M.Sc. Thesis

Modeling and Extraction of Through-Silicon Vias

J.C. Verwer

Abstract

3D is entering the world of Integrated Circuits. While interconnects have always been three-dimensional, the actual silicon inside an IC was essentially still planar. The introduction of the Through-Silicon Via changes that, by allowing connections to be made from one side of a die through its silicon substrate to the other side of the die, so that multiple dies can be interconnected efficiently inside a single package.

Like any revolutionary development, TSVs require changes in the tools that deal with them. Not all current Electronic Design Automation software is able to extract chip layouts containing TSVs to a correct circuit. In order to adapt extraction software, a methodology to extract TSVs is required. This methodology, in turn, requires a model.

Several models that can be found in literature are compared and contrasted. The one that is selected is improved upon by making some minor corrections. The model is also simplified and the conditions for validity of this simplification are shown. The resulting model is then used to implement an extraction methodology, both with a model-based approach and with a formula-based approach.

Simulating the different dies that make up a 3D IC together is demonstrated, giving the designer access to a complete toolchain necessary for designing 3D ICs and verifying their correctness.
Modeling and Extraction of Through-Silicon Vias

Thesis

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

by

J.C. Verwer
born in Leidschendam, The Netherlands

This work was performed in:
Circuits and Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics and Computer Science
TU Delft
Abstract

3D is entering the world of Integrated Circuits. While interconnects have always been three-dimensional, the actual silicon inside an IC was essentially still planar. The introduction of the Through-Silicon Via changes that, by allowing connections to be made from one side of a die through its silicon substrate to the other side of the die, so that multiple dies can be interconnected efficiently inside a single package.

Like any revolutionary development, TSVs require changes in the tools that deal with them. Not all current Electronic Design Automation software is able to extract chip layouts containing TSVs to a correct circuit. In order to adapt extraction software, a methodology to extract TSVs is required. This methodology, in turn, requires a model.

Several models that can be found in literature are compared and contrasted. The one that is selected is improved upon by making some minor corrections. The model is also simplified and the conditions for validity of this simplification are shown. The resulting model is then used to implement an extraction methodology, both with a model-based approach and with a formula-based approach.

Simulating the different dies that make up a 3D IC together is demonstrated, giving the designer access to a complete toolchain necessary for designing 3D ICs and verifying their correctness.
After spending ten years of my life as a TU Delft student I am now on the brink of graduation and it feels very surreal. I’ve certainly taken more time than I wanted to, but all’s well that ends well. Many people have supported and encouraged me over the years, but here I would like to take the opportunity to specifically thank the people at the TU that have worked with me during my graduation project. Gratitude to all others who deserve it is implied.

First of all I would like to thank Nick. Whenever I was stumbling towards another dead end, you gave me the nudge that brought me back on track. You always kept the task at hand in mind and reminded me of what needed to be done in the end. Thank you for trusting me and giving me the support I needed to see this project to completion.

I’d also like to thank Simon. You explained the internals of Space to me when I needed to know about something unfamiliar to me. I usually came by unannounced and you often made some time for me right at that moment. Likewise, I’d like to thank Antoon for keeping the computers running and solving whatever little problems occurred. You even arranged that I could use a second computer to run my simulations on. Minaksie and Rosario, thank you for putting up with me while I was sitting in your room to work on that computer. Additionally, I’d like to thank Minaksie for taking care of some of the paperwork as well.

Without Rob’s good advice I would not have been given another try at performing my EM simulations, so I’d like to thank him for his suggestion to use CST Microwave Studio, even though it didn’t work out in the end. I’d like to thank Amir for his advice and explanations about the physical aspects of TSVs. Our conversations helped to clarify the big picture for me.

Finally, I’d like to thank whomever I’ve forgotten to mention here in advance for forgiving me my unintentional omission. When deadlines are getting too close for comfort it’s things like these acknowledgments that suffer the most. It is left to the reader to decide if the rest of this thesis makes up for any imperfections found on this page.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Formula-based extraction</td>
<td>25</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Additions to the technology file</td>
<td>26</td>
</tr>
<tr>
<td>5.2</td>
<td>Model-based extraction</td>
<td>29</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Output file formats</td>
<td>29</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Input file format</td>
<td>30</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Model generation</td>
<td>31</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Layout generation</td>
<td>32</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Additions to the technology file</td>
<td>33</td>
</tr>
<tr>
<td>5.2.6</td>
<td>Usage of the layout and model generator</td>
<td>34</td>
</tr>
<tr>
<td>5.2.7</td>
<td>Importing the TSV layout</td>
<td>34</td>
</tr>
<tr>
<td>5.2.8</td>
<td>Importing the device model</td>
<td>34</td>
</tr>
<tr>
<td>5.3</td>
<td>Integral simulation</td>
<td>35</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Introduction</td>
<td>35</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Input file format</td>
<td>35</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Output file format</td>
<td>35</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Geometric matching</td>
<td>38</td>
</tr>
<tr>
<td>5.3.5</td>
<td>Usage of the integral simulation circuit generator</td>
<td>39</td>
</tr>
<tr>
<td>5.3.6</td>
<td>Importing the combined circuit</td>
<td>39</td>
</tr>
<tr>
<td>6</td>
<td>Verification</td>
<td>41</td>
</tr>
<tr>
<td>6.1</td>
<td>Spice setup</td>
<td>41</td>
</tr>
<tr>
<td>6.2</td>
<td>Spice results</td>
<td>45</td>
</tr>
<tr>
<td>6.3</td>
<td>3D model setup</td>
<td>47</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Source and load</td>
<td>49</td>
</tr>
<tr>
<td>6.4</td>
<td>Field solver results</td>
<td>50</td>
</tr>
<tr>
<td>6.5</td>
<td>Conclusions on the verification of the model</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>Application</td>
<td>53</td>
</tr>
<tr>
<td>7.1</td>
<td>Example design</td>
<td>53</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Lower die</td>
<td>53</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Upper die</td>
<td>53</td>
</tr>
<tr>
<td>7.2</td>
<td>Top-level process directory</td>
<td>54</td>
</tr>
<tr>
<td>7.3</td>
<td>Top-level project directory</td>
<td>55</td>
</tr>
<tr>
<td>7.4</td>
<td>Process directory for the lower die</td>
<td>57</td>
</tr>
<tr>
<td>7.5</td>
<td>Project directory for the lower die</td>
<td>58</td>
</tr>
<tr>
<td>7.6</td>
<td>Process directory for the upper die</td>
<td>59</td>
</tr>
<tr>
<td>7.7</td>
<td>Project directory for the upper die</td>
<td>60</td>
</tr>
<tr>
<td>7.8</td>
<td>Integral simulation</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>Conclusions</td>
<td>63</td>
</tr>
<tr>
<td>8.1</td>
<td>Contributions</td>
<td>63</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Modeling</td>
<td>63</td>
</tr>
<tr>
<td>8.1.2</td>
<td>Extraction</td>
<td>63</td>
</tr>
<tr>
<td>8.2</td>
<td>Future work</td>
<td>65</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Refining the model</td>
<td>65</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Diagonal parasitics</td>
<td>65</td>
</tr>
<tr>
<td>8.2.3</td>
<td>Physics-based extraction</td>
<td>65</td>
</tr>
</tbody>
</table>
8.2.4 Circular layout shapes ............................................. 66

A XML input file for model generation ............................ 67
B 3D model ..................................................................... 69
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>An example of a TSV, based on [3]</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>An example of a pair of TSVs, based on [3]</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>Electrical model of a TSV, based on [3]</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Dimensions of a pair of TSVs, based on [3]</td>
<td>17</td>
</tr>
<tr>
<td>4.1</td>
<td>A die, flipped in both directions and rotated half a turn</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>A die, flipped in both directions and superimposed on itself</td>
<td>23</td>
</tr>
<tr>
<td>4.3</td>
<td>A die, rotated a quarter turn, rotated half a turn and rotated three quarters of a turn</td>
<td>24</td>
</tr>
<tr>
<td>6.1</td>
<td>Simulation results of a ring oscillator and its piecewise linear approximation</td>
<td>44</td>
</tr>
<tr>
<td>6.2</td>
<td>Simulation results of 1 INV_X8 cell ⇒ 1 INV_X32 cell ⇒ pair of TSV 0</td>
<td>47</td>
</tr>
<tr>
<td>6.3</td>
<td>Simulation results of 64 INV_X32 cells ⇒ 256 INV_X32 cells ⇒ pair of TSV 0</td>
<td>48</td>
</tr>
<tr>
<td>6.4</td>
<td>Simulation results of 512 INV_X32 cells ⇒ pair of TSV 0</td>
<td>49</td>
</tr>
<tr>
<td>6.5</td>
<td>Simulation results of 4 INV_X32 cells ⇒ 16 INV_X32 cells ⇒ pair of TSV 1</td>
<td>50</td>
</tr>
<tr>
<td>6.6</td>
<td>Simulation results of 16 INV_X32 cells ⇒ 64 INV_X32 cells ⇒ pair of TSV 1</td>
<td>51</td>
</tr>
<tr>
<td>6.7</td>
<td>Simulation results of 512 INV_X32 cells ⇒ 2048 INV_X32 cells ⇒ pair of TSV 1</td>
<td>52</td>
</tr>
<tr>
<td>7.1</td>
<td>Simulation results of applying the extraction methodology to a ring oscillator</td>
<td>62</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Comparison of compact models .................................................. 14
3.2 Parameters of the TSV model, based on [3] ................................. 16
3.3 Variables of the TSV model, based on [3] ................................. 18

5.1 Variables of the TSV model translated to parameters in the Space technology file .......................................................... 27

6.1 Parameters of TSV 0 ................................................................. 42
6.2 Component values of TSV 0 ..................................................... 42
6.3 Parameters of TSV 1 ................................................................. 43
6.4 Component values of TSV 1 ..................................................... 43
6.5 Input source parameters ......................................................... 44

7.1 Parameters of TSV 2 ................................................................. 54
1.1 Context

Traditionally, Integrated Circuits (ICs) have always consisted of a single layer of semiconductor material (usually silicon) and several layers of interconnect on top of it, which are connected to the outside world through the IC’s package. Although the interconnect layer is three-dimensional, the active area of the IC is still essentially planar. Recent developments have changed that situation, so that it has now become possible to package multiple layers of semiconductor material inside a single IC. Connecting these layers inside the IC can be done in various ways. One possibility is that of the Through-Silicon Via (TSV), which is an opening in the silicon filled with conductive material, with an insulator in between.

Before a 3D IC can be produced, it has to be designed first. Existing Electronic Design Automation (EDA) software is not fully prepared for designs containing TSVs. One example is the process of extraction. After designing the IC layout, this layout has to be verified. One step in this verification process is to extract a circuit description from the IC layout. This circuit can then be simulated to verify the correctness of the layout. Current extraction tools aren’t all able to deal with TSVs. This thesis aims to describe an approach to the extraction of TSVs.

1.2 Problem statement

In order to be able to extract TSVs, several steps have to be taken first. It must be clear what kind of circuit has to result from the extraction process. Once this is clear and an extraction methodology has been found, the methodology needs to be implemented. Finally, the results of this implementation have to be verified for correctness.

Formally, the problem can be stated as follows:

1. Find a good model for TSVs.
2. Determine the relationship between the chip layout and the model’s parameters.
3. Develop a methodology to fill in the model’s parameters based on the chip layout.
4. Implement the methodology in Space.
5. Verify the result using a field solver.
1.3 Results

1.3.1 Solution

This thesis describes the implementation of two similar but different approaches to extraction: formula-based extraction and model-based extraction. Both approaches were implemented in Space. A third approach, physics-based extraction, was considered but not implemented due to time constraints. The results of both approaches were verified in a circuit simulation using Spice. In order to allow users to simulate all dies in an IC together, a layout and model generator was implemented. The combined utilities and methodologies developed for this thesis were applied to a ring oscillator spanning two dies.

1.3.2 Contributions

The main contributions to the current state of the art that this thesis makes are:

1. A compact model for TSVs is selected from literature and improved upon.
2. A methodology is presented for extracting TSV circuits from chip layouts, both in a formula-based and in a model-based fashion.
3. The implementation of this method is shown to yield acceptable results.
4. The usability of this method is improved by implementing a circuit generator for integral simulation of multiple dies.

A complete list of contributions, along with a more detailed explanation, can be found in Chapter 8.

1.4 Thesis organization

This thesis is organized as follows:

- Chapter 2 explains what TSVs are, what different kinds of TSVs exist and how this relates to the extraction process. It also discusses the different possible approaches to extraction and the issue of die bonding in 3D ICs.
- Chapter 3 discusses several possible parametric electrical models of TSVs and selects the one that will be used in the rest of this thesis.
- Chapter 4 describes a methodology to generate an electrical circuit that models a specific TSV, based on the parametric model and the layout of the IC that contains the TSV.
- Chapter 5 shows how this methodology was implemented in Space.
- Chapter 6 compares the results that were obtained using Space to the results that would have been obtained if the model that was used as a basis had been used without change.
- Chapter 7 applies the extraction methodology described by this thesis to a ring oscillator IC consisting of two different dies.
• Chapter 8 concludes the thesis, showing its contributions to the current state of the art and discussing possible directions for future work.
2.1 Through-Silicon Vias

A Through-Silicon Via is literally a via that goes through silicon. Regular vias connect different layers of interconnect through the Inter-Metal Dielectric (IMD). A TSV connects a layer of interconnect to a bump lying under the silicon substrate. This makes it an important element of three-dimensional ICs. While it is possible to construct 3D ICs without using TSVs, the number of connection that can be made between different dies will be limited and wiring will be long. TSVs allow for a higher connection density with shorter paths, which implies lower losses and higher speeds.

2.1.1 Structure of a TSV

There are several different kinds of TSVs, but the essence is always the same: a TSV is an electrical connection between a point below the silicon and a point above the silicon. Other ways to construct non-planar ICs exist, but they do not qualify as TSVs [1]. The electrical connection can be made of copper, but tungsten is also an option, as is polysilicon. It will normally be separated from the silicon substrate by an oxide layer. At both ends of the TSV, a connection has to be made either to a structure on a different die, or to interconnect on the same die. This usually means that some kind of bump will be formed that is larger than the TSV itself [2].

2.1.2 Examples

Figure 2.1 shows an example of a TSV. Other dies can be connected to this one using either of the bumps. The TSV is separated from the substrate by an insulating layer.

TSVs will often be placed in pairs — one ground TSV and one signal TSV. One such pair is shown in Figure 2.2. Apart from the substrate and IMD, three more layers can be distinguished. The insulating layer between the bumps is called the underfill. The underfill itself is also separated from the substrate, along with the bumps. This bottom oxide layer is distinct from the oxide layer between the TSV and the substrate.

This is just one possible realization of a TSV out of the many potential options. The various possible realizations will be discussed next.
2.1.3 Shapes and sizes

TSVs can have several different shapes. The most obvious shapes are the rectangular cuboid and the cylinder, of which the cylinder is the most common. Rectangular cuboids have length, width and height. Cylinders have height and a radius. All of them will influence a TSV’s electrical properties.

The resistance and inductance of a TSV are proportional to its height, so when everything else is equal, higher TSVs will have higher losses. However, the height of a TSV is not a property that can be chosen freely — it’s determined by the thickness of the substrate. Thicker substrates lead to higher substrate conductances and capacitances.

The square of the radius of a TSV is directly proportional to its cross-sectional area. Although, in principle, the area of the TSV is inversely proportional to its resistance, there are several reasons why its area can’t effectively be increased at will. One obvious reason is that of
space constraints, but other factors come into play as well. When subjected to AC currents, conductors will exhibit the so-called skin effect. Only the outer “skin” of the conductor will actually carry any current. This skin becomes progressively smaller as the current’s frequency increases. If the radius of the TSV is increased beyond its skin depth at a certain frequency, its resistance at that frequency will decrease linearly with its radius, instead of quadratically. The manufacturing process usually also imposes limits on the radius of TSVs.

The radius and height of a particular TSV realization depend on many factors. Generally, 3D ICs with many TSVs that connect components at a low hierarchy level will have TSVs with a smaller radius and 3D ICs with fewer TSVs that connect bigger components will have TSVs that are bigger. Height, being dependent on substrate thickness, will be less for Silicon on Insulator (SOI) processes than for bulk silicon processes. The next section describes the only freedom a designer has left in choosing TSVs’ heights once the substrate thickness has been decided on.

