MODELING OF THE THERMAL BEHAVIOR OF A POWER ELECTRONIC MODULE

By

Madelaine Hernández Mora

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

MECHANICAL ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2004

Approved by:

______________________________	____________
Nihad Dukhan, Ph.D. Member, Graduate Committee	Date
______________________________	____________
Miguel Vélez Reyes, Ph.D. Member, Graduate Committee	Date
______________________________	____________
Jorge E. González, Ph.D. President, Graduate Committee	Date
______________________________	____________
Jorge A. Cruz Emeric, Ph.D. Representative of Graduate Studies	Date
______________________________	____________
Paul Sundaram, Ph.D. Chairperson of the Department	Date
______________________________	____________
Jose A. Mari Mutt, Ph.D. Director of Graduate Studies	Date

ABSTRACT

This work presents a reduced mathematical model using a practical numerical formulation of the thermal behavior of an Integrated Power Electronics Module (IPEM). This model is based on the expanded Lumped Thermal Capacitance Method (LTCM), in which the number of variables involved in the analysis of heat transfer is reduced only to time. By applying this procedure a simple, non-spatial, but highly non-linear model is obtained. Transient results of the model were validated using FLOTHERM 3.1^{TM}, a thermal analysis software tool. Two experimental set-up, for low- and high-speed thermal response, were developed. Comparisons between thermal model results and experimental data are also presented to demonstrate the need to obtain the electrical performance and to make the electrothermal coupling. The development of this model presents an alternative to reduce the complexity level developed in commercial multidimensional and transient thermal analysis software tools.

RESUMEN

Este trabajo presenta un modelo matemático reducido usando una formulación numérica práctica del comportamiento térmico de un Módulo Integrado de Electrónica de Potencia (IPEM). Este modelo está basado sobre la expansión del Método de la Capacitancia Térmica de un Conglomerado (LTCM), en el cual el número de las variables envueltas en el análisis de transferencia de calor es reducido a solo el tiempo. Aplicando este método, un modelo simple, no espacial, no lineal es obtenido. Los resultados transitorios del modelo son validados contra resultados de un software de análisis térmico, FLOTHERM 3.1^{TM}. Dos arreglos experimentales, para respuestas térmicas de baja y alta velocidad, fueron desarrollados. Comparaciones entre los resultados del modelo térmico y los datos experimentales son también presentados para demostrar la necesidad de obtener el comportamiento eléctrico y hacer el acople electrotérmico en el análisis electrotérmico. Se demuestra en esta investigación que la metodología desarrollada presenta una alternativa para reducir el nivel de complejidad desarrollado en softwares comerciales de análisis térmico transitorio y multidimensional.

DEDICATORIA

A mi amada Fanny, por su perseverante amor, a mi Daniel por iluminarme el camino, a Alexei, Hugo, Jorge y Tony por su valiosa presencia en mi vida.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation Engineering Research Centers Program under grant EEC-9731677. I thank to Dr. Allen Hefner to give me the opportunity to work with him in the Semiconductor Division at the National Institute of Standard and Technology, in a summer internship. I wish to express my most sincere gratitude to Dr. Jorge E. González for his discussions and for giving me the opportunity to work on this research. I sincerely acknowledge the guidance of Dr. Miguel Vélez Reyes and thank him for his advise and support through out the course of this study. I would like to recognize the help and discussions on electrical aspects by my friend José Miguel Ortíz. My special thanks are due to Carlos Alicea for all the help provided and support on Saber circuit simulator and my friends Hermes, Jairo, Zory and Nilsa for their constant encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

NOMENCLATURE

English Characters

A_{cont}	Contact area (m^2)
A_{hor}	Horizontal convection area (m^2)
A_{ver}	Vertical convection area (m^2)
$A_{j\text{-}jnext}$	Radiation area (m^2)
Bi	Biot number
C_{pj}	Specific heat (J/m K)
$F_{j\text{-}jnext}$	Geometry factor for the radiation between lumped
H_c	The surface microhardness of the softer of the two contacting solids (N/m^2)
h_{hor}	Heat transfer coefficient on a horizontal plane (W/m^2 K)
h_{ver}	Heat transfer coefficient on a vertical plane (W/m^2 K)
k_j	Lumped thermal conductivity (W/m K)
k_f	Fluid thermal conductivity (W/m K)
L	Equivalent length (m)
m	Effective mean absolute asperity slope of the interface
M	Fluid parameter (m)
P	Contact pressure (Pa)
q_{gen}	Power dissipated (W)
$R_{cont\ j\text{-}jnext}$	Contact resistance from the contact surface phenomenon (K/W)
$R_{j\text{-}jnext}$	Total thermal contact resistance (K/W)
$R_{spreader\ j\text{-}jnext}$	Heat spreader resistance due the conduction to the contact surface (K/W)
s	Tranversal horizontal or vertical length to the convective flow (m)
T_∞	Ambient temperature (K)
T_{Lj}	Lumped temperature (K)
t	Time (sec)
V_j	Lumped volume (m^3)

x_j	Lumped equivalent length for spreader effects (m)
Y	Effective gap thickness (μm)

Greek Characters

δ	Effective surface roughness of the contacting asperities (μm)
ε_j	Lumped emissivity
ρ_j	Lumped mass density (Kg/m^3)
σ	Stefan-Boltzmann's constant (W/m^2 K^4)

CHAPTER 1

INTRODUCTION

1.1 Introduction

Power electronics may be defined as the application of electronics for the control and conversion of electric power in a form that is optimally suited for user loads (with highest efficiency, high availability, and high reliability with the lowest cost, smallest size and weigh). Power electronics is based on the extensive use of inductors, capacitors and, mainly, of power semiconductor devices operating as switches. These devices can be fabricated in discrete form or as integrated circuits (ICs).

The growth of the power electronic field has permitted the development of several applications; in consequence, today it is possible to build power supplies, battery chargers, electric drives, DC transmission systems, and high frequency and power converters, among others. These power electronic applications require the use of electronic power converter: rectifier (transform voltage from ac to dc), inverter (from dc to dc), chopper or a switch-mode power supply (from dc to dc), and cycloconverter and cycloinverter (from ac to ac).

Current technology used for energy conversion with power electronic, like all energy transformations, is not completely 100% efficient due to losses in switching and energy storage devices. These losses are in the form of heat dissipated. The operation of all power electronic devices is highly temperature dependent. In fact, an electrical and thermal interdependence exists; electrical characteristics of the power device depend on the device temperature, and the device temperature depends on the device losses. In consequence, the design of power electronic systems requires the characterization of this codependence in order to describe the actual system operation. Most failures in power electronic systems are due to thermal mechanisms, thus, the thermal behavior analysis of power systems has become an increasingly important issue in the design and operation of the power electronic systems.

The electrical design at the system level for power electronic modules has significantly progressed over the past years. The electronic industry demands systems more compact: with an increasingly higher number of integrated components into the system, combined with more capacity of converted power, and higher switching frequencies. However, the components integration in a system increases the complexity of the thermal process due to interactive heating of the system components. This way, the imperative need of an accurate thermal study, in the general analysis of power electronic systems, is accentuated. There are tools that can be employed to characterize the thermal behavior in electronic packages based on computational fluid dynamic techniques, or implementing a specific thermal model in a simulator. Nonetheless, there are cases where commercial software tools and established models in the simulator are not suitable in the early design stage because they do not have the enough accuracy or because it takes long time to obtain a result. In the later case, the use of fast and simple model may be a better choice.

When commercial software tools are used for the description of the electrothermal behavior of power systems, it is requires multiple commercial software tools, since no single package can currently be used to perform all this modeling and analysis by itself. It is necessary to make independent analyses in each software tool that, later, can be integrated to another specialized software tool. This is due to the fact that at the development of power system requires the integration of mechanical and electrical modeling and analysis activities to represent the existing electrothermal interaction. Therefore, computational fluid dynamics and heat transfer simulation must be completed in parallel with the electrical design in order to obtain the overall optimal analysis. For example, FLOTHERM^{TM} or I-DEAS^{TM} simulators can be used to make the thermal analysis, SABER^{TM} and MAXWELL^{TM} simulators can be used to make the electrical analysis, and iSIGHT^{TM} software can be used to control the simulations. At this point, it is clear that many important details required by the system in its real operation are lost in the information exchange between the different software tools. In addition, it is evident that each simulation requires great effort from both the designer and the simulator software to show reasonable results, resulting in long design cycles that unduly increase cost and time to market.

Reduced-order models to describe the thermal behavior of power electronic systems are an alternative to surpass limitations when software tools are used. Currently, these models are generated using simplified semi-empirical formulas of the thermal resistances and capacitances of simple shapes [1-4], or using finite element (FE) simulation data in equivalent circuit topology [5], or using the thermal impedance of a equivalent circuit [6]. However, these thermal reduced-order models can be highly complicated, because traditionally they are based on a conduction analysis through the heat transfer equation's use. This implies the development of methodologies that require a significant effort. The objective of this work is to provide a simple technique that allows an accurate description of thermal behavior of any electronic package. Here is presented a simple methodology that can be used to describe the thermal performance of the Generation II IPEM, based on the Lumped Thermal Capacitance Method (LTCM) [7-9], which neglect the conduction effect and instead considers other forms of heat transfer through energy balance.

1.2 Power Electronic Overview

The electronic devices used in power electronic systems are based on semiconductor technology. That is to say, the devices are constituted by semiconductor materials, such as pure silicon, silicon with impurities (for example, boron, indium, arsenic or phosphorus added), and compound semiconductors (for example, gallium arsenide or indium antimonite) [7]. All with specific atomic configuration that allow a determined electron flow (current) through the device.

According to [10], the atoms inside a crystal (a single crystal is formed by atoms bound together in an orderly structure), in a semiconductor material, are constantly vibrating due to their thermal energy. Under equilibrium conditions, these vibrations define the temperature of the crystal.

If a electrical current is applied to the material, the electron flow interaction with the crystalline structure results in heat which increases thermal vibrations of atoms and the bound electrons begin to gain energy. Some of these electrons gain enough energy to jump across the energy gap ε_g between the valence and conduction bands, creating a new free electron-hole pair that is available to conduct electrical charge. As the temperature increases, more and more electrons make this transition, being created an electric field that produces an average drift velocity and then a current flow is observed. Thus, when increasing the temperature, the current also increases. In the absence of the applied external voltage, the electron motions are random and cancel one another.

For the case of an applied voltage to the material, processes of electron excitation and recombination are shown in Figure 1.1. Here, when an electron leaves its valence state, the host atom is left short an electron, producing an ion. The ion tends to reach out and captures an electron from a nearby atom, leaving this neighbor now short an electron. The process repeats itself, and the missing electron location moves around in the crystal, this is the excitation process. The inverse process is named recombination.

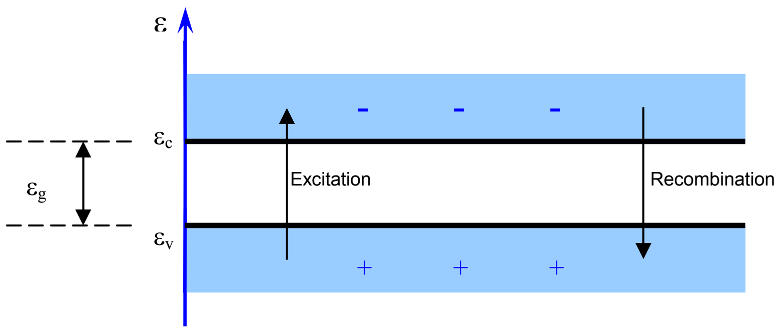


Figure 1.1. Electron excitation and recombination processes.

The previous electronic process is valid for any semiconductor device. However, each electronic application requires a specific electronic process, depending of semiconductor material and, obviously, of the device task.

Power electronic systems process large amounts of power that only one particular type of device can achieve. In that case, it is necessary to rely on semiconductors with conductivity precisely controlled and relatively constant over a wide temperature range. For this, it is necessary to add carefully measured amounts of certain impurities to the material. There are two types of impurities: donor and acceptor; and two corresponding classes of doped semiconductors: n-type and p-type. Generally, the devices are created by forming a junction between n- and p-type materials from a single crystal, that allows to the carriers to move it across the boundary region. An example is shown in Figure 1.2, to fabricate the p-n junction diode, or junction semiconductor diode, one end of the crystal is doped with donor material and the other end is doped with acceptor material.

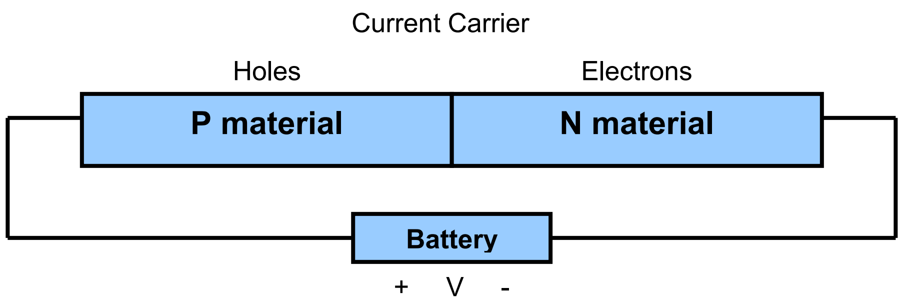


Figure 1.2. PN junction diode.

When more than two doped semiconductor materials are joined, the device takes another name: transistor. A transistor is an active semiconductor device with three or more terminals [11]. Each terminal is a doped semiconductor. A regular transistor includes a middle material called base region, and other materials on either side of the middle, called the collector and the emitter since their function is to emit and collect charge carriers (electrons).

Some transistor types are the bipolar junction transistor (BJT), the insulated gate bipolar transistor (IGBT), field effect transistor (FET), and metal oxide semiconductor field effect transistor (MOSFET), among others. The transistor of interest for this investigation is the MOSFET.

Consistent with [12], in the MOSFET, wells of highly doped n-type silicon called source and drain are diffused or implanted into a p-type substrate. A conducting gate is insulated from the silicon by a thin layer of SiO_2. The MOSFET is physically symmetric, with source and drain ultimately defined by the current direction. In operation, the positive charge is placed on the gate by an external source. This source attracts electrons from the wells into the region just beneath the oxide, creating a conducting channel between source and drain. The physical current in the channel is an electron flow, thus one terminal is considered as the source of electrons that flow through the channel to the drain when external voltage is applied (see Figure 1.3).

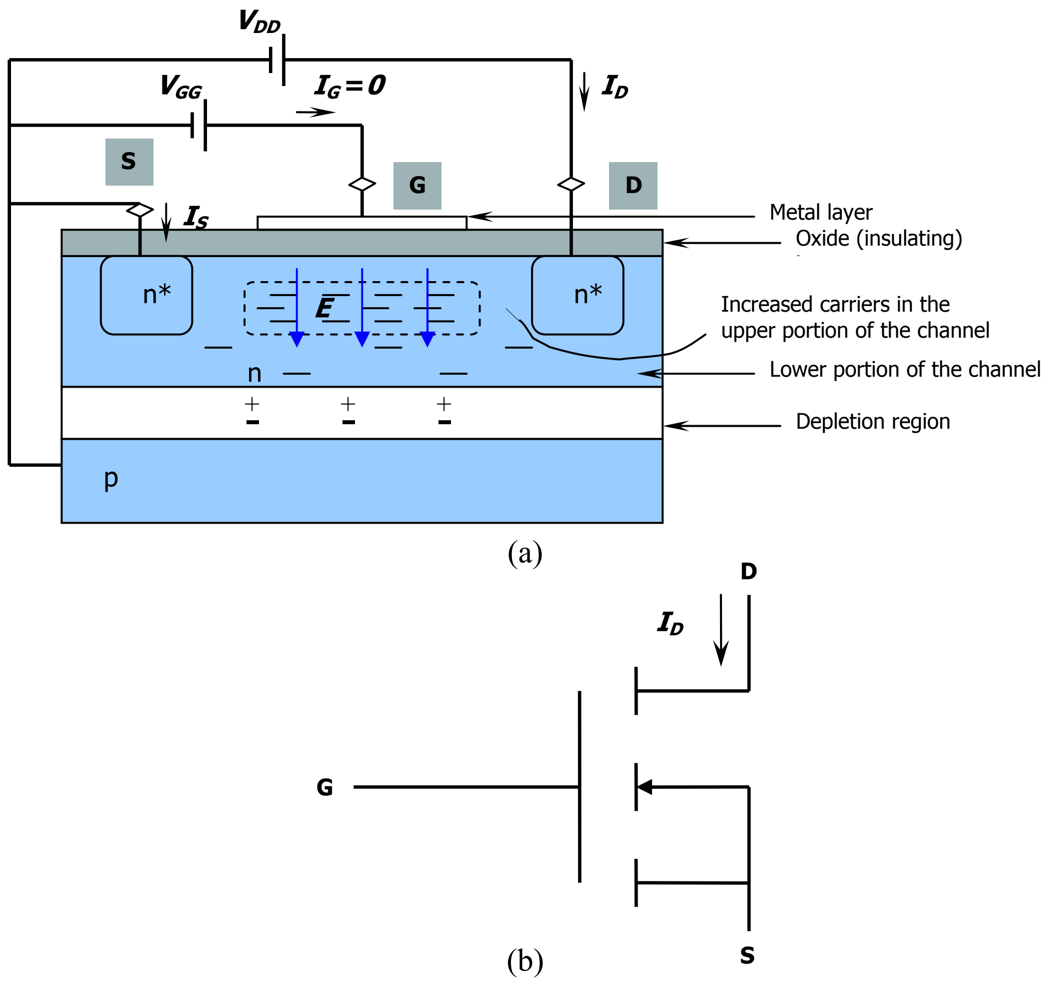


Figure 1.3. N-channel MOSFET: (a) physical structure; (b) schematic symbol.

It is evident that temperature changes will have a significant effect on the device electrical properties. For example, the transistor normal operating voltages and currents will result in heat dissipation, which in turn will increase temperature. This increase in temperature will produce changes in the electrical characteristic not only of these devices in the vicinity. When there are systems that include one or more transistors, the thermal analysis is extremely necessary given that the device's temperature sensitivity is the focal point.

Now, the present technology provides the continuous development of new semiconductor materials, which in turn allow the development of power electronic system innovations for several applications.

The next step is to get integrated systems, more powerful, durable, smaller, lighter, and less costly to the consumer. An Engineering Research Centers Program sponsored by the National Science Foundation, Center for Power Electronics Systems (CPES), is developing such systems. CPES' Integrated Power Electronics Modules (IPEM) are systems whose principal characteristic is to have high integration levels of power semiconductor devices (transistors), gate drivers, and control circuitry for a wide range of power electronics applications. Figure 1.4 presents a picture of the Generation II IPEM a half bridge switch. A detailed description of the Generation II IPEM is given in [13, 14]. Two power semiconductor devices (MOSFETs) and a hybrid gate driver constitute this IPEM. MOSFETs are buried in a ceramic frame (that constitutes the base substrate and its material is aluminum oxide) and covered by dielectrics with holes on the aluminum pads of the chips. The power devices are interconnected to other circuits by metal deposition, which additionally provides thermal paths for the devices.

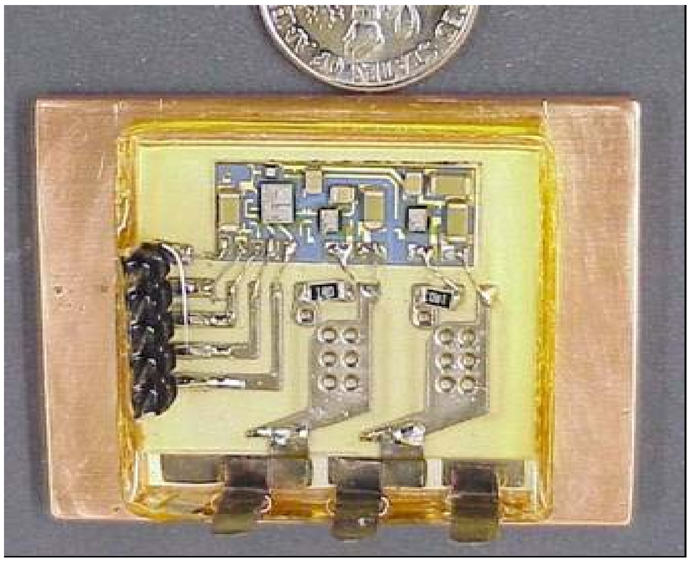


Figure 1.4. Picture of generation II IPEM.

The operation of these systems encloses an amount of heat generated that must be dissipated. If the temperature of such systems is allowed to rise with no control, the electronic performance of the device may be degraded or the component looses its physical integrity. This is an operational limitation for these components. Thereby changes in the temperature and its distribution inside the device can affect drastically the device’s electrical performance. According to [15], as soon as the temperature inside the devices rises, two effects can occur: thermal runaway or self-heating. In thermal runaway, the electrical energy dissipated causes a temperature rise over an extended area of a device resulting in an increase of dissipated power. When the self-heating takes place, the device temperature increases, which leads to a catastrophic device failure. Nevertheless, an equilibrium situation can be reached by placing the device in contact with a lower temperature solid or fluid, which facilitates heat flow away from the component. In consequence, it is extremely necessary to have a good grasp of the thermal process for reliability and optimum performance. Thermal conduction, convection, radiation, interface effects, as well as phase change processes are necessary to explain the thermal behavior of any device or system.

Often the temperature effect on the device can be described by finding how the voltage changes with the temperature for a constant current. The temperature effect makes necessary the characterization of the thermal process, in order consequently to represent the electrothermal performance of any IPEM. For the design of any IPEM it is desired to simulate the electrothermal phenomenon, to now and understand how the system performance is affected by thermal effects. Electrothermal simulations in the design stage can be useful to evaluate performance, to analyze IPEM reliability, to optimize the package design, among other uses.

Several methodologies can be used to model the thermal behavior of power electronic systems. These include: three-dimensional finite difference and finite element simulation [1-4], the lumped concept or empirical extraction of thermal network element values from the measured thermal step response [3, 4, 6, 16], reduced model techniques

[17, 18], the method of images which uses Green's function approach to obtain the resulting temperature distribution from a heat source [19], integrated analysis with design tools [20], among others. All these methods require a very exhaustive and complex mathematical analysis, because they are based on the three-dimensional heat diffusion equation. This limitation represents a computational "straight-jacket", because it restricts the easy addition of new elements, as occurs, for example, with specific software used by the electronic industry like FLOTHERM^{TM}, ANSYS^{TM} and IDEAS^{TM}.

The need to reduce the transient 3D thermal analysis, using a simple methodology, fueled the present research project. Since in CPES such technical analysis has not yet been implemented, this point can be considered as the major justification for this research work.

The main rationale of this research is the development, validation and implementation of a reduced mathematical model using a practical numerical formulation of the thermal behavior of a typical active IPEM, from geometrical IPEM data (Generation II IPEM). This model is based on the expanded Lumped Thermal Capacitance Method (LTCM) [7-9]. Applying this procedure a simple, non-spatial, highly non-linear model is obtained. With the LTCM model and using a circuit simulator, such as SABER^{TM}, it is possible to obtain the electrothermal interaction.

The SABER^{TM} circuit simulator possesses model libraries to simulate the electrical behavior of electronic devices and of other components, making the electrothermal coupling more accessible. The LTCM model is coupled with SABER's^{TM} electrical model for different components. In the electrothermal simulation, SABER^{TM} solves for the temperature distribution (using the implemented LTCM model) as well temperature dependent electrical parameters of the device simultaneously. The LTCM model is implemented in SABER^{TM}, the corresponding electrothermal simulation results are comparated with experimental data and with results from a commercial package, FLOTHERM 3.1^{TM}, under the same simulation conditions.

At the end of the research project, a clear understanding of the interactions of the electrical and thermal behavior and vice versa was developed. In the same way, a full and easy modeling methodology to simulate the dynamic electrothermal performance of the Generation II IPEM has been developed, which will be used for virtual prototyping of IPEMs.

The subsequent chapters of this thesis are organized as follows. Chapter 2 presents a background on previous work in thermal simulation in the electronics area conducted with different concepts: traditional three-dimensional finite difference and finite element, the lumped concept or empirical extraction of thermal network element values from the measured thermal step response, reduced model techniques, and integrated analysis with design tools such as specific software, among others. Chapter 3 describes the reduced thermal model from LTCM, also this chapter shows the validation of the LTCM model and the thermal analysis software FLOTHERM 3.1^{TM} with constant power. Chapter 4 describes the data acquisition systems, based on LabWindows/CVI^{TM} from National Instrument, and the experimental setup used to acquire the data from low- and high-speed thermal response experiments, some comparisons between low speed thermal response experiment data and the LTCM results are also presented in Chapter 4. Chapter 5 presents the model implementation in the circuit simulator SABER^{TM} to obtain the active electrothermal model from the LTCM model and SABER^{TM} tools, in this chapter the comparison of the high speed thermal response experiment data with SABER^{TM} simulations is also presented. Finally, Chapter 6 gives a summary and conclusions of the research with recommendations for future efforts.

CHAPTER 2

BACKGROUND

This chapter presents the diverse types of modeling approaches to describe the electrothermal behavior of microelectronic devices and packages. According to analyzed components in different simulations, an appropriate categorization of previous work is as follows: (i) semiconductor devices analysis, those based on the thermal or electrothermal analysis of diverse semiconductor devices; (ii) electronic packages analysis, those simulation techniques used to describe the electrothermal behavior of packages; (iii) IPEMs analysis, those that describe the electrothermal analysis used to study IPEMs operation.

2.1 Power Semiconductor Devices Analysis

The first study of the semiconductor devices described here corresponds to work of Hefner in [1]. He developed, based on the lumped concept (or thermal networks), a dynamic electro-thermal model for the Insulated Gate Bipolar Transistor, IGBT, from the temperature-dependent IGBT silicon chip. The temperature-dependent IGBT electrical model describes the instantaneous electrical behavior in terms of the instantaneous temperature of the IGBT silicon chip surface. The instantaneous power dissipated in the IGBT is calculated using the electrical model and determines the instantaneous rate of heat applied to the surface of the silicon chip. The electrothermal model of the IGBT was implemented in SABER^{TM}, and is available in the SABER^{TM}

components library. Although this technique to describe the electrothermal behavior of a semiconductor device is a reduced-order model, the approach still complex and possibly inaccurate, since it is based in heat diffusion equation (which entails to the multidimensional analysis and to reduce the boundary conditions to only convection).

In [6], also from the lumped concept and from a thermal multidimensional analysis, Codecasa *et al*. describe the thermal response of electronic devices by means of the thermal impedance. The derivation developed to obtain this parameter is based on very general electrical equations, so that the electrothermal behavior of semiconductor devices is described through an equivalent electrical network, implemented in SPICE^{TM}. They define an electrothermal network where the thermal impedance is reported, through a transformation matrix, at the electrical terminals. Then a purely electrical compact model embedding the thermal effects is obtained.

A more general study of the semiconductor devices is presented by Min *et al.* in [19]. where a full and nontraditional analysis of a chip is presented. They developed an analytical three-dimensional transient temperature solution of a two-layer semi-infinite plate structure with embedded heat sources. In this solution, the thermal behavior of a typical semiconductor device from the thermal diffusion equation is described. It employs the method of images, which uses Green's function approach to obtain the resulting temperature distribution from a heat source. In addition, the principle of superposition is applied in order to adjust the correct boundary conditions. This solution technique has been programmed and is particularly useful for devices operating under pulsed or switching conditions.

2.2 Package Analysis

The study selected as a starting point in this section, is the one corresponding to Adams *et al.* in [21]. It provides a study of the thermal complex interactions between the components of a enclosure with horizontal narrow aspect ratio: heat sources, substrate, and enclosure. The authors examined the thermal behavior of this array from conservation equations for continuity, momentum and energy in the three-dimensional problem, considering natural convection in air, coupled with conjugate conduction and radiation within an enclosure, and assuming constant properties. Those models were solved using a finite volume method. The study determines which physical effects and level of detail are necessary to accurately predict thermal performance of discretely heated enclosures.

Numerical solutions have also been chosen to solve the electrothermal models of electronic packages. The first one was presented by Adams *et al.* in [22]. They suggested a methodology for the validation of geometric and physical compact thermal models implemented in computational tools (in this case FLOTHERM^{TM}) with simple but realistic conditions. This validation is made by comparing the geometric reduced model of an electronic package accomplished in the software with experimental data of the real package. Here, the importance of reducing the complexity of the many problems that to obtain design tools more accessible using existing software is visible.

Another numerical lumped concept methodology is given by Hsu and Vu-Quoc in [16]. They presented a rational approach for constructing thermal circuit networks, equivalent to the discretization of the thermal diffusion equation using the finite element method. These thermal circuit networks are connected to the electrical networks of power electronic systems to provide complete electrothermal models that can be conveniently used in any circuit simulator package. Later, in [17], two reduced model techniques are applied to the previously obtained models, Modal Superposition Method (MS) and Component Mode Synthesis (CMS). In the first, the governing differential equations for the reduced model are uncoupled so they can be easily solved. In the second, the idea is to find reduced models for various substructures independently, and to use compatibility conditions to connect these reduced substructure models. Both techniques were implemented in the SABER^{TM} simulator.

In similar fashion, Lee and Allstot in [18] presented another reduced-order modeling technique to simulate the transient electrothermal performance of integrated circuits. From an efficient macromodeling method, based on the Asymptotic Waveform Evaluation (AWE), the time domain response of a linear circuit is efficiently evaluated in terms of a few dominant poles and residues. In general, from the heat diffusion equation, the thermal behavior is represented by an equivalent thermal circuit, and is coupled with the electrical model in the SPICE^{TM} simulator.

Following the reduced-order model approach, a new methodology to describe the dynamic electrothermal behavior of power electronic circuits and systems is given by Hefner and Blackburn in [2]. They proposed, developed, and validated a typically space technique that consists of defining the temperature at various positions within any package from the heat diffusion equation for various three-dimensional coordinate system symmetry conditions and include the nonlinear thermal conductivity of silicon and nonlinear convection heat transfer. Later, the resulting models are discretized into a finite number of first-order ordinary differential equations (using finite differences). The thermal component models include the nonlinear thermal conductivity of silicon and nonlinear convection heat transfer. The model solutions define a thermal network.

Hefner and Blackburn's work in [3] explain how the interconnection between the electrical network and the thermal network can be represented through electrothermal models. The electrical and thermal networks are coupled through the electrothermal models for the semiconductor devices. The electrothermal models for semiconductor devices and other components (with electrical interaction) have electrical terminals that are connected to the electrical network and a thermal terminal that is connected to the thermal network. The thermal nodes in the thermal network have units of temperature across the nodes and units of power flowing through and across. Whereas the through and across variable for electrical network are current and voltage. The thermal network is represented using thermal network component models so that the thermal models for different packages and heatsinks can be readily interconnected in the same way that the electrical network components are connected. The thermal network models for power modules and heatsink contain multiple terminals and account for the thermal coupling between the adjacent semiconductor devices. The SABER^{TM} circuit simulator is used for electrothermal network simulation. The models are formulated such that the components of power flow between the thermal nodes are expressed in terms of the node temperature.

From Hefner's work and following with the lumped concept, Digele *et al.* in [23] developed a fully coupled dynamic electrothermal simulation on chip and circuit level, implemented in SABER^{TM}. The approach in this work was to discretize the chip in three dimensions and build these equations like a behavioral model into SABER^{TM}, using finite difference methods (FDM) for the heat diffusion equation. The contribution of that research is the attainment of the isolines of temperature at a critical time step during the simulation or under steady state condition. The temperature isolines at every simulation time step can be drawn, which helps to identify the temperature dependence components.

2.2.1 IPEMs Analysis

Chen *et al.* [20] provide an integrated analysis of IPEMs from several software tools: mechanical CAD software I-DEAS^{TM}, Maxwell^{TM} Q3D Parameter Extractor, FEA software FLOTHERM^{TM} and SABER^{TM}, for modeling of thermal and electrical behavior of those systems. Each software has a specific analysis technique. The exchange of geometry information among the resulting models, from each software, was achieved. I-DEAS^{TM} is used to model the physical layout and material information of IPEM. It is possible to use that information to calculate the parasitic inductance of module layout using Maxwell. Later, these inductances are included in the SABER^{TM} simulation for electrical performance evaluation. With the power loss calculated in SABER^{TM} and geometry translated from the I-DEAS^{TM} model, the thermal analysis is performed using FLOTHERM^{TM}. Finally, with the aid of all the software tools, tradeoffs between electrical performance and thermal management are investigated.

Rodriguez *et al.* [4] described with a certain degree of detail the general features of the lumped parameter thermal model for electrothermal analysis of a commercial IGBT power electronic module from Toshiba^{TM}. This model was obtained using the analysis methodology developed in [2]. The previous thermal network component models [3] are not applicable for high power modules because high power devices contain multiple chips within the same package, inducing heat conduction through the electrical insulator layers, and are typically used with large multi-module heat sinks that have a highly non-uniform surface temperature. However, in this work the thermal model is formulated similarly to the single silicon chip thermal model except that the expressions used to calculate the thermal resistances, thermal capacitances, and the heat energies are different. The model describes the two-dimensional lateral heat spreading, the die attachment thermal resistance, and the heat capacity of the commercial IPEM periphery. The lateral heat spreading in the commercial IPEM results in an effective heat flow area that increases with depth into this electronic package.

In the Rodriguez's model, the effective heat flow area at each depth into the commercial IPEM is obtained by combining the components of heat flow area due to the cylindrical heat spreading along the edges of the chip, the spherical heat spreading at the corners of the chip, and the rectangular coordinate component of heat flow directly beneath the chip.

They developed an experimental system to validate the electrothermal models of commercial IPEMs [26]. The system consisted of a computer-based data acquisition system based on LabWindows/CVI from National Instrument with digital multimeters and recorders. The system acquires temperature at different points in the three-phase inverter and voltages at the module inputs. Communication between instruments and the computer is based on the GPIB protocol.

Recently Pang *et al.* [14] developed a methodology to optimize the three-dimensional geometrical design layout of an active IPEM by considering both electrical and thermal performance. The thermal analysis was made using a commercial finite element and computational fluid dynamic (CFD) solver, I-DEAS^{TM}, in steady state. Then, a parametric study was conducted to determine the thermal performance of several design layouts and a sensitivity analysis was performed to determine the overall uncertainty of the simulations. Using this integrated design analysis three alternatives of geometric configuration were achieved, according to electrical and thermal requirements. From this study, Generation II IPEM design was obtained. This IPEM is the module used in this thesis.

It can be concluded through this literature review that the study of the electrothermal behavior of semiconductor devices and, mainly, power electronic packages has been only described by means of a few methodologies ranging from heat diffusion equation, neglecting the radiation but, in many cases, including convection effects.

Few full descriptions reporting a couple electrothermal analyses have used multidimensional approaches requiring a large amount of effort and computational resources. It is therefore necessary to develop a simple but efficient methodology for the design and analysis of the next generation of power electronics modules such as IPEMs. As described in detail in the next chapter, this method is based on a reduced thermal model that incorporates the coupled thermal and electrical performances. This is the main objective of the work presented in this thesis.

CHAPTER 3

REDUCED-ORDER THERMAL MODEL

3.1 Introduction

Traditionally, computational fluid dynamics (CFD) simulations of electronic system with detailed modeling of the electronic devices and conjugate heat transfer interactions are utilized to predict the thermal state in electronic applications. Despite large variability in length scales at the device and system levels results the demand considerable computational requirements, which is translated in an expense in design time. In that case, compact or reduced modeling of electronic components without the use of system CFD simulations may be a viable alternative to meet the design process requirements [1].

A reduced-order thermal model of a component is a model that has modest complexity, but captures the main thermal features for a particular analysis. This moderate complexity improves computational efficiency, allowing thermal simulations of the electronic system to be completed, using personal computers (PCs) or workstations, in reasonable time.

The main characteristic of an IPEM is the high component integration level. The new packaging method employed in IPEM manufacturing eliminates wire bonds, which lead to potential benefits from both the electrical and thermal prospective [14].

Nevertheless, components layout of this package do not permit an easy thermal analysis implementation in a typical software, because they require a complex configuration at the same time. This way, the use of some reduced model that allows the fast and simple analysis of all elements that compose the IPEM is necessary.

From this point, there are many possibilities to develop reduced models, each one with specific applicability in the thermal analysis of the electronic components, such as the lumped concept. However, until now such components have not been described using a methodology of non-spatial analysis. All have started from the heat diffusion equation (HDE), which generally are multidimensional. This fact increases the magnitude of the analysis since it makes necessary the employment of the complex techniques to discretize the HDE, for example finite element or volume, some matricial technique, etc.

It is the final goal to develop a simple mathematical model that describes the thermal behavior of a typical IPEM. In most cases, in some unsteady situations the use of the lumped capacitance theory greatly simplifies the analysis. The lumped capacitance theory assumes that the temperature within a solid is spatially uniform at any instant throughout an unsteady heat dissipation process. Thus, the reduced thermal model is based on the Lumped Thermal Capacitance Method (LTCM), also named Lumped Capacitance Method or Lumped Capacitance Heating and Cooling [7-9]. The use of this model implies that the unsteady heat transfer analysis only depends on the time, representing spatial thermal distribution by the physical and thermal characteristics of the IPEM.

3.2 Lumped Thermal Capacitance Method (LTCM)

In order to determine the validity of the LTCM approach, certain criterion must be satisfied. The Biot number (Bi), a dimensionless parameter, relates the internal conduction resistance to the external convection resistance during transient heat transfer. This dimensionless parameter is used to test for the validity of the LTCM approach. The Biot number is defined as the ratio of temperature differences across the solid itself (conduction heat transfer), and between the solid and fluid (convection heat transfer). It is given by [7-9]:

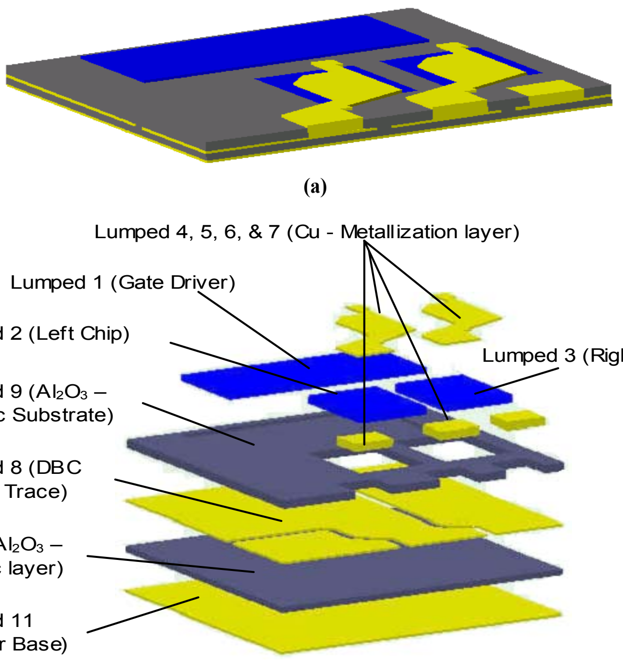
$$\mathrm{Bi} = hL / k\,, \qquad (3.1)$$

The reader is referred to the nomenclature section for the definition of the symbols used throughout this text.

Values of the Biot number larger than 1 imply that the heat conduction inside the body is slower than at its surface, and temperature gradients are non-negligible inside it. On the contrary, a Biot number value lowest than 1 suggest that LTCM approach can be used to describe the transient heat transfer phenomenon. This way, to validate the LTCM approach it is necessary that [7-9]:

$$\mathrm{Bi} << 0.1$$

Unless previous requirement is satisfied, the method will be inaccurate. A small Biot number is an indication of a very efficient conduction heat transfer inside the body, and temperature variations can be neglected inside the body. Whenever the nondimensional parameter is smaller of the stipulated value, the condition of minimum temperature variations could be used for most power electronic modules.


(a)
Lumped 4, 5, 6, & 7 (Cu - Metallization layer)
Lumped 1 (Gate Driver)
Lumped 2 (Left Chip)
Lumped 3 (Right Chip)
Lumped 9 (Al_2O_3 – Ceramic Substrate)
Lumped 8 (DBC Copper Trace)
Lumped 10 (Al_2O_3 – DBC Ceramic layer)
Lumped 11 (Copper Base)

(b)

Figure 3.1. Generation II IPEM: (a) geometry model; (b) lumped decomposition.

