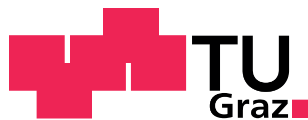
Master Thesis

Modelling and Simulation of Heat Transfer in Chip Cards based on Power Emulation

Michael Schaffernak

Institute for Technical Informatics
Graz University of Technology



Reviewer: Ass.-Prof. Dipl.-Ing. Dr. techn. Christian Steger

Advisor: Ass.-Prof. Dipl.-Ing. Dr. techn. Christian Steger
Dipl.-Ing. Dr. techn. Christian Bachmann

Graz, January 2013

Kurzfassung

Die Verwendung der Chipkarte zur Authentifizierung und zur Verschlüsselung von vertraulichen Daten hat in den letzten Jahren stark zugenommen. Mit Zunahme der Komplexität der Verschlüsselung und der Authentifizierung zugrundeliegenden Algorithmen wurde es notwendig High-End Systeme neben einem Mikroprozessor, mit einem Smartcard OS und einem zusätzlichen kryptografischen Coprozessor auszustatten. Die zum Betrieb notwendige Energie wird im Falle einer kontaktlosen Chipkarte mittels induktiver Koppelung aus dem elektromagnetischen Feld entnommen. Kommt es dabei zu einer Resonanzüberhöhung im Resonanzkreis, so wird eine für den Betrieb des Transponderkreises zu hohe Spannung induziert. Um etwaigen Schäden am System vorzubeugen, wird die überschüssige Energie über eine Shunt Regulierung in Form von Wärme an die Umgebung abgegeben. Aufgrund des exponentiellen Anstieges der Fehlerrate mit steigender Temperatur sind die Zuverlässigkeit und die Sicherheit des Systems gefährdet. Daher ist es notwendig den Einfluss der Wärmeübertragung auf das System bereits im frühen Stadium des Designs, als auch später unter realen Betriebsbedingungen zu berücksichtigen. Dazu ist ein genaues aber dennoch effizientes thermisches Modell von Nöten. Das Ziel dieser Masterarbeit ist es, die Wärmeübertragung in einer Chipkarte unter Verwendung des Power Emulation Ansatzes, welches im PowerHouse Forschungsprojektes entwickelt wurde, zu modellieren. Aus diesem Grund wird ein thermisches Modell auf Basis eines RC- Netzwerk auf mikro-architektur Ebene eingesetzt. Zusätzlich wird neben der Power Emulation ein Modell für die Berechnung der Leistungsaufnahme der Shunt Regulierung eingeführt. Ein Testsystem für Chip und Security ICs der Firma INFINEON Technologies in Graz, erweitert um einer Power Emulation Einheit, wird dazu verwendet um laufzeitgenaue Leistungsschätzwerte zu generieren welche dem thermischen Model zugeführt werden. Die Leistungsaufnahme eines Mikrochips bezogen auf dessen Oberfläche ist ungleichmäßig verteilt und führt zu räumlichen Temperatur Gradienten. Daher verwendet das thermische Model laufzeitgenaue Leistungsschätzwerte und Floorplan Information (Digitaler IC, Shunt, Memory etc.) um genaue Information zur Temperaturverteilung zu erhalten.

Stichwörter: Thermal Transient Simulation, Steady State Simulation, SoC, Chip Card, RFID, Power Emulation, Contactless-only Card, Dual Interface Card

Abstract

In recent years Chip Cards have been widely used for authentication and encryption of confidential data. With the increase in complexity of encryption and authentication algorithms, modern high-end systems involving a microprocessors, a smart card OS together with a cryptographic coprocessor are used. In the case of passive contactless smart cards, the needed energy has to be provided by use of inductive coupling. Due to the resonance step-up in the resonant circuit, the induced voltage in the transponder coil reaches higher values than necessary for the operation of the microchip. To avoid damage to the system, the voltage is regulated by means of a shunt regulator, generating a high amount of heat per unit volume, resulting in high operating temperatures. Due to the exponential increase of the failure rate with the raise of temperature, the reliability and safety of the system is being put at risk. Therefore the control of heat transfer in an early stage of the design and later under operating conditions is necessary. This requires a thermal model that is accurate, but still efficient.

The aim of this master thesis is to model the heat transfer in chip cards based on the Power Emulation approach derived in the PowerHouse research project. For this purpose a RC-circuit based thermal model at the micro-architectural level will be employed. In addition to the power emulation a power model of the resonant circuit including the shunt regulator will be introduced. The test system for Chip and Security ICs at INFINEON Technologies in Graz, augmented by a power emulation unit, will be used to obtain run-time power estimates, that will be fed to the thermal estimator. Since power dissipation is non-uniform across the chip, thus leads to spatial gradients. Therefore the thermal model uses run-time power consumption estimates and floorplanning information (digital ic, shunt, memory etc.) to gain detailed temperature information.

Keywords: Thermal Transient Simulation, Steady State Simulation, SoC, Chip Card, RFID, Power Emulation, Contactless-only Card, Dual Interface Card

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....	....
date	(signature)

Danksagung

Diese Diplomarbeit wurde in der Firma Infineon Technologies im Design Center Graz in Zusammenarbeit mit dem Institut für Technische Informatik an der Technischen Universität Graz durchgeführt.
Im Besonderen bedanke ich mich bei Herrn Dr. Josef Haid und DI Thomas Leutgeb für die Betreuung und Unterstützung bei der Firma Infineon. Des Weiteren möchte ich mich recht herzlich bei Andreas Müller-Hipper und Bettina Latka, Mitarbeiter der Firma Infineon Regensburg, für die Beschaffung von Spezifikation und Daten, bedanken. Einen herzlicher Dank gilt auch Ulrich Fröhler, Mitarbeiter der Firma Infineon München, für die Durchführung der Referenzsimulationen.
Für die Betreuung seitens des Instituts und die Durchsicht meiner Diplomarbeit bedanke ich mich recht herzlich bei Ass.-Prof. Dipl.-Ing. Dr. techn. Christian Steger. Mein besonderer Dank gilt meinem Betreuer Dipl.-Ing. Dr. techn. Christian Bachmann, der mit seiner Unterstützung und hilfreichen Ratschlägen einen wichtigen Teil zum erfolgreichen Abschluss meiner Diplomarbeit beigetragen hat.
Bedanke möchte ich mich auch bei Andreas Schwarz, der mich bei vielen Formulierungen und auch bei der Korrektur der Diplomarbeit hilfreich unterstützt hat.
Nicht zuletzt gebührt meiner Familie großer Dank, insbesondere meinen Eltern, für die jahrlange Unterstützung, ihre viele Geduld und ihren Glaube an mich.

Graz, Jänner 2013 Michael Schaffernak

Contents

1 Introduction **11**
1.1 Motivation . . . 11
1.2 Goals of the Thesis . . . 12
1.3 Thesis Overview . . . 12

2 Related Work **13**
2.1 Power Estimation . . . 13
2.1.1 Introduction . . . 13
2.1.1.1 Dynamic Power . . . 13
2.1.1.2 Static Power . . . 14
2.1.2 Power emulation . . . 15
2.1.2.1 RTL-level Power Emulation . . . 15
2.1.2.2 High-Level Power Emulation . . . 16
2.2 Thermal Estimation . . . 17
2.2.1 Introduction . . . 17
2.2.1.1 Conduction . . . 18
2.2.1.2 Convection . . . 19
2.2.1.3 Radiation . . . 19
2.2.1.4 Heat diffusion equation . . . 20
2.2.1.5 Boundary conditions . . . 20
2.2.2 Methods of Thermal Modelling . . . 21
2.2.2.1 Analytical Thermal Model . . . 21
2.2.2.2 Numerical Thermal Model . . . 23
2.2.2.2.1 Finite Difference Method . . . 24
2.2.2.2.2 Finite Element Method . . . 25
2.2.2.2.3 Boundary Element Method . . . 26
2.2.2.3 RC Network . . . 27
2.2.2.4 RC Network by Reduction . . . 28
2.3 Chip Card . . . 29
2.3.1 Introduction . . . 29
2.3.2 Transponder Overview . . . 30
2.3.3 Inductive Coupling . . . 31
2.3.3.1 Inductive Voltage u_i . . . 31
2.3.3.2 Resonance Frequency f_{res} . . . 32
2.3.3.3 Quality Factor Q_0 . . . 33

3 Design **34**
3.1 Overview . . . 34
3.2 Requirements . . . 36
3.3 Power Model . . . 36
3.3.1 The Analogue Power Model . . . 36
3.3.2 Manual Power Profile . . . 37
3.4 Package Model . . . 37
3.4.1 Introduction . . . 37
3.4.2 Package Data . . . 38
3.4.3 Meshing . . . 38
3.4.4 Layer Building and RC Calculation . . . 38
3.5 Thermal Simulator . . . 39
3.5.1 Power Information Matching . . . 39
3.5.2 Components of the Thermal Network . . . 39
3.5.3 Convection and Radiation Heat Transfer . . . 40
3.5.4 Transient Simulation . . . 40
3.5.5 Steady State Simulation . . . 41
3.6 Visualisation and Validation . . . 41
3.6.1 Validation . . . 41

4 Implementation **43**
4.1 Overview . . . 43
4.2 The Package Model . . . 45
4.2.1 Introduction . . . 45
4.2.2 R-C Representation . . . 45
4.2.3 Floorplan . . . 46
4.2.4 Package Data . . . 47
4.2.5 Meshing . . . 49
4.2.6 Layer Building . . . 52
4.3 The Thermal Simulator . . . 54
4.3.1 Power Information Matching . . . 54
4.3.2 Building the network . . . 54
4.3.3 Convection and Radiation Heat Transfer . . . 57
4.3.4 Transient Simulation . . . 57
4.3.5 Steady State Simulation . . . 61
4.4 The Power Model . . . 62
4.4.1 Power Emulator Mode . . . 62
4.4.1.1 Analog Power Model . . . 63
4.4.1.2 Building the Power Matrix . . . 65
4.4.2 Manual and Default Mode . . . 65
4.5 Thermal GUI . . . 66
4.5.1 Main Interface . . . 66
4.5.2 Saving Point Table Interface . . . 67
4.5.3 Settings . . . 67

5 Experimental Results **69**
5.1 Introduction . . . 69
5.2 Validation with Hotspot . . . 69
5.2.1 Block Model versus ThermalCube . . . 69
5.2.2 Grid Model versus ThermalCube . . . 71
5.3 Speedup by Optimised Algorithm . . . 73
5.4 Validation of ThermalCube with FEM Solver Ansys . . . 75
5.4.1 Contactless Only Module COM . . . 76
5.4.2 Dual Interface Module DIM . . . 82
5.5 Thermal Simulation with PE power estimates . . . 86

6 Conclusion and Future Work **89**
6.1 Conclusion . . . 89
6.2 Future Work . . . 90

Abbreviations and Symbols **91**

Literaturverzeichnis **95**

List of Figures

2.1 Charge and discharge process in a CMOS inverter during switching [20]. . . 13
2.2 I-V characteristics of a semiconductor diode [21]. 14
2.3 Binary search at RTL level augmented by power emulation hardware [11]. . 15
2.4 The architecture of the high-level power emulation platform [14]. 16
2.5 The temperature change over time until reaching the systems steady state [8]. 17
2.6 The three types of heat transfer [8, modified]. 18
2.7 The variation of the thermal conductivity of copper an silicon in relation to temperature [12, materials data from]. 20
2.8 Schematic of the thermal model with boundaries [19]. 21
2.9 Thermal subvolume [5]. 22
2.10 The variable x is subdivided in discrete points, $x_i = i\Delta x$ from $i = 0$ to L . 24
2.11 Shapes of three-dimensional finite elements [23, modified]. 25
2.12 VLSI chip with packaging [31]. 26
2.13 Example of the compact thermal model. 28
2.14 The DELPHI compact model. 29
2.15 Transponder equivalent circuit diagram (ECD). 31
2.16 Inductive coupling between reader and tag. 32

3.1 Thermal Estimation Architecture. 35
3.2 Design of the analogue power model. 36
3.3 Structure of the manual power profile. 37
3.4 Thermal Estimation Architecture. 39
3.5 The thermal core of the transient simulation 40

4.1 Implementation overview / Calling Graph. 44
4.2 R-C representation of one element. 45
4.3 Possible floorplan of a μChip and the difference in its definition as FLP or SVG format. 46
4.4 Possible smartcard package. 47
4.5 The base grid of the smartcard package in Figure 4.4 extended with additional segmentation defined by the user. 49
4.6 Adjacency of one element. 54
4.7 Matrix Density of sparse matrix S. 55
4.8 Convection heat transfer coefficients gain from the ANSYS macro compared to the implemented h_c with C = 0.289. 58

4.9 Illustration of the power trace matrix notation and the flow chart of building the power matrix. 63
4.10 Illustration of three three manual power traces as a result of the Listing 4. . 66
4.11 Main Thermal GUI. 67
4.12 Saving Point Table GUI. 68
4.13 Setting GUI. 68

5.1 Hotspot Block versus ThermalCube Manual. 70
5.2 16x16 Mesh of the DIE layer in ThermalCube. 71
5.3 The core's steady state temperatures. 72
5.4 Hotspot Grid versus ThermalCube. 72
5.5 Schematic of the contactless-only smartcard. 73
5.6 Security controller schematic. 75
5.7 COM Setup: (a) and (b) are the real modules, (c) is the representation used in the simulation and (d) is the cross section of the smartcard. 76
5.8 Load case 1: COM package with 4.1 x 3.2 $mm^2 \mu$Chip in a PVC ID-1 Card. 78
5.9 Load case1: Transient behaviour of 4.1 x 3.2 $mm^2 \mu$Chip in COM package enveloped in a PVC ID-1 Card. 79
5.10 Load case 6: COM package with 4.1 x 3.2 $mm^2 \mu$Chip in a PVC ID-1 Card. 80
5.11 Load case6: Transient behaviour of 4.1 x 3.2 $mm^2 \mu$Chip in COM package enveloped in a PVC ID-1 Card. 81
5.12 DIM Setup: (d) shows the real module (ISO and Chipside), (a) and (b) are the modelled ISO contact and chip side, and (c) is the cross section of the dual interface smartcard. 83
5.13 Load case 1: DIM package with 4.6 x 5.5 $mm^2 \mu$Chip in a PVC ID-1 Card. . 84
5.14 Load case1: Transient behaviour of 4.6 x 5.5 $mm^2 \mu$Chip in DIM package enveloped in a PVC ID-1 Card. 85
5.15 The chip floorplan (a) and the package floorplan (b). 86
5.16 Transient behaviour of the C2012 controller during core performing an alu arithmetic benchmark test. The total power is gained from the electromagnetic field with $H_{max} = 12 \left[Am^{-1}\right]$. 87
5.17 Temperature distribution (a) after 1 second (b) in steady state. 88

List of Tables

2.1 Comparison of power estimation and power emulation approach for the MPEG4 design [11, with modification]. 16
2.2 Duality between thermal and electrical quantities [27]. 27

5.1 Thermal Properties and Package Dimensions. 70
5.2 Comparison of the Steady State Temperatures. 71
5.3 Performance Speedup. 73
5.4 Package components and material properties. 74
5.5 Load Cases [mW]. 75

Chapter 1

Introduction

1.1 Motivation

Heat Generation in integrated circuits is one of the top items in the development of high integrated electronic circuits. Increased miniaturisation means an increased rate of heat generation $[Wm^{-3}]$, resulting in higher operating temperatures and hotspots. The impact of temperature effects are:

- Degradation in carrier mobility (Mobility Variation with Temperature):
  $\mu(T) = \mu(T_0) \cdot (\frac{T_0}{T})^{m_\mu}$ with $m_\mu \approx 1.5$ at measured Temperature T_0 [29]
- An exponential increase in sub-threshold leakage (Threshold variation with voltage).
- The increase in interconnect resistivity.
- Reliability problems and safety risks on the system caused by a higher failure rate, increasing exponentially with higher temperatures.

A system on a chip or System on Chip (SoC) represents one of the highest integrated circuits solutions containing all components of a computer system on a single substrate. Several components on the substrate performs different activities on different power dissipation levels resulting in non-uniform power dissipation with spatial thermal gradients and hot spots. Operation modes with high dynamic power peaks, temperatures with timely delay are typically for this kind of applications and reducing the efficiency of these techniques. Proper thermal packaging solutions for covering worst case scenarios is usual to expensive. Thermal estimation and controlled power limitations are necessary to prevent the system from malfunction.

A typical application of SoC is in the area of contactless chip cards used for authentication and encryption of confidential data. The high complexity of encryption and authentication algorithm requires the introduction of modern high-end systems containing microprocessors with a smart card operating system (OS) together with a cryptographic coprocessor. In the case of passive contactless smart cards the energy has to be transmitted by inductive coupling to the passive on chip transponder. Due to the resonance step-up principle in the resonant circuit the induced voltage in the transponder coil reaches higher values than necessary for the operation of the microchip. To avoid damage to the system,

the voltage is regulated by means of a shunt regulator, generating a high amount of heat per unit volume, which results in high operating temperatures. Furthermore the enhanced cryptographic procedures in RFID systems lead to a raise in power performance and subsequently, in power dissipation in the multiprocessor and the cryptographic coprocessor.

Due to the exponential increase of the failure rate with the raise of temperature, the reliability and safety of the system is putt at risk. Therefore, the control of heat transfer in an early stage of the design and later under operating conditions, is necessary. This requires a compact thermal model that is accurate but still efficient.

1.2 Goals of the Thesis

The aim of this master thesis is to model and simulate the heat transfer in chip cards based on the power emulation approach derived in the "Power House" research project. For this purpose, a RC circuit-based thermal model at the micro-architectural level will be employed. In addition to the power emulation, a power model of the resonant circuit including the shunt regulator will be introduced. The test system for chip and security ICs at INFINEON Technologies in Graz, augmented by a power emulation unit, will be used to obtain runtime power estimates that will be fed to the thermal estimator. Power dissipation is non-uniform across the chip, thus leads to spatial gradients. Therefore the thermal model uses run-time power consumption estimates and floorplan information (digital IC, shunt, memory, etc.) to gain detailed temperature information.

1.3 Thesis Overview

In Chapter 2 the power consumption and the heat transfer fundamentals of CMOS based integrated circuits are explained. Power emulation approach and different techniques to model the thermal behaviour of integrated circuits are introduced. Chapter 3 presents the design of the thermal simulator architecture using the power emulation approach to obtain power estimates. The implementation of the transient and steady state simulator together with the power model are presented in Chapter 4. Furthermore the requirements are given. The validation of a first version against HOTSPOT [16] as well as the final version against ANSYS [17] are discussed in Chapter 5. Also a use case of a contactless smartcard executing a benchmark program is shown. In Chapter 6 a conclusion of the master thesis and some ideas for possible future work are presented.

Chapter 2

Related Work

2.1 Power Estimation

2.1.1 Introduction

Reducing the power consumption and increasing the power efficiency respectively are today's most important design constraints, along with performance and the upcoming temperature awareness, in the development of CMOS based integrated circuits. In the following, the mayor sources of power consumption and its twofold relationship to heat dissipation in CMOS based integrated circuits will be identified. On the one hand the thermal behaviour strongly depends on the amount of power that is dissipated per unit volume and on the other hand the amount of power consumption depends on temperature. The source of power dissipation can be classified into dynamic and static power consumption [21].

2.1.1.1 Dynamic Power

The dynamic power represents the most dominate factor of power dissipation in digital CMOS circuits. The process of charging and discharging of parasitic capacitance during the high-to-low and the low-to-high transition respectively, as seen in Figure 2.1, and the short-circuit path from voltage source to ground, when the inputs are in transition, causes a energy dissipation E_{dyn} that is converted into thermal energy [24, 9].

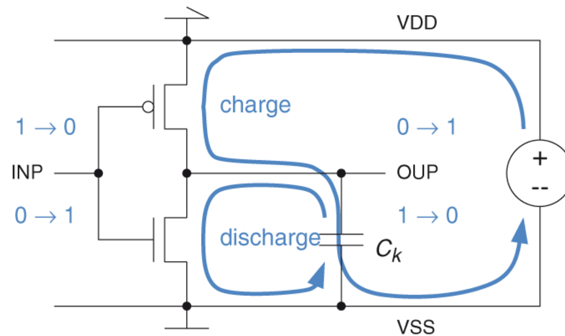


Figure 2.1: Charge and discharge process in a CMOS inverter during switching [20].

The Power that is dissipating during one computing cycle is given by

$$P_{dyn} = E_{dyn} \cdot T_{cycle}^{-1} = C_L \cdot V_{dd} \cdot \alpha \cdot f \text{ [W]} \quad (2.1)$$

with the capacitance load C_L, that comprises the parasitic capacitance of CMOS gates and interconnects, the supply voltage V_{dd} $[V]$, the clock frequency f $[s^{-1}]$ and the switching activity α [24].

2.1.1.2 Static Power

Static power consumption refers to power consumption during non-switching activity of transistors. In CMOS technology static power come up in form of leakage power $P_{leak} = V_{dd} \cdot I_{leak}$, that has become a dominate part of the total power consumption in recent years. Due to the analogue nature of transistors, leakage currents flow although the gate voltage is below the threshold voltage, as seen in Figure 2.2, that in an idealised view would mean the off state of the transistor [20]. The *Subthreshold leakage* and the *gate-oxide leakage* are

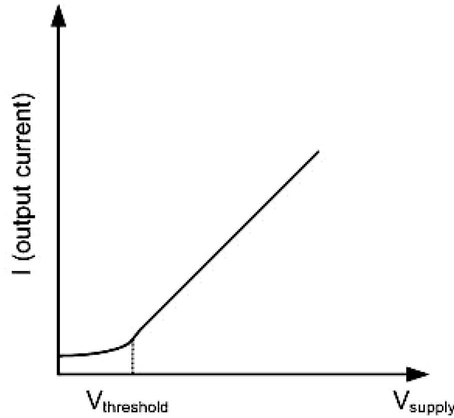


Figure 2.2: I-V characteristics of a semiconductor diode [21].

the two dominant factors of leakage currents. The need of reducing the dynamic power consumption by scaling the supply voltage has led to a huge increase in both subthreshold and gate leakage.

Together with supply voltage scaling it is necessary to lower the threshold voltage to avoid an increase of the transistor delay and hence prevent the system of switching speed loses. This in turn results in an exponentially increase in subthreshold leakage due to lowering of the threshold voltage V_T, as can be seen easily from the following equation [21]:

$$I_{Dsub} = I_{s0} \left(1 - e^{\frac{-V_{ds}}{v_t}}\right) e^{\frac{V_{gs} - V_T - V_{off}}{n \cdot v_t}} \quad (2.2)$$

where V_{off} is an empirically determined model parameter and v_t is the **thermal voltage**. This thermal voltage term is critical. If any source of power consumption causes a raise in temperature, the thermal voltage v_t increases due to its proportional dependency to temperature. As seen from Equation 2.2 this results in an exponential increase in leakage power leading, in turn, to an increase in temperature.

With the scaling of the supply voltage, other physical quantities are scaled as well. The gate insulator, that separates the gate from the channel gets thinner, and enables the direct tunnelling of electrons due to quantum mechanics. This leads to a lesser temperature dependent leakage current, called the *Gate leakage*.

2.1.2 Power emulation

2.1.2.1 RTL-level Power Emulation

The analysis of the temperature behaviour of a system requires a detailed knowledge of its power consumption. In order to gain the necessary power information a large number of power profiling techniques exist. In contrast to measurement based methods that physically cover the current consumption of the physical chip, estimation based methods determine power values by using the circuit activity information during executing of a program on a simulator [14].

Since power estimation can be performed at different layers of abstraction, power considerations can be taken into account at early stages of the design process. The lower the level of abstraction, the more accurate estimation results can be achieved, but also the higher the computational effort raises. Therefore the authors in [11] introduced a new hardware-accelerated power estimation approach called power emulation. The idea is to build up the power models of the the estimation process in hardware circuits. By emulating any given system augmented with the power estimation hardware and mapping onto a hardware prototyping platform, they achieved a high order of magnitude speedup [11]. Figure 2.3 shows the new enhanced RTL circuit, augmented with power models for each component. Each power model takes the input signals of one RTL component as input for computing the desired power value. The power generator finally outputs the accumulated total power of all power models. The Strobe Generator acts as a clock to ensure synchronous behaviour of the power estimation process [11]. The power emulation

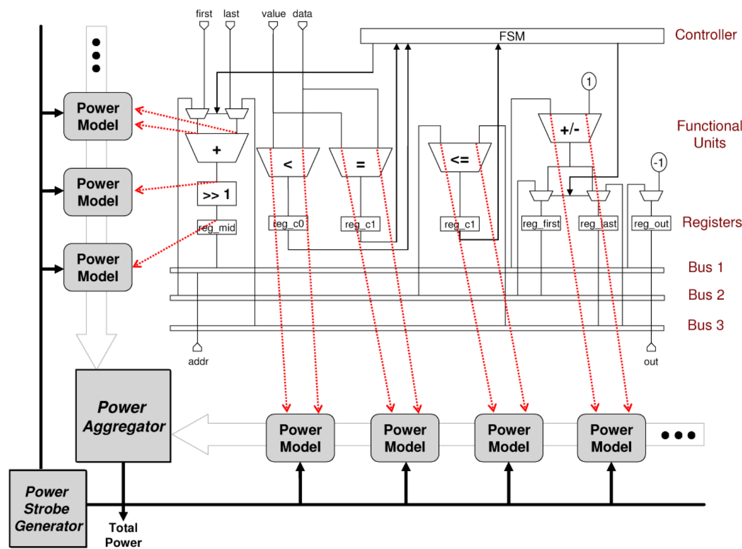


Figure 2.3: Binary search at RTL level augmented by power emulation hardware [11].

approach was applied to a MPEG4 decoder and its performance was compared with two commercial available power estimators. By comparison of the execution times, as seen in table 2.1, the result of the power emulator exhibits a huge speedup of 524 and 411 times respectively [11]. However, this raise in speedup comes along with an increase

	NEC-RTPower	PowerTheater	Power Emulation
Run Time	3300 sec	2587sec	6.3 seconds

Table 2.1: Comparison of power estimation and power emulation approach for the MPEG4 design [11, with modification].

of hardware overhead that is 18.2 times the area of the original design. With different proposed optimization techniques as resource sharing, reuse through clustering and inter component power correlation a reduction to an average of 3.2 times the area was achieved by an mean estimation error of 3.4%[11].

2.1.2.2 High-Level Power Emulation

For developing of power efficient applications it is necessary to provide software designers with power information of the underlying system. Low level power estimation tools analysing complex systems taking a lot of computation effort and are often not available for software developer. Although power emulation at RTL level is able to speedup the estimation the hardware overhead for complex system still enormous [14]. Therefore Genser

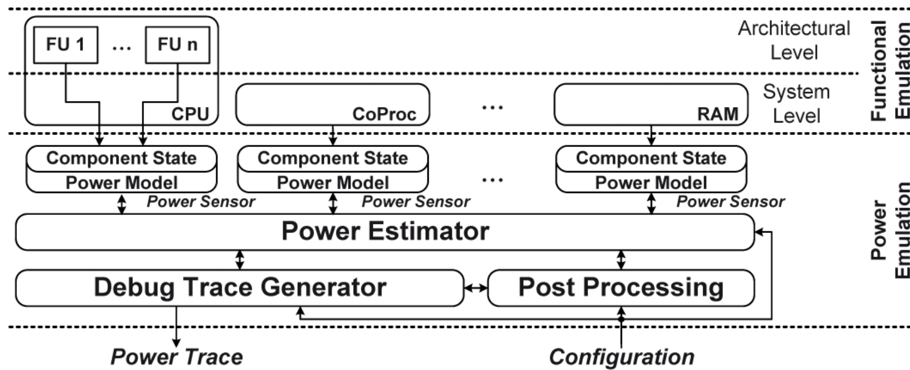


Figure 2.4: The architecture of the high-level power emulation platform [14].

et al. [14] have introduced a system level power profiling approach based on the above power emulation principle. The high level power emulation architecture is illustrated in Figure 2.4. The power model implements a linear regression model of the form

$$y = \sum_{i=0}^{n-1} c_i + x_i \epsilon \tag{2.3}$$

whereas the model parameters x_i represents the system states, the model coefficients c_i contains power information and ϵ is the error between the real power value and the estimate y. In a beforehand performed characterisation process the power coefficients are determined and together with the state vectors stored as configurable lookup table inside

appropriate power sensors. Each power sensor is able to monitor the state information of the assigned component and translate it to the corresponding power value using its lookup table. The power values of all power sensors are then accumulated by the power estimation unit to the overall power estimate. Finally the debug-trace generator modules provide the host computer with the power information for evaluation [14].

The high-level power emulation approach is applied to a contactless smart card system which runs power critical applications. In comparison to a gate-level simulation by executing a typical payment application the power emulation platform takes 338 microseconds in contrast to 98 hours with a 90% accuracy and a hardware increase of 1.5% to the overall system [14]. In [3] Bachmann et al. refine this approach by an automated power characterisation process and presenting an automated adaption of the HDL model implementation of the system under test for integrating the power emulation unit.

The high level power emulation, which is implemented as extension to the test system for Chip and Security ICs at INFINEON Technologies in Graz during the POWER-HOUSE project will be used to obtain runtime power estimates for the thermal simulator presented in this master thesis.