In case TSVs are cuboid-shaped, the length and width more or less take the place of the radius. In the special case where the length and width are equal (the TSV’s cross-sectional area is square, as opposed to circular for cylindrical TSVs), its length can be squared much like the diameter of a cylinder.

2.1.4 Via first, via middle and via last

TSVs can be formed at different points in the manufacturing process. They can be formed before any metalization is performed (via first), between different metalization steps (via middle), or after all metalization has been completed (via last). This difference isn’t merely cosmetic. In a via first process, the via connects to the lowest interconnect layer, which is used for local routing. In a via middle process, the via connects to a higher interconnect layer, which is used for less local routing — possibly global routing, depending on the exact layer. In a via last process, the via connects to the highest metal layer, which can be a Redistribution Layer (RDL) on top of the regular interconnect layers. The TSV can also merely be a connection between a die located below it and a die located above it. In this case, it will ideally not interact with the die it is located in. In practice, however, parasitics will cause unwanted interactions that may or may not be problematic.

In terms of difficulty of performing a correct extraction, via last ranks the highest. Because the TSV goes through the entire IMD, many parasitic components may manifest themselves. Conversely, the TSV does not reach the top of the IMD in a via first configuration. This means that the impact of the presence of via first TSVs on the extraction of the IMD is limited compared to via last TSVs. A via middle process requires a combination of the approaches used for via first and via last.

2.1.5 Configurations of multiple TSVs

TSVs often don’t appear in isolation. The presence or absence of other TSVs has an impact on the electrical characteristics of the TSV. All TSVs have a parasitic capacitance to the silicon substrate. For isolated TSVs this will basically be a capacitance to ground. If there are other TSVs nearby, TSV-to-TSV capacitances will manifest themselves. These other
TSVs can either carry a related current (as in a ground-signal pair) or an unrelated current. TSV-to-TSV capacitances can also exist in oxide layers.

2.1.6 Implications

Any extraction methodology is going to have to make assumptions about the TSVs it will have to extract. These assumptions come in two forms: assumptions about what kinds of designs the software needs to handle and assumptions about how a design is going to turn out in practice. Designers will sometimes do new or unusual things that weren’t thought of at the time the extraction methodology was decided on. Likewise, manufacturing technology need not behave the same way in the future as it does today, so the way a design is translated into a physical IC may change sometime.

Given that TSVs can have different shapes, it’s important to model them in such a way that all reasonable shapes can be extracted acceptably. It’s not unreasonable to model TSVs as cylindrical, as long as cuboid-shaped TSVs can also fit into this model without losing too much accuracy. If this is impossible, it would be better to come up with two models: one for cylinders and one for cuboids. Having a single model with the highest possible generality is to be preferred, however. Handling more exotic shapes is not important at present.

When it comes to extraction, the simplest case is that of a via first process. Because the TSV only goes through the silicon, the only parasitics that have to be modeled are those between the TSV and the silicon and between the TSV and the lowest interconnect layer. If the signal has to travel all the way from the connection to the TSV to an RDL several layers up, this will clearly need to be modeled as well, but this can be handled with existing extraction algorithms.

In a via middle process, the TSV goes through several layers of interconnect and IMD as well, introducing additional parasitics. The way these parasitics are extracted need not be consistent with the extraction of parasitics between regular interconnect layers, so it has to be taken into account as part of the TSV extraction methodology.

In a via last process, there are even more parasitics, because the TSV goes through the entire IMD and all metal layers. On top of that, it connects to the highest metal layer (normally an RDL) in a way that’s somewhat unusual, which could also mean that existing algorithms won’t necessarily extract the parasitics of such a connection correctly.

Besides the TSV itself, the RDL also has to be modeled. Because extraction methodologies that can extract interconnects properly are already available, this issue is not given much consideration in this thesis.

It’s important to note that being able to model and extract a single TSV is not enough. In practice, there will almost always be more than one TSV on a die and these different TSVs are likely to have parasitics between them, which are essentially a special case of the parasitics between the TSV and the substrate and those between the TSV and an oxide layer. Being able to model these parasitics properly for any reasonable configuration is of vital importance. It is advantageous that parasitics between TSVs that are spaced far apart will be vanishingly small, so the extraction methodology can employ a more or less local approach.
2.2 Extraction methodologies

There’s more than one way to perform layout to circuit extraction. In general, different needs imply different approaches. Broadly, two different approaches can be distinguished: formula-based extraction and physics-based extraction. Specific items that are difficult to extract correctly using either method can be kept apart. A model is then inserted into the circuit instead of performing a regular extraction. Such an approach can also be applied to the extraction of TSVs.

2.2.1 Formula-based extraction

Formula-based extraction considers the masks themselves and the relations between them. This information is then combined with a set of process-specific values to generate a circuit. For instance, if two points are connected in a straight line, the resistance between them can be computed based on the length and width of the connection and a scaling factor that represents both the interconnect thickness and its resistivity. The resistance of a via can be calculated in a similar but different manner. This approach can be extended to capacitances and even inductances. It’s also possible to extract the conductances and capacitances in the substrate by supplying typical values for certain geometries and interpolating if necessary.

2.2.2 Physics-based extraction

Physics-based extraction takes a different approach. Instead of processing the masks directly, they are first used to build a spatial model of the IC. This model is then combined with information about the various materials used in the IC to generate a circuit. The laws of physics form the basis of this extraction method. Because extraction deals with electric circuits, the relevant laws of physics are Maxwell’s laws and the laws that can be derived from them. For instance, the resistance between two points can be computed based on the resistivity of the conducting material and its geometry. Multiple paths can contribute to this value and the current can flow in three dimensions. This contrasts with formula-based extraction, because formula-based extraction is focused on the shape of the masks themselves, instead of the shape of the physical objects that the masks specify. Capacitances can be calculated based on the permittivity of the insulating material and the geometry of both the conducting material and the insulating material. Inductances can theoretically also be extracted based on Maxwell’s laws, although this is more complicated. Extraction of the substrate isn’t fundamentally different from extraction of the interconnect and IMD.

2.2.3 Model-based extraction

Not everything lends itself well to either formula-based or physics-based extraction. Transistors are a prime example. The specifics of transistor operation are so complex that it’s very difficult to get satisfying results with either approach, even if this is theoretically possible. In addition, transistors are typically dimensioned in only a limited number of different ways. Analog designs show a greater variety of geometries, but digital designs are usually based on
libraries which are built around a very small number of different sizes of transistors. In short, there are likely to be many millions of identical transistors in a single IC. It’s not unrealistic to hand-craft a circuit that models a transistor’s actual behavior very accurately. Such a model can be made for every kind of transistor that will occur in the circuit. This will yield excellent accuracy, but it can only work if designers limit themselves to a sufficiently low number of different transistors. Alternatively, the model can be made parametric and the extraction software can then extract the relevant parameters from the layout and pass them to the model. Model-based extraction can potentially be applied to TSVs as well.

2.3 Die bonding

When dies are bonded together, this can be done in three ways: front-to-front, front-to-back and back-to-back [4]. Front-to-front bondings don’t necessarily involve TSVs, but the other two always do. In case of back-to-back bonding, a TSV connects to a TSV of another die. In case of front-to-back bonding, a TSV either connects to the interconnect layer of another die or to a TSV of that die. This also applies to front-to-front bonding, if TSVs are involved at all. Whether this bonding is performed between two individual dies, between two complete wafers, or between a wafer and several dies, is not relevant to this thesis.

What is relevant, however, is the way these dies line up. There is a significant difference between front-to-back bonding and back-to-back (or front-to-front) bonding. The former bonds two dies with the same orientation, while the latter has to flip one of the two dies. This flipping can be done either horizontally or vertically. Either approach is equivalent to the other with a rotation of 180 degrees. Likewise, a die can also be rotated 180 degrees during front-to-back bonding, which is equivalent to flipping in both directions. Rotations over 90 or 270 degrees are also feasible, but angles that are not an integer multiple of 90 degrees do not make much sense in the mostly rectangular world of ICs. Implementation of such rotations is also far more complicated. Therefore, they will not be considered in this thesis.

While the issue of die bonding in 3D ICs has no immediate bearing on the way TSVs are extracted, it is an important aspect of the extraction procedure to consider. After all, being able to extract a TSV is no good if it can’t be connected to anything else.

More information about the geometry of 3D ICs can be found in [5].
3.1 Requirements

Modeling TSVs is an important first step towards being able to extract them. When selecting a model, several requirements have to be met:

1. The model should accurately describe the electrical characteristics of TSVs.
2. The model should be parametric.
3. The model should be applicable to a wide range of process technologies and TSV configurations.
4. The model should be as simple as possible, but not any simpler.
5. The model should offer opportunities for selective simplification, to reduce extraction and simulation times at the expense of accuracy.
6. The model should be based on the actual physics that govern TSVs.

These requirements are listed from most to least important.

The first requirement is trivial. Any model that does not describe the electrical characteristics of a TSV is not a model of a TSV. A model that describes them with insufficient accuracy may technically be a model of a TSV, but it is still not useful. Of course, the required accuracy is not the same for every application, but this is related to the fifth requirement.

The second requirement was included because there is a very wide range of different TSV configurations [6, 5]. It’s impossible to describe the electrical characteristics of all possible TSVs with a single, invariant model. Any model will have to allow for these differences or it will only be applicable to a small subset of TSVs.

The third requirement is related to the second one in that it specifies it further. A model that can only handle TSVs up to a diameter of 10 μm, for instance, will be useful in that specific niche, but there will still be a need for other models to cover other cases. Ideally, a single model can be used for all possible configurations.

The fourth requirement consists of two parts: “as simple as possible” and “not any simpler”. The second part is already covered by the first three requirements, so the focus is on the first part of this requirement. If this had not been the case, this would arguably be the most important requirement on the list. The reason that the model should be as simple as possible is threefold. First of all, simple models are easier to verify for correctness and less likely to exhibit unexpected and unwanted behavior. Secondly, a simple model will be faster during both extraction and subsequent simulation. Finally, implementing a simple model could potentially take less time and effort.
The fifth requirement builds on the fourth and a remark that was made during the discussion of the first requirement. Not all applications require the same accuracy. Rather than adopting a one-size-fits-all approach, it would be better if the model could be simplified when less accuracy is desired in specific applications. This would yield the same benefits in terms of extraction and simulation speed as having a simpler model in the first place, without the disadvantage of reducing accuracy for everybody.

The final requirement may seem like the odd one out — and in many ways it is. A model that is empirically fitted to measurement results may be perfectly accurate, but the model does not offer any insight into the workings of a TSV. It will also be difficult to verify the model for correctness. In fact, any formal verification would be impossible. Another disadvantage is that the model will generally only be accurate for TSVs that are similar to the one that was measured. Interpolation between a finite set of data points is possible and will usually yield acceptable results, but extrapolation is far more difficult to do correctly. Basing a model on known physical constants and laws of nature means the accuracy won’t be limited by the measurements that were performed to construct the model.

### 3.2 Theory

Electric circuit theory is based on four fundamental components: resistances, capacitances, inductances and memristors. Of these, resistances relate voltage to current, capacitances relate charge to voltage, inductances relate flux linkage to current and memristors relate flux linkage to charge. Resistances can alternatively be described as conductances, which relate current to voltage. Electric circuit theory deals with voltages and currents instead of flux linkages and charges. Therefore, capacitances are usually described either in terms of a relationship between current and the time derivative of voltage or in terms of a relationship between voltage and the time integral of current. Likewise, inductances are usually described either in terms of a relationship between voltage and the time derivative of current or in terms of a relationship between current and the time integral of voltage. This thesis won’t refer to these reciprocal descriptions of capacitances and inductances by their own names, unlike resistances and conductances. Memristors aren’t relevant to this thesis and won’t be covered here.

The values of resistances (or, equivalently, conductances) and capacitances can easily be calculated for regular structures (straight wires and parallel plate capacitors), based on their material properties:

\[
R = \frac{\rho}{A} \quad (3.1)
\]

\[
G = \frac{\sigma}{l} \quad (3.2)
\]

\[
C = \frac{\epsilon}{t} \quad (3.3)
\]

Here, \(\rho\) represents resistivity, \(\sigma\) represents its reciprocal, conductivity, \(l\) represents length and \(A\) represents area. The thickness of the capacitance’s insulator is given by \(t\) and its electric permittivity is given by \(\epsilon\). Unfortunately, there’s no simple formula that will calculate
the value of an inductance. Such calculations depend on assumptions, simplifications and relatively complicated mathematics.

One important phenomenon that occurs in practice, but is complicated to model within the boundaries of circuit theory, is the skin effect. In short, the skin effect describes a reduction in the effective area of a wire as the frequency of a signal increases. This implies that resistance goes up as frequency goes up, although the relationship between the two isn’t linear. Conversely, the wire’s inductance will actually go down. Because of the problems associated with modeling the skin effect correctly, this thesis ignores the effect entirely. An immediate consequence is that the model’s predictions for larger TSVs (in terms of radius) will almost certainly be too optimistic at higher frequencies.

This thesis neglects inductive effects, including the skin effect. The resulting simplification of the models that are used implies that TSVs that are affected significantly by inductance will not be modeled correctly. The reason for neglecting inductances is the limited time that was available to perform the work that lead to this thesis. Inductances are generally less important than resistances and capacitances in ICs, which is why they were considered the best option for simplification.

### 3.3 Models under consideration

Several different models were considered. Pak et al. [7] describe an electrical characterization of TSVs, but it is based on S-parameters instead of a circuit model. Bermond et al. [8] describe a similar methodology, although they do translate these S-parameters to an RLCG model. Because the intended result of extraction is a circuit description, this makes their model either impractical or completely useless for extraction purposes. S-parameters would have to be determined first and then translated to component values. The model proposed by Weerasekera et al. [9] does not have this drawback, but it contains numerical values that were derived experimentally instead of having their basis in real-world physics. An advantage of that model is that it can be applied to several different topologies of multiple TSVs. The model proposed by Bandyopadhyay et al. [10] does not take TSV-to-TSV parasitics into account. The same objection applies to the model that Katti et al. [11] propose. The model described by Kim et al. [3] uses the geometry of a TSV, which is assumed to be cylindrical, to assign values to its RLCG circuit. Parasitics between TSVs are an important part of this model, although it is tied to a specific ground-signal pair topology. Another disadvantage is that it does not model the capacitances through the substrate to ground. Another property is that it describes more than just a TSV, by also including a model for an RDL.

Table 3.1 lists the different options, along with information about how they perform with regard to three important criteria. Being based on a circuit instead of S-parameters was chosen as a criterion because using S-parameters would make the extraction process significantly more complex. It’s uncertain if models based on S-parameters can realistically be used at all. Another criterion is being based on physics, which is one of the requirements imposed on the model — although it’s the least important one. The final criterion is modeling of the inter-TSV parasitics. These are considered more important than parasitic conductances and capacitances to the rest of the IC, because it is assumed that TSVs are more likely to be placed together instead of in isolation, so the parasitics between them are dominant. Another
criterion that is important but not listed here is applicability to different topologies. As has been mentioned before, the model described by Kim et al. [3] does not meet this criterion entirely. As long as the model can be adapted to work in more than one topology this is not a major problem.

<table>
<thead>
<tr>
<th>Model</th>
<th>Type</th>
<th>Physics</th>
<th>Inter-TSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pak [7]</td>
<td>S-parameters</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Bermond [8]</td>
<td>Both</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Weerasekera [9]</td>
<td>Circuit</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Bandyopadhyay [10]</td>
<td>Circuit</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Katti [11]</td>
<td>Circuit</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Kim [3]</td>
<td>Circuit</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of compact models

Based on this comparison, the model described by Kim et al. [3] was selected. The RDL isn’t part of the model presented here. The main difference between the way the RDL is modeled in the original model and the way it would be extracted by Space is the presence of inductances in that model.

### 3.4 Resulting model

Some changes were made to the selected model for the sake of consistency and correctness, but the major difference is that the model used in this thesis omits inductances altogether. This way, the complexity is reduced. The model is shown in Figure 3.1. It describes two TSVs and their respective bumps, plus the parasitics between them. The changes that were made are listed in full at the end of this chapter. Chapter 6 describes the simulations that were performed to determine the validity of the simplified model.