LTCM allows to deal with the heat transfer between the body and the ambient fluid by convection, between the body and surroundings by radiation, thermal contact and spreader resistance effects, and transient effects. In order to obtain the temperature profile of the Generation II IPEM, each material is treated as a control volume or lumped, as shown in Figure 3.1. A mathematical model of the IPEM is constituted by several lumped. Applying an energy balance in each lumped, the temperature at each lumped is,

$$\dot{E}_{stj} = \dot{E}_{gj} + \dot{E}_{inj} - \dot{E}_{outj}, \qquad (3.2)$$

or,

$$\underbrace{\rho_j V_j C_{pj} \frac{dT_{Lj}}{dt}}_{\text{Transient Effect}} = \underbrace{q_{gen.}}_{\substack{\text{Heat from}\\ \text{power devices}}} - \underbrace{(h_{hor}A_{hor} + h_{ver}A_{ver})(T_{Lj} - T_{\infty})}_{\text{Convection}} \pm$$
$$\underbrace{\sum_j \left[\varepsilon_j \sigma F_{j-jnext} A_{j-jnext} (T_{Lj}^4 - T_{Ljnext}^4)\right]}_{\text{Radiation}} \pm \underbrace{\sum_j \left[\frac{(T_{Lj} - T_{Ljnext})}{R_{j-jnext}}\right]}_{\substack{\text{Thermal Contact and}\\ \text{Spreader Resistance}}}, \qquad (3.3)$$

where

$\dot{E}_{stj}$ is the stored energy in the lumped,

$\dot{E}_{gj}$ is the generated energy inside lumped,

$\dot{E}_{inj}$ is the input energy to lumped,

$\dot{E}_{outj}$ is the output energy from lumped.

In Equation 3.2, the generated energy (in heat form) within the lumped corresponds to the electrical power dissipated by the device, this parameter must be calculated in the electrical circuit analysis and constitutes the input to the thermal model. In some cases, this power is null because many lumped do not have a dissipated power. Generally the lumped that dissipate power are those associated to semiconductor devices. In addition, Equation 3.2 considers the heat that enters the lumped through thermal contact and spreader resistances with adjacent devices. The heat that leaves the lumped is due to mostly convection, radiation, and, in some cases, due to the phenomenon of heat exchange between lumped through thermal contact and spreader resistances. Finally, the stored heat is due to the variation of the lumped internal energy.

According to Equation 3.3, the full model of a typical IPEM includes the form of heat dissipation that generally appears in electrothermal processes from the devices (power semiconductor devices and gate driver). These heat forces are:

1) Free or forced convection of each lumped with the surrounding ambient.
2) Radiation between lumped and between each lumped and the surrounding environment.
3) Heat spreading due to conduction in the direction of the contact surface.
4) Thermal contact resistances.
5) Transient effects.

The energy balance applied to each lumped can contain some or all the expressions that appear the Equation 3.3. For example, in the case of the convection, if this affects a lumped vertical as much as horizontally, then the convection expression must be specified similar to the Equation 3.3’s expression; in the contrary case, only will appear the convection form that is present. The same happens to the remaining expressions contained in the corresponding energy balances.

3.3 Model Description

The validity of the LTCM was verified by calculating the Biot number for each lumped. According to the Equation 3.1, this parameter is function of the temperature since the heat transfer coefficient depends of the temperature. Likewise, in order to calculate the Biot number, the used characteristic or equivalent length must be defined so that it corresponds to the heat transfer thickness according to the heatflow direction (airflow direction for each lumped is shown in Table 3.1). In the transient analysis, for the IPEM under study, this parameter varies between 4.75×10^{-6} and 3.73×10^{-4} (see Figure 3.2). The Biot numbers presented in Figure 3.2 were calculated from transient temperature of each lumped assuming devices' dissipated power of 1, 7 and 12W for gate drive, left and right chip, respectively.

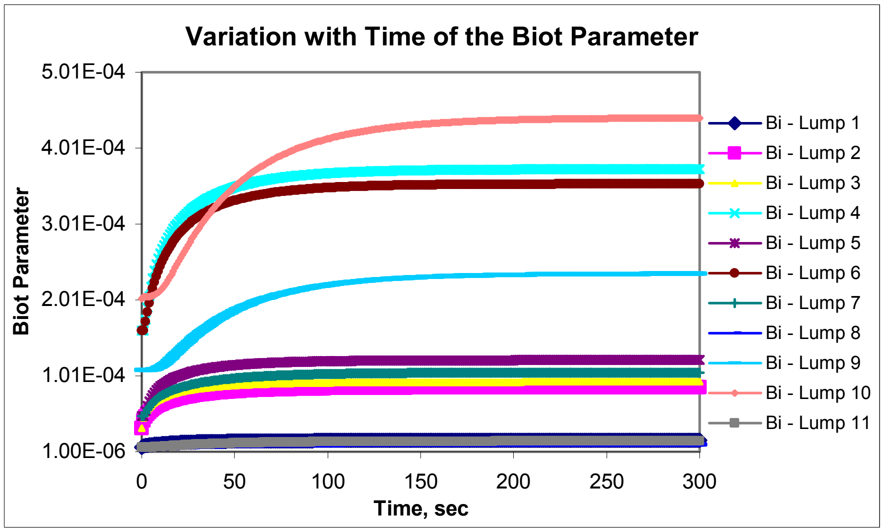


Figure 3.2. Corroboration of Biot parameter condition.

From the properties of each lumped material, the model can be used to calculate the energy balance of each lumped from Equation 3.3 resulting in a set of simultaneous non-linear ordinary differential equations that can be solved by means of an efficient numerical integration method, such as the fourth order Runge Kutta method [24]. The solution using fourth-order Runge Kutta method can be obtained as follows:

$$T_{Lj(i+1)} = T_{Lj(i)} + \frac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right)\Delta t\,, \tag{3.4}$$

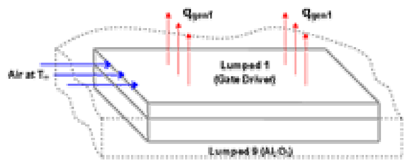
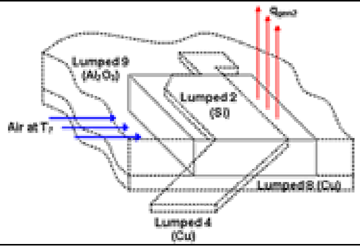
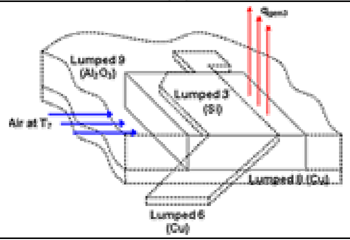
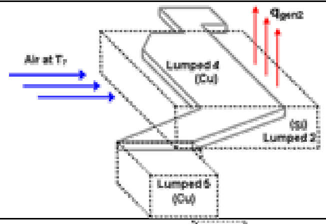
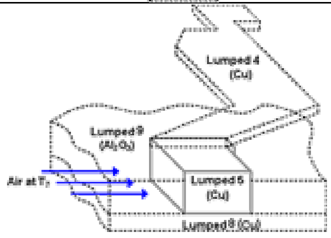
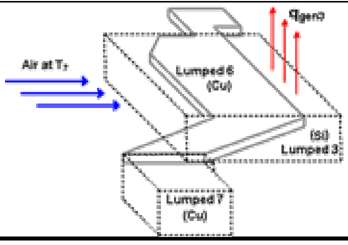
where,

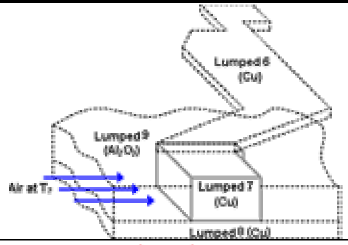
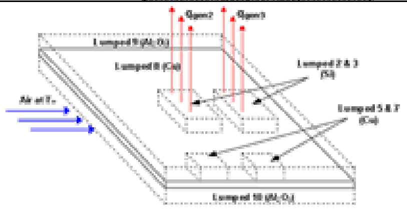
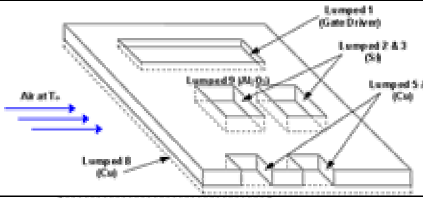
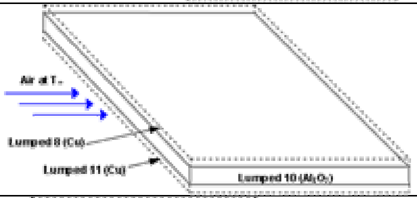
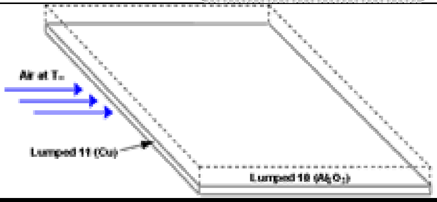
$$\begin{aligned} K_1 &= f(t_i, T_{Lj(i)}) \\ K_2 &= f(t_i + \frac{\Delta t}{2}, T_{Lj(i)} + K_1\frac{\Delta t}{2}) \\ K_3 &= f(t_i + \frac{\Delta t}{2}, T_{Lj(i)} + K_2\frac{\Delta t}{2}) \\ K_4 &= f(t_i + \Delta t, T_{Lj(i)} + K_3\Delta t) \end{aligned}\,, \tag{3.5}$$

After many previous divisions of the Generation II IPEM to find the number of appropriate lumped, this IPEM was dividing in eleven parts or lumped and a significant change in the IPEM temperature distribution was not observed. With less lumped number, the model do not catch the real temperature distribution. With greater lumped number the model require more computational time to obtain the temperature distribution without considerable temperature changes in the IPEM's thermal profile.

Since the IPEM under study consists of eleven lumped, the assembly can be modeled by eleven energy balance equations, which correspond to temperature profile of the IPEM. These equations are listed in Table 3.1.

Table 3.1. Model Expanded for each IPEM's Lumped

No	Geometry	Energy Balance	Eq
1	 q_{gen1} q_{gen1} Lumped 1 (Gate Driver) Air at T_∞ Lumped 9 (Al_2O_3)	$\rho_1 V_1 C_{p1}\frac{dT_1}{dt} = q_{gen1} - (h_{hor1}A_{hor1} + h_{ver1}A_{ver1})(T_1 - T_\infty) - \varepsilon_1\sigma F_{1\text{-}\infty}A_{1\text{-}\infty}(T_1^4 - T_\infty^4) - \varepsilon_1\sigma F_{1\text{-}2}A_{1\text{-}2}(T_1^4 - T_2^4) - \varepsilon_1\sigma F_{1\text{-}3}A_{1\text{-}3}(T_1^4 - T_3^4) + \varepsilon_2\sigma F_{2\text{-}1}A_{2\text{-}1}(T_2^4 - T_1^4) + \varepsilon_3\sigma F_{3\text{-}1}A_{3\text{-}1}(T_3^4 - T_1^4) - \frac{(T_1 - T_9)}{R_{1\text{-}9}}$	3.6
2	 q_{gen2} Lumped 9 (Al_2O_3) Lumped 2 (Si) Air at T_∞ Lumped 8 (Cu) Lumped 4 (Cu)	$\rho_2 V_2 C_{p2}\frac{dT_2}{dt} = q_{gen2} - h_{hor2}A_{hor2}(T_2 - T_\infty) - \varepsilon_2\sigma F_{2\text{-}\infty}A_{2\text{-}\infty}(T_2^4 - T_\infty^4) - \varepsilon_2\sigma F_{2\text{-}1}A_{2\text{-}1}(T_2^4 - T_1^4) + \varepsilon_1\sigma F_{1\text{-}2}A_{1\text{-}2}(T_1^4 - T_2^4) - \frac{(T_2 - T_4)}{R_{2\text{-}4}} - \frac{(T_2 - T_8)}{R_{2\text{-}8}} - \frac{(T_2 - T_9)}{R_{2\text{-}9}}$	3.7
3	 q_{gen3} Lumped 9 (Al_2O_3) Lumped 3 (Si) Air at T_∞ Lumped 8 (Cu) Lumped 6 (Cu)	$\rho_3 V_3 C_{p3}\frac{dT_3}{dt} = q_{gen3} - h_{hor3}A_{hor3}(T_3 - T_\infty) - \varepsilon_3\sigma F_{3\text{-}\infty}A_{3\text{-}\infty}(T_3^4 - T_\infty^4) - \varepsilon_3\sigma F_{3\text{-}1}A_{3\text{-}1}(T_3^4 - T_1^4) + \varepsilon_1\sigma F_{1\text{-}3}A_{1\text{-}3}(T_1^4 - T_3^4) - \frac{(T_3 - T_6)}{R_{3\text{-}6}} - \frac{(T_3 - T_8)}{R_{3\text{-}8}} - \frac{(T_3 - T_9)}{R_{3\text{-}9}}$	3.8
4	 q_{gen2} Air at T_∞ Lumped 4 (Cu) (Si) Lumped 2 Lumped 5 (Cu)	$\rho_4 V_4 C_{p4}\frac{dT_4}{dt} = -(h_{hor4}A_{hor4} + h_{ver4}A_{ver4})(T_4 - T_\infty) - \varepsilon_4\sigma F_{4\text{-}\infty}A_{4\text{-}\infty}(T_4^4 - T_\infty^4) + \frac{(T_2 - T_4)}{R_{2\text{-}4}} - \frac{(T_4 - T_5)}{R_{4\text{-}5}}$	3.9
5	 Lumped 4 (Cu) Lumped 9 (Al_2O_3) Air at T_∞ Lumped 5 (Cu) Lumped 8 (Cu)	$\rho_5 V_5 C_{p5}\frac{dT_5}{dt} = -(h_{hor5}A_{hor5} + h_{ver5}A_{ver5})(T_5 - T_\infty) - \varepsilon_5\sigma F_{5\text{-}\infty}A_{5\text{-}\infty}(T_5^4 - T_\infty^4) + \frac{(T_4 - T_5)}{R_{45}} - \frac{(T_5 - T_8)}{R_{5\text{-}8}} - \frac{(T_5 - T_9)}{R_{5\text{-}9}}$	3.10
6	 q_{gen3} Air at T_∞ Lumped 6 (Cu) (Si) Lumped 3 Lumped 7 (Cu)	$\rho_6 V_6 C_{p6}\frac{dT_6}{dt} = -(h_{hor6}A_{hor6} + h_{ver6}A_{ver6})(T_6 - T_\infty) - \varepsilon_6\sigma F_{6\text{-}\infty}A_{6\text{-}\infty}(T_6^4 - T_\infty^4) + \frac{(T_3 - T_6)}{R_{3\text{-}6}} - \frac{(T_6 - T_7)}{R_{6\text{-}7}}$	3.11

7	 Lumped 6 (Cu) Lumped 9 (Al$_2$O$_3$) Air at T$_\infty$ Lumped 7 (Cu) Lumped 8 (Cu)	$\rho_7 V_7 C_{p7}\frac{dT_7}{dt} = -(h_{hor7}A_{hor7}+h_{ver7}A_{ver7})(T_7-T_\infty) - \varepsilon_7\sigma F_{7-\infty}A_{7-\infty}(T_7^4-T_\infty^4)+\frac{(T_6-T_7)}{R_{6-7}} - \frac{(T_7-T_8)}{R_{7-8}}-\frac{(T_7-T_9)}{R_{7-9}}$	3.12
8	 q$_{gen2}$ q$_{gen3}$ Lumped 9 (Al$_2$O$_3$) Lumped 8 (Cu) Lumped 2 & 3 (Si) Air at T$_\infty$ Lumped 5 & 7 (Cu) Lumped 10 (Al$_2$O$_3$)	$\rho_8 V_8 C_{p8}\frac{dT_8}{dt} = -h_{ver8}A_{ver8}(T_8-T_\infty) - \varepsilon_8\sigma F_{8-\infty}A_{8-\infty}(T_8^4-T_\infty^4) + \frac{(T_9-T_8)}{R_{9-8}}+\frac{(T_5-T_8)}{R_{5-8}}+\frac{(T_2-T_8)}{R_{2-8}} + \frac{(T_7-T_8)}{R_{7-8}}+\frac{(T_3-T_8)}{R_{3-8}}-\frac{(T_8-T_{10})}{R_{8-10}}$	3.13
9	 Lumped 1 (Gate Driver) Lumped 2 & 3 (Si) Lumped 9 (Al$_2$O$_3$) Lumped 5 & 7 (Cu) Air at T$_\infty$ Lumped 8 (Cu)	$\rho_9 V_9 C_{p9}\frac{dT_9}{dt} = -(h_{hor9}A_{hor9}+h_{ver9}A_{ver9})(T_9-T_\infty) - \varepsilon_9\sigma F_{9-\infty}A_{9-\infty}(T_9^4-T_\infty^4)+\frac{(T_1-T_9)}{R_{1-9}}+\frac{(T_2-T_9)}{R_{2-9}} + \frac{(T_3-T_9)}{R_{3-9}}-\frac{(T_9-T_8)}{R_{9-8}}+\frac{(T_5-T_9)}{R_{5-9}}+\frac{(T_7-T_9)}{R_{7-9}}$	3.14
10	 Air at T$_\infty$ Lumped 8 (Cu) Lumped 11 (Cu) Lumped 10 (Al$_2$O$_3$)	$\rho_{10} V_{10} C_{p10}\frac{dT_{10}}{dt} = -h_{ver10}A_{ver10}(T_{10}-T_\infty) - \varepsilon_{10}\sigma F_{10-\infty}A_{10-\infty}(T_{10}^4-T_\infty^4) + \frac{(T_8-T_{10})}{R_{8-10}}-\frac{(T_{10}-T_{11})}{R_{10-11}}$	3.15
11	 Air at T$_\infty$ Lumped 11 (Cu) Lumped 10 (Al$_2$O$_3$)	$\rho_{11} V_{11} C_{p11}\frac{dT_{11}}{dt} = -h_{ver11}A_{ver11}(T_{11}-T_\infty) - \varepsilon_{11}\sigma F_{11-\infty}A_{11-\infty}(T_{11}^4-T_\infty^4) + \frac{(T_{10}-T_{11})}{R_{10-11}}$	3.16

3.4 Physical and Geometric Properties of the Generation II IPEM

In the previous set of equations, only the temperatures are unknown, all of remaining parameters are known values that are obtained from the physical and geometric features of the Generation II IPEM.

For convection heat transfer, in this study the natural convection will be the only component to be considered, because in the experiments there is a considerable airflow (from a fan) that acts on the package. There are several empirical correlations to obtain heat transfer coefficient values. However, for free convection on vertical and horizontal plates, two empirical correlations can be used (based on the studies of Churchill & Chu and McAdams [8, 9], respectively), since they provide the necessary precision in this case. From vertical and horizontal plates exposed to air at atmospheric pressure, these correlations are determined by [8, 9],

$$\overline{\mathrm{Nu}}_{\mathrm{ver}} = \left\{0.825 + \frac{0.387\mathrm{Ra}^{1/6}}{\left[1+(0.492/\mathrm{Pr})^{9/16}\right]^{8/27}}\right\}^{2}, \qquad (3.17)$$

$$\overline{\mathrm{Nu}}_{\mathrm{hor}} = 0.54\mathrm{Ra}^{1/4}$$

where the Nusselt number, Nu, defines the heat transfer coefficient from:

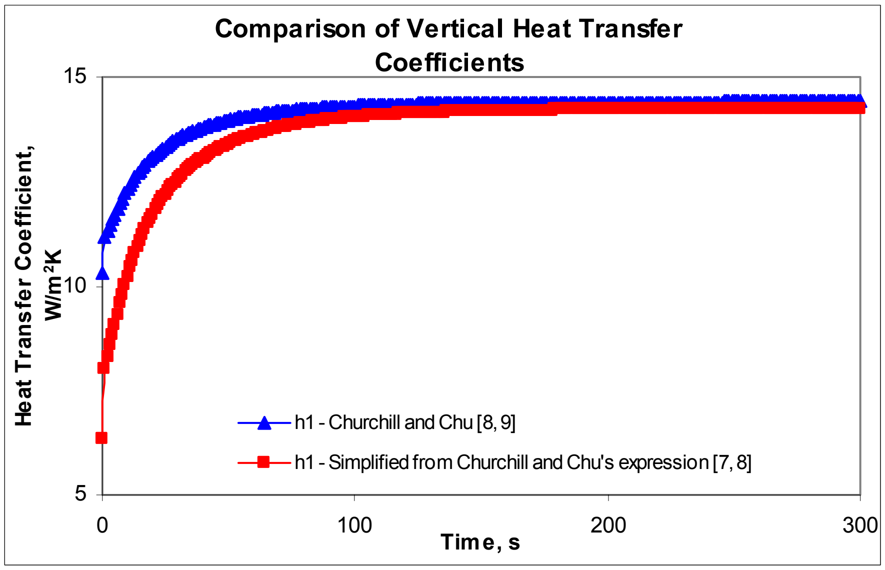
$$\overline{\mathrm{Nu}} = \frac{\mathrm{hL}}{\mathrm{k}}, \qquad (3.18)$$

These empirical equations can be reduced, as shown in [8], and to give results approximated to the ones obtained when previous equations are used (see Figures 3.3(a) and 3.3(b)). The simplified equations for the convective coefficient for horizontal and vertical plates are given by [7, 8]:

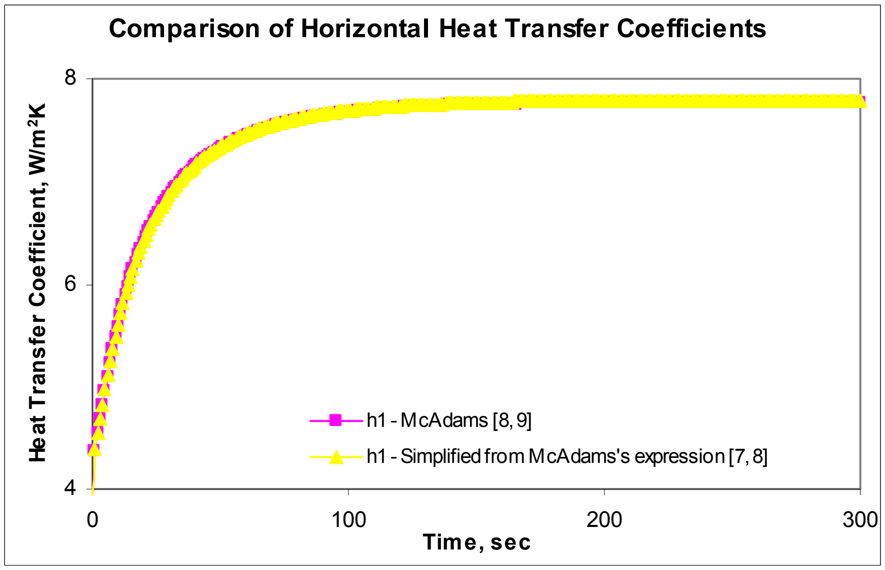
$$\begin{aligned} h_{\mathrm{hor}} &= 1.32\left(\frac{\Delta T}{s}\right)^{1/4} \\ h_{\mathrm{ver}} &= 1.42\left(\frac{\Delta T}{s}\right)^{1/4}, \end{aligned} \qquad (3.19)$$

For microelectronics applications, this coefficient is acceptable within the following range [1]:

$$1\ \mathrm{W/m^2\,K} < \ h \ < 15\ \mathrm{W/m^2\,K}$$



(a)



(b)

Figure 3.3. Comparison of correlations used to calculate the heat transfer coefficients of the gate driver (Lumped 1): (a) on a vertical side; (b) on a horizontal side.

According to Figure 3.3, the use of simplified correlations (Equation 3.19) is valid because the behavior of convective coefficients is identical to the extended equations (Equation 3.17).

The thermal contact resistance is a combination of the contact surface and heat spreading effects. The first is given by [25],

$$R_{\text{cont}\,j-\text{jnext}} = \frac{1/A_{\text{cont.}}}{\left[2.5\left(\frac{k_j k_{\text{jnext}}}{k_j + k_{\text{jnext}}}\right)\left(\frac{m}{\sigma}\right)\left(\frac{P}{H_c}\right)^{0.95} + \left(\frac{k_f}{Y+M}\right)\right]}, \tag{3.20}$$

This empirical correlation is valid for the types of materials used in typical IPEMs. The heat spreading effects are calculated from [8, 9]:

$$R_{\text{spreader}\,j-\text{jnext}} = \frac{1}{A_{\text{cont}}}\left(\frac{\Delta x_j}{k_j} + \frac{\Delta x_{\text{jnext}}}{k_{\text{jnext}}}\right), \tag{3.21}$$

The total thermal contact resistance is

$$R_{j-\text{jnext}} = R_{\text{cont}\,j-\text{jnext}} + R_{\text{spreader}\,j-\text{jnext}}, \tag{3.22}$$

On the other hand, for radiation, in the case of those lumped that only have radiation heat transfer with the environment, the geometry factors are calculated from direct estimation. In the cases of radiation exchange between lumped surfaces, such factors are calculated using: the Reciprocity Relationship, the Summation Rule, tables and curves (in [8, 9]) corresponding to geometry factors for rectangles with a common edge, and also using direct estimation. The expressions used to define the Reciprocity Relation and Summation Rule are, respectively:

$$\begin{aligned} A_j F_{j-\text{jnext}} &= A_{\text{jnext}} F_{\text{jnext}-j} \\ \sum_j F_{j-\text{jnext}} &= 1 \end{aligned}, \tag{3.23}$$

Table 3.2. Physical Properties of the Generation II IPEM

No.	Description	Material	K_j, W/m.K	C_{pj}, J/kg.K	ρ_j, kg/m^3	ε_j	Volume, m^3	Equivalent Length, m
1	Gate Driver	Silicon	124	702	2329	0.86	1.14E-07	2.54E-04
2	Left Chip	Silicon	124	702	2329	0.86	5.70E-08	8.89E-04
3	Right Chip	Silicon	124	702	2329	0.86	5.70E-08	8.89E-04
4	Metallization Layer	Copper	386	390	8900	0.05	1.01E-07	1.24E-02
5	Metallization Layer	Copper	386	390	8900	0.05	1.48E-08	3.56E-03
6	Metallization Layer	Copper	386	390	8900	0.05	1.01E-07	1.24E-02
7	Metallization Layer	Copper	386	390	8900	0.05	1.48E-08	3.56E-03
8	DBC Traces	Copper	386	390	8900	0.05	2.05E-07	2.54E-04
9	Ceramic Substrate	Al_2O_3	26	850	3900	0.34	5.72E-07	8.89E-04
10	DBC Ceramic Layer	Al_2O_3	26	850	3900	0.34	5.13E-08	6.35E-04
11	DBC Base	Copper	386	390	8900	0.05	2.05E-07	2.54E-04

Table 3.3. Geometrical Properties for Convection Effects

	Tranversal Length		Area	
Number	$S_{horizontal}$, m	$S_{vertical}$, m	$A_{horizontal}$, m^2	$A_{vertical}$, m^2
Lumped 1	2.11E-02	2.54E-04	1.79E-04	1.50E-05
Lumped 2	7.19E-03	----	6.36E-05	----
Lumped 3	7.19E-03	----	6.36E-05	----
Lumped 4	4.90E-03	2.54E-04	4.86E-05	9.72E-06
Lumped 5	4.06E-03	1.14E-03	1.49E-05	6.66E-06
Lumped 6	4.90E-03	2.54E-04	4.86E-05	9.72E-06
Lumped 7	4.06E-03	1.14E-03	1.49E-05	6.66E-06
Lumped 8	----	1.14E-03	----	3.25E-05
Lumped 9	3.00E-02	8.89E-04	5.16E-04	9.04E-05
Lumped 10	----	6.35E-04	----	7.23E-05
Lumped 11	----	2.54E-04	----	2.89E-05

To calculate the previous parameters the material physical properties shown in Table 3.2 are used. The dimensions of the Generation II IPEM are given in Appendix

A. The calculated parameters are presented in Tables 3.3 and 3.4. For each variation of the number of lumped, it is necessary to recalculate all parameters, until the Biot number condition is satisfied. The system of eleven equations is the result of that iteration.

Table 3.4. Geometrical Properties for Radiation and Contact Resistance Effects

RADIATION				CONTACT RESISTANCE	
Factor		Area		$R_{j\text{-}jnext}$	Value, K/W
$F_{j\text{-}jnext}$	value	$A_{j\text{-}jnext}$	value, m^2	$R_{1\text{-}9}$	3.61
$F_{1\text{-}\infty}$	9.9E-01	$A_{1\text{-}\infty}$	1.94E-04	$R_{2\text{-}8}$	12.85
$F_{2\text{-}\infty}$	9.4E-01	$A_{2\text{-}\infty}$	3.64E-05	$R_{3\text{-}8}$	12.85
$F_{3\text{-}\infty}$	9.6E-01	$A_{3\text{-}\infty}$	3.64E-05	$R_{2\text{-}9}$	12.46
$F_{4\text{-}\infty}$	1.0E+00	$A_{4\text{-}\infty}$	5.84E-05	$R_{3\text{-}9}$	12.46
$F_{5\text{-}\infty}$	1.0E+00	$A_{5\text{-}\infty}$	3.39E-05	$R_{2\text{-}4}$	10.39
$F_{6\text{-}\infty}$	1.0E+00	$A_{6\text{-}\infty}$	5.84E-05	$R_{3\text{-}6}$	10.39
$F_{7\text{-}\infty}$	1.0E+00	$A_{7\text{-}\infty}$	3.39E-05	$R_{9\text{-}8}$	11.71
$F_{8\text{-}\infty}$	1.0E+00	$A_{8\text{-}\infty}$	4.49E-05	$R_{5\text{-}9}$	11.77
$F_{9\text{-}\infty}$	1.0E+00	$A_{9\text{-}\infty}$	6.06E-04	$R_{7\text{-}9}$	11.77
$F_{10\text{-}\infty}$	1.0E+00	$A_{10\text{-}\infty}$	7.23E-05	$R_{8\text{-}10}$	11.24
$F_{11\text{-}\infty}$	1.0E+00	$A_{11\text{-}\infty}$	2.89E-05	$R_{10\text{-}11}$	10.51
$F_{1\text{-}2}$	4.9E-03	$A_{1\text{-}2}$	5.35E-06	$R_{4\text{-}5}$	16.59
$F_{1\text{-}3}$	5.2E-03	$A_{1\text{-}3}$	5.35E-06	$R_{6\text{-}7}$	16.59
$F_{2\text{-}1}$	5.8E-02	$A_{2\text{-}1}$	3.64E-05	$R_{5\text{-}8}$	11.96
$F_{3\text{-}1}$	4.6E-02	$A_{3\text{-}1}$	3.64E-05	$R_{7\text{-}8}$	11.96

3.5 Numerical Model Implementation and Some Simulation Results

In order to facilitate the solution of the numerical scheme, it was programmed in FORTRAN allowing variable properties, variable convective coefficient, and adaptive time step. The program is presented in Appendix B. The generated code, first verifies that Bi<<0.1. If this is satisfied, then the program reads an external file to obtain the required properties values. The final step is to solve the model's simultaneous equations. The program will abort if Bi>>0.1 since the model will not be accurate. A flowchart of the strategy is shown in Figure 3.4, bellow.

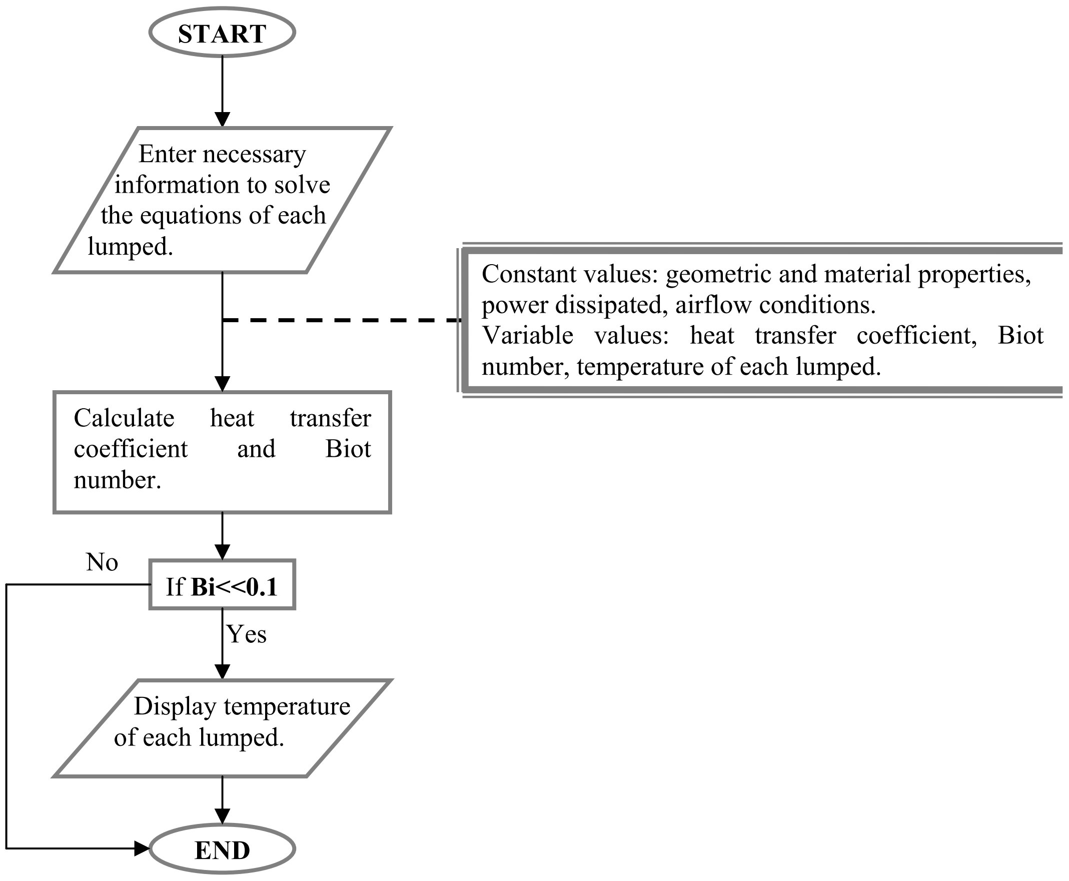

START
Enter necessary information to solve the equations of each lumped.
Constant values: geometric and material properties, power dissipated, airflow conditions.
Variable values: heat transfer coefficient, Biot number, temperature of each lumped.
Calculate heat transfer coefficient and Biot number.
No
If Bi<<0.1
Yes
Display temperature of each lumped.
END

Figure 3.4. Flowchart of the generated FORTRAN code for the solution of the reduced thermal model.

The coded model was executed using the values of power dissipated for the gate driver and silicon devices given for the Generation II IPEM. The constant dissipated powers by the gate driver (lumped 1), the leftmost (lumped 2) and rightmost (lumped 3) semiconductors were 1 W, 7 W and 12 W, respectively. These values were specified into the design stage of this IPEM [5, 21]. A first approach was to expand the model of the Generation II IPEM, with the data presented in the tables, varying the time step to study the model solution stability. Figures 3.5, 3.6 and 3.7 present few results for time step increments. In Figure 3.8, it is observed that for a time increment of 1 sec, the lumped reaches quickly the steady state, whereas for greater time increments, over 4 sec, the transient state range is extended, this is because the method of solution of the model for small time steps is quite stable. No significant change was observed below time steps of one second (see Figure 3.9).

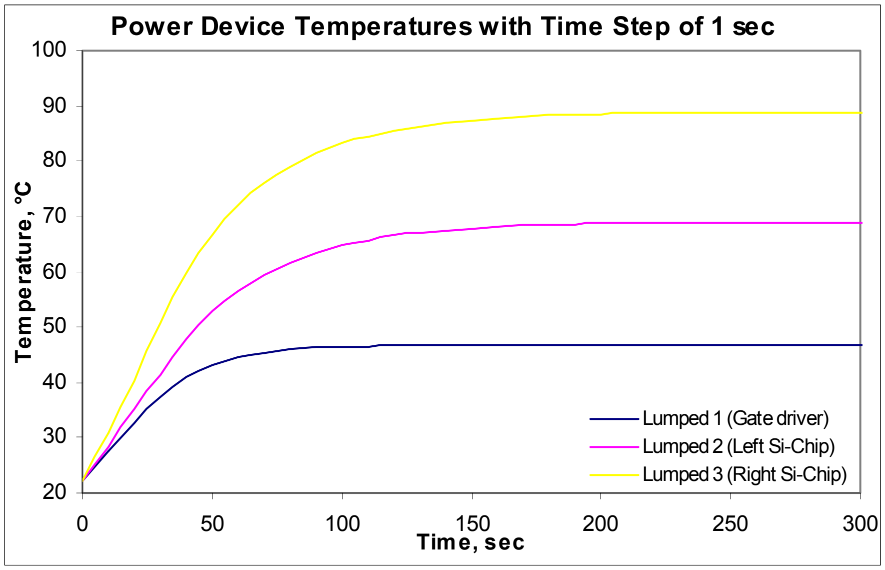


Figure 3.5. Transient curves of the power devices for a time step of 1 sec.

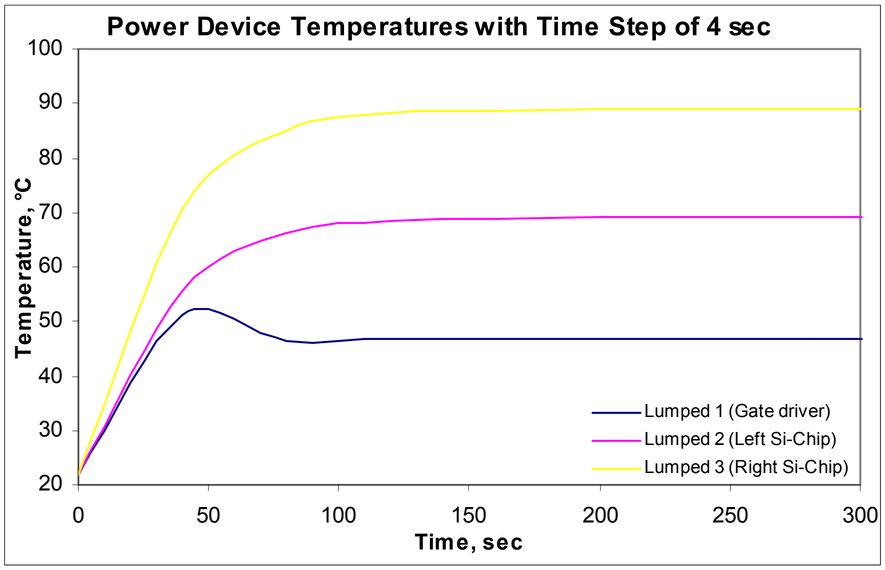


Figure 3.6. Transient curves of the power devices for a time step of 4 sec.

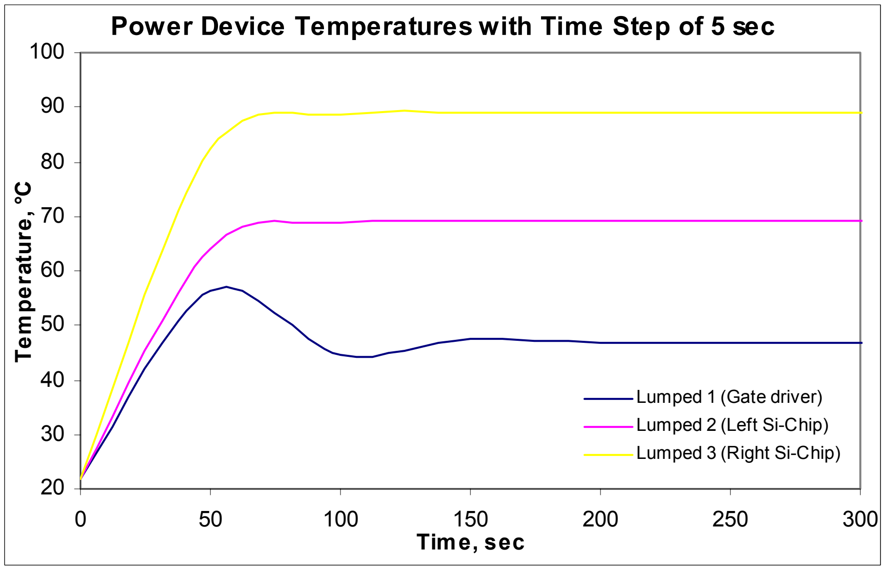


Figure 3.7. Transient curves of the power devices for a time step of 5 sec.