2.2 Thermal Estimation

2.2.1 Introduction

In the design of integrated circuits the knowledge of the thermal behaviour of the system at any working load under varying environmental conditions is important for avoiding thermal runaway. The interest in the thermal analysis of a design usually lies in determining not only the spatial but also the temporal distribution of the temperature. If only the temperature gradient is required when the system is in equilibrium with its environment a so called steady state analysis can be performed. Solving such a steady heat transfer problem is easy as the temperature does not vary with time. As a result, the steady operating temperature which is usual reached after a particular period of time as stated in Figure 2.5 is almost gained instantaneous. While the time independency implies constant

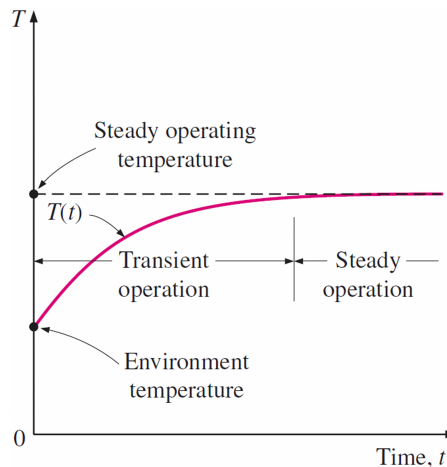


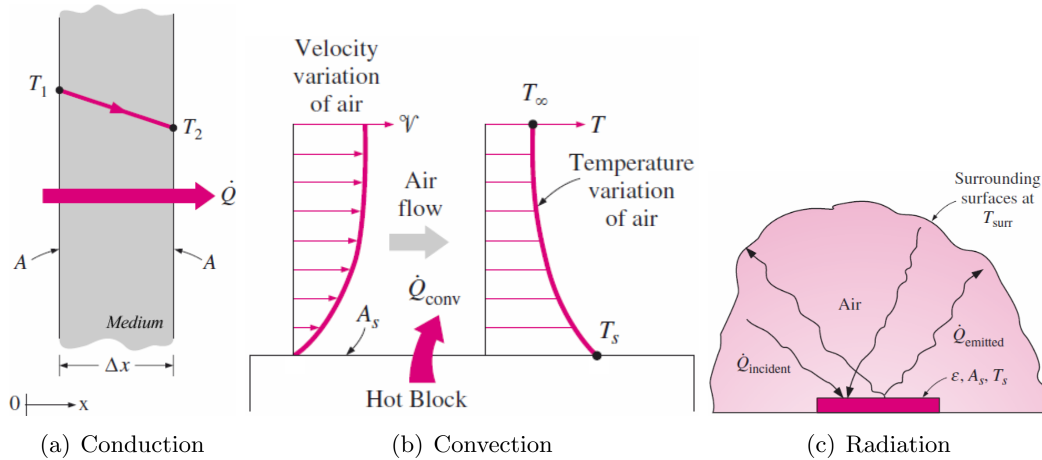
Figure 2.5: The temperature change over time until reaching the systems steady state [8].

power consumption over time, a system with dynamic power behaviour underlies a temperature change over time and may never reach a steady state temperature. Therefore a transient thermal analysis can be done to catch the thermal dynamics. Performing either analysis techniques requires the knowledge of the heat transfer path and the environmental conditions of the system.

The heat is transferred from the active elements of a system through its package to the ambient in the direction of decreasing temperature, as stated by the second law of thermodynamics. Thus the requirement for any heat transfer is the presence of a temperature difference as driving force analogue to the voltage difference for electric current [8]. The heat transfer is classified into three types [8]:

- Conduction
- Convection
- Radiation

2.2.1.1 Conduction



(a) Conduction (b) Convection (c) Radiation

Figure 2.6: The three types of heat transfer [8, modified].

Before reaching the systems ambient medium, the heat usually passes several layers, which consist mainly of solids as well as gases or liquids. The heat transfer between two layers result from interaction between adjacent particles of different energetic levels and is named conduction (see Figure 2.6(a)). The amount of heat that is transferred per time unit across the area of a medium with the thickness $\triangle x$ underlying a temperature difference $\triangle T$ is described by **Fourier's law of heat conduction**

$$\dot{Q}_{cond} = -kA\frac{dT}{dx}\,[W] \tag{2.4}$$

where $\dot{Q}$ is the *heat transfer rate* [W], k is the *thermal conductivity* $[Wm^{-1}K^{-1}]$ of the material, A is the *area* $[m^2]$ and dT/dx is the *temperature gradient* $[Km^{-1}]$. Since the temperature gradient is negative due to the decreasing temperature in the positive x direction the negative sign is necessary to gain a positive quantity.

2.2.1.2 Convection

The heat from the packaging surface to the adjacent ambient air is transferred by means of *convection* (see Figure 2.6(b)) which is a combination of conduction and fluid motion. The fluid motion over the surface can be caused by external forces, likely by a fan (*forced convection*) or occurs as a result of buoyancy forces due the raise of warmer and fall of colder fluid (*natural convection*). Aside from fluid motion, convection depends on a large number of physical parameters as dynamic viscosity, thermal conductivity, density, fluid velocity and specific heat as well the geometry and the surface orientation, leads to a more complex determination of the convection heat transfer. However, the rate of the convection heat transfer can expressed by **Newton's law of cooling**

$$\dot{Q}_{convection} = hA_s(T_s - T_\infty)\,[W] \tag{2.5}$$

where h is the convection heat transfer coefficient $[Wm^{-2}K^{-1}]$, A_s is the surface area $[m^2]$, T_s is the surface temperature [K] and T_∞ is the temperature of the ambient air [K].

Although this expression seems quite simple, the convection heat transfer coefficient h is not a material property but covers the relation of the package to its environment with all physical parameters mentioned above.

2.2.1.3 Radiation

In addition to the convection heat transfer, energy is emitted from the surface in form of electromagnetic waves as a result of the body's temperature, called *thermal radiation.* The rate of radiation $\dot{Q}_{rad}$ which is emitted from the surface A_s of a black body at an absolute Temperature T_s (in Kelvin) is described by the **Stefan-Boltzmann law**

$$\dot{Q}_{rad} = \rho A_s(T_s^4)\,[W] \tag{2.6}$$

with the *Stefan-Boltzmann constant* $\rho = 5.67 \cdot 10^{-8}[Wm^{-2}K^{-4}]$. Since the law describes the maximum rate of radiation a general form is available that defines the radiation emitted from a real body as a portion of the black bodies radiation, characterised by the emissivity $0 \leq \epsilon \leq 1$

$$\dot{Q}_{rad} = \epsilon\rho A_s T_s^4 \tag{2.7}$$

By involving the ambient air as a much larger surface which covers the surface area of the body, as illustrated in Figure 2.6(c), the rate of radiation heat transfer between them can be expressed as

$$\dot{Q}_{rad} = \epsilon\rho A_s(T_s^4 - T_{surr}^4)\,[W] \tag{2.8}$$

Since radiation and convection takes place in parallel, the total heat transfer from the surface can be written as

$$\dot{Q}_{total} = h_{comb}A_s(T_s - T_\infty)\,[W] \tag{2.9}$$

where h_{comb} is the combined heat transfer coefficient including both radiation and convection effects.

2.2.1.4 Heat diffusion equation

The formulation of the conduction heat transfer, Equation 2.4 in Section 2.2.1.1 , describes the rate of heat in only one dimension. However, in general, heat in a medium propagates in all three primary directions. Thus, the temperature of a body can be expressed as a function of time and position $T = f(x, y, z, t)$ [8]. The three-dimensional temperature distribution can be described by the heat diffusion equation

$$\frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\partial T}{\partial z}\right) + \frac{\dot{g}}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{2.10}$$

where $\alpha = k/\rho C$ is the thermal diffusivity. The thermal conductivity in the equation is assumed to be constant as a matter of simplification. However, it naturally varies with temperature k(T) leading to a more complex heat diffusion equation to solve [22]. To obtain an approximately constant value of k(T) an average value over a desired range can be determined. This method is purely desirable as long as the variation in the specific interval is not large [8]. As depicted in Figure 2.7 the value for copper in the range of $190[K]$ to 400 $[K]$ can be assumed as almost constant whereas the value for silicon in the same interval can not without introducing some error.

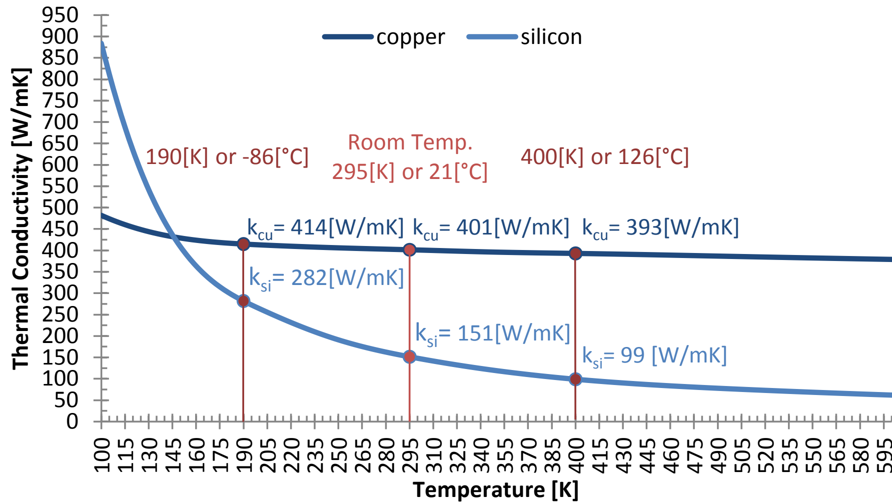


Figure 2.7: The variation of the thermal conductivity of copper an silicon in relation to temperature [12, materials data from].

2.2.1.5 Boundary conditions

The heat conduction equation 2.10 is a partial differential equation (PDE). In order to solve the equation it is necessary to specify the thermal conditions at the boundaries of the medium and the initial thermal state. The mathematical expressions of these conditions are called boundary conditions and initial conditions, respectively. For each direction, the heat transfer is significant two conditions are needed. Typical boundary conditions besides convection and radiation are

heat flux $-k \cdot \partial T/\partial n = \dot{q}[W/m^2]$

insulation $\partial T/\partial n = 0$

isothermal $T = T_s$ where T_s is a specified temperature at the surface

perfect interface $-k_1 \partial T_1/\partial n = -k_2 \partial T_2/\partial n$ where k_1 and k_2 are the conductivities of two bodies in perfect contact

imperfect interface $-k_1 \partial T_1/\partial n = h_{int} A \Delta T_{int}$ where h_{int} is the thermal interface conductance $[Wm^{-2}K^{-1}]$ and ΔT_{int} is the temperature difference at the interface

where n means normal to the surface[22, 8]. Selecting the right set of boundary conditions important to satisfy real world conditions, and thus achieving appropriate accuracy.

2.2.2 Methods of Thermal Modelling

2.2.2.1 Analytical Thermal Model

Analytical thermal models are based on the description of the heat diffusion problem by means of partial differential equation similar to equation 2.10 introduced above. Solving the equation together with the right set of boundary conditions results in a unique solution that is able to describe the temperature on each spatial point at any point in time.

Janicki et al. [19] present an analytical thermal model of an integrated circuit attached to a heat sink using the steady state form ($\partial T/\partial t = 0$) of the heat diffusion equation. The schematic of the model with the chosen boundaries are illustrated in Figure 2.8. The

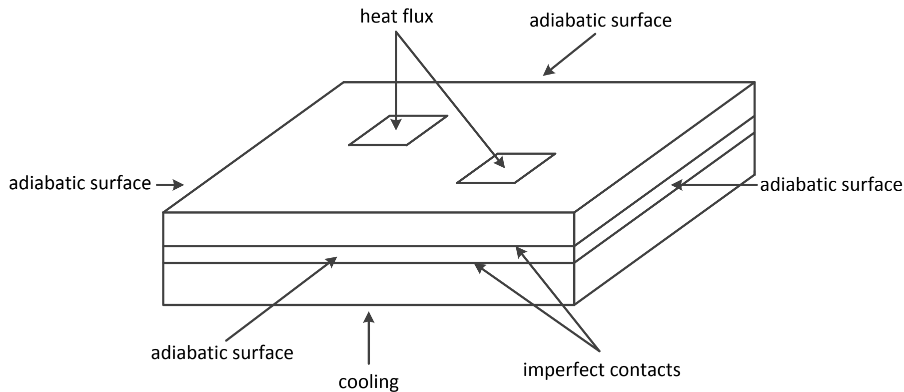


Figure 2.8: Schematic of the thermal model with boundaries [19].

power dissipation is modelled as heat flux at the top surface. The lateral sides of all layers are assumed to be perfectly isolated (adiabatic surface) while the contact between them is treated as an imperfect contact interfaces. The heat is removed from the bottom side by means of convection which is simplified as an uniform process. For solving the PDE, separation of the variables is used to gather the solution function

$$T_{x,y,z} = a(x, y, z) \cdot q \tag{2.11}$$

where q is a single heat source with constant density and the coefficients a(x,y,z) are expressed as sums of Fourier series.

For validating the model thermographical measurements on a MC 33186 circuit containing four large power transistors were performed while one transistor was dissipating power. The result shows a temperature overestimate at high power values and a maximum deviation of 13%.

In contrast to the authors Chiueh et al. [10] present an analytical transient solution for on-chip heat dissipation in VLSI design. The power dissipation of the chip is posed as periodic heat source at the centre of the top surface. The upper and surrounding sides contact with the ambient air are specified as isolated while the bottom side of the attached heat sink is assumed to be isothermal. The solution function of a chip with the dimension $L \times L \times w$ describes a periodic function with infinite unit heat sources, which enables the determination of the heat temperature distribution at any point in time for $t > 0 \wedge 0 < \tau < t$

$$u(x,y,z,t) = \sum_{\tau=0}^{\infty} 8 \cdot \left[\sum_{k=0}^{\infty} e^{\frac{-4k^2\pi^2\left(t+\frac{\tau}{f}\right)\lambda}{L^2}} \cos\frac{2k\pi x}{L} \right] \cdot \left[\sum_{k=0}^{\infty} e^{\frac{-4k^2\pi^2\left(t+\frac{\tau}{f}\right)\lambda}{L^2}} \cos\frac{2k\pi y}{L} \right] \cdot \left[\sum_{k=0}^{\infty} e^{\frac{-4k^2\pi^2\left(t+\frac{\tau}{f}\right)\lambda}{L^2}} \cos\frac{(2k+1)\pi z}{2w} \right] \quad (2.12)$$

where $\lambda = \frac{K}{\rho C_p}$, K is the *thermal conductivity*, ρ is the *density*, C_p is the *heat capacity* of silicon and f is the frequency of the circuit. By varying the index k of the summation it is possible to adjust the granularity of the result. The model is validated using a test chip with several spirally arranged circuit modules. Each of them contains a pad driver as heat sources, a diode and a poly resistor as temperature sensor. The differences between measurement and the model estimates came to 13%.

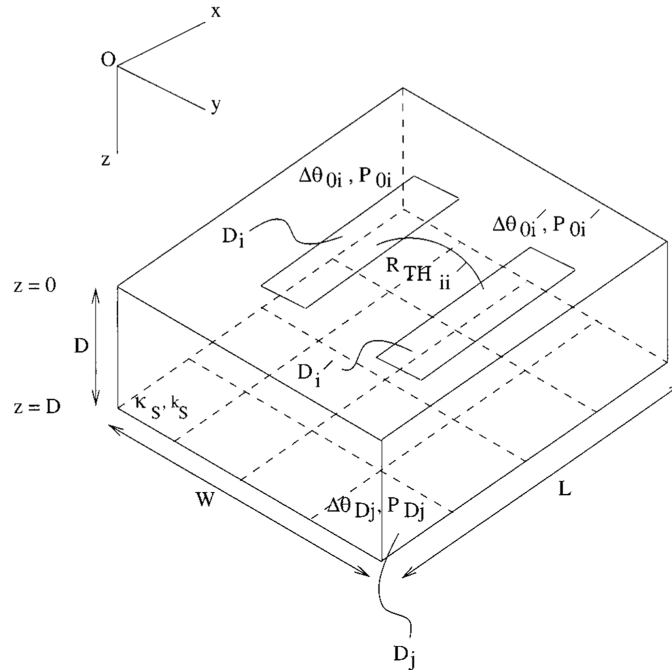


Figure 2.9: Thermal subvolume [5].

Batty et al. [4] proposed a time dependent fully analytical thermal model approach applicable for electrothermal simulation of semiconductor integrated circuits. To circumvent the nonlinearity of the three dimensional heat diffusion equation due the temperature dependency of the material properties, the Kirchhoff transformation and an additional time variable transformation is performed. The linearised equation becomes

$$\nabla^2\theta - \frac{1}{k_s}\frac{\partial\theta}{\partial\tau} = -\frac{g}{\kappa_s} \tag{2.13}$$

where θ is the transformed temperature, k_S is the temperature independent thermal diffusivity and κ_S is the temperature independent thermal conductivity. To solve the equation, the problem domain is divided into regular sub-volumes, see Figure 2.9, by means of domain decomposition.

After a discretisation of only the interfaces between subvolumes, the power dissipation elements and thermal sensitive elements of each subvolume is solved in the Laplace S-space under specified boundary conditions. The resulting equation in the Laplace domain of the subvolume elements becomes the form

$$\overline{\Delta\theta_i} = \sum_j R_{TH_{ij}}(s)\overline{P_j} \tag{2.14}$$

where $\overline{\Delta\theta_i}$ is the temperature rise of element i, $R_{TH_{ij}}(s)$ is the thermal impedance matrix and $\overline{P_j}$ is the corresponding heat flux of the element j. The thermal impedance matrix is obtained from the structural information and involves neither power nor temperature information and hence can be precalculated.

The conditions at the top and bottom boundaries of every subvolume are described as a generalised boundary condition while the sides are treated as perfect isolated. The generalised condition for the top $z = 0$ and bottom surfaces $z = D$ is expressed as

$$\alpha_{0,D}\kappa S\frac{\partial\theta}{\partial z} + H_{0,D}\left(\theta - \theta_{0,D}\left(x, y, \tau\right)\right) + p_{0,D}\left(x, y, \tau\right) = 0. \tag{2.15}$$

The coefficient $\alpha_{0,D}$ is equal to 0 for a specified temperature boundary condition and equals 1 if a specified heat flux is useful. $H_{0,D}$ is the surface heat flux coefficient which combines the radiation and convection effects (see Equation 2.9) which can be set to 0 if no radiation and convection is supposed. Due to the power dissipation the heat flux is expressed as $p_{0,D}\left(x, y, \tau\right)$. This generality of the boundary condition allows the vertical matching of arbitrary subvolumes by matching the temperature and heat flux at their discretised interfaces and thus building a more complex structure.

2.2.2.2 Numerical Thermal Model

The solving of the heat diffusion problem in Equation 2.10 by means of numerical thermal models are widely used in practice. They are based on dividing the time and the spatial domain into finite points and replacing the differential equation by algebraic equations at those discrete points. The numerical methods typical used in the thermal analysis of integrated circuits are the Finite Difference Method (FDM), the Finite Element Method (FEM) and the Boundary Element Method (BEM) [30].

2.2.2.2.1 Finite Difference Method

The Finite Difference Method approximates a differential equation by replacing its deri-

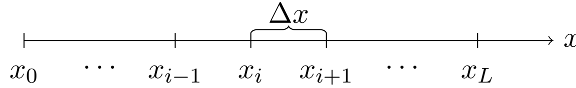


Figure 2.10: The variable x is subdivided in discrete points, $x_i = i\Delta x$ from $i = 0$ to L

vatives with differences terms at discrete points (see Figure 2.10). Such difference formulations are obtained from the **finite difference form** of the the first derivative with respect to one independent variable x. This is done by truncating the Taylor series expansion of the function $T(x)$ about a point x_i after the first derivative term as described in [8] and [25].

$$T\left(x_i + \Delta x\right) = T(x_i) + \Delta x \left.\frac{\partial T\left(x\right)}{\partial x}\right|_{x=x_i} + \frac{1}{2}\Delta x^2 \left.\frac{\partial^2 T\left(x\right)}{\partial x^2}\right|_{x=x_i} + \dots \tag{2.16}$$

After rearranging, the first order forward difference equation is given by

$$\left.\frac{\partial T\left(x\right)}{\partial x}\right|_{x=x_i} = \frac{T\left(x_i + \Delta x\right) - T(x_i)}{\Delta x} = \frac{T_{i+1} - T_i}{\Delta x} \tag{2.17}$$

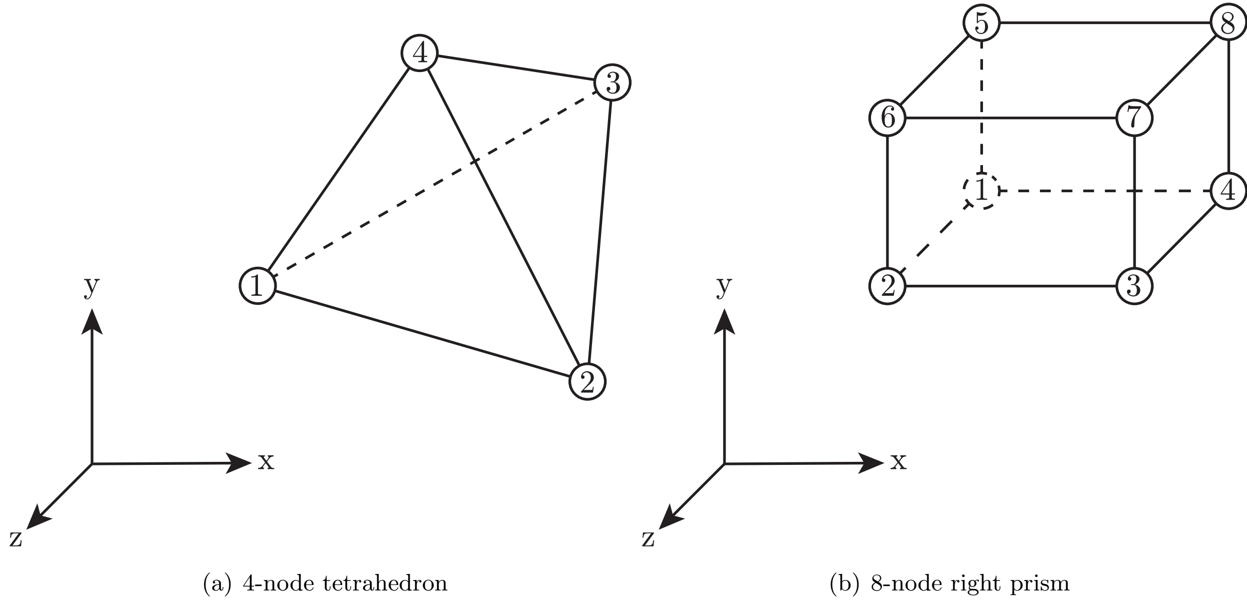
The above function can be combined with its backward difference variant to form the second order central difference equation. The backward difference is easily obtained by using the increment $-\Delta x$ instead of Δx in Equation 2.16 [25]. Finally the second order derivative with respect to x can be approximated by its central difference expressed by

$$\left.\frac{\partial^2 T\left(x\right)}{\partial x^2}\right|_{x=x_i} = \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} \tag{2.18}$$

In the case of the heat diffusion equation 2.10 a PDE involving partial derivatives with respect to three variables in space, x, y, z and one in time t is treated. There exist two basic approaches in replacing them.

A first scheme replacing all derivatives resulting in a recurrence equation also referred to as full discretisation (FD). For this purpose the second order central difference introduced above is usually used for replacing the spatial independent variables [18]. Recktenwald et al. [25] introduce three schemes for a 1D heat equation using the first order forward and the backward difference equation for the time discretisation in combination with the second order central difference.

The second approach called the semi discretisation (SD) is a common technique used in practice applying the finite discrete scheme only to the spatial derivatives. As a result a set of coupled ordinary differential equation (ODE) is obtained which are easily solved by standard ODE solver. Both techniques finally offer a spatial and time discretised solution. However, the SD variant is easier to derive and only half the computational effort that can be undone by the wrong choice of the solver for the equation [25].



(a) 4-node tetrahedron

(b) 8-node right prism

Figure 2.11: Shapes of three-dimensional finite elements [23, modified].

2.2.2.2.2 Finite Element Method

The finite element method (FEM) subdivide the geometrical domain Ω into a finite number of independent sub-domains namely finite elements Ω^e. In the three-dimensional case several shape types for a finite element such as a tetrahedron (see Figure 2.11(a)) and a right prism (see Figure 2.11(b)) can be used. Over each element the governing equation is approximated by the linear combination of interpolating functions usually polynomial equations and nodal points of form [26]

$$T(x, y, u, t) \approx T_N^{(e)}(x, y, z, t) = \sum_{j=1}^{N} T_j^{(e)}(t)\, \phi_j^e(x, y, z) \tag{2.19}$$

where $T_j^{(e)}(t)$ is the value of temperature $T_N^{(e)}(x, y, z, t)$ at the jth nodal point of the element e at time t and $\phi_j^e(x, y, z)$ are the interpolating functions in spatial coordinates only. To satisfy the three-dimensional time dependent problem the equation 2.19 is described as space-time decoupled formulation. Similar to the semi discretisation of the finite difference scheme the spatial approximation is performed treating the problem as a static one while keeping all the time-dependent terms in the formulation resulting in a set of ODEs.

To perform the spatial approximation the interpolating polynomials are chosen in such a way that the 2.19 is a complete polynomial which satisfies the boundary conditions and is continuous over the element and differentiable. To determine the unknown nodal points of the element the approximation equation has to satisfy the governing equation in a weighted-integral sense. That is, substituting the approximation function into the governing equation usually yields in a difference (residual) afflicted formulation. However, by multiplying this formulation with a weighted function, integrating them over Ω^e and

imposing the boundary conditions the residual is minimised. This results in a system of N-algebraic ordinary differential equations, which can be written in matrix form as

$$\left[M^{(e)}\right]\left\{\dot{T}^{(e)}\right\}+\left[K^{(e)}\right]\left\{T^{(e)}\right\}=\left\{f^{(e)}\right\}+\left\{Q^{(e)}\right\} \tag{2.20}$$

where $K^{(e)}$ and $\left[M^{(e)}\right]$ are the coefficient matrix, the column vectors $\{f^e\}$ and $\left\{Q^{(e)}\right\}$ are the source vectors, whereas $f^{(e)}$ refers to heat generation and $Q^{(e)}$ to heat. The equation 2.20 includes $2N$ unknowns, the temperature vector $T^{(e)}$ and the heat vector $Q^{(e)}$. By assembling the element equations of the whole domain Ω some of the missing N conditions are provided by the balance of the heat variables $Q_i^{(e)}$ at common nodes while the remainder are obtained by the boundary conditions.

To satisfy the time dependent behaviour of the PDE the time variable is usually approximated by well known schemes replacing the derivative with difference formulas. As a result a solution that is continuous in space but not in time is obtained.

2.2.2.2.3 Boundary Element Method

In the Boundary Element Method (BEM) approach the governing differential equation is translated into an integral equation applicable over the boundary by means of a Green function (fundamental solution) and numerically integrated over the boundary, that is segmented into several boundary elements [6]. This domain transformation reduces the three dimensional in a two dimensional problem and thus a minor number of elements are needed for the calculation. The Green function is the response of a field region (temperature distribution) to a load point (power source) located at a source region and in case of several power sources the responses at a field point are summed up according to the superposition principle [30].

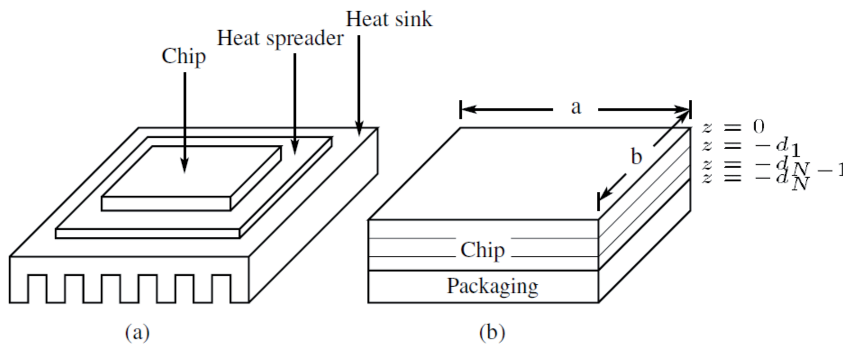


Figure 2.12: VLSI chip with packaging [31].

Zhan et al. [31] present a cell level full-chip steady state simulation approach based on the BEM using the Green function, which generates a piece-wise constant temperature map by taking a piece-wise power density map as power sources. In Figure 2.12 the schematic of a VLSI chip, the problem formulation is based on, is illustrated. The temperature distribution on the field region is established as a convolution of the green function with the power distribution function. By transferring the convolution integral in the frequency

domain by the discrete cosine transformation (DCT), the frequency representation of the temperature map is obtained by performing simple multiplications. By performing an inverse discrete cosine function, the temperature map is finally gained. Together with a look up table approach, the authors could accelerate the calculation compared to previous green function based methods. However, the approach does not support the transient thermal analysis.

2.2.2.3 RC Network

Assume a wall of thickness L with an area of A, as illustrated in Figure 2.6(a). By considering a steady heat conduction through the wall with the temperatures at the one side $x = 0$ to be T_1 and on the other side x = L to be T_L , the one dimensional Fourier's law of heat conduction can be integrated and rearranged as

$$\dot{Q}_{cond} = \frac{T_1 - T_2}{R_{th}} \, [W] \tag{2.21}$$

where the term

$$R_{th} = \frac{kA}{L} \, [KW^{-1}] \tag{2.22}$$

is called the conduction resistance. In other words the heat flow $\dot{Q}_{cond}$ across the thermal resistance R_{th} and causes a temperature difference $T_1 - T_L$ [K] analogue to an electrical current flow causes a voltage difference at an electrical resistance [8].

Considering an transient heat conduction transfer through the same wall, the three-dimensional heat conduction equation is written into its one-dimensional form and integrated over the same variable x, leading to equation

$$(\rho c_p A L) \frac{dT(t)}{dt} = kA \frac{\Delta T(t)}{L} \cdot \dot{Q} \tag{2.23}$$

where $(\rho c_p A L)$ is defined as the thermal capacitance C_{th} due to the analogy of the the left side of the formulation with the definition of the current flow through and electrical capacitance [15]. The thermal capacitance defines the capability of a structure to absorb heat, similar to an electrical capacitor that accumulates electrical charges. In Table 2.2 the duality between the thermal and electrical quantities are summarised.