Because this model is parametric, it is important to define the parameters first. These parameters are inputs to the model. Because both the radius and the diameter of parts of the structure are used, both parameters are defined here. This way, inconvenient multiplications or divisions with a factor of 2 are avoided wherever possible. The parameters can be found in Table 3.2. Some of these parameters describe the dimensions of the TSV. Which parameter refers to which dimension can be seen in Figure 3.2.

Using these parameters, the variables that were shown in Figure 3.1 can now be given their correct value. These values are given by Table 3.3, which is derived from the model presented by Kim et al. [3], but adapted for this thesis.

This model as it is presented here is sufficient to describe a structure of two TSVs with bumps and oxide layers, but it doesn’t cover cases with more than two TSVs. This problem can be solved, as long as care is taken to use a correct definition for \( p_{TSV} \). If a TSV has more than one neighbor, multiple instances of \( C_{\text{Underfill}}, C_{\text{Bottom}}, C_{\text{IMD}}, C_{\text{Si sub}} \) and \( G_{\text{Si sub}} \) will show up.

One remark should be made about the computation of \( C_{\text{Insulator}} \). The formula given here may seem unintuitive, but assuming that \( r_{TSV} \gg t_{ox} \) (recalling that, for \( x = 1 \), \( \ln x = 0 \) and...
\[
\frac{d}{dx} \ln x = 1 \]

It can be approximated as follows:

\[
C_{\text{Insulator}} = \epsilon_{\text{ox}} \frac{\pi (h_{\text{TSV}} - t_{\text{ox,bot}} - h_{\text{IMD}})}{\ln \left( \frac{r_{\text{TSV}} + t_{\text{ox}}}{r_{\text{TSV}}} \right)} \tag{3.4}
\]

\[
\ln \left( \frac{r + t}{r} \right) = \ln(r + t) - \ln(r) \tag{3.5}
\]

\[
\frac{\partial}{\partial r} (\ln(r + t) - \ln(r)) = \frac{1}{r + t} - \frac{1}{r} \tag{3.6}
\]

\[
\frac{\partial}{\partial r} (\ln(r + t) - \ln(r)) \approx -\frac{t}{r^2} \tag{3.7}
\]

\[
\ln \left( \frac{r + t}{r} \right) \approx \frac{t}{r} \tag{3.8}
\]

\[
C_{\text{Insulator}} \approx \frac{\epsilon_{\text{ox}} \pi r_{\text{TSV}}}{t_{\text{ox}}} (h_{\text{TSV}} - t_{\text{ox,bot}} - h_{\text{IMD}}) \tag{3.9}
\]

This approximation will not be used for extraction. It’s included here merely to justify the formula’s seemingly implausible form. In its simplified form, the formula for \(C_{\text{Insulator}}\) is
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSV radius</td>
<td>$r_{TSV}$</td>
</tr>
<tr>
<td>TSV diameter</td>
<td>$d_{TSV}$</td>
</tr>
<tr>
<td>TSV height</td>
<td>$h_{TSV}$</td>
</tr>
<tr>
<td>TSV-to-TSV pitch</td>
<td>$p_{TSV}$</td>
</tr>
<tr>
<td>Oxide thickness</td>
<td>$t_{ox}$</td>
</tr>
<tr>
<td>Bottom oxide thickness</td>
<td>$t_{ox,bot}$</td>
</tr>
<tr>
<td>IMD height</td>
<td>$h_{IMD}$</td>
</tr>
<tr>
<td>Bump height</td>
<td>$h_{Bump}$</td>
</tr>
<tr>
<td>Bump diameter</td>
<td>$d_{Bump}$</td>
</tr>
<tr>
<td>Bump radius</td>
<td>$r_{Bump}$</td>
</tr>
<tr>
<td>Bump resistivity</td>
<td>$\rho_{Bump}$</td>
</tr>
<tr>
<td>TSV resistivity</td>
<td>$\rho_{TSV}$</td>
</tr>
<tr>
<td>Substrate resistivity</td>
<td>$\rho_{Si}$</td>
</tr>
<tr>
<td>Substrate conductivity</td>
<td>$\sigma_{Si}$</td>
</tr>
<tr>
<td>Substrate permittivity</td>
<td>$\epsilon_{Si}$</td>
</tr>
<tr>
<td>Oxide permittivity</td>
<td>$\epsilon_{ox}$</td>
</tr>
<tr>
<td>Bottom oxide permittivity</td>
<td>$\epsilon_{ox,bot}$</td>
</tr>
<tr>
<td>IMD permittivity</td>
<td>$\epsilon_{IMD}$</td>
</tr>
<tr>
<td>Underfill permittivity</td>
<td>$\epsilon_{Underfill}$</td>
</tr>
</tbody>
</table>

Table 3.2: Parameters of the TSV model, based on [3]

similar to that for $C_{Bump1}$ and $C_{Bump2}$. The formulas for the inter-TSV parasitics show similarities without any simplification.

A two-dimensional configuration of multiple TSVs will not only contain horizontal and vertical TSV parasitics, but diagonal parasitics as well. Assuming that $p_{TSV}$ is isotropic, its value only has to be multiplied by $\sqrt{2}$ to compute the diagonal inter-TSV parasitics.

### 3.5 Differences with the original model

#### 3.5.1 Improvements to the model

Because some minor flaws were found in the model as it is described by Kim et al. [3], this model contains a few small improvements over that version of the model, namely:
1. \( R_{\text{Bump}} \) was replaced by \( 2R_{\text{Bump}} \). There’s a bump at both ends of the TSV, so both contribute \( R_{\text{Bump}} \) to the total resistance. Kim et al. [3] clearly define \( h_{\text{Bump}} \) as the height of a single bump, so this value has to be doubled to take the full resistance into account. Of course, \( R_{\text{Bump}} \) will generally be significantly smaller than \( R_{\text{TSV}} \), so this change doesn’t make a very big difference.

2. \( h_{\text{TSV}} \) is redefined to include \( t_{\text{ox}, \text{bot}} \). Kim et al. [3] include \( h_{\text{IMD}} \) but not \( t_{\text{ox}, \text{bot}} \), which was probably an accidental omission. The effect of this change should be negligible, given the low value of \( t_{\text{ox}, \text{bot}} \).

3. Some parameters that were used incorrectly by Kim et al. [3] were corrected. For instance, there appeared to be some confusion between \( t_{\text{ox}} \) and \( t_{\text{ox}, \text{bot}} \). Based on structural arguments, the correct parameter was substituted where appropriate. Again, the effect of this change should be negligible.

### 3.5.2 Simplifications of the model

In order to be able to implement the model in Space, some simplifications had to be made. These simplifications are:

1. Inductances were removed from the model completely. This will affect simulation results at higher frequencies.

2. The skin effect was also removed from the model. Like the removal of inductances, this will affect simulation results at higher frequencies.
Table 3.3: Variables of the TSV model, based on [3]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{TSV}$</td>
<td>$\frac{\rho_{TSV} h_{TSV}}{\pi r_{TSV}^2}$</td>
</tr>
<tr>
<td>$R_{Bump}$</td>
<td>$\frac{\rho_{Bump} h_{Bump}}{\pi r_{Bump}^2}$</td>
</tr>
<tr>
<td>$C_{Insulator}$</td>
<td>$\frac{\epsilon_{ox}}{\pi} \left( h_{TSV} - t_{ox,bot} - h_{IMD} \right) \frac{\pi h_{Bump}}{\ln \left( \frac{r_{TSV} + t_{ox}}{r_{TSV}} \right)}$</td>
</tr>
<tr>
<td>$C_{Bump1}$</td>
<td>$\frac{\epsilon_{ox,bot}}{t_{ox,bot}} \pi \left( \frac{r_{Bump}^2}{r_{TSV}^2} - \left( r_{TSV} + t_{ox} \right)^2 \right) $</td>
</tr>
<tr>
<td>$C_{Bump2}$</td>
<td>$\frac{\epsilon_{IMD}}{h_{IMD}} \pi \left( \frac{r_{Bump}^2}{r_{TSV}^2} - \left( r_{TSV} + t_{ox} \right)^2 \right) $</td>
</tr>
<tr>
<td>$C_{Underfill}$</td>
<td>$\frac{\epsilon_{Underfill}}{acosh \left( \frac{r_{TSV}}{d_{Bump}} \right)}$</td>
</tr>
<tr>
<td>$C_{Bottom}$</td>
<td>$\frac{\epsilon_{ox,bot}}{acosh \left( \frac{r_{TSV}}{d_{T SV}} \right)}$</td>
</tr>
<tr>
<td>$C_{IMD}$</td>
<td>$\frac{\epsilon_{IMD}}{acosh \left( \frac{r_{TSV}}{d_{TSV}} \right)}$</td>
</tr>
<tr>
<td>$C_{Si,sub}$</td>
<td>$\frac{\epsilon_{Si}}{acosh \left( \frac{r_{TSV}}{d_{TSV}} \right)} \pi \left( h_{TSV} - t_{ox,bot} - h_{IMD} \right)$</td>
</tr>
<tr>
<td>$G_{Si,sub}$</td>
<td>$\sigma_{Si} \frac{\epsilon_{Si}}{acosh \left( \frac{r_{TSV}}{d_{TSV}} \right)} \pi \left( h_{TSV} - t_{ox,bot} - h_{IMD} \right)$</td>
</tr>
</tbody>
</table>

Spice simulations were performed to compare the original model to its simplified version. The results can be found in Chapter 6.

3.5.3 Other changes in the model

Besides the improvements and simplifications, a few other changes were made. These changes do not affect the electrical behavior of the model. They are as follows:

1. The names of $C_{Bump1}$ and $C_{Bump2}$ were switched around for a more intuitive, bottom-up numbering scheme.
2. Where appropriate, $d_{TSV}$ and $d_{Bump}$ were replaced by $r_{TSV}$ and $r_{Bump}$, respectively. This way, unnecessary factors of 2 could be eliminated.
3. Factors that cancel each other were also eliminated.
4. Several equations were restructured to improve consistency.

5. All occurrences of $\epsilon_0 \epsilon_r$ were replaced by a single $\epsilon$ with the appropriate subscript.
4.1 Formula-based extraction

Formula-based extraction can be applied to TSVs as long as the methodology is implemented in such a way that there is enough expressiveness to allow a TSV to be described accurately. The difficulty of actually preparing a formula-based extraction is greatly influenced by the design of the implementation platform.

In order to extract $R_{TSV}$ and $R_{Bump}$, both their area and the product between their resistance and their area have to be known. The area can theoretically be determined from the layout, but the user has to specify the area-resistance product. This can be computed as follows:

$$A_{TSV} R_{TSV} = \rho_{TSV} h_{TSV}$$  \hspace{1cm} (4.1)

$$A_{Bump} R_{Bump} = \rho_{Bump} h_{Bump}$$  \hspace{1cm} (4.2)

The extraction software can then divide this value by the area it has extracted from the layout to calculate the values of $R_{TSV}$ and $R_{Bump}$.

However, the other component values have complex non-linear relations to the parameters of the TSV. It would be pointless to describe a formula-based extraction method here that cannot be implemented in Space. Instead, a different approach will be chosen. Model-based extraction is used as a template for formula-based extraction. Instead of trying to generate a technology file that will allow Space to perform formula-based extraction of any TSV topology, which is not possible under Space’s current implementation of formula-based extraction, a single TSV shape is assumed for the entire die. Then all values of this TSV’s components can be calculated using the compact model. These formulas are then altered such that if Space extracts the layout representing the TSV, it will produce the correct results. The implementation of this approach will be described in Chapter 5.

4.2 Physics-based extraction

Physics-based extraction can be applied to TSVs as long as the implementation platform is powerful enough. For instance, an implementation platform that assumes that the substrate consists of only a single semiconductor material (and added dopants) will have to be adapted to allow TSVs to be extracted. This is because the presence of a TSV constitutes an interruption of the semiconductor, with a conductor surrounded by an insulator taking its place. If it’s impossible or impractical to perform this adaptation, physics-based extraction is not a viable alternative.
Assuming that physics-based extraction is possible, the layout has to be translated into a spatial model in some way. This can lead to several complications. If the TSVs are cylindrical while the layout is based on straight lines, some kind of translation has to be performed. This becomes even more complicated if both cylindrical and cuboid-shaped TSVs can be present in the same design, because there has to be some way to distinguish between them. Furthermore, the thickness of the insulator must either be specified by the layout, which increases its complexity, or inferred through other means. Specifying the thickness of the horizontal insulator layers can be done in the same way as it is for regular designs, so this does not pose an additional problem.

4.3 Model-based extraction

Model-based extraction using a parametric model can be done in two ways. One is to implement the parametric model in such a way that it can accept parameters from the extraction software. The implementation of the model would then behave according to the parameters it is given. In order for this to work, however, the layout has to contain all the necessary information that the model requires. If this difficult or impossible to ascertain, it’s also possible to implement the model the other way around. Instead of using the layout to extract the parameters for the model, both the layout and the circuit are generated simultaneously based on the parameters the user enters.

The second option was chosen for this thesis. Given that Space was chosen as the implementation platform, there are certain difficulties in creating a complete set of masks for a TSV, especially if its cross-section is cylindrical. Therefore, it is more convenient to generate a dummy layout component that can be used to specify a TSV. The circuit will be generated as a model that Spice-based simulators can understand.

A user wishing to add a TSV to a design can run a layout and model generator. This utility will accept the parameters of the TSV as input, as listed in Table 3.2. Some of these parameters are redundant, so not all of them have to be specified separately. These parameters are then used to generate both a layout and a Spice model.

Generating the layout is relatively simple. Because the layout will only be used as a dummy, it can be chosen almost at will, as long as its bounding box is correct. Of course, a user looking at his design will want his screen to display something that will at least come somewhat close to the design he intended, but this is of secondary importance.

Generating the Spice model is slightly more complicated. The layout and model generator has all the relevant parameters at its disposal, so to calculate the various component values, it can use the formulas given in Table 3.3. The result then has to be written to a file that also describes the circuit topology.

This circuit topology, however, cannot be the entirety of the model shown in Figure 3.1. Only a single TSV is included in the layout, so the Spice model should also refer to a single TSV. This means that only $R_{TSV}$, $R_{Bump}$, $C_{Insulator}$, $C_{Bump1}$ and $C_{Bump2}$ can be part of the Spice model — and only for one TSV. The inter-TSV parasitics have to be extracted in a different way, using either of the two other extraction methodologies.
4.4 Integral simulation

A prerequisite for the integral simulation of all dies in an IC is that the coordinates of the terminals of one die can be compared to the coordinates of the terminals of another die. Assuming that these coordinates can be scaled correctly to identical length units, the remaining obstacle is that of orientation differences. Given that dies may both be flipped horizontally or vertically and rotated any integer multiple of 90 degrees, the different coordinate systems have to be reconciled in some way.

Figure 4.1 shows the issue of flipping dies. The bottom-left corner of the picture shows a die. To its right, the die is shown flipped horizontally. Above it, the die is shown flipped vertically. The top-right corner of the picture shows the die flipped in both directions — or, equivalently, rotated half a turn.

![Figure 4.1: A die, flipped in both directions and rotated half a turn](image)

Figure 4.2 shows three of these die orientations in color, with its top-right corner being different. The rotated version of the original die is left out. Instead, the colored dies in the other three corners of the image are superimposed on each other. This shows that, in both back-to-back and front-to-front bonding, a corner on one of the dies will connect to a different corner on the die it is bonded to.

![Figure 4.2: A die, flipped in both directions and superimposed on itself](image)
Mathematically, flipping a die in one direction is equivalent to negating one of its planar coordinates. With the ability to flip a die in both directions simultaneously it’s possible to rotate that die half a turn. Rotating the die a quarter turn, however, can not be accomplished in this manner. Such a rotation requires somewhat more complicated arithmetic which is best implemented by using complex numbers. One multiplication with the imaginary unit is equivalent to a rotation of a 90 degrees in the complex plane. By combining a 90 degree rotation with a 180 degree rotation, a 270 degree rotation can be obtained. Figure 4.3 shows all four orientations of a die when it is rotated over integer multiples of 90 degrees.