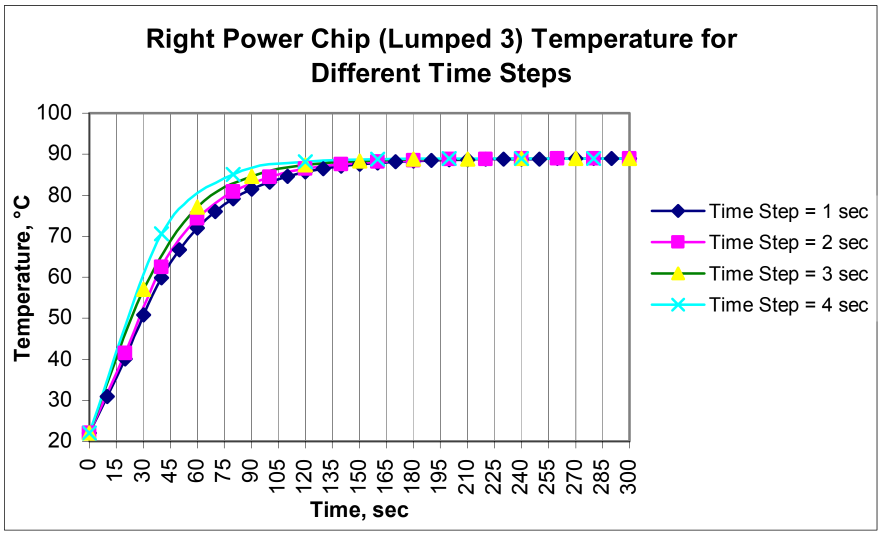
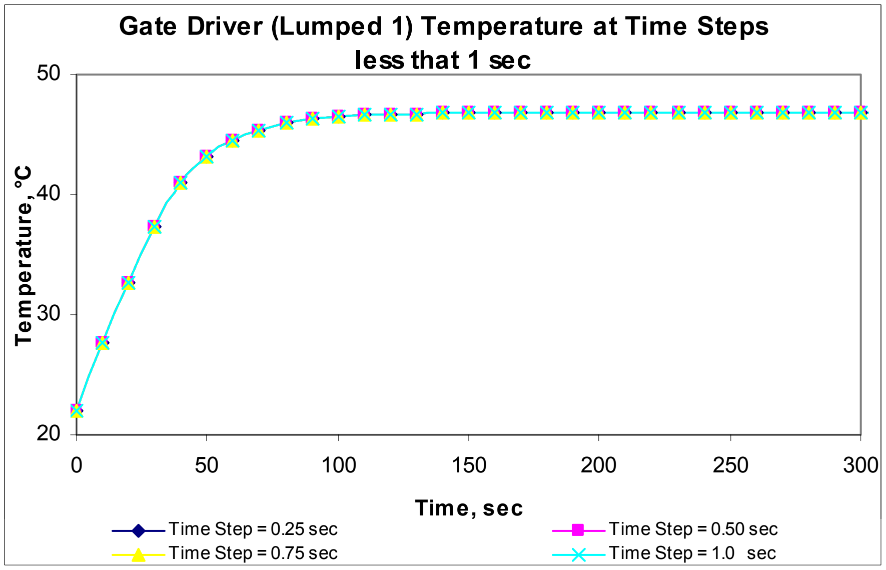
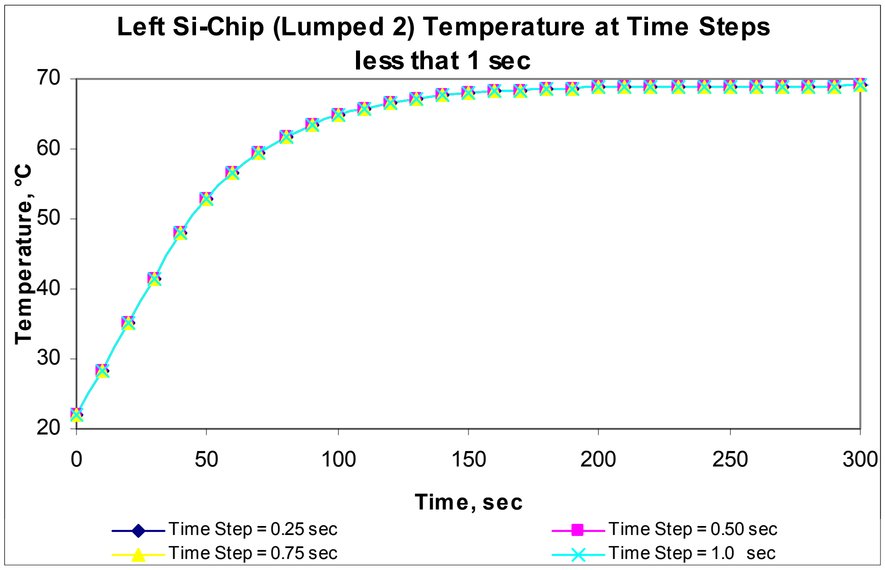


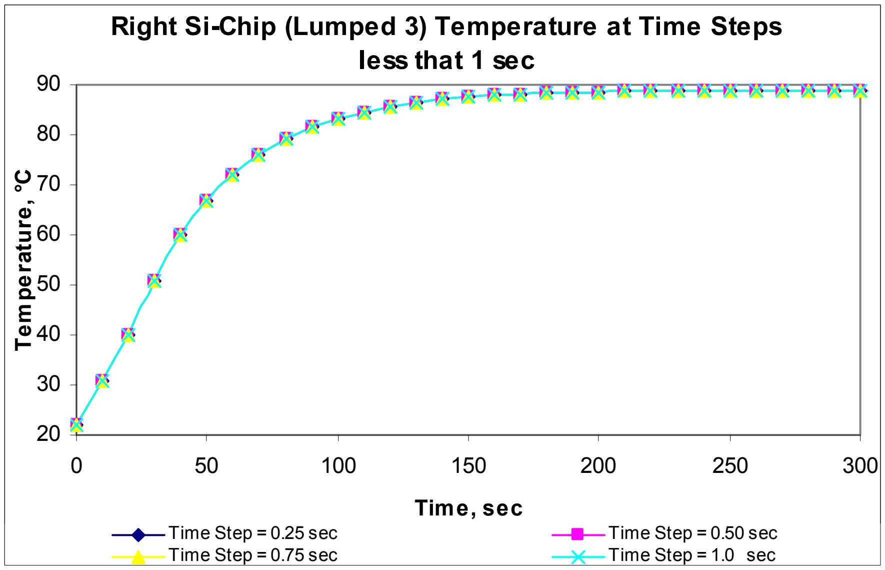
Figure 3.8. Study of the model's stability to the time step integration.



(a)



(b)



(c)

Figure 3.9. Transient curves of the power devices for time steps less that 1 sec: (a) gate driver; (b) left Si-chip; (c) right Si-chip.

For the right chip (Lumped 3) the maximum temperature in steady state was 88.95°C for each of the runs at different time intervals, as is observed in Figure 3.8. For time steps between 1 and 4 seconds, the transient curves for this power device are uniform. In this chart, the optimal time step corresponds to two seconds (this was the final time step used in the full thermal analysis), where the lumped reaches quickly steady state.

An additional interesting feature investigated was the impact of the radiation and convective effects on the electrical performance of the IPEM. Figure 3.10 shows the effects of increased heat dissipation. The convection effects are more import than the radiation effects in general.

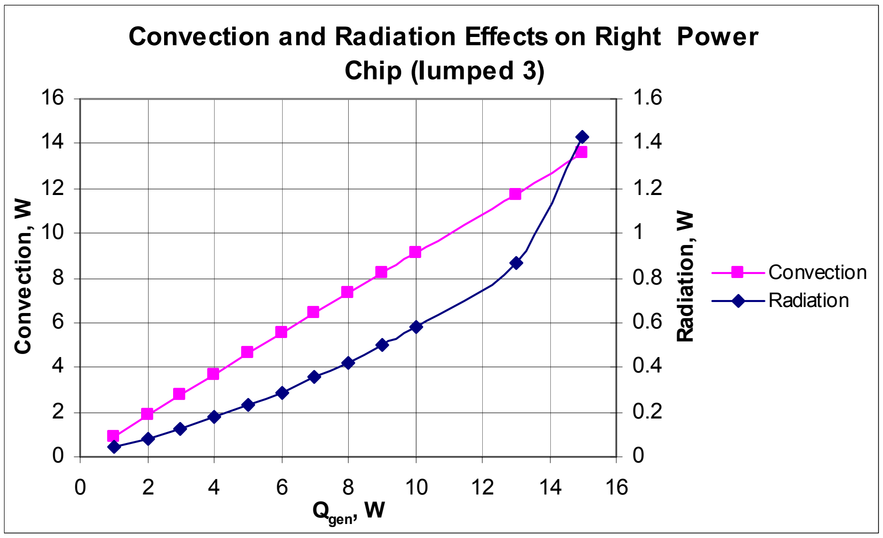


Figure 3.10. Radiation and Convection Effects on Electrical Performance of the Generation II IPEM.

In the Generation II IPEM, when the dissipated power of one of the devices exceeds the 14 W, the radiation starts to be of considerable importance (see Figure 3.10). However, in actual operation, the left and right chip does not exceed the 7 and

12W, and the total dissipated power by the devices together is 20 W (including the gate driver with 1W), and the temperature of them does not surpass the 100°C (a suitable criterion to consider that the radiation is relevant). It could then be said that the radiation effects are not significant in this work. Maybe in other applications, where the power goes over the actual power of the devices, the electrical functionality of the package could be influenced. Here the radiation has been considered because it is desired to demonstrate the generality of the model from the proposed methodology.

3.6 Validation of the Reduced Thermal Model using FLOTHERM^{TM}

Commonly, some CFD simulations are used for the thermal analysis of any electronic component. However, in most instances, such analysis is not simple due to the complexities of components, or their operational conditions, which may result in large computational efforts. For that reason, the use of the reduced model is appropriate. The CFD simulation is used here to demonstrate the validity of the reduced model.

The CFD solver selected to verify reduced model results is FLOTHERM 3.1^{TM}, which is specially designed for the electronics industry. FLOTHERM 3.1^{TM} takes into account all modes of heat transfer: convection, conduction, and radiation effect, according to the following analysis capabilities:

6) Two and three-dimensional.
7) Steady state and transient.
8) Laminar and turbulence flows.

9) Forced, natural and mixed convection.
10) Internal and external flow.
11) Buoyancy and viscous effects.
12) Conduction only, flow only or flow and heat transfer.

Regularly, the steps for every FLOTHERM simulation are to: define the geometry (geometry model acquisition and addition of physical properties and modeling parameters), configure the mathematical model (modeling type and boundary conditions), add the grid, solve, and analyze the results.

Geometry Model

The graphical view of the model data can be obtained from imported solids (from others graphical software and introduced directly into FLOTHERM) using the FLO/MCAD tool, or creating it from FLOTHERM facilities (using the Drawing Board).

In order to build the geometrical model from FLOTHERM, it is necessary to use a rectangular coordinate method for modeling electronic systems and then all the geometry is constructed from fundamental cuboids and prism shapes. The number of cuboids and prisms utilized in an approximate representation determines how accurately the object is represented. Nonetheless, there is a trade-off between modeling accuracy and computer efficiency, as the greater the number of modeling components, the greater the storage requirements and longer the solution time.

From the given dimensions for the Generation II IPEM [21], the corresponding geometric model built in FLOTHERM (into the determined solution domain) is acquired and it is shown in Figure 3.1a.

After the geometry has been described, the next step is defining and adding the physical properties and modeling parameters. Each created primitive element in FLOTHERM must have attached attributes such as material, thermal, surface, surface exchange, equivalent thermal resistance for some applications and source. Modeling parameters should also be assigned to the developed project such as, grid constrain, radiation or not, steady state or transient, and temperature solution option, and solution parameter for the convergence, among others. The simulation uses attached attributes shown in Table 3.2, the dissipated powers used were the same employed in the reduced model, thus the conditions and properties were the same for both cases.

Mathematical Model Configuration

The FLOTHERM 3.1^{TM} solution activates the CFD algorithms, which provides an integration of the fluid flow and heat transfer equations (Navier-Stokes equations) within the solution domain. In the CFD technique used by FLOTHERM, the conservation equations (set of coupled, non-linear, second order, partial differential equations) are discretized by sub-division of the domain of integration within a set of non-overlapping, contiguous finite volumes (or grid cells if the analysis is two-dimensional) that when are associated to the respective boundary conditions result in a set of algebraic expressions.

From available modeling options in the software, the corresponding mathematical model (using FLOTHERM 3.1^{TM}) of the IPEM can be acquired, which includes tree-dimensional conduction, radiation effects, free convection with the surrounding, and transient analysis, using the same initial and operational conditions from the reduced LTCM model. Only the energy equation was used in this simulation, since the thermal analysis proposed with the LTCM model for the Generation II IPEM just involves the heat transfer phenomenon.

Grid and Solution

The grids are spatial locations where the software solves the equations in the computational domain. When the geometry is created, a default grid is also generated. Nevertheless, it is convenient to define a specific grid number according to an appropriate solution accuracy and convergence. For this work, simulations showed that for a grid size greater than 8315 elements the solution does not vary, defining this element number as the optimal grid. In order to obtain the optimal grid size, it was necessary to make many simulations monitoring temperatures corresponding to gate driver, left and right power chip (lumped 1, 2 and 3, respectively). The process to obtain the optimal grid size is presented in Table 3.5.

Table 3.5. Optimal Grid Size at FLOTHERM^{TM} Simulations

GRID SIZE	GATE DRIVER TEMPERATURE, ºC	LEFT POWER CHIP TEMPERATURE, ºC	ROGHT POWER CHIP TEMPERATURE, ºC	COMPUTATIONAL TIME
17528	48.95	67.10	84.01	11 hours, 10 minutes
9012	48.39	66.93	83.11	5 hours, 58 minutes
8315	**48.96**	**67.06**	**83.77**	**5 hours, 17 minutes**
7054	41.54	63.18	82.96	4 hours, 48 minutes
6842	46.30	70.59	89.23	3 hours, 55 minutes

The transient simulation duration must be divided into time steps. It is important to correctly define this time step. Small time steps are required to capture details when there is a rapid rate of change in any variable, as is the case here. A sensitivity analysis was performed to determine the incidence of time step on the problem solution and its computational time. With the established optimal grid size (8315 elements) and considering the optimal time step obtained in the LTCM model's analysis (whole value is 1.75 sec), several sensitivity runs were done (using grid size small, medium and great), varying the time step. The option used, with each grid size, was to run each simulation with a different the time step. According to the LTCM model's stability analysis, an appropriate time step must be established between 0 and 3 seconds (see Figure 3.8), then considering these range, the used values for this parameter were 0.5, 1.75 and 3 seconds.

Table 3.6 summarizes the steady state temperature values of the right Si-chip (lumped 3), as well as the corresponding computational times required to reach each steady state temperature. Here it can be seen that the simulations are sensible to changes in time step and that the optimal time step corresponds to the value obtained in the LTCM model's stability analysis, which one was of 1.75 seconds. When the time step is small the computational time is very large, and when it is large the steady state temperature is not accurate.

Some important computational characteristics are presented in Table 3.7. The FLOTHERM simulation was made in a workstation with UNIX operative system, and the LTCM model was implemented in a personal computer with Windows 95 operative system. The FLOTHERM solution must be as a reference to compare the LTCM model solution in steady state conditions. This is, because FLOTHERM solution uses a very different thermal model to define the thermal transient behavior of the package under study (it is based on multidimensional conduction basically), while the LTCM model is based on thermal contact resistances and heat spreader effects, neglecting the temperature distribution in a specific volume (lumped). However, the simplicity of the analysis by using the LTCM model is clearly manifested, according to the values shown in Table 3.7.

Table 3.6. Steady State Temperatures and Computational Times in the Sensitivity Analysis

TIME STEP sec	**GRID SIZE**		
	6842	**8315**	**17528**
0.50	≈87ºC / 4.5 hours	≈84ºC / 6.8 hours	≈84ºC / 16.3 hours
1.75	≈89ºC / 3.9 hours	≈84ºC / 5.3 hours	≈84ºC / 11.2 hours
3.00	≈64ºC / 2.3 hours	≈93ºC / 3.9 hours	≈80ºC / 8.1 hours

Table 3.7. Computational Quality Evaluation of Both Solutions

	FLOTHERM	LTCM
Computational Time	5 hours, 17 minutes	21 minutes
Step size	1.75 sec	1.75 sec
Number of grids	8315	11 lumped

Figure 3.11 shows the upper surface temperature distribution of the Generation II IPEM obtained from the developed simulation in FLOTHERM using a time step of 1.75 seconds and a grid size of 8315 elements. In concordance with the solution obtained with the LTCM model, FLOTHERM solution also report the highest temperature at the right power chip (lumped 3), this device is the principal heat dissipation element.

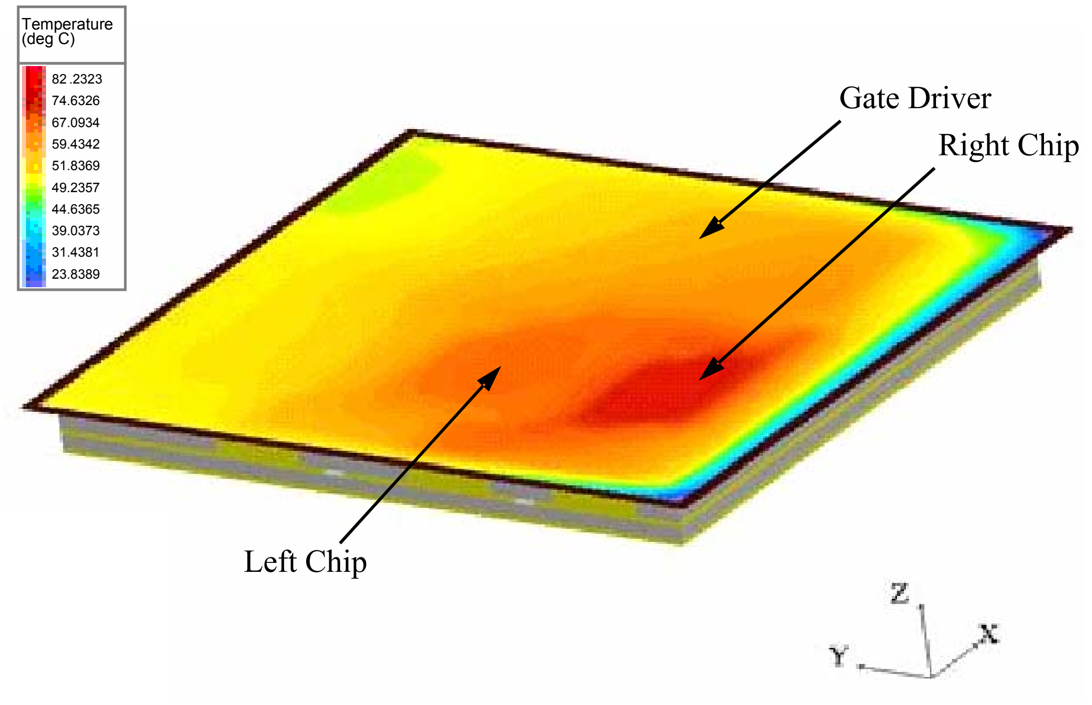


Figure 3.11. IPEM Visualization surface from FLOTHERM.

The conduction phenomenon dominate the thermal transient obtained from FLOTHERM, whereas in the case of the LTCM model, the heat spreader effects and thermal contact resistance represent such heat transfer mechanism, then it is to hope that the resulting transients from both analysis will tend to be different. In Figures 3.12, 3.13 and 3.14 the comparisons between the FLOTHERM simulation and the reduced model for the power devices are illustrated. Considering the existing differences in both analyses, simulation results from FLOTHERM compare very favorably with the LTCM model. The validation of the reduced model against simulation data for gate driver, left power chip and right power chip generated average errors (for the steady state temperature) of 4.75%, 3.79% and 5.37%, respectively. The remaining IPEM's comparison points are presented in Table 3.8. In FLOTHERM, first the lumped were identified and in each one was placed a monitor point (at the center of mass), which collects the steady state temperature for that lumped.

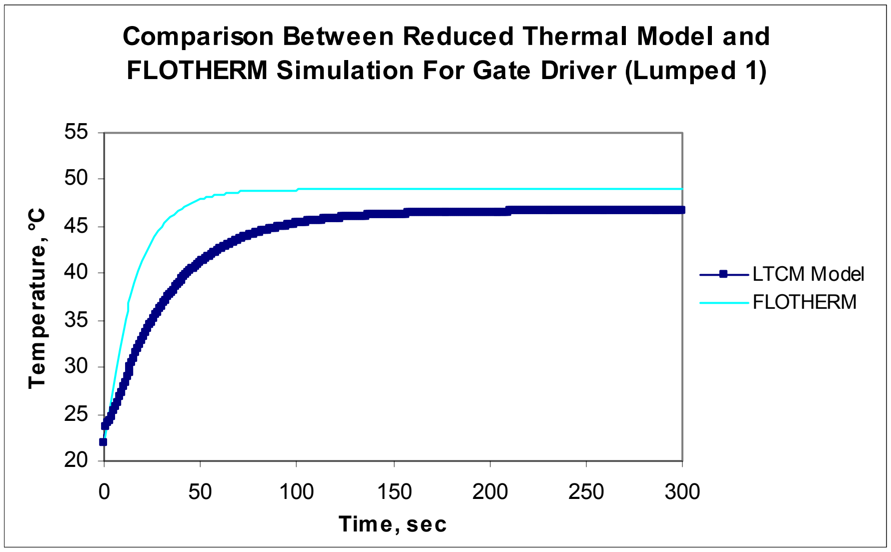


Figure 3.12. LTCM vs. FLOTHERM for Lumped 1, with a Power Density of $5.574x10^3$ W/m^2.

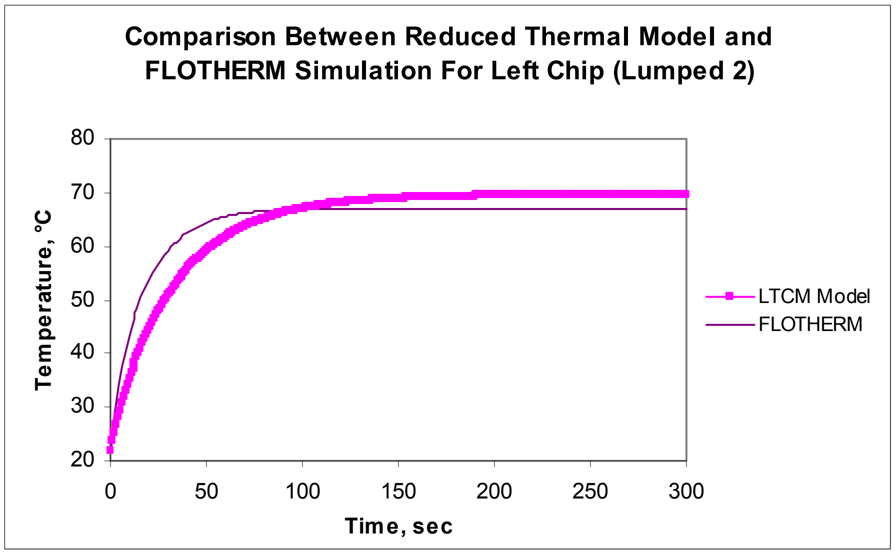


Figure 3.13. LTCM vs. FLOTHERM for Lumped 2, with a Power Density of 1.101x10^5 W/m^2.

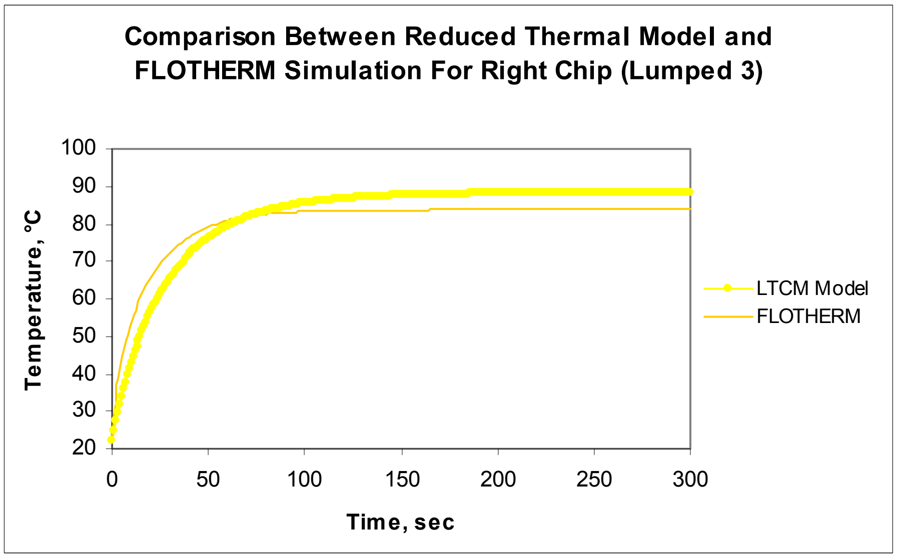


Figure 3.14. LTCM vs. FLOTHERM for Lumped 3, with a Power Density of 1.888x10^5 W/m^2.

Table 3.8. Steady State Temperature Errors between FLOTHERM and LTCM

Component (Lumped)	LTCM Model SS Temp, °C	FLOTHERM SS Temp, °C	% Error
Left Cu - metallization layer (Lumped 4)	6378	61.38	3.91
Left Cu - metallization frame (Lumped 5)	58.24	55.86	4.27
Right Cu - metalliz. layer (Lumped 6)	79.49	75.19	5.73
Central Cu - Metalliz. frame (Lumped 7)	67.44	63.61	6.01
DBC copper trace (lumped 8)	50.39	46.56	8.23
Al_2O_3 - ceramic substrate (Lumped 9)	61.51	57.24	7.46
Al_2O_3 – DBC ceramic layer (Lumped 10)	44.72	41.06	8.92
Copper base (Lumped 11)	43.11	39.44	9.29

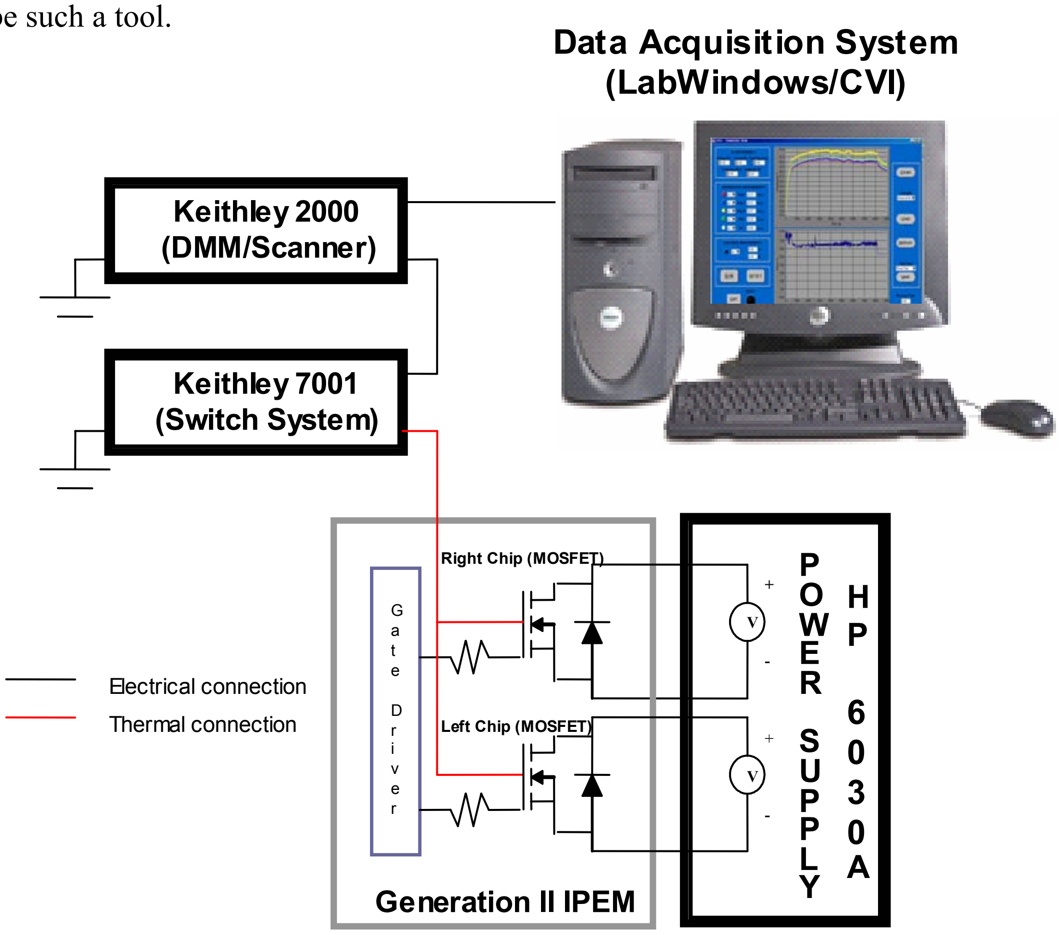
According to Table 3.8, the tendency is that the resulting comparisons in the lumped close to hot points are more accurate. Meanwhile for the points far from the power devices is not as accurate. The cause of this discrepancy is that in the point far from the hot point the heat conduction becomes more relevant. Then, the LTCM model may be not appropriate to compare the steady state temperature and, of course, the transient thermal analysis with the software tool solution in such points. However, the fact that LTCM is not as accurate for cold points as for hot spots, it does not imply that the LTCM is not adequate or that it cannot be used to obtain preliminary temperature values in such points, since the maximum error was 9.29% (a large error is considered a error value over 10%). The lumped position affect the LTCM model accuracy, this is, each lumped must be defined so that this one have enough contact area and heat spreader effect compared with its adjacent lumped. For example the central lumped have small steady state temperature errors, lumped 2, 4 and 5, while for the lumped 3, 6 and 7, with the same geometries and adjacent lumped, large errors are achieved.

CHAPTER 4

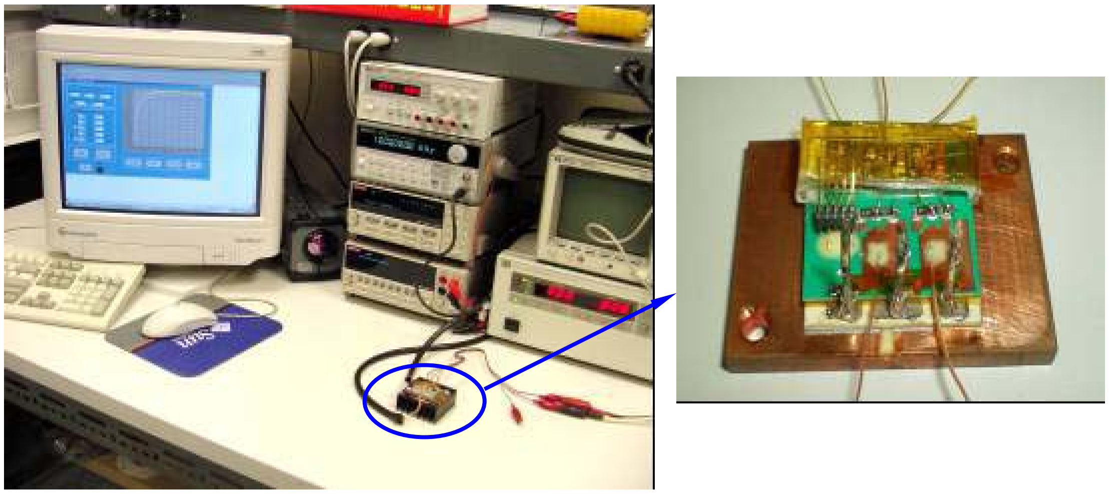
EXPERIMENTAL MODEL VALIDATION

4.1 Introduction

An important component in electrothermal model validation is the comparison with actual data. It is essential to develop a tool that allows having a real description of the interaction between thermal and electrical parameters (within the package under study), where these parameters can be extracted and analyzed. An experimental system can be such a tool.



(a)



(b)

Figure 4.1. Experimental testbed for low-speed thermal response: (a) set-up picture; (b) IPEM picture.

Two experimental set up, for low- and high-speed thermal responses, were developed. The first one is employed to validate the LTCM-heat sink model with constant input (simulated using the program developed in FORTRAN), and the second one is used to validate the electrothermal data from the implemented model in SABER simulator (based on LTCM with a variable input, which is established for the electrical component).

4.2 Experimental System for Low-Speed Thermal Response

The experimental set-up consists of a computer-based data acquisition interface based on LabWindows/CVI from National Instrument with digital power supplies, multimeters and switches, as shown in Figure 4.1.

Basically, the experiment works in the following form: Power is supplied (HP 6030A Power Supply), separately, to each MOSFET; this power produces temperature changes (inside the device and the other components of the package) that are collected by five thermocouples, placed in specific points (gate driver, MOSFETs, DBC base and heat spreader). Gathered data from thermocouples are received by a multiplexer thermal card (Keithley 7014 Card) inside a switch system (Keithley 7001 Switch System). A multimeter (Keithley 2000 Multimeter) reads temperature data from the switch system and stores them temporarily. The data acquisition system (the project was named "transient ipem") reads temperature values, from the multimeter, and store them, for later display and to plotting in a computer window. In the same way, whenever thermocouples collect a set of values, the power supply acquires the voltage and current values and sends them to the transient ipem program, where the values are stored, displayed and plotted in a computer window, see Figure 4.1a.

4.2.1 Data Acquisition System (Transient IPEM Program)

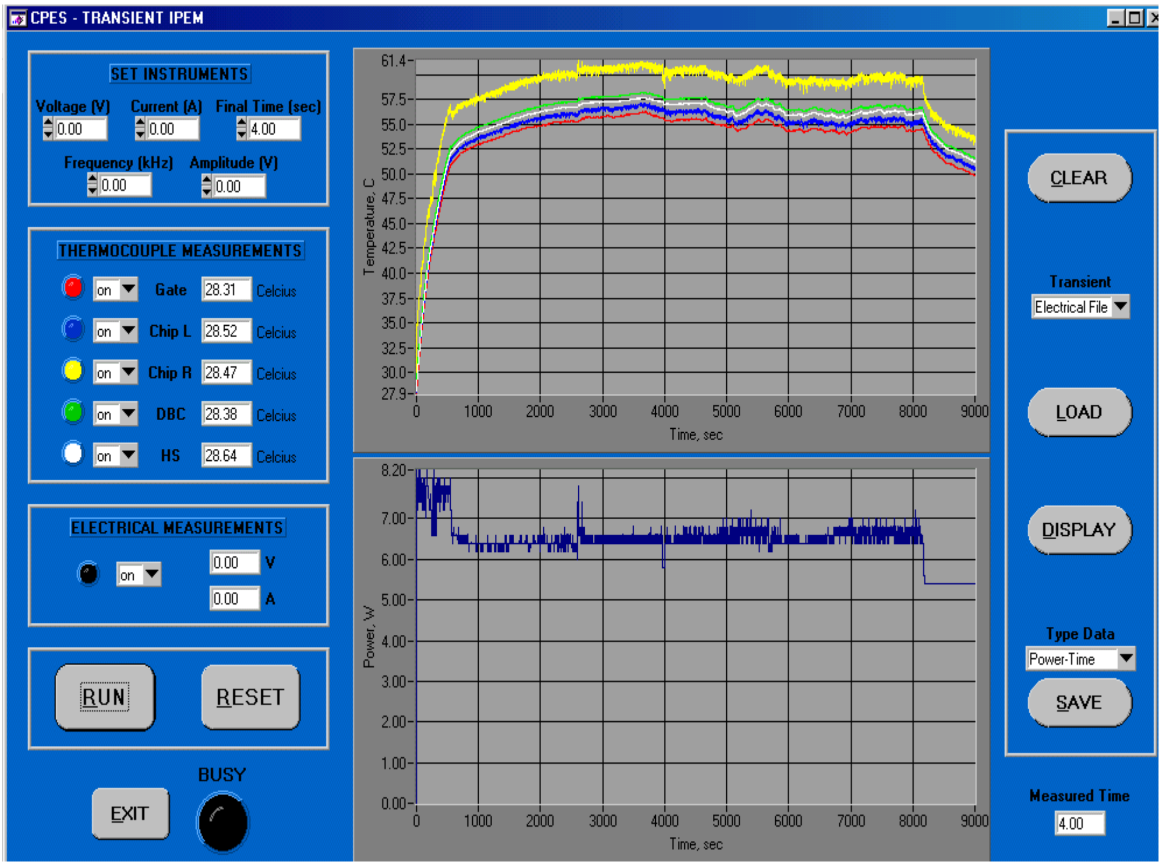
LabWindows/CVI is an integrated interactive development environment with development tools that allows easy creation, configuration, and display of measurements on a graphical user interface (GUI). LabWindows/CVI is used to create virtual instruments (combination of hardware and software that provide complete flexibility of designing and controlling the elements of stand alone or embedded instruments from a computer system) to acquire, analyze, and display data from an experiment. The virtual instrumentation application is created using all features of the American National

Standard Institute (ANSI) C programming language along with LabWindows/CVI built-in tools and libraries.

LabWindows/CVI contains a drag and drop editor for creating user interfaces, tools for automatic code generation, and a complete multithreaded ANSI C environment for building test, measurement, control, and automation applications. Multithreading is a method of programming in which a program can perform more than one operation at a time. The used methodology by LabWindows/CVI to make any data acquisition system is very simple:

- Creating the graphical user interface (GUI) with defined controls for a specific application.
- Making the source code, in C++, required to define the control functions at the created GUI.
- Obtaining or making instrument driver codes (also in C++) of each one of the instruments used in the experiment, with which the program has communication.
- Establishing the communication between the program and instruments. The communication is based on the General Purpose Interface Bus (GPIB) protocol.

LabWindows/CVI program acquires temperature at different points in the Generation II IPEM through the use of thermocouples, at the same time it collects the changes in voltage and current into power chips. The developed program at LabWindows/CVI for this experiment was named “transient ipem”, and the transient ipem program panel is shown in the Figure 4.2. The complete C code can be found in Appendix C of this document.


CPES - TRANSIENT IPEM

SET INSTRUMENTS
Voltage (V) 0.00
Current (A) 0.00
Final Time (sec) 4.00
Frequency (kHz) 0.00
Amplitude (V) 0.00

THERMOCOUPLE MEASUREMENTS
on Gate 28.31 Celcius
on Chip L 28.52 Celcius
on Chip R 28.47 Celcius
on DBC 28.38 Celcius
on HS 28.64 Celcius

ELECTRICAL MEASUREMENTS
on 0.00 V
0.00 A

Transient: Electrical File
Type Data: Power-Time
Measured Time 4.00

Figure 4.2. Data acquisition system program created in LabWindows/CVI.

In the transient ipem program panel (see Figure 4.2), the first box (named SET INSTRUMENTS) is used to fix program input parameters (voltage, current, waveform frequency and amplitude), in addition to establishing the final time of the experiment, it is to say, the required time that the device needs to reach the steady state temperature. The central box, named THERMOCOUPLE MEASUREMENTS, is used to define which thermocouples will be measured (which points with thermocouple measurements:

gate driver, left and/or right power chip, DBC base, and/or heat spreader). When measurement points are selected (putting “on” or “off” to each thermocouple), those points in “on”, display actual point temperatures in each time step. All temperatures are stored until the final time is reached. Similarly, in the next box named ELECTRICAL MEASUREMENTS, the option to acquire electrical parameters, such as voltage and current (and with these the power), in each time step is available.