Thermal quantity	unit	Electrical quantity	unit
P, Heat flow, power	W	I, Current flow	A
T, Temperature difference	K	V, Voltage	V
R_{th}, Thermal resistance	K/W	R, Electrical resistance	Ω=V/A
C_{th}, Thermal mass, capacitance	J/K	C, Electrical capacitance	F=A/V
$\tau_{th} = R_{th} \cdot C_{th}$, Thermal RC constant	s	$\tau = R \cdot C$, Electrical RC constant	s

Table 2.2: Duality between thermal and electrical quantities [27].

Based on this duality, Skadron et al [27] establish a compact thermal RC circuit at the micro architectural level that calculates the thermal resistances and capacitance based on dimensions and physical properties of the used materials, implemented as open software tool named Hotspot. The R and C values of the model represent the heat flow path

among the functional units of the chip (Caches, Integer Unit, HW Multiplier etc.) as well as from each unit to the thermal packaging towards the ambient air. The thermal model provides two modes of granularity for the die, the block and the grid based mode. The block mode uses the segmentation defined by the position and dimension of the functional units, whereas the grid mode parts the die in certain number of identically grid cells. This enables the finding of a tradeoff between accuracy and computational effort based on the users needs. In Figure 2.13 an example of HOTSPOT's compact thermal model with 3x3 grid cells for the die is shown. The solution of the steady state temperatures is based on

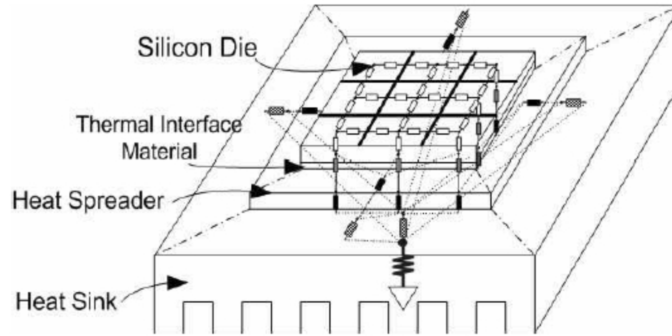


Figure 2.13: Example of the compact thermal model.

the nodal analysis of an electric network by arranging and solving a matrix system of the form

$$T_{steady} = B^{-1} \cdot P \; [K] \tag{2.24}$$

where the conductance matrix B describes he resistance network and the power vector P holds the average power dissipation of each unit. For the transient solution the differential equation of the form

$$\Delta T + A^{-1} \cdot B \cdot T \cdot \Delta t = A^{-1} \cdot P \cdot \Delta t \tag{2.25}$$

is solved by using a fourth order Runge-Kutta (RK4) method.

2.2.2.4 RC Network by Reduction

An alternative approach to a RC network based on physical geometries and material properties is the reduction of a high dimensional thermal model into a model of reduced order with acceptable accuracy. Bohm et al. [7] propose a formal reduction method that uses moment matching via the Arnoldi algorithm to iteratively reduce a high dimensional model, created with ANSYS, to reduced models of maximal order specified by the user. The gained matrices are then further used for transient simulation by means of the finite difference method.

In [1] the DELPHI approach is presented which generates a thermal resistance network, as seen in Figure 2.14, in various steps from detailed model simulations. The gained resistances are mathematical constructs and thus the resistance between two nodes does not necessary correspond to the physical resistance in the detailed model.

The thermal resistance network consists of a limited number of either internal or surface nodes, whose location and number have to be predefined. Each surface node is allocated

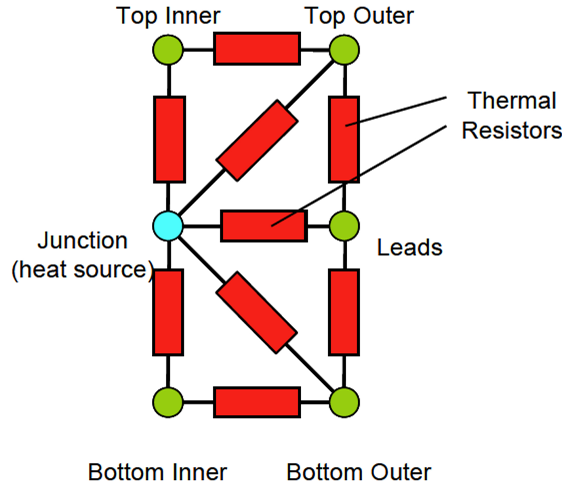


Figure 2.14: The DELPHI compact model.

to a physical region of the actual package (detailed model) surface having an one-to-one association. That is, the surface nodes have to interact with the ambient as identically as to the detailed model. An internal node however may or my not correspond to a physical region within the package, but the node temperature has only a physical meaning if it is associated to an area within the package. Moreover an optimisation process over a boundary condition set (training set) is performed. By exercising the gained compact model against the test set the accuracy can finally be determined by means of an error estimate. This enables a compact thermal model that is able to predict junction as well as heat fluxes of the surfaces nodes for a wide spectrum of environments. Although the accuracy is high and the computational effort of the resulting compact thermal model is low, the approach is limited to the packages with a single junction temperature of interest.

All these reduction model have in common to accurately characterise existing packages, but can not be used for testing new materials or package configurations without performing the whole reduction process again.

2.3 Chip Card

2.3.1 Introduction

A chip card is an electronic device integrated in a credit card sized plastic card able to process and save data. The data and energy exchange with the reader take place either contact based by means of galvanic connection or contactless using the electromagnetic field. Depending on the their internal configuration both systems can be classified into memory and microprocessor cards. Memory cards using sequential logic and are able to process simply security algorithm whereas microprocessor cards are capable in using complex encryption and authentication algorithms by an additional cryptographic coprocessor. The requirement of fast and simply calculation of complex cryptographic algorithms in the typical payment market of *contact smart cards* combined with the benefits of a user friendly system with short transaction times of *contact less only systems* led to the deve-

lopment of the *dual interface card* providing both interfaces.

This work mainly focuses on the **contactless proximity-coupling smart card** described in the ISO/IEC standard 14444 and the **dual interface card** that is covered as part of ISO 14443 and ISO 7816. Both systems belongs to RFID Systems (Radio Frequency Identification) at 13.56 MHz operating frequency at the range of 7 - 15 cm.

The raise in computational effort leads to a higher power consumption and hence a larger amount of heat generation.

RFID systems mainly using inductive coupling providing the transponder - the chip card - with more energy than necessary for processing operations. This excess energy is dissipated by an active resistance in the form of heat [13]. In the following we deal with the basic principle of RFID systems required for estimating the power dissipation of the microchip.

2.3.2 Transponder Overview

The transponder consists of the antenna coil and silicon chip separated into the analogue front end and the digital controller.

The **analogue front end** forms the interface between the digital circuit and the coupled EM field and provides the system prerequisites for the operation of the digital core. The main components of a typical RF frontend are [32]:

- Rectifier
- Voltage regulator (protector)
- Clock extraction
- Demodulator
- Modulator for load modulation
- Power on Reset

The antenna coil of the transponder is penetrated by an electromagnetic alternating field inducing a voltage u_i. A low loss bridge rectifier generates a rectifed signal that serves as power supply for both the analogue frontend and the digital circuits [13]. The voltage regulator (protector) usually consist of two separate circuits [32]. One regulates the operating voltage to the required voltage level of the digital core (e.g Shunt regulator). Since the RF input voltage can reach high values in the same magnitude of the electromagentic field strength, the second circuit structure limits the the supply voltage to a non critical level (e.g serial regulator). The digital core of todays modern microprocessor cards typical consists of:

- CPU
- Cryptographic Coprocessor
- Decoder/Encoder
- Memory RAM/EEPROM/ROM

- I/O ports
- Error recognition (CRC)

Considering the power consumption of both the analogue and the digital circuit the regulator/protector circuits, the CPU and the Cryptographic Coprocessor are the main factors of the transponders power consumption and thus representing the main sources of heat generation.

2.3.3 Inductive Coupling

For the analysis of the electrical characteristics of the transponder chip the complex circuit can be described with the use of an equivalent circuit diagram (ECD), as shown in Figure 2.15.

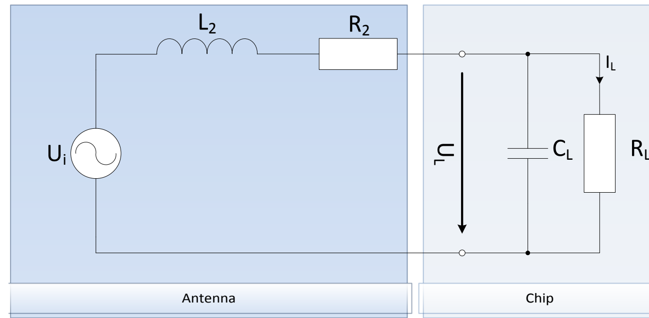


Figure 2.15: Transponder equivalent circuit diagram (ECD).

The antenna characterised by the inductivity L_2 and the ohmic loss resistance R_2 together with the input impedance of the chip $R_L||C_L$ form a serial resonant circuit. The capacitance C_L describes the parasitic input capacitance of the rectifier and the load resistance R_L corresponds to the power consumption of the chip and the operation point of the voltage regulator. The voltage induced by the antenna coil can be expressed by an AC voltage source u_i, in series with the inductivity and the ohmic loss.

2.3.3.1 Inductive Voltage u_i

The reader of the RFID system produces an alternating electromagnetic field that penetrates the antenna coil, as illustrated in Figure 2.16. According to Faraday's law of induction this change in the magnetic flux Φ generates an electric field strength E_i. Since the antenna coil acts as a conductor loop, a voltage, called induced voltage, is build up at the end of the loop. The induced voltage u_i corresponds with the path integral of the field strength E_i which is proportional to the rate of change of the magnetic flux [13]

$$u_i = \oint E_i ds = N \frac{d\Phi}{dt} \tag{2.26}$$

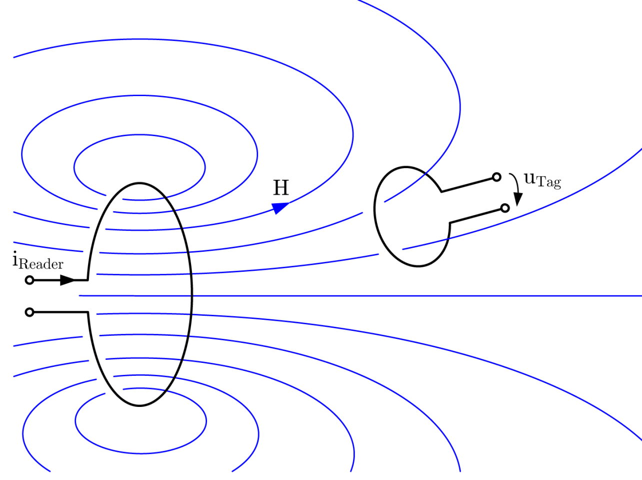


Figure 2.16: Inductive coupling between reader and tag.

where N is the number of windings of the antenna coil. Assuming a homogeneous, sinusoidal magnetic field H in air (μ_0)

$$H = H_0 \cdot sin(\omega t) \tag{2.27}$$

and replacing the magnetic flux Φ passing through a loop with the constant cross-sectional area A with

$$\Phi = B \cdot A = \mu_0 \cdot H \cdot A \tag{2.28}$$

we get

$$u_i = N \cdot \mu_0 \cdot A \cdot \frac{dH}{dt} = N \cdot \mu_0 \cdot A \cdot H_0 \cdot \omega \cdot cos(\omega t) \tag{2.29}$$

or

$$u_i = N \cdot \mu_0 \cdot A \cdot H_{eff} \cdot \omega \tag{2.30}$$

where H_{eff} is the effective field strength of a sinusoidal magnetic field.

2.3.3.2 Resonance Frequency f_{res}

The resonance frequency is the frequency point at which the impedance of the circuit in Figure 2.15 becomes real. By setting the imaginary part of the impedance equal zero

$$Im\left\{R_2 + j\omega L_2 + \frac{1}{j\omega C_L + \frac{1}{R_L}}\right\} = 0 \tag{2.31}$$

the resonance frequency gets

$$f_{res} = \frac{1}{2\pi}\sqrt{\frac{1}{L_2 C_L} - \frac{1}{C_L^2 R_L^2}} \tag{2.32}$$

However, in practice the resonance frequency f_{res} of the transponder is chosen higher than the system frequency of the RFID system ($f_{sys} = 13.56MHz$) to counter detuning effects.

2.3.3.3 Quality Factor Q_0

The maximum of the voltage and current step-up at the resonance frequency of the circuit is affected by the quality Q factor. Q can be derived from the ratio of the resonance frequency and the bandwidth at 3 db relative to its centre frequency $Q = f_{res}/B$ by determining the corner frequencies ($B = \omega'' - \omega'$). Consequently the unloaded quality factor Q_0 of the antenna coil becomes

$$Q_0 = \frac{\omega_0 L_2}{R_2} \text{ with } \omega_0 = \frac{1}{\sqrt{L_2 C_L}} \quad (2.33)$$

Chapter 3

Design

3.1 Overview

In this chapter the design of the thermal estimator architecture using the power emulation approach to obtain run-time power estimates is introduced. The architecture mainly consists of the core thermal simulator, the package model, the analogue power model of the resonant circuit, the validation process and the visualisation of the results, as seen in Figure 3.1.

The power consumption of the digital integrated circuit is estimated by the power emulation unit of the functional test system while running a particular benchmark (e.g. Payment application) program. The result is provided as current profile to the analogue power model which determines the analogue and digital power consumption. The obtained power information is taken together as one power profile before it is made available to the core thermal simulator. The package model forms the package into a three-dimensional rectilinear grid and provides the resistance and conductances values of each cube with their linking information. Together with the power information the thermal simulator generates the transient thermal profile and the steady state temperatures, respectively. Finally the results are visualised for evaluation and optional verified if the required data are available. Along the design requirements the components of the architecture are discussed in detail in the next sections.

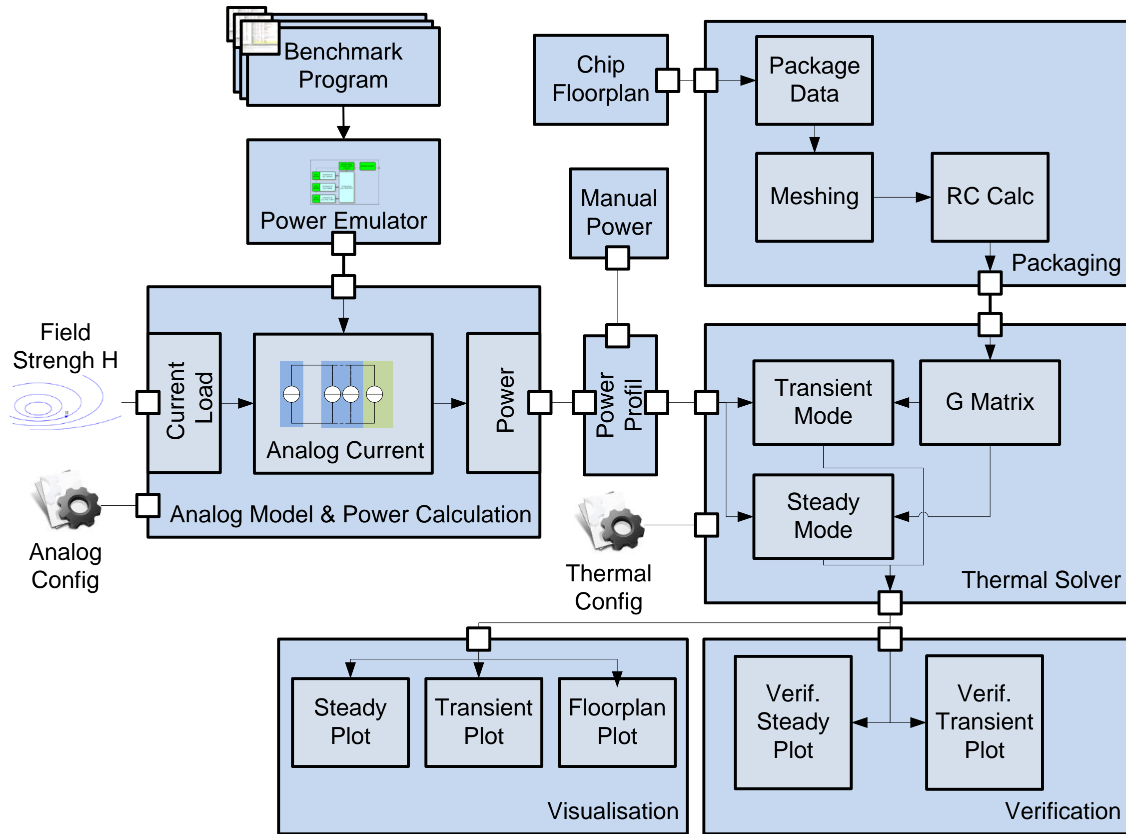


Figure 3.1: Thermal Estimation Architecture.

3.2 Requirements

- The thermal simulator should work on both Windows and Unix.
- The thermal network should be dynamical build to evaluate different package configurations with different material properties.
- The floorplan of the chip should be supplied separately from the package to test different package chip configurations.
- Existing power profiles gained from the power emulator and analogue power model should be able to process.
- The definition of multiple saving intervals in transient mode should be possible.
- The thermal simulator should be able to perform both the transient and a steady state simulation of the system.
- The cooling of the system by means of convection and radiation must be possible to simulate.

3.3 Power Model

The required power information for particular units of the μChip can be applied to the thermal simulator in two ways. The primary approach uses the power profile of the digital circuit gained from the power emulator determining the analogue power consumption and providing an overall power profile to the simulator. In addition, a manual mode allows the user to define a simply power profile by hand. In the following, both designs are presented.

3.3.1 The Analogue Power Model

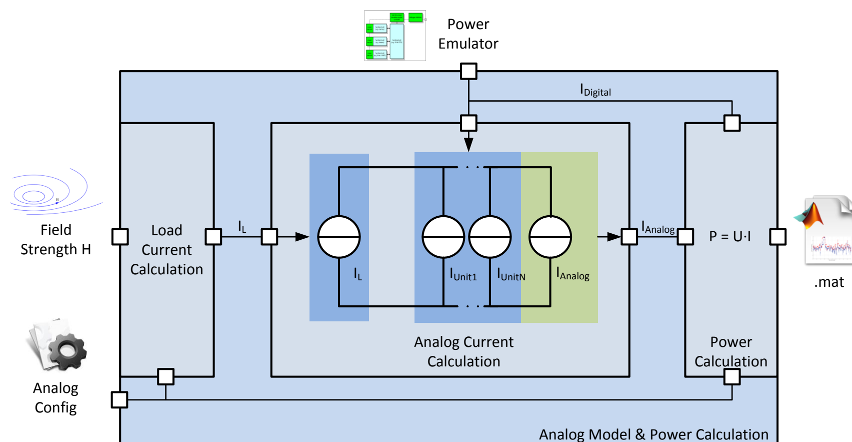


Figure 3.2: Design of the analogue power model.

In Figure 3.2 the design of the analogue power model together with the overall power calculation is illustrated. As described in 2.3.2, the transponder chip consists of the analogue front end and the digital controller. Since the power emulator only supplies the power profile for the digital circuit, the dissipating power along the analogue frontend has to be determined. Therefore the entire μChip is treated as an electric network of one current source and an arbitrary number of current sinks. To determine the analogue current the load current of the equivalent circuit diagram shown in Figure 2.15 is calculated. An analogue config file is built as a structure MAT file containing the analogue parameters for the inductive coupled network, together with the field strength H are provided as input for the load current calculation. The resulting load current I_L acts as source for the analogue current calculation. Together with the current profile gained from the power emulator, the current network is solved, and the so obtained analogue current is used for the power calculation. The supply voltage of both the analogue front end and the digital circuit required for the power calculation are defined in the analogue config file. The resulting power profiles for each unit are separately stored as power vectors in MAT file named after the identifier of the chip's floorplan. The power vectors are finally taken together as power matrix before provided as input to the thermal solver.

3.3.2 Manual Power Profile

Building the power profile by hand requires the definition of three cell vectors as pictured in Figure 3.3. Each cell of the UnitName vector contains the name of one unit as string. The Power and the TimeDuration vector holds a double array per cell. For each unit, one double array holds several power values, which last defined time periods, stored in a corresponding double array.

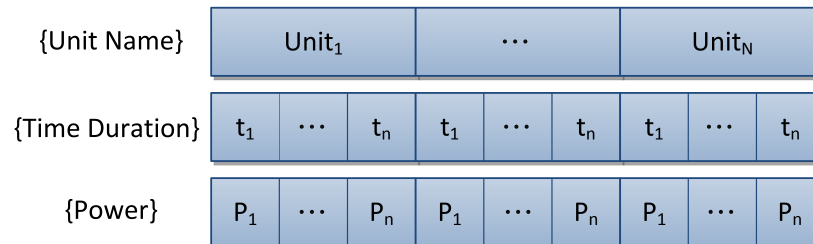


Figure 3.3: Structure of the manual power profile.

3.4 Package Model

3.4.1 Introduction

An integrated circuit chip is usually assembled on a substrate by an assembly process, encapsulated with resin and embedded into a package. In the case of smart cards the silicon chip is attached to the substrate with glue, encapsulated with epoxy resin and embedded within the card. The layout of the substrate as the used materials are manifold.

In the field of contact-less applications the substrate consist of antenna pads and/or part of the entire antenna in different forms. Contact types as well as dual interface cards need metal contacts to get in contact with a reader. Therefore a packaging model is necessary that is not limited to various packaging build-ups and used materials.

As a first step in mapping the real package into a computable model it is necessary to define the grid type including the coordinate system. The grid describes the fragmentation of the spatial volume of the the actual system into elements of certain form. Based on the computation methodology underlying the thermal simulator, the elements are of rectangular shape and adjacent to each other with just one side in contact. As a consequence the package can be defined as a build up of layers consisting of the same element configuration. In the following the design of the package model is discussed.

3.4.2 Package Data

The definition of the package implies that each component of the package can be described as a cuboid with a rectangular base and thickness composed of a certain material. To specify a component and its dimension a cartesian coordinate system in the 2D plain has to be established. Therefore the coordinate points are defined as real number pairs and the origin of the coordinate axis is set to the coordinates (0,0). This enables the description of the rectangular base as a (x,y) pair of the left bottom vertex with two edges of certain length declared in meter. The partition of the μChip is separately defined in a floorplan file using the same coordinate system so it can be included into the package data easily.

3.4.3 Meshing

The coordinates of the components including the μChip together with the length of the edges span the base grid used in the meshing function as minimal segmentation. To raise the grid density it is possible to define an additional fragmentation of an area and/or one or more elements.

3.4.4 Layer Building and RC Calculation

The resulting grid provided by the meshing function is used to fragmentate each layer which can been seen in Figure 3.4 where a cut-out of a possible packaging layout is illustrated. Each layer consists of parts of one ore more components with different material properties. To build a particular layer the components that are part of the layer have to be declared and a primary component has to be chosen defining the layer thickness. In the case of the package in Figure 3.4 the first layer contains part of the Glob Top and PVC Card - with the Glob Top defined as primary component. The declaration of the components together with the grid are then used to determine the resistances and capacitances of the layer elements. The resulting layer together with their built-up configuration are provided as output to the thermal solver. From this point on, the cubic room that is filled by the package is denoted as the model room with the lower left corner as the origin.

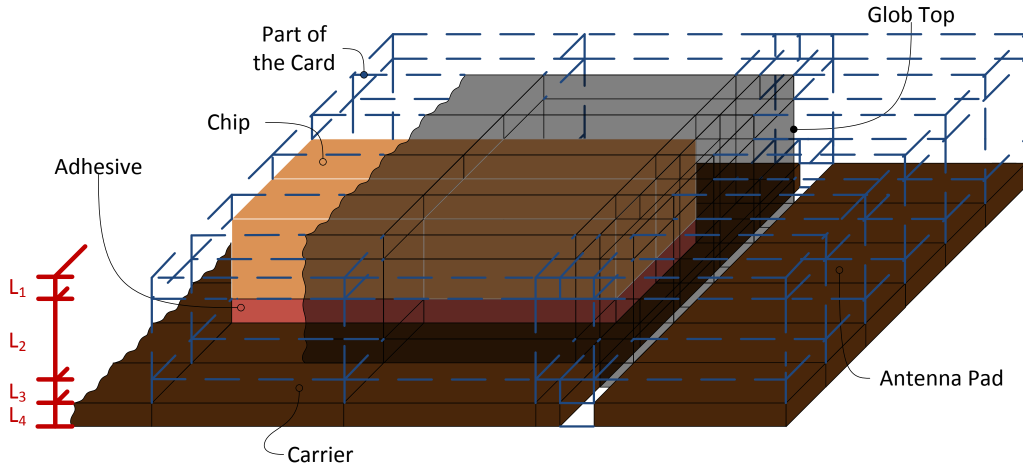


Figure 3.4: Thermal Estimation Architecture.

3.5 Thermal Simulator

The thermal solver determines the dynamical thermal behaviour of the package as its temperature values when staying in thermal equilibrium. Therefore the power information gained from the power model as the R and C values from the package model are formed to a thermal network. The building of the thermal network and the applied methodology in solving the network are discussed in the next sections.

3.5.1 Power Information Matching

The power estimates of the μChip gained from the power model in section 3.3 are assigned to the corresponding units of the floorplan before the meshing algorithm has been executed and thus no information of the final mesh resolution is available. For matching the power information according the new resolution of the μChip floorplan an identification of the power related elements is necessary. Therefore, the corresponding elements are flagged by the package model during the calculation of the R and C values and the resulting IDs are then provided to the thermal solver. By knowing the power related elements the power information can easily matched by an area based method.

3.5.2 Components of the Thermal Network

The thermal network consists of nodes representing the temperature points linked with each other by thermal resistances and one capacitance per node towards the ambient temperature in order to model the transient behaviour. In addition nodes part of the μChip are connected with current sources delivering the power information to the circuit. By using the resistances from each element of the package model together with the built-up configuration a connection matrix can be formed containing the conductances between

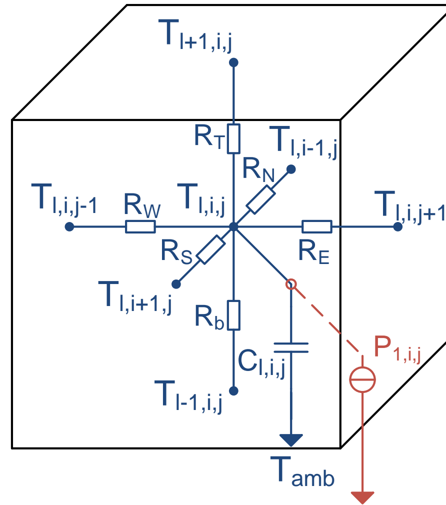


Figure 3.5: The thermal core of the transient simulation

each element, whereas one element or cell is treated as one node of the thermal network. To determine the heat flow to the environment the elements or rather the corresponding resistances in contact with the ambient have to be identified.

3.5.3 Convection and Radiation Heat Transfer

The cooling of the package is assumed to be done by natural convection and radiation which is applicable in the case of a smart card. Therefore a combined heat transfer coefficient as described in Equation 2.9 is defined as a function of the surface and ambient temperature. While determining the radiation heat transfer coefficient is straightforward, obtaining the natural heat transfer correlations is usually based on experimental studies. However, when lesser accuracy is acceptable, a constant value can be used producing less computational overhead. To enable the simulation of various cooling mechanism the overall heat transfer coefficient is provided by a routine that can easily be exchanged or adapted.

3.5.4 Transient Simulation

In performing the analysis of the transient behaviour of the package the time dependent temperatures across the thermal capacitors of each one node has to be determined. By applying the Kirchhoff's current law to each node of the network each node can be described by one ordinary differential equation (ODE). In Figure 3.5 the model of one node of the thermal network is shown. By using the adjacent nodes and the linked thermal resistances in each direction together with the capacity and the optional power source towards the ambient temperature the solution function of the ODE for the node can be determined. With the resulting solution function the thermal behaviour of each node of the thermal network can be described at each point in time by taking the temperatures of the previous time step of each adjacent node. In addition the heat transfer coefficient for

each resistance towards the ambient temperature has to be updated. By invoking the the solution function for each node of the thermal network the thermal profile of the package is obtained. Since the solution function only needs the temperature of the previous time step the storage of the temperature values needed for the analysis of the thermal profile is separated from those used by the calculation. Thus, managing the points in time where the temperatures of the profile are saved, can reduce the memory usage.

3.5.5 Steady State Simulation

If the transient analysis is running long enough, the package reaches its thermal equilibrium or at least converge to its steady state. However the required amount of time until the result of the transient simulation provides the steady temperature can be huge. Especially in testing the influence of new cooling mechanism, new materials or even new package components, a fast steady state estimate can help in saving development time. Therefore a power vector is constructed including the heat flow towards the ambient and the connection matrix is extended to fit as conductance matrix for the well known nodal analysis. To last the time dependency of the resistance towards the ambient temperature, an iterative approach executes the steady state calculation as often as the differences of the current heat transfer coefficients and its previous ones is higher than a certain limit.

3.6 Visualisation and Validation

After a thermal simulation is finished, the results are visually presented depending on the simulation type. When a transient analysis is performed the main interest lies in the progress of the temperature of one or more particular nodes. Therefore a time-temperature graph is able to present the thermal data of any point according to its position at one or more desired layers. The positions of several coordinates can be declared by their names (e.g: CORE, DIF, ...) or their x-y coordinates. In contrast to a time depending graph the result gained from the steady state simulation is presented as grid with temperature dependent colour gradient. Besides defining a layer of interest the display region can be narrowed to a field like the area covered from the muChip in a particular layer. The steady state visualisation can also be used in case of a transient simulation to display the temperature gradient at a desired point in time. In addition, a visualisation of the grid without any temperature information can be invoked so that it enables the user to verify the meshing before any simulation is performed.

3.6.1 Validation

The validation of the resulting temperature data is an optional task and requires the presence of thermal reference data provided as an Excel-file or MAT-file. The temperature points to be verified are declared as x-y coordinates and matched towards the reference points by their unit names. The result of the validation process is then visualised by a bar graph in case of a steady state simulation or as a time-temperature graph when a transient simulation was performed. The time-temperature graph contains the temperature progress over time of the simulation data together with the corresponding references. For a better

visualisation of the simulation accuracy an additional graph shows the absolute deviation from the reference data.