Arithmetically, these two abilities — the ability to negate both coordinates separately and the ability to multiply both coordinates, interpreted as a complex number, by the imaginary unit — together allow the coordinates of dies that may be rotated integer multiples of 90 degrees and flipped in any way back to standard coordinates. This means that the terminals of any die can be compared geometrically to the terminals of any other die, irrespective of both their orientations.

Figure 4.3: A die, rotated a quarter turn, rotated half a turn and rotated three quarters of a turn
5.1 Formula-based extraction

Although the compact model assumes cylindrical TSVs, the layouts supported by Space are rectangular or octagonal. In order to use rectangular layouts to derive results for TSVs with a circular cross-section, correction factors have to be added that convert areas and distances from the rectangular layout to their circular equivalent. Instead of being circular, the cross-section in the layout extracted by Space will be a square, with all four sides of the square having a length equal to the diameter of the circle it represents. In other words, the square can be thought of as the circle’s bounding box.

For $R_{TSV}$, Space will divide the value in the technology file by the area of the TSV. Even though the actual TSV has a cylindrical cross-section, the layout specifies a square. Therefore, the value of the resistance has to be multiplied by the area seen by Space:

$$A_{TSV} = d^2_{TSV} = 4r^2_{TSV} \quad (5.1)$$

For $R_{Bump}$, the situation is slightly more complicated. Since $R_{TSV}$ and $R_{Bump}$ together form a single resistance in the compact model, their extraction has to be based on a single contact. The area of this contact is given by Equation 5.1. This means that the values of $R_{TSV}$ and $R_{Bump}$ are multiplied by the same area to compute the value in the technology file, even though the area of the TSV is smaller than the area of the bump. Because of this, the formula for $R_{Bump}$ is more complex than the one for $R_{TSV}$.

$C_{Insulator}$ is an edge capacitance, which means Space will multiply the value in the technology file by the combined length of the contact’s edges. Once again, these are the edges of a square, because that is how the layout is specified. Before it can be added to the technology file, the value of the capacitance has to be divided by the following correction factor:

$$l_{Insulator} = 4d_{TSV} \quad (5.2)$$

$C_{Bump1}$ and $C_{Bump2}$ are surface capacitances, which means their value is computed by multiplying the value in the technology file by the part of the area of the bump that is not connected to the TSV. In other words, the area of the TSV has to be subtracted from the area of the bump. Once again, the area to be computed is the one in the layout, not the physical one. This means that both $C_{Bump1}$ and $C_{Bump2}$ have to be divided by the following correction factor:

$$A_{Bump1,Bump2} = d^2_{Bump} - d^2_{TSV} \quad (5.3)$$

$C_{Underfill}$ is a lateral capacitance, which means that Space will multiply the value in the technology file by the length of the two edges that the capacitance is formed between. These
two edges are of equal length. Since $C_{\text{Underfill}}$ is formed between the bumps of two TSVs, its value has to be divided by the length of an edge of the bump in the layout:

$$l_{\text{Underfill}} = d_{\text{Bump}}$$ (5.4)

$C_{\text{Bottom}}$ and $C_{\text{IMD}}$ are both also lateral capacitances, but they are formed between the TSVs themselves instead of the bumps. Consequently, their value has to be divided by the length of an edge of the TSV in the layout:

$$l_{\text{Bottom, IMD}} = d_{\text{TSV}}$$ (5.5)

$G_{\text{Si sub}}$ is a substrate conductance. Its value can be specified directly in the technology file, although Space expects a resistance instead of a conductance. This is not a problem, because the resistance is the reciprocal of the conductance and the conductance is strictly positive. To compute $C_{\text{Si sub}}$, which is a substrate capacitance, Space multiplies the value of $G_{\text{Si sub}}$ by a time constant which is the ratio of $C_{\text{Si sub}}$ and $G_{\text{Si sub}}$:

$$\tau_{\text{RC}} = \frac{\varepsilon_{\text{Si}}}{\sigma_{\text{Si}}}$$ (5.6)

Finally, all values that depend on $p_{\text{TSV}}$ ought to be computed twice: once with the actual value of $p_{\text{TSV}}$ and once with a value of $p_{\text{TSV}} \sqrt{2}$. The latter value can be used for the computation of diagonal inter-TSV parasitics. It should be noted, however, that the current version of Space isn’t able to extract diagonal substrate conductances and capacitances correctly. Due to the way the substrate is modeled, of the two diagonal conductances and capacitances that are present in a configuration with a TSV in each of the four corners of a square, only one is generated. Therefore, it’s safest to configure Space not to generate them at all.

### 5.1.1 Additions to the technology file

After computing the various values required to perform the extraction, they have to be made available to Space. The important elements that need to be specified are listed below.

#### 5.1.1.1 Physical masks

The masks that are part of a certain technology are listed in the `maskdata` file. In addition to other masks used by the design, three masks need to be part of this file to perform formula-based extraction of TSVs:

1. A metal layer below the substrate, containing the bottom bumps
2. A top metal layer, containing the top bumps
3. A TSV layer

These masks can be named at will. The `maskdata` file requires some additional data to be specified, most of which is not important for the extraction process. There is one column that specifies the type of the mask. This should be set to interconnect (1) for both masks containing the bumps, since that is where the terminals will be defined.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value in technology file</th>
<th>Value after extraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{TSV}$ Ωm$^2$ ⇒ Ω</td>
<td>$\rho_{TSV} h_{TSV} \frac{4}{\pi}$</td>
<td>$\frac{\rho_{TSV} h_{TSV}}{2 \pi^2}$</td>
</tr>
<tr>
<td>$R_{Bump}$ Ωm$^2$ ⇒ Ω</td>
<td>$\rho_{Bump} h_{Bump} \frac{d^2_{TSV}}{\pi^2 Bump}$</td>
<td>$\frac{\rho_{Bump} h_{Bump}}{4 \pi^2}$</td>
</tr>
<tr>
<td>$C_{Insulator}$ F/m ⇒ F</td>
<td>$\epsilon_{ox} \frac{\pi (h_{TSV} - t_{ox,bot} - h_{IMD})}{\ln \left( \frac{h_{TSV} + t_{ox}}{h_{TSV}} \right) 4 d_{TSV}}$</td>
<td>$\frac{\epsilon_{ox} (h_{TSV} - t_{ox,bot} - h_{IMD})}{\ln \left( \frac{h_{TSV} + t_{ox}}{h_{TSV}} \right)}$</td>
</tr>
<tr>
<td>$C_{Bump1}$ F/m$^2$ ⇒ F</td>
<td>$\frac{\epsilon_{ox,bot} \pi \left( r^2_{Bump} - (r_{TSV} + t_{ox})^2 \right)}{t_{ox,bot} \left( d^2_{Bump} - d^2_{TSV} \right)}$</td>
<td>$\frac{\epsilon_{ox,bot} \pi \left( r^2_{Bump} - (r_{TSV} + t_{ox})^2 \right)}{t_{ox,bot} \left( d^2_{Bump} - d^2_{TSV} \right)}$</td>
</tr>
<tr>
<td>$C_{Bump2}$ F/m$^2$ ⇒ F</td>
<td>$\frac{\epsilon_{IMD} \pi \left( r^2_{Bump} - (r_{TSV} + t_{ox})^2 \right)}{h_{IMD} \left( d^2_{Bump} - d^2_{TSV} \right)}$</td>
<td>$\frac{\epsilon_{IMD} \pi \left( r^2_{Bump} - (r_{TSV} + t_{ox})^2 \right)}{h_{IMD} \left( d^2_{Bump} - d^2_{TSV} \right)}$</td>
</tr>
<tr>
<td>$C_{Underfill}$ F/m ⇒ F</td>
<td>$\frac{\epsilon_{Underfill} \pi h_{Bump}}{acosh \left( \frac{h_{TSV}}{d_{Bump}} \right) d_{Bump}}$</td>
<td>$\frac{\epsilon_{Underfill} \pi h_{Bump}}{acosh \left( \frac{h_{TSV}}{d_{Bump}} \right) d_{Bump}}$</td>
</tr>
<tr>
<td>$C_{Bottom}$ F/m ⇒ F</td>
<td>$\frac{\epsilon_{ox,bot} \pi t_{ox,bot}}{acosh \left( \frac{h_{TSV}}{d_{TSV}} \right) d_{TSV}}$</td>
<td>$\frac{\epsilon_{ox,bot} \pi t_{ox,bot}}{acosh \left( \frac{h_{TSV}}{d_{TSV}} \right) d_{TSV}}$</td>
</tr>
<tr>
<td>$C_{IMD}$ F/m ⇒ F</td>
<td>$\frac{\epsilon_{IMD} \pi h_{IMD}}{acosh \left( \frac{h_{IMD}}{d_{TSV}} \right) d_{TSV}}$</td>
<td>$\frac{\epsilon_{IMD} \pi h_{IMD}}{acosh \left( \frac{h_{IMD}}{d_{TSV}} \right) d_{TSV}}$</td>
</tr>
<tr>
<td>$C_{Si,sub}$ s ⇒ F</td>
<td>$\frac{\epsilon_{Si}}{\sigma_{Si}}$</td>
<td>$\frac{\epsilon_{Si} \pi \left( h_{TSV} - t_{ox,bot} - h_{IMD} \right)}{acosh \left( \frac{h_{TSV}}{d_{TSV}} \right) d_{TSV}}$</td>
</tr>
<tr>
<td>$G_{Si,sub}$ Ω ⇒ S</td>
<td>$\frac{\rho_{Si} \pi \left( h_{TSV} - t_{ox,bot} - h_{IMD} \right)}{acosh \left( \frac{h_{TSV}}{d_{TSV}} \right) d_{TSV}}$</td>
<td>$\frac{\rho_{Si} \pi \left( h_{TSV} - t_{ox,bot} - h_{IMD} \right)}{acosh \left( \frac{h_{TSV}}{d_{TSV}} \right) d_{TSV}}$</td>
</tr>
</tbody>
</table>

Table 5.1: Variables of the TSV model translated to parameters in the Space technology file

5.1.1.2 Pseudo-masks

In principle, the three physical masks allow connections to a TSV to be made at three points: the bottom bump, the top bump and the TSV itself. However, this leads to an ambiguity. Since the TSV connects the bumps together with a certain resistance, it’s not clear where a connection would go: the top, the bottom, or somewhere in between? As Figure 3.1 shows, the resistance of both the TSV itself and the bumps is lumped into one component, so only two circuit nodes would seem to suffice. Because of the conceptual differences between the parasitics between the bumps and the parasitics between the TSVs, four circuit nodes are preferred. There would be two at the top and two at the bottom. The lumped resistance would be placed between the bottom TSV node and the top TSV node. The rest of the circuit
would connect to the bottom TSV node or the top TSV node.

This can be accomplished by specifying two pseudo-masks in the `space.def.s` file. One would be defined as the intersection of the bottom bump and the bottom TSV, the other would be defined as the intersection between the top bump and the top TSV. These new pseudo-masks can then be used in subsequent sections of the technology file.

5.1.1.3 Conductors

Because a TSV is a vertical structure, its resistance is defined in the vertical direction. Therefore, the resistance of the four masks can be set to zero. If, however, one of the masks (most likely the one containing the top bumps) is also used a regular interconnect layer, its resistance should be specified as it would otherwise be. This has no bearing on the extraction of TSVs.

5.1.1.4 Contacts

In order to implement the model correctly, the two bump masks should be connected to their respective TSV pseudo-masks with a resistance of zero. The connection between the two pseudo-masks should specify the resistance given by Table 5.1, adding the values for $R_{TSV}$ and $2R_{Bump}$ together.

5.1.1.5 Capacitances

Specifying the capacitances is more complicated than specifying the contact resistance. Furthermore, it requires different approaches for different capacitances. The various capacitances will all be treated separately.

As can be seen in Figure 3.1, there are two different instances of $C_{\text{Underfill}}$. One instance is located between the bottom bumps and the other instance is located between the top bumps. They should be specified as lateral capacitances in `space.def.s`, between one edge of their respective masks and the opposite edge. Their value is given by Table 5.1 and their distance should be specified as $p_{TSV} - d_{Bump}$. In order to distinguish between the two instances, one capacitance should be named $C_{\text{Underfill1}}$ and the other $C_{\text{Underfill2}}$. Other names are also possible, as long as they are distinct.

Specifying $C_{\text{Bottom}}$ is not much different, given that it is also a lateral capacitance. It is located between the bottom nodes of the TSVs. The other important difference with $C_{\text{Underfill}}$ is that its distance should be specified as $p_{TSV} - d_{TSV}$ instead. The same approach can be used for $C_{\text{IMD}}$. It differs in that it is located between the top nodes of the TSVs instead of the bottom nodes.

There are also two different instances of $C_{\text{Insulator}}$. One instance is located between the bottom node of the TSVs and the substrate and the other instance is located between the top node and the substrate. The value of the two instances is identical and can be found in Table 5.1, which is the only number that needs to be specified. There’s no need to specify a distance.
for edge (or surface) capacitances. Once again, both instances should be given unique names. The substrate is denoted by the special mask $@\text{sub}$.

Finally, $C_{\text{Bump1}}$ and $C_{\text{Bump2}}$ are surface capacitances located between the bottom bump and the substrate and between the top bump and the substrate, respectively. Their values can be found in Table 5.1, but they are not identical.

### 5.1.1.6 Substrate

In order to extract the substrate, Space requires information about both parasitic conductances to the substrate node and parasitic coupling conductances between other nodes. The model in Figure 3.1 does not specify any conductances to the substrate (or ground), only coupling conductances. This means that the conductances have to be specified such that only the coupling conductances remain for pairs of TSVs.

When specifying conductances to the substrate node, several parameters need to be specified in addition to the conductance itself. Furthermore, the conductance itself has to be specified as a resistance. Its value can be found in Table 5.1. In addition to the resistance, the minimum conductance that remains after all coupling conductances have been subtracted should also be specified — as zero.

Space requires a value for the area of the substrate connection and one for its perimeter. If no other sources of substrate specifications are available, three sets of values should be specified. This way, Space can perform interpolation correctly. The first line should contain the area and perimeter of the TSV itself. The second line should contain the difference between the area of the bump and the area of the TSV itself, along with the sum of their perimeters. The third line should contain the area and perimeter of the bump. They correspond to the TSV itself, the part of the bump that does not cover the TSV and the entire bump, respectively.

If another source of substrate specifications is available, only the second line is required, as that is the one that describes the actual substrate terminal.

Finally, a time constant should be specified in `space.def.p` to allow Space to generate a substrate capacitance for every extracted substrate conductance. Its value can be found in Table 5.1, like the other values.

### 5.2 Model-based extraction

#### 5.2.1 Output file formats

Implementing model-based extraction for use with Space requires that both the model and the layout are generated in a format that Space can understand. This means that the right file formats have to be chosen. For layout generation, the LDM file format was chosen. This is the preferred format for Space — LDM files can easily be imported into the Nelsis database. Another advantage is that it’s a relatively straightforward ASCII-based format. For model generation, Spice subcircuits were chosen. Space comes with a utility called `putdevmod`, which can be used to import files that contain a Spice subcircuit along with a small header.
5.2.2 Input file format

Another important issue is the input file format. In order to make the solution future-proof, XML was chosen as the basis of the file format. Its syntax will now be discussed in detail.

The `<cells>` element is the root element. The `<cells>` element contains two kinds of elements: `<count>` and `<cell>`. There should be exactly one `<count>` element and it should come before the `<cell>` elements. The number of `<cell>` elements should be equal to the number specified inside the `<count>` element.

