Electro-Thermal Co-Design of Chip-Package-Board Systems

Christoph Sohrmann*, Andy Heinig, Michael Dittrich, Roland Jancke, Peter Schneider

Fraunhofer IIS/EAS, Zeunerstraße 38, D-01069 Dresden, Germany

* Corresponding Author: christoph.sohrmann@eas.iis.fraunhofer.de, +49 351 4640 788

Abstract

We propose an innovative thermal modeling approach that takes into account different “levels-of-knowledge” as they become available during the design phases of microelectronic systems. There are many tools available for thermal simulations which are well-suited for high-precision modeling. However, their applicability severely diminishes if CAD models are not available or modeling time is limited. Especially for electro-thermal co-design, where several spins between modeling and simulation are required, modeling time quickly becomes prohibitively large. By a combination of different techniques our solution offers fast and intuitive modeling, continuous addition of detail, very short simulation times, and open interfaces. In this paper we present our thermal modeling approach based on constructive solid geometry (CSG). We discuss advantages and limits of this solution and demonstrate its performance on an example of industrially-relevant complexity.

1 Introduction

1.1 Electro-thermal co-design

Decisions on thermal solutions are required during all stages of a product’s design-cycle [1]-[4]. This ideally requires an electro-thermal co-design approach, where thermal simulations accompany the entire design process. However, this poses a challenge to thermal simulation, which could be called level-of-knowledge-problem: The available knowledge about the system is highly dynamic during the design process. At the beginning, only a coarse design-space-exploration may be available for testing potential thermal issues and possible system-level-solutions. Later, a package solution may have been chosen and a detailed package model can be used to increase the predictive power of the simulation.

Most of the thermal tools available rely on either very coarse or very detailed information [5]-[9]. However, for an electro-thermal co-design, a smooth transition between the two extremes is desirable. Such a tool would add new possibilities to the modeling process.

1.2 Proposed solution

Our proposed solution of the above problem is a new thermal modeling approach which is capable of handling different levels-of-knowledge equally. Details can be added as they become available from the early concept phase to the final design of the entire system. Industry-standard solutions [6],[7] for thermal modeling are the tools of choice for high-precision modeling. However, they usually require significant modeling effort, especially for electro-thermal co-design. Our solution is designed to address such dynamic modeling conditions by a combination of the following techniques:

The model geometry is formulated using constructive solid geometry (CSG) [10]. In contrast to the more common boundary description, CSG uses simple primitive objects and combines them by Boolean operators. A CSG model can be easily created in a hierarchical fashion, solving the level-of-knowledge-problem, since details may be added after a coarse model has already been specified. This geometry description is then mapped to an octree grid [11] by virtue of distance fields (DF) [12]. Using octree grids is much faster than using the more common triangulation without losing flexibility regarding local refinement. The ability to refine the mesh locally is of paramount importance to handle the scale-differences within a chip-package-board system, where length scales between 1mm and 1cm occur. Finally, the mathematical problem is solved by a finite-volume method (FVM) [13].

2 Thermal simulation basics

2.1 The heat equation

When dealing with solid, microelectronic systems, heat propagation in the bulk is well described as a time-dependent heat equation of diffusion type [14].

\[ c_p(r)\rho(r) \frac{\partial T(r)}{\partial t} = \nabla \cdot [k(r)\nabla T(r)] + p(r) \]

where \( c_p(r) \) is the specific heat capacity, \( \rho(r) \) the mass density, \( k(r) \) the thermal conductivity tensor, \( p(r) \) the power density, and \( T(r) \) is the temperature field. The material properties are temperature-dependent, rendering this equation in fact non-linear. However, the temperature-dependence is usually weak enough to be
neglected or sufficiently accounted for by a few iterative solves. For electro-thermal problems with short time scales in the electrical domain, the thermal time-dependence may also be neglected. Especially for optimization applications such as thermally-aware floorplanning, a static solution is usually sufficient. Then, the heat equation simplifies further to
\[ \nabla \cdot [k(r)\nabla T(r)] = -p(r) \]
which is a variable-coefficient Poisson equation, describing the static heat distribution in the bulk of the solid. At the boundary of the material, this equation is not valid. Boundary conditions (BCs) have to be specified towards the exterior describing the surface behavior. Table 1 lists commonly used boundary conditions. When discretizing the problem at the surface, the respective equation has to be used instead of the bulk equation above.