Chapter 4

Implementation

4.1 Overview

In this chapter the implementation of the RC circuit based thermal model suitable for a security controller in a smart card system with a radio frequency interface is presented. An overview of the subroutines and its calling relationships are shown in Figure 4.1. The main routine processes the user input obtained from a graphical user interface and forms as appropriate the required data by invoking the corresponding subroutines of the power estimation and package creation before starting the steady or transient simulation routines. In addition, the representation of the thermal results are initiated as well as the optional validation process and the saving of all relevant data for a later review are executed. The details of the implementation follow in the next sections.

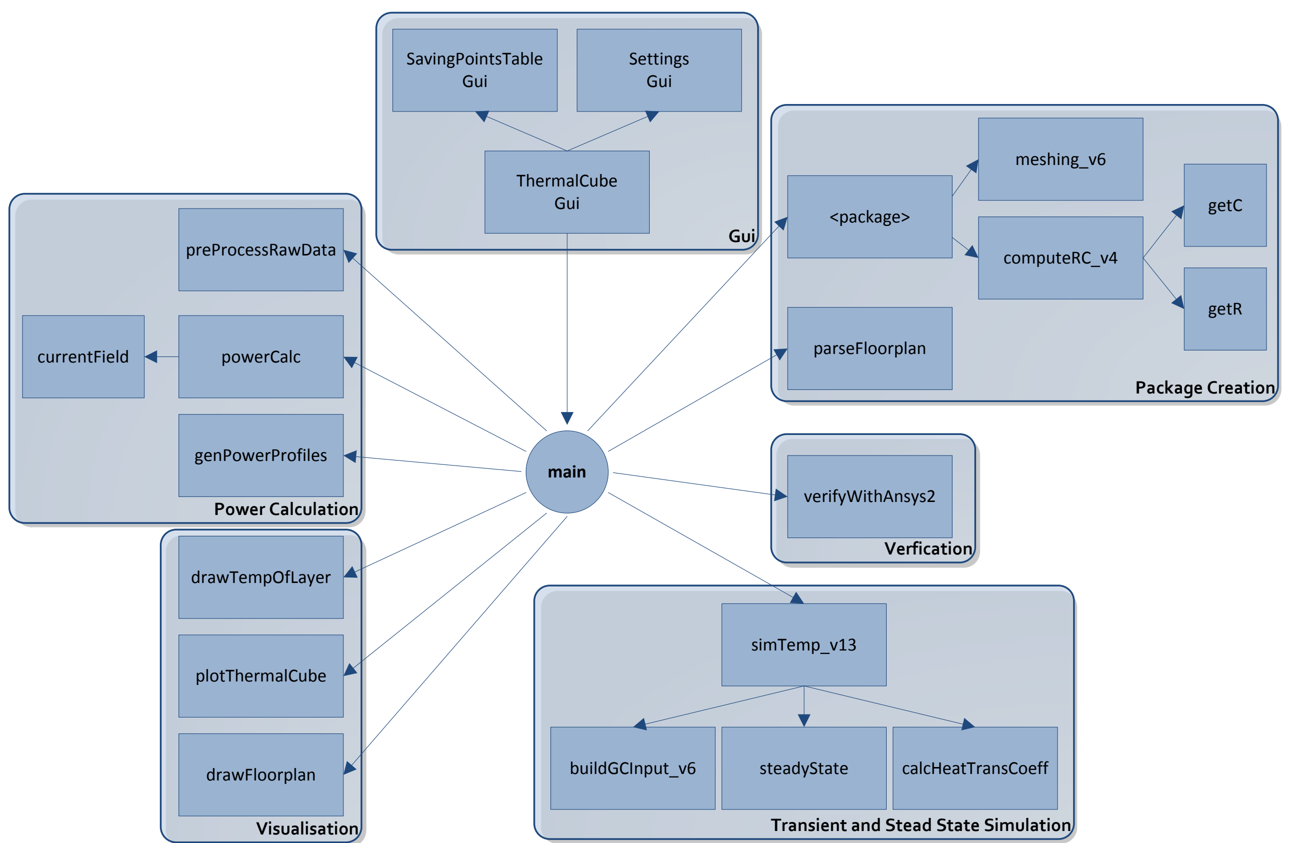


Figure 4.1: Implementation overview / Calling Graph.

4.2 The Package Model

4.2.1 Introduction

Each package is implemented as a function taking the floorplan of a suitable μChip as input. Within the function the dimension of each component, the position in the model room as the material properties are defined. Using that information a grid is generated by invoking the mesh sub function dividing the model room in elements of certain form. With the resulting grid the thermal resistances and the conductance of each element are calculated by a corresponding sub function invoked for each layer. Finally the package function provides the R-C representation of the package elements including their spatial configuration as output. In the following the implementation of the package function is presented in detail.

4.2.2 R-C Representation

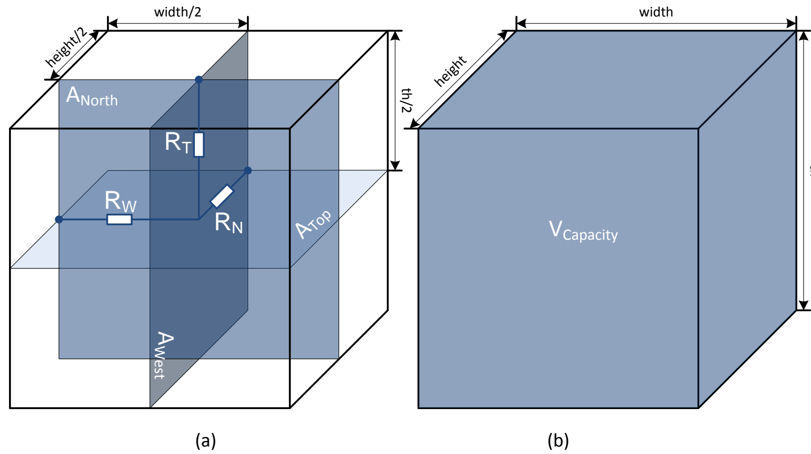


Figure 4.2: R-C representation of one element.

Each element of the package consists of six thermal resistances arranged around the midpoint of the cuboid, parallel to the lateral surfaces. Due to the given symmetry only three of them has to be determined. Similar to the RC network in Chapter 2.2.2.3 the thermal resistance is implemented based on the following equation

$$R = \frac{th}{k \cdot A} \left[\frac{K}{W}\right] \tag{4.1}$$

whereas the thickness th and the cross-sectional area A changes depending on the direction of the resistance, as illustrated in Figure 4.2, while the thermal conductivity k per unit volume remains the same. In addition the thermal capacity of one element is calculated depending on the volume and the material property as given by

$$C = c_v \cdot th \cdot height \cdot width \left[\frac{J}{m^{-3}K}\right] \tag{4.2}$$

where c_v is the thermal capacitance per unit volume.

4.2.3 Floorplan

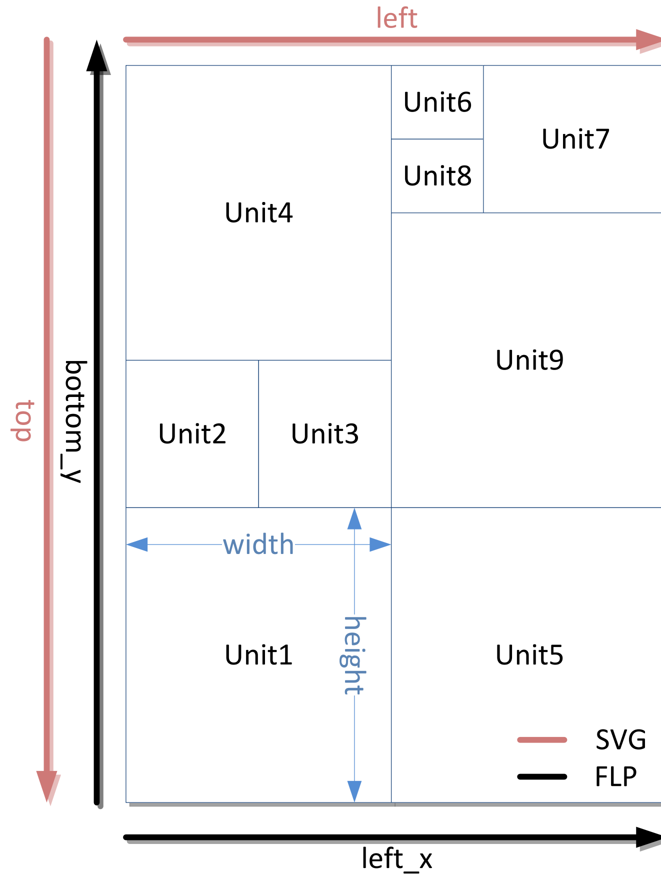


Figure 4.3: Possible floorplan of a μChip and the difference in its definition as FLP or SVG format.

The floorplan usually specifies the partition of the μChip in functional units at micro-architectural level or higher and has to be available either in the scalable vector graphics (SVG) or in the HotSpots floorplan (FLP) format. The units are described as rectangles defined by one corner, the width and the hight, which are adjacent to each other without leaving a gap and not allowed to overlap. In contrast to the FLP format where a rectangular is defined by the position of its left lower corner, the SVG format uses the left upper corner as illustrated in Figure 4.3. Moreover the SVG specification 1.1 (Second Edition) supports various unit measures (pixel px is used in this thermal thesis), thus a conversion factor for converting the chosen measure to the intern unit meter is established such that

$$value[unit] * conversion_factor \widehat{=} value[meter] \tag{4.3}$$

The floorplan in the SVG format is translated into the FLP format first before it is converted into the intern data representation which is presented in the next section.

4.2.4 Package Data

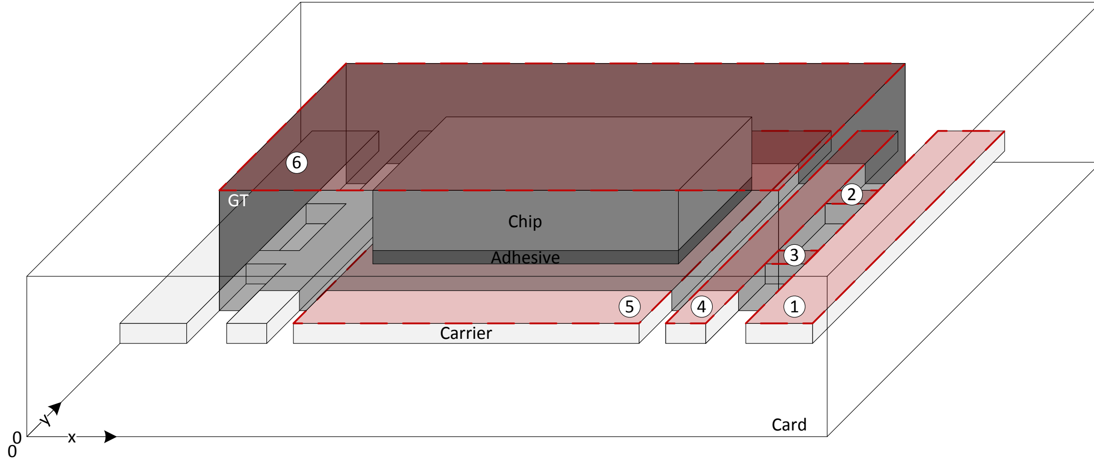


Figure 4.4: Possible smartcard package.

Based on the following example the translation of a possible smartcard packaging into the intern data representation of the thermal simulator will be illustrated. The build up of such a smartcard package, which is composed of different components (chip, adhesive, glob-top (GT), carrier and the pvc card) with certain materials, is shown in Figure 4.4. Each component has to be split up into one or more parts of rectangular shape. In the case of the carrier the right and the left side, both usual antenna pads, are separated into four (1-4) parts while the center one (5)remain the same as well as the glob-top (6), the adhesive and the card body. The rectangular base (width and height) and the position ($\text{left}_\text{x}, \text{bottom}_\text{y}$) together with a unique name for each part are then saved into a *struct* as given in Listing 4.1. The origin of the coordinate system is always the left lower corner of the model box which is in that case identical to the cuboid of the card body.

Listing 4.1: The package structure.

```
package_sa.unit_name(id_1,...,id_N,id_N+1,...,id_K)={'str_1',...,'str_N','str_N+1',...,'str_K'};
package_sa.left_x(id_1,...,id_N,id_N+1,...,id_K)    =[x_1,...,x_N,x_N+1,...,x_K];
package_sa.bottom_y(id_1,...,id_N,id_N+1,...,id_K)  =[y_1,...,y_N,y_N+1,...,y_K];
package_sa.height(id_1,...,id_N,id_N+1,...,id_K)    =[h_1,...,h_N,h_N+1,...,h_K];
package_sa.width(id_1,...,id_N,id_N+1,...,id_K)     =[w_1,...,w_N,w_N+1,...,_K];
package_sa.field(id_1,...,id_N,id_N+1,...,id_K)     =[field_1,...,field_N,field_N+1,...,field_K];
package_sa.k(id_1,...,id_N,id_N+1,...,id_K)         =[k_1,...,k_N,k_N+1,...,k_K];
package_sa.th(id_1,...,id_N,id_N+1,...,id_K)        =[th_1,...,th_N,th_N+1,...,th_K];
package_sa.cv(id_1,...,id_N,id_N+1,...,id_K)        =[cv_1,...,cv_N,cv_N+1,...,cv_K];
package_sa.em(id_1,...,id_N,id_N+1,...,id_K)        =[em_1,...,em_N,em_N+1,...,em_K];
```

The package structure consists of several fields holding arrays, one of *string* and the others of *double* type. The use of arrays in this way makes finding common properties much more easier as well as finding parts of the components within a certain range by the use of logical and relational operations together with logical indexing. By simple invoking `package_sa.left_x` $> x_1$ `&` `package_sa.bottom_y` $> y_1$ a logical array is returned, with elements set to logical 1 where the relation is true. That array can further used as subscript for any of the package arrays. It also makes the code more readable.

The identifier $id_1, ..., id_N$ belong to the units gained from the chip floorplan that are only known during runtime. Therefor the entire identifier $id_1, ..., id_N, id_{N+1}, ..., id_K$ are gained from the *idxInit* function that assigns the id's to user defined variables by invoking `[var_id`$_1$`,var_id`$_{N+1}$`,...,var_id`$_K$`] = idxInit(1:N)`, with N as the number of units at the chip floorplan and K is the total number of all package parts including N. The first variable `var_id`$_1$ belongs to the units of the chip floorplan containing the identifier from 1 to N. As an example, assuming the chip (DIE) contains three units Listing 4.2 shows how the automated generation of the identifier is used.

Listing 4.2: Example of using *idxInit* function for generating identifiers of the component parts.

```
% Content of flp_struct.unit_name = {'unit1', 'unit2','unit3'}
N_units                              = numel(flp_struct.unit_name);
[DIE_id CARRIER_1_id CARRIER_2_id]   = idxInit(1:N_units)
% Content of the variables:
%   DIE_id = [1,2,3]
%   CARRIER_1_id = 4;
%   CARRIER_2_id = 5;

% Defining the names of the component parts
package_sa.unit_name(DIE_id)       = flp_struct.unit_name;
package_sa.unit_name(CARRIER_1_id) = 'carrier1';
package_sa.unit_name(CARRIER_1_id) = 'carrier2';
```

For the implementation of the package as a three dimensional model the segmentation of the model box into several layers is established. Each layer consists of different components and parts of them respectively, thus consisting of various materials. Moreover within a layer no component parts are superimposed upon one another. Consequently one component part within the layer defines the thickness of it, e.g the carrier in Figure 4.4 defines the thickness of one layer. Each part of a component holds the same field id to be able to determine the material affinity of the elements within the layer after the meshing is performed. The field id of the component that defines the thickness of the layer is used to identify the layer itself. The generation of the field id's is similar to the component part ones by calling `[field_id`$_1$`,...,field_id`$_L$`,field_id`$_{L+1}$`,..,field_id`$_K$`] = idxInit(1)` with the integer starting argument 1.

The generated field variables `field_id`$_1$`,...,field_id`$_L$ are also used as layer identifiers whereas `field_id`$_1$ always belongs to the chip (DIE). An additional vector of the same size holds the *string* term of each field id to be able to choose a specific component for a finer grid by its *string* name and for debug purposes. To determine which layer is adjacent to each other the column vector $layer_{order}$ `= [field_id`$_{Top}$`;...;field_id`$_{Bottom}$`]` describes the vertical order of the layers.

4.2.5 Meshing

As discussed in Section 3.4 the model box has to be segmented into elements based on the position and dimension of the components and parts of them respectively such that the elements are adjacent to each other with just one side in contact. The resulting segmentation represents an equal mesh for each layer.

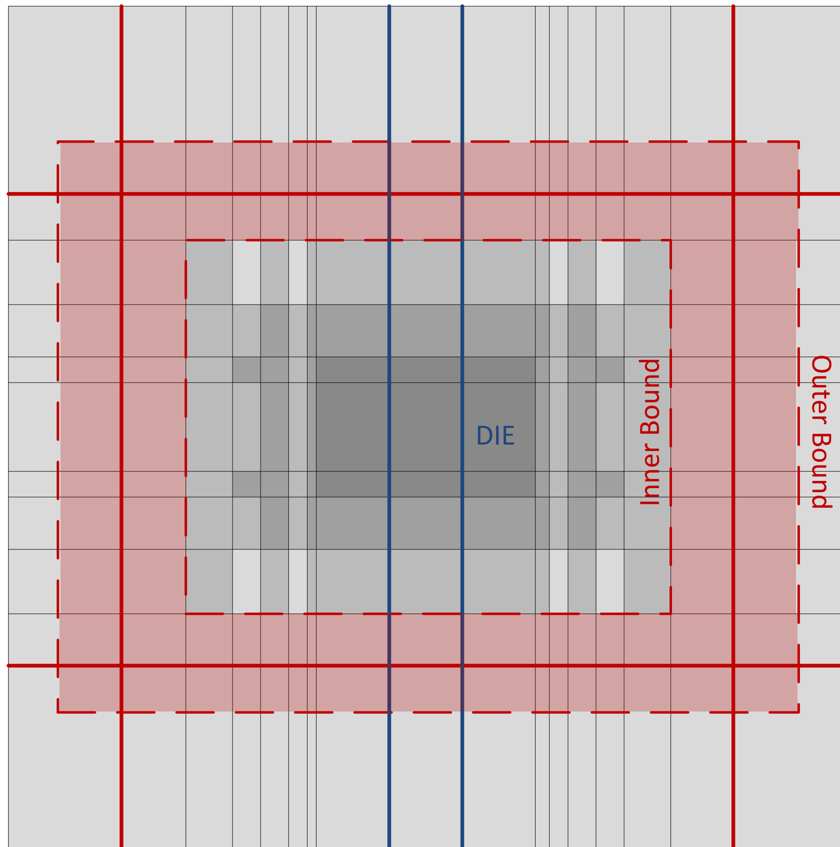


Figure 4.5: The base grid of the smartcard package in Figure 4.4 extended with additional segmentation defined by the user.

The meshing algorithm is composed of three sub algorithm, as described in Algorithm 1, whereas the first and the second step (2) are executed when the user fills the corresponding arguments, while the third step (Line 3)describes the main routine of generating the grid. In Figure 4.5, the resulting base grid and the two additional segmentation steps are illustrated on the basis of the smartcard package in Figure 4.4.

In the first optional step a sector, defined by an inner and outer bound, is segmented by a defined number of edges in the vector I_{sub}, illustrated as dashed lines. Therefore the inner bound is described by a rectangle, either stated as matrix $InBound =$ `([x w]; ... [y h])` or the name of a component denoted as string. In case of the smart card package this bound should be equally set to to the outer limits of the carrier, otherwise the base

Algorithm 1: Meshing Step 1

Input: Package structure P, Array of fields A_F of length n_F, Field subdivision matrix F_{sub} of size $2 \times n_F$, Part selection $part_{sel}$, Outer bound factor vector $OutBoundFact$, Inner bound Matrix $InBound$, Inter bound subdivision vector I_{sub} of length 2, Array of layer names $LayerName$

Output: Grid structure G with array fields of size $n_x \times n_y$

$\mathrm{x_{edge}} \leftarrow$ `UniqueAndSort(Union(`$P.x$,$P.x+P.w$`));`

$y_{edge} \leftarrow$ `UniqueAndSort(Union(`$P.y$,$P.y+P.h$`));`

`/* Step 1: Create edges between inner and outer bounds */`

if $InBound$ *is String* **then**

  $InBound \leftarrow$ `BuildInBoundMatrix(`P,$InBound$`)`

left $\leftarrow InBound(1,1)$, width $\leftarrow InBound(1,2)$;

bottom $\leftarrow InBound(2,1)$, height $\leftarrow InBound(2,2)$;

$\Delta x \leftarrow width \cdot OutBoundFact(1)/I_{sub}(1)$;

$\Delta y \leftarrow height \cdot OutBoundFact(2)/I_{sub}(2)$;

for $i \leftarrow 1$ **to** $I_{sub}(1) - 1$ **do**

  $\mathrm{x_{edge}} \leftarrow$ `Union(`$\mathrm{x_{edge}}$, $left + width + i \cdot \Delta x$, $left - i \cdot \Delta y$`);`

for $j \leftarrow 1$ **to** $I_{sub}(2) - 1$ **do**

  $y_{edge} \leftarrow$ `Union(`y_{edge}, $bottom + height + j \cdot \Delta x$, $bottom - i \cdot \Delta y$`);`

grid would lead to a coarse grained mesh of the surrounding card body, decreasing the accuracy of the performed thermal simulation. In contrast, the outer bound is defined by a multiple of the inner bound rectangular. This makes the finding of a suitable range for the meshing sector far more easier by simply changing the outer bound factor $OutBoundFact$ in the x or y direction.

Another method to affect the mesh granularity is to increase the number of segments for certain parts defined in the package structure, implemented in the optional second step. For this purpose the field array A_F holds the names of certain parts, the name or the field identifiers of certain components. To distinguish the name related to a part of a component from the name of the component itself, which actually is the string name of the field identifier, a boolean variable $part_{sel}$ has to be set to true, otherwise to false. If an entire component is chosen defined as a set of parts, like the carrier in the smart card package, the outer limit of the set is determined and taken as sector to segment. The number of segments is defined by the field subdivision matrix F_{sub}, where the first row correspond to the divisions in x-direction and the second to one in y-direction of each entry in the field array. e.g.:

Listing 4.3: Example of defining certain parts to be segmented.

```
A_F = {'c1','c2','c3','C4'};
F_sub = [1 1 1 1;2 2 2 1];
part_sel = true;
```

The third sub-algorithm performs the actually meshing of the package data. The intersections of the x an y edges with size of n_x and n_y, respectively, gained from the package

Algorithm 2: Meshing Step 2

```
/* Step 2: Create edges within components or their parts                */
for i ← 1 to n_F do
    if A_F(i) is String then
        if part_sel is set then
            idx(i) ← FindStrInArray(P,A_F(i));
        else
            idx(i) ← Find(P.field == FindStrInArray(LayerName,A_F(i)));
    else                                   /* Entry of A_F(i) is field id.  */
        idx(i) ← Find(P.field ==A_F(i));
for i ← 1 to Length(idx) do
    /* Min Max for segmentation a set of parts as one sector.             */
    width ← Max(P.x(idx(i)) + P.w(idx(i))) − Mín(P.x(idx(i)));
    height ← Max(P.y(idx(i)) + P.h(idx(i))) − Mín(P.y(idx(i)));
    Δx ← width/(F_sub(1,i) + 1), Δy ← height/(F_sub(2,i) + 1);
    for j ← 1 to F_sub(1,i) do
        x_edge ← Union(x_edge, ← Mín(P.x(idx(i))) + j · Δx);
    for k ← 1 to F_sub(2,i) do
        y_edge ← Union(y_edge, ← Mín(P.y(idx(i))) + k · Δy);
```

data and the optional steps building a set of $< x, y >$ pairs defining the position of each element in the grid. Therefore the x and y values of the edges are saved in two vectors such that by accessing the vectors with the same index the x and the y coordinate of on element is gained. The first n_y entries of of both arrays are related to first column of the grid starting at the left lower corner from $Grid(1, 1)$ to $Grid(n_y, 1)$. The next n_y entries starting from $Grid(1, 2)$ to $Grid(n_y, 2)$ are related two the second column. This last n_x entries starting from $Grid(1, n_x)$ to the top right corner $Grid(n_y, n_x)$. In addition, the width w and height h of each element are saved in two more arrays. The meshing algorithm finally provides the position and dimension arrays as fields in a structure G at the output.

Algorithm 3: Meshing Step 3

```
/* Step 3: Grid generation                                               */
n_x ← Length(x_edge) − 1, n_y ← Length(y_edge) − 1;
for i ← 1 to n_x do
    for j ← 1 to n_y do
        G.x(j + (i − 1) · n_y) ← x_edge(i);
        G.y(j + (i − 1) · n_y) ← y_edge(j);
        G.w(j + (i − 1) · n_y) ← x_edge(i + 1) − x_edge(i);
        G.h(j + (i − 1) · n_y) ← y_edge(j + 1) − y_edge(j);
```

4.2.6 Layer Building

The resulting grid from the meshing algorithm in the last section together with the material properties defined in the package data structure are used to build up a layer structure containing the thermal resistances R and the thermal capacities C of each element in every layer. The package components of one layer are mapped onto the grid by assigning the different material properties to the corresponding elements of the grid. A list of field identifiers $L_{ids} = [field_1, ..., field_n]$ specifies the used components in one layer.

The order of the identifier in the list defines the priorities p of the components such that $p(field_i) > p(field_{i+1})$. So an element which already got material properties assigned can not be chosen again. The layer building algorithm is composed of two sub algorithm, as described in Algorithm 4, and invoked for each layer to be built.

Algorithm 4: BuildLayer

Input: Grid structure G with array fields of size $n_x \times n_y$, Package structure P, List of component identifier L_{ids}

Output: Grid structure G with additional *north*, *west*, *top*, *name* and *em* arrays as fields.

1 Logical array of components $LA_{com} \leftarrow$ **false**;
2 Logical priority array $LA_p \leftarrow$ **false**;
3 Logical power array $LA_{pwr} \leftarrow$ **false**;
4 **foreach** $field_i$ *in* L_{ids} **do**
5   Get the indices idx_{part} of the component parts matching $field_i$;
6   left $\leftarrow P.x(idx_{part})$, right $\leftarrow$ left $+P.w(idx_{part})$;
7   lower $\leftarrow P.y(idx_{part})$, upper $\leftarrow$ left $+P.h(idx_{part})$;
8   Logical set array $LA_{set} \leftarrow$ **false**;
    `/* Step 1:Determine logical vector of the current component */`
9   **for** $i \leftarrow 1$ *to* `Length(`idx_{part}`)` **do**
10         $LA_{set} \leftarrow (G.x \geqslant left(idx)$ **And** $G.x \leqslant right(idx))$ **And** $(G.y \geqslant lower(idx)$ **And** $Grid.y \leqslant upper(idx))$;
11         $LA_{com}(LA_{set}) \leftarrow$ **true**;
12         $LA_{com} \leftarrow (LA_{com}$ **Xor** $LA_p)$ **And** LA_{com};
13         $LA_{set} \leftarrow LA_{com}$ **And** LA_{set};
14         $G.name(LA_{set}) \leftarrow P.name(idx_{part}(i))$;
15         $G.em(LA_{set}) \leftarrow P.em(idx_{part}(i))$;
    `/* Step 2:Determine R and C values of the elements in G. */`
16   Get material properties *th*,*k*,*cv*,*em* of component from $field_1$ in L_{ids};
17     $G.north(LA_{com}) \leftarrow$ `GetR(`$G.h(LA_{com}) \cdot 0.5, G.w(LA_{com}) \cdot th, k$`)`;
18     $G.west(LA_{com}) \leftarrow$ `GetR(`$G.w(LA_{com}) \cdot 0.5, G.h(LA_{com}) \cdot th, k$`)`;
19     $G.top(LA_{com}) \leftarrow$ `GetR(`$th \cdot 0.5, G.w(LA_{com}) \cdot G.h(LA_{com}), k$`)`;
20     $G.C(LA_{com}) \leftarrow$ `GetC(`$th, G.w(LA_{com}) \cdot G.h(LA_{com}), cv$`)`;
21   **if** $field_i$ *belongs to chip (die)* **then**
22         $power(LA_{com}) \leftarrow$ true;
23     $LA_p(LA_{com}) \leftarrow$ true;

In a first step the elements of the grid, which cover the parts of the component specified by the field id $field_i$ are determined. As mentioned in Section 4.2.4, logical and relational operations on an array of elements can be used to find specific entries. Hence, each entry of the $G.x$ and $G.y$ grid arrays can be compared with the values of the component parts specified in the package structure P, implemented in Line 10. The resulting logical vector LA_{set} holds entries set to one, where the elements correspond to the component parts. The same entries are also set in the logical vector LA_{com} that holds all parts of one component after one iteration of the outer for loop. To prevent the overwriting of already set entries a XOR and AND operation with the priority vector LA_p is performed. Finally the vector LA_{set} is updated before it is applied to assign the correct names and the emissivity to each element.

After all parts of a component are matched a second step uses the the vector LA_{com} to calculate the R and C values of each element and saving the the result in arrays as additional fields of the grid structure. In addition the entries of the priority vector are set according to LA_{com}. When the elements, that are related to the chip (die), are processed, the appropriate entries in the logical power vector are set to true.

The resulting struct itself is again saved into a struct, called the layer struct array. The layer struct array can be accessed with the field variables used as the layer identifiers `field_id`$_1$`,...,field_id`$_L$.

With the function call `layer(field_id`$_i$`)=buildLayer(grid, package,` L_{ids}`)` one layer identified by `field_id`$_i$ is built, where the array L_{ids} holds one or more field identifier beginning with `field_id`$_i$. In Listing 4.4 the possible layer build up of the smart card package in Figure 4.4 is listed.