The main function is defined by the RUN button. With this button, program input parameters are set and the program run in order to acquire temperatures of selected points. In the same box, the RESET button is located, which is used to return to off-conditions.

The last box, at the right side, is used for other functions, such as load, display, and save the data, and to clear graphics. The buttons in this box allow to manipulate measured or file data. The program acquire and store three types of datasets: temperature, power, and voltage-current. The data can be analyzed from the panel or extracted to analyze it in an excel file.

4.2.2 Results

With the experimental testbed used both thermal and electrical transient curves were obtained. In each run, temperature, voltage, and current were acquired at each time step. From the input voltage and current, the power dissipated by the device can be calculated.

According to Figure 4.1.a, each device is supplied by an electrical source, but it cannot be made simultaneously. It is necessary that while a device is turned on, the other remains as a passive thermal element, that is to say, the other device only conducts the generated heat in the turned on device. In view of that, a set of experiments with different electrical sources (below the allowed maximum value by device) applied to left power chip, and other set of experiments from the right power chip, were made. In total, fourteen experimental runs were made, seven of them corresponding to the left power chip and the remaining seven to the right power chip.

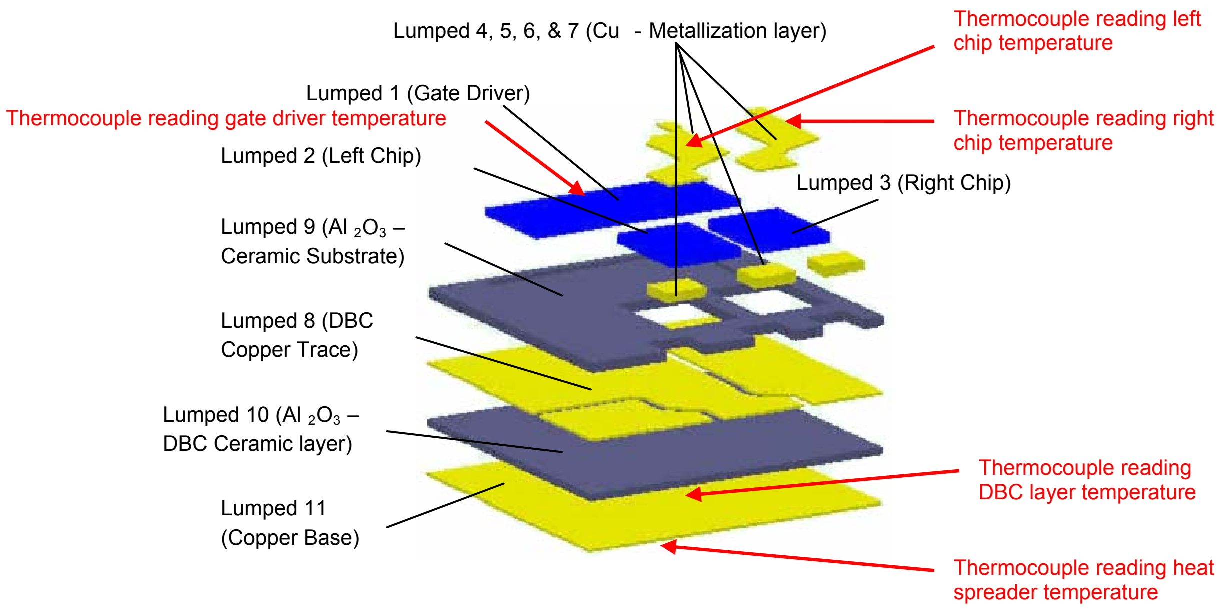
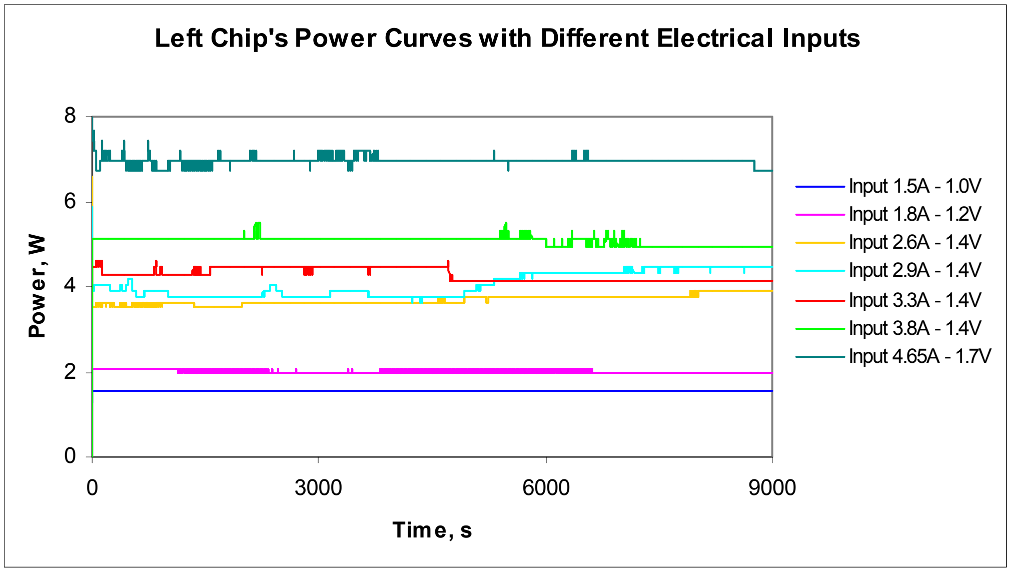
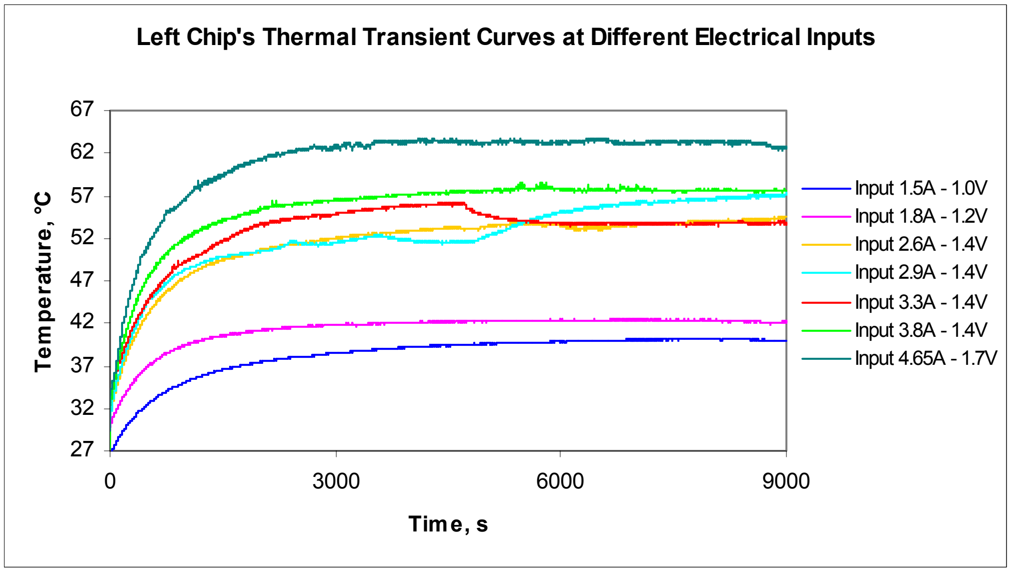

Lumped 4, 5, 6, & 7 (Cu - Metallization layer)
Thermocouple reading left chip temperature
Lumped 1 (Gate Driver)
Thermocouple reading gate driver temperature
Thermocouple reading right chip temperature
Lumped 2 (Left Chip)
Lumped 3 (Right Chip)
Lumped 9 (Al_2O_3 – Ceramic Substrate)
Lumped 8 (DBC Copper Trace)
Lumped 10 (Al_2O_3 – DBC Ceramic layer)
Thermocouple reading DBC layer temperature
Lumped 11 (Copper Base)
Thermocouple reading heat spreader temperature

Figure 4.3. Localization of thermocouples at the Generation II IPEM.

In each experimental sample the temperature were read and stored in five specific places of IPEM, as shown in Figure 4.3.

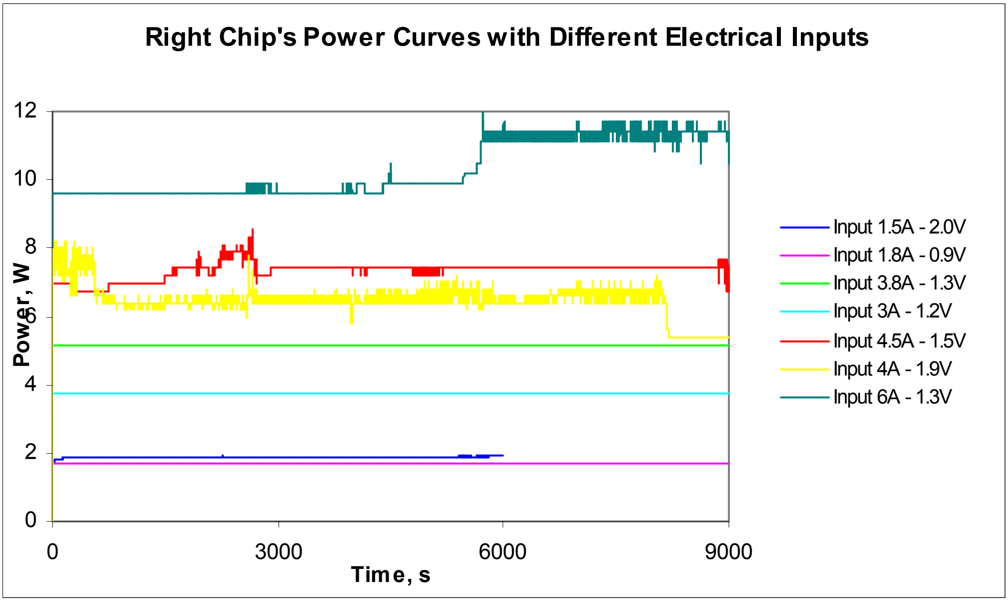


(a)

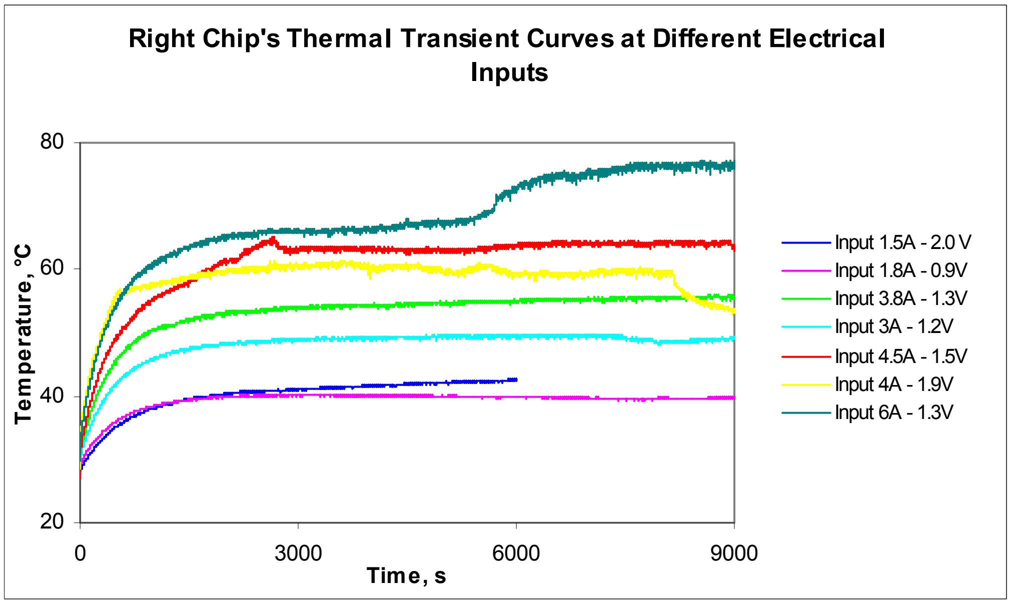


(b)

Figure 4.4. Left power chip’s transient curves: (a) electrical; (b) thermal.



(a)



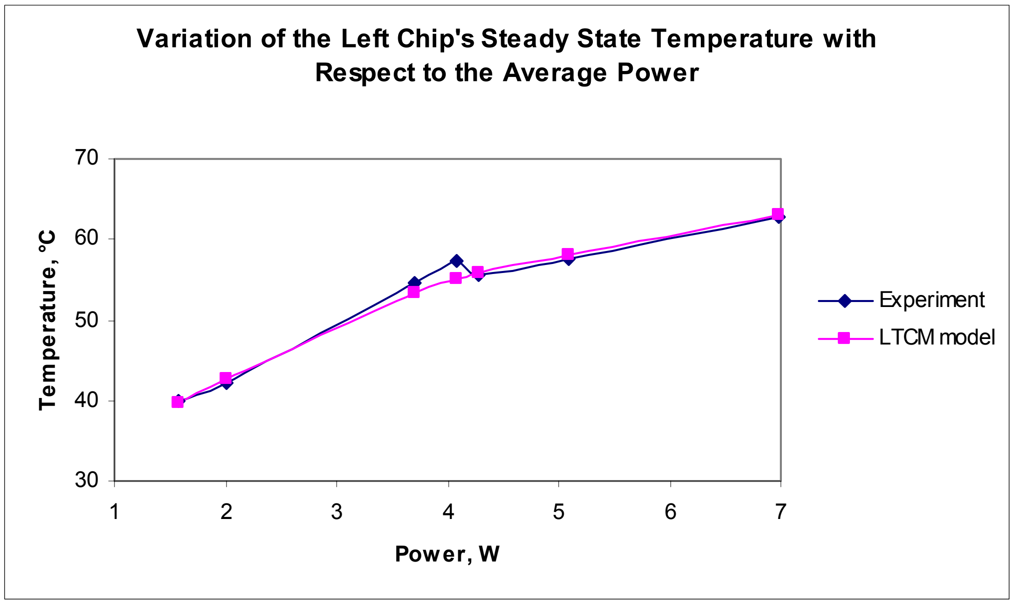
(b)

Figure 4.5. Right power chip's transient curves: (a) electrical; (b) thermal.

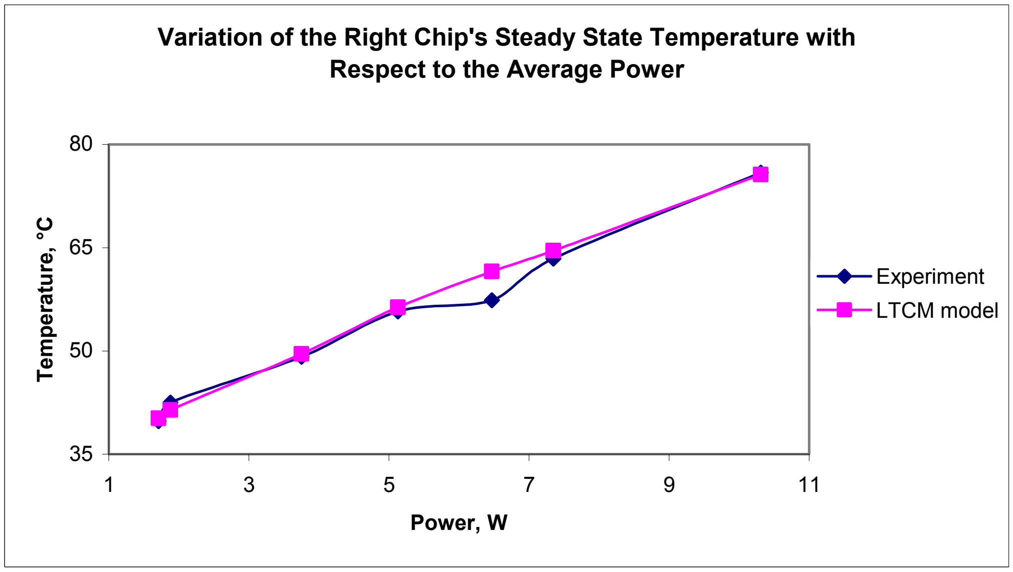
In Figure 4.4, thermal and electrical transient curves of the left chip with diverse electrical inputs are presented. Here, it is possible to observe the device electrical nonlinearity with respect to the temperature dependency. In Figure 4.4.a, when the input provided power is low, the electrothermal process can be considered as only thermal, such as in the case of the input of 1.5A – 1.0V (equivalent to 1.5W). But as the input power increases, the thermal effects become more evident when a specific temperature value is reached. Likewise, Figure 4.5 shows, for the right silicon-chip, the power and temperature variation with respect to time.

In order to compare the experimental data with simulated data, it is necessary that the simulation considers a heat sink effect, because until this moment, in the LTCM model it has not been necessary to include it. In the experiment, the Generation II IPEM is mounted on a heat sink. A simple heat sink analysis was developed and included in the LTCM model (see Appendix D).

Consequently, the average power value of each experimental sample was plotted against the steady state temperature value. In the same way, using the LTCM model steady state temperatures were obtained from the average experimental power values. Figure 4.6 shows the incidence of the power on the steady state temperature in the thermal analysis. In the experimental curves, there are some values where the behavior is irregular (peak points), this can be due to external factors such as the power supply performance, or air flow variation. In general terms, both curves compare very well.



(a)



(b)

Figure 4.6. Comparison between experimental and simulated values of the steady state temperature against average power for: (a) left Si-chip; (b) right Si-chip.

Using the Figure 4.6, it is possible to choose the simulations that better compare with the experimental curves, for the same input conditions in the left and right power chips. As a result, for the left chip the selected average power inputs are 1.58, 3.7 and 6.98W, and for the right chip are 7.71, 5.13 and 7.35W. The comparisons of experimental data with simulations results are presented from Figure 4.7 to Figure 4.12.

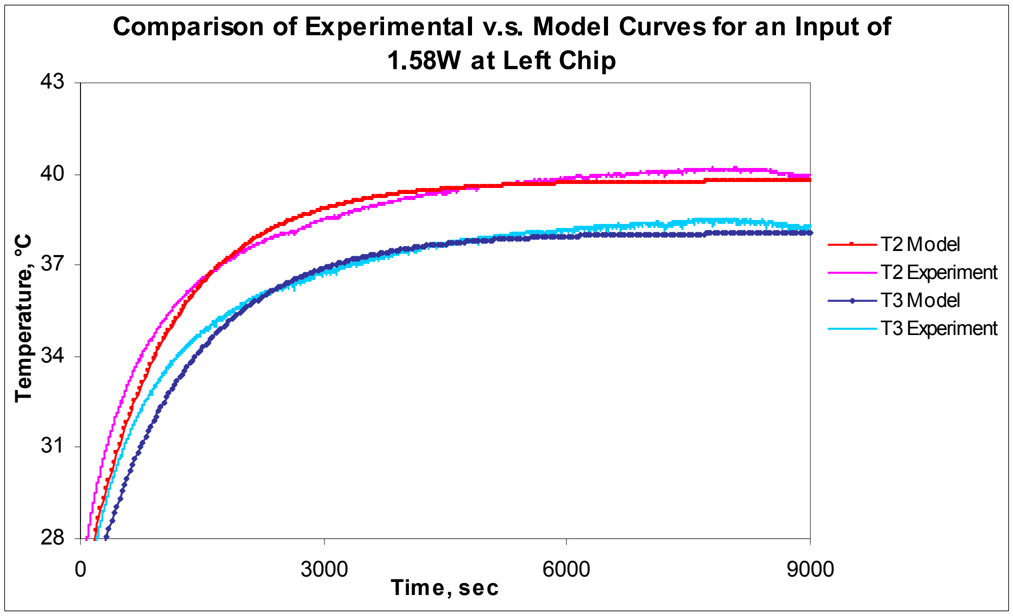


Figure 4.7. Experimental data and the LTCM-heat sink model results for 1.58W left chip power input.

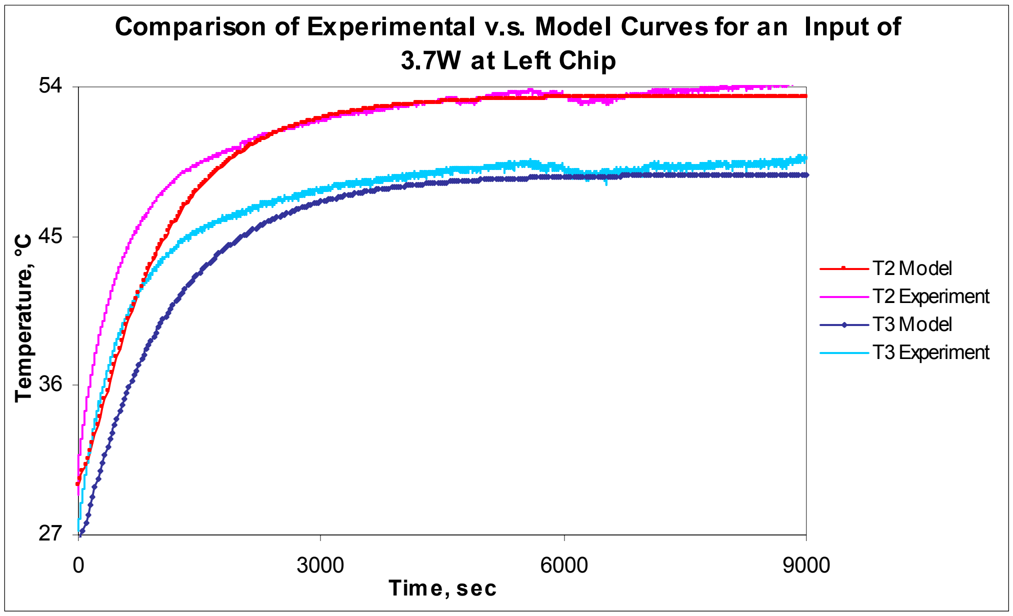


Figure 4.8. Experimental data and the LTCM-heat sink model results for 3.7W left chip power input.

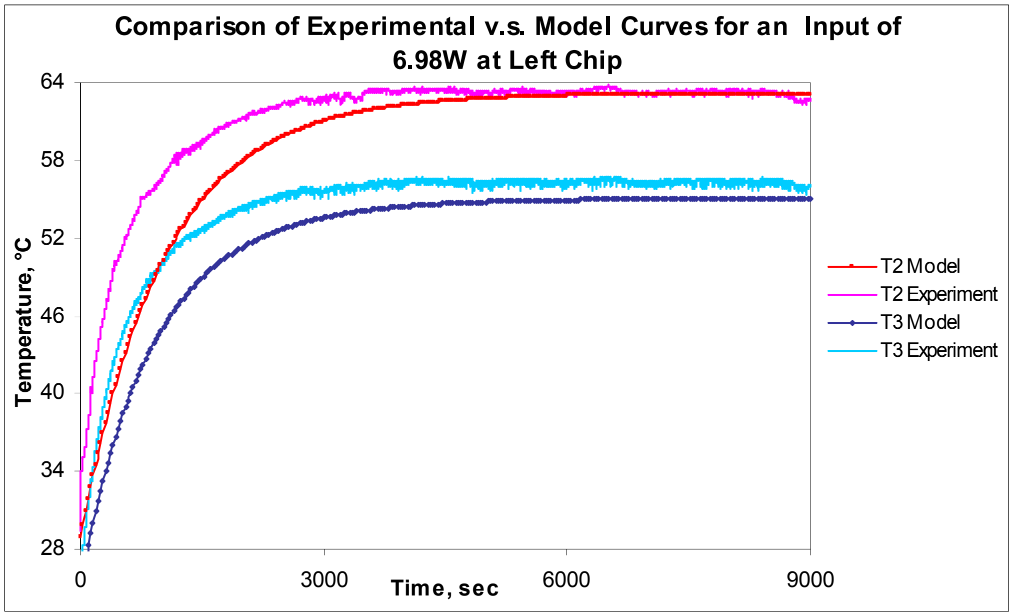


Figure 4.9. Experimental data and the LTCM-heat sink model results for 6.98W left chip power input.

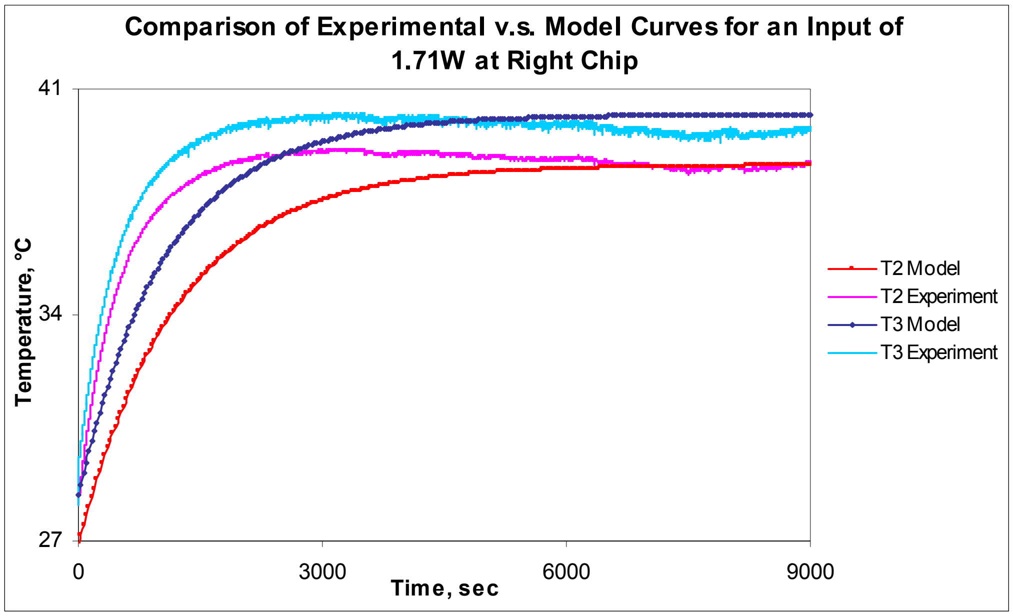


Figure 4.10. Experimental data and the LTCM-heat sink model results for 1.71W right chip power input.

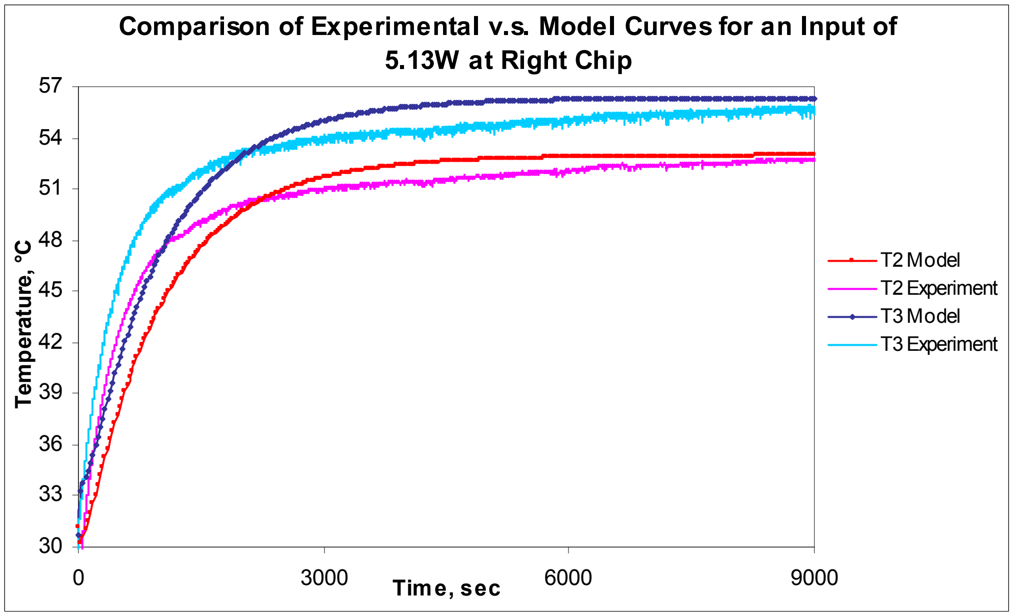


Figure 4.11. Experimental data and the LTCM-heat sink model results for 5.13W right chip power input.

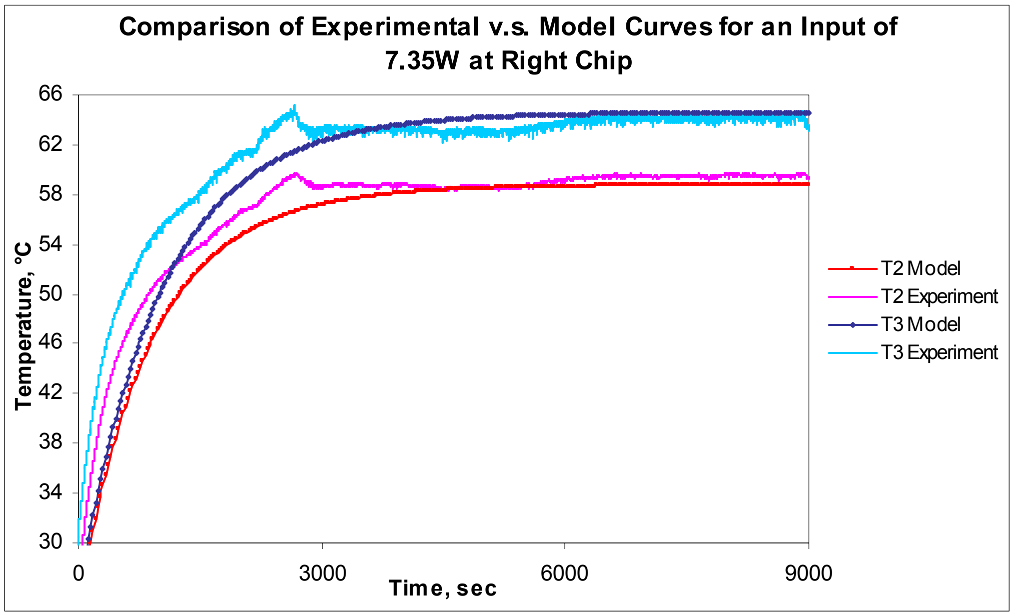


Figure 4.12. Experimental data and the LTCM-heat sink model results for 7.35W right chip power input.

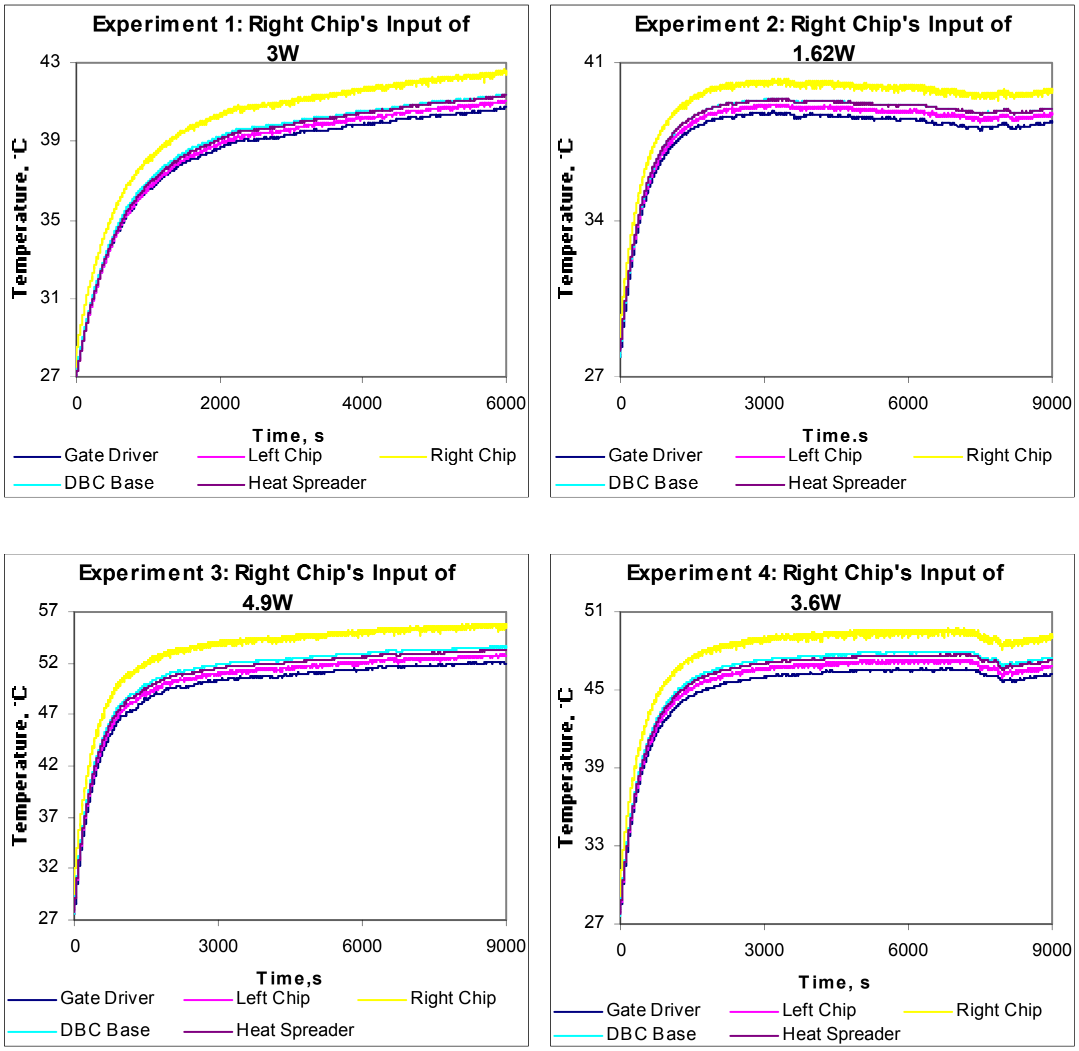
The previous curves show that for the left chip the comparisons are better, this is because the left chip is in the center of the IPEM and the heat dissipation is more symmetric, therefore the presumption of an uniform temperature in some areas (LTCM) is more realistic, whereas for the right chip the transient curves are less accurate. However in both cases the steady state temperatures compare very well.

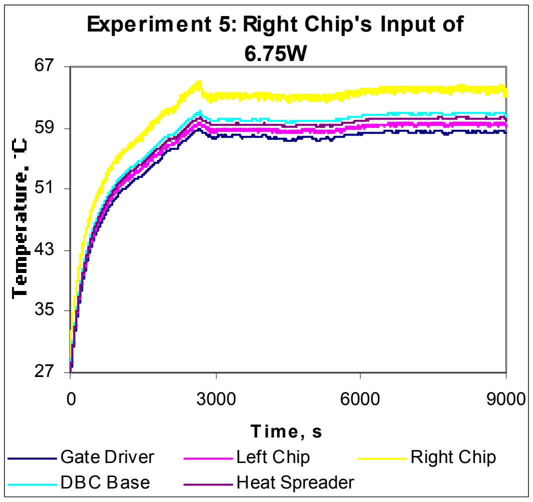
Table 4.1. Steady State Temperature Error Percentage of the Chip Turned On

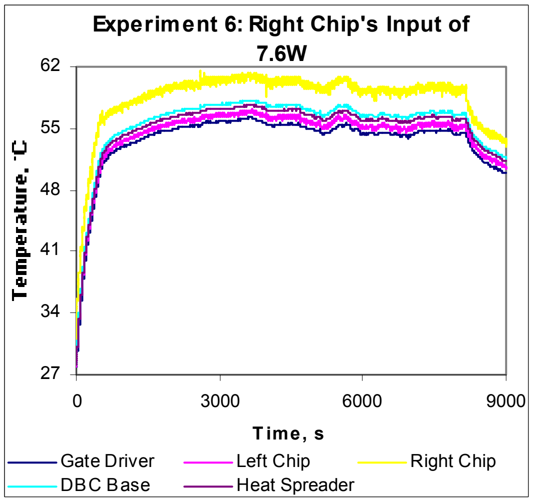
Left Chip		Right Chip	
Power Input, W	**% Error - SS Temp**	**Power Input, W**	**% Error - SS Temp**
1.58	0.53	1.71	1.06
3.70	1.96	5.13	1.08
6.98	0.76	7.35	1.84

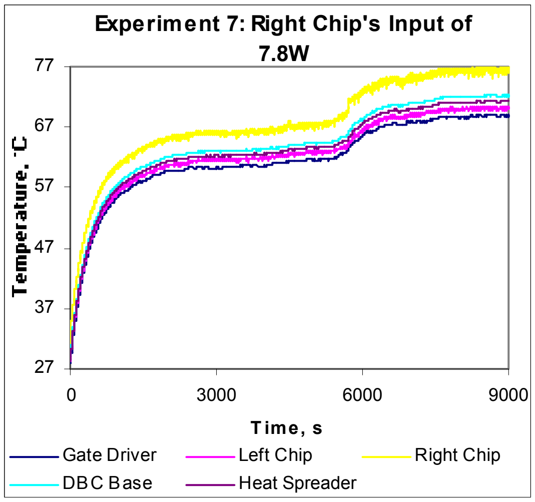
Table 4.1 presents the steady state temperature errors for the turned on device. In general the error increases when the power input is increased, but in the case of the power input of 3.7W applied to the left chip the error is irregular maybe because the

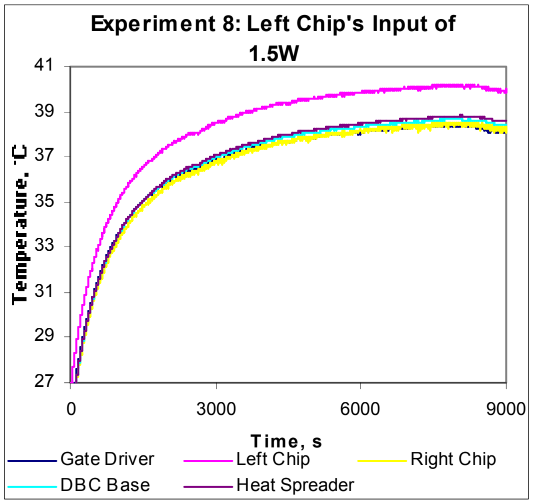
experimental curve is not very uniform. In Table 4.1 and in Figure 4.7 it is shown that for a low constant power input the comparison is favorable. It is important to emphasize that in the experiment the power vary whereas in the simulations the power is constant (corresponding to the average power in the experiment), due to this fact all experimental curves reveal a moderate power incidence that is not present in the simulations. Figure 4.13 shows all experimental temperature profiles for Generation II IPEM obtained from the set up of the low-speed thermal response experiment.