Table 1: Common types of boundary conditions for the heat equation. Symbol \(\varepsilon\) is the heat transfer coefficient, \(\sigma\) the Stefan-Boltzmann constant, and \(n\) the face normal at the surface.

<table>
<thead>
<tr>
<th>BC</th>
<th>Type</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirichlet</td>
<td>Fixed temperature</td>
<td>(T = T_{amb})</td>
</tr>
<tr>
<td>Neumann</td>
<td>Fixed heat flux</td>
<td>(k\nabla T \cdot n = j_{amb})</td>
</tr>
<tr>
<td>Robin</td>
<td>Convection</td>
<td>(k\nabla T \cdot n = c(T_{amb} - T))</td>
</tr>
<tr>
<td>Radiation</td>
<td>Radiation</td>
<td>(k\nabla T \cdot n = \varepsilon\sigma(T_{amb}^4 - T^4))</td>
</tr>
</tbody>
</table>

Even for the static model, analytic solutions exist only rarely and numerical techniques need to be used. Efficient linear solvers exist such as algebraic multigrid (AMG) [15] methods which handle systems with several million unknowns in a few seconds on standard hardware. Whereas some years ago semi-analytic schemes [16], e.g. Fourier methods, promised dramatic performance gains, the highly optimized linear solvers of today exhibit a similar performance in a much more general setting.

2.2 Finite Volume Discretization

There is a great variety of numerical methods available to solve partial differential equations [17],[18]. Among them are many semi-analytic methods with the promise to outperform conventional numerical schemes [16]. However, in regard of industrial applicability, the only robust and flexible methods according to the experience of the authors are finite differences (FDM), finite elements (FEM), and finite volume (FVM) schemes. Some slightly more elegant methods exist such as boundary elements (BEM) or Green’s functions (GF) which, however, usually require significantly more work to handle inhomogeneous materials, non-rectangular geometries, and non-trivial boundary conditions.

However, for most real-life applications this is a standard requirement. We therefore chose a finite-volume method (FVM) which is powerful, flexible and well-suited for conservative problems especially on unstructured grids. The FVM relies on the flux through a small control volume \(V\). The discretized version of the heat equation is found by integration over this control volume,

\[
\int_V c_p(r)p(r)\frac{\partial T(r)}{\partial t}d^3r - \int_V \nabla \cdot [k(r)\nabla T(r)]d^3r = \int_V p(r)d^3r
\]

and then applying Gauss’ divergence theorem to the static term, thereby removing one derivative,

\[
\int_V c_p(r)p(r)\frac{\partial T(r)}{\partial t}d^3r - \oint_{\partial V} [k(r)\nabla T(r)] \cdot n\,dS = P_V
\]

where \(\partial V\) is the surface of the control volume and \(P_V\) is the total power generated within the control volume.

As argued earlier, we restrict the discussion to the static case and drop the dynamic term. The more general case can be derived similarly to the following steps. For any numerical method with polyhedral grid cells, the surface integral from above can be approximated as a sum over the faces of the cell,

\[
\oint_{\partial V} [k(r)\nabla T(r)] \cdot n\,dS \approx \sum_f A_f k\nabla T_f \cdot n_f = P_V
\]

where \(f\) labels the faces of the cell, \(A_f\) is the face area, \(k\) is the average thermal conductivity of the cell, \(\nabla T_f\) is the temperature gradient over the face and parallel to the face normal \(n_f\).