Listing 4.4: The layer build up of the smart card package.

```
layer_sa(CB)       = buildLayer(grid,package,[CB]);
layer_sa(GT)       = buildLayer(grid,package,[GT CB]);
layer_sa(DIE)      = buildLayer(grid,package,[DIE GT CB]);
layer_sa(DA)       = buildLayer(grid,package,[DA GT CB]);
layer_sa(CR)       = buildLayer(grid,package, [CR GT_MC CB]);
```

The corresponding vector containing the order of the layers looks like $layer_{order}$ `= [CB;GT;DIE;DA;CR;CB]`. Since the card body (CB) above the glob top (GT) and under the carrier (CR) is assumed to be equal, the layer has only to be built ones.

4.3 The Thermal Simulator

The thermal solver is able to analyse the thermal behaviour - the transient analysis - of the package as well as estimating its temperatures when staying in thermal equilibrium - steady state analysis - with the ambient environment. With the segmentation of the package into a set of elements, each represented by three thermal resistances and one thermal capacity, together with the power information of the active elements, a thermal network is formed. To account for the acting of the thermal network with its environment by radiation and natural convection effects a combined resistance towards the ambient is implemented. The building of the thermal network as the implementation of the network is presented in the next sections.

4.3.1 Power Information Matching

The power information of the active units of the μChip, stored in the power matrix P, has to be matched due to new segmentation of the die in certain elements. During the building process of the die layer, a logical power vector, as introduced in Section 4.2.6, is generated, marking each element of the die by setting the according entry to true. Together with element names, the corresponding unit for each element is found and the unit power values are split up based on the following area based method.

$$P_{element} = \frac{Area_{element}}{Area_{unit}} \cdot P_{unit} \tag{4.4}$$

with $Area_{unit}$ is the area of one unit at the floorplan before the meshing is performed. The Power matrix P is updated with the power information of each element.

4.3.2 Building the network

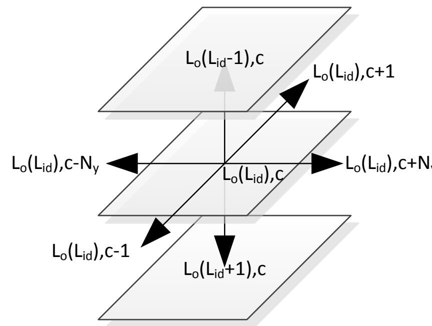


Figure 4.6: Adjacency of one element.

The building network algorithm, as described in Algorithm 5, implements the structure of the thermal network as a sparse matrix S, with entries s_{ij} of the conductance between

two elements i and j. All entries of the matrix are symmetric with respect to the main diagonal, where all entries are zero, due to $s_{ij} = s_{ji} \forall i, j$ holding the same entries.

The thermal conductivity between two elements can simply be calculated by the summation of the resistances, related to the sides both are in contact with, and taking the result's reciprocal. Therefore the adjacency of the elements are determined based on the layer order L_o and the positioning of the elements in the grid arrays as illustrated with the index pairs in Figure 4.6. As an example, the index for a west adjacent cell is gained by the id of the current layer $L_o(l)$ and the current element(cell) id c minus the dimension n_y. The current implementation treats the west, east, north or south side of an element in contact with the ambient as perfect isolated (conductance G = 0). This assumption holds as long as the amount of heat deduced from this side areas by the cooling mechanism is negligible. That meets in case of a smart card with small side areas in contrast to the upper or lower side, due to the minor thickness of the card body.

After the sparse matrix is built the sum of each row is determined and saved into the Sum_{SP} array. In addition the R_{amb} array is built, that holds the resistances to the ambient surface of the bottom and top layer. For calculating the dynamical resistance of the cooling mechanism the inverse of the areas in contact with ambient A_{inv} as well as an emissivity array E is determined.

In Figure 4.7 the density of a possible conductance matrix described by the sparse matrix S is illustrated; the black spots represent the non zero (nz) elements. In this example the matrix holds the connecting conductances of 7920 nodes, leading to a full matrix of the size 7920×7920 with 41364 non zero elements. Hence, the use of a sparse matrix only needs the saving of approximately 0.068% of the full matrix elements, leading to a reduced memory usage of only 0.1527% of that the full matrix would require.

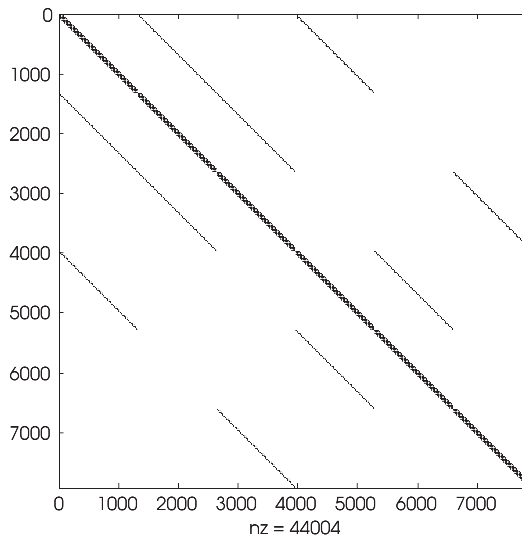


Figure 4.7: Matrix Density of sparse matrix S.

Algorithm 5: Building network

```
Input: Layer nested structure L of n_L grid structures with arrays of size
       n_x × n_y, Layer order array L_o of length n_L
Output: Sparse Matrix S, Row sum array S_sum, Debug string D and C array
        A_C
n_all ← n_x · n_y;
for l ← 1 to n_L do
    for c ← 1 to n_all do
        idx_all ← c + ((L_o(l)) − 1) · n_all;
        D(idx_all) ← Generate debug string L_⟨layer⟩ C_⟨cell⟩ N_⟨name of cell⟩;
        A_C(idx_all) ← L(L_o(idx_L)).C(c);
        if Top or Bottom layer then
            if Top layer then
                to ← ((L_o(l+1)) − 1) · n_all + c;                /* layer beyond */
                S(idx_all,to)* ← {L(L_o(idx_L)+1).top(c) + L(L_o(l)).top(c)}^−1;
            else                                                  /* Bottom layer */
                to ← ((L_o(l-1)) − 1) · n_all + c;                 /* layer above */
                S(idx_all,to)* ← {L(L_o(idx_L)).top(c) + L(L_o(l-1)).top(c)}^−1;
        else                                                       /* Other layer */
            to ← ((L_o(l-1)) − 1) · n_all + c;                     /* layer above */
            S(idx_all,to)* ← {L(L_o(l-1)).top(c) + L(L_o(l)).top(c)}^−1;
            to ← ((L_o(l+1)) − 1) · n_all + c;                    /* layer beyond */
            S(idx_all,to)* ← {L(L_o(idx_L)).top(c) + L(L_o(l+1)).top(c)}^−1;
        if Cell is no at west bound then
west        to ← ((L_o(l)) − 1) · n_all + c−n_y;                     /* west cell */
            S(idx_all,to)* ← {L(L_o(l)).west(c)+L(L_o(l)).west(c−n_y)}^−1;
        if Cell is no at east bound then
            to ← ((L_o(l)) − 1) · n_all + c+n_y;                     /* east cell */
            S(idx_all,to)* ← {L(L_o(l)).east(c)+L(L_o(l)).east(c+n_y)}^−1;
        if Cell is no at north bound then
            to ← ((L_o(l)) − 1) · n_all + c+1;                      /* north cell */
            S(idx_all,to)* ← {L(L_o(l)).north(c)+L(L_o(l)).north(c+1)}^−1;
        if Cell is no at south bound then
            to ← ((L_o(l)) − 1) · n_all + c-1;                      /* south cell */
            S(idx_all,to)* ← {L(L_o(l)).south(c)+L(L_o(l)).south(c-1)}^−1;
    S_sum ← Calculate the sum of each row.
Note: Value assigned to S(idx_all,to)* is also assigned to S(to,idx_all)
```

4.3.3 Convection and Radiation Heat Transfer

The cooling of the package is implemented as a combined heat transfer coefficient of the natural convection and radiation. The thermal relations in natural convection are usually based on experimental studies. A widely used empirical correlation method is the correlation of the average Nusselt number usually expressed in terms of the Rayleigh number to the power of n and multiplied by a constant C [8], given by

$$Nu = \frac{h_c \cdot L_c}{k} = C \cdot Ra_L^n \tag{4.5}$$

where the Rayleigh number is a product of the Grashof and Prandtl numbers of the form

$$Ra_L = Gr_L \cdot Pr = \frac{g\beta\left(T_s - T_a\right) \cdot L_c^3}{\nu^2} \cdot \frac{\nu}{\alpha} \tag{4.6}$$

with the gravitational acceleration $g[ms^{-2}]$, the volume expansion coefficient for a idea gas $\beta[1K^{-1}]$, the kinematic viscosity $\nu[m^2s^{-1}]$, the thermal diffusivity αm^2s^{-1}, the characteristic length $L_c[m]$, the surface $T_s[K]$ and the ambient $T_a[K]$ temperatures.

In the case of the smart card package, the card body is assumed to be in vertical position and the fluid (air) flow is laminar. The fluid property β is evaluated at the film temperature $T_f = \frac{1}{2}\left(T_s + T_a\right)$ at each iteration whereas the others (ν, α) are taken as average values for the film temperatures in the range of 25 to 120[°C]. By rearranging the above equations under the assumption of the constant n to be 1/4 ,due to the laminar property, only the constant C is left to be determined. Therefore an ANSYS macro, provided by Infineon Munich, is used that generates convection heat transfer coefficients in a certain temperature range. The result is then compared to the coefficients under varying C values until a proper one is found. In Figure 4.8 the convection coefficients from the ANSYS macro are compared to the those generated for the constant C set to 0.289. The implemented combined convection and radiation heat transfer coefficient gets

$$h_c = 0.289 \cdot \left(\frac{g\beta\left(T_s - T_a\right) \cdot L_c^3}{\nu \cdot \alpha}\right)^{\frac{1}{4}} \tag{4.7}$$

$$h_{rad} = \rho \cdot em \cdot (T_s + T_s) * (T_s^2 + T_s^2) \tag{4.8}$$

$$h = h_c + h_{rad} \tag{4.9}$$

with the *Stefan-Boltzmann constant* $\rho = 5.67 * 10^{-8}[Wm^{-2}K_{-4}]$ and emissivity em [1].

4.3.4 Transient Simulation

The transient simulation is based on an ordinary differential equation (ODE) that describes each element of the thermal network. By applying the Kirchhoff's current law to one node

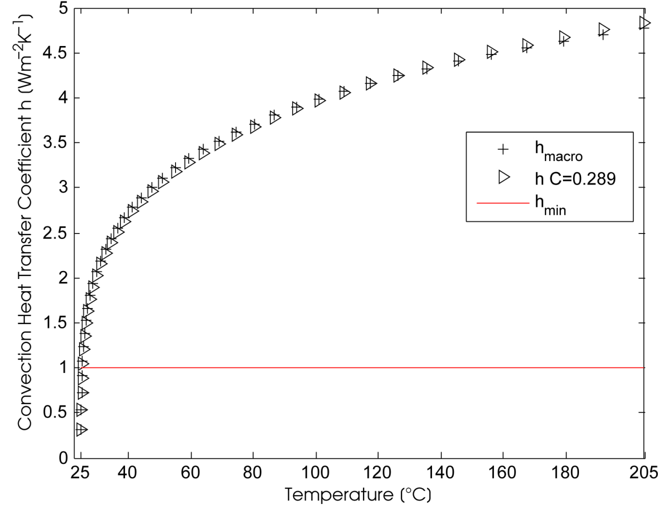


Figure 4.8: Convection heat transfer coefficients gain from the ANSYS macro compared to the implemented h_c with C = 0.289.

of the network, the ODE in terms of the variable of interest $T(t)$ is written

$$G_N(T(t)-T_N)+G_S(T(t)-T_S)+G_W(T(t)-T_W)$$
$$+G_E(T(t)-T_E)+G_B(T(t)-T_B)+G(T(t)-T_T)+C\frac{dT(t)}{dt}=0 \quad (4.10)$$
$$G_{sum}=G_E+G_N+G_S+G_W+G_T+G_B \quad (4.11)$$
$$P_{src}=P+G_BT_B+G_TT_T+G_ET_E+G_NT_N+G_ST_S+G_WT_W \quad (4.12)$$
$$C\frac{d}{dt}T(t)+G_{sum}T(t)=P_{src} \quad (4.13)$$

By setting the source term4.12 in 4.13 to 0 and substituting $T_h(t) = K \cdot e^{-\frac{t}{\tau}}$ as the homogeneous solution, the time constant τ gets

$$\tau=\frac{C}{G_{sum}} \quad (4.14)$$

Assuming the particular solution of constant form $\frac{dT_p(t)}{dt}=0$, $T_p(t)$ gets

$$T_p(t)=\frac{P_{src}}{G_{sum}} \quad (4.15)$$

Combining the homogeneous and particular solutions $T(t) = T_h(t)+T_p(t)$ and finding the missing variable K by using the initial condition $T(t=0) = T_{init}$, the complete solution becomes

$$T(t)=\frac{P_{src}}{G_{sum}}-\mathrm{e}^{-\frac{t}{\tau}}\left(\frac{P_{src}}{G_{sum}}-T_{init}\right) \quad (4.16)$$

Since the temperature of an element is determined at the discrete time points $i\cdot\Delta t$, with $i \in \mathbb{N}\backslash\{\emptyset\}$, the source term 4.12 as the complete solution 4.16 are rewritten. The source

term is composed of the power term, $P(i \cdot \Delta t) > 0$, when an element dissipates power, at the current time point $i \cdot \Delta t$, and the products of the temperatures at previous time point $(i-1) \cdot \Delta t$ and connecting conductances of the adjacent temperature points. With the source term for $i > 0$ gets

$$\begin{aligned} P_{src}(i \cdot \Delta t) &= P(i \cdot \Delta t) \\ &+ G_B T_B((i-1) \cdot \Delta t) + G_T T_T((i-1) \cdot \Delta t) + G_E T_E((i-1) \cdot \Delta t) \\ &+ G_N T_N((i-1) \cdot \Delta t) + G_S T_S((i-1) \cdot \Delta t) + G_W T_W((i-1) \cdot \Delta t) \end{aligned} \tag{4.17}$$

the complete solution becomes

$$T(i \cdot \Delta t) = \frac{P_{src}(i \cdot \Delta t)}{G_{sum}} - \mathrm{e}^{-\frac{\Delta t}{\tau}} \left(\frac{P_{src}(i \cdot \Delta t)}{G_{sum}} - T((i-1) \cdot \Delta t) \right) \tag{4.18}$$

with

$$T(0) = T_B(0) = T_T(0) = T_E(0) = T_N(0) = T_S(0) = T_W(0) = T_{init} \tag{4.19}$$

Based on this solution function the transient simulation algorithm, as described in Algorithm 6, determines the temperature profile of each package element over a certain amount of time. Since MATLAB® [28] efficiency lies in performing matrix and vector operations, the processing of the package elements in terms of their spatial distribution can be implemented without the explicit use of for loops. The only for-loop used describes the time duration as an iteration over n_s samples. Hence, the sparse matrix S and the arrays from the building network algorithm 4 are directly used in the calculations.

Before the basic algorithm, step 2 and 3, are executed, some precalculation in step 1 (line 1-2) are performed to reduce the computational effort (by $\sim 60\%$) of the euler function and its exponent.

$$\mathrm{e}^{-\frac{\Delta t}{\tau}} = \underbrace{\mathrm{e}^{-\frac{\Delta t}{C} \cdot G_{sum}}}_{e_{pre}} \cdot \mathrm{e}^{-\overbrace{\frac{\Delta t}{C}}^{exp_{pre}} \cdot G_{amb}} \tag{4.20}$$

where the i-th entry of the vector G_{sum} includes G_T and G_B if the element i is not in contact with the ambient, thus the i-th entry of G_{amb} contains 0, otherwise G_T or G_B if it is in contact with. Due to the radiation and the convection heat transfer, the G_{amb} of length $n_{all} = n_x \times n_y \times n_L$ is determined at each iteration by means of the R_{amb} , the A_{inv} and the h_{comb} vectors of length $2 \times n_x \times n_y$. The latter two together form the combined radiation and convection resistances, whereas the R_{amb} contains the top or the bottom resistances of each element in contact with the ambient.

Besides the involving of the temperatures from the previous iteration step, the main difference between the initial calculation in step 2 and the main one in step 3 lies in the use of a logical array LA_{save} for saving the temperature values at user defined time intervals. Therefore the specified time intervals are converted into corresponding sampling intervals and saved as a logical vector. At the iteration step i where the logical array entry i holds a logical one, the current temperature values are saved to a temperature monitor matrix $T_{monitor}$. This is possible since the calculation of the current temperature values only

---

Algorithm 6: Transient Simulation

Input: C array A_C, Sampling interval Δt_s, Conductance sum array G_{sum} of length n_all, Temperature ambient array TA_{amb}, Contact to ambient logical array LA_{amb}, unit logical array LA_{unit}, Sparse conductance matrix S_G, Resistance to ambient array R_{amb}, Area inverse array A_{inv}, Power matrix P of size $n_u \times n_s$

Output: Temperature array T of size n_{all}, Temperature monitor matrix $T_{monitor}$

Note: Elementwise multiplication of arrays is expressed by the $\cdot$ symbol. The colon operator : without operand means entire range of column/row.

`/* Step 1:Precalculation */`

1. $exp_{pre} \leftarrow -\Delta t_s \cdot A_C^{-1}$;
2. $e_{pre} \leftarrow \texttt{exp}(exp_{pre} \cdot G_{sum})$;

`/* Step 2:Initial Calculation` $i = 1$ `*/`

3. $h_{comb} \leftarrow$`getHeatTransferCoef(`TA_{amb},TA_{amb},em`);`
4. $G_{amb}(LA_{amb}) \leftarrow \left(R_{amb} + A_{inv} \cdot h_{comb}^{-1}\right)^{-1}$;
5. $e_{vec} \leftarrow e_{pre} \cdot \texttt{exp}(exp_{pre} \cdot G_{amb})$;
6. $G_{inv} \leftarrow (G_{amb} + G_{sum})^{-1}$;
7. $P_{vec}(LA_{unit}) \leftarrow$ assigning first row of P;
8. $P_{src} \leftarrow (P_{vec} + S_G TA_{amb} + G_{amb} \cdot T_{amb}) \cdot G_{inv}^{-1}$;
9. $T \leftarrow e_{vec} \cdot (TA_{amb} - P_{src}) + P_{src}$;
10. $T_{monitor}(1, :) \leftarrow T$, saveto $\leftarrow 1$;

`/* Step 3: Calculation` $i > 1$ `*/`

11. **for** $i \leftarrow 2$ **to** n_s **do**
12. $P_{vec}(LA_{unit}) \leftarrow$ assigning i-th row of P;
13. $h_{comb} \leftarrow$`getHeatTransferCoef(`TA_{amb},$T(\text{LA}_{amb})$,em`);`
14. $G_{amb}(LA_{amb}) \leftarrow \left(R_{amb} + A_{inv} \cdot h_{comb}^{-1}\right)^{-1}$;
15. $e_{vec} \leftarrow e_{pre} \cdot \texttt{exp}(exp_{pre} \cdot G_{amb})$;
16. $G_{inv} \leftarrow (G_{amb} + G_{sum})^{-1}$;
17. $P_{src} \leftarrow (P_{vec} + S_G T + G_{amb} \cdot T_{amb}) \cdot G_{inv}^{-1}$;
18. $T \leftarrow e_{vec} \cdot (T - P_{src}) + P_{src}$;
19.   **if** $LA_{save}(i)$ **then** $T_{monitor}(\text{saveto}, :) \leftarrow$ T, saveto $\leftarrow$ saveto + 1;

---

need the temperatures of the previous iteration step $i-1$. Since there is no direct data dependency between current and previous temperature values, both can be stored in one temperature column array T of length n_{all}. With this approach it is possible to reduce the memory usage on an user based need.

4.3.5 Steady State Simulation

In addition to the simulation of the dynamical behaviour, an analysis of the steady state temperatures can be performed. An system is said to be in thermal steady state or stays in thermal equilibrium with the environment when the heat generated equals the heat removed by the cooling mechanism, thus the net change in the thermal capacitor temperature $C\frac{dT(t)}{dt}=0$ in Equation 4.10 equals zero. As a consequence the network can simply be analysed by performing the well known branch current method producing a linear set of compact equations represented in matrix form by

$$\begin{pmatrix} G_{11} & -G_{12} & \cdots & -G_{1n} \\ -G_{21} & G_{22} & \cdots & -G_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ -G_{n1} & -G_{n2} & \cdots & G_{nn} \end{pmatrix} \cdot \begin{pmatrix} T_1 \\ T_2 \\ \cdots \\ T_n \end{pmatrix} = \begin{pmatrix} P_1 \\ P_2 \\ \cdots \\ P_n \end{pmatrix} \tag{4.21}$$

where the non-diagonal negative elements $G_{ij}\,\forall i \neq j$ are the connecting thermal conductances between the node and the diagonal elements are the absolute sum of conductances a node is connected with its adjacent nodes. The sparse matrix S gained from the building network Algorithm 4 already contain the connecting conductances and has to be modified by the multiplication of -1 and extended by the sum of the conductance as well as the conductances to ambient. Moreover the power source vector $P_1 ... P_n$ is composed of the average power the units of the μChip dissipates and the heat flow towards the ambient $G_{amb} \cdot T_{amb}$.

For solving the Equation 4.21 towards the temperature vector an LUP decomposition is used, as implemented in Algorithm 7. As the conductances to the ambient are dependent

Algorithm 7: steadyState

Input: Sparse conductance matrix S_G, Average power P_{av}, Ambient conductance vector G_{amb}, Row sum vector G_{sum} and the Area inverse array A_{inv}

Output: Steady state temperature T_{steady}

1 $S_G \leftarrow -S_G +$ `sparse(diag(`$G_{sum} + G_{amb}$`))`;

2 $P_{src} \leftarrow P_{steady} + G_{amb} \cdot T_{amb}$;

3 $[L, U, P] \leftarrow$ `lu` (S_G);

4 $T_{steady} \leftarrow U \backslash (L \backslash (P_{source}(P, :)))$;

on the surface temperatures an iterative approach, as implemented in Algorithm 8 takes account of the non linearity by executing the steady state calculation as often as the differences of the current heat transfer coefficients to its previous ones is higher than a certain limit. The iteration approach was taken from HOTSPOT and adapted to the thermal simulator.

Algorithm 8: Steady State Iteration

Input: Sparse conductance matrix S_G, Average power P_{av}, Ambient conductance vector G_{amb}, Row sum vector G_{sum} and the Area inverse array A_{inv}.
Output: Stead state temperature vector T_{steady}.

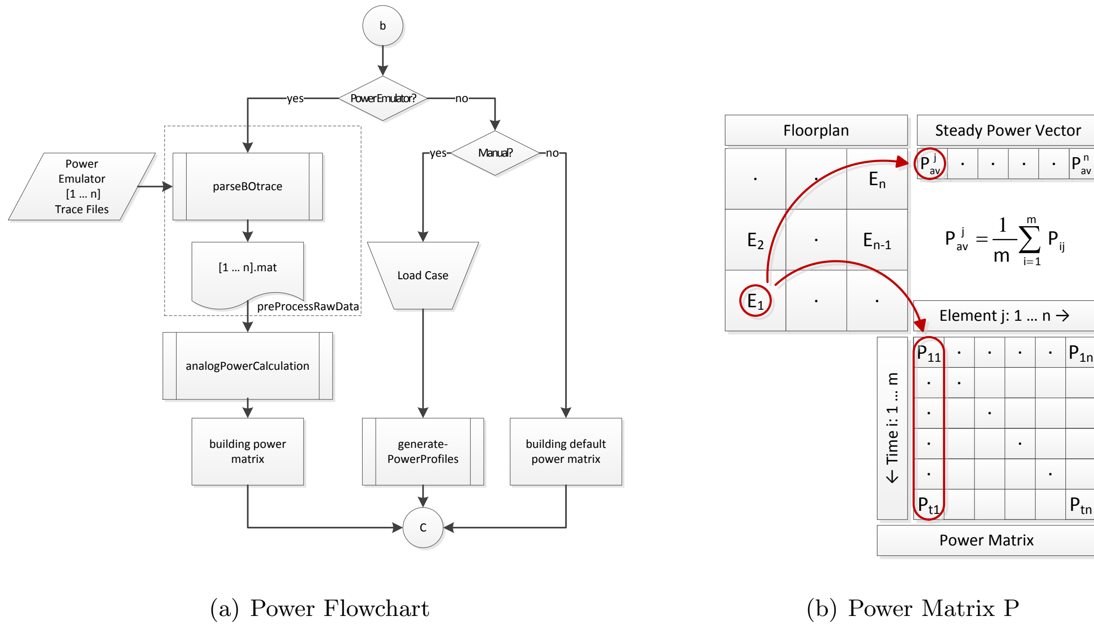
```
convergence ← 0; idx_f ← getSurfaceCellIds(L_o,n_x × n_y);
count ← 1;
h_comb ←getHeatTransferCoef(TA_amb,TA_amb,em);
while convergence do
    h_prev ← h_comb;
    T_steady ← steadyState(P_av,S_G,G_amb,G_sum);
    T_s ← T_steady(idx_f);
    h_comb ← getHeatTransferCoef(TA_amb,T_s,em);
    G_amb(LA_amb) ← (R_amb + A_inv · h_comb^-1)^-1;
    if abs(h_comb - h_prev) < NATURAL_CONVEC_TOL then
        convergence ← 1;
    count ← count +1;
```

4.4 The Power Model

Depending on the users choice, the power data for the simulation is obtained from the power emulator or a predefined power profile can be used. For more simple power assumptions, a single power value for the the whole chip can be declared independent of the die's fragmentation. The power data are represented as a m x n matrix of double values. Each element that is capable of consuming power (not necessarily restricted to the die) belongs to one column vector containing the power values at any time step $i \cdot \Delta t$, see Figure 4.9(b). In addition the steady power vector contains the average power of each power consuming element necessary for performing the steady state analysis. In the following the various processes of building the power data are described as illustrated in Figure 4.9(a).

4.4.1 Power Emulator Mode

The power emulator provides a power profile of the target system while running a certain application to be evaluated. By configuring the power sensors the profiles of the modules desired for a thermal analysis at system level or at architectural level (e.g. functional until units of the CPU) can be determined. For evaluation and further processing the debug trace generator finally delivers the power information as a trace to the host PC, as described in Chapter 2.1.2.2. The generated power trace files (ASCII text format) are then parsed in a preprocessing step of the thermal simulator that saves the power values and the corresponding time stamps of each trace file as a structure array. For calculating the transient thermal network the power values at a integer multiple of Δt are needed. Since the interval of the time stamps are non-uniform the Δt is determined by taking the minimal interval of all time stamps. To get the corresponding power values a piecewise cubic hermite interpolate (MATLAB® intern pchip function) is applied. In addition the



(a) Power Flowchart (b) Power Matrix P

Figure 4.9: Illustration of the power trace matrix notation and the flow chart of building the power matrix.

mean and the peak value are determined and added to the structure array together with the new time line and interpolated power data before saved into the MATLAB® intern MAT-file format. In the current implementation the power trace gained from the emulator is assumed as an overall power estimate of the digital circuit of the μChip, thus one MAT-file correspond to one particular application.

4.4.1.1 Analog Power Model

The analogue power model uses the power trace structure from above along with the analogue config structure, a digital power mode variable and the saving directory as input. The power mode variable enables the use of the mean or the maximal value of the power trace as alternative to the original one. The analogue configuration structure contains the electrical characteristics of the electromagnetic coupling an the μChip necessary for calculating the load current and in further consequence determining the remaining power consumption of the analogue circuit. As described in 3.3.1 the μChip can be treated as network of a current source and an arbitrary number of current sinks. By determining the ohmic loss of the μChip which is defined by the shunt regulator and current consumption of the security controller the load current (source) can be calculated. The resonant circuit described in 2.15 is used to establish a formulation of the load current that is implemented as follows.

The input voltage of the analogue frontend is assumed to be kept at a constant voltage U_L by a serial regulator and stated as RMS value equal to the induced voltage U_i. The

transfer function from the induced voltage to the chip voltage can then be formulated as

$$\frac{U_i}{U_L} = \frac{j\omega L_2 + R_2 + Z_L}{Z_L} \tag{4.22}$$

with

$$Z_L = \frac{R_L}{1 + R_L/jX_C} \qquad X_C = -\frac{1}{\omega C_L} \qquad Q = \frac{X_L}{R_2} \qquad X_L = \omega L_2 \qquad \omega = 2\pi f_{sys} \tag{4.23}$$

By dividing transfer function into real and imaginary part

$$\frac{U_i}{U_L} = \frac{X_L X_C + Q R_L X_C + Q R_L X_L}{Q R_L X_C} - j\frac{X_L \left(R_L - Q X_C\right)}{Q R_L X_C} \tag{4.24}$$

and calculating the absolute value leading to

$$\sqrt{\frac{2X_L}{X_C} + \left(\frac{X_L}{R_L}\right)^2 + \left(\frac{X_L}{X_C}\right)^2 + \left(\frac{X_L}{QR_L}\right)^2 + \left(\frac{X_L}{QX_C}\right)^2 + \frac{2X_L}{QR_L} + 1} = \frac{U_i}{U_L} \tag{4.25}$$

and solving the the equation for R_L gets

$$\begin{aligned} R_L = \{ & U_L X_C X_L \left[Q^4 U_i^2 X_C^2 - Q^4 U_L^2 X_C^2 - 2Q^4 U_L^2 X_C X_L - Q^4 U_L^2 X_L^2 + \right. \\ & \left. Q^2 U_i^2 X_C^2 - 2\, Q^2 U_L^2 X_C X_L - 2Q^2 U_L^2 X_L^2 - U_L^2 X_L^2 \right]^{\frac{1}{2}} - Q U_L X_L \,\} \cdot \\ & \{ Q^2 U_L^2 X_C^2 - Q^2 U_i^2 X_C^2 + 2Q^2 U_L^2 X_C X_L + Q^2 U_L^2 X_L^2 + U_L^2 X_L^2 \}^{-1} \end{aligned} \tag{4.26}$$

Dividing the input voltage by the ohmic loss the load current is calculated

$$I_L = \frac{U_L}{R_L} \tag{4.27}$$

Together with the known overall current consumption (sink) of the security controller the outstanding current for the shunt regulator (sink) is determined by

$$I_{Shunt} = I_L - I_{Digital} \tag{4.28}$$

By assuming a constant input voltage V_{DD} of the security controller the desired traces for each power consuming component is calculated as follows

$$P_{Digital} = I_{Digital} \cdot V_{DD} \tag{4.29}$$

$$P_{Shunt} = I_{Shunt} \cdot V_{DD} \tag{4.30}$$

$$P_{Reg} = (U_L - V_{DD}) \cdot I_L; \tag{4.31}$$

The results are separately saved as double arrays into the .MAT file format named after the application and the component, thus each power consuming component has its own power trace file.