The `<cell>` element specifies a cell that will describe a TSV. It should contain 16 elements, in the following order:

1. `<name>`
2. `<model>`
3. `<layout>`
4. `<bottom>`
5. `<via>`
6. `<top>`
7. `<substrate>`
8. `<tsv>`
9. `<bump>`
10. `<oxide>`
11. `<boxide>`
12. `<imd>`
13. `<underfill>`
14. `<silicon>`
15. `<pitch>`
16. `<lambda>`

The `<name>` element contains the name of the cell. The `<model>` element contains the name of the file the model should be written to. The `<layout>` element contains the name of the file the layout should be written to. The `<bottom>` element contains the name of the bottom bump mask. The `<via>` element contains the name of the TSV mask. The `<top>` element contains the name of the top bump mask. The `<substrate>` element contains the name of the mask that can be used to make substrate connections. The elements `<tsv>` through `<silicon>` contain other elements and will be discussed separately. The `<pitch>` element contains the TSV-to-TSV pitch, in m. The `<lambda>` element contains the value of λ (the minimum feature size — all distances in layouts are expressed in units of λ) that Space expects, in m.
The `<tsv>` element describes the geometric and electrical properties of the TSV (excluding the bump). The `<bump>` element describes the geometric properties of the bump. These elements should contain 3 elements, in the following order:

1. `<diameter>`
2. `<height>`
3. `<resistivity>`

The `<diameter>` element contains the diameter of either the TSV or the bump, in m. The `<height>` element contains its height, in m. The `<resistivity>` element contains its resistivity, in Ωm.

The `<oxide>` element describes the geometric and electrical properties of the oxide between the TSV and the silicon substrate. The `<boxide>`, `<imd>` and `<underfill>` elements do the same for the bottom oxide, IMD and underfill, respectively. All four are electrical insulators. These elements should contain 2 elements, in the following order:

1. `<thickness>`
2. `<permittivity>`

The `<thickness>` element contains the thickness or height of the insulator, in m. The `<permittivity>` element contains its relative permittivity.

The `<silicon>` element described the electrical properties of the silicon substrate. It should contain 2 elements, in the following order:

1. `<permittivity>`
2. `<conductivity>`

The `<permittivity>` element contains the relative permittivity of the silicon substrate. The `<conductivity>` element contains its conductivity, in S/m.

An example of an input file can be found in Appendix A.

5.2.3 Model generation

Generating a model means performing the computations described by the formulas making up the model and then translating the result to the required format. No special steps have to be taken.

The structure of the device model file is simple. The first line specifies the name of the cell, the second line signals the start of the model, the third line specifies the names of the terminals, the fourth line repeats the name of the cell and specifies the numbers of the terminals nodes, the fifth line specifies the value of $R_{TSV} + 2R_{Bump}$, the sixth line specifies the value of $C_{Insulator} + C_{Bump1}$, the seventh line specifies the value of $C_{Insulator} + C_{Bump2}$, the eighth line repeats the name one more time and the final line ends the file. The file was reformatted to fit.

device name  
begin spicemod
In the first, fourth and eighth lines, \texttt{name} will be replaced by the name of the cell. In the fifth line, \texttt{resistance} will be replaced by the value of $R_{TSV} + 2R_{Bump}$, in $\Omega$. In the sixth line, \texttt{capacitance} will be replaced by the value of $C_{\text{Insulator}} + C_{\text{Bump1}}$, in F. In the seventh line, \texttt{capacitance} will be replaced by the value of $C_{\text{Insulator}} + C_{\text{Bump2}}$, in F.

This Spice model does not include the inter-TSV parasitics, so they have to be extracted separately using a different approach. To facilitate the implementation of a formula-based extraction, the layout and model generator also generates the required values that need to be entered into Space’s technology file.

\subsection*{5.2.4 Layout generation}

In order to generate a layout that matches reasonably well with what the actual TSV looks like, the cross-section of the TSV is represented with a square instead of a circle. The length of its sides is equal to the specified diameter of the circle. This way, the bounding box will have the correct size, but the area will be off by a factor of $4/\pi$. The bumps are represented in the same manner.

After division by $\lambda$, the value of $d_{\text{Bump}}$ is used to specify the location of the bump. The bottom left and top right coordinate of the TSV are calculated in a slightly more complicated manner, leading to the following set of equations:

\begin{align*}
BL_{\text{Bump}} &= 0 \\
TR_{\text{Bump}} &= \frac{d_{\text{Bump}}}{\lambda} \\
BL_{\text{TSV}} &= \frac{r_{\text{Bump}} - r_{\text{TSV}}}{\lambda} \\
TR_{\text{TSV}} &= \frac{r_{\text{Bump}} + r_{\text{TSV}}}{\lambda}
\end{align*}

After conversion of these coordinates to an unsigned integer, they can be written to the LDM file. Because all shapes are square, the x-coordinates and y-coordinates are identical.

The structure of the LDM file is simple. The first line specifies the name of the cell, the second, fourth and fifth lines define terminals, the third line defines a square in the TSV mask and the last line ends the file.
In the first line, name will be replaced by the name of the cell. In the second line, name will be replaced by the name of the bottom bump mask. In the third line, name will be replaced by the name of the TSV mask. In the fourth line, name will be replaced by the name of the bottom bump mask. In the fifth line, name will be replaced by the name of the mask that can be used to make substrate connections. In the second through fifth lines, x1 will be replaced by the leftmost coordinate of the object it refers to, xr will be replaced by its rightmost coordinate, yb will be replaced by its bottom coordinate and yt will be replaced by its top coordinate.

5.2.5 Additions to the technology file

Because the extraction of inter-TSV parasitics has to be performed using formula-based extraction, some of the same additions to the technology file are required. Furthermore, the model-based approach requires a few specific additions of its own.

5.2.5.1 Masks

Model-based extraction requires the same masks as formula-based extraction, plus one. Both the physical masks and the pseudo-masks are required. The extra mask is not a physical mask, although it does have to be specified in the maskdata file. It is the mask that will be used to implement connections to the substrate. Like the other masks, it can be given any name, as long as the name in the maskdata file matches the name used by the layout and model generator.

5.2.5.2 Conductors

The conductors that need to be specified are the same as for formula-based extraction, with the addition of a conductor for the mask that is used to create substrate connections. Its resistance should be set to zero.

5.2.5.3 Contacts

The contacts between the bumps and the TSV pseudo-masks are the same as for formula-based extraction, but the contact between the two pseudo-masks should be omitted, because it is already part of the model. Instead, a connection should be made between the mask that is used to create substrate connections and the special mask called @sub. Its value should be zero. This ensures that Space will generate the appropriate substrate terminals.

5.2.5.4 Capacitances

The capacitances $C_{\text{Underfill}}$ (both instances), $C_{\text{Bottom}}$ and $C_{\text{IMD}}$ should be specified exactly as they are for formula-based extraction. The capacitances $C_{\text{Insulator}}$ (both instances), $C_{\text{Bump1}}$ and $C_{\text{Bump2}}$ are already part of the model, so they should not be specified here.
5.2.5.5 Substrate

The substrate parameters should be specified exactly as they are for formula-based extraction.

5.2.6 Usage of the layout and model generator

Usage of the layout and model generator is relatively simple. The command line options are as follows:

```bash
tsv input [e_capacitance] [a_capacitance] [c_resistance]
```

Here, `input` is the name of the input file, `e_capacitance` is the value of the scaling factor that is used for edge capacitances (and, in our case, lateral capacitances) in the technology file, `a_capacitance` is the scaling factor for area capacitances and `c_resistance` is that for contact resistance. These scaling factors are all optional, although it is not possible to omit one scaling factor but include another one that should come after it on the command line. Scaling factors that are not specified are automatically set to 1 by the layout and model generator.

Besides the model and layout files, the layout and model generator will also emit values to be added to the technology file for formula-based extraction. Using these values (or computing them in some other way) is a requirement for the extraction of inter-TSV parasitics, although some values remain unused in case of model-based extraction. The values that will normally always remain unused are the diagonal ones. These may be useful in a future extension of the extraction methodology, but at present they can be safely ignored.

5.2.7 Importing the TSV layout

In order to import the TSV layout, the following command should be run:

```bash
cldm tsv.ldm
```

Here, `tsv.ldm` should be replaced by the name and location of the TSV layout.

5.2.8 Importing the device model

In order to import the device model, the following command should be run:

```bash
putdevmod tsv.dev
```

Here, `tsv.dev` should be replaced by the name and location of the device model file. If the model is imported correctly, the cell can then be set to device status using the `xcontrol` utility:

```bash
xcontrol tsv -device -freemasks:tsub,mtop,tsvl,sub
```

Here, `tsv` should be replaced by the name of the TSV cell and `tsub`, `mtop`, `tsvl` and `sub` should be replaced by the names of the masks that were specified in the input file of the layout and model generator.
5.3 Integral simulation

5.3.1 Introduction

A designer wanting to simulate a three-dimensional IC will generally need to simulate (parts of) the different dies together. These dies are separate designs, which means they can only be extracted separately. In fact, they may even be manufactured in different technologies. Therefore, it should be possible to combine the circuits resulting from the extraction by Space to form a circuit modeling the entire 3D IC. To make this easier, an integral simulation circuit generator was written.

5.3.2 Input file format

3D ICs are described using an ASCII-based file format. XML was not considered necessary here, because the file format needs to store less information. Although XML would still have been an option, the simplest working solution was chosen.

The first line of the file contains the number of dies that follows. Each die is described by a separate line consisting of ten fields separated by whitespace. These fields are:

1. The name of the top-level cell describing the die
2. The project directory containing the cell
3. The name of the mask that other cells can connect to at the bottom of this cell, or a single ! character if there is no such mask
4. The name of the mask that other cells can connect to at the top of this cell, or a single ! character if there is no such mask
5. The x-coordinate of the cell’s origin in nanometers
6. The y-coordinate of the cell’s origin in nanometers
7. 1 if the cell should be flipped in the x-direction, 0 otherwise
8. 1 if the cell should be flipped in the y-direction, 0 otherwise
9. 1 if the cell should be rotated 90 degrees, 0 otherwise
10. The layer the cell is located in as a natural number — a higher number means a higher layer and the bottom layer is numbered 0

5.3.3 Output file format

The SLS file format was chosen as output file format, because these files can easily be imported into the Nelsis database, allowing the circuit to be exported in various formats, including Spice.

The xsls command is used to generate the SLS descriptions of the various cells in the design. Each unique cell is only included once, irrespective of the number of instances. After all
the individual cell descriptions have been generated, the integral simulation circuit generator
adds the description for the complete design. All top-level cells are referenced, given their
own signals for each port and connected with 0 Ω resistors. This yields a circuit description
that covers the highest two hierarchical levels of the IC. Subcells of the individual dies are
not included.

network component1 (terminal signal1, signal2)
{
    res 1 (signal1, signal2);
}
network component2 (terminal signal1, signal2)
{
    cap 1 (signal1, signal2);
}
network ic (terminal S0_signal1, S0_signal2, S1_signal1, S1_signal2)
{
    component1 (S0_signal1, S0_signal2);
    component2 (S1_signal1, S1_signal2);
    res 0 (S0_signal1, S1_signal2);
    res 0 (S1_signal1, S0_signal2);
}

The first eight lines are generated by xsls and describe the cells in the design. The ninth
line defines the network that will be used to connect them together, including a terminal for
every terminal of every cell included. The eleventh and twelveth line connect these cells to
their respective terminals. The thirteenth and fourteenth line connect the terminals to each
other.

An alternative version of the integral simulation circuit generator uses VHDL as its output
format. This version does not use the xsls command (or the xvhdl command), but generates
its own component descriptions. An additional difficulty is that the signal assignment operator
is unidirectional in VHDL. Because the circuit generator does not know the directionality of
the signals in use, it has to make bidirectional connections. This is implemented by converting
signals of the nine-valued std_logic type to the range X01Z and then converting them to their
weak equivalents, the range WLHZ, before assignment. This guarantees that output ports can
override any signal they are connected to, while input ports can still be assigned a different
value externally. Furthermore, two assignments have to be made for each pair of signals.
Admittedly, this approach is somewhat artificial, but it does function correctly.

The generated VHDL code is also very terse (by VHDL standards) and not easily readable
by humans due to the near absence of whitespace. Generating pretty VHDL code was not
a design goal. The VHDL version of the integral simulation circuit generator should be
considered a prototype, while the finished version generates SLS files. Users that do not need
extraction to VHDL are recommended to use the SLS version of the circuit generator instead.
The code listed below has been beautified before inclusion and only serves to illustrate the
mechanism for bidirectional signal assignment.

library ieee;
use ieee.std_logic_1164.all;

entity ic is
end;
architecture structural of ic is

component component1
  port ( signal1 : inout std_logic;
         signal2 : inout std_logic);
end component;

component component2
  port ( signal1 : inout std_logic;
         signal2 : inout std_logic);
end component;

signal S0_signal1, S0_signal2 : std_logic;
signal S1_signal1, S1_signal2 : std_logic;

begin
  P0 : component1 port map (S0_signal1, S0_signal2);
  P1 : component2 port map (S1_signal1, S1_signal2);

  S0_signal1 <= 'L' when to_x01z(S1_signal2) = '0' else
                 'H' when to_x01z(S1_signal2) = '1' else
                 'W' when to_x01z(S1_signal2) = 'X' else
                 'Z';

  S0_signal2 <= 'L' when to_x01z(S1_signal1) = '0' else
                 'H' when to_x01z(S1_signal1) = '1' else
                 'W' when to_x01z(S1_signal1) = 'X' else
                 'Z';

  S1_signal1 <= 'L' when to_x01z(S0_signal2) = '0' else
                 'H' when to_x01z(S0_signal2) = '1' else
                 'W' when to_x01z(S0_signal2) = 'X' else
                 'Z';

  S1_signal2 <= 'L' when to_x01z(S0_signal1) = '0' else
                 'H' when to_x01z(S0_signal1) = '1' else
                 'W' when to_x01z(S0_signal1) = 'X' else
                 'Z';

end;

This format will not be discussed in as much detail as the SLS variant, but the important thing to note is that the number of signal assignments has doubled, as is required to obtain working bidirectional connections.

An alternative approach would have been to connect multiple components to the same signal. This would have reduced the amount of code generated, especially in the VHDL case. Implementation, however, would have been far more complicated when opting for such a design.
5.3.4 Geometric matching

In order to generate a circuit that connects the subcircuits of the various dies, the integral simulation circuit generator needs to know how many instances of each die are present, and for each such instance:

1. The XY-coordinates
2. The layer (dimensionless Z-coordinate)
3. The size
4. The name and location of the terminals at the bottom, if any
5. The name and location of the terminals at the top, if any
6. The geometric orientation

If this information is available, it can be combined to generate the final circuit. The information that is not specified by the user can be gathered from the Nelsis database.

The integral simulation circuit generator was written such that it only tries to connect adjacent layers. Each of the dies in one layer is checked for terminals that connect to the other layer. This is done by selecting the mask that specifies the terminals that face the other die. In order to determine which side is up, the circuit generator inspects the orientation of the die. If it is flipped in either the X-direction or the Y-direction, but not both, it is considered to be upside-down. If it is flipped in neither direction or both, a normal orientation is assumed. If both dies specify a mask with terminals, their coordinates are compared to see if there is a match. This is done in a somewhat naive way by comparing each terminal to every terminal on the other die, leading to a complexity of $O(n^2)$ (with $n$ representing the number of terminals in a die’s mask of interest). If there is a big difference between the number of terminals in the two dies, a better approximation would be $O(nm)$. This complexity is higher than strictly necessary. However, modern desktop PCs are fast enough to compute the connections between two dies in under a minute, as long as the number of terminals is only several tens of thousands.

Depending on whether the output file format is SLS or VHDL, cells are either matched with those cells in the opposite layer they have not yet been matched with, or with all cells in the opposite layer. The former approach means that each connection between cells will only be made once, making it suitable for SLS. The latter approach means that each connection between cells will be made twice, albeit in opposite directions. This makes it suitable for VHDL, but also takes approximately twice as much time. Rewriting the integral simulation circuit generator to use the same approach for both file formats would be feasible and it would only change the order in which the connections are generated. Keeping its status as a prototype in mind, the VHDL version of the circuit generator was kept in its original, simpler form.
5.3.5 Usage of the integral simulation circuit generator

The integral simulation circuit generator is very simple in use: it reads its input from stdin and writes its output to stdout. There are no command line options. Depending on which version is used, the output is in SLS or VHDL format. Only output in SLS format can be imported back into the Nelsis database.