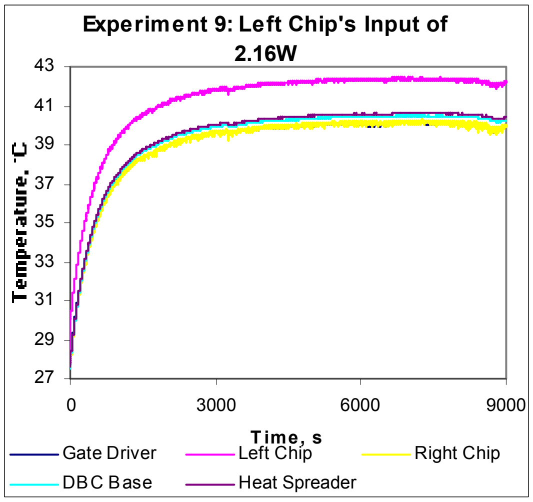


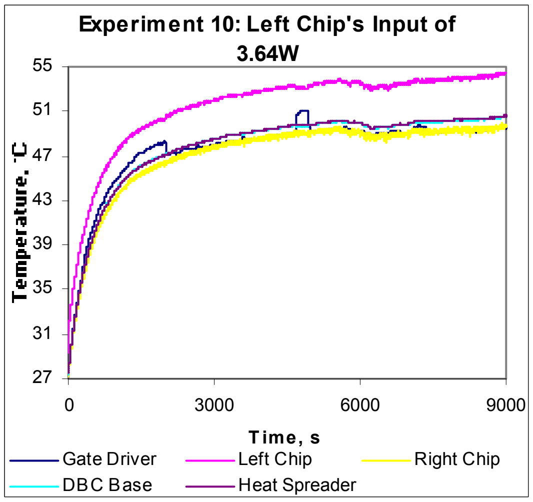

Experiment 5: Right Chip's Input of 6.75W
Temperature, °C
27
35
43
51
59
67
0
3000
6000
9000
Time, s
Gate Driver
Left Chip
Right Chip
DBC Base
Heat Spreader


Experiment 6: Right Chip's Input of 7.6W
Temperature, °C
27
34
41
48
55
62
0
3000
6000
9000
Time, s
Gate Driver
Left Chip
Right Chip
DBC Base
Heat Spreader


Experiment 7: Right Chip's Input of 7.8W
Temperature, °C
27
37
47
57
67
77
0
3000
6000
9000
Time, s
Gate Driver
Left Chip
Right Chip
DBC Base
Heat Spreader


Experiment 8: Left Chip's Input of 1.5W
Temperature, °C
27
29
31
33
35
37
39
41
0
3000
6000
9000
Time, s
Gate Driver
Left Chip
Right Chip
DBC Base
Heat Spreader


Experiment 9: Left Chip's Input of 2.16W
Temperature, °C
27
29
31
33
35
37
39
41
43
0
3000
6000
9000
Time, s
Gate Driver
Left Chip
Right Chip
DBC Base
Heat Spreader


Experiment 10: Left Chip's Input of 3.64W
Temperature, °C
27
31
35
39
43
47
51
55
0
3000
6000
9000
Time, s
Gate Driver
Left Chip
Right Chip
DBC Base
Heat Spreader

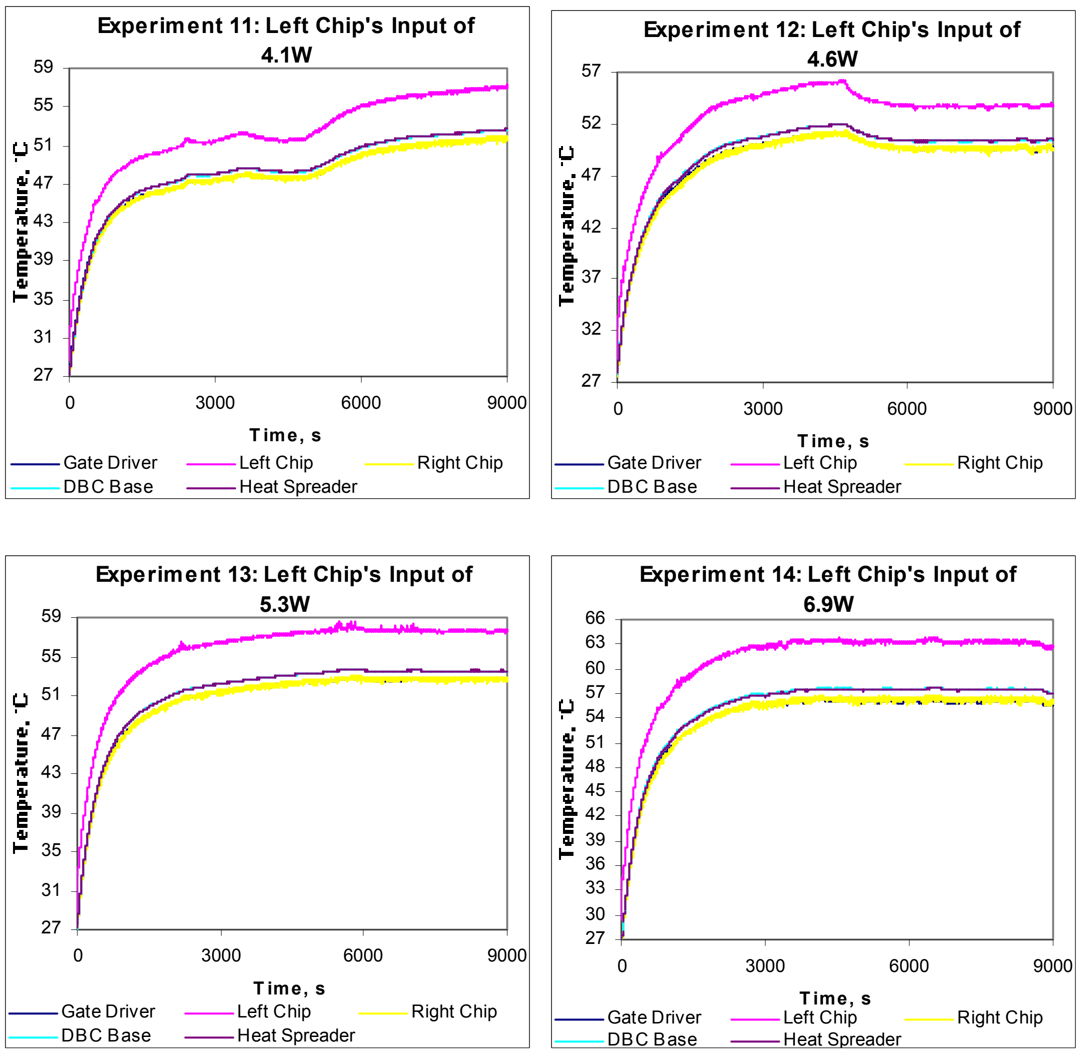


Figure 4.13. All experimental temperature profiles measured from five thermocouples in the IPEM.

4.3 Experimental System for Hight-Speed Thermal Response

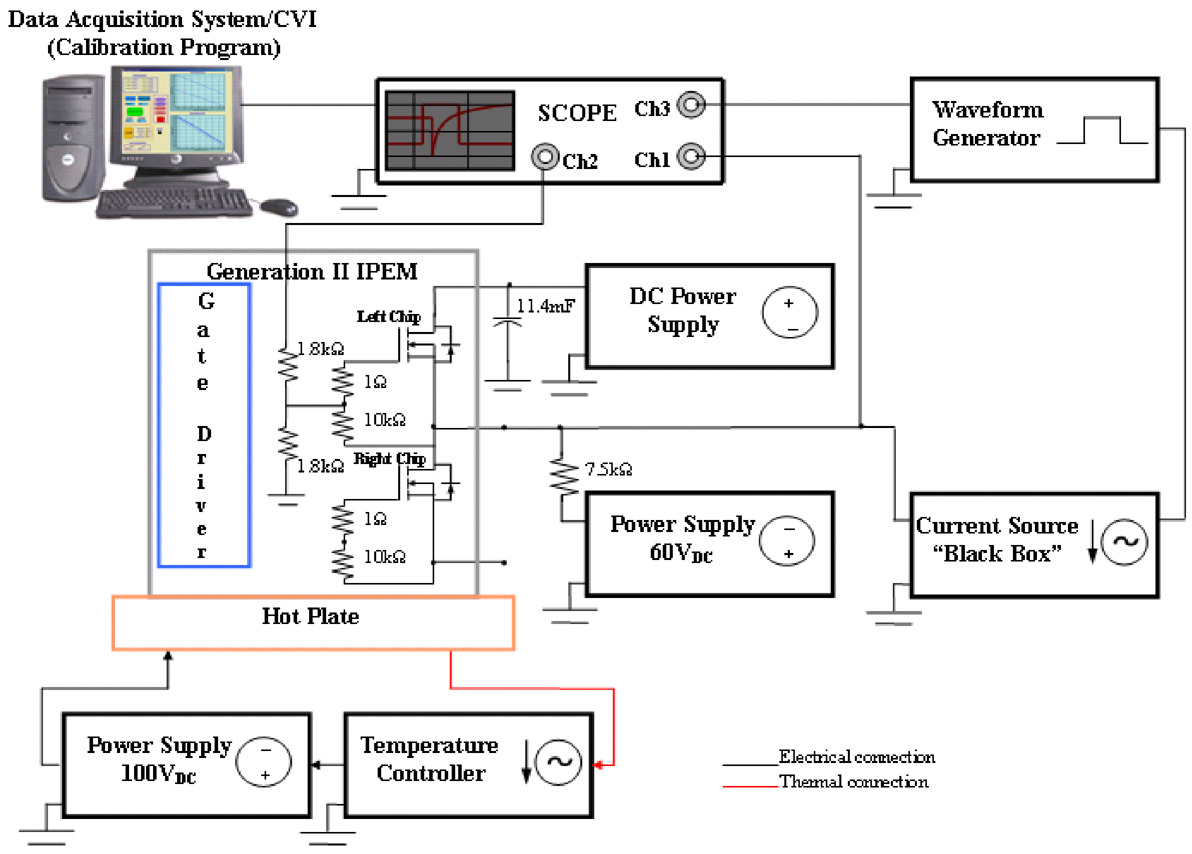
In the previous experiment the process is essentially thermal, because the temperature incidence on the electrical parameters is very insignificant, as it is observed in Figures 4.4a and 4.5a, basically the power stays approximately constant throughout the measurements. Then, in order to obtain the electrothermal interaction inside the package it is necessary to suggest and implement a different experimental system type in which the electrothermal interface can be observed and measured. Under this experiment, the device thermal response will be measured due to a continuously applied electrical stimulus. This way, accurate measurements of chip heating under high-power transient conditions can be observed. However, at the experimental level it is very difficult to acquire this parameter, since it is impossible to place a thermocouple in the power device junction (point between the device and the copper metallization layer) due to the component dimensions, or the module geometry configuration. To measure the chip operation temperature there are few diverse methods: infrared microradiometry, the use of liquid crystal, thermographic phosphors, among others. Nonetheless, all these methods require the semiconductor chip to be exposed to chemical substance at its surface. A simple method (that does not require the use of those substances) utilizes a temperature-sensitive electrical parameter (TSEP) of the device as a thermometer [26, 27], given a single average temperature value for the chip, which is very appropriate in this case.

This method consists of choosing a TSEP to obtain the device junction temperature. Using the TSEP, the electrical measurement of the temperature is done in two stages: the calibration phase and the measurement phase. In this case, the emitter-base voltage of the MOSFET (IPEM's power device) has been chosen as the TSEP, and then the emitter-base voltage value is established at various MOSFET temperatures. This is accomplished by an externally device heating (such as a temperature-controlled hotplate) and measuring the emitter-base voltage with a constant current passing through the chip. The constant current must be very small as it is assumed that the device is at the same temperature as the hotplate which requires that no power be dissipated in the device, or at least minimum power [27]. This is the calibration phase. A curve is obtained from this phase, the MOSFET's emitter-base voltage against its device operation temperature. But the required curve is the thermal transient, which is obtained in the next phase, the measurement phase.

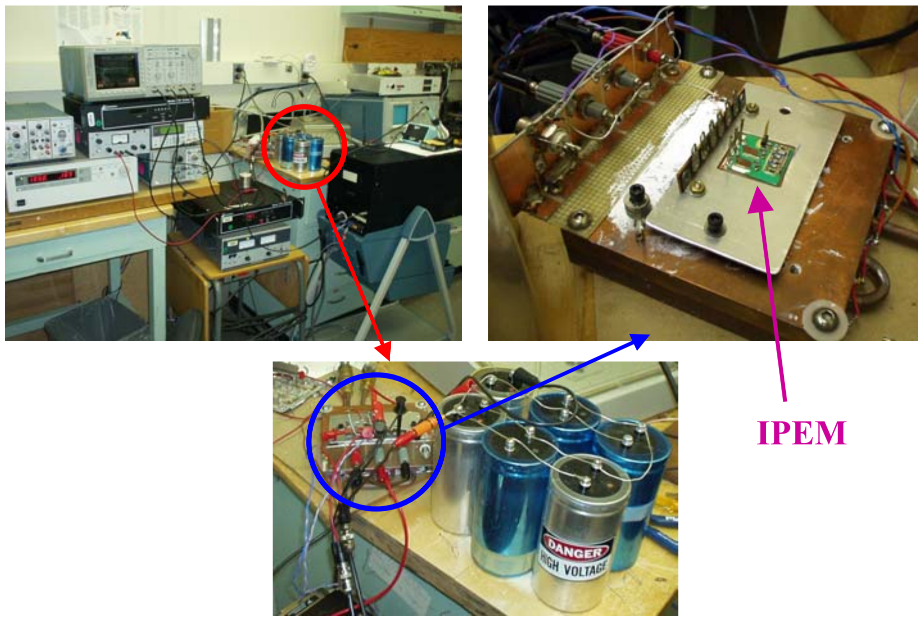
In the measurement phase, the chip is self-heated by dissipating power at a much larger power than that at which the calibration was done. In order to measure the temperature, the device must be switched from the heating condition (high power) to the measurement condition (the same low current and power at which the calibration was done). Both phases of the high-speed thermal response were developed thanks to the collaboration of Dr. Allen Hefner at the National Institute of Standard and Technology (NIST).

4.3.1 Calibration Phase

The experimental system is composed of a data acquisition system (from LabWindows/CVI), an oscilloscope, a waveform generator, a current generator in the state-of-the-art ("Black Box") designed at NIST, a temperature controller, power supplies, and multimeters, as is presented in Figure 4.14.



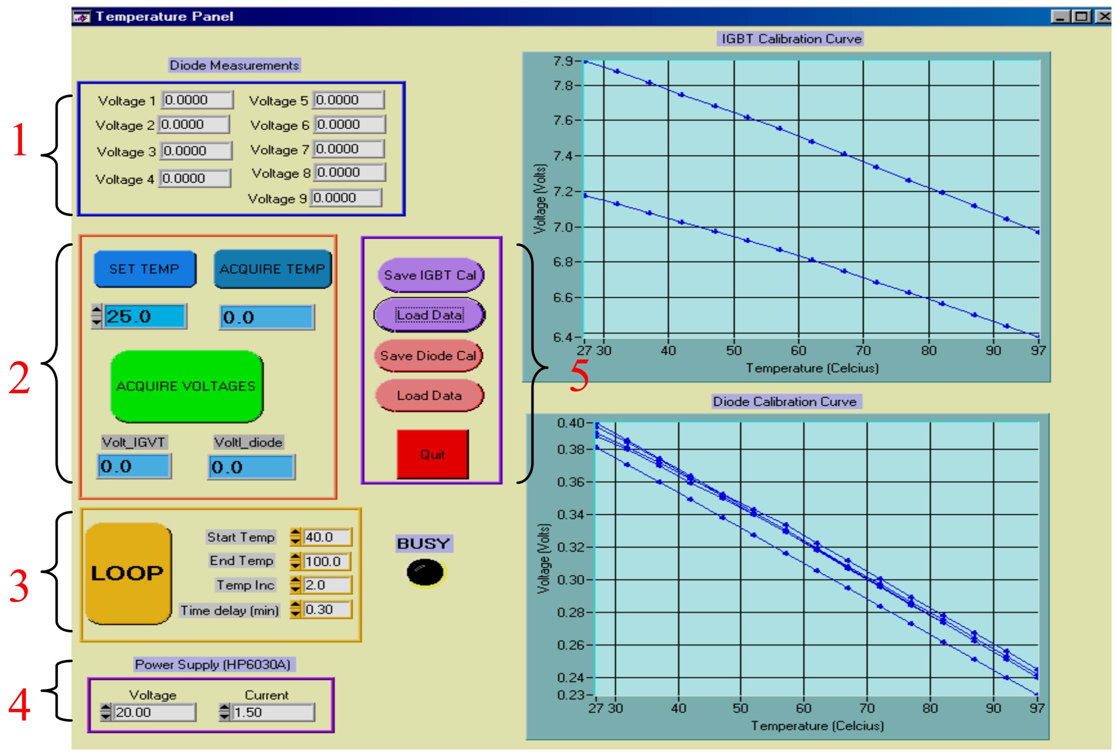
(a)



(b)

Figure 4.14. Experimental testbed for the calibration phase of the high-speed thermal response developed at NIST: (a) schematic of experiment; (b) picture of actual set-up.

In essence, the calibration experiment operates in the following manner: the first step is to set the input parameters (as frequency, wave amplitude, duty cycle, etc) at the waveform generator through the data acquisition system (calibration program), as well as other parameters in the remaining instruments. The previously defined wave is used for the black box as trigger to generate the same current wave, but the current magnitude must be manually fixed in each experimental sample. When the parameters in these equipments are ready, a heat exchanger (placed under the hotplate) extracts the existing heat in the IPEM until the chip operation temperature reaches the minimum temperature established from the calibration program to the temperature controller. In other cases it was necessary to warm up (using the DC-power supply of 100V) the hotplate to obtain the required temperature. Now, with the temperature range fixed in the program and a specific temperature interval (also defined in the program), from the minimum temperature the voltage measurements are made (by means of a signal sent to the oscilloscope, which takes the voltage values) for each temperature defined (when the temperature controller read the temperature corresponding to the fixed value, through a thermocouple placed in the IPEM, in the program automatically this temperature is stored and voltage is read and stored). This process is repeated until the device operating temperature reaches its maximum value fixed in the program (less than the boiling point of water inside the heat exchanger), see Figure 4.14a. Similarly to the “transient ipem program”, here the calibration curve values (voltage against temperature) are stored, displayed and plotted in a computer window. These data can be extracted to analyze it in an excel file.



Voltage 1 0.0000
Voltage 2 0.0000
Voltage 3 0.0000
Voltage 4 0.0000
Voltage 5 0.0000
Voltage 6 0.0000
Voltage 7 0.0000
Voltage 8 0.0000
Voltage 9 0.0000

25.0
0.0

Volt_IGVT 0.0
Voltl_diode 0.0

Start Temp 40.0
End Temp 100.0
Temp Inc 2.0
Time delay (min) 0.30

Voltage 20.00
Current 1.50

Figure 4.15. Calibration data acquisition system program created by Parrilla *et al.* [26].

4.3.1.1 Data Acquisition System (Calibration Program)

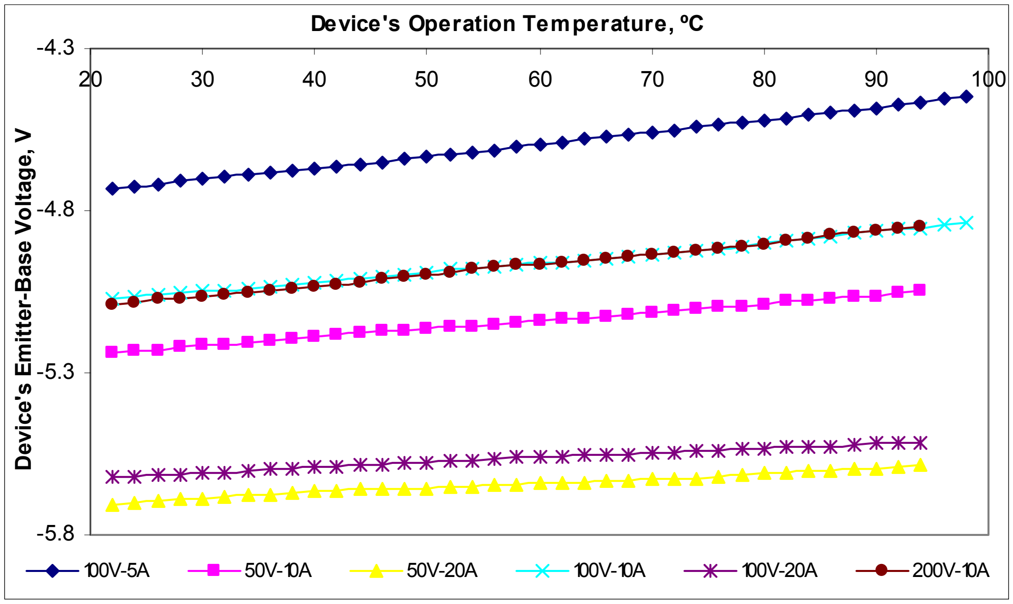
This data acquisition system program was developed and presented by Parrilla *et al.* in [26]. This program is used to obtain the voltage at the emitter-base of the IPEM's MOSFETs at different device operation temperatures. The changes in temperature are done through a temperature controller and a $100V_{DC}$ power supply and the module is place in a close heater where is expected that the whole module will reach the same temperature. Then the voltages are measured and saved. The calibration program panel is shown in Figure 4.15. According to [26] and the Figure 4.15, the program description is:

1) This blue box contains the display for the voltages measured.

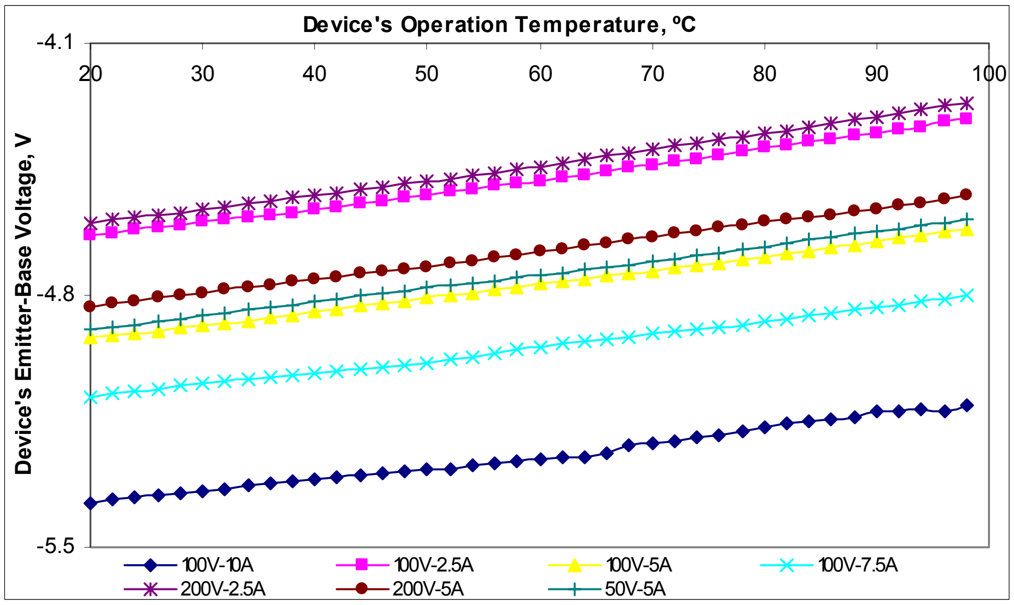
2) What is included in this box are buttons to set the temperature of the temperature controller (*set temp*) and to obtain the temperature measured by the temperature controller (*acquire temp*). The *Acquire Voltages* button obtains the module to a certain temperature and then measure voltages and display them.
3) The text boxes are for the user to specify the first and last temperature at which voltages will be measured; this is used to set the temperature controller. The increment is the difference between the temperatures after the initial temperature. By pressing the Loop button starts a cycle where the temperature is varied and voltage is measured until the final temperature is reached. Basically what it does is that wait until the next temperature is reached and stays there the time indicated by the user. This is a way of waiting for the whole module to be at the same temperature. Finally, the voltages are measured and displayed. This is done for every temperature reached.
4) Here the user set the voltages and the current that will be supplied by the DC Power Supply.
5) In this box are the buttons for saving and loading the data and also for exiting the program.

4.3.1.2 Results

The calibration curves must have a linear behavior, since in the experiment the device voltage was increased proportional to a device operation temperature increase, as it is indicated in [26]. Many calibration experimental samples were made, but some measurements were neglected, because the resulting calibration curves had too many fluctuating points and therefore such curves were unreliable. In some cases, some curves had a shifting with respect to an existing curve at the same initial conditions of electrical parameters and temperature (maybe because the instruments were not in a stable operation point, or because connections between points and instruments were broken, or because the controller could not take the reading of temperature well, etc), then those curves were ignored.



(a)



(b)

Figure 4.16. Calibration curves for: (a) left power device; (b) right power device.

For the left and right power devices, the number of obtained acceptable calibration curves was of six and seven, respectively. The calibration curves for the devices are presented in Figure 4.16. Here, as in [26], all curves tend to be parallel, with approximately equal slopes. Both figures ("a" and "b") shown that for high input power conditions the curves have better linear behavior, whereas for low inputs the curves are more fluctuating.

4.3.2 Transient Measurement Phase

In this stage, the same instruments employed in the calibration phase with the exception of the $100V_{DC}$ Power Supply and the Temperature Controller are used, as it is observed in Figure 4.17.

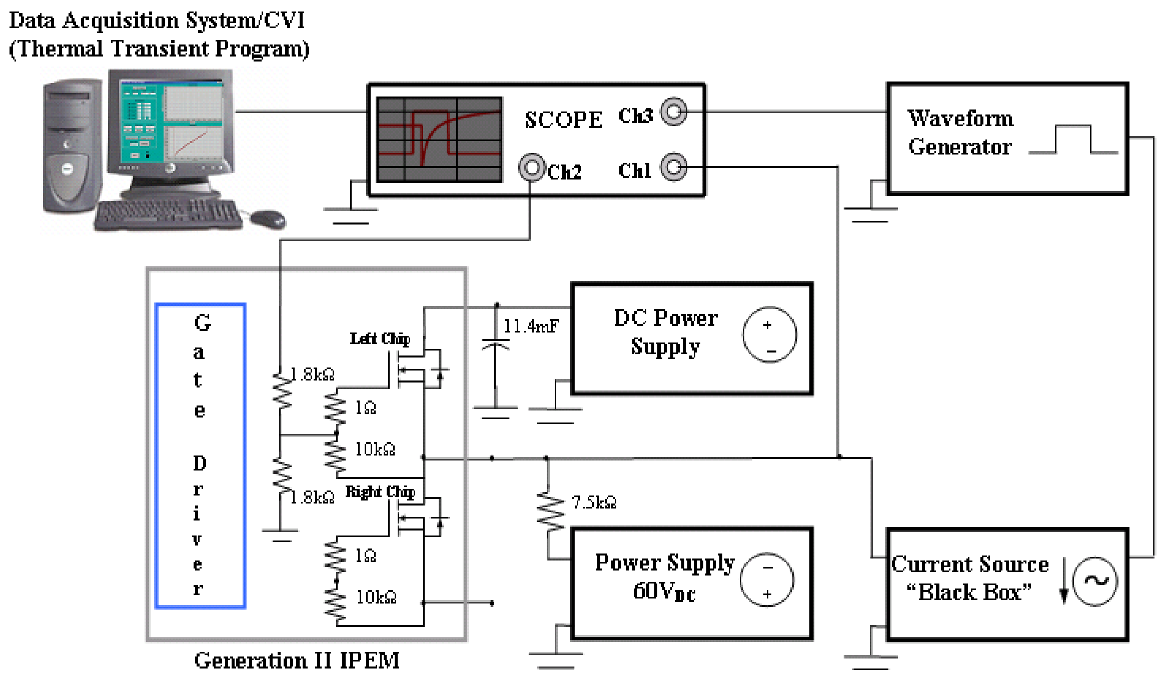


Figure 4.17. Schematic of the experimental testbed for the thermal measurement phase of the high-speed thermal response developed at NIST.

With a constant temperature of the hotplate (25ºC) and knowing the temperature and voltage ranges obtained from the calibration phase for a determined initial condition of voltage and current, it was possible to establish the required power pulse. For example if a calibration measurement was made at 100V and 5A, then it is necessary to define the voltage range, as it is shown in Figure 4.18 (in the figure, that curve corresponds to the calibration curve for 100V and 5A), for that the voltage tip in the oscilloscope do not exceed the fixed range, avoiding exceed the maximum operating temperature for the device.

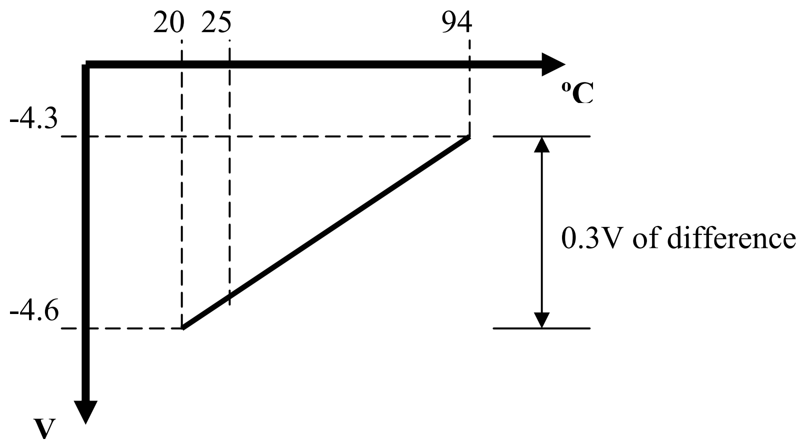
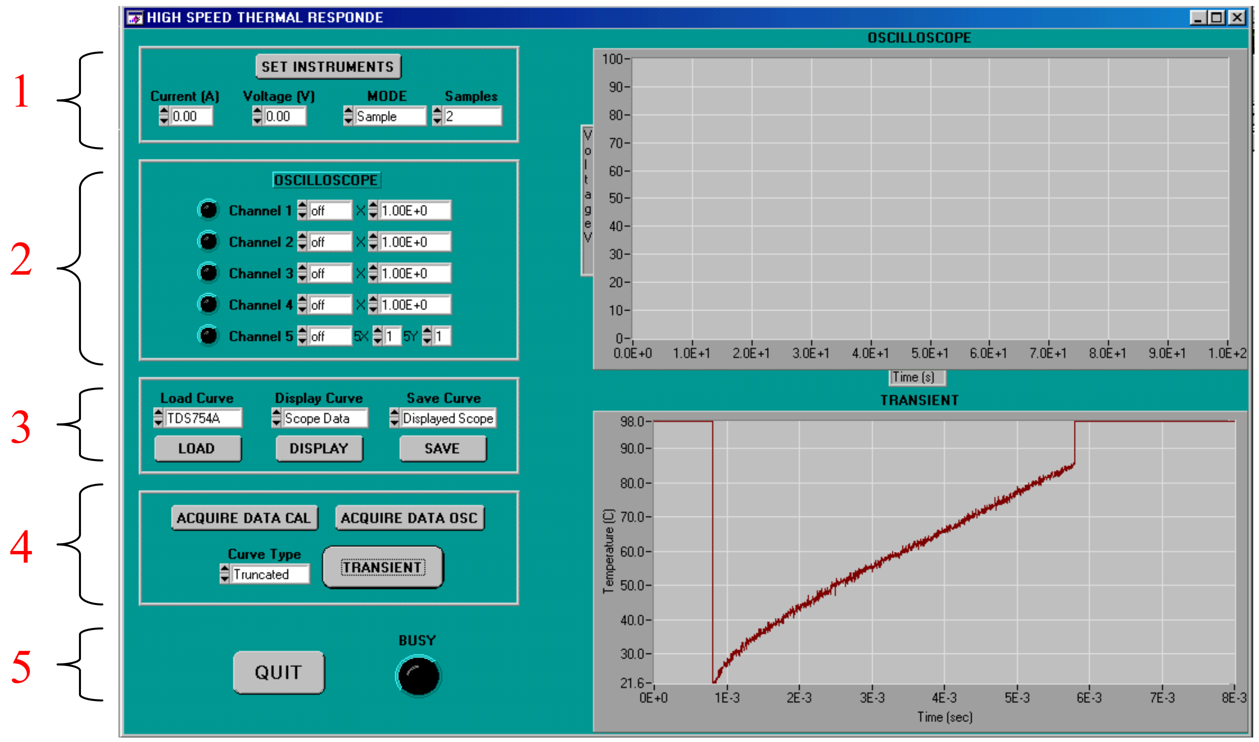


Figure 4.18. Example of the how to obtain the power pulse in the thermal measurement phase.

Excluding the oscilloscope and the DC power supply parameters, all initial parameters of the instruments were set manually. Therefore, with 0V and 0A in the sources, a pulse in the pulse generator is fixed, posterior it is fixed the current in the specified values (in this case 5A) and then, considering the wave given by the oscilloscope (making sure that this one did not exceed the fixed voltage range), the voltage is increased in small increments in the source from 0V until an approximated maximum value (for the example this value is approximately 100V). The objective was to obtain a voltage transient curve near 300mV of wide but never reaching this value, since when exceeding the 300mV the device could be burned. If the pulse is not enough to obtain the maximum operation temperature given by the calibration curve, then it is necessary to redefine the pulse (in others words, the frequency) and again set the current and the power in the sources until an appropriate frequency is found that provides the maximum operation temperature value of the device given by the calibration curve. For the mentioned example, the strategy was to lower the frequency so that the time interval in which the pulse was turned on was greater, with the intention of allowing more chip heating in that time interval.

When the optimal wave was obtained for the voltage and temperature ranges fixed in the calibration, then the oscilloscope data (voltage against time) is acquired from the program developed in the CVI platform (thermal transient program). With these data, using the thermal transient program, the transition between voltage and temperature is done using the calibration curves. It is to say, each voltage value is introduced in the calibration curve to interpolate it or to extrapolate it, according to the case. The full C code is presented in Appendix E.



Current (A) 0.00
Voltage (V) 0.00
MODE Sample
Samples 2

Channel 1 off × 1.00E+0
Channel 2 off × 1.00E+0
Channel 3 off × 1.00E+0
Channel 4 off × 1.00E+0
Channel 5 off 5X 1 5Y 1

Load Curve TDS754A
Display Curve Scope Data
Save Curve Displayed Scope

Curve Type Truncated

Figure 4.19. Thermal transient program created for the high-speed thermal response in the transient measurement phase.

4.3.2.1 Data Acquisition System (Thermal Transient Program)

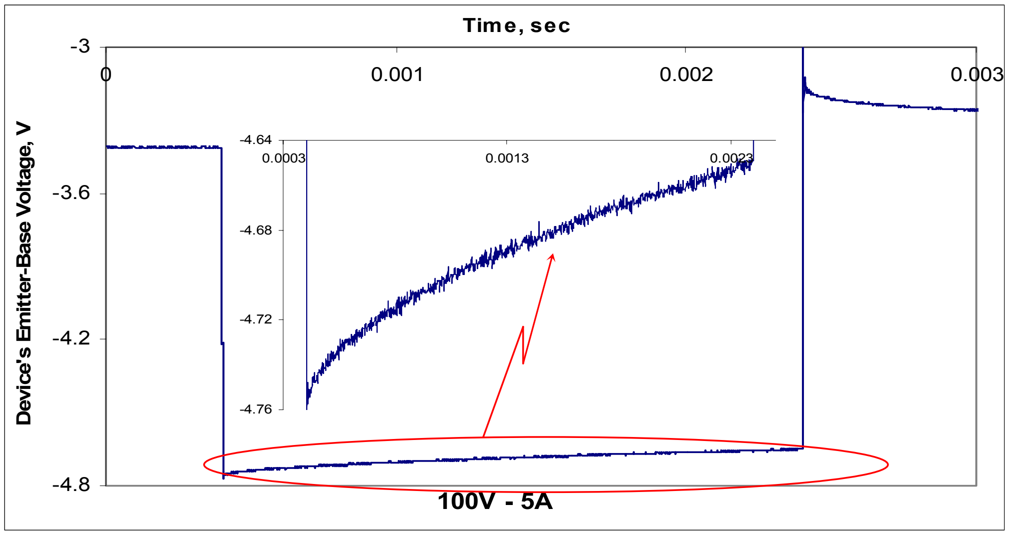
The data acquisition system used here is named the thermal transient program and is presented in Figure 4.19 [28]. The structure of the thermal transient program is as follows:

1) This box set the voltage and current in the DC power supply, and determine the acquisition mode (as well as the sample numbers) of the signal that will be read by the oscilloscope.
2) In this section the oscilloscope channels are turned on in order to establish communication between the instrument and the computer. For each channel, the scale of the data acquired from the equipment is defined here.
3) This part is used to load, display and save three types of data: current data from oscilloscope, data file from oscilloscope, thermal file.
4) Here, the transition between the voltage data and the temperature data is made. First it is necessary to obtain the desired calibration curve (using the “ACQUIRE DATA CAL” button) to compare it with the measurement made in the oscilloscope (which is accessed using the “ACQUIRE DATA OSC” button). With the “TRANSIENT” button the change from voltage to temperature is made, introducing each voltage value to the loaded calibration curve for interpolation or extrapolation, and this way to acquire the electrothermal transient.
5) This GUI section is used to exit off the program.

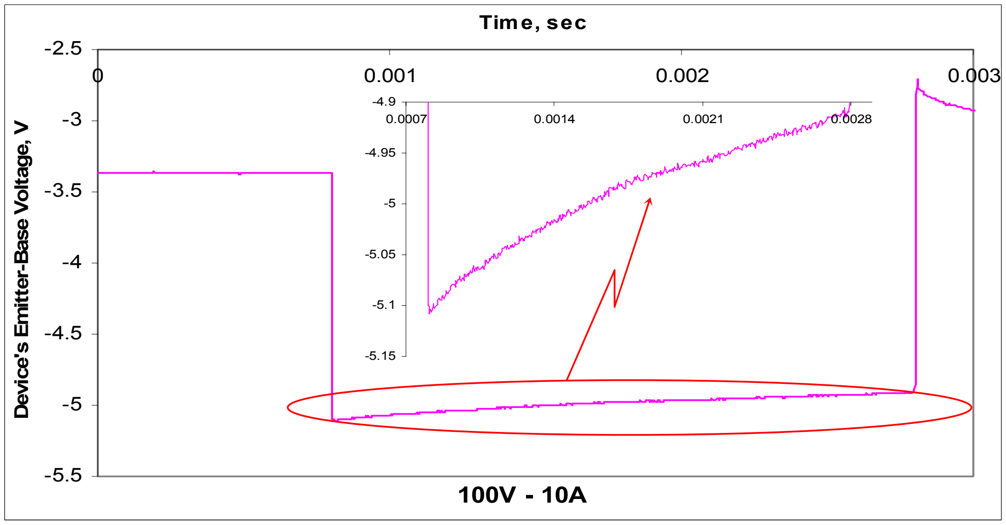
4.3.2.2 Results

From the measurement, phase two type of curves are obtained, one electrical and one thermal. In the electrical curves, the MOSFET's transient emitter-base voltages are acquired from several electrical conditions. In the thermal case, the curves are acquired making the interface between the calibration information and the electrical measurements through the "thermal transient program". In Figures 4.20 and 4.21 all voltage measurements for the left and right power devices (MOSFETs) are given. Each plot includes a transient voltage zoom (expanded from the oval) which demonstrates the temperature incidence on the electrical parameters (in this case the MOSFET's emitter-base voltage).