4.4.1.2 Building the Power Matrix

Building the power matrix requires a power trace file of each functional unit that is part of the μChip's floorplan. The building process searches the saving folder and matches the files with the components of the floorplan by name comparison. The name of the component has to exist in the file name and are not case-sensitive. Finally the power values are combined to the power matrix illustrated in Figure 4.9(b) arranged identically to the order of the floorplan components. In addition the mean of each column is calculated for a possible steady state analysis.

4.4.2 Manual and Default Mode

As an alternative to the power emulation for each unit a power trace can be build by hand as discussed in Section 3.3. The unit name corresponding with the unit at the chip floorplan, its power values and the amount of time each power value should last are saved in three separate cell vectors. Hence, the power profile definition of a unit i can easily accessed by using the same index i for all three vectors. A possible power profile definition of a chip consisting of three units is shown in Listing 4.

```
1 pow_unit_name_ca = {'blue','black','red'};
2 pow_val_ca       = {[1.5 0.75 1.25 1.0],[1.0 0.5 1.5],[0 1 0.5]};
3 dur_val_ca       = {[1 1 1 1],[1 1 1],[1 1 1]};
```

Listing 4.5: The definition of three power traces as cell vectors.

The power values of one unit as their according time durations are represented as double vectors stated as unit milliwatt [mW] and unit second [s], respectively. The *generatePowerProfile* process takes the cell vectors as input to build the overall power matrix P in the power trace matrix notation, illustrated in Figure 4.9(b). Since the power values are needed in the time discrete domain the power profile has to be sampled by the sampling interval Δt_s. In Figure 4.10 the power profiles defined in Listing 4 with an additional zero-padding in case of power vectors with different size is illustrated.

If either the power emulation nor the manual power mode is used a single power value for the entire μChip is applied although the chip model contains several functional units. Therefore the power value is distributed according to the covered area of the units and assigned to them.

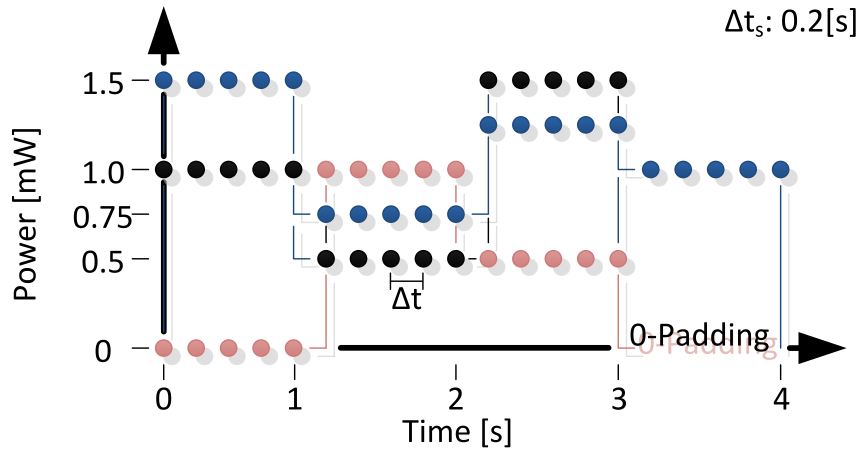


Figure 4.10: Illustration of three three manual power traces as a result of the Listing 4.

4.5 Thermal GUI

4.5.1 Main Interface

Due to the variety of possible settings and to achieve better usability the thermal simulator is provided by a graphical user interface (GUI), as illustrated in Figure 4.11. The GUI is not only acting as interface between the user and the main thermal routine, it also performs specific tasks as triggering the translation of the chosen floorplan into the .flp data type as well creating the logical vector for the saving points of the temperatures values. The primary function of the former process is to obtain the unit names of the chip floorplan for a possible Power GUI enabling a more comfortable way to generate a manual power profile, as described in Section 3.3. In the current implementation the manual power profile has to be defined in the main function, whereas six load cases are predefined and can be chosen by setting the *load_case* variable to the appreciate case number.

The gui provides one listbox for choosing a particular floorplan of the μChip and the corresponding packaging as a second one, respectively. Tough the Figure 4.11 suggest that the association of packaging to the floorplan is checked by name, the chosen μChip must fit into the packaging and is independent of its naming. Besides the options of performing a transient or/and steady state simulation the user can enable the optional validation process if the appropriate reference data are available. When the transient behaviour should be analysed the time to be simulated as the sampling interval (time step) can be set, whereas the time step is automatically set to the sampling interval of the power trace if the power emulation option is active. The time settings are linked with the save point settings, described in the next section.

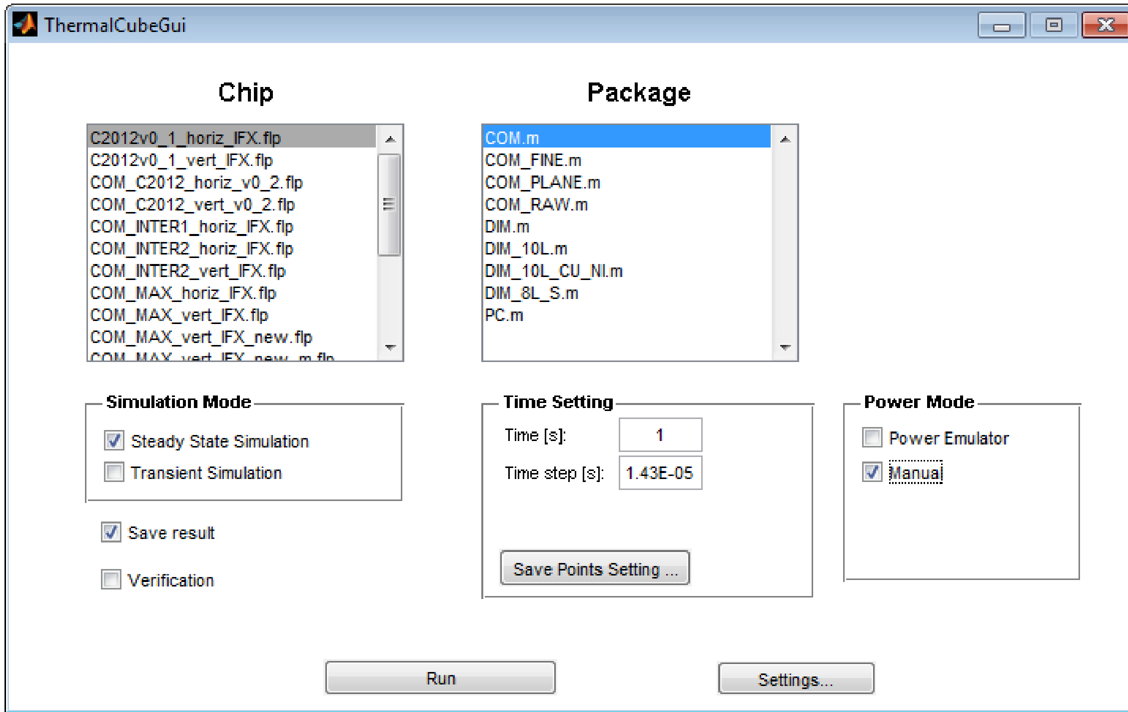


Figure 4.11: Main Thermal GUI.

4.5.2 Saving Point Table Interface

When the transient simulation is chosen in the main thermal GUI, the interval, within the temperature value should be stored, can be comfortable defined with the help of the GUI shown in Figure 4.12. The Saving Point Table Interface enables the definition of an arbitrary number of intervals by declaring the start and the end point in time as well as the step size such that one row in the table represents one interval. In the beneath plot the entered intervals are visualised within a typical temperature curve of a system in a warm up process.

4.5.3 Settings

The location of the floorplan and packaging data, the raw and parsed power profile data of the digital IC as well as the analogue power data and the analogue configuration file are declared in the Settings GUI, shown in Figure 4.13. When a validation process is performed, the directory where the reference data are resided, the name of the excel file and the number of reference points have to be filled into the validation panel. The Sheet name is automatically determined from the chosen floorplan by taking the letters till the second underscore symbol an can be changed when the naming in the excel file is a different one.

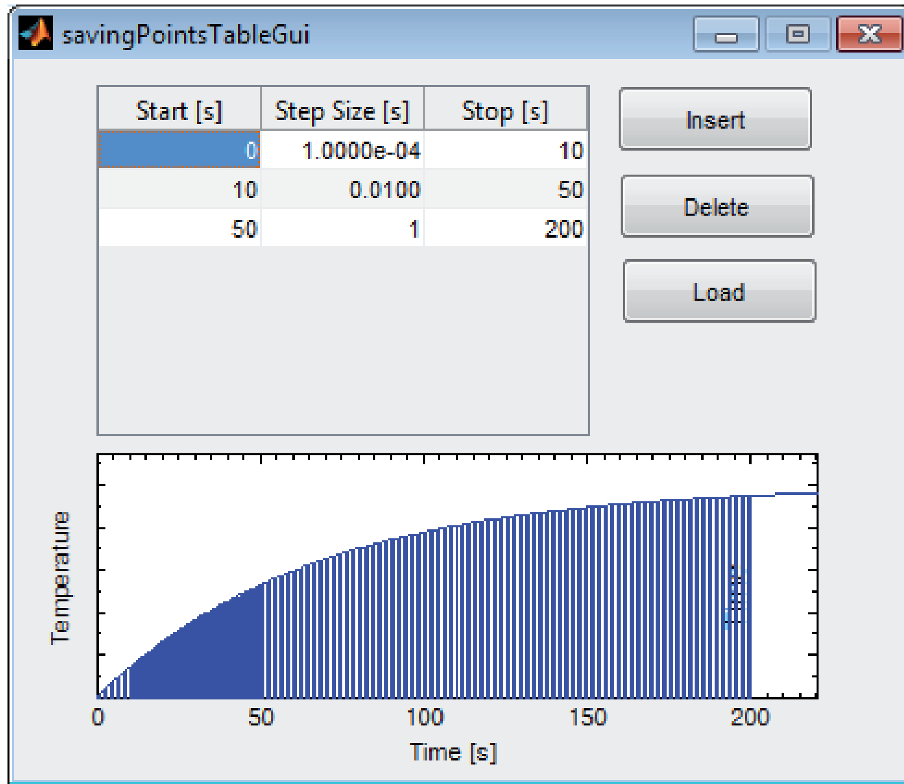


Figure 4.12: Saving Point Table GUI.

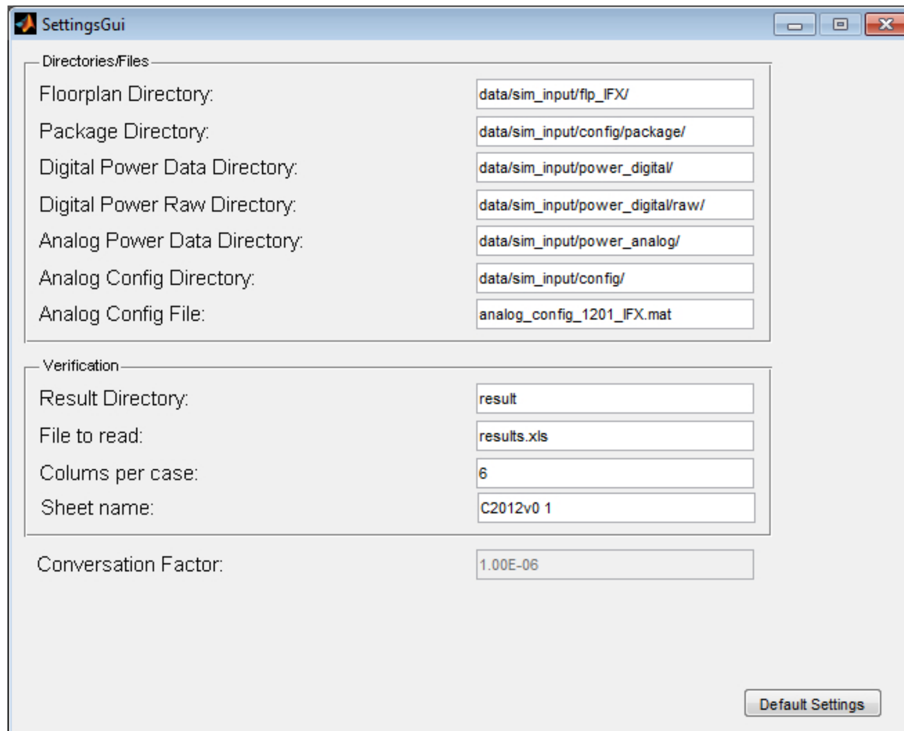


Figure 4.13: Setting GUI.

Chapter 5

Experimental Results

5.1 Introduction

To validate the correct functionality of the thermal simulator, several experiments including transient and steady state simulations are performed. In a first and early step the thermal simulator is validated against the software simulator HotSpot in its block mode. In addition, the thermal simulator with the automatic meshing is compared to the more accurate HotSpot Grid model. Alongside the accuracy enhancement, a huge performance increase from a first to the final model can be shown. In consequence of using the thermal simulator for determining the temperature behaviour of smartcards with their more complex architectures the final validation is performed against the commercial FEM simulator ANSYS.

Additionally, different use cases have been established to determine the heat transfer of both contactless-only and dual interface card under various work load conditions. As an final use case the transient behaviour of an contactless smartcard performing an alu arithmetic test is presented.

5.2 Validation with Hotspot

5.2.1 Block Model versus ThermalCube

In a first validation step the Thermal Cube design is compared to the block model of HotSpot in a manual matter. Therefore one node of the HotSpot Block Model is represented by one Thermal Cube. The thermal conductivities in connection with one node, its thermal capacitance towards ambient, the power consumption if any, and the adjacent temperatures of its neighbours are taken as input for the thermal core function, see Equation 4.16.

Instead of calculating the values, HotSpot is extended by an export function that writes the desired data into separate text files allowing an easy import into the thermal cube model. In Table 5.1 the thermal configuration with the package dimension used in the first validation step are listed. The thermal properties together with the thickness of each layer are taken from the HotSpot configuration file, whereas the side dimension of each layer is chosen freely.

For this experiment a constant power dissipation of $400mW$ over a duration of $20s$, a μChip

Table 5.1: Thermal Properties and Package Dimensions.

μChip (Silicon)	Thermal Conductivity	100 $Wm^{-1}K^{-1}$
	Heat Capacity	$1.75\times10^6 Jm^{-3}K^{-1}$
	Thickness	150 μm
	Side	30mm
Interface (Epoxy)	Thermal Conductivity	4 $Wm^{-1}K^{-1}$
	Heat Capacity	4.0×10^6 $Jm^{-3}K^{-1}$
	Thickness	20 μm
	Side	30mm
Spreader (Copper)	Thermal Conductivity	400 $Wm^{-1}K^{-1}$
	Heat Capacity	3.55×10^6 $Jm^{-3}K^{-1}$
	Thickness	1000 μm
	Side	90mm
Spreader to Air	Convection Resistance	40 KW^{-1}

with one core dissipates, is assumed. The evolved heat is removed from the spreader surface by means of convection expressed by a convection resistance R_{convec}. To gain a distinguished temperature rise within the simulation time, the resistance was set to 40 KW^{-1}. Simulating with a sampling interval of 5×10^6 the DUT results in two temperature curves shown in Figure 5.1 beginning at room temperature of 23 °C until reaching the steady state of 39.47 °C. The Deviation between Hotspot and ThermalCube lies between 0 and 3×10^{-3} whereas the timing comparison yields a speed up from 79.93 to 44.94 seconds.

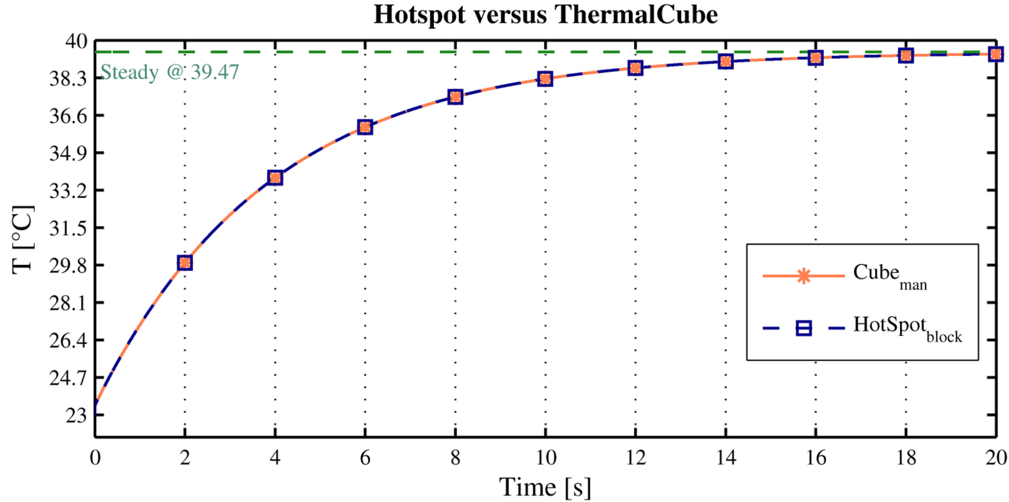


Figure 5.1: Hotspot Block versus ThermalCube Manual.

5.2.2 Grid Model versus ThermalCube

In the next set of experiments the final version of the simulator is compared to HotSpot's Grid model using the same setting as in 5.2.1 with the exception of calculating the convection resistance rather than assuming a fixed value. Determining the resistance is done at each time point during execution due to the dependency of the varying surface temperature. To gain an equivalent resistance to the ambient air the spreader's emissivity value was set to em = 47.879 such that the depending average heat transfer coefficient corresponds to

$$h_{av} = \frac{1}{R_{convec} \cdot A_{surface}} = \frac{1}{40 \cdot 81 \times 10^{-6}} = 308.642\,[Wm^{-2}K^{-1}] \tag{5.1}$$

Similar to the Grid model, which enables the change of the die floorplan roughness to gain higher accuracy, the meshing algorithm of the thermal simulator allows to define a specific mesh size for a particular area. In this experiment the size of the grid and the mesh size of the chip area are set to 16x16. Due to the mesh algorithm and the possible package build-up - each layer has the same mesh and the aspect ratio - the number of nodes used in the calculation is higher. In detail the core area is meshed into a 16x16 grid leading to a 18x18 mesh size for the die layer, see Figure 5.2. The three layers 972 (18x18x3) nodes have to be calculated in comparison to 772 nodes(16*16*3+4). Running

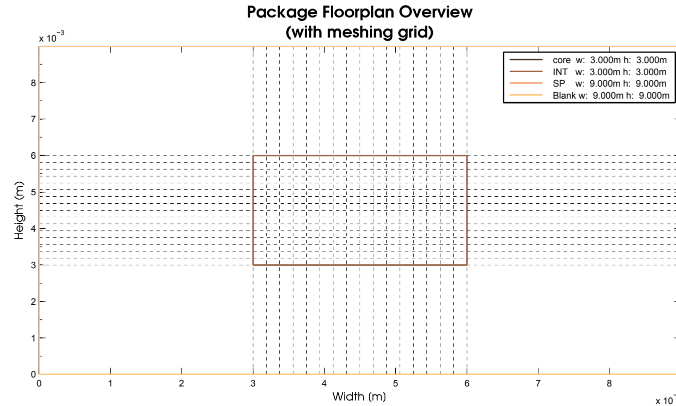


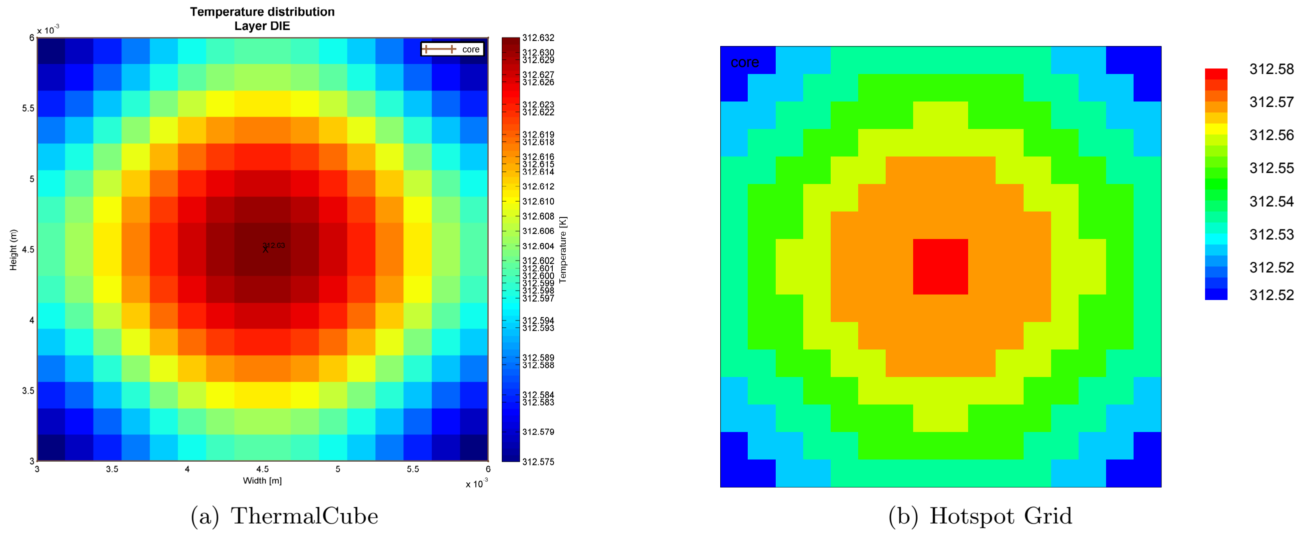
Figure 5.2: 16x16 Mesh of the DIE layer in ThermalCube.

the steady state simulation results in Figure 5.3 with low divergences, see Table 5.2.

Table 5.2: Comparison of the Steady State Temperatures.

Simulator	T_{max}	T_{min}	Diff.
ThermalCube	312.632	312.575	0.057
Hotspot Grid	312.58	312.52	0.06
Diff.	0.052	0.055	

The second experiment validates the transient simulation results, see Figure 5.4, showing



(a) ThermalCube (b) Hotspot Grid

Figure 5.3: The core's steady state temperatures.

a maximum deviation of −0.16 °C. Although the thermal simulator uses 200 more nodes, the simulation duration of 817[s] lies clearly below than that of Hotspot with 7300[s].

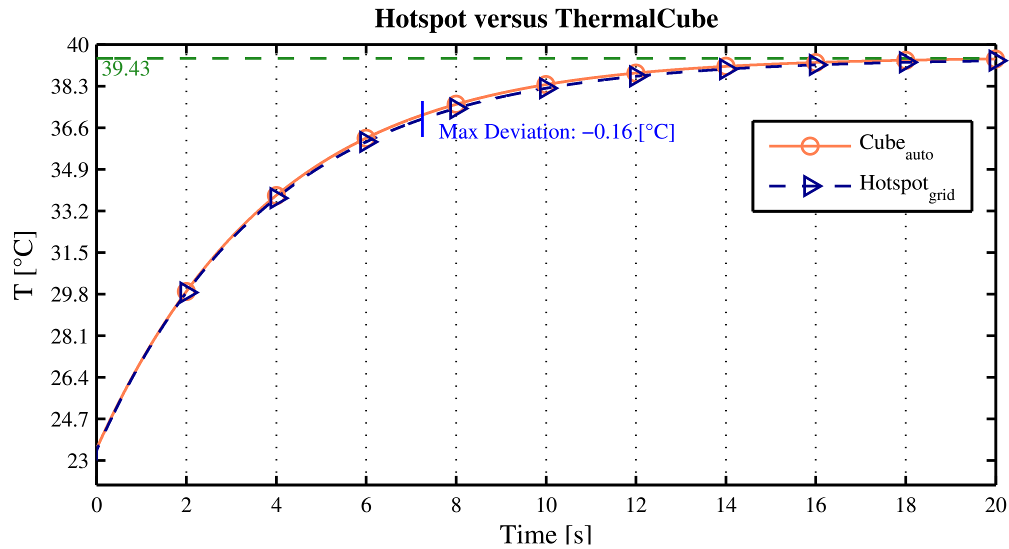


Figure 5.4: Hotspot Grid versus ThermalCube.

5.3 Speedup by Optimised Algorithm

The focus in the development of the program relies on two considerations: Accuracy and performance enhancement. From program version v1.1 on the mesh algorithm divides the assembly into a rectilinear grid. With a higher grade of granularity the simulation time rises, thus optimising the algorithm in terms of computational effort is necessary. In the following the performance of the simulation at different stages of the development is shown by simulating the transient thermal behaviour of the contactless-only smartcard setup illustrated in Figure 5.5.

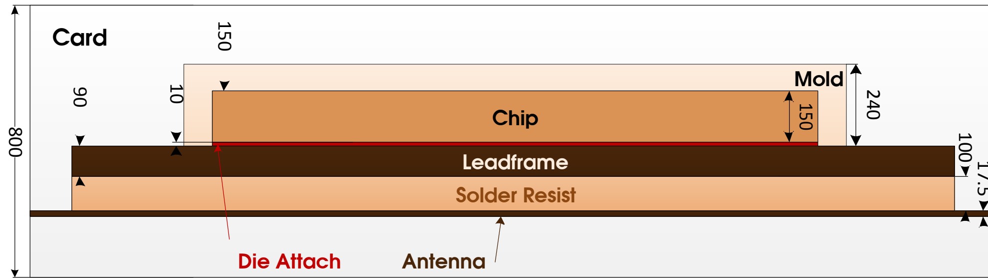


Figure 5.5: Schematic of the contactless-only smartcard.

As load case a static power dissipation of 500 [mW] over 2 [s] is used. Depending on the implementation, cooling is done by means of convection or/and radiation at a room temperature of 23 C. The used package components and material properties are listed in Table 5.4.

Table 5.3: Performance Speedup.

Simulated Time	Simulation Time	δt	Version	rad|conv
2[s]	2720[s]	1.00×10^{-5}	v1.1	no | fix
	1436[s]	1.00×10^{-5}	v1.2	no | fix
	954[s]	1.00×10^{-5}	v1.3	no | fix
	16.15[s]	1.00×10^{-5}	v1.4	no | fix
	20.598 [s]	1.00×10^{-5}	v1.5	yes | fix
	17.215 [s]	1.00×10^{-5}	v1.6	yes | auto

Table 5.3 shows the performance of the different programm code versions. The first basic approach implemented in version 1.1 invokes the thermal core function, see Equation 4.16, for each cell at each point in time with the help of three nested for-loops. The thermal cube core is implemented as M-file function producing some overhead every time is invoked due to the calling mechanism in MATLAB. To avoid this overhead, the function call is replaced in version 1.2 by its underlying code leading to a 1.9 time faster execution time. During each loop iteration conditional assignments are performed to determine the neighbour temperatures and connected conductances of each cell taking into account that

the assigned values are pre-estimated outside the nested loops. By arranging the required values in a matrix and vectors and accessing them by an index vector in version 1.3 the performance is improved by 1.5 times. Taking advantage of the MATLAB design by means of vector and matrix operations the core algorithm is reduced to the outer timing loop in version 1.4 that barely differ from the final version except the implementation of the cooling mechanism. The use of the vectorisation improves the execution time by factor 50. This performance gain is somewhat reduced by the implementation of the radiation mechanism of the subsequent version. At the same time version 1.5 is the last that uses a fixed value for the convection mechanism. The final version as described in Section 4.3, uses the surface temperature results from the previous time step for calculating the convection together with the radiation resistance at each point in time. As it is expected, the additional computational effort limits the performance gain as compared to version 1.4 but was increased against 1.5 by eliminating last performance bottlenecks.

Table 5.4: Package components and material properties.

μChip (Silicon)	Thermal Conductivity	148 $Wm^{-1}K^{-1}$
	Heat Capacity	$1.75\times10^{6} Jm^{-3}K^{-1}$
	Size	5.5x4.2x150 mm^3
Die Attach (Epoxy)	Thermal Conductivity	0.2 $Wm^{-1}K^{-1}$
	Heat Capacity	1.2×10^{6} $Jm^{-3}K^{-1}$
	Thickness	10 μm
Mold Compound	Thermal Conductivity	0.9 $Wm^{-1}K^{-1}$
	Heat Capacity	1.85×10^{6} $Jm^{-3}K^{-1}$
	Thickness	80um above die
Lead Frame (CuSn6)	Thermal Conductivity	65 $Wm^{-1}K^{-1}$
	Heat Capacity	3.31×10^{6} $Jm^{-3}K^{-1}$
	Thickness	90um
Solder Resist	Thermal Conductivity	0.26 $Wm^{-1}K^{-1}$
	Heat Capacity	1.2×10^{6} $Jm^{-3}K^{-1}$
	Thickness	100um
Antenna (Cu)	Thermal Conductivity	388 $Wm^{-1}K^{-1}$
	Heat Capacity	3.45×10^{6} $Jm^{-3}K^{-1}$
	Thickness	17.5um
Card Body (PVC)	Thermal Conductivity	0.16 $Wm^{-1}K^{-1}$
	Heat Capacity	2.48×10^{6} $Jm^{-3}K^{-1}$
	Size	54*85.6*0.8mm^3
Radiation	Emissivity	0.92 (PVC)

5.4 Validation of ThermalCube with FEM Solver Ansys

The ThermalCube simulator is primary evolved to simulate the transient and steady-state temperature behaviour of embedded system such as smardcards. To validate the functionality of the simulator in case of a smartcard build up, a contactless-only and a dual interface card are tested against the commercial FEM simulator ANSYS. Both system consist of a security controller, the module (or package) and the PVC card.