5.3.6 Importing the combined circuit

In order to import the circuit into the Nelsis database of the top-level project, the following command should be run:

csls ic.sls

Here, ic.sls should be replaced by the name and location of the circuit generated by the integral simulation circuit generator.
Verification

6.1 Spice setup

The first step in verifying the model that this thesis is based on is to run simulations in Spice. The reason for this is twofold. First of all, because the model used for extraction is based on the one proposed by Kim et al. [3], the extraction will be compared to the results that would have been obtained if the original model had been used without change. Furthermore, this Spice simulation can be performed with a realistic CMOS gate model on the input of the TSV pair. The results of this Spice simulation will then be used to determine a simpler linear source model for use in a full-wave field solver.

In order to simulate a TSV, its parameters have to be chosen. Kim et al. [3] performed simulations with Ansoft HFSS and built a physical prototype to verify their model. Both the parameters of the TSV with the smallest $d_{TSV}$ they simulated and those of their physical prototype were used for verification purposes. The parameters of the TSV that Kim et al. only simulated, which will be called TSV 0, can be found in Table 6.1. The resulting component values can be found in Table 6.2. The parameters of the physical prototype, which will be called TSV 1 and which will also be used to show the application of the methodology described in this thesis to an example design, can be found in Table 6.3. The resulting component values can be found in Table 6.4.

The input source model will be based on the Nangate Open Cell Library (www.nangate.com). The supply voltage is 1.1 V. In order to obtain a realistic input signal shape, a seven-stage ring oscillator was built out of INV_X1 cells. Simulations with INV_X32 cells showed no significant differences. Based on this simulation, a piecewise linear input signal was defined. The values that were used can be found in Table 6.5.

The time from the falling edge to the rising edge was calculated by subtracting the time on which the output voltage crossed 50 % (0.55 V) in the downward direction from the time on which the output voltage next crossed 50 % in the upward direction. The time from the rising edge to the falling edge was calculated likewise. The rise time was calculated by subtracting the time on which the output voltage crossed 10 % (0.11 V) in the upward direction from the time on which the output voltage next crossed 90 % (0.99V) in the upward direction and dividing the result by 0.8 (($0.99 \text{ V} - 0.11 \text{ V}) / 1.1 \text{ V}$) in order to convert it to the full voltage range. The fall time was calculated likewise. The comparatively high rise time can be explained by the $W/L$ of the PMOS transistors in the library, which is only about 1.5 times that of the NMOS transistors in the library. A higher ratio would have decreased the rise time of the library cells. The simulation results of both the ring oscillator and its piecewise linear approximation can be found in Figure 6.1.

Several different drivers were used during the simulations. For TSV 0, the smallest driver that
was simulated was a single INV.X32 cell from the Nangate Open Cell Library, which in turn was driven by a single INV.X8 cell driven by the signal described in Table 6.5. The strength of the driver was gradually increased during the subsequent simulations. Each driver was twice as big as the one used in the previous simulation. The largest driver in this series consisted
Parameter | Value  
---|---  
TSV diameter | $d_{\text{TSV}} = 32 \, \mu\text{m}$  
TSV height | $h_{\text{TSV}} = 117 \, \mu\text{m}$  
TSV-to-TSV pitch | $p_{\text{TSV}} = 220 \, \mu\text{m}$  
Oxide thickness | $t_{\text{ox}} = 0.52 \, \mu\text{m}$  
Bottom oxide thickness | $t_{\text{ox,bot}} = 0.5 \, \mu\text{m}$  
IMD height | $h_{\text{IMD}} = 6.5 \, \mu\text{m}$  
Bump height | $h_{\text{Bump}} = 15 \, \mu\text{m}$  
Bump diameter | $d_{\text{Bump}} = 70 \, \mu\text{m}$  
Bump resistivity | $\rho_{\text{Bump}} = 1.68 \times 10^{-8} \, \Omega\text{m}$  
TSV resistivity | $\rho_{\text{TSV}} = 1.68 \times 10^{-8} \, \Omega\text{m}$  
Substrate conductivity | $\sigma_{\text{Si}} = 10 \, \text{S/m}$  
Substrate permittivity | $\epsilon_{\text{Si}} = 11.9 \epsilon_0$  
Oxide permittivity | $\epsilon_{\text{ox}} = 4 \epsilon_0$  
Bottom oxide permittivity | $\epsilon_{\text{ox,bot}} = 4 \epsilon_0$  
IMD permittivity | $\epsilon_{\text{IMD}} = 4 \epsilon_0$  
Underfill permittivity | $\epsilon_{\text{Underfill}} = 7 \epsilon_0$  

Table 6.3: Parameters of TSV 1

| Component | Value  
---|---  
$R_{\text{TSV}} + 2R_{\text{Bump}}$ | 2.574985 mΩ  
$C_{\text{Insulator}} + C_{\text{Bump1}}$ | 594.5447 fF  
$C_{\text{Insulator}} + C_{\text{Bump2}}$ | 398.9738 fF  
$C_{\text{Underfill}} + C_{\text{Bottom}}$ | 1.63318 fF  
$C_{\text{Underfill}} + C_{\text{IMD}}$ | 1.8884 fF  
$C_{\text{Si sub}}$ | 13.9203 fF  
$R_{\text{Si sub}}$ | 756.914 Ω  

Table 6.4: Component values of TSV 1

of 512 INV_X32 cells driven by 128 INV_X32 cells. As a special limiting case, these 512 INV_X32 cells were also simulated while they were driven by an ideal voltage source. This voltage source’s output signal differed from that described in Table 6.5 in that its output signal was inverted and its rise and fall time were 1 ps, but it was otherwise identical. The
<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total period</td>
<td>195 ps</td>
</tr>
<tr>
<td>Falling edge to rising edge</td>
<td>101 ps</td>
</tr>
<tr>
<td>Rising edge to falling edge</td>
<td>94 ps</td>
</tr>
<tr>
<td>Rise time</td>
<td>47 ps</td>
</tr>
<tr>
<td>Fall time</td>
<td>19 ps</td>
</tr>
</tbody>
</table>

Table 6.5: Input source parameters

![Graph showing simulation results of a ring oscillator and its piecewise linear approximation](image)

Figure 6.1: Simulation results of a ring oscillator and its piecewise linear approximation

drivers that were used with TSV 1 were four times as big as the ones that were used with TSV 0, so the smallest driver consisted of 4 INV\_X32 cells driven by a single INV\_X32 cell and the largest regular driver consisted of 2048 INV\_X32 cells driven by 512 INV\_X32 cells. The limiting case driver consisted of 2048 INV\_X32 cells driven by the same ideal voltage source as the one driving the limiting case driver for TSV 0.

The load was modeled as a 20 fF capacitance. This value was chosen because the input capacitance of a typical modern CMOS gate is in the order of magnitude of 1 fF. Interconnect
can add more effective capacitance, but given a reasonable distance between the TSV and the input of the gate it is connected to, input capacitances of more than 20 fF are unlikely. This means that the capacitance of the TSV pair will likely dominate the combined behavior. If the TSV model is simplified by setting $R_{TSV}$, $R_{Bump}$ and $R_{Si\ sub}$ all to 0, an equivalent capacitance of 119 fF for TSV 0 and 500 fF for TSV 1 results. If $R_{Si\ sub}$ is set to infinity instead (or, equivalently, setting $G_{Si\ sub}$ to 0), an equivalent capacitance of 35 fF for TSV 0 and 17 fF for TSV 1 results. This means that the behavior of the TSV pairs, to the degree that it is capacitive, will be comparable to that of a capacitance with a value between 35 fF and 119 fF for TSV 0 and between 17 fF and 500 fF for TSV 1. For lower frequencies, the value will be closer to the higher value of the two and for higher frequencies it will be closer to the lower value.

Three different models are used for each of the two pairs of TSVs:

1. An ideal TSV pair with no parasitics
2. The result of extraction of the TSV pair with Space
3. The hypothetical result of extraction of the TSV pair with Space if the original model by Kim et al. [3] had been used

The ideal TSV pair represents the best attainable result. Being ideal, this model is identical for TSV 0 and TSV 1. Physical implementations will always perform worse than this.

The result of extraction is the simulation result of interest, because this is the model that needs to be verified. It’s okay if it deviates from the results of the ideal TSV. In fact, it should deviate — if it does not, it probably means that some parasitics were modeled incorrectly.

The model is different for TSV 0 and TSV 1, as can be seen from Table 6.2 and Table 6.4.

The model that was proposed by Kim et al. [3] forms the basis of the model that was implemented in this thesis, so a comparison of its result to that of the extracted circuit should show the impact of the simplifications that were made. Given the relatively high output impedance of the source (compared to the TSV’s resistance), it’s likely that the skin effect won’t result in a big difference in this simulation. The minor improvements that were made to the model are probably also negligible. This means that the presence of an inductance in this TSV model is what will cause the major differences, if any. Like the result of extraction, this model is different for TSV 0 and TSV 1.

### 6.2 Spice results

In this section, the ideal model will be referred to as “Ideal”, the simplified model that was implemented in Space for this thesis will be referred to as “Space” and the original model proposed by Kim et al. [3] will be referred to as “Kim”. Five different signals will be shown for all simulations that are included here. For all three models, the signal at the far end of the TSV pair is shown. Additionally, both the signal at the near end of the TSV pair and the input signal of the inverters that drive the TSVs are shown for Kim. For Ideal, the signal at the near end of the TSV pair is identical to the signal at the far end by definition. The input signal of the inverters that drive the TSVs is not considered to be of sufficient interest for inclusion. For Space, the signal at the near end of the TSV pair is almost identical to
the signal at the far end and would not be distinguishable as a separate line. The input signal of the inverters that drive the TSVs is almost identical between Space and Kim for all simulations that were performed.

When TSV 0 is used with an underdimensioned driver, Space yields simulation results that are very close to those of Kim, as can be seen in Figure 6.2. In fact, the difference between Space and Kim is so small that the two different lines can no longer be distinguished in this graph. This difference becomes somewhat bigger when the size of the driver is increased, but even with a very large driver the differences are small, as can be seen in Figure 6.3. Kim shows some ringing at the far end of the TSV pair that does not show up with Space, but the differences are minor. Using an even larger driver doesn’t give a noticeable improvement of rise and fall times, so the use of an even larger driver would be wasteful and therefore unrealistic. However, to show the limiting case of driver strength for TSV 0, a simulation was performed in which 512 INV_X32 cells were driven directly by an ideal voltage source with a rise and fall time of 1 ps. The results, which can be seen in Figure 6.4, clearly show severe ringing at the far end of Kim. This ringing is far less pronounced at the near end of Kim, but for Space it is completely absent. Furthermore, the Miller effect can be seen in the form of voltage peaks just before a rising or falling edge, caused by the steep edges at the input. These peaks are present at the near end of all three models, but they do not show up at the far end of Kim. The presence of the inductances prevents the sudden change in voltage from reaching the load capacitance.

From this it can be concluded that, for TSV 0, Space forms a good approximation of Kim as long as a realistic driver is used. If the driver strength is increased significantly beyond what is required to operate the TSVs at 5 GHz, Kim predicts that ringing will occur, which is not the case with Space.

When TSV 1 is used with an underdimensioned driver, the results are close to those of TSV 0. The major difference is that the driver needs to be about four times as large. This is consistent with the ratio between the equivalent capacitances of the TSV pairs when all resistances are neglected. When the driver strength is increased, Kim shows ringing at the far end, which is absent for Space. This first becomes apparent when driving the TSV pair with 16 INV_X32 cells driven by 4 INV_X32 cells, as can be seen in Figure 6.5. At this point, it would seem that the difference would typically still be considered acceptable. The model loses its validity when the TSV pair is driven by 64 INV_X32 cells driven by 16 INV_X32 cells, as can be seen in Figure 6.6. The difference becomes more severe as the driver strength is increased further, as can be seen in Figure 6.7. Here, the TSV pair is driven by 2048 INV_X32 cells that are driven by 512 INV_X32 cells. While such a driver would be very expensive in terms of chip area, it is not unreasonable for a designer to at least consider such a large driver. The simulation results of Ideal and Space can no longer be distinguished as separate lines, along with the near end of Kim. The far end of Kim, however, differs significantly, showing severe ringing. Simulations in which the 2048 INV_X32 cells are driven directly by an ideal voltage source show even more pronounced ringing, more than what was seen with TSV 0.

From this it can be concluded that, for TSV 1, Space only forms a good approximation of Kim as long as small drivers are used. When the TSV pair is driven by 64 INV_X32 cells driven by 16 INV_X32 cells, or an even larger driver, the model loses it validity. Kim predicts ringing that becomes more severe as the driver becomes stronger, but this effect is completely absent with Space.
Figure 6.2: Simulation results of 1 INV_X8 cell ⇒ 1 INV_X32 cell ⇒ pair of TSV 0

### 6.3 3D model setup

Having performed the simulations in Spice, the next step in the verification process is to perform a simulation using a field solver. TSV 1 will be used for this purpose. Before the actual simulation can start, a 3D model of the pair of TSVs has to be constructed. That procedure will be described in this section.

The pair of TSVs shown in Figure 2.2 can best be described by means of a series of subtractions of elementary geometric shapes. In order to specify the configuration in a manner that is both exact and understandable, its components will be specified in order of precedence. This means that when a shape is defined, all shapes that are specified before it are then subtracted from it. In order to make this subtraction work, the order is chosen as follows:

1. The conductors
2. The insulating layers
3. The substrate

The insulating layers are specified before the conductors, mainly because the actual TSV is surrounded by a thin oxide layer. Specifying the oxide after the conductors allows it to be
defined as a simple cylinder, just like the TSV itself. The substrate is specified last, so it is present wherever no other material is. At the boundaries of the structure there is an infinite vacuum.

Because the bumps and underfills are separated from the actual TSV in the Z-dimension, they can be specified separately. The bottom bumps and underfill are specified first, then the actual TSV and its surroundings, and finally the top bumps and underfill. This way, the order of the specifications mimics the vertical structure of the TSV.

Above and under the structure, two volumes of vacuum are added in order to separate the ports from the boundary of the structure.

The materials are mostly the obvious choices, but the underfill forms a notable exception. Although Kim et al. [3] do not specify the material it is made of, their paper does specify an $\epsilon_r$ of 7. This means that it can’t be silicon dioxide. A possible candidate is silicon nitride and that is the material that will be assumed here.

The exact shapes required to build a 3D model can be found in Appendix B.
In order to be able to simulate the TSV in a field solver, a linear source model has to be obtained. If a voltage source in series with a resistance of 25 Ω is used, the slope will be close to that of 8 INV_X32 cells driving 32 INV_X32 cells, which was the maximum driver size for which the difference between the original model and the simplified version would typically still be deemed acceptable. It also corresponds to the highest resistance for which the slope at the near end of the TSV still matches the slope of the ideal input voltage near 50% (0.55 V) for each of the three models. By also slightly delaying the input signal, the match between the results of the two source models can be made even better. Matching the CMOS gate exactly is impossible, but the approximation is reasonable for both models. Furthermore, many different CMOS gates could be used to drive the pair of TSVs, so there is no single correct source model. In contrast to the other two TSV models, the ideal TSV model will just show a replica of the input signal because of its significantly smaller capacitances, so those results are no longer useful as a means of comparison.

Both source and load are modeled as lumped components. For reasons given above, the source is a voltage source with a resistance of 25 Ω and the load is a capacitance with a value of 20 fF. Because CST Microwave Studio focuses on S-parameter calculations, the source is given an amplitude that would cause a peak power of 1 W to be dissipated in a perfectly matched
load. This means that its open circuit peak-to-peak amplitude is $\sqrt{100} \text{V} = 10 \text{V}$. The source is connected to the centers of the two bottom bumps and the load is connected to the centers of the two top bumps.

6.4 Field solver results

Initially, Ansoft HFSS was chosen as the field solver to be used for verification purposes, because of its availability. This did not yield useful results.

A second attempt was performed with CST Microwave Studio. A significant leakage current was found to be flowing inside the substrate, but its exact origin could not be determined.

Time constraints meant that no further attempts could be performed.

6.5 Conclusions on the verification of the model

Kim et al. [3] used a field solver to validate their model and those simulations showed a good match. Therefore, the good match for relatively small TSVs driven by realistic sources
between that model and the one that was implemented for this thesis is convincing enough that the results that were obtained are correct under those circumstances. For drivers larger than 512 INV_X32 cells driven by 128 INV_X32 cells, the simplified model may no longer be valid, although such drivers would be uselessly large. Kim et al. also performed an experimental verification, although the physical prototype also included an RDL, so it was essentially a joint verification. This physical prototype contains a big TSV, with its diameter of 32 μm. In this case, the match between the original model and the simplified version is not as good. The simplified model is only a good approximation as long as the driver does not become too large. With drivers as large as 64 INV_X32 cells driven by 16 INV_X32 cells or even larger drivers, ringing will occur, which the simplified model does not show because it doesn’t include inductances.