For cells at the boundary, the BC according to Table 1 needs to be included. Using convective BCs as the most important BCs, the discretized equation becomes

\[
\sum_f A_f k\nabla T_f \cdot n_f + \sum_g cA_g(T_{amb} - T_g) = P_V
\]

where \(f\) runs only over internal faces, and \(g\) only over the boundary faces.

3 Constructive Solid Geometry

3.1 General concept

One of the most time-consuming tasks when using CAD tools is model creation. Since there are not always simulation-ready CAD models of the system of interest available, the engineer dedicates tedious modeling effort. Using Constructive Solid Geometry (CSG) [10], it is possible to greatly reduce this effort. In CSG, complex objects are described as Boolean combinations of basic objects. Figure 1 shows a visual example of combining a cube and a sphere by the common Boolean operators.
Figure 1: The three basic CSG operations applied to two primitive objects, cube and ball. Additional spatial transformation operations can be applied.

Additionally, linear transformations are allowed to shift, rotate or scale the objects and compounds thereof. The final CSG object can be most conveniently represented by a tree, which will be made use of the following.

### 3.2 Hierarchical description in XML

The tree representation of a CSG model offers a highly useful application: Models can be refined by combining different CSG trees, enabling a hierarchical modeling approach. With increasing information about the system available, the model can be continuously improved by adding details. This is very similar to a design process of microelectronic systems: Whereas only coarse information is available at the beginning, more and more details about the system gradually occur.

We have created an XML container format to formulate and store a CSG model. There is a basic structure providing a hierarchical syntax for library and components, as well as a CSG format which is inherited from the basic structure and adds syntax for describing the CSG tree.

Figure 3 shows some example XML according to our CSG format.

```xml
<library xmlns=
"http://www.ess.iis.fraunhofer.de/geometry"
xmlns:basic=
"http://www.ess.iis.fraunhofer.de/basic"
name="EXAMPLE_LIBRARY">
  <basic:component name="TOP_LEVEL_COMPONENT">
    <rotate angle="90,0,0" x="0" y="0" z="1"/>
    <basic:instance component="EXAMPLE_COMPONENT"/>
  </basic:component>

  <basic:component name="EXAMPLE_COMPONENT">
    <translate x="1,0,0" y="1,0,0" z="1,0"/>
    <difference>
      <cubeoid length="10" width="10" height="10"/>
      <ball radius="10"/>
    </difference>
    </translate>
  </basic:component>
</library>
```

Figure 2: Example for a CSG description with XML.

Since the intention of this format is solely a geometry description, object properties (e.g. material) are not part of this description. Such non-geometry-related information is annotated using additional mapping files.

### 3.3 Querying the CSG tree

For numerical computation, the CSG model has to be mapped to a discrete geometric grid. For this purpose, some query point functions on the CSG tree have to be implemented. Due to the usually very large number of elements in the tree, this has to be carefully implemented to avoid performance issues. We have solved this by using bounding boxes and spatial search structures. In our implementation, each node of the CSG tree carries an axis-aligned bounding box (BB) embedding all child nodes, as well as an R-tree [11] for efficiently finding relevant child nodes when querying. By using these techniques, we are able to query CSG trees with millions of objects (e.g. transistors), as encountered in practical applications. One major aspect for efficiency is the use of BBS in the global coordinate system, to avoid unnecessary recalculation of the transformations. Figure 4 shows the recursive process to create global BBs from the local transformations of each node.

Figure 3: Process which converts local transformations to global bounding boxes. Since each leaf node can be a parent itself, the process is recursive.

Starting with the transformation T0 of the root node, the transformations are passed top-down the tree and combined at each node, yielding the global transformations T0, T1, etc. Starting at the leaves, the global BBs are computed from those global transformations. The BBs are then gradually passed bottom-up the tree and combined at each node, depending on the function of the node. After this V-cycle, each node has its global BB. This is the basis for efficiently querying the tree.