The security controller is partitioned in three areas the analogue IC (dif), the digital IC (core) and the memory (mem) as a coarse schematic of the real layout as seen in Figure 5.6. Except memory, each circuit is assumed to consume constant power over time. For the analogue and digital share of the overall power different load cases covering the range from minimal to maximal overall power are defined, see Table 5.5. For the following experiments the ambient and initiate temperature is set to 25°C (295K).

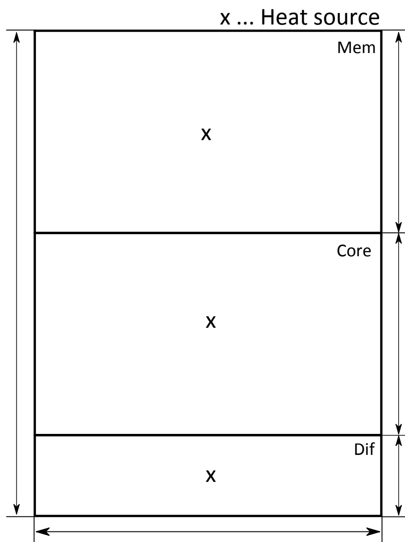


Figure 5.6: Security controller schematic.

Load case	Dif	Core	Mem
1	76	24	0
2	226	24	0
3	376	24	0
4	99.68	0.32	0
5	249.68	0.32	0
6	399.68	0.32	0

Table 5.5: Load Cases [mW].

5.4.1 Contactless Only Module COM

The first set of experiments are performed using a contactless-only chip card setup of the company Infineon. The setup consists of a security controller attached to the module carrier of the wire bond mold package P-COM-2-3 [2],see Figure 5.7(a) – (b), fully embedded in a PVC matrix.

The μChip is typical attached at the the module carrier by an epoxy adhesive and enveloped in the mold compound as shown in Figure 5.7(d). The module carrier consists of three electric isolated parts made of bronze, see Figure 5.7(c), whereas the middle one carries the μChip and the wings function as antenna contacts. Besides the mechanic and electric characteristics bronze exhibit a fine thermal conductivity. Thus, the heat will conduct mainly from the chip trough the module carrier while the PVC matrix serve as good isolator.

The physical dimensions of the module carrier and the mold compound are modelled on the original CAD drawing of the package also the ANSYS simulation is based on. Due to the rectilinear representation the model in Figure 5.7(c) is only an approximation of those used in the reference simulation. In total the setup is constructed as a 6 layer package consisting of 20 component parts belonging to the carrier, one component describing the mold compound above the chip, the chip (die) itself, the die attachment and two components defining the upper and the beyond card body, respectively.

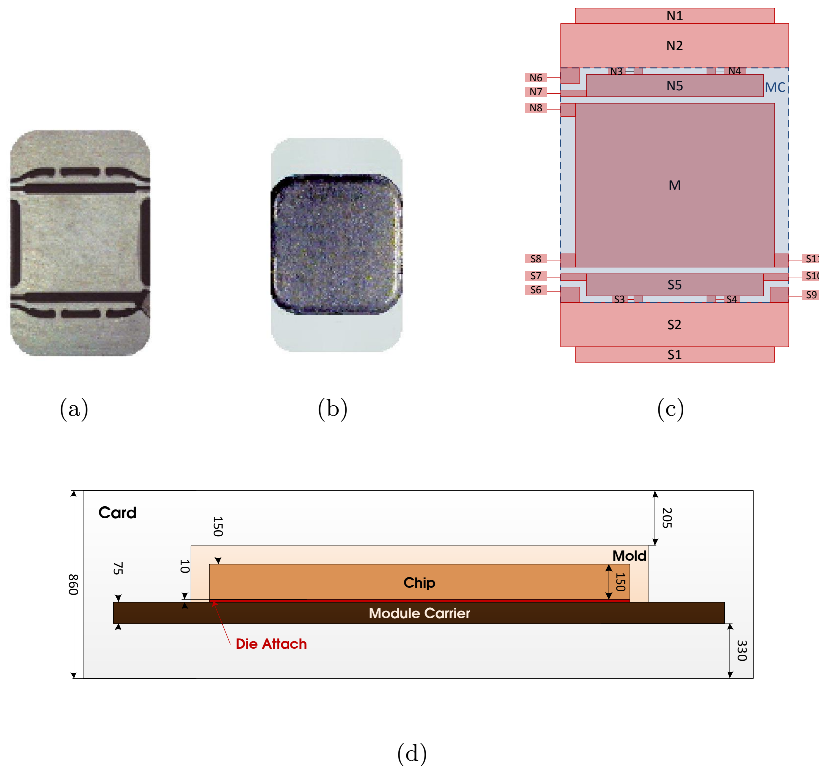


Figure 5.7: COM Setup: (a) and (b) are the real modules, (c) is the representation used in the simulation and (d) is the cross section of the smartcard.

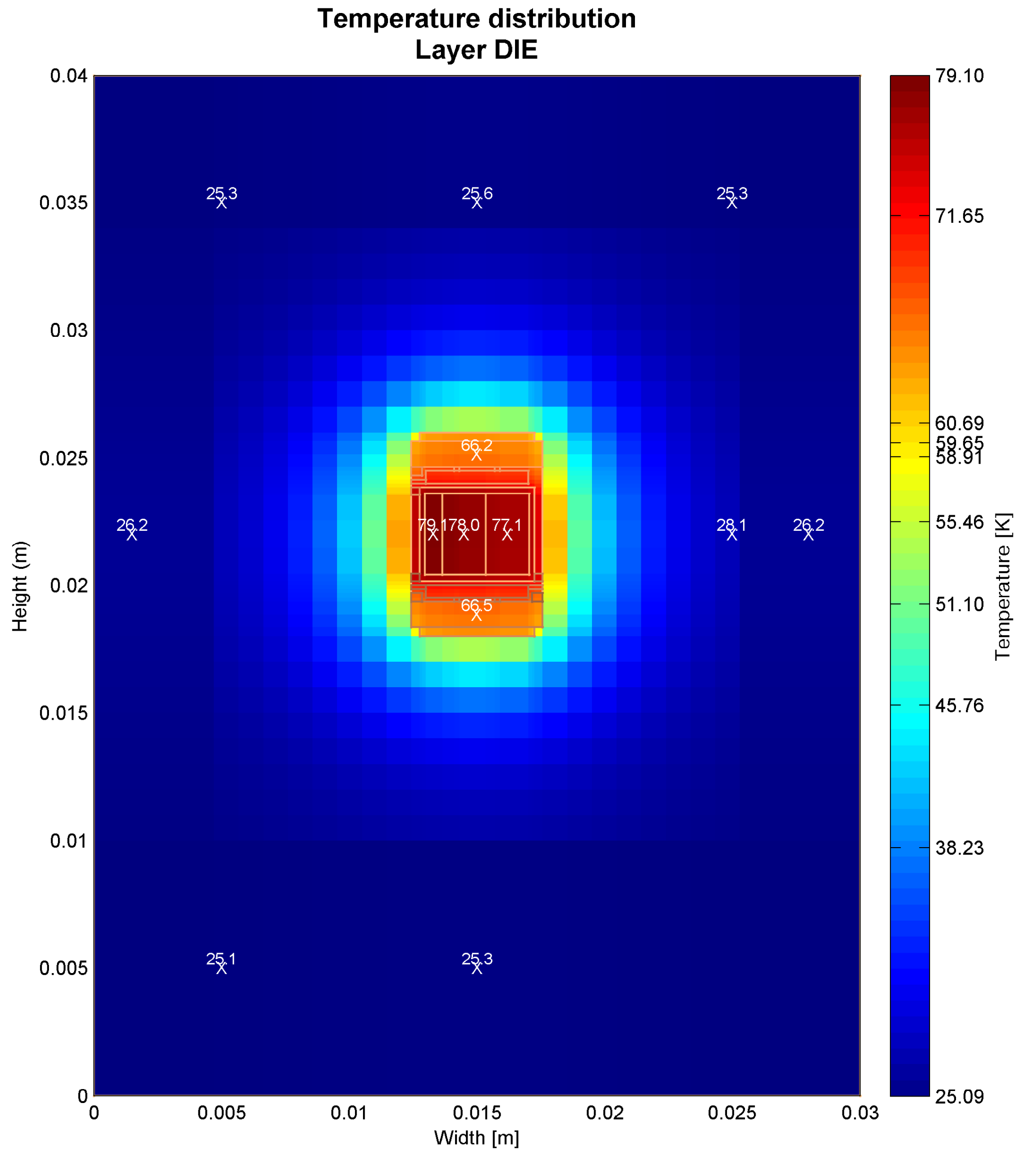
As introduced in 5.4 the μChip is divided into three parts which dissipate power and hence act as heat sources. The load case 1 and 6 used for that heat sources in the following experiments are taken from Table5.5. Before the actual simulation is started the granularity of the meshing is adjusted. Since the basic mesh algorithm only generates edges that result from the defined package fields the area around the package, parts of the PVC card, would be inadequate fragmented. To avoid a raise in inaccuracy an area outside the package outline and its meshing is defined as described in Chapter 3.4. The area extends from the package outline to 1.5 times the width of the package in west and east and 1.0 times the height in north and south direction, whereas the corresponding mesh size is set to 9 in all directions. Additional the chip fields and the fields N2 and S2 are vertical segmented by the factor of three and two, respectively. As a consequence a mesh grid of size 33 x 41 is generated.

The first experiment simulates the steady state of the package for the power load case 1, resulting in Figure 5.8(a) and is compared to the reference simulation in Figure 5.8(b). The Validation results show a absolute deviation of 2.02 °C for the DIF, 1.00 °C for the CORE and 0.70 °C for the MEM with a total simulation time of about 637 seconds (6 iterations).

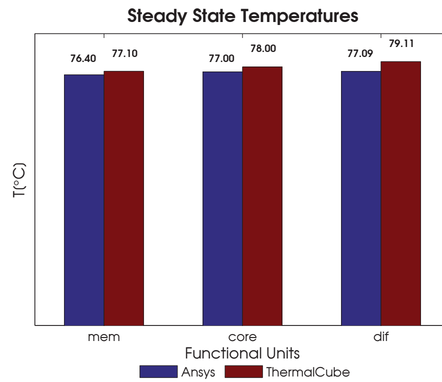
As a second experiment the validation of the transient behaviour under the load case 1 is performed. Therefore the time duration to be simulated was set to 500 seconds where the thermal equilibrium will be reached about. The results, listed in Figure 5.11, show a similar temperature deviation of each unit at steady state compared to the experiment before. Although the equality of the power consumption with its average increases the accuracy of the steady state estimate, the experiment shows the correct working of the iterative approach. The MEMs greater deviation at the time step of 20 seconds is affected by the lumped capacitance modelling, due to their missing lateral components. The total simulation duration is about 7 hours and 19 minutes by a sampling interval of 1.43E-05 seconds.

In contrast to load case 1, which represents the minimal power consumption with a higher consumption of the core, the load case 6 used in the next experiment indicates the idle state of the core under consideration of a simultaneous maximal power consumption of the whole chip. Due to the higher power consumption and hence the resulting higher temperatures, the absolute deviation is higher than that of the lesser power consuming experiment above. As the highest amount of power is dissipated by the DIF unit, also the deviation of about 6 °C is the highest in comparison to the other units, whereas both deviations are < 1.00 °C, as shown in Figure 5.10(b). The time duration of the simulation is about 18.6 minutes after 10 iterations. The large number of iterations results from the higher surface temperatures that affects the calculation of the heat transfer coefficient. In the thermal plot in Figure 5.10, which shows the temperature distribution at the die (chip) layer, the great isolation property of the PVC card can be seen as the typical circular flow of the temperature gradient.

The last experiment in validating the COM package compares the transient behaviour during the load case 6. As the beforehand experiment the temperature progress over 500s is simulated. The deviation of the temperatures at end of the simulated time is again nearly equal to those determined in the steady state simulation. Similar to load case 1 a greater derivation after 20-30 seconds of both the MEM and the CORE units can be observed. Needless to say the time duration of the simulation equals nearly the duration



(a) Thermal plot of the die in steady state.



(b) Validation result.

Figure 5.8: Load case 1: COM package with 4.1 x 3.2 mm$^2\mu$Chip in a PVC ID-1 Card.

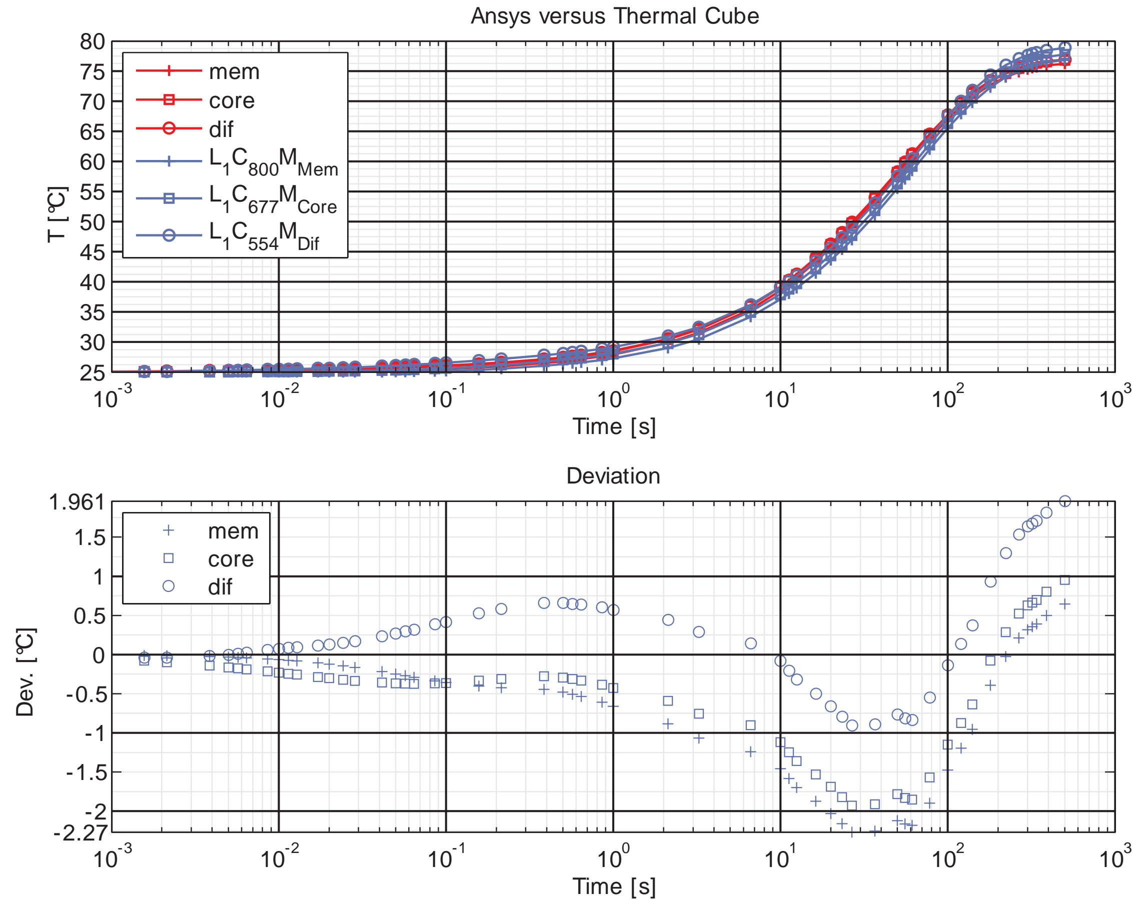
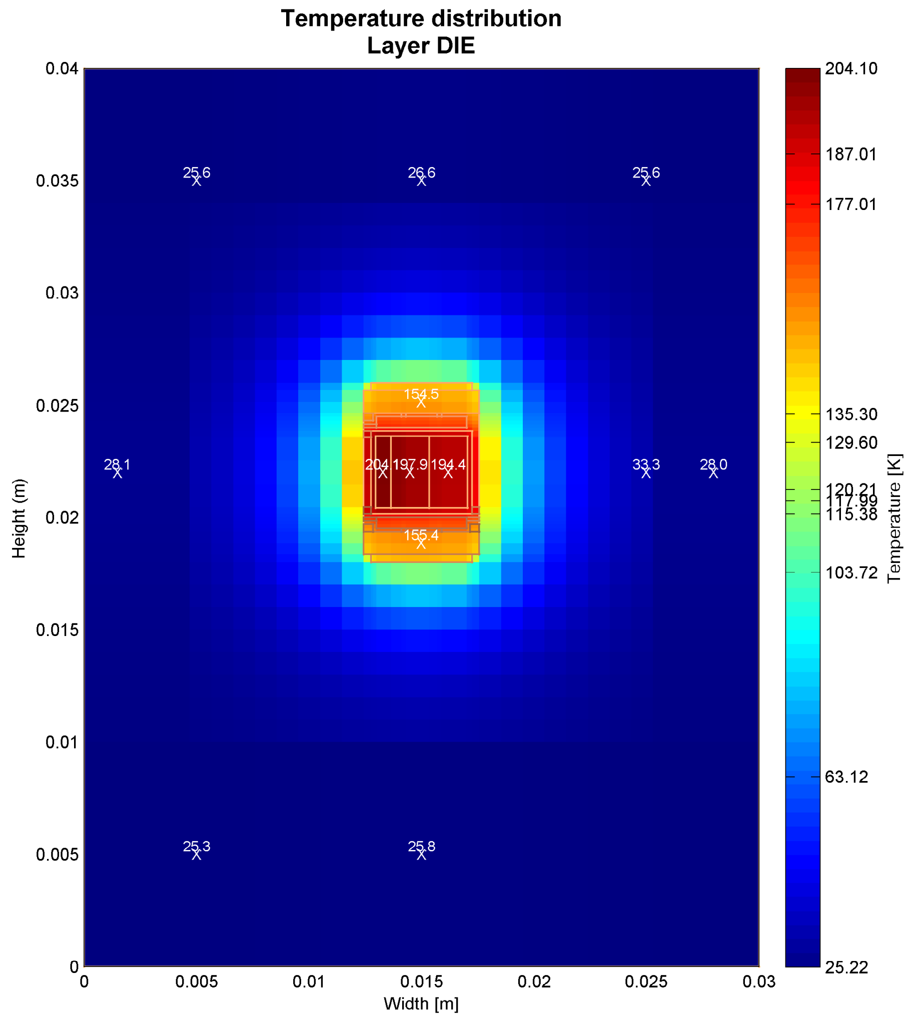
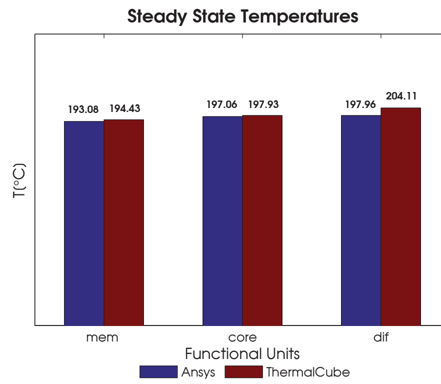


Figure 5.9: Load case1: Transient behaviour of 4.1 x 3.2 mm$^2\mu$Chip in COM package enveloped in a PVC ID-1 Card.

of load case 1.



(a) Thermal plot of the die in steady state.



(b) Validation result.

Figure 5.10: Load case 6: COM package with 4.1 x 3.2 mm^2 μChip in a PVC ID-1 Card.

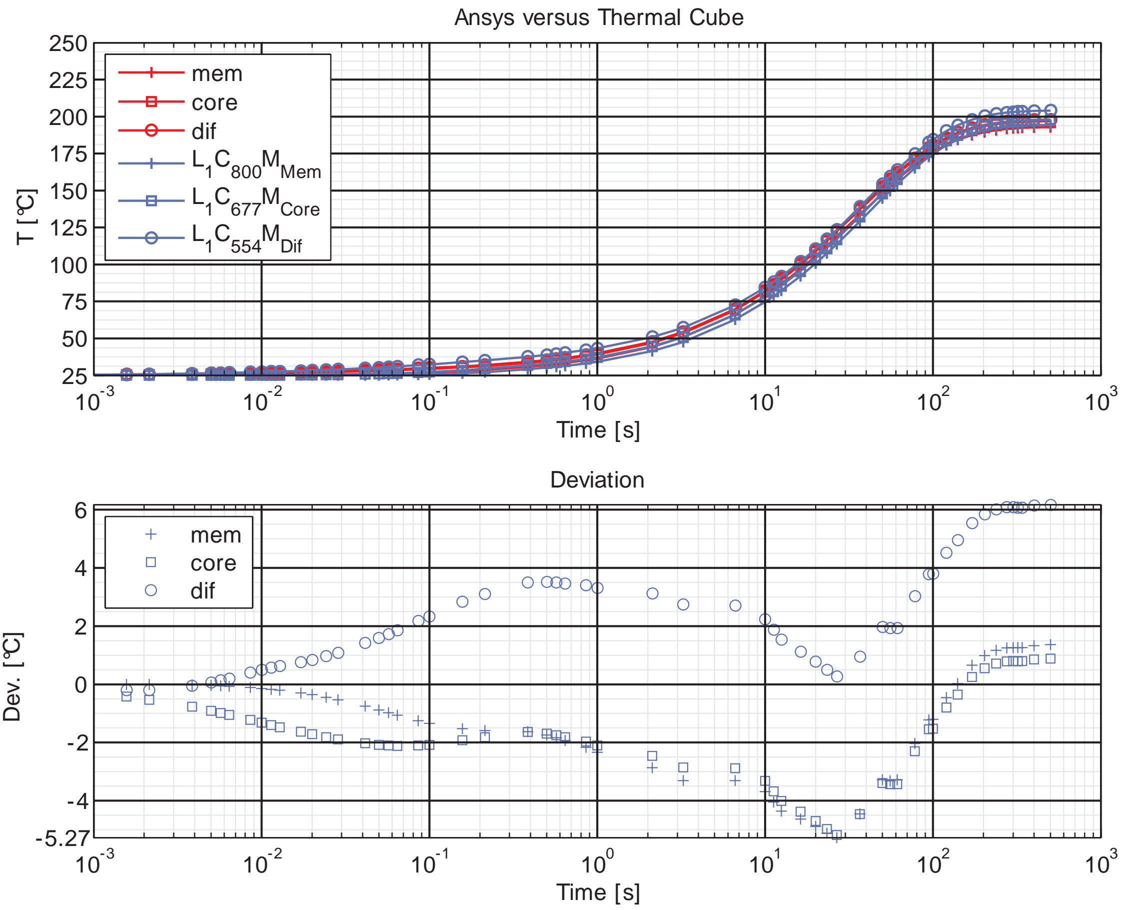


Figure 5.11: Load case6: Transient behaviour of 4.1 x 3.2 mm^2 μChip in COM package enveloped in a PVC ID-1 Card.

5.4.2 Dual Interface Module DIM

The next set of experiments are performed using a dual-interface chip card setup of the company Infineon. The setup consisted of a security controller attached to the module carrier of the wire bond glob top package T-M8.4 [2],see Figure 5.7(a) – (b), embedded in a PVC matrix whereas one side stays in contact with the ambient.

The DIM package is suitable for a dual interface controller obtaining the power from the electromagnetic field or via galvanic contacts (ISO side) in connection with a smartcard reader. The μChip is attached at an epoxy tape (carrier) by an epoxy adhesive. In addition two metal contacts are assigned located around the contact area of the μChip acting as antenna pads, as illustrated in Figure 5.12(d). To enable the data and power transfer of the ISO contacts with the μChip parts of the antenna pads as the carrier are constructed such that several openings extend through the carrier to the ISO contacts. In Figure 5.12(b) the model of the metal pads with the openings ($N_3 ... N_6$ and $S3 ... S6$) together with the area covered by the protective glob top are illustrated. With the ISO contact component of 36 parts, see 5.12(a), the metal pads divided into 30 parts, the μChip and the card body, the DIM package is described as a 6 layer build up (see cross section 5.12(c)).

Similar to the COM experiments the same schematic of the floorplan for the μChip with the load case 1 for its power dissipation is used in the following experiments. At the beginning the granularity of the meshing is adjusted such that an area outside the package is additionally fragmented. The area extends from the package outline to 0.5 times the width and the height of the package in all four directions with the corresponding division set to 5 leading to a mesh grid of size 52 x 47. The first experiment simulates the steady state of the security controller of size 4.6 x 5.5 under load case 1 condition. The result, as illustrated in Figure 5.13(a), is obtained after 6 iterations in 48.8 minutes. The validation with the results from the ANSYS reference simulation shows a greater deviation as the experiments before, as shown in Figure 5.13(b). With a raise of the meshing grid the difference can be lowered, but in the same matter the duration of the simulation time increases. However, a huge problem in the modelling of this package comes from the complexity of the build up itself going along with a higher possibility to produce inaccuracies. It turns out from the distribution of the temperature along the direction of the CM and C5 parts as well as from the isolating properties of the PVC card that direction of the temperature flow of higher magnitude is correct.

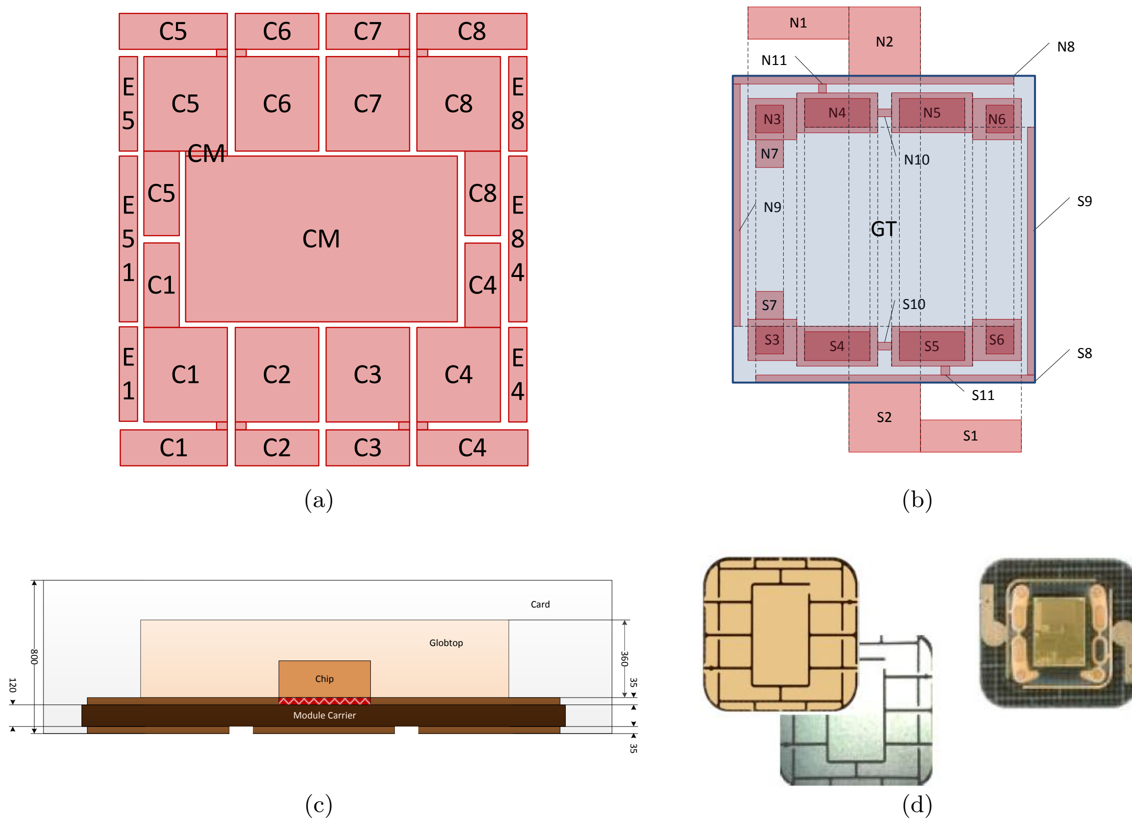
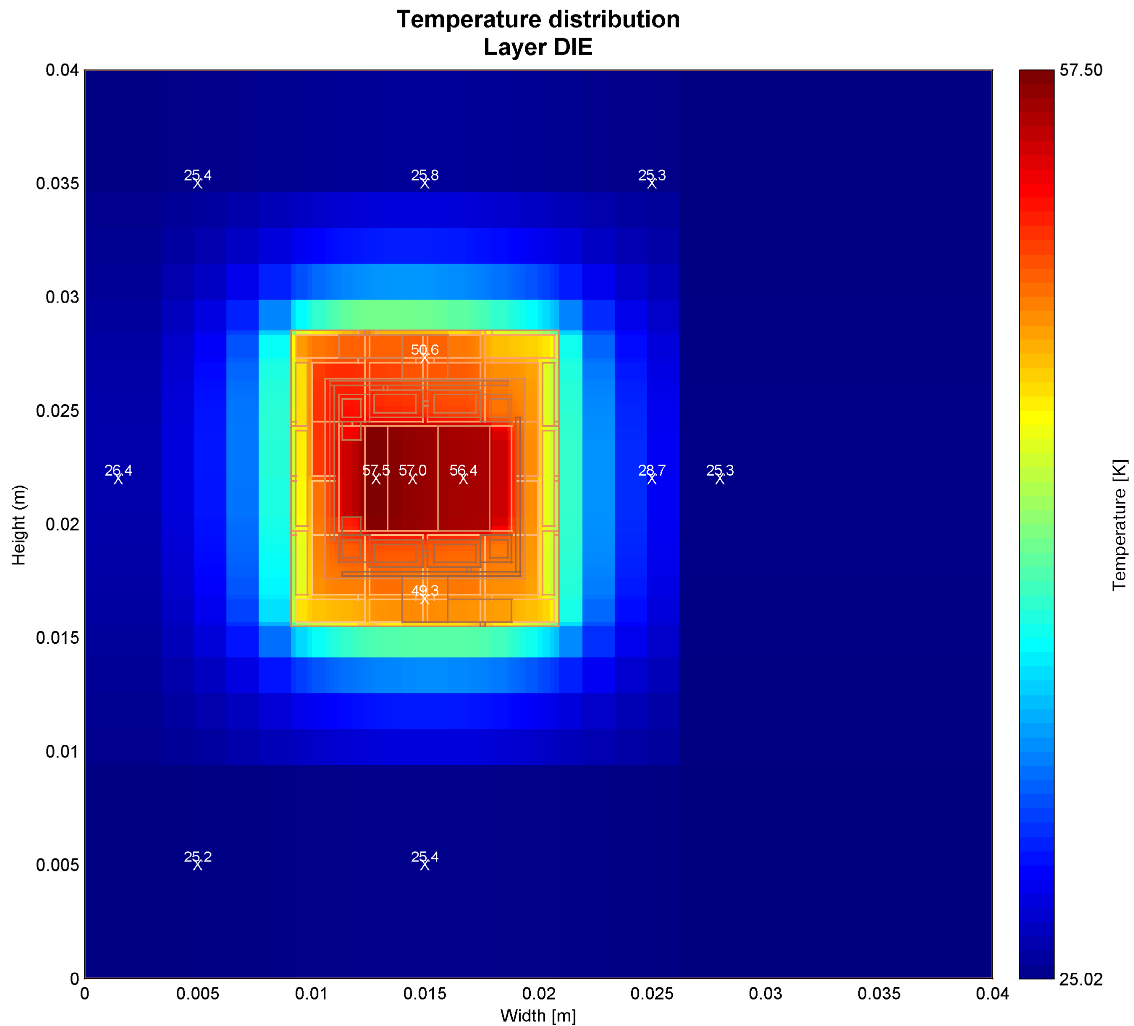
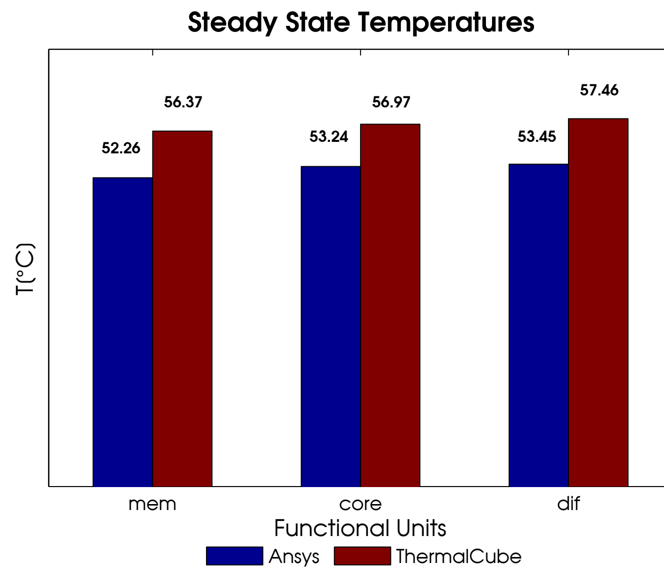


Figure 5.12: DIM Setup: (d) shows the real module (ISO and Chipside), (a) and (b) are the modelled ISO contact and chip side, and (c) is the cross section of the dual interface smartcard.