Designers can use the simplified model to check if their driver is underdimensioned. Overdimensioning the driver can lead to ringing, which can’t be determined by using the simplified model. As long as ringing is absent and the model proposed by Kim et al. [3] is valid for the given TSV configuration, the simplified model that was implemented in Space for this thesis can be used in circuit simulations of TSVs and will yield correct results.

One final important aspect that should be mentioned is that of the different kinds of TSVs in existence. The extraction process was shown to function correctly for cylindrical via-last
TSVs, but not for other kinds of TSVs.

Since via-last TSVs are the most complicated to model correctly, altering the model for use with via-middle or via-first TSVs is easier than altering it the other way around. The parasitics at the bottom of the TSV would remain unchanged, but at the top there would not be such a large bump, so $C_{Bump1}$ and one instance of $R_{Bump}$ (per TSV) and $C_{Underfill}$ (per pair of TSVs) should be removed for via-first extraction. Via-middle TSVs will still exhibit part of these parasitics. Performing these adjustments was not attempted because of time constraints.

Another kind of TSV that was not covered is the cuboid-shaped TSV. Two approaches could be attempted: applying another set of correction parameters on top of those that are used to convert the rectangular layout to circular or altering the model to take the cuboid shape into account. The latter approach would probably be better, but was not pursued. It would have been very time-consuming to verify a model that differs significantly from any other model available [5], while recent examples of TSVs in literature are overwhelmingly of the cylindrical kind.

Figure 6.7: Simulation results of 512 INV_X32 cells ⇒ 2048 INV_X32 cells ⇒ pair of TSV 1...
7.1 Example design

This chapter will apply the extraction methodology described by this thesis to an IC design that serves as an example of some of the possible applications. The chosen application is a simple ring oscillator.

The design that will be used as an example consists of two dies. These dies contain far less components than a normal die would. They also connect to the outside world using TSVs only. This way, the complexity is kept within bounds. The dies are stacked on top of each other in a front-to-back fashion. This means that the TSVs of the upper die will connect to the highest metal layer of the lower die. The TSVs are constructed in a via-last manner. The connections between the IC and the outside world are made through the TSVs in the lower die. Both power and signals have to be carried by TSVs. The lower die is extracted using a formula-based extraction methodology. The TSVs of the upper die are extracted using a model-based extraction methodology. The dies have different λs and are based on different processes, both of which are modified versions of the scmos.n process that is provided as part of Space. Finally, the upper die will be rotated 270 degrees relative to the lower die. This combination ensures that most features of the utilities are demonstrated in practice.

7.1.1 Lower die

The lower die consists of two large TSVs (Table 7.1), four additional connections to the upper die and three inverters. The two large TSVs are used for power. The four additional connections are used for signal. All connections are made in pairs, so the two signals (an input and an output) each have their own ground connection. The inverters are connected between the input and the output. They are extremely large (having a channel length of 4 μm), but still tiny compared to the TSVs.

The entire design consists of a single cell and will be extracted using formula-based extraction.

7.1.2 Upper die

The upper die consists of six TSVs and two inverters. Four of these TSVs are of type TSV 1 (Table 6.3) and the other two are of type TSV 2 (Table 7.1). The instances of TSV 1 connect to the additional connections on the lower die. The instances of TSV 2, being larger, connect to the TSVs on the lower die. Like on the lower die, the larger TSVs are used for power and the relatively smaller TSVs for signals. Having the signal assignments of the two dies match is a necessary condition for a working design, of course. One thing that should be kept in
mind is that this die’s inputs are the other die’s outputs and vice versa. The transistors have
the same size as on the lower die, so they are huge as well.

This design consists of four cells: two cells for the two different TSVs, a cell for the inverter
and a top-level cell. Model-based extraction will be performed on this die, so that both
approaches are used and the ability of model-based extraction to deal with different TSV
dimensions in a single design is demonstrated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSV diameter</td>
<td>$d_{TSV} = 64 \mu m$</td>
</tr>
<tr>
<td>TSV height</td>
<td>$h_{TSV} = 117 \mu m$</td>
</tr>
<tr>
<td>TSV-to-TSV pitch</td>
<td>$p_{TSV} = 440 \mu m$</td>
</tr>
<tr>
<td>Oxide thickness</td>
<td>$t_{ox} = 0.52 \mu m$</td>
</tr>
<tr>
<td>Bottom oxide thickness</td>
<td>$t_{ox,bot} = 0.5 \mu m$</td>
</tr>
<tr>
<td>IMD height</td>
<td>$h_{IMD} = 6.5 \mu m$</td>
</tr>
<tr>
<td>Bump height</td>
<td>$h_{Bump} = 15 \mu m$</td>
</tr>
<tr>
<td>Bump diameter</td>
<td>$d_{Bump} = 140 \mu m$</td>
</tr>
<tr>
<td>Bump resistivity</td>
<td>$\rho_{Bump} = 1.68 \cdot 10^{-8} \Omega m$</td>
</tr>
<tr>
<td>TSV resistivity</td>
<td>$\rho_{TSV} = 1.68 \cdot 10^{-8} \Omega m$</td>
</tr>
<tr>
<td>Substrate conductivity</td>
<td>$\sigma_{Si} = 10 \text{ S/m}$</td>
</tr>
<tr>
<td>Substrate permittivity</td>
<td>$\epsilon_{Si} = 11.9 \epsilon_0$</td>
</tr>
<tr>
<td>Oxide permittivity</td>
<td>$\epsilon_{ox} = 4 \epsilon_0$</td>
</tr>
<tr>
<td>Bottom oxide permittivity</td>
<td>$\epsilon_{ox,bot} = 4 \epsilon_0$</td>
</tr>
<tr>
<td>IMD permittivity</td>
<td>$\epsilon_{IMD} = 4 \epsilon_0$</td>
</tr>
<tr>
<td>Underfill permittivity</td>
<td>$\epsilon_{Underfill} = 7 \epsilon_0$</td>
</tr>
</tbody>
</table>

Table 7.1: Parameters of TSV 2

7.2 Top-level process directory

The process directory for the top-level project can be relatively simple. Only a dummy
maskdata file and a copy of the spice3f3.lib file of scmos.n are required. The contents of
the maskdata file should be:

"pseudo" "Pseudo-process for integral simulation"
bbox 2 0 0 0 0 0 0 0 0
7.3 Top-level project directory

The top-level project directory should be created as follows:

```
mkpr ic -p pseudo -l 1
```

Here, `ic` is the name of the project, `pseudo` is the name of the process and 1 is the value of \( \lambda \).

After creating the top-level project, the input file for the integral simulation circuit generator can be written. Its contents should be:

```plaintext
2
lower formula ! mtop 0 0 0 0 0 0 0
upper model tsub ! 0 58e4 1 1 1 1 1
```

The command file for `simeye` can also be created now:

```plaintext
option simperiod = 10
option sigunit = 1e-8
option outacc = 10p

option level = 3
plot S0_out_gnd, S0_out_sig, S1_out_gnd, S1_out_sig
print S0_out_gnd, S0_out_sig, S1_out_gnd, S1_out_sig
set S0_ground = l~
set S0_power = h~

/* spice commands:
 *
trise 0.1n
tfall 0.1n
tstep 0.02n
*/
.options nomod
.ic v("S0_out_sig") 0v
.ic v("S1_out_sig") 5v
.options cptime=30
/*
*/
```

A suitable input file for the layout and model generator, `tsv.xml`, can be found in Appendix A.

Running the layout and model generator with the appropriate scaling factors (which, for `scmos_n`, are \( 10^{-12} \), \( 10^{-6} \) and \( 10^{-12} \), respectively) yields the following values:

- Cell `tsv1`:
  - \( R_{TSV,2R,Bump} \): 2.636785e+00
  - \( C_{Insulator} \): 2.989658e+03
  - \( C_{Bump1} \): 5.46164e+01
  - \( C_{Bump2} \): 4.204741e+00
  - \( C_{Underfill} \): 2.302732e+01
  - \( C_{Underfill_Diagonal} \): 1.921055e+01
  - \( C_{Bottom} \): 6.646446e-01

55
C_Bottom_Diagonal : 5.863544e-01
C_IMD : 8.640380e+00
C_IMD_Diagonal : 7.622607e+00
R_Silicon : 7.569140e+02
R_Silicon_Diagonal : 8.579775e+02
RC_Silicon : 1.053648e-11

Cell tsv2:
R_TSV_2R_Bump : 2.636785e+00
C_Insulator : 2.965944e+03
C_Bump1 : 5.515095e+01
C_Bump2 : 4.242381e+00
C_Underfill : 1.151366e+01
C_Underfill_Diagonal : 9.605276e+00
C_Bottom : 3.323223e-01
C_Bottom_Diagonal : 2.931772e-01
C_IMD : 4.320190e+00
C_IMD_Diagonal : 3.811304e+00
R_Silicon : 7.569140e+02
R_Silicon_Diagonal : 8.579775e+02
RC_Silicon : 1.053648e-11

Of course, the necessary .ldm and .dev files are generated as well. In order, the contents of tsv1.ldm, tsv1.dev, tsv2.ldm and tsv2.dev are (some files were reformatted to fit):

```
ms tsv1
term tsub 0 70 0 70 bottom
box tsvl 19 51 19 51
term mtop 0 70 0 70 top
term sub 0 70 0 70 substrate
me
device tsv1
begin spicemod
* terminals  bottom  top  substrate
..subckt tsv1 1 2 3
RTSV_2RBump 1 2 2.574985e-03
C_Insulator_CBump1 1 3 5.945447e-13
C_Insulator_CBump2 2 3 3.989738e-13
..ends tsv1
end
ms tsv2
term tsub 0 140 0 140 bottom
box tsvl 38 102 38 102
term mtop 0 140 0 140 top
term sub 0 140 0 140 substrate
me
device tsv2
begin spicemod
* terminals  bottom  top  substrate
..subckt tsv2 1 2 3
RTSV_2RBump 1 2 6.437462e-04
C_Insulator_CBump1 1 3 1.614342e-12
C_Insulator_CBump2 2 3 8.250555e-13
..ends tsv2
end
```
For now, this completes the necessary work to be performed in the top-level directory. The process directories for the two dies can now be created. They are called formula for the lower die and model for the upper die.

### 7.4 Process directory for the lower die

The information obtained from the layout and model generator can be used to complete the file `space.def.p` which sets some general extraction parameters. The following lines should be added to the `space.def.p` file that came with `scmos.n`, removing the original `min_res` and `lat_cap_window` lines:

```
add_sub_caps 1
sub_rc_const 1.053648e-11
low_contact_res 0
low_sheet_res 0
min_res 0
min_sep_res 0
lat_cap_window 376
```

The first line specifies that substrate capacitances should also be extracted and the second line specifies the required time constant, obtained from the layout and model generator. The next four lines tell Space not to remove any resistances from the circuit because of their low value. The last line ensures that lateral capacitances are actually extracted. It corresponds to the maximum distance between the edges of two TSVs in μm.

The following lines should be appended to the `maskdata` file that came with `scmos.n` to define the extra masks (reformatted to fit):

```
tsub 1 13 1 3 1 3 1 3 0 "through silicon"
mtop 1 14 1 4 1 6 1 8 0 "top metal"
cvt 0 15 1 6 1 7 1 6 0 "contact metal2 to top metal"
tsvl 0 16 1 0 1 2 2 4 0 "through silicon via"
```

The `space.def.s` file requires the most changes compared to its `scmos.n` version. The first change is to replace the `maxkeys` line with the following:

```
maxkeys 17
```

```
new: tsub tsvl : tsvm
new: mtop tsvl : tsvn
```

This way, enough keys are made available and the necessary pseudo-masks are created. Next, the new masks should be added to the `colors` section:

```
tsub yellow
mtop cyan
tsvl green
tsvm green
tsvn green
```

They also need to be specified in the `conductors` section:
The value of \texttt{mtop} was chosen to be realistic compared to the other two metal layers. The next addition should be made to the \texttt{contacts} section:

\begin{verbatim}
cont_l : tsvm tsvn : tsvm tsvn : 2.636785
cont_m : tsvm : tsub tsvm : 0
cont_n : tsvn : mtop tsvn : 0
cont_o : mtop cms cvt : mtop cms : 1
\end{verbatim}

Vias between \texttt{mtop} and \texttt{cms} (the second metal layer) have the same contact resistance as vias between \texttt{cmf} (the first metal layer) and \texttt{cms}. The \texttt{capacitances} section undergoes the biggest change (reformatted to fit):

\begin{verbatim}
C_Underfill1 : ! tsub -tsub =tsub :-tsub = tsub: 300e-6 1.151366e+01
C_Bottom : ! tsvm -tsvm =tsvm :-tsvm = tsvm: 376e-6 3.323223e-01
C_Bump1 : tsub !tsvm mtop : tsub @sub : 5.515095e+01
C_Insulator1 : ! tsvm -tsvm :-tsvm @sub : 2.965944e+03
C_Insulator2 : ! tsvn -tsvn :-tsvn @sub : 2.965944e+03
C_Bump2 : mtop !tsvn tsub : mtop @sub : 4.242381e+00
C_IMD : !tsvn -tsvn =tsvn :-tsvn =tsvn: 376e-6 4.320190e+00
C_Underfill2 : ! mtop -mtop =mtop :-mtop = mtop: 300e-6 1.151366e+01
\end{verbatim}

It should be noted that an extra condition was added to the specification of \texttt{C\_Bump1} and \texttt{C\_Bump2}. This was done to prevent the presence of interconnects in the \texttt{mtop} layer from generating a bigger substrate contact. That way, the substrate capacitances and conductances computed by Space are as expected. The substrate resistance has to be specified in the \texttt{selfsubres} section as follows:

\begin{verbatim}
15504 816 756.914 0
\end{verbatim}

This specifies the layout area of the bump minus that of the TSV, the combined length of their perimeters and the resulting substrate resistance. It also tells Space that the conductance to the substrate node may be replaced entirely by a coupling conductance. The \texttt{coupsubres} section requires a similar addition, including the area twice (once for each bump), omitting the perimeter and including the distance between the centers of two TSVs, while telling Space not to leave any conductance to the substrate node:

\begin{verbatim}
15504 15504 300 756.914 1
\end{verbatim}

Finally, \texttt{tecc} should be run to generate the \texttt{space.def.p} file.

### 7.5 Project directory for the lower die

The lower die will only contain instances of TSV 2, not TSV 1. It does have to connect to instances of TSV 1, but this does not require a layout that is accurate down to the micron.
Therefore, \( \lambda \) will be set to its highest feasible value: 2 \( \mu \)m. This means that the project directory should be created as follows:

\[
\text{mkpr ic/formula -p formula -l 2}
\]

Here, ic is the name of the top-level project, formula is the name of this project, the second occurrence of formula is the name of the corresponding process and 2 is the value of \( \lambda \) in \( \mu \)m.

The file lower.1dm, which is not included in this thesis because of its length and lack of easily human-readable information, describes the layout of the lower die. It can be imported by running the cldm command:

\[
cldm lower.1dm
\]

After importing the layout, the circuit can be extracted by Space:

\[
space -b -l -z lower
\]

When running Space 5.4.4, warnings about node joins will show up. They can be ignored because they refer to nodes that shouldn’t be joined. After performing the extraction the work in this directory is done.

