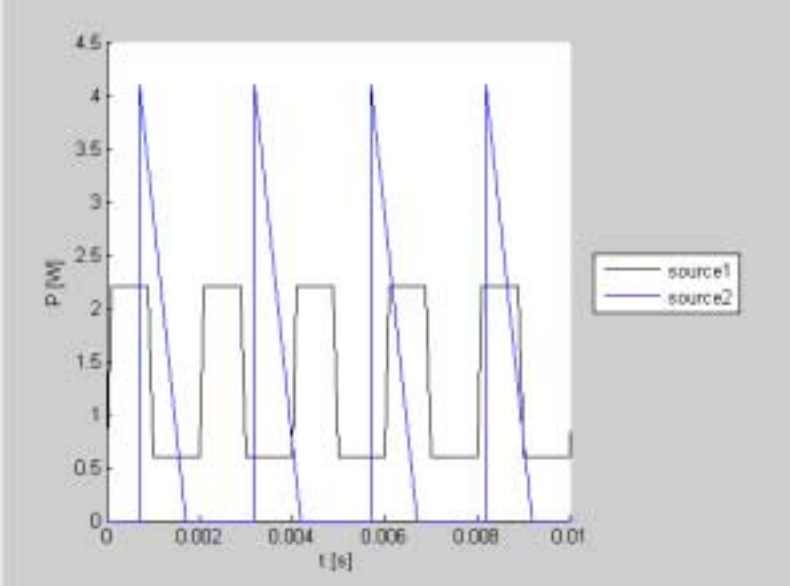

Figure 6: Plot power: example with two power sources.

5.3 Simulate and plot temperature

After full specification of die size, power sources, sensors, and simulation setup, a simulation can be run by pressing the *'simulate / plot temperature'* button. Before completion, a fractional bar will show the progress of the simulation (see Figure 7). The simulation may be aborted by pressing the *'Cancel'* button.

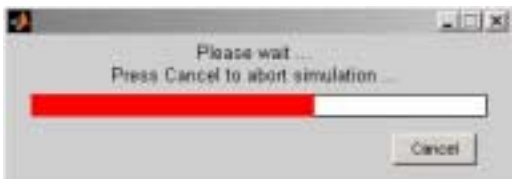

Figure 7: Fractional bar showing progress of simulation.

5.3.1 Temperature distribution

After completion of a temperature distribution simulation, two graphs are created: a surface plot and a contour plot, of which examples are shown respectively on Figure 8 and Figure 9. The contour plot shows the isotherms on the die surface. These examples correspond to the inputs shown on Figure 5 (floorplan) and Figure 6 (power waveforms) at t = 9 ms.

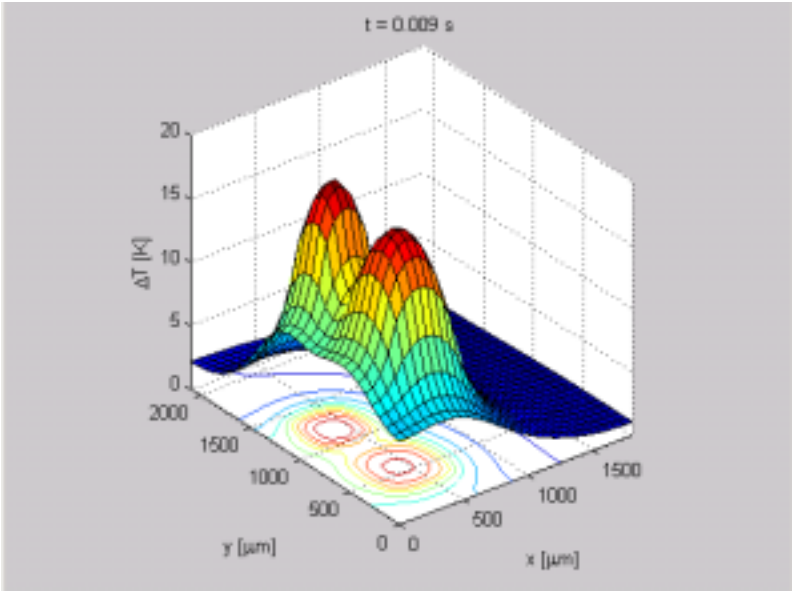

Figure 8: Temperature distribution: example of surface plot.