Although the area of the μChip is greater than the one used in the COM experiments a better thermal performance can be expected. The difference of $20°C$ can be reduced to a better cooling due to the heat path through the ISO contacts against the ambient. The second experiment validates the transient behaviour over a simulated time of 500 seconds. As expected, the end temperatures equals the steady state temperatures. Similar to the experiments with the simpler COM package a higher deviation at time steps between 20 to 30 seconds can be observed. Due to the higher amount of elements to calculate, the simulation duration is about 11.8 hours.



(a) Thermal plot of the die in steady state.



(b) Validation result.

Figure 5.13: Load case 1: DIM package with 4.6 x 5.5 $mm^2 \mu$Chip in a PVC ID-1 Card.

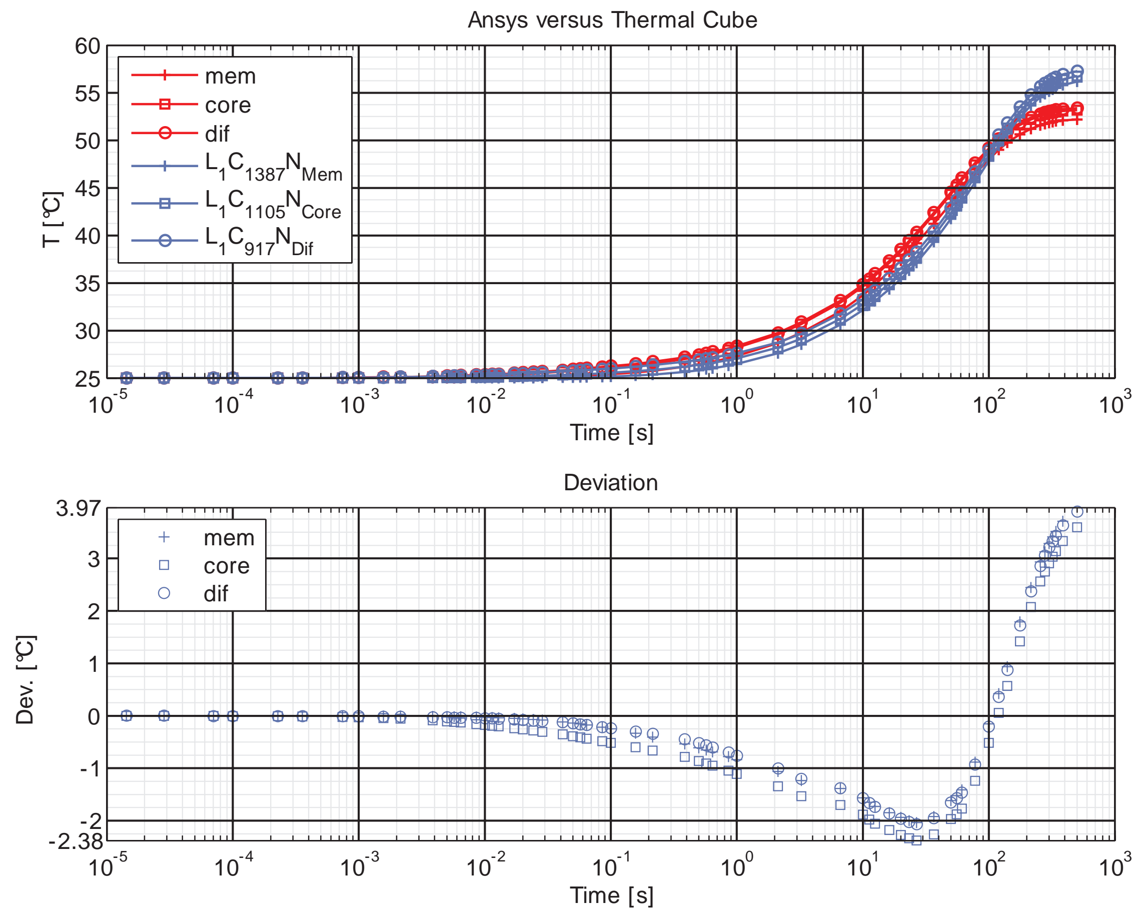
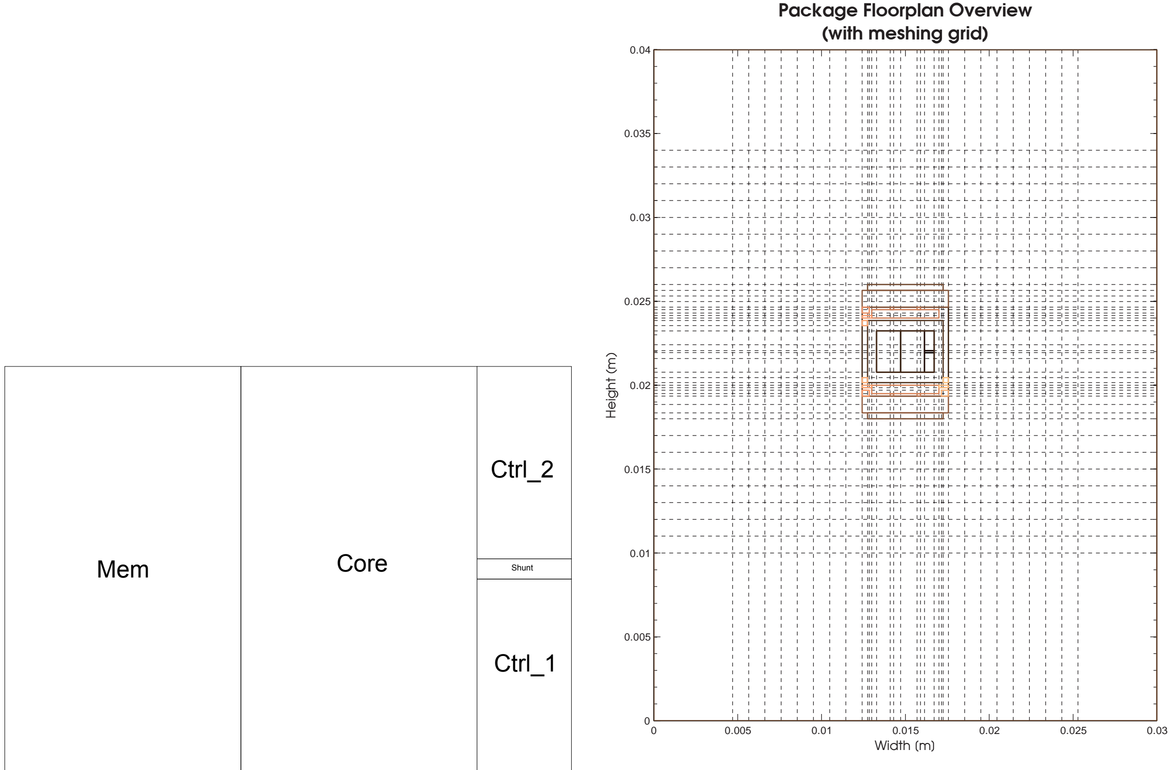


Figure 5.14: Load case1: Transient behaviour of 4.6 x 5.5 $mm^2 \mu$Chip in DIM package enveloped in a PVC ID-1 Card.

5.5 Thermal Simulation with PE power estimates

In contrast to the experiments in the last section, where manual predefined power load cases are used to produce power traces, the power profile in the following use case is obtained from a test system for Chip and Security ICs at INFINEON Technologies in Graz, augmented by a power emulation unit, and fed to the thermal simulator.



(a) Floorplan of C2012 security controller.

(b) Package floorplan of COM and security controller C2012 with mesh.

Figure 5.15: The chip floorplan (a) and the package floorplan (b).

The μChip used in this experiment is the C2012 security controller, which consists of an analogue frontend as implemented in Section 3.3, the core and the memory, shown in Figure 5.15(a). Although the C2012 controller is suitable for dual interface applications the contact-less only package COM was chosen due the better validation results. The meshing of the this packaging results in the grid shown in 5.15(b).

The running application on the security controller is an arithmetic benchmark test that is repeated as many times until a certain time is reached, in this case the application is assumed to be interrupted after 1 second. The antenna configuration consists of the inductivity $L_2 = 2.3\mu H$, the tag area $A_{tag} = 0.004m^2$, and the number of windings $N_{tag} = 4$, whereas the analogue characteristic of the chip is defined as the total input capacity $C_{tot} = 0.6pF$, the digital IC voltage V_{DD} and the input voltage U_{LA}. With a maximal allowed field strength of $12Am^{-1}$ and the system frequency of $f_{sys} = 13.56MHz$ the power model calculates the overall power consumption. Together with the power trace

from the emulator the power profile of the remaining units are derived.

In an first experiment the transient behaviour of the security controller during a time period of 1 second is simulated. For the sampling interval of the the thermal simulation, the same interval from the benchmark application is taken automatically. According to the frequency of the power values in the trace of about 30 MHz the sampling interval is set to 3e-008 seconds. As a consequence, the result, as shown in Figure 5.16, is obtained after 6 hours. The temperature distribution of the die layer after 1 seconds is illustrated

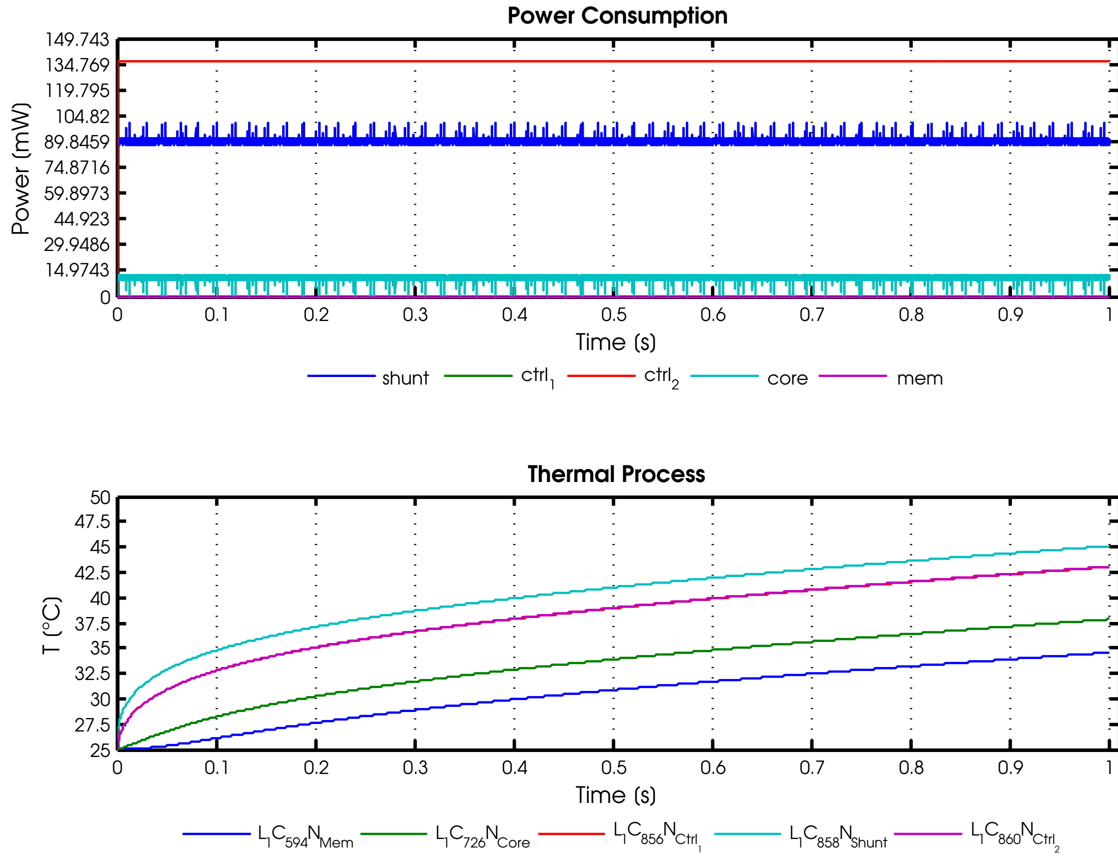
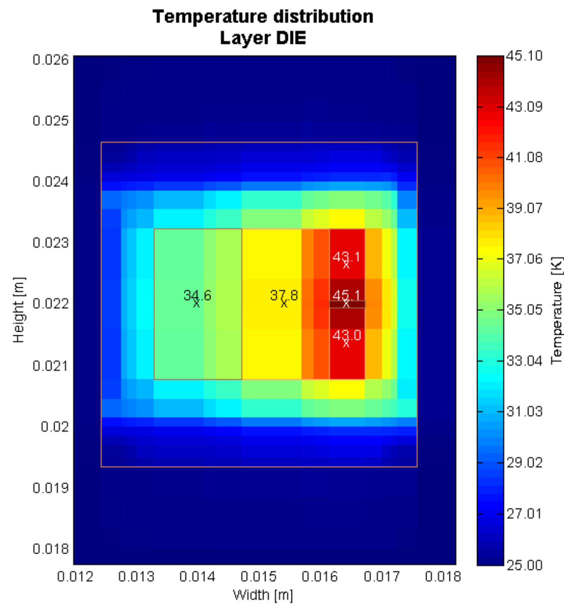
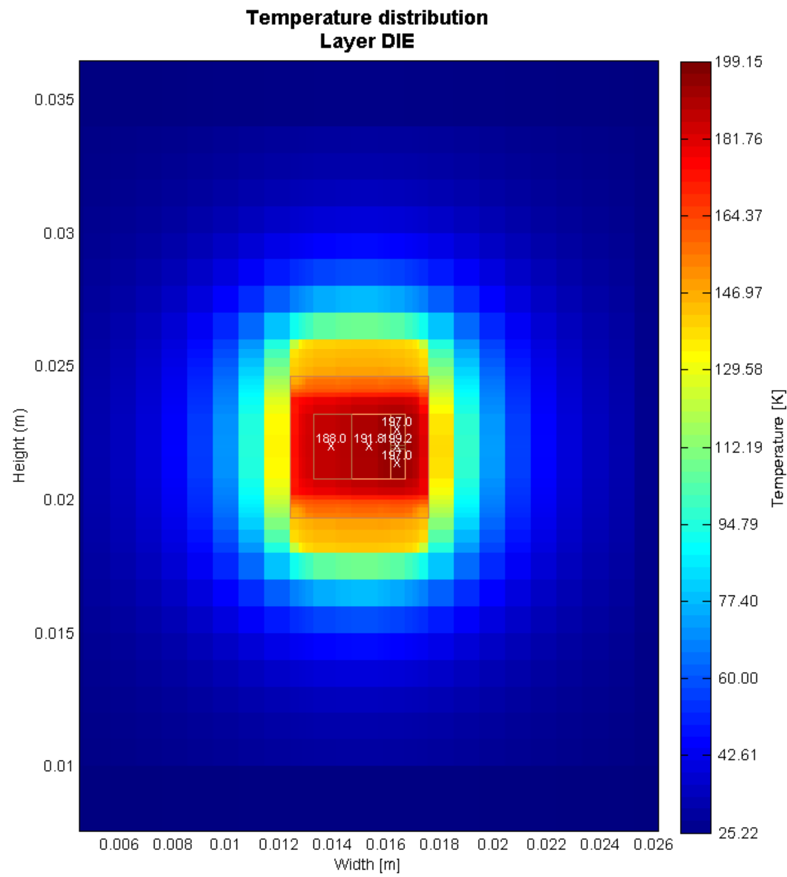


Figure 5.16: Transient behaviour of the C2012 controller during core performing an alu arithmetic benchmark test. The total power is gained from the electromagnetic field with $\mathrm{H}_{\max} = 12\left[\mathrm{Am}^{-1}\right]$.

in Figure 5.17(a) where a hotspot at the shunt can be identified. The second experiment simulates the steady state when the package stays in thermal equilibrium. Therefore the power traces are averaged before the actually simulation is performed. The maximum Temperature of $199.15°C$ can be observed at the shunt. The high temperatures of the simulation are obtained as a result of the missing antenna in the package model and of the assumption that the security controller gains maximal power over the whole time until the thermal equilibrium is reached.



(a) Temperature patch plot of die layer after 1 second.



(b) Temperature patch plot of die layer when package is in steady state.

Figure 5.17: Temperature distribution (a) after 1 second (b) in steady state.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Within the thesis, the design and implementation of a thermal simulator in MATLAB suitable for temperature evaluation of an embedded system on chip such as a smartcard system is illustrated. A thermal simulator has been implemented that is able to analyse the transient behaviour and the steady state temperatures of a DUT. The necessary power information for the digital IC in form of run-time power estimates is gained from the power emulation platform and used to determine the power consumption of the whole chip including both the analogue and digital IC's.

For determining the power consumption of the analogue IC, the power model is described as a current network. The power consumption of each analogue unit can be determined by the current flow and the corresponding voltage. In knowing the input current flow of the analogue circuit and the power (current) traces from the emulator the remaining current consumption can be determined.

The introduced packaging model enables the simulation of arbitrary packaging designs with different structures and material properties allowing the exploration of new designs, both of the μChip and packaging during the whole development process. The floorplan of the μChip is defined separately from the package to make different configurations possible. The semi-automatic meshing algorithm allows the user to define regions of lesser granularity to achieve reasonable accuracy.

The functional unit names and dimensions are automatically extracted from the floorplan file. The power profiles gained from the power model are then assigned by name to the corresponding units. Besides using the power emulation mode, a manual approach is implemented allows the defining of power profile for each functional unit by hand for validation purposes or performing simple load case experiments.

The interaction with the environment is implemented as heat transfer coefficient which is determined at each iteration. The calculation of the coefficient can easily be exchanged by any desired method, making the thermal model boundary independent. In the current implementation, the heat transfer coefficient is determined as combination of the convectional and radiation heat transfer depending on the ambient and surface temperatures. While the transient simulation uses the heat transfer coefficient at each time step, the steady state model uses an iterative approach to incorporate the heat exchange with the

ambient.

In experiments, the correct functionality of the thermal simulator performing transient and steady state analysis can be shown. Therefore two smartcard packages of different complexities in two different load cases were verified against the commercial FEM solver ANSYS. The less complex package showed a absolute derivation of 1-2 °C in the steady state case and a maximal one of 2.27 °C in the transient case, both under a load case of 100 [mW]. In contrast the more complex package showed a maximal deviation of approximate 4°C. In addition the thermal model was verified against HOTSPOT showing a derivation of $< 0.06\,°C$ in the steady state and $0.16\,°C$ in the transient case.

6.2 Future Work

- **Thermal model in HW**: The simulation results showed that a RC network thermal model based on physical geometries needs a minimum number of cells to gain accurate results, which can be huge when the system is more complex and thus more computational effort is necessary. To improve the simulation time a hardware based thermal model could be implemented that uses the concept of FPGA acceleration of sparse matrix-vector multiplication with Network-on-CHIP (NOC).

- **Leakage Model**: The continuous scaling of the transistor supply voltage has led to an huge increase in subthreshold leakage and thus leading to an increase in temperature. To cover the influence of the leakage to the temperature distribution the thermal model can be extended by determining the subthreshold leakage.

- **Changing the Cell Shape**: The rectangular shape of the cell could be changed to better fit package designs and thus leading to a higher accuracy in modelling.

- **CAD interface**: As the current model of package model has to be defined by hand and most of the package designs are available as CAD drawings, an algorithm could be implemented that parses the CAD design into the internal data representation.

Abbreviations and Symbols

Abbreviations

AC Alternating Current

BEM Boundary Element Method

C Capacitor

CAD Computer Aided Sesign

CB Card Body

CMOS Complementary Metal Oxide Semiconductor

COM Contactless Only Modul

CR Carrier

DA Die Adhesive

DCT Discrete Cosine Transformation

DIM Dual Interface Modul

ECD Equivalent Circuit Diagram

EM Electromagnetic Field

FDM Finite Differenz Method

FEM Finite Element Method

FLP Floorplan

FPGA Field Programmable Gate Array

GT Glob Top

GUI Graphical User Interface

HDL Hardware Description Language

HW Hardware

IC Integrated Circuit

LUP Lower Upper Permutation Decomposition

MEM Memory

NOC Network-on-CHIP

nz Non Zero

ODE Ordinary Differential Equation

OS Operating System

PDE Partial Differential Equation

PVC PolyVinyl Chloride

R Resistance

RF Radio Frequency

RFID Radio Frequency Identification

RK4 Fourth Order Runge-Kutta method

RTL Register Transfer Level

SD Semi Discretisation

SVG Scalable Vector Graphics

VLSI Very Large Scale Integration

Symbols

V_T Threshold Voltage

v_t Thermal Voltage

α Thermal Diffusivity

β Volume Expansion Coefficient

Δt_s Sampling Interval

$\dot{Q}$ Heat Transfer Rate

$\mu(T)$ Mobility Variation with Temperature

ν Kinematic Viscosity

ρ Stefan-Boltzmann Constant

A Area

A_2 Area of Cross-Section of the Coil

C_L Load Capacity

C_p Heat Capacity

C_{th} Thermal Capacitance

f Frequency

f_{sys} System Frequency

g Gravitational Acceleration

Gr_L Grashof Number

h_c Convection Heat Transfer Coefficient

h_c Thermal Interface Conductance

h_r Radiation Heat Transfer Coefficient

I_L Load Current

I_{leak} Leakage Current

k Thermal Conductivity

$k(T)$ Temperature dependent Thermal Conductivity

L_2 Coil Inductance

L_c Characteristic Length

N_2 Coil Windings

Nu Nusselt Number

P Power

Pr Prandtl Number

R_2 Coil Resistance

R_L Load Resistance

R_{th} Thermal Resistance

Ra_L Rayleigh Number

T Temperature

T_a Ambient Temperature

T_s Surface Temperature

U_i Induced Voltage

U_L Input Voltage

V_{dd} Supply Voltage

Bibliography

[1] Delphi compact thermal model guideline, jesd15-4, Oktober 2008.

[2] Infineon Technologies AG. P-com-2-3, 2010.

[3] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid. Accelerating embedded software power profiling using run-time power emulation. In José Monteiro and René van Leuken, editors, *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, volume 5953 of *Lecture Notes in Computer Science*, pages 186–195. Springer Berlin / Heidelberg, 2010.

[4] W. Batty, C. E. Christoffersen, S. David, A. J. Panks, R. G. Johnson, C. M. Snowden, and M. B. Steer. Fully physical time-dependent compact thermal modelling of complex non linear 3-dimensional systems for device and circuit level electro-thermal cad. In *Proc. Seventeenth Annual IEEE Symp. Semiconductor Thermal Measurement and Management*, pages 71–84, 2001.

[5] W. Batty, C. E. Christoffersen, A. J. Panks, S. David, C. M. Snowden, and M. B. Steer. Electrothermal cad of power devices and circuits with fully physical time-dependent compact thermal modeling of complex nonlinear 3-d systems. 24(4):566–590, 2001.

[6] A. A. Becker. *The Boundary Element Method in Engineering : A Complete Course*. McGraw-Hill, 1992.

[7] C. Bohm, T. Hauck, E. B. Rudnyi, and J. G. Korvink. Compact electro-thermal models of semiconductor devices with multiple heat sources. In *Proc. 5th Int. Conf. Thermal and Mechanical Simulation and Experiments in Microelectronics and Microsystems EuroSimE 2004*, pages 101–104, 2004.

[8] Y. A. Cengel. *Heat Transfer: A Practical Approach*, volume 2nd. McGraw-Hill, 2003.

[9] Wai-Kai Chen. *The VLSI handbook*. CRC Press, 2 edition, 2007.

[10] H. Chiueh, J. Draper, L. Luh, and Jr. Choma, J. A novel model for on-chip heat dissipation. In *Proc. IEEE Asia-Pacific Conf. Circuits and Systems IEEE APCCAS 1998*, pages 779–782, 1998.

[11] Joel Coburn, Srivaths Ravi, and Anand Raghunathan. Power emulation: a new paradigm for power estimation. In *Proceedings of the 42nd annual Design Automation Conference*, DAC '05, pages 700–705, New York, NY, USA, 2005. ACM.

[12] Inc. eFunda. Efunda materials home, 2011.

[13] K. Finkenzeller. *RFID Handbook: Fundamentals and Apllications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication.* John Wiley & Sons Ltd., 2 edition, 2010.

[14] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss. An emulation-based real-time power profiling unit for embedded software. In *Systems, Architectures, Modeling, and Simulation, 2009. SAMOS '09. International Symposium on*, pages 67 –73, july 2009.

[15] Wei Huang. *HotSpot. A Chip and Package Compact Thermal Modeling Methodology for VLSI Design.* PhD thesis, School of Engineering and Applied Science. University of Virginia., January 2007.

[16] Wei Huang, K. Sankaranarayanan, S. Velusamy, D. Tarjan, S. Ghosh, J. Burr, R.J. Ribando, M.R. Stan, and K. Skadron. `http://lava.cs.virginia.edu/HotSpot/ (07.02.2013)`.

[17] Ansys Inc. `http://www.ansys.com/(07.02.2013)`.

[18] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations.* Cambridge University Press, second edition, 2009.

[19] M. Janicki and A. Napieralski. Temperature monitoring of electronic circuits based on analytical thermal model. In *Proc. 22nd Int Microelectronics Conf*, volume 2, pages 581–584, 2000.

[20] H. Kaeslin. *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication.* Cambridge University Press, 2008.

[21] S. Kaxiras and M. Martonosi. *Computer Architecture Techniques for Power-Efficiency.* Morgan & Claypool, 2008.

[22] John H IV Lienhard and John H V Lienhard. *A Heat Transfer Textbook.* Phlogiston Press, third edition, 2008.

[23] Erdogan Madenci and Ibrahim Guven. *The finite element method and applications in engineering using Ansys.* Springer, New York, NY, 2006.

[24] C. Piguet. *Low-Power Electronics Design.* CRC Press, 2005.

[25] Gerald W. Recktenwald. Finite-difference approximations to the heat equation[lecture notes], March 2011.

[26] J. N. Reddy. *An Introduction to the Finite Element Method.* McGraw-Hill, 2006.

[27] Kevin Skadron, Mircea R. Stan, Wei Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and David Tarjan. Temperature-aware microarchitecture: Extended discussion and results. 2003.

[28] Inc. The MathWorks. `http://www.mathworks.de/products/matlab/(07.02.2013).`

[29] C.Schenk U. Tietze. *Halbleiter Schaltungstechnik*, volume 12. Springer Verlag Berlin, 2002.

[30] Yong Zhan, B. Goplen, and S. S. Sapatnekar. Electrothermal analysis and optimization techniques for nanoscale integrated circuits. In *Proc. Asia and South Pacific Conf. Design Automation*, 2006.

[31] Yong Zhan and S.S. Sapatnekar. A high efficiency full-chip thermal simulation algorithm. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, nov. 2005.

[32] Zheng Zhu, Behnam Jamali, and Peter H. Cole. An hf/uhf rfid analogue front-end design and analysis. Technical report, Auto-ID Labs, 2005.