### 7.6 Process directory for the upper die

The space.def.p file that was created in the project directory for the lower die can be used in this directory without change. The maskdata file requires only one extra line:

\[
\text{sub 1 17 1 0 1 0 1 0 1 "substrate"}
\]

The space.def.s file requires slightly more changes. First of all, maxkeys should be changed to 18 and the substrate mask should be added to the colors section:

\[
\text{sub pink}
\]

An extra line should also be added to the conductors section:

\[
\text{cond sb : sub : sub : 0 : m}
\]

In the contacts section, the cont line should be removed and another one added:

\[
\text{cont u : sub !tsvl : @sub sub : 0}
\]

In the capacitances section, the lines that were added should be replaced by this new description of distance-value pairs (reformatted to fit):

\[
\begin{align*}
\text{C_Underfill1 : !tsub -tsub =tsub :-tsub =tsub:} & \quad 150e-6 \quad 2.302732e+01 \quad 300e-6 \quad 1.151366e+01 \\
\text{C_Bottom : !tsvm -tsvm =tsvm :-tsvm =tsvm:} & \quad 188e-6 \quad 6.646446e-01 \quad 376e-6 \quad 3.323223e-01 \\
\text{C_IMD : !tsvn -tsvn =tsvn :-tsvn =tsvn:} & \quad 188e-6 \quad 8.640380e+00 \quad 376e-6 \quad 4.320190e+00 \\
\text{C_Underfill2 : !mtop -mtop =mtop :-mtop =mtop:} & \quad 150e-6 \quad 2.302732e+01 \quad 300e-6 \quad 1.151366e+01
\end{align*}
\]
One line should be added to the `selfsubres` section between its original contents and the line that was added earlier:

```
3876 408 756.914 0
```

Another line should be added to the `coupsubres` section between its original contents and the line that was added earlier:

```
3876 3876 150 756.914 1
```

Afterwards, `tecc` should be run again.

### 7.7 Project directory for the upper die

The upper die contains instances of both TSV 1 and TSV 2. TSV 1 requires a value of \( \lambda \) that is no higher than 1 \( \mu \)m, so this project directory should be created as follows:

```
mkpr ic/model -p model -l 1
```

In order to use the device models and layouts of the TSVs, they need to be imported from the top-level project directory. Furthermore, these cells need to be set to device status:

```
cldm ../tsv1.ldm
cldm ../tsv2.ldm
putdevmod ../tsv1.dev
putdevmod ../tsv2.dev
xcontrol tsv1 -device -freemasks:tsub,mstop,tsv1,sub
xcontrol tsv2 -device -freemasks:tsub,mstop,tsv1,sub
```

The upper die is designed in a hierarchical fashion, unlike the lower die, so the two inverters are also instantiated from a separate cell. This means that both `inverter.ldm` and `upper.ldm` (neither file is included in this thesis) need to be imported:

```
cldm inverter.ldm
cldm upper.ldm
```

Warnings about no-origin mode can be ignored. Finally, the last extraction is performed:

```
space -b -l -z upper
```

Afterwards, the complete IC can be simulated from the top-level project directory.

### 7.8 Integral simulation

In order to perform a simulation of the complete IC, the `xspicerc` and `jun.lib` files should be copied from Space’s `suboscil` demo directory to the top-level project directory. Next, the TSVs’ device models should be imported:

```
putdevmod tsv1.dev
putdevmod tsv2.dev
xcontrol tsv1 -device
xcontrol tsv2 -device
```
When this is done the integral simulation circuit generator can be run. The part of its output that is generated by the circuit generator itself (as opposed to xsls) is as follows (reformatted to fit):

```
network ic (terminal S0_ground, S0_power, S0_vss, S0_vdd,
            S0_in_gnd, S0_in_sig, S0_out_sig, S0_out_gnd, S1_vss, S1_vdd,
            S1_out_gnd, S1_out_sig, S1_in_sig, S1_in_gnd)
{
    lower (S0_ground, S0_power, S0_vss, S0_vdd, S0_in_gnd,
            S0_in_sig, S0_out_sig, S0_out_gnd);
    upper (S1_vss, S1_vdd, S1_out_gnd, S1_out_sig,
            S1_in_sig, S1_in_gnd);
    res 0 (S0_vss, S1_vss);
    res 0 (S0_vdd, S1_vdd);
    res 0 (S0_in_gnd, S1_out_gnd);
    res 0 (S0_in_sig, S1_out_sig);
    res 0 (S0_out_gnd, S1_in_gnd);
    res 0 (S0_out_sig, S1_in_sig);
}
```

After this file is imported using cs1s, simeye can be run. It should be set to run Spice on circuit ic, using the command file that was created earlier. If all steps were taken correctly, the screen should show that the IC oscillates with a frequency of just under 30 MHz, as can be seen in Figure 7.1.
Figure 7.1: Simulation results of applying the extraction methodology to a ring oscillator
8.1 Contributions

The contributions made to the state of the art by this thesis can be divided into two categories: modeling and extraction. While extraction of TSVs is the main problem this thesis tries to solve, some interesting conclusions regarding modeling were also reached in the process.

8.1.1 Modeling

Being focused on extraction, this thesis does not propose a new model designed from scratch. Instead, several models described in literature were contrasted and in the end the model proposed by Kim et al. [3] was selected, because it’s both based on actual physics and complete in the sense that it models the parasitics between TSVs as well. The fact that it’s based on physics allows others to verify and improve the model based on their understanding of the physics involved. The fact that it models the parasitics between TSVs means that the most important factor influencing a TSV’s performance, the parasitic capacitance and conductance inside the substrate to which the TSVs are capacitively coupled, is taken into account.

This thesis improves on the work by Kim et al. [3] in two ways. One is by simplifying the model and showing the circumstances under which these simplifications are valid. The simplified model is valid for relatively small TSVs ($d_{TSV} = 10\mu m$) with realistic drivers at the input and for larger TSVs ($d_{TSV} = 32\mu m$) as long the driver at the input is small enough. The other improvement on the model is the correction of a few minor mistakes. The first contribution is the most important, because it allows the model to be used in an environment where inductances and especially the skin effect are difficult or impossible to represent correctly. In fact, this simplification and its validity are what allows TSVs to be extracted correctly by Space without requiring any major changes to the code.

The first item of the problem statement, “Find a good model for TSVs” is thus solved by this thesis.

8.1.2 Extraction

The model that was found and improved in this thesis was not the main objective. Extraction is the ultimate goal. In order to achieve that goal, various approaches were considered and both model-based extraction and formula-based extraction were implemented, both based on the aforementioned model.
Solving the second item of the problem statement, “Determine the relationship between the chip layout and the model’s parameters”, required a two-pronged approach. Many of the model’s parameters had clear relationships with the layout and those were easily found. Some parameters, however, depend on aspects of the design that aren’t part of the layout itself, such as the substrate thickness. This meant that certain aspects of a design had to be specified as part of the definition of the technology. Technically, that means that some of the model’s parameters aren’t actually derived from the layout, but this is not something a designer will worry about. Any differences that might make it impossible to extract all TSVs correctly in a formula-based fashion (for instance, if different TSVs have different values of $t_{ox}$) can be solved by means of model-based extraction.

Solving the third item of the problem statement, “Develop a methodology to fill in the model’s parameters based on the chip layout”, required an extra intermediate step. Because circular shapes can’t be specified in layouts that Space accepts, these circular shapes first have to be translated to rectangular shapes to derive a layout that is suitable for extraction. Then, the relationship between the chip layout and the model’s parameters have to be reformulated in terms of the new layout. This thesis shows the implementation of this approach for both model-based and formula-based extraction. Unfortunately, this translation means that formula-based extraction is limited to a single TSV configuration per die, unless inexact extraction results are considered acceptable. Model-based extraction, as implemented in this thesis, does not have that limitation. The inter-TSV parasitics, which can’t practically be extracted with model-based extraction, can be extracted correctly using formula-based extraction, even with multiple TSV configurations.

Solving the fourth item of the problem statement, “Implement the methodology in Space”, meant writing a utility that can be used to generate both a model and a matching layout to be used for model-based extraction. Formula-based extraction, as described in this thesis, does not require the use of the layout and model generator, although it can still be used in order to avoid manual computations.

Solving the fifth item of the problem statement, “Verify the result using a field solver” was done indirectly. The result of an extraction with Space was simulated in Spice to compare it to the model proposed by Kim et al. [3], which the model was based on. These simulations showed a good match between the two models for relatively small TSVs with realistic drivers at the input and for larger TSVs as long as the driver at the input is small enough. Because Kim et al. [3] verified their model using both a field solver and a physical prototype, the assurance that their model is correct carries over to the model that was implemented for this thesis, as long as the simplifications that were made are valid.

In addition to solving the problem statement, a utility was developed to allow a designer to simulate the different dies that make up a 3D IC integrally, as a single circuit. This increases the usefulness of the extraction of TSVs, as was demonstrated by using the integral simulation circuit generator in an example application of the extraction methodology.
8.2 Future work

During the course of this work, several opportunities for improving on the results presented here came to light. Four of them are important enough to mention here: refining the model, diagonal parasitics, physics-based extraction and circular layout shapes. They will all be covered separately.

8.2.1 Refining the model

Verification simulations that were performed on the model that was implemented for this thesis show that these simplifications aren’t always valid. Specifically, large TSVs with large drivers at the input exhibit ringing, an effect that’s not included in the simplified model because it neglects inductances.

Adding inductances to the model would improve the model’s accuracy. In order to do so with formula-based extraction, changes would need to be made to Space. These changes would not be necessary for model-based extraction, although the inductances in the rest of the circuit would still not be included in that case.

Adding the skin effect to the model would also improve its accuracy, but the way this effect is included by Kim et al. is such that a useful Spice model can’t be based on it. A different model would have to be selected and verified in order to model the skin effect correctly. Doing so would only be worth the effort after inductances have been implemented, as the skin effect is an inductive effect itself. Given that the skin depth at Gigahertz frequencies is in the order of magnitude of microns, the impact on the value of the resistances in the model is significant, but this does not automatically translate into different behavior. As long as the resistance of the TSV remains small compared to the resistance of its driver, the skin effect does not play an important role. Simulation results indicate that the effect of the series inductance is far more significant than the skin effect.

8.2.2 Diagonal parasitics

The layout and model generator that was developed for this thesis can compute values for inter-TSV parasitics that would apply in a diagonal direction. If TSVs are arranged in a rectangular grid, parasitics won’t just manifest themselves in horizontal and vertical directions, but also diagonally. Currently, Space models the substrate in such a way that only half of these diagonal parasitics will appear. If Space’s substrate extraction methodology could be changed to extract all diagonal parasitics correctly, this would allow for better extraction results. Extraction of diagonal capacitances outside the substrate may also require changes.

8.2.3 Physics-based extraction

This thesis describes the implementation of both model-based and formula-based extraction. Physics-based extraction is also discussed, but was not implemented. The extraction results
obtained using this method could potentially be even better than what formula-based extraction allows. However, modeling the TSV inside the substrate will not be easy. Space currently uses a hybrid of the Boundary Element Method and the Finite Element Method for substrate extraction. If a usable 3D model of a TSV can be implemented for Space, physics-based extraction of the TSV would be possible. Applying physics-based extraction to the parasitic capacitances outside the substrate is likely less difficult than modeling the substrate correctly. Together, these additions would allow for physics-based extraction of ICs containing TSVs using Space.

8.2.4 Circular layout shapes

Currently, the layouts used for extraction by Space consist of horizontal, vertical and diagonal (strictly 45 degrees) lines. Circles can’t be defined. This means that circular structures, including TSVs, have to be approximated by a different shape. Allowing circular (and possibly even elliptical) shapes in layouts would ease the extraction process because there would no longer be a need to apply correction factors. It would also bring the layout seen by Space closer to the actual layout of the IC.
XML input file for model generation

The file tsv.xml can serve as an example of a layout and model generator input file and as input to the layout and model generator, as described in Chapter 7.

```xml
<cells>
  <count>2</count>
  <cell>
    <name>tsv1</name>
    <model>tsv1.dev</model>
    <layout>tsv1.ldm</layout>
    <bottom>tsub</bottom>
    <via>tsv1</via>
    <top>mtop</top>
    <substrate>sub</substrate>
    <tsv>
      <diameter>32e-6</diameter>
      <height>117e-6</height>
      <resistivity>1.68e-8</resistivity>
    </tsv>
    <bump>
      <diameter>70e-6</diameter>
      <height>15e-6</height>
      <resistivity>1.68e-8</resistivity>
    </bump>
    <oxide>
      <thickness>0.52e-6</thickness>
      <permittivity>4</permittivity>
    </oxide>
    <boxide>
      <thickness>0.5e-6</thickness>
      <permittivity>4</permittivity>
    </boxide>
    <imd>
      <thickness>6.5e-6</thickness>
      <permittivity>4</permittivity>
    </imd>
    <underfill>
      <thickness>15e-6</thickness>
      <permittivity>7</permittivity>
    </underfill>
    <silicon>
      <permittivity>11.9</permittivity>
      <conductivity>10</conductivity>
    </silicon>
    <pitch>220e-6</pitch>
    <lambda>1e-6</lambda>
  </cell>
  <cell>
    <name>tsv2</name>
  </cell>
</cells>
```
This appendix describes the various geometric shapes to be used to create a 3D model of a pair of TSVs for verification purposes, as described in Chapter 6.

**Bottom vacuum**

Shape: Cuboid  
X coordinates: 0 — 300 $\mu$m  
Y coordinates: 0 — 100 $\mu$m  
Z coordinates: 0 — 26.5 $\mu$m  
Material: Vacuum

**Bottom ground bump**

Shape: Cylinder  
XY radius: 35 $\mu$m  
Center (X, Y): (40, 50) $\mu$m  
Z coordinates: 26.5 — 41.5 $\mu$m  
Material: Copper

**Bottom signal bump**

Shape: Cylinder  
XY radius: 35 $\mu$m  
Center (X, Y): (260, 50) $\mu$m  
Z coordinates: 26.5 — 41.5 $\mu$m  
Material: Copper

**Bottom underfill**

Shape: Cuboid  
X coordinates: 0 — 300 $\mu$m  
Y coordinates: 0 — 100 $\mu$m  
Z coordinates: 26.5 — 41.5 $\mu$m  
Material: Silicon nitride

**Ground TSV**

Shape: Cylinder  
XY radius: 16 $\mu$m  
Center (X, Y): (40, 50) $\mu$m  
Z coordinates: 41.5 — 158.5 $\mu$m  
Material: Copper
Signal TSV
- Shape: Cylinder
- XY radius: 16 \( \mu m \)
- Center (X, Y): (260, 50) \( \mu m \)
- Z coordinates: 41.5 — 158.5 \( \mu m \)
- Material: Copper

Bottom oxide
- Shape: Cuboid
- X coordinates: 0 — 300 \( \mu m \)
- Y coordinates: 0 — 100 \( \mu m \)
- Z coordinates: 41.5 — 42 \( \mu m \)
- Material: Silicon dioxide

Ground oxide
- Shape: Cylinder
- XY radius: 16.52 \( \mu m \)
- Center (X, Y): (40, 50) \( \mu m \)
- Z coordinates: 42 — 152 \( \mu m \)
- Material: Silicon dioxide

Signal oxide
- Shape: Cylinder
- XY radius: 16.52 \( \mu m \)
- Center (X, Y): (260, 50) \( \mu m \)
- Z coordinates: 42 — 152 \( \mu m \)
- Material: Silicon dioxide

Substrate
- Shape: Cuboid
- X coordinates: 0 — 300 \( \mu m \)
- Y coordinates: 0 — 100 \( \mu m \)
- Z coordinates: 42 — 152 \( \mu m \)
- Material: Silicon

IMD
- Shape: Cuboid
- X coordinates: 0 — 300 \( \mu m \)
- Y coordinates: 0 — 100 \( \mu m \)
- Z coordinates: 152 — 158.5 \( \mu m \)
- Material: Silicon dioxide

Top ground bump
- Shape: Cylinder
- XY radius: 35 \( \mu m \)
- Center (X, Y): (40, 50) \( \mu m \)
- Z coordinates: 158.5 — 173.5 \( \mu m \)
- Material: Copper

70
Top signal bump
Shape: Cylinder
XY radius: 35 μm
Center (X, Y): (260, 50) μm
Z coordinates: 158.5 — 173.5 μm
Material: Copper

Top underfill
Shape: Cuboid
X coordinates: 0 — 300 μm
Y coordinates: 0 — 100 μm
Z coordinates: 158.5 — 173.5 μm
Material: Silicon nitride

Top vacuum
Shape: Cuboid
X coordinates: 0 — 300 μm
Y coordinates: 0 — 100 μm
Z coordinates: 173.5 — 200 μm
Material: Vacuum
Bibliography