(a)



(b)



(c)

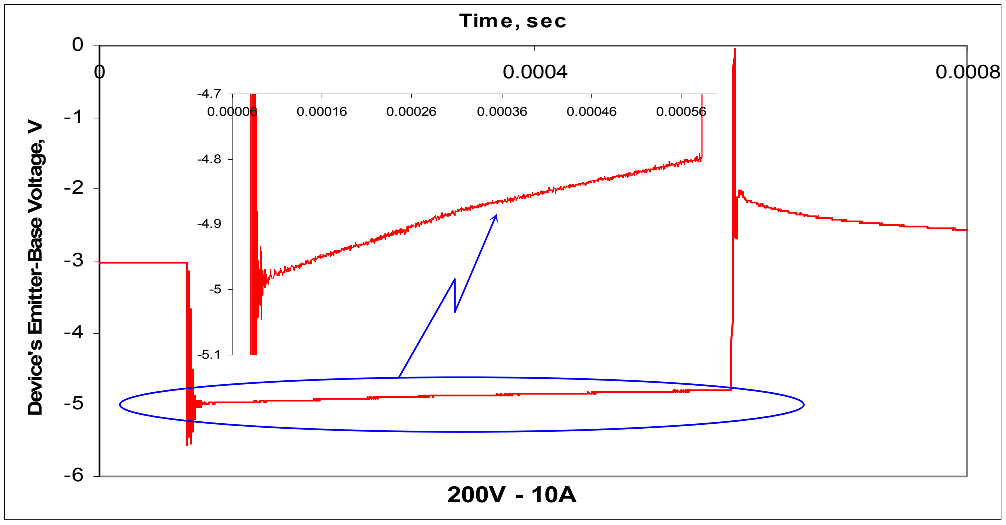
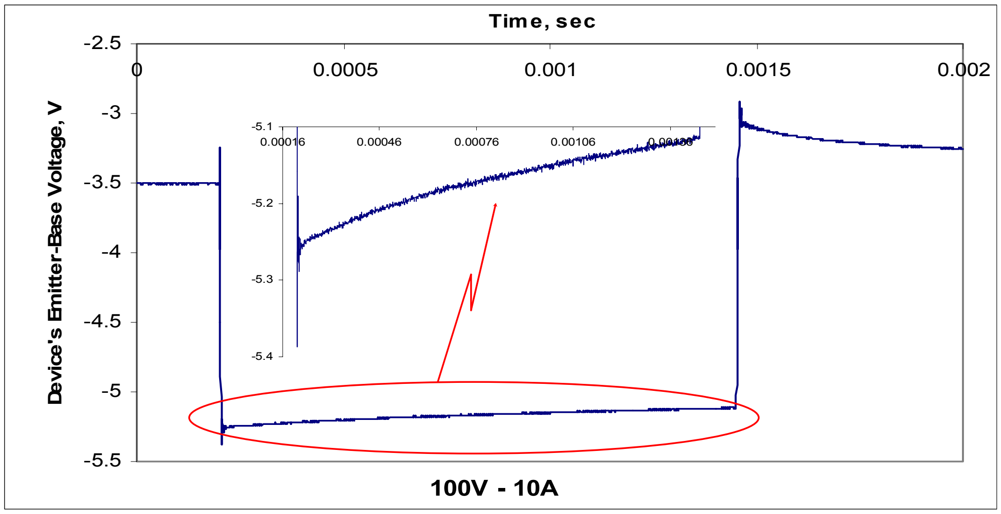
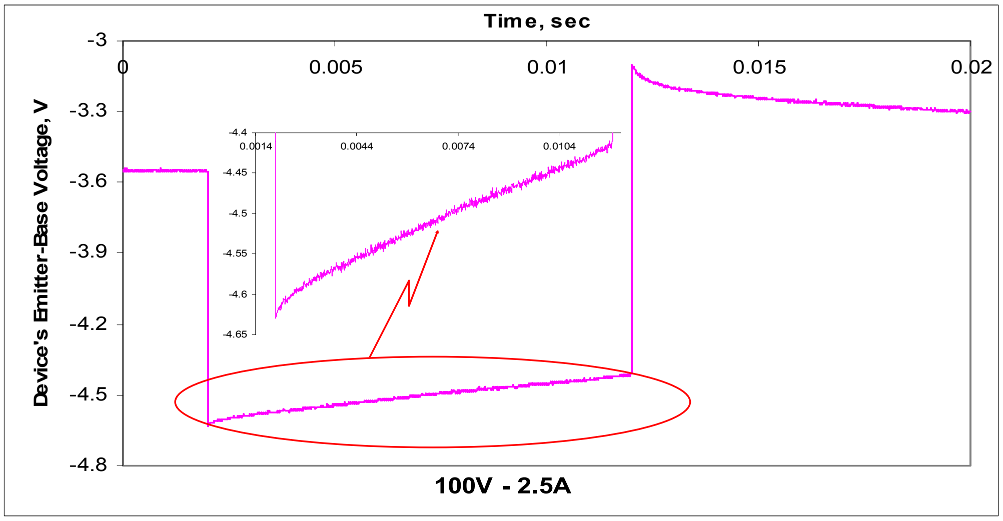


Figure 4.20. Left MOSFET's transient emitter-base voltages with input electrical parameters of: (a) 100V-5A; (b) 100V-10A; (c) 200V-10A.

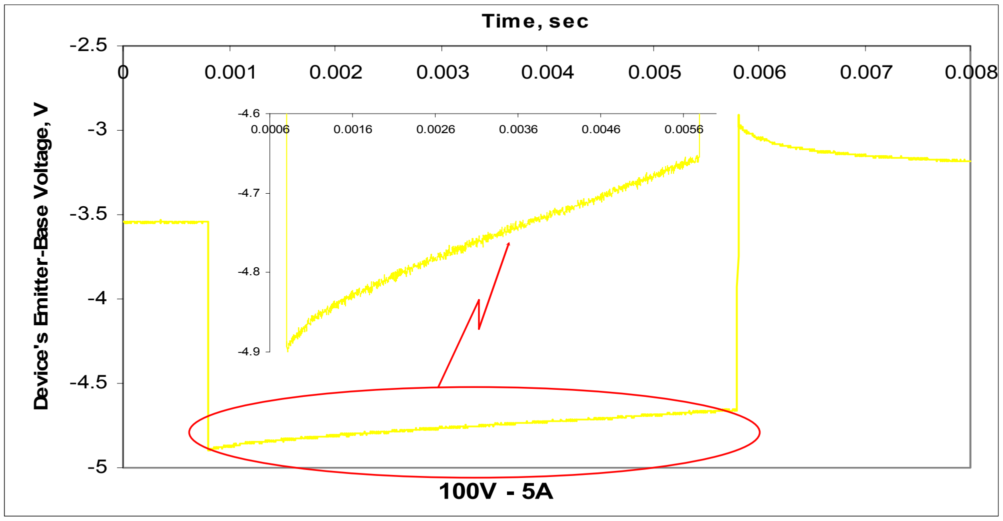
(a)


Time, sec
Device's Emitter-Base Voltage, V
100V - 10A
-2.5
-3
-3.5
-4
-4.5
-5
-5.5
0
0.0005
0.001
0.0015
0.002
-5.1
-5.2
-5.3
-5.4
0.00016
0.00046
0.00076
0.00106
0.00136

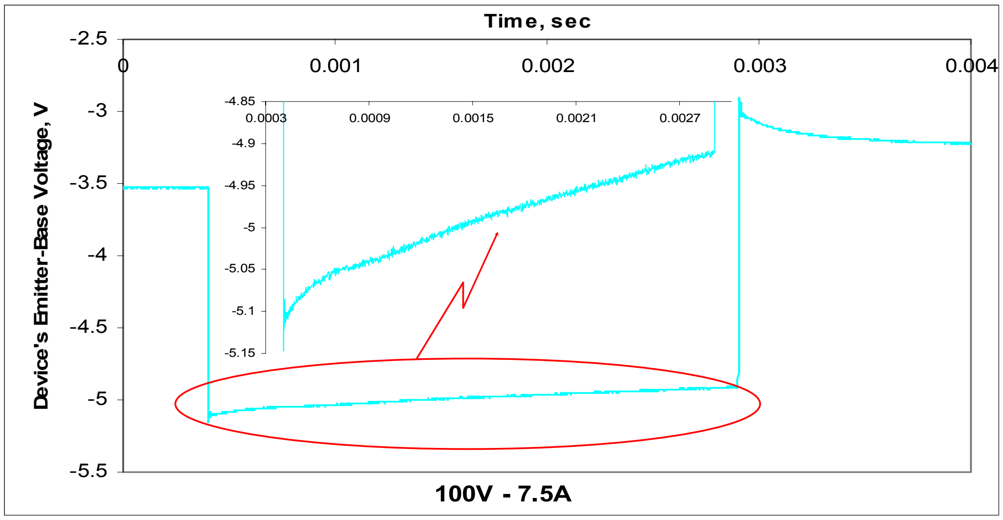
(b)


Time, sec
Device's Emitter-Base Voltage, V
100V - 2.5A
-3
-3.3
-3.6
-3.9
-4.2
-4.5
-4.8
0
0.005
0.01
0.015
0.02
-4.4
-4.45
-4.5
-4.55
-4.6
-4.65
0.0014
0.0044
0.0074
0.0104

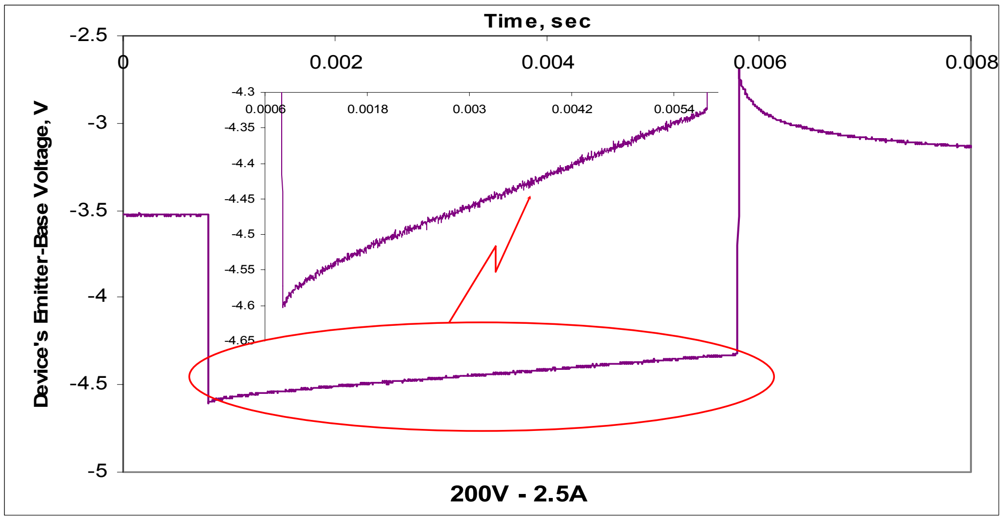
(c)


Time, sec
Device's Emitter-Base Voltage, V
100V - 5A
-2.5
-3
-3.5
-4
-4.5
-5
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
-4.6
-4.7
-4.8
-4.9
0.0006
0.0016
0.0026
0.0036
0.0046
0.0056

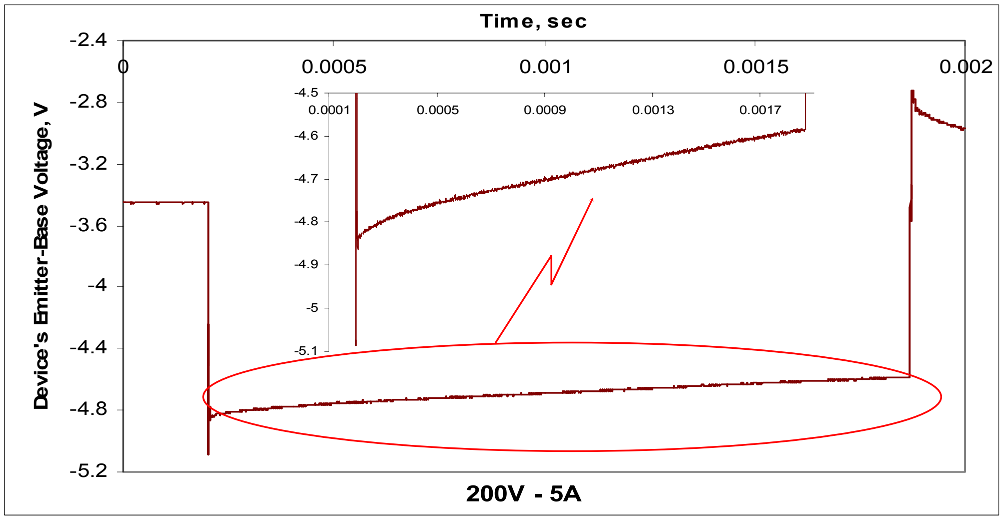
(d)


Time, sec
Device's Emitter-Base Voltage, V
100V - 7.5A

(e)


Time, sec
Device's Emitter-Base Voltage, V
200V - 2.5A

(f)


Time, sec
Device's Emitter-Base Voltage, V
200V - 5A

(g)

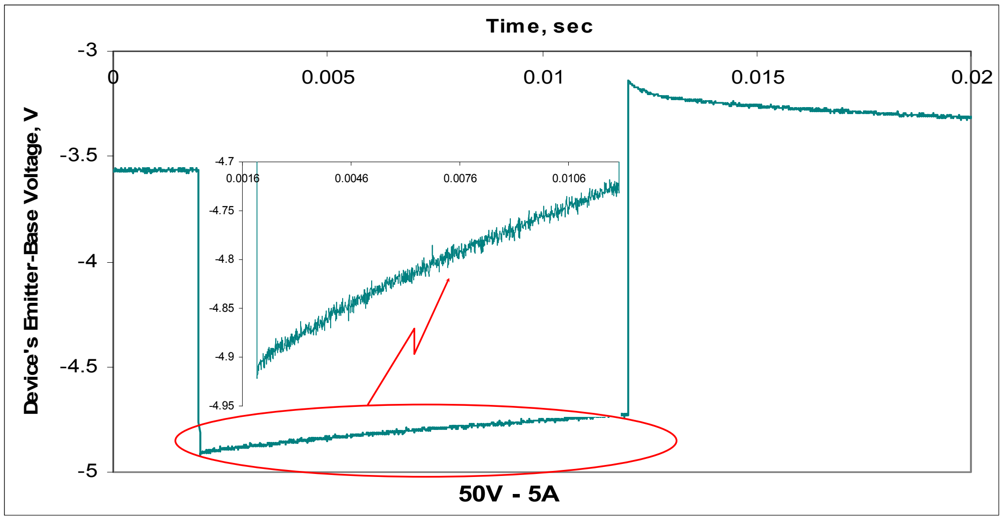
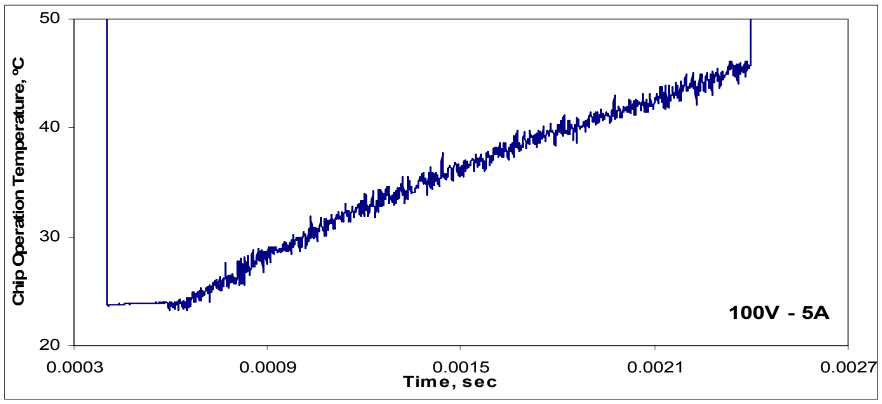
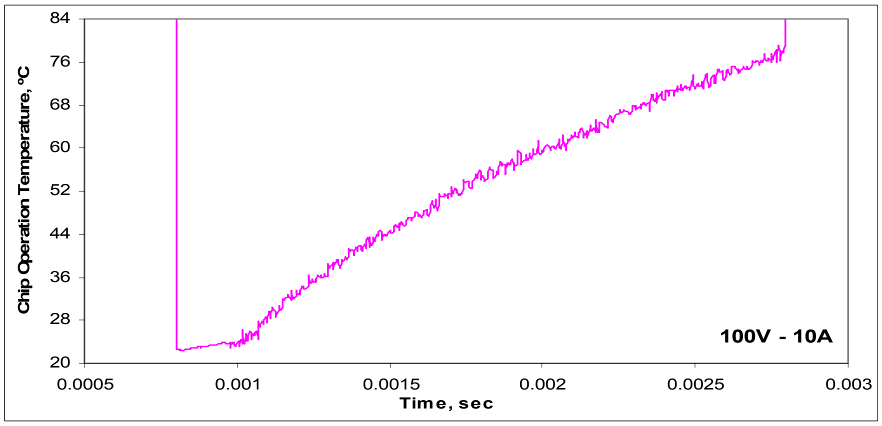


Figure 4.21. Right MOSFET's transient emitter-base voltages with input electrical parameters of: (a) 100V-10A; (b) 100V-2.5A; (c) 100V-5A; (d) 100V-7.5A; (e) 200V-2.5A; (f) 200V-5A; (g) 50V-5A.

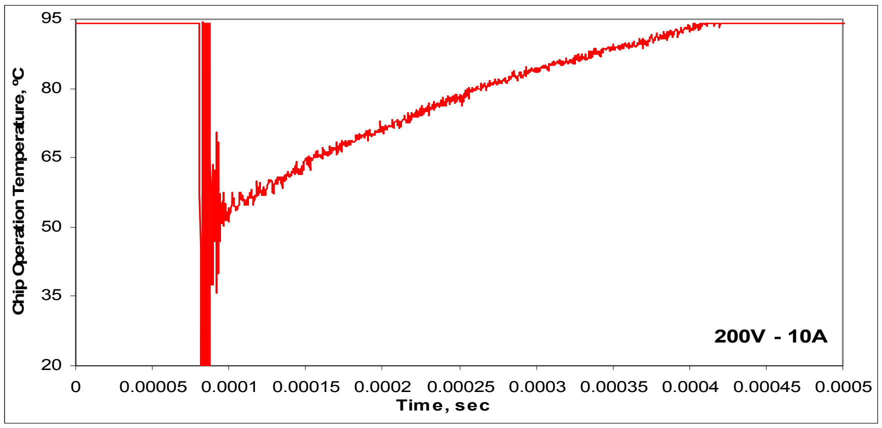
Figures 4.22 and 4.23 show all transient temperature curves obtained from the calibration and voltage data (Figures 4.16, 4.20 and 4.21) for left and right chip, respectively. The number of curves of voltage and temperature for the left chip is less than for the right chip because during the experiment the left chip was burned and only three curves of each type were obtained. The next step is to compare these data with the electrothermal model output. This task was developed using a commercial circuit simulator and the details are shown in the next chapter.



(a)

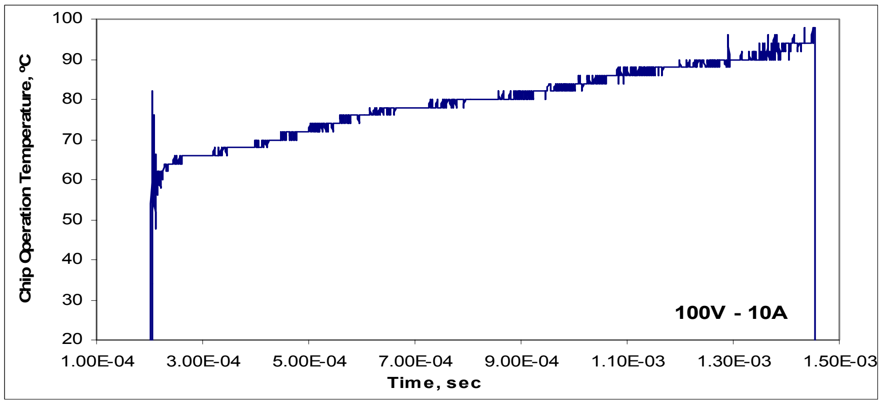


(b)

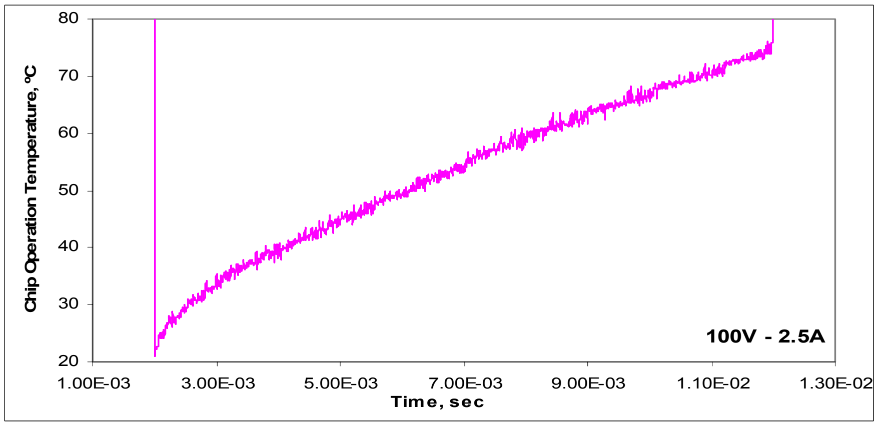


(c)

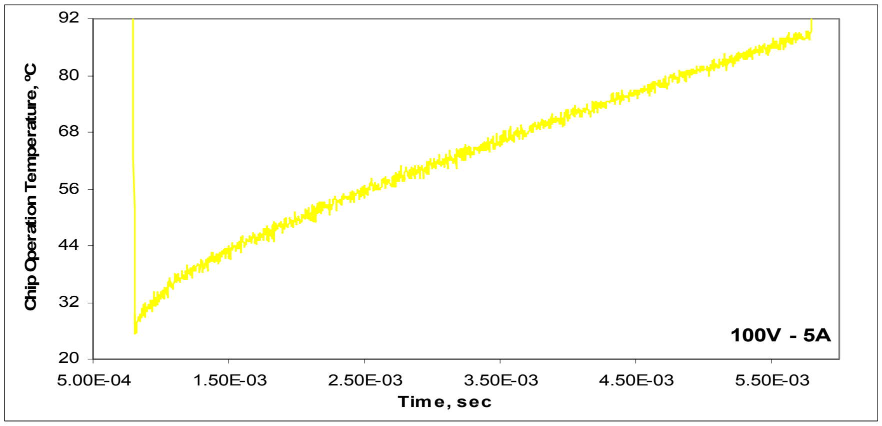
Figure 4.22. Left device's transient operation temperature with input electrical parameters of: (a) 100V-5A; (b) 100V-10A; (c) 200V-10A.


Chip Operation Temperature, ºC
Time, sec
100V - 10A
100
90
80
70
60
50
40
30
20
1.00E-04
3.00E-04
5.00E-04
7.00E-04
9.00E-04
1.10E-03
1.30E-03
1.50E-03

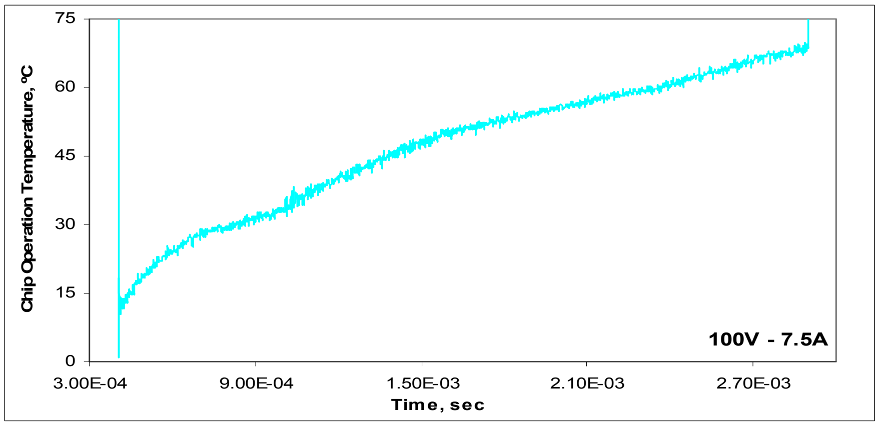
(a)


Chip Operation Temperature, ºC
Time, sec
100V - 2.5A
80
70
60
50
40
30
20
1.00E-03
3.00E-03
5.00E-03
7.00E-03
9.00E-03
1.10E-02
1.30E-02

(b)

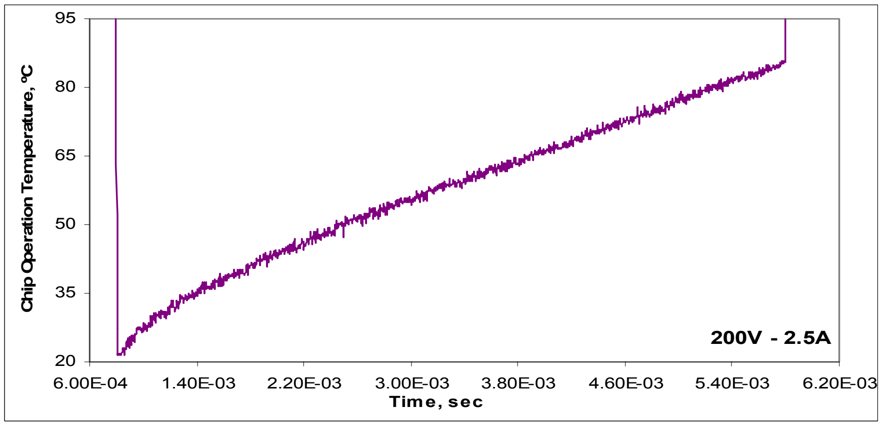

Chip Operation Temperature, ºC
Time, sec
100V - 5A
92
80
68
56
44
32
20
5.00E-04
1.50E-03
2.50E-03
3.50E-03
4.50E-03
5.50E-03

(c)



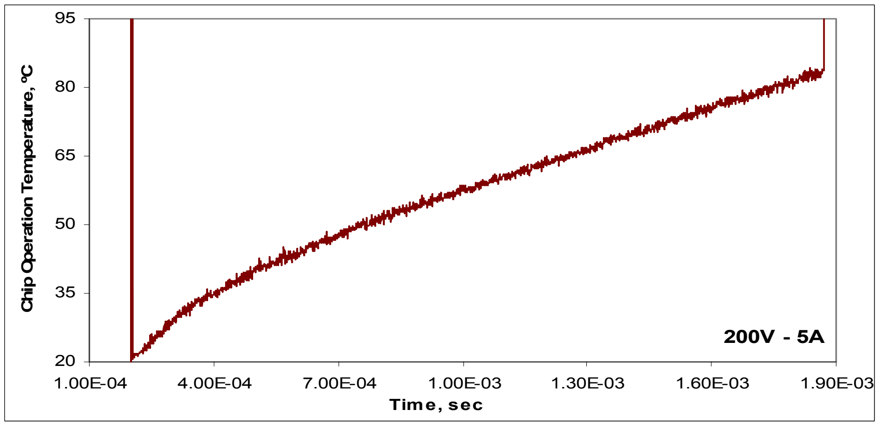
Chip Operation Temperature, ºC
75
60
45
30
15
0
3.00E-04
9.00E-04
1.50E-03
2.10E-03
2.70E-03
Time, sec
100V - 7.5A

(d)



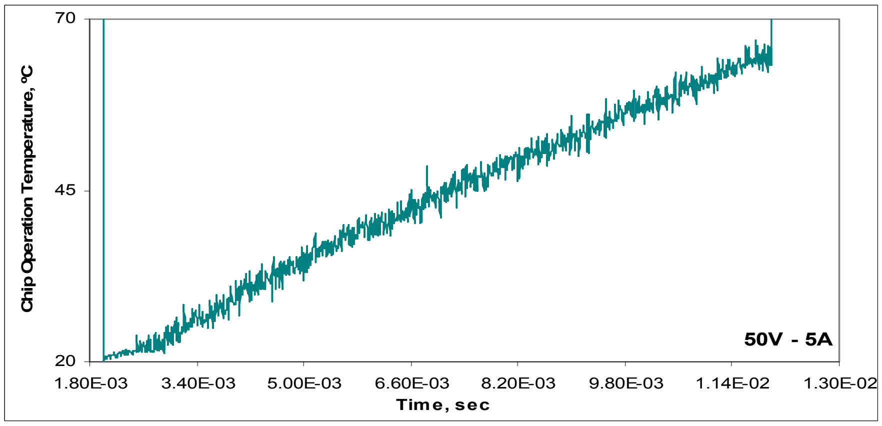
Chip Operation Temperature, ºC
95
80
65
50
35
20
6.00E-04
1.40E-03
2.20E-03
3.00E-03
3.80E-03
4.60E-03
5.40E-03
6.20E-03
Time, sec
200V - 2.5A

(e)



Chip Operation Temperature, ºC
95
80
65
50
35
20
1.00E-04
4.00E-04
7.00E-04
1.00E-03
1.30E-03
1.60E-03
1.90E-03
Time, sec
200V - 5A

(f)



(g)

Figure 4.23. Right device's transient operation temperature with input electrical parameters of: (a) 100V-10A; (b) 100V-2.5A; (c) 100V-5A; (d) 100V-7.5A; (e) 200V-2.5A; (f) 200V-5A; (g) 50V-5A.

CHAPTER 5

ELECTROTHERMAL MODEL IMPLEMENTED IN SABER^{TM} SIMULATOR

5.1 Introduction

In order to acquire the electrothermal performance of the Generation II IPEM, the LTCM model was implemented in SABER^{TM}, a circuit simulator capable of coupling electrical and thermal signals, between others. Its versatility and ease of model implementation makes it an attractive platform for the simulation of the electrothermal interactions inside the electronic package. It also eliminates the process of having to exchange data between different simulations, because both electrical and thermal models run in the same simulator.

SABER^{TM} simulator is a mathematical engine that solves the network of equations represented by models and their interconnections in a circuit or system. Simulator access is via a highly-interactive and easy-to-use graphical user interface for analyzing designs, operating the simulator, and obtaining and viewing results. This simulator is designed to perform simulations based on very few preconceptions about the target system. Consequently, the simulator can analyze designs containing multiple technologies, using the analysis units native to these technologies:

1) Electronic.
2) Power electronics.

3) Thermal management.
4) Electro-mechanical.
5) Mechanical.
6) Electro-optical.
7) Optical.
8) Hydraulic.
9) Control systems.
10) Sample-data systems.

The simulator has an extensive list of models for different types of component included in its parts library. All SABER's ^{TM} models were written in MAST language, the unique programming language reserved for this simulator. MAST is a mixed-signal (analog-digital) hardware description language (HDL) that allows the description (or modeling) of hardware that performs a continuous-time function, for example electrical sources, resistors, transistors, capacitors, hydraulic and digital systems, among others.

When a device is not contained in the simulator's parts library and if the device behavior can be expressed in first order mathematical terms, SABER^{TM} simulator can model and simulate it (up to the system level) with an accurate representation of interactivity between the technologies. Given this capability, models can be created directly using the actual equations and relationships that govern the behavior of devices, not only electrical macromodel equivalents. With SABER^{TM} simulator, any mix of technologies can be simulated, and all simulation results will be presented with their corresponding units. The new model (of analog system or element) also must be built in MAST language, which can be defined in terms of nonlinear "lumped" algebraic or differential equations that eventually will be solved by the simulator using numerical method (the simulator contain two numerical solution methods: Newton-Raphson and Katzenelson, in this case the most appropriate to solve our set of equations is Newton-Raphson). Each component or system model (new or existing) must be stored in a text file named template (which have a specific internal structure), in which is included the model code in MAST language, in order that the simulator can recognize it. In addition, each component or system's template must be associated to a symbol (new or existing) in the SABER's ^{TM} graphical user interface (Sketch tool).

5.2 LTCM Model Template

The initial task is to convert the thermal model at hand into a MAST template. The FORTRAN program, presented in Chapter 3, can be used because SABER^{TM} can use foreign routines as extensions of the MAST language. This means that the simulator uses the FORTRAN file as a function in a MAST template. However, this option is most complicated than building the new template based uniquely in MAST language.

In order to begin, the template file must be identified by the form *filename.sin* so that the simulator may read it. Also a symbol must be created for the template with the same connections stated in the first line of the template which is described in the following paragraph. This symbol is presented in Figure 5.1.

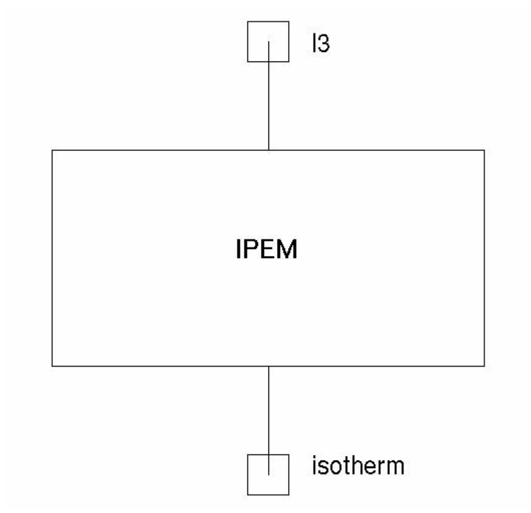
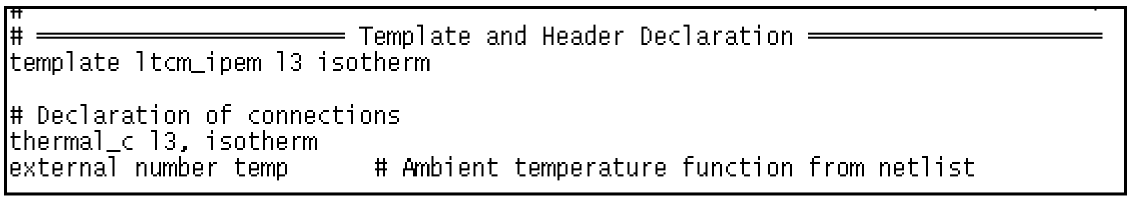


Figure 5.1. Symbol of the IPEM's LTCM model.

The script commences with the template definition or the *header* (see Figure 5.2). This is a line necessary in the script for SABER^{TM} to be able to identify the connections of the template, and to identify the template to a symbol.



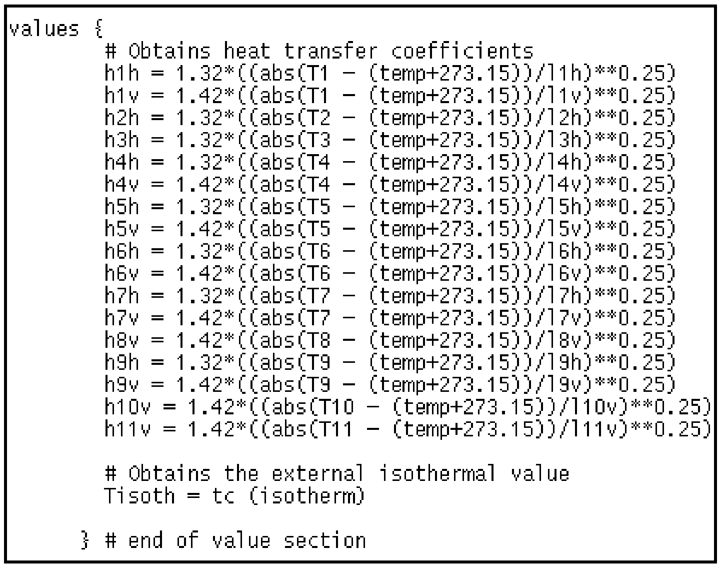
```
#
# ====================== Template and Header Declaration =====================
template ltcm_ipem l3 isotherm

# Declaration of connections
thermal_c l3, isotherm
external number temp      # Ambient temperature function from netlist
```

Figure 5.2. Template header.

The header starts with the word *template* and then the name of the template, in this case *ltcm_ipem*. Followed by this and separated by spaces are the names of the connections to the template, as they would appear named on the symbol. The IPEM's template model have two connections or terminals, the top terminal is employed by the template as input point to the model of the external power from the devices or sources, the bottom terminal is used to obtain the constant temperature from the assumed hotplate (the hotplate is assumed as a temperature constant source at 25ºC). After this first line, the template connections are defined. In this template all connections are thermal and expressed in degrees Celsius, therefore the connections are declared as *thermal_c***.** This is so because in those terminals there are power-temperature interaction with other components, this connection declaration include, watts as power unit. Declaring a thermal connection such as the above implies that the simulator will automatically declare *through* and *across* values for these connections. This will be described more in detail ahead. The template body is constituted by a parameters declaration (both numbers and variables) and three sections, each one with a specific function (the sections can be written in any order, because the simulator can identify and execute them according to its internal structure). The sections are: *values*, *control* and *equation* section.

Values Section

This section is used to transform variables into the form needed in the *equation* section, as it is shown in Figure 5.3. In this case, in the *value* section the heat transfer coefficients are calculated. In this section, also a terminal's external value is assigned to variables defined into the template, that later will be used in other sections, like in the case of the terminal named *isotherm*, which passes its value to a variable defined before (*Tisoth*).

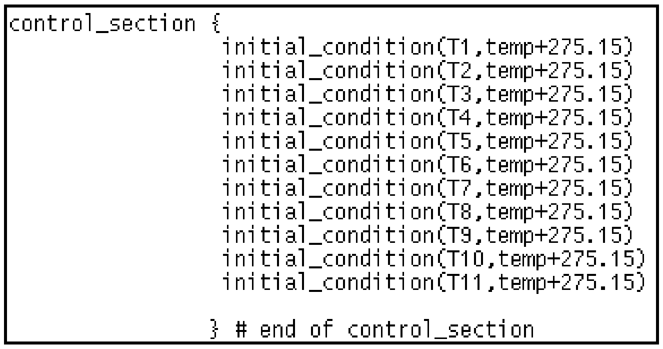


```
values {
        # Obtains heat transfer coefficients
        h1h = 1.32*((abs(T1 - (temp+273.15))/l1h)**0.25)
        h1v = 1.42*((abs(T1 - (temp+273.15))/l1v)**0.25)
        h2h = 1.32*((abs(T2 - (temp+273.15))/l2h)**0.25)
        h3h = 1.32*((abs(T3 - (temp+273.15))/l3h)**0.25)
        h4h = 1.32*((abs(T4 - (temp+273.15))/l4h)**0.25)
        h4v = 1.42*((abs(T4 - (temp+273.15))/l4v)**0.25)
        h5h = 1.32*((abs(T5 - (temp+273.15))/l5h)**0.25)
        h5v = 1.42*((abs(T5 - (temp+273.15))/l5v)**0.25)
        h6h = 1.32*((abs(T6 - (temp+273.15))/l6h)**0.25)
        h6v = 1.42*((abs(T6 - (temp+273.15))/l6v)**0.25)
        h7h = 1.32*((abs(T7 - (temp+273.15))/l7h)**0.25)
        h7v = 1.42*((abs(T7 - (temp+273.15))/l7v)**0.25)
        h8v = 1.42*((abs(T8 - (temp+273.15))/l8v)**0.25)
        h9h = 1.32*((abs(T9 - (temp+273.15))/l9h)**0.25)
        h9v = 1.42*((abs(T9 - (temp+273.15))/l9v)**0.25)
        h10v = 1.42*((abs(T10 - (temp+273.15))/l10v)**0.25)
        h11v = 1.42*((abs(T11 - (temp+273.15))/l11v)**0.25)

        # Obtains the external isothermal value
        Tisoth = tc (isotherm)

      } # end of value section
```

Figure 5.3. Template value section.

Control Section

This section declares specific information to the simulator that does not fit in other template and that will be used by the simulator to expand the created model. Here, the initial conditions are defined for the solution of the model (see Figure 5.4).

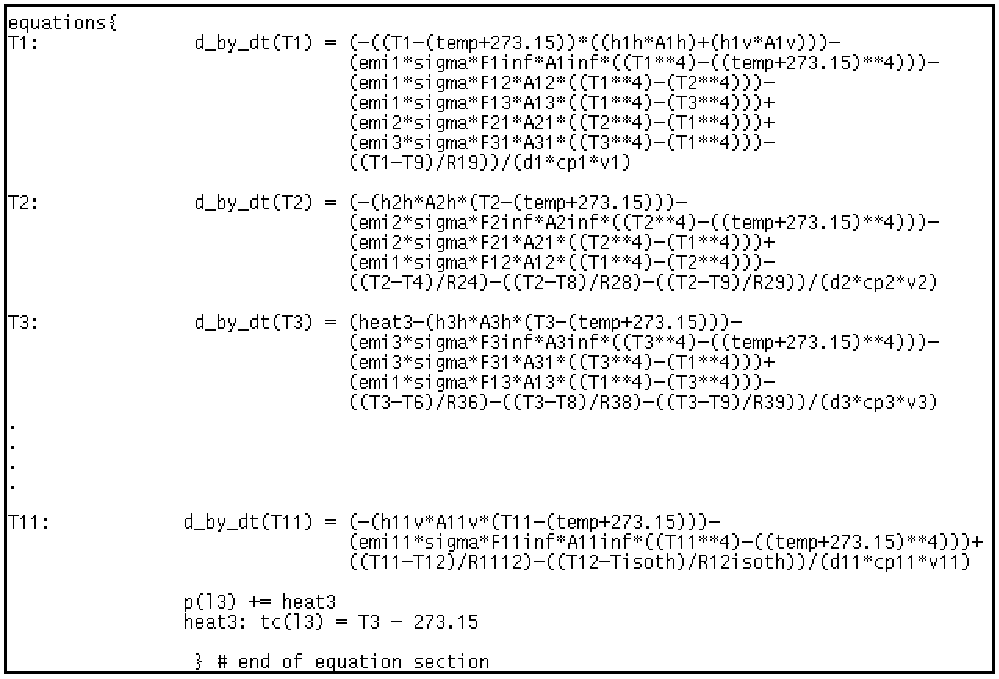


```
control_section {
                  initial_condition(T1,temp+275.15)
                  initial_condition(T2,temp+275.15)
                  initial_condition(T3,temp+275.15)
                  initial_condition(T4,temp+275.15)
                  initial_condition(T5,temp+275.15)
                  initial_condition(T6,temp+275.15)
                  initial_condition(T7,temp+275.15)
                  initial_condition(T8,temp+275.15)
                  initial_condition(T9,temp+275.15)
                  initial_condition(T10,temp+275.15)
                  initial_condition(T11,temp+275.15)

                } # end of control_section
```

Figure 5.4. Template control section.