### 4 Signed Distance Fields (SDF)

#### 4.1 General concept

A distance field (DF) [12] of an object is a scalar function defined in the entire space giving the minimal distance to the object’s surface. The signed distance field (SDF) additionally indicates whether the point is inside
or outside of the object. Mathematically, the SDF can be rewritten as
\[ d(p) = \text{sgn}(p) \inf_{x \in \mathbb{S}} \|x - p\| \]
where \( \text{sgn}(p) \) is negative if \( p \) is inside or positive if \( p \) is outside the object, respectively. Due to this property, DFs can be used as an implicit representation of a geometric object. In contrast to an explicit description (e.g. position, extent, etc.) the distance field only “answers” the questions: a) How far is the object from a query point and b) Is the point inside or outside the object? The SDF of an object can be interpreted as follow:

- \( d(p) < 0 : \) \( p \) is inside
- \( d(p) > 0 : \) \( p \) is outside
- \( d(p) = 0 : \) \( p \) is on the surface

The fact that the surface of the object is simply the isosurface for \( d(p) = 0 \) additionally offers an elegant way to represent an object visually. Any visualization library that provides isosurface plots can be used to draw the object by plotting its SDF zero-isosurface.

Although many numerical techniques exist for finding distance fields of arbitrary objects, only analytic expressions are of practical interest when performance is critical. Using SDF for representing the primitive objects in the CSG tree limits the variety of shapes which can be used. Two examples for which analytical SDF expressions exist are the ball
\[ d_{\text{ball}}(p) = \|p - c\|_2 - r \]
with center \( c \) and radius \( r \) and the cuboid
\[ d_{\text{cuboid}}(p) = \max_i(|p_i - c_i| - h_i/2) \]
with center \( c \) and extent \( h \). Other objects are possible but may affect performance. One other important primitive is the half-space, which has a similarly simple SDF and can be used to create a variety of convex polyhedra by intersection (see the following section).

### 4.2 Boolean operations on SDFs

Using CSG and SDFs together is a truly synergetic combination. The real strength of SDFs reveals itself when looking at Boolean combination of objects. The distance fields of two objects A and B are easily combined by Boolean operators according to the following rules:

- **Union:** \( d(A \cup B) = \min(d_A(p), d_B(p)) \)
- **Difference:** \( d(A - B) = \max(d_A(p), -d_B(p)) \)
- **Intersection:** \( d(A \cap B) = \max(d_A(p), d_B(p)) \)

Thus, Boolean operations in the CSG tree are easy to implement. Furthermore, new primitive objects such as convex polyhedral may be constructed as an intersection of multiple half-spaces.

### 4.3 Transformations and SDFs

The use of SDFs combined with affine transformations is serious limited: The SDF is not invariant under transformations and cannot even be computed for most objects. This is due to the fact that not only the distances to the surface are changing but also the position of the closest point on the surface (think of a ball vs. ellipsoid). The most practical solution is to allow only similarity transformations, i.e. rotation, translation and homogeneous scaling (zoom) or to implement SDFs respecting affine transformations.

### 4.4 Detecting intersections with SDFs

When creating the numerical grid, intersection tests are heavily checking for features within a grid cell. This is usually a demanding computational task. SDFs, however, offer an efficient way of doing an approximate intersection check by a falsification argument: An SDF cannot state anything about the exact position of an object in space. It can, however, be used to exclude the presence of an object within a certain radius – this radius being the value of the SDF. Starting with a hypothesis of intersection, this allows disproving the hypothesis (falsification). If no disprove is found, intersection should be assumed to remain on a conservative side.

As an example, the intersection between a Square A (center \( e_A \) and side length \( h_A \) ) and an Object B is tested. Intersection is impossible for the following condition:
\[ d_B(e_A) > \frac{h_A}{\sqrt{2}} \]

Figure 3 gives a graphical representation of this test.

![Figure 4: Testing the intersection hypothesis by checking the SDF value of B against a bounding sphere of A.](image)

The SDF value of B at the center of A is compared with the radius of a bounding sphere of A. If the above test is true, the intersection is impossible. But if the test fails, it only implies intersection with the bounding sphere, not necessarily with A. In that case, more stringent tests can be carried out giving more accuracy on the result. However, finally accepting a potential intersection is mostly a “conservative” and safe assumption.