Remark that on the contour plot (see Figure 9) also the power sources and sensors are indicated. In the legend also the corresponding temperature increase seen by each sensor is written.

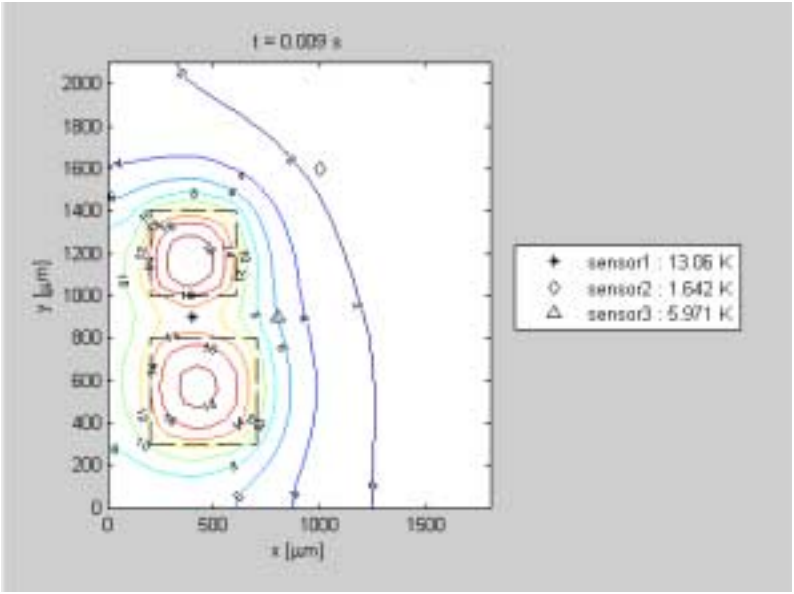

Figure 9: Temperature distribution: example of contour plot.

Next to the surface and contour plots, also a message box is generated which shows the maximum temperature on the distribution, together with the coordinates at which the maximum occurs (see Figure 10).

Figure 10: Message box showing maximum temperature(s) on temperature distribution(s).

5.3.2 Temperature evolution

After completion of a temperature evolution simulation, a graph is created which shows the temperature rise at all sensor positions as a function of time. Figure 11 shows an example corresponding to the inputs shown on Figure 5 (floorplan) and Figure 6 (power waveforms).

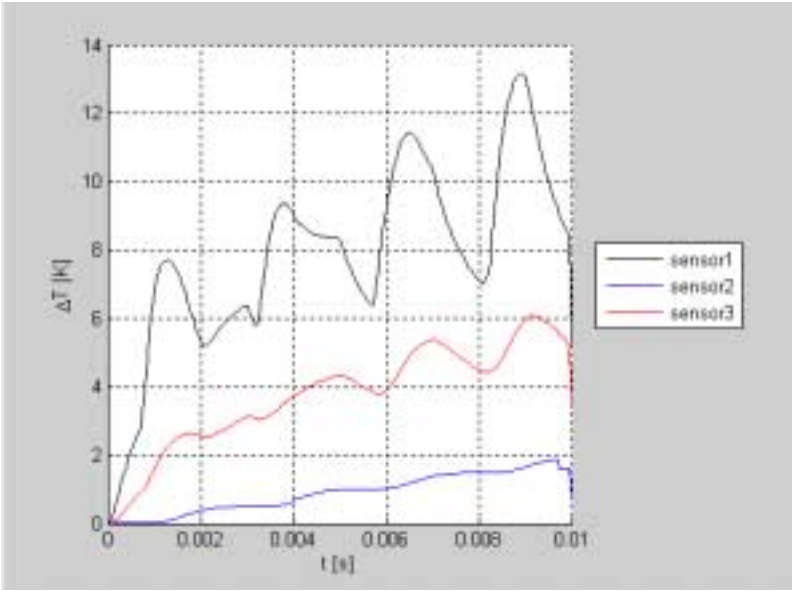

Figure 11: Temperature evolution: example.