Equation Section

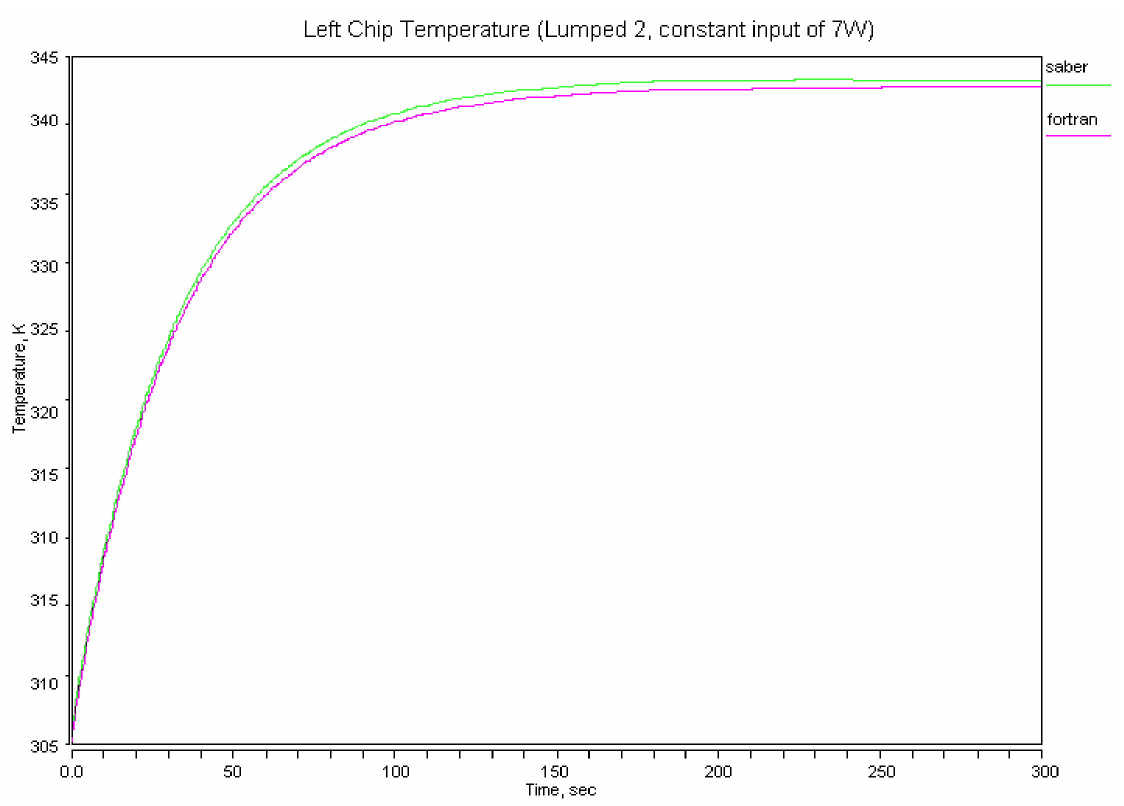
This is the most important section of the template. This part describes the analog characteristics at the terminals of the element being modeled, it is to say, this section describes the dependent *through* variable (power) of the system in terms of the *across* variable (temperature), according to Kirchoff's circuit law. Here, the model is developed and it also defines its external power input, as is presented in Figure 5.5.



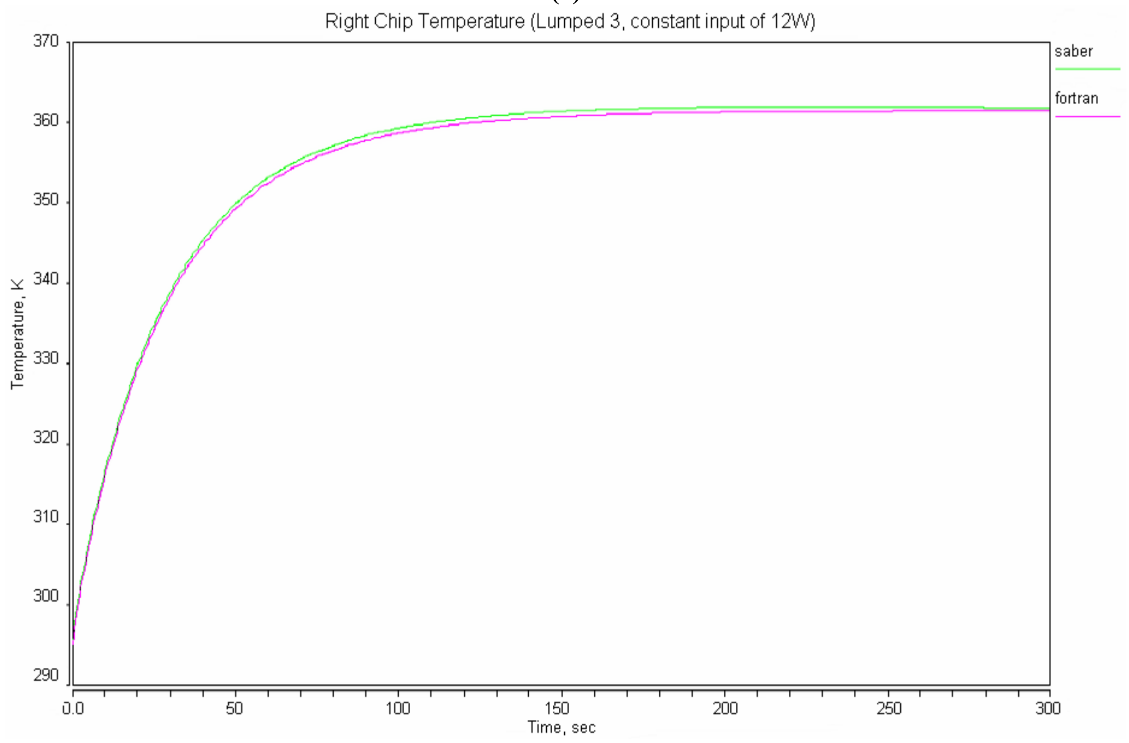
```
equations{
T1:            d_by_dt(T1) = (-((T1-(temp+273.15))*((h1h*A1h)+(h1v*A1v)))-
                             (emi1*sigma*F1inf*A1inf*((T1**4)-((temp+273.15)**4)))-
                             (emi1*sigma*F12*A12*((T1**4)-(T2**4)))-
                             (emi1*sigma*F13*A13*((T1**4)-(T3**4)))+
                             (emi2*sigma*F21*A21*((T2**4)-(T1**4)))+
                             (emi3*sigma*F31*A31*((T3**4)-(T1**4)))-
                             ((T1-T9)/R19))/(d1*cp1*v1)

T2:            d_by_dt(T2) = (-(h2h*A2h*(T2-(temp+273.15)))-
                             (emi2*sigma*F2inf*A2inf*((T2**4)-((temp+273.15)**4)))-
                             (emi2*sigma*F21*A21*((T2**4)-(T1**4)))+
                             (emi1*sigma*F12*A12*((T1**4)-(T2**4)))-
                             ((T2-T4)/R24)-((T2-T8)/R28)-((T2-T9)/R29))/(d2*cp2*v2)

T3:            d_by_dt(T3) = (heat3-(h3h*A3h*(T3-(temp+273.15)))-
                             (emi3*sigma*F3inf*A3inf*((T3**4)-((temp+273.15)**4)))-
                             (emi3*sigma*F31*A31*((T3**4)-(T1**4)))+
                             (emi1*sigma*F13*A13*((T1**4)-(T3**4)))-
                             ((T3-T6)/R36)-((T3-T8)/R38)-((T3-T9)/R39))/(d3*cp3*v3)
.
.
.
.
T11:          d_by_dt(T11) = (-(h11v*A11v*(T11-(temp+273.15)))-
                             (emi11*sigma*F11inf*A11inf*((T11**4)-((temp+273.15)**4)))+
                             ((T11-T12)/R1112)-((T12-Tisoth)/R12isoth))/(d11*cp11*v11)

              p(l3) += heat3
              heat3: tc(l3) = T3 - 273.15

               } # end of equation section
```

Figure 5.5. Template equation section.

The full Generation II IPEM model code in MAST modeling language is accessible in the Appendix F. The model implemented in the simulator was corroborated comparing the results with the model developed in FORTRAN (see Figure 5.6). While the FORTRAN model is solved using Runge-Kutta, the SABER model uses Newton-Raphson to solve the equations, and there is a very small discrepancy among results.



(a)



(b)

Figure 5.6. Comparison between FORTRAN and SABER^{TM} solution for the chips: (a) left; (b) right.

5.3 Simulated System Electrothermal Coupling

The second step in the development of the electrothermal model is the construction of an electrical circuit that holds similar characteristics to those of the high-speed thermal response experiment. However, a good option for simulating the experiment's similar condition is to use a schematic as that one shown in Figure 5.7. This schematic suggests that the electrothermal modeling is "one way", in other words, the full interaction between the electrothermal parameters is incomplete. The thermal model has no incidence on the pulse generator's model, but the power supplied by this electrical component affects the IPEM's transient temperatures. A fully electrothermal modeling of any package would have interaction between the electrical parameters (provided for electrical component models) and the component temperatures (from the thermal models) making it a cyclic process.

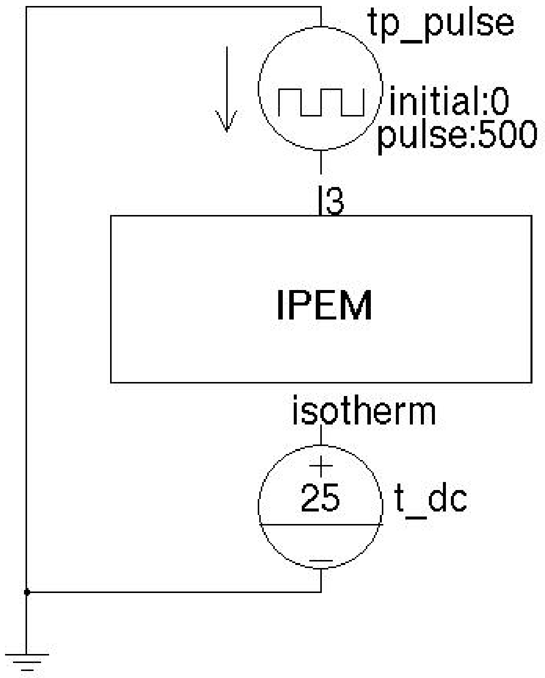


Figure 5.7. Simulated schematic in SABER^{TM}.

The coupling of the electrical components with the thermal model is quite simple. A pulse is directly applied to the thermal model (instead of to the devices). The thermal connections of the *thermal power source – pulse* (this SABER's ^{TM} element is a combination of a power supply and a pulse generator) are connected to the thermal connections of the *ltcm_ipem* element (that contain the reduced-order thermal model). The hotplate is simulated as a *constant temperature source* (at 25ºC), and connected to the *isotherm* terminal of the thermal model. The remaining connections of the thermal model are connected to thermal ground to serve as reference for the temperature signals generated in the model.

5.4 Comparison of the Simulation with Experimental Data

For the right power device, the experimental curves are more adequate to validate the simulations at the corresponding values of 100V-2.5A, 100V-5A, 200V-2.5A and 200V-5A. The comparisons between simulation and experimental data are presented in Figures 5.8, 5.9, 5.10 and 5.11, respectively. The comparisons are very favorable, maybe an inclusion of the device in the simulation could have improved the transients. It is evident that for low power the trajectory of the curves are more different, but for medium and high power the curves compare very well, in fact all curves achieve the experimental steady state temperature.

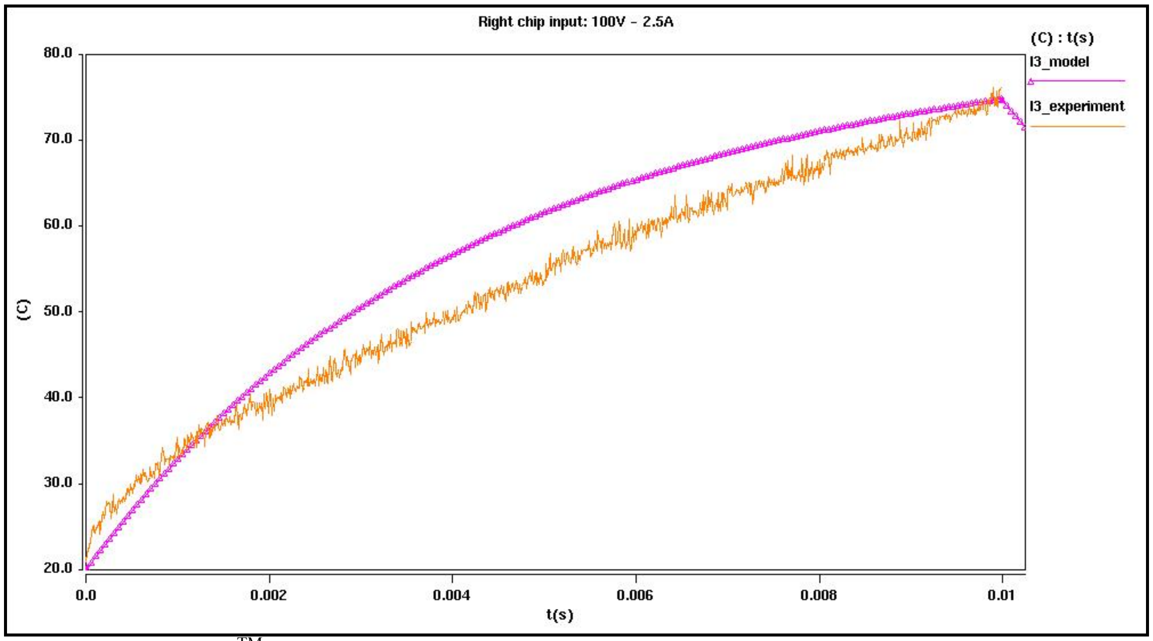


Figure 5.8. SABER^{TM} and experimental comparison for the right semiconductor device using inputs of 1Hz, 100V and 2.5A.

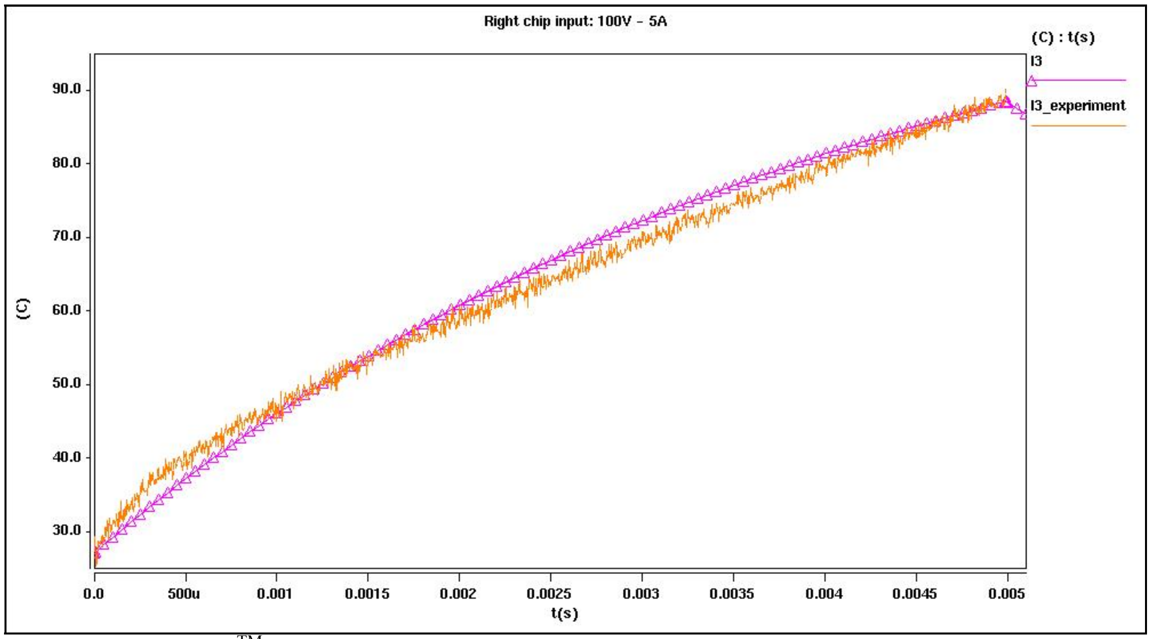


Figure 5.9. SABER^{TM} and experimental comparison for the right semiconductor device using inputs of 2Hz, 100V and 5A.

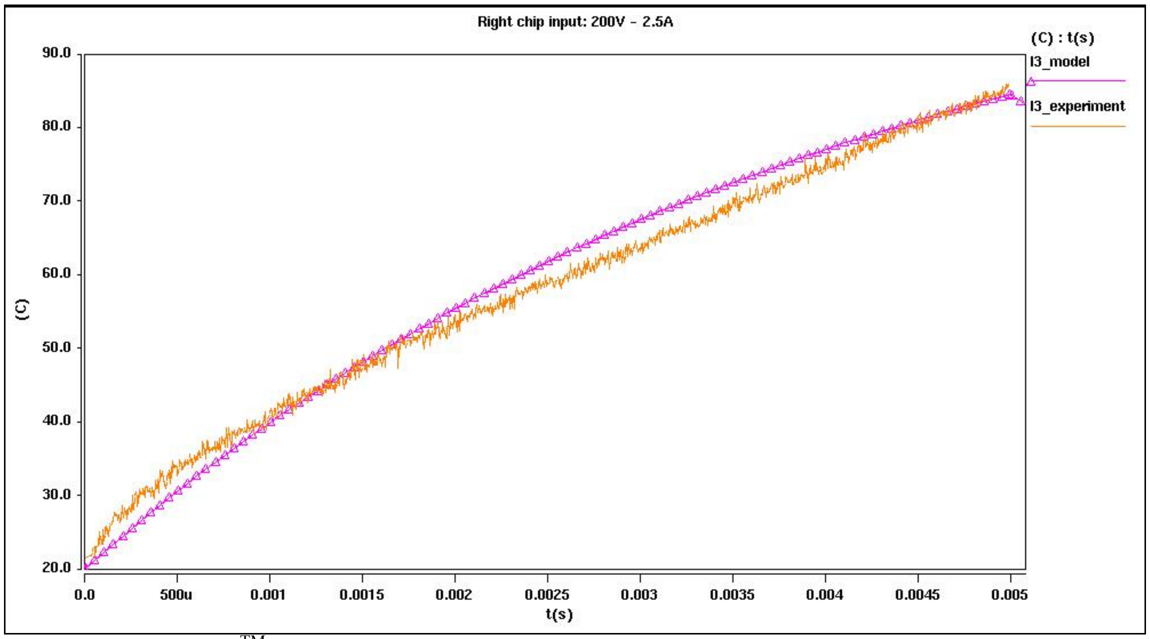


Figure 5.10. SABER^{TM} and experimental comparison for the right semiconductor device using inputs of 2Hz, 200V and 2.5A.

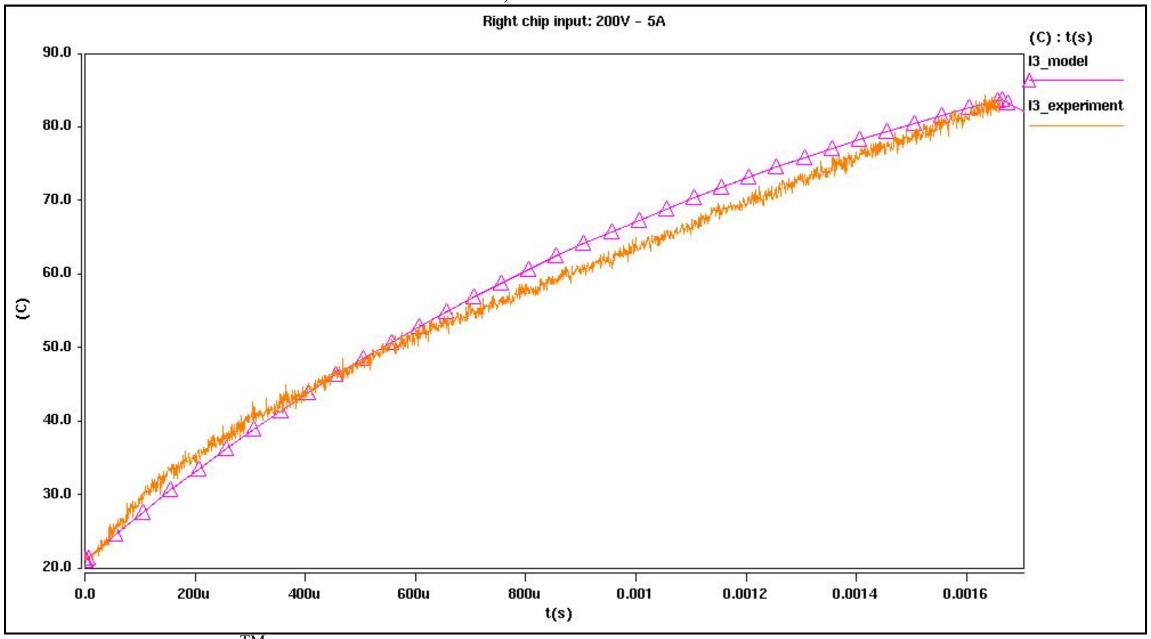


Figure 5.11. SABER^{TM} and experimental comparison for the right semiconductor device using inputs of 6Hz, 200V and 5A.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Summary

The major findings from this research study are the following:

- A simplified reduced-order model has been developed based on LTCM, using the energy balances and physical and geometry properties of each lumped. The model consists of a set of nonlinear differential equations, which were discretized using fourth-order Runge-Kutta method. This solution has been programmed using FORTRAN language.
- The model developed in FORTRAN establishes, with a satisfactory degree of agreement, the dynamic temperature variations inside the Generation II IPEM.
- Comparisons between a commercial software tool and the LTCM model reveal that the model is very truthful and useful to predict the thermal behavior of the working package.
- The thermal validation, with constant power input, is made using a slow thermal response testbed. The process in the experiment is merely thermal, because the electrical parameters do not have some thermal influence on them.
- The IPEM's electrical performance can be obtained using existing description of electrical components in a circuit simulator. The coupling of the LTCM model with component electrical models can also be made in the simulator platform.

- The electrothermal interaction within the working IPEM requires the experimental validation.
- The electrothermal validation, with dynamic interaction between thermal and electrical parameters, implicates the use of a fast thermal response experiment that includes an efficient data acquisition system.

6.2 Conclusions

A reduced-order model for the Generation II Integrated Power Electronics Modules (IPEM), based on the LTCM, has been developed in this work. IPEMs represent the new generation of power electronic modules that are significantly smaller in size and capable of handling large current densities. This scalability has brought new challenges in thermal management that requires immediate attention. One of the main challenges is to generate better simulation and design tools so they can be more efficient computationally without sacrificing reliability and accuracy.

The model was validated against a full 3-D finite volume approach demonstrating that the proposed methodology is very efficient and accurate. The validation of LTCM model using FLOTHERM^{TM} showed the reduced-order model advantages on the software tool. The computational time is visibly reduced, as well as the problem definition in terms of geometry construction, heat transfer, and grid solution. The new methodology established in this work provides a fast and accurate tool to quickly analyze the thermal behavior of any IPEM.

Two data acquisition system were developed (in UPRM and NIST), using LabWindows/CVI platform, to implement the low- and high-speed thermal response experiments. These programs can be used to obtain the transient temperature of any electronic package (obviously, including the IPEMs). The experimental set ups have also been built to acquire the experimental data used in the LTCM model validations. Using the low-speed thermal response experiment data, comparisons between experimental and FORTRAN-expansion simulation data have been made. The thermal model with constant power input compare favorably with the experiment at low and medium power input.

A full electrothermal model for the IPEM could be obtained utilizing the SABER^{TM}' electrical component models library and the reduced-order thermal model implemented in the simulator. The resulting electrothermal coupling's data compare favorably with the experimental data (which represent the dynamic electrothermal interface of the devices within the package) from the high-speed thermal response testbed.

An important thermal model issue is the effect of the thermal contact resistance on the model operation. When the model is used with the purpose of obtaining a thermal transient based only in a thermal process (this is, there are not external factors that influences the model solution, for instance, some electrical parameter), those resistance must have high values, as it happens in the slow thermal experiment, where the transient is obtained increasing the electrothermal parameters in small increments resulting in a purely thermal process. This way, the model requires that the lumped number agrees with the required volumes so that such resistances have sufficiently high values. The contrary case is when there are external factors that incident on the model (a dynamic power input), as it happens in the fast thermal experiment, where the temperature have a effect on the device voltage, and this one vary constantly in a short enough time interval, then in order that the model can capture the thermal transient this one must have relatively low resistances. Low resistances can be obtained increasing the lumped's contact surfaces (in the contact surface parameter) or increasing the lumped's volumes (in the heat spreading effects parameter).

6.3 Recommendations

In case that future research continues along the lines of the topic presented in this thesis, the following recommendations are suggested that could improve the quality of the results to be obtained.

- Improve the LTCM model resolution by increasing the lumped numbers to obtain a better temperature distribution of the IPEM.

- Develop an IPEM's general electrical model that can be implemented in any simulator and/or platform, and this way the full electrothermal can be obtained.

- Create an interface between the reduced-order thermal model and some M-CAD tool to automate and facilitate the power electronic package division in lumped (according to the Biot parameter validation) and the geometry and physical properties extraction process.

- Improve the experimental curves from high-speed thermal response testbed. Develop more calibration curves for the left chip.

- Include the MOSFET's electrical model in the SABER^{TM} simulation to obtain better curves to compare with the experimental data, like the transient voltages that allow the electrical validation of the electrothermal model.

BIBLIOGRAPHY

1. Hefner Jr. A. R., 1994, “A Dynamic Electro-Thermal Model for the IGBT”, Industry Applications, IEEE Transactions on, 30, Issue 2, pp. 394–405.

2. Hefner A. R. and Blackburn D. L., 1992, “Simulating the Dynamic Electro-Thermal Behavior of Power Electronic Circuits and Systems”, Computers in Power Electronics, IEEE Workshop on 1992, pp. 143-151.

3. Hefner A. R. and Blackburn D. L., 1994, “Thermal Component Models for Electrothermal Network Simulation”, Components, Packaging, and Manufacturing Technology, Part A, IEEE Transactions on [see also Components, Hybrids, and Manufacturing Technology, IEEE Transactions on], pp. 413-424.

4. Rodriguez J., Parrilla Z., Vélez-Reyes M., Hefner A., Berning D., Reichl J. and Lai J., 2002, “Thermal Component Models for Electrothermal Analysis of Multichip Power Modules”, Industry Applications Conference, 37th IAS Annual Meeting, Conference Record of the , pp. 234-241, vol. 1.

5. Bikdash M., Pang Y. P. and Scott E. P., 2002, “Generation of Equivalent Models from Simulation Data of a Thermal System”, ASME International Mechanical Engineering Congress & Exposition, pp 323-330, vol. 1.

6. Codecasa L., D’Amore D. and Maffezzoni P., 2002, “Modeling the Thermal Response of Semiconductor Devices through Equivalent Electrical Networks”, Circuits and Systems - I, IEEE Transactions on, 49, Issue 8, pp. 1187-1197.

7. Tummala R. R., 2001, “Fundamentals of Microsystems Packaging”, McGraw-Hill, pp. 239-262.

8. Holman J. P., 2001, “Heat Transfer”, McGraw-Hill Science/Engineering/Math, Ninth edition, pp. 226-244

9. Incropera F. and DeWitt D., 2002, “Introduction to Heat Transfer”, John Wiley and Sons, Fourth edition, pp. 751-766.

10. Mitchell F. H. Jr. and Mitchell F. H., 1988, “Introduction to Electronic Design”, Prentice Hall, pp. 259-268.

11. Dorf R. and Svoboda J., 1996, “Introduction to Electric Circuit”, John Wiley and Sons, Third edition, pp. 154-205.

12. Malik N., 1995, “Electric Circuit: Analysis, Simulation and Design”, Prentice Hall, pp. 245-268.

13. Liang Z., Lee F. C., Wyk V. and Lu G. Q., 2001, “Embedded Power Technology for IPEMs Packaging Applications”, Applied Power Electronics Conference and Exposition, APEC 2001. Sixteenth Annual IEEE, pp. 1057-1061 vol.2.

14. Chen J., Pang Y. F., Boroyevich D., Scott E. and Thole K., 2002, “Electrical and Thermal Layout Design Considerations for Integrated Power Electronics Modules”, Industry Applications Conference, 37th IAS Annual Meeting, Conference Record of the , pp. 242-246 vol.1.

15. Selberherr S., 1984, “Analysis and Simulation of Semiconductor Devices”, Springerverlang, pp. 123-138.

16. Hsu J. T. and Vu-Quoc L., 1996, “A Rational Formulation of Thermal Circuit Models for Electrothermal Simulation – Part I: Finite Element Method”, Circuit and Systems - I, IEEE Transactions on, 43, Issue 9, pp 721-732.

17. Hsu J. T. and Vu-Quoc L., 1996, “A Rational Formulation of Thermal Circuit Models for Electrothermal Simulation - Part II: Model Reduction Techniques”, Circuit and Systems - I, IEEE Transactions on, 43, Issue 9, pp 733-744.

18. Lee S. S. and Allstot D., 1993, “Electrothermal Simulation of Integrated Circuits”, IEEE Journal of Solid-State Circuits, 28, Issue 12, pp. 1283-1293.

19. Min Y. J., Palisoc A. and Lee C. C., 1990, “Transient Thermal Study of Semiconductor Devices”, Components, Hybrids, and Manufacturing Technology, IEEE Transaction on, 13, Issue 4, pp. 980-988.

20. Chen J. Z., Wu Y., Borojevich D. and Bohn J. H., 2000, “Integrated Electrical and Thermal Modeling and Analysis of IPEMs”, Computers in Power Electronics, 2000, COMPEL 2000, The 7th Workshop on, pp. 24–27.

21. Adams V. H., Joshi Y. and Blackburn D. L., 1999, “Three-Dimensional Study of Combined Conduction, Radiation, and Natural Convection From Discrete Heat Sources in a Horizontal Narrow-Aspect-Ratio Enclosure”, Journal of Heat Transfer, 121, pp. 992-1001.

22. Adams V. H., Blackburn D. L., Joshi Y. K. and Berning D. W., 1997, “Issues in Validating Package Compact Thermal Models for Natural Convection Cooled Electronic Systems”, Components, Packaging, and Manufacturing Technology, Part A, IEEE Transactions on [see also Components, Hybrids, and Manufacturing Technology, IEEE Transactions on] ,Volume: 20 , Issue: 4 , pp. 420-431.

23. Digele G., Lindenkreuz S. and Kasper E., 1997, "Fully Coupled Dynamic Electrothermal Simulation", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , Issue: 3 , pp. 250 – 257, vol. 5.

24. Chapra S. C and Canale R. P., 1999, "Numeric Method for Engineers", McGraw Hill, Second edition, pp. 159-168.

25. Yovanovich M. M., Culham J. R. and Teertstra P., 1997, "Calculating interface resistance", http://www.electronicscooling.com/Resources/EC_Articles/MAY97/article3.htm

26. Parrilla Z., Rodriguez J.J., Hefner A., Velez-Reyes M. and Berning D., 2002, "A Computer-Based System for Validation of Thermal Models for Multichip Power Modules", Computers in Power Electronics, Proceedings, 2002 IEEE Workshop on, pp. 42 – 46.

27. Blackburn D. L., 1988, "A Review of Thermal Characterization of Power Transistors", Semiconductor Thermal and Temperature Measurement Symposium, SEMI-THERM IV, Fourth Annual IEEE, pp. 1-7.

28. Berning D., Reichl, J., Hefner A., Hernandez M., Ellenwood C. and Lai J. S., 2003, "High Speed IGBT Module Transient Thermal Response Measurements for Model Validation", Industry Applications Conference, 38th IAS Annual Meeting. Conference Record of the , pp. 1826 – 1832, vol.3.

29. Khalid S. F., 2002, "Advanced Topics in LabWindows/CVI", National Instrument Virtual Instrumentation Series, Prentice Hall, pp. 35-243.

30. FLOTHERM User Manual Version 3.2, Flomerics, Ltd., Marlborough, MA, http://www.flomerics.com

31. Saber Designer User Guide Release 5.1, Analogy Inc., Beaverton, OR, http://www.analogy.com

APPENDIX A

DIMENSIONS AND CALCULATION OF SOME PROPERTIES USED IN THE THERMAL MODELING

A.1 Generation II IPEM's Dimensions

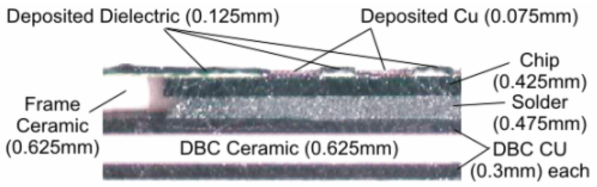


Figure A.1. Identification of layer materials and thicknesses.

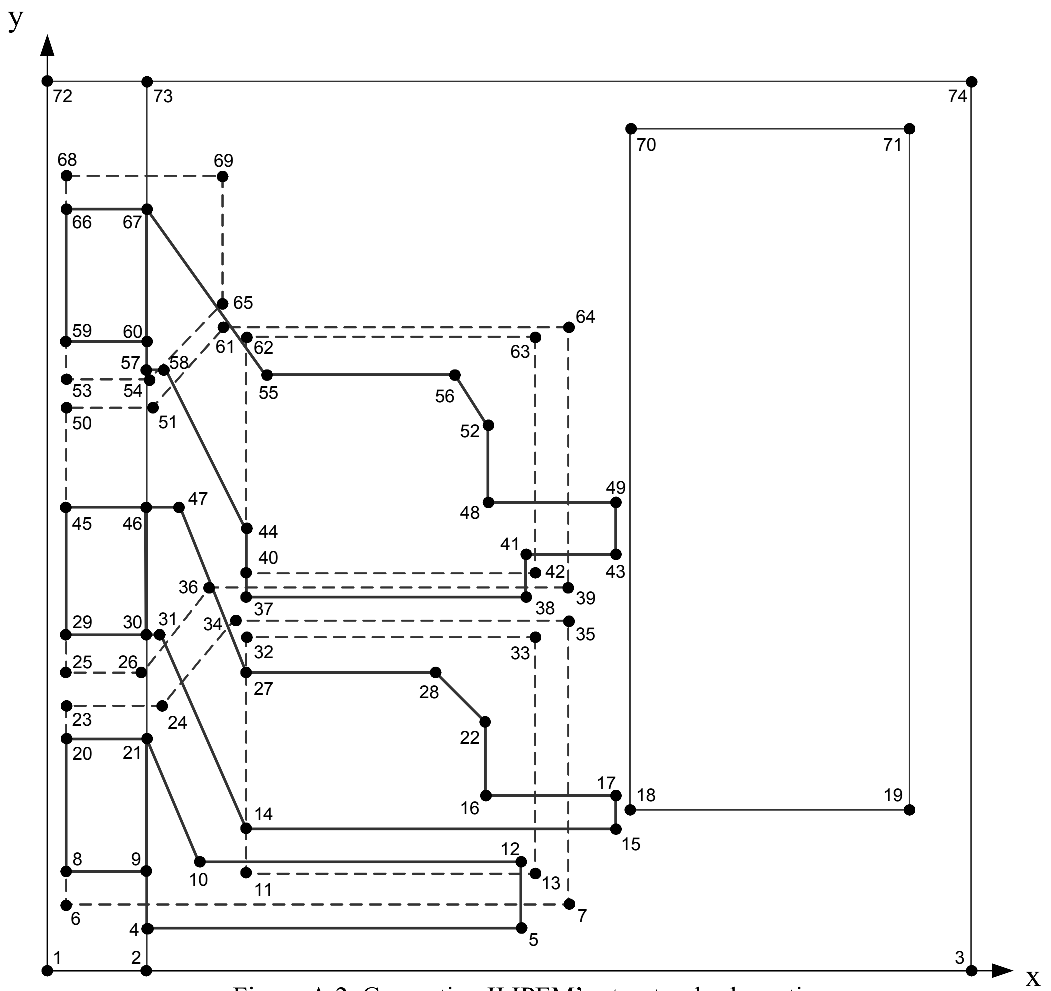


Figure A.2. Generation II IPEM's structural schematic.

Table A.1. Coordinates (x, y, z) for the Locations of the IPEM Layout Schematic Depicted in Figure A.2. (All dimensions are given in centimeters)

	Coordinates		
Point	x	y	z
1	0.0000	0.0000	0.0762
2	0.3048	0.0000	0.1892
3	2.8448	0.0000	0.1892
4	0.3048	0.1346	0.2159
5	1.4478	0.1346	0.2159
6	0.0508	0.2032	0.1029
7	1.5956	0.2032	0.1029
8	0.0508	0.3061	0.2159
9	0.3048	0.3061	0.2159
10	0.4572	0.3251	0.2159
11	0.6098	0.3048	0.1892
12	1.4478	0.3251	0.2159
13	1.4937	0.3048	0.1892
14	0.6096	0.4244	0.2159
15	1.7475	0.4244	0.2159
16	1.3462	0.5260	0.2159
17	1.7475	0.5260	0.2159
18	1.7983	0.4895	0.2527
19	2.6492	0.4895	0.2527
20	0.0508	0.7125	0.2159
21	0.3048	0.7112	0.2159
22	1.3462	0.7571	0.2159
23	0.0508	0.8147	0.1029
24	0.3567	0.8147	0.1029
25	0.0508	0.9197	0.1029
26	0.2911	0.9197	0.1029
27	0.6096	0.9146	0.2159
28	1.1887	0.9146	0.2159
29	0.0508	1.0213	0.2159
30	0.3048	1.0213	0.2159
31	0.3556	1.0213	0.2159
32	0.6098	1.0236	0.1892
33	1.4937	1.0236	0.1892
34	0.5588	1.0744	0.1029
35	1.5956	1.0744	0.1029
36	0.5047	1.1760	0.1029
37	0.6096	1.1491	0.2159
38	1.4732	1.1491	0.2159
39	1.5953	1.1760	0.1029
40	0.6098	1.2268	0.1892
41	1.4732	1.2888	0.2159
42	1.4937	1.2268	0.1892
43	1.7475	1.2888	0.2159
44	0.6096	1.3472	0.2159
45	0.0508	1.4277	0.2159
46	0.3048	1.4277	0.2159
47	0.4064	1.4277	0.2159
48	1.3462	1.4412	0.2159
49	1.7475	1.4412	0.2159
50	0.0508	1.7363	0.1029
51	0.3184	1.7363	0.1029
52	1.3462	1.6698	0.2159
53	0.0508	1.8417	0.1029
54	0.3184	1.8417	0.1029
55	0.6604	1.8324	0.2159
56	1.2395	1.8324	0.2159
57	0.3048	1.8527	0.2159
58	0.3556	1.8527	0.2159
59	0.0508	1.9373	0.2159
60	0.3048	1.9373	0.2159
61	0.5348	1.9811	0.1029
62	0.6098	1.9456	0.1892
63	1.4937	1.9456	0.1892
64	1.5953	1.9811	0.1029
65	0.5348	2.0697	0.1029
66	0.0508	2.3437	0.2159
67	0.3048	2.3437	0.2159
68	0.0508	2.4513	0.1029
69	0.5348	2.4513	0.1029
70	1.7983	2.5977	0.2527
71	2.6492	2.5977	0.2527
72	0.0000	2.7315	0.0762
73	0.3048	2.7315	0.1892
74	2.8448	2.7315	0.1892

A.2 Calculation of Properties used in the Modeling

Some parameters can be estimated directly from the IPEM's dimensions, such as the diverse areas and volumes, the transversal length for the heat transfer coefficient estimation, the equivalent length for the Biot number evaluation, among others. Other parameters must be calculated from some expression, such as contact surface resistance, the heat spreading effect resistance, the geometry factors for the radiation, , among others.