### 5 Octree grid

#### 5.1 General concept

Octrees [19][11] are a highly versatile data structure often used in computer graphics. Starting with a root cell which entirely embeds the object, the cells are repeatedly refined by subdividing them into 8 child cells of equal size. In numerical mathematics it is often used as an unstructured discretization grid for partial differential
equations. Table 2 shows a comparison of node counts for full rectangular grids and octree grids on some typical example geometry for different depths of the grid (number of subdivisions).

<table>
<thead>
<tr>
<th>d</th>
<th>(N_f) (full grid)</th>
<th>(N_o) (octree)</th>
<th>(N_f/N_o)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2 097 152</td>
<td>42 647</td>
<td>49</td>
</tr>
<tr>
<td>8</td>
<td>16 777 216</td>
<td>191 912</td>
<td>87</td>
</tr>
<tr>
<td>9</td>
<td>134 217 728</td>
<td>863 604</td>
<td>155</td>
</tr>
</tbody>
</table>

Table 2: Comparison of full regular grid and octree grid at grid depth level \(d\).

The spatial scale-differences between a package (about 1cm) and a microstructure (1nm) cannot be entirely resolved even by using octrees. Thus, all features below this length scale have to be averaged. This concerns the power distribution and material properties.

### 6 Example: Chip-package-board system

In order to demonstrate the effectiveness of the techniques presented in the previous chapters, we have applied them to a chip-package-board system of practically relevant size and complexity as shown in Figure 7.

Figure 5: An example of a chip-package-board system: a) Board, b) Solder balls, c) Laminate substrate, d) Underfill + bumps, e) Die metal stack, f) Die active layer, g) Die substrate, h) Thermal interface material (TIM), i) Heat spreader (lid), j) Heat sink (aluminum).

The example contains a die packaged as a flip-chip ball grid array [20], where the chip is mounted upside-down within the package. This package type avoids the use of bond wires. It uses a thermally conducting lid for protection and heat-spreading and a straight-fin heat sink.

Taking into account a realistic chip environment is important for obtaining meaningful results from the thermal simulation [14]. If package modeling is ignored or too crude models are used, this may result in an entirely wrong temperature assessment. This is where the CSG description has its major advantage: The geometry can be described clearly and with very little effort and all parts can be intuitively combined to a complex system. In this example, the package was constructed manually whereas the IC with its heat dissipation was directly exported from Cadence Virtuoso. But even if such detailed data is not available, some average power consumption is usually known, and a “thermal prototype” can be constructed from the average power.

Figure 8 shows the thermal model including all details of the example system and its final static temperature solution. The numerical model as shown contains about 200k degrees of freedom and solves in less than 20 seconds on standard hardware.

Figure 7: View of the IC inside an opaque package.

It can be seen that the IC temperature is about twice the temperature of the package. Similar packages without heat sink typically leak 10-20% of the heat through the lid and 80-90% through the board [20]. The low heat conductivity of the board will then lead to an increased temperature on the chip, which may be critical for the function of the IC.
In order to demonstrate the importance of a proper package model, we have modified the package and removed the aluminum heat sink. While the hot spots on the IC remain at nearly the same positions, the temperature range on the IC roughly doubles as compared to the solution shown in Figure 9. While using the heat sink the temperature ranges between 63°C and 130°C, without a heat sink it is increased to range between 102°C and 223°C maximum temperature.

Once the temperature on the IC is obtained, this information may be used in various ways: It may be annotated to the devices and fed back into Cadence to run a temperature-aware simulation or to find a valid electro-thermal operating point. If the IC is sensitive to temperature gradients, the entire temperature field may be annotated to the layout view in the form of isothermal lines, as shown in Figure 10.

Our approach was demonstrated on a chip-package-board system of real-world complexity. A comparison with a modified package solution emphasized the importance of using a correct package model in order to achieve reliable thermal simulation results.

8 Literature