Next to the temperature evolution plot, also a message box is generated which shows the maximum temperature for every sensor, together with the time at which the maximum occurs (see Figure 12).

Figure 12: Message box showing maximum temperature(s) on temperature evolution(s).

Tip : In order to find the worst-case situation of the highest temperature increase, it may be useful to use the information displayed in the message box shown below. After performing a temperature evolution simulation at some critical sensor positions (e.g. the middle of a power source), the time at which the maximum occurs can then be used to perform a temperature distribution, which can be further exploited to pin down the real maximum.

Tip : The temperature increase at sensor locations may be fed back to Spice-like circuit simulations, where individual devices each have a ***'trise'*** property, which can be used to let the simulator use the device model at an increased local temperature. A typical example is a pair of matched transistors which is experiencing a temperature gradient. Giving each of the transistors a different *'trise'* value will result in different currents through each of them, enabling more accurate circuit simulation results.

6 SAVE AND LOAD STATE

6.1 Save state

The current state, comprising all inputs and generated figures, can be saved by selecting *'File'* and then *'Save state…'* in the menu. A file browser window will open, in which the user can select the location and name of the file. The user needs to select a file name with the extension '.mat' (standard Matlab data format), in which all inputs entered in the user interface will be saved. The figures will then automatically be saved in a file with the same base name, however with the extension '.fig' (standard Matlab figure format). If no figures are open (anymore), no figure file will be saved.

After saving the current state, the name of the file (without extension) will appear in the title of the main window.

6.2 Load state

A previously saved state, comprising inputs and generated figures, can be retrieved by selecting *'File'* and then *'Load state…'* in the menu. A file browser window will open, in which the user can select the location and name of the file. The user needs to select a file name with the extension '.mat' (standard Matlab data format), in which all inputs are present which were entered in the user interface. The figures will then automatically be loaded from a file with the same base name, however with the extension '.fig' (standard Matlab figure format). If the latter file is not found in the same path, no figures will be loaded.

After loading a state, the name of the file (without extension) will appear in the title of the main window.

7 MANIPULATION OF FIGURES

Every generated figure can be manipulated in the standard Matlab way, which is described in detail in the Matlab documentation. Some interesting manipulations will be briefly described here.

7.1 Changing figure properties

Properties related to the plot itself can be changed by selecting *'Edit'* and then *'Axes properties…'* in the menu of the figure. These properties include the axes limits, labels and scales, the title, and the legend. Another way to change properties is to push the arrow button on the figure and to select properties using the right mouse button.

7.2 Visualising figures

Zooming in or out on figures is possible using the magnifying glass buttons. Also a button to rotate surface plots is available.

7.3 Saving figures

Instead of saving the complete state (see Section 6.1), where all figures are grouped into one '.fig' file, it is also possible to save individual figures. This can be done by selecting *'File'* and then *'Save as…'* in the menu of the figure to be saved. Several format types are possible:

- By default the standard '.fig' Matlab format will be proposed, which contains all information about the figure, but which cannot easily be imported into other software.
- Saving the figure as a bitmap allows to import the picture into other software (like MS Word). Using the uncompressed '.bmp' format usually takes too much disk space. A good compression format for plots is the '.png' (Portable Network Graphics) format, which provides the best compromise between quality and file size. The commonly known '.jpg' format should be avoided, as it transforms plots into blurry images with shadows.
- It is also possible to save figures in vector formats like '.eps' (Encapsulated Postscript) or '.pdf' (Portable Document Format).

8 DOCUMENTATION

The present user manual can be opened by selecting *'Help'* and then *'User manual…'* in the menu.

More information on physical background and validation by measurements can be found in the Engineering Forum 2006 paper entitled *'A Fast and Flexible Thermal Simulation Tool Validated on Smart Power Devices'*. This paper can be loaded by selecting *'Help'* and then *'Engineering Forum paper…'* in the menu.