Thermal contact resistance

Five resistances were given by Pang in [14], these resistance are of the type of thermal coupling and correspond to these one: the interface between the gate driver and the ceramic frame ($R_{1\text{-}9}$), the interface between the silicon chips and the ceramic frame ($R_{2\text{-}9}$ and $R_{3\text{-}9}$), the interface between silicon chips and the etched copper trace ($R_{2\text{-}8}$ and $R_{3\text{-}8}$). The remaining resistances are calculated from the equations 3.20, 3.21 and 3.22.

The following example shows how to calculate the resistances associated to the interfaces between the silicon chips and the copper metallization layers ($R_{2\text{-}4}$ and $R_{3\text{-}6}$). First, the contact surface resistance is calculated using the Equation 3.20, where $A_{cont.} = 3.49\text{x}10^{-5}$ m^2, $k_{Si} = 124$ W/m.K, $k_{Cu} = 386$ W/m.K, $m = 0.24$, $\sigma = 2.86\text{x}10^{-6}$ m, $(P/H_c) = 1.5\text{x}10^{-2}$, $k_f = 28\text{x}10^{-3}$ W/m.K, $Y = 3.78\text{x}10^{-7}$ m and $M = 3.0049\text{x}10^{-7}$ m. From Equation 3.20, these resistances are:

$$R_{cont.\,2-4} = R_{cont.\,3-6}\ \frac{1/A_{2-4}}{\left[2.5\left(\frac{k_{Si}k_{Cu}}{k_{Si}+k_{Cu}}\right)\left(\frac{m}{\sigma}\right)\left(\frac{P}{H_c}\right)^{0.95}+\left(\frac{k_f}{Y-M}\right)\right]}$$

Substituting the corresponding values,

$$R_{cont2-4} = \frac{1/(3.49x10^{-5}\,m^2)}{\left[2.5\left(\frac{(124*386)(W/m.K)^2}{(124+386)\,W/m.K}\right)\left(\frac{0.24}{2.86x10^{-6}\,m}\right)\left(1.5x10^{-2}\right)^{0.95}+\left(\frac{28x10^{-3}\ W/m.K}{(3.78x10^{-7}-3.0049x10^{-7})m}\right)\right]}$$

$$R_{cont.\,2-4} = R_{cont.\,3-6} = 9.27\ K/W$$

The same values of the effective mean absolute asperity slope of the interface (m), thus as effective surface roughness (σ), the relationship between the contact pressure and the surface microhardness (P/H_c) (this parameter is named the relative contact pressure), the thermal conductivity of the air in the gap (k_f), the effective gap thickness (Y) and the gas-surface parameter (M) (this parameter represent the rarefaction effects), are used in order to find the remaining resistances. Now, the resistance by heat spreading effects must be determined from the Equation 3.21, where $\Delta x_{Si} = 4.45x10^{-3}$ m and $\Delta x_{Cu} = 1.27x10^{-3}$ m, this way,

$$R_{spreader\,2-4} = R_{spreader\,3-6} = \frac{1}{A_{2-4}}\left(\frac{\Delta x_{Si}}{k_{Si}}+\frac{\Delta x_{Cu}}{k_{Cu}}\right)$$

$$R_{spreader\,2-4} = \frac{1}{3.49x10^{-5}\,m^2}\left(\frac{4.45x10^{-3}\,m}{124\ W/m.K}+\frac{1.27x10^{-3}\,m}{386\ W/m.K}\right)$$

$$R_{spreader2-4} = R_{spreader3-6} = 1.12\,K/W$$

The total thermal contact resistance is calculated using the Equation 3.22, then:

$$R_{2-4} = R_{3\text{-}6} = R_{\text{cont}\,2-4/3-6} + R_{\text{spreader}2-4/3-6}$$

$$R_{2-4} = R_{3\text{-}6} = (9.27 + 1.12)\,\text{K/W}$$

$$R_{2-4} = R_{3\text{-}6} = 10.39\,\text{K/W}$$

The values of all resistances by thermal contact resistance are consigned in the Table 3.3.

Geometry factors

All geometry factors of the IPEM under study are given in the Table 3.3. In order to facilitate the calculations of these parameters in the model, in the Figure A.3 the areas "A" and "B" have been neglected, because, in comparison with the other areas involved, these ones are very smalls.

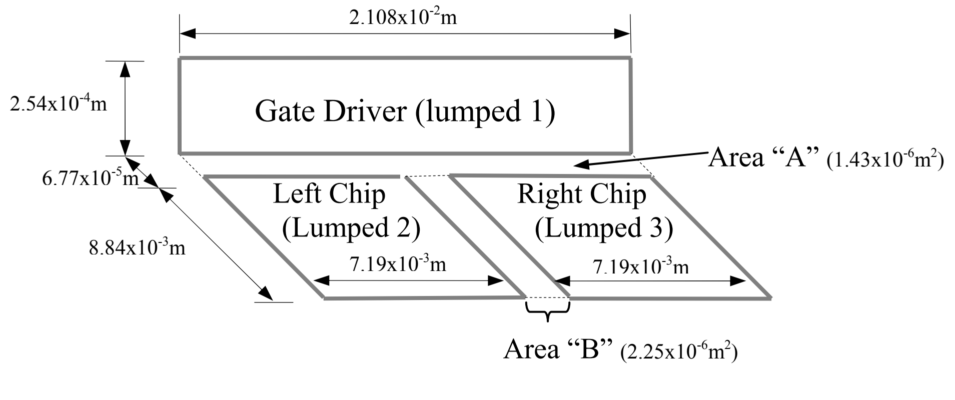


Figure A.3. Calculations of some geometry factors (F_{21}, F_{31}, F_{12}, F_{13}, $F_{1\infty}$, $F_{2\infty}$ and $F_{3\infty}$).

If the two chips and the area "B" are considered as a single area (which can be named "$A_{chips+B}$"), then, using Figure A.3 and the corresponding curves to the geometry factor for perpendicular rectangles with a common edge (see Figure A.4), it is possible to find some factors. This way,

$$\frac{Z}{X} = \frac{2.54x10^{-4}}{2.108x10^{-2}} = 0.012 \quad \text{and} \quad \frac{Y}{X} = \frac{8.84x10^{-3}}{2.108x10^{-2}} = 0.42 \quad \Rightarrow \quad F_{(chips+B)-1} = 0.1$$

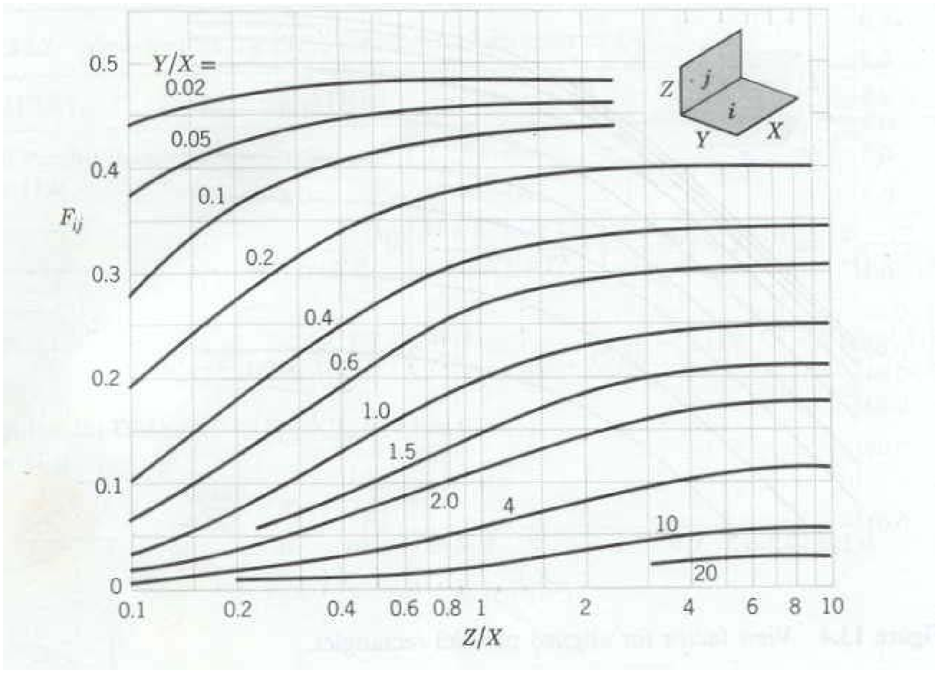


Figure A.4. Geometry Factor for Perpendicular Rectangles with a Common Edge from Incropera in [9] (Figure 13.6, pp. 755)

Since the area "B" can be ignored, then the area $A_{chips+B} = A_{chips} = A_2 + A_3$, and $A_2 = A_3$, therefore $A_{chips+B} = 2A_2$. Similarly, it can be considered that the half of the energy that leaves area "$A_{chips+B}$" towards the gate driver (lumped 1), corresponds to the energy that leaves each chips (left chip-lumped 2 and right chip-lumped 3) and flows to the gate driver, this is

$$\frac{1}{2}F_{(\text{chips+B})-1} = F_{2-1} = F_{3-1} = \frac{1}{2}(0.1) = 0.05$$

Using the reciprocity relation,

$$F_{1-2} = F_{1-3} = \frac{A_2 F_{2-1}}{A_1} = 0.49$$

By the summation rule,

$$F_{1-1} + F_{1-2} + F_{1-3} + F_{1-\infty} = 1, \text{ where } F_{1-1} = 0 \;\Rightarrow\; F_{1\text{-}\infty} = 1 - (F_{1-2} + F_{1-3}) = 1 - 2F_{1-2}$$

$$F_{1-\infty} = 1 - 2(0.49) = 0.02$$

$$F_{2-1} + F_{2-2} + F_{2-3} + F_{2-\infty} = 1, \text{ where } F_{2-2} = F_{2-3} = 0 \;\Rightarrow\; F_{2\text{-}\infty} = F_{3-\infty} = 1 - F_{2-1} = 1 - 0.05$$

$$F_{2-\infty} = F_{3-\infty} = 0.95$$

APPENDIX B

THERMAL MODEL EXPANDED USING FORTRAN

```
      implicit none

c-- constant values

******Dissipated Powers
      real    q1, q2, q3

**    CONVECTION
******Heat Transfer Coefficients
      real    h1h, h1v, h2h, h3h, h4h, h4v, h5h, h5v, h6h, h6v, h7h, h7v,
     &        h8v, h9h, h9v, h10v, h11v
******Environment Temperature
      real    Tinf
******Convection Lengths
      real    L1h, L1v, L2h, L3h, L4h, L4v, L5h, L5v, L6h, L6v, L7h, L7v,
     &        L8v, L9h, L9v, L10v, L11v
******Convection Areas
      real    Ac1h, Ac1v, Ac2h, Ac3h, Ac4h, Ac4v, Ac5h, Ac5v, Ac6h,
     &        Ac6v, Ac7h, Ac7v, Ac8v, Ac9h, Ac9v, Ac10v, Ac11v

**    RADIATION
******Boltzmann's Constant
      real    sigma
******Emisivities
      real    emi1, emi2, emi3, emi4, emi5, emi6, emi7, emi8, emi9,
     &        emi10, emi11
******Geometry Factors
      real    F1inf, F2inf,  F3inf, F4inf, F5inf, F6inf, F7inf, F8inf,
     &        F9inf, F10inf, F11inf, F12, F13, F21, F31
******Radiation Areas
      real    A1inf,  A2inf, A3inf, A4inf, A5inf, A6inf, A7inf, A8inf,
     &        A9inf, A10inf, A11inf, A12, A13, A21, A31

**    CONTACT RESISTENCE
      real    R19, R28, R38, R29, R39, R24, R36, R98, R59,
     &        R79, R810, R1011, R45, R67, R58, R78

**    TRANSIENT EFFECTS
******Densities
      integer d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11
******Volumes
      real    v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11
******Heat Capacities
      integer cp1, cp2, cp3, cp4, cp5, cp6, cp7, cp8, cp9, cp10, cp11

**    TIME STEP
      real    deltat

**    BIOT'S VARIABLES
******Heat Flow Thickness
      real    L1, L2, L3, L4, L5, L6, L7, L8, L9, L10, L11
******Thermal Conductivies
      integer kd1, kd2, kd3, kd4, kd5, kd6, kd7, kd8, kd9, kd10, kd11

      integer n

c-- variables
```

```
**    BIOT NUMBER
      real    Bi

**    LUMPED TEMPERATURES
      real    T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11
      real    T1next, T2next, T3next, T4next, T5next, T6next, T7next,
     &        T8next, T9next, T10next, T11next

**    RUNGE-KUTTA'S VARIABLES
      real    k1, k2, k3, k4

      sigma = 5.67E-8             !W/m**2.K**4
      deltat = 1.0                !sec

c--   Call all files that contain the necessary data
      open (unit=10, file='lconv.dat', status='unknown')
      open (unit=11, file='kcond.dat', status='unknown')
      open (unit=12, file='length.dat', status='unknown')
      open (unit=13, file='tamb.dat', status='unknown')
      open (unit=14, file='power.dat', status='unknown')
      open (unit=15, file='emi.dat', status='unknown')
      open (unit=16, file='factor.dat', status='unknown')
      open (unit=17, file='aconv.dat', status='unknown')
      open (unit=18, file='arad.dat', status='unknown')
      open (unit=19, file='contact.dat', status='unknown')
      open (unit=20, file='density.dat', status='unknown')
      open (unit=21, file='volume.dat', status='unknown')
      open (unit=22, file='cp.dat', status='unknown')
      open (unit=24, file='temp.dat', status='unknown')
      open (unit=30, file='resultados.dat', status='unknown')

c--   First, we get the given information

      read (10,10)        h1h, h1v, h2h, h3h, h4h, h4v, h5h, h5v, h6h, h6v, h7h, h7v,
     &        h8v, h9h, h9v, h10v, h11v
10    format (17F6.4)

      read (11,55)        kd1, kd2, kd3, kd4, kd5, kd6, kd7, kd8, kd9, kd10, kd11
55    format (11I3)

      read (12,25)        L1, L2, L3, L4, L5, L6, L7, L8, L9, L10, L11
25    format (11F6.4)

      read (13,70)        Tinf
70    format (F6.2)

      read (14,50)        q1, q2, q3
50    format (3F4.1)

      read (15,45)        emi1, emi2, emi3, emi4, emi5, emi6, emi7, emi8, emi9,
     &                    emi10, emi11
45    format (11F5.3)

      read (16,30)        F1inf, F2inf,  F3inf, F4inf, F5inf, F6inf, F7inf,
     &                    F8inf, F9inf, F10inf, F11inf, F12, F13, F21, F31
30    format (15F6.4)

      read (17,75)        Ac1h, Ac1v, Ac2h, Ac3h, Ac4h, Ac4v, Ac5h, Ac5v, Ac6h,
     &        Ac6v, Ac7h, Ac7v, Ac8v, Ac9h, Ac9v, Ac10v, Ac11v
75    format (17F10.8)

      read (18,80)        A1inf,  A2inf, A3inf, A4inf, A5inf, A6inf, A7inf,
     &                    A8inf, A9inf, A10inf, A11inf, A12, A13, A21, A31
80    format (15F9.7)
```

```
     read (19,35)          R19, R28, R38, R29, R39, R24, R36, R98, R59,
    &          R79, R810, R1011, R45, R67, R58, R78
35    format (16F7.4)

     read (20,20)          d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11
20    format (11I4)

     read (21,40)          v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11
40    format (11F12.10)

     read (22,55)          cp1, cp2, cp3, cp4, cp5, cp6, cp7, cp8, cp9, cp10, cp11

     read (24,60)          T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11
60    format (11F6.2)


c--   Here start the loop that develops the Runge-Kutta method on each lumped

     do n = 1, 30

**    FOR THE LUMPED 1

c--   Calculates the heat transfer coefficients
      h1h=1.32*(((T1-Tinf)/L1h)**0.25)
      h1v=1.42*(((T1-Tinf)/L1v)**0.25)

c--   calculates the Biot number
      Bi=(h1h*L1)/kd1

      if (Bi .GT. 0.1) then
       stop
      end if

      T1next = T1
      k1= (1/(d1*v1*cp1))*(q1-((T1next-Tinf)*((h1h*Ac1h)+(h1v*Ac1v)))
    &    -(emi1*sigma*F1inf*A1inf*((T1next**4)-(Tinf**4)))
    &    -(emi1*sigma*F12*A12*((T1next**4)-(T2**4)))
    &    -(emi1*sigma*F13*A13*((T1next**4)-(T3**4)))
    &    +(emi2*sigma*F21*A21*((T2**4)-(T1next**4)))
    &    +(emi3*sigma*F31*A31*((T3**4)-(T1next**4)))
    &    -((T1next-T19)/R19))

      print*, 'k1=', k1
      pause

      T1next = T1+(deltat*k1/2)

      print*, 'T1next', T1next
      pause

      k2= (1/(d1*v1*cp1))*(q1-((T1next-Tinf)*((h1h*Ac1h)+(h1v*Ac1v)))
    &    -(emi1*sigma*F1inf*A1inf*((T1next**4)-(Tinf**4)))
    &    -(emi1*sigma*F12*A12*((T1next**4)-(T2**4)))
    &    -(emi1*sigma*F13*A13*((T1next**4)-(T3**4)))
    &    +(emi2*sigma*F21*A21*((T2**4)-(T1next**4)))
    &    +(emi3*sigma*F31*A31*((T3**4)-(T1next**4)))
    &    -((T1next-T19)/R19))

      T1next = T1+(deltat*k2/2)

      k3= (1/(d1*v1*cp1))*(q1-((T1next-Tinf)*((h1h*Ac1h)+(h1v*Ac1v)))
    &    -(emi1*sigma*F1inf*A1inf*((T1next**4)-(Tinf**4)))
    &    -(emi1*sigma*F12*A12*((T1next**4)-(T2**4)))
    &    -(emi1*sigma*F13*A13*((T1next**4)-(T3**4)))
    &    +(emi2*sigma*F21*A21*((T2**4)-(T1next**4)))
    &    +(emi3*sigma*F31*A31*((T3**4)-(T1next**4)))
```

```
     &    -((T1next-T19)/R19))

      T1next = T1+(deltat*k3)

      k4= (1/(d1*v1*cp1))*(q1-((T1next-Tinf)*((h1h*Ac1h)+(h1v*Ac1v)))
     &    -(emi1*sigma*F1inf*A1inf*((T1next**4)-(Tinf**4)))
     &    -(emi1*sigma*F12*A12*((T1next**4)-(T2**4)))
     &    -(emi1*sigma*F13*A13*((T1next**4)-(T3**4)))
     &    +(emi2*sigma*F21*A21*((T2**4)-(T1next**4)))
     &    +(emi3*sigma*F31*A31*((T3**4)-(T1next**4)))
     &    -((T1next-T19)/R19))

c--   The solution for the fourth order equation for Lumped 1 will be

      T1next = T1 + (deltat/6)*(k1+(2*k2)+(2*k3)+k4)
.
.
.
.
.
.
.
.
.
.
.


**    FOR THE LUMPED 11

c--   Calculates the heat transfer coefficients
      h11v=1.42*(((T11-Tinf)/L11v)**0.25)

c--   calculates the Biot number
      Bi=(h11h*L11)/kd11

      if (Bi .GT. 0.1) then
       stop
      end if

      k1= (1/(d11*v11*cp11))*(-( h11v*Ac11v* (T11next-Tinf))
     &    -(emi11*sigma*F11inf*A11inf*((T11next**4)-(Tinf**4)))
     &    +((T10-T11next)/R1011))

      T11next = T11+(deltat*k1/2)

      k2= (1/(d11*v11*cp11))*(-( h11v*Ac11v* (T11next-Tinf))
     &    -(emi11*sigma*F11inf*A11inf*((T11next**4)-(Tinf**4)))
     &    +((T10-T11next)/R1011))

      T11next = T11+(deltat*k2/2)

      k3= (1/(d11*v11*cp11))*(-( h11v*Ac11v* (T11next-Tinf))
     &    -(emi11*sigma*F11inf*A11inf*((T11next**4)-(Tinf**4)))
     &    +((T10-T11next)/R1011))

      T11next = T11+(deltat*k3)

      k4= (1/(d11*v11*cp11))*(-( h11v*Ac11v* (T11next-Tinf))
     &    -(emi11*sigma*F11inf*A11inf*((T11next**4)-(Tinf**4)))
     &    +((T10-T11next)/R1011))

c--   The solution for the fourth order equation for Lumped 11 will be

      T11next = T11 + (deltat/6)*(k1+(2*k2)+(2*k3)+k4)
      T1 = T1next
      T2 = T2next
      T3 = T3next
      T4 = T4next
      T5 = T5next
      T6 = T6next
      T7 = T7next
      T8 = T8next
      T9 = T9next
      T10 = T10next
      T11 = T11next

      write (30,65)        T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11
65     format (1X, 11F12.4)

     end do

     end

c-------------------------------------------------------------
```

APPENDIX C

FULL C++/CVI CODE OF THE TRANSIENT IPEM PROGRAM DEVELOPED FOR SLOW THERMAL RESPONSE EXPERIMENT

```
#define TRUE                    1
#define FALSE                   0
/* Used in selecting state of the termocouples to plot curves */
#define on                      1
#define off                     0
/* Used in selecting data options to load, display and save    */
#define TT                      0
#define ET                      1
#define MEAS                    2
/* Used in selecting output electrical data to save                 */
#define VI                      0
#define PWR                     1

/*============================FUNCTION DECLARATION==========================*/
void SelectTC (int, int, int, int, int, int, int, int);
void ClearBuff(char[]);
/*============================VARIABLE DECLARATION==========================*/
static double tc1[10000],tc2[10000],tc3[10000],tc4[10000],tc5[10000],tcair[10000];
static double tp1[10000],tp2[10000],tp3[10000],tp4[10000],tp5[10000],tpair[10000];
static double temp1[10000],temp2[10000],temp3[10000],temp4[10000],temp5[10000],tempair[10000];
static double time_array[10000],time_meas[10000,time[10000],voltms[10000],currms[10000],watts[10000];
static double volt[10000],curr[10000],voltage[10000],current[10000],pwr[10000],power[10000];
static double meas1,meas2,meas3,meas4,meas5,measair,measvolt,meascurr,meastime;
static double initial_time,current_time,no_of_samples;
static char filin[512];
static int status1,status2,status3,status4,status5,statusair,elec_meas,keit2001,hp6030a,err,pts;
int h_i=0,m_i=0,s_i=0,h_a=0,m_a=0,s_a=0,time_i=0,time_a=0,time_t=0,sampling_interval;
int actual_sample_no,i;
/*===============================MAIN PROGRAM=============================*/
main()      {
double freq,ampl,frequency,voltios,amper;
char buf[21];
int int1,int2,c,i,j,id,points,pt,max_pt,panel,handle,flhand,loader,display,saver,elect_option;
int grap_t1,grap_t2,grap_t3,grap_t4,grap_t5,grap_air,grap_all;
short int waveform;
panel = LoadPanel (0, "ipem2_2.uir", PNL);
if (panel < 0) {
  FmtOut ("Unable to load the required panel from the resource file.\n");
  return;
  } //end if
DisplayPanel (panel);
//Initialize switch system
kei7001_init (7, 1, 1, &int1);
//Initialize DMM
kei2001_init (16, 1, &int2);
keit2001 = ibdev (0, 16, NO_SAD, T10s, 1, 0);
//Initialize Waveform Generator
//hp33120a_init (10, 1, 1, &waveform);
//hp33120 = ibdev (0, 10, NO_SAD, T10s, 1, 0);
//Initialize Power Supply instrument
hp6xxxa_init ("GPIB::5::INSTR", 1, 1, 0, &hp6030a);
//Configure to read temperature
kei7001_conf_slots (1, 2, 26, 1);   //OJO VERIFICAR SLOT (yo elegi el 2) Y 7014!!!
kei2001_change_func (1, 7);
kei2001_conf_temp (1, -1.0, 0, 2, 0, 2);//OJO DEBO VERIFICAR EL TIPO DE TC!!!
while (TRUE) {
            GetUserEvent (TRUE, &handle, &id);
            switch (id) {
```

```
case PNL_LOAD :
        SetCtrlVal (panel, PNL_BUSY, 1);
        GetCtrlVal (panel, PNL_STATUS, &loader);
        switch (loader)         {
                //Load thermal transient data from file
                case TT :
                        err = FileSelectPopup ("therm_file", "*.dat", "", "Load Window",
                                                VAL_LOAD_BUTTON, 0, 0, 1, 1, filin);
                        if (err!=0){
                                flhand = OpenFile (filin, 1, 2, 1);
                                ClearBuff (buf);
                                ScanFile (flhand, "%s>%i", &c);
                                ScanFile (flhand, "%s>%i", &pt);
                                Clear1D (temp1, pt);
                                Clear1D (temp2, pt);
                                Clear1D (temp3, pt);
                                Clear1D (temp4, pt);
                                Clear1D (temp5, pt);
                                Clear1D (tempair, pt);
                                Clear1D (time, pt);
                                for (i=0; (i<pt); i++) {
                                        for (j=0; (j<c); j++) {
                                         switch (j) {
                                           case 0 :
                                             ScanFile (flhand, "%s>%f", &temp1[i]);
                                             break;
                                            case 1 :
                                              ScanFile (flhand, "%s>%f", &temp2[i]);
                                              break;
                                            case 2 :
                                              ScanFile (flhand, "%s>%f", &temp3[i]);
                                              break;
                                            case 3 :
                                              ScanFile (flhand, "%s>%f", &temp4[i]);
                                              break;
                                            case 4 :
                                              ScanFile (flhand, "%s>%f", &temp5[i]);
                                              break;
                                            case 5 :
                                              ScanFile (flhand, "%s>%f", &tempair[i]);
                                              break;
                                            case 6 :
                                              ScanFile (flhand, "%s>%f", &time[i]);
                                              break;  }          }          }
                                err = CloseFile (flhand);          } //end if (err!=0)
                        break; //end case TT
                //Load electrical transient data from file
                case ET :
                        err = FileSelectPopup ("elect_file", "*.dat", "", "Load Window",

                                               VAL_LOAD_BUTTON, 0, 0, 1, 1, filin);
                        if (err!=0){
                                flhand = OpenFile (filin, 1, 2, 1);
                                ClearBuff (buf);
                                ScanFile (flhand, "%s>%i", &pt);
                                Clear1D (power, pt);
                                Clear1D (time, pt);
                                for (i=0; (i<pt); i++) {
                                        ScanFile (flhand, "%s>%f", &power[i]);
                                        ScanFile (flhand, "%s>%f", &time[i]);
                                        }
                                err = CloseFile (flhand);
                                } //end if (err!=0)
                        break; //end case ET
                } /*end switch (loader) */
        SetCtrlVal (panel, PNL_BUSY, 0);
        break; //end case LOAD
    case PNL_DISPLAY :
        SetCtrlVal (panel, PNL_BUSY, 1);
        GetCtrlVal (panel, PNL_STATUS, &display);
        switch (display)        {
            //Display the loaded thermal transient data from file
            case TT :
                DeleteGraphPlot (panel, PNL_CURVE, -1, 1);
                SetCtrlVal (panel, PNL_LEDT1, 1);
                PlotXY (panel, PNL_CURVE, time, temp1, pt, VAL_DOUBLE,
                VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE,
                VAL_SOLID, 1,VAL_RED);
                SetCtrlVal (panel, PNL_LEDT2, 1);
                PlotXY (panel, PNL_CURVE, time, temp2, pt, VAL_DOUBLE,
                VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE,
                VAL_SOLID, 1,VAL_BLUE);
                SetCtrlVal (panel, PNL_LEDT3, 1);
                PlotXY (panel, PNL_CURVE, time, temp3, pt, VAL_DOUBLE,
                VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE,
                VAL_SOLID, 1,VAL_YELLOW);
                SetCtrlVal (panel, PNL_LEDT4, 1);
                PlotXY (panel, PNL_CURVE, time, temp4, pt, VAL_DOUBLE,
                VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE,
                VAL_SOLID, 1,VAL_GREEN);
                SetCtrlVal (panel, PNL_LEDT5, 1);
                PlotXY (panel, PNL_CURVE, time, temp5, pt, VAL_DOUBLE,
                VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE,
                VAL_SOLID, 1,VAL_WHITE);
                SetCtrlVal (panel, PNL_LEDTair, 1);
                PlotXY (panel, PNL_CURVE, time, tempair, pt, VAL_DOUBLE,
                VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE,
                VAL_SOLID, 1,VAL_CYAN);
                SetCtrlVal (panel, PNL_BUSY, 0);
                break; //end case TT
            case ET :
                DeleteGraphPlot (panel, PNL_CURVELECTRIC, -1, 1);
                PlotXY (panel, PNL_CURVELECTRIC, time, power, pt,
                VAL_DOUBLE,VAL_DOUBLE, VAL_THIN_LINE,
                VAL_EMPTY_SQUARE, VAL_SOLID, 1,VAL_DK_BLUE);
                break; //end case ET
            case MEAS :
                DeleteGraphPlot (panel, PNL_CURVE, -1, 1);
                GetCtrlVal (panel, PNL_T1, &status1);
                if (status1==1) {
                    SetCtrlVal (panel, PNL_LEDT1, 1);
                PlotXY (panel, PNL_CURVE, time_array, tc1, pts,
                VAL_DOUBLE,VAL_DOUBLE, VAL_THIN_LINE,
                VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED);           }
                GetCtrlVal (panel, PNL_T2, &status2);
                if (status2==1) {
                    SetCtrlVal (panel, PNL_LEDT2, 1);
                    PlotXY (panel, PNL_CURVE, time_array, tc2, pts,
                    VAL_DOUBLE, VAL_DOUBLE, VAL_THIN_LINE,
                    VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_BLUE);
                                }
                GetCtrlVal (panel, PNL_T3, &status3);
                if (status3==1) {
                    SetCtrlVal (panel, PNL_LEDT3, 1);
                    PlotXY (panel, PNL_CURVE, time_array, tc3, pts,
                    VAL_DOUBLE, VAL_DOUBLE, VAL_THIN_LINE,
                    VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_YELLOW);
                                }
                GetCtrlVal (panel, PNL_T4, &status4);
                if (status4==1) {
                    SetCtrlVal (panel, PNL_LEDT4, 1);
```

```
                                        PlotXY (panel, PNL_CURVE, time_array, tc4, pts,
                                        VAL_DOUBLE, VAL_DOUBLE, VAL_THIN_LINE,
                                        VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_GREEN);
                                                        }
                                GetCtrlVal (panel, PNL_T5, &status5);
                                if (status5==1) {
                                        SetCtrlVal (panel, PNL_LEDT5, 1);
                                        PlotXY (panel, PNL_CURVE, time_array, tc5, pts,
                                        VAL_DOUBLE, VAL_DOUBLE, VAL_THIN_LINE,
                                        VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_WHITE);
                                                        }
                                GetCtrlVal (panel, PNL_Tair, &statusair);
                                if (statusair==1) {
                                        SetCtrlVal (panel, PNL_LEDTair, 1);
                                        PlotXY (panel, PNL_CURVE, time_array, tcair, pts,
                                        VAL_DOUBLE, VAL_DOUBLE, VAL_THIN_LINE,
                                        VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_CYAN);
                                                        }
                                DeleteGraphPlot (panel, PNL_CURVELECTRIC, -1, 1);
                                GetCtrlVal (panel, PNL_MEASELECT, &elec_meas);
                                if (elec_meas==1) {
                                        SetCtrlVal (panel, PNL_LEDELECT, 1);
                                        PlotXY (panel, PNL_CURVELECTRIC, time_array, pwr, pts,
                                        VAL_DOUBLE, VAL_DOUBLE, VAL_THIN_LINE,
                                        VAL_EMPTY_SQUARE, VAL_SOLID, 1,VAL_DK_BLUE);
                                                }
                                break; //end case MEAS          } /*end switch (display) */
                SetCtrlVal (panel, PNL_BUSY, 0);
                break; //end case DISPLAY
        case PNL_CLEAR :
                SetCtrlVal (panel, PNL_BUSY, 1);
                SetCtrlVal (panel, PNL_LEDT1, 0);
                SetCtrlVal (panel, PNL_LEDT2, 0);
                SetCtrlVal (panel, PNL_LEDT3, 0);
                SetCtrlVal (panel, PNL_LEDT4, 0);
                SetCtrlVal (panel, PNL_LEDT5, 0);
                SetCtrlVal (panel, PNL_LEDTair, 0);
                SetCtrlVal (panel, PNL_LEDELECT, 0);
                DeleteGraphPlot (panel, PNL_CURVE, -1, 1);
                DeleteGraphPlot (panel, PNL_CURVELECTRIC, -1, 1);
                SetCtrlVal (panel, PNL_BUSY, 0);
                break; //end case CLEAR
        case PNL_SAVE :
                SetCtrlVal (panel, PNL_BUSY, 1);
                GetCtrlVal (panel, PNL_STATUS, &saver);
                switch (saver)          {
                        //Save the measured thermal transient data
                        case TT :
                                err = FileSelectPopup ("save_therm", "*.dat", "", "Save Window",
                                                VAL_SAVE_BUTTON, 0, 0, 1, 1, filin);
                                if (err!=0) {
                                        flhand = OpenFile (filin, 2, 0, 1);
                                        ClearBuff (buf);
                                        c=7;
                                        err = Fmt (buf, "%s<%i", c);
                                        err = WriteLine (flhand, buf, 7);
                                        ClearBuff (buf);
                                        err = Fmt (buf, "%s<%i", pts);
                                        err = WriteLine (flhand, buf, 7);
                                        for (i=0; (i<pts); i++) {
                                                ClearBuff (buf);
                                                err = Fmt (buf, "%s<%f", tc1[i]);
                                                err = WriteFile (flhand, buf, 15);
                                                ClearBuff (buf);
                                                err = Fmt (buf, "%s<%f", tc2[i]);
                                                err = WriteFile (flhand, buf, 15);
                                                ClearBuff (buf);
                                                err = Fmt (buf, "%s<%f", tc3[i]);
                                                err = WriteFile (flhand, buf, 15);
                                                ClearBuff (buf);
                                                err = Fmt (buf, "%s<%f", tc4[i]);
                                                err = WriteFile (flhand, buf, 15);
                                                ClearBuff (buf);
                                                err = Fmt (buf, "%s<%f", tc5[i]);
                                                err = WriteFile (flhand, buf, 15);
                                                ClearBuff (buf);
                                                err = Fmt (buf, "%s<%f", tcair[i]);
                                                err = WriteFile (flhand, buf, 15);
                                                ClearBuff (buf);
                                                err = Fmt (buf, "%s<%f", time_array[i]);
                                                err = WriteLine (flhand, buf, 7);
                                                } //end for
                                        CloseFile (flhand);
                                        } //end if (err!=0)
                                break; //end case TT
                        //Save the measured electrical transient data
                        case ET :
                                GetCtrlVal (panel, PNL_SAVELECT, &elect_option);
                                switch (elect_option)  {
                                  //Save the voltage and current data
                                  case VI :
                                        err = FileSelectPopup ("save_vi", "*.dat", "", "Save
                                        Window",VAL_SAVE_BUTTON, 0, 0, 1, 1, filin);
                                        if (err!=0) {
                                                flhand = OpenFile (filin, 2, 0, 1);
                                                ClearBuff (buf);
                                                err = Fmt (buf, "%s<%i", pts);
                                                err = WriteLine (flhand, buf, 7);
                                                for (i=0; (i<pts); i++) {
                                                        ClearBuff (buf);
                                                        err = Fmt (buf, "%s<%f", volt[i]);
                                                        err = WriteFile (flhand, buf, 15);
                                                        ClearBuff (buf);
                                                        err = Fmt (buf, "%s<%f", curr[i]);
                                                        err = WriteLine (flhand, buf, 15);  }
                                                        CloseFile (flhand);      } //end if (err!=0)
                                                break; //end case VI
                                        //Save the power and time data
                                  case PWR :
                                        err = FileSelectPopup ("save_power", "*.dat", "", "Save
                                        Window",VAL_SAVE_BUTTON, 0, 0, 1, 1, filin);
                                        if (err!=0) {
                                                flhand = OpenFile (filin, 2, 0, 1);
                                                ClearBuff (buf);
                                                err = Fmt (buf, "%s<%i", pts);
                                                err = WriteLine (flhand, buf, 7);
                                                for (i=0; (i<pts); i++) {
                                                   ClearBuff (buf);
                                                   err = Fmt (buf, "%s<%f", pwr[i]);
                                                   err = WriteFile (flhand, buf, 15);
                                                   ClearBuff (buf);
                                                   err = Fmt (buf, "%s<%f", time_array[i]);
                                                   err = WriteLine (flhand, buf, 15);             }
                                                CloseFile (flhand);                     } //end if (err!=0)
                                        break; //end case PWR                }
                                break; //end case ET   }
                SetCtrlVal (panel, PNL_BUSY, 0);
                break; //end case SAVE
        case PNL_RUN :
                SetCtrlVal (panel, PNL_BUSY, 1);
                GetCtrlVal (panel, PNL_VOLT, &voltios);
                GetCtrlVal (panel, PNL_AMP, &amper);
```

```
//Set voltage and current from panel
hp6xxxa_volt_curr (hp6030a, voltios, amper, 1);
//GetCtrlVal (panel, PNL_FREQ, &freq);
//GetCtrlVal (panel, PNL_AMPLIT, &ampl);
//frequency=freq*1000;
//Set amplitude and frequency from panel
//hp33120a_wf_config (waveform, 2, "", frequency, ampl, 0, 0.0, 50);
//Initialize the timer control
//GetCtrlVal (panel, PNL_TIMER1, &initial_time);
GetCtrlVal (panel, PNL_SAMPLING_INTERVAL, &sampling_interval);
SetCtrlAttribute (panel, PNL_TIMER1, ATTR_INTERVAL, sampling_interval+0.0);
GetCtrlVal (panel, PNL_NO_OF_SAMPLES, &no_of_samples);
actual_sample_no=0;
//Make the first measurement at time 0
//Make electrical measurements from power supply
GetCtrlVal (panel, PNL_MEASELECT, &elec_meas);
if (elec_meas==1) {
        hp6xxxa_read_output (hp6030a, 1, &measvolt, &meascurr);
        SetCtrlVal (panel, PNL_VOLTMEAS, measvolt);
        SetCtrlVal (panel, PNL_CURRMEAS, meascurr);
        voltms[actual_sample_no]=measvolt;
        currms[actual_sample_no]=meascurr;
        watts[actual_sample_no]=measvolt*meascurr;                    }
        //Make calibration for thermal measurement
        kei7001_opn_cls_ch_lst (1, 2, "");    //open all channels
        kei7001_opn_cls_ch_lst (1, 0, "2!1"); //close channel 1 from Slot 2
        Delay (0.01);
        ibwrt (keit2001, ":TEMP:RJUN1:RSEL REAL", 21);
        ibwrt (keit2001, ":TEMP:RJUN1:REAL:TCO 0.0002", 27);
        ibwrt (keit2001, ":TEMP:RJUN1:REAL:OFFS 0.05463", 29);
        ibwrt (keit2001, ":TEMP:RJUN1:ACQ", 15);
        Delay (0.10);
        //Make measuret for thermocouple 1 from channel 2
        GetCtrlVal (panel, PNL_T1, &status1);
        if (status1==1) {
                kei7001_opn_cls_ch_lst (1, 2, "");          //open all channels
                kei7001_opn_cls_ch_lst (1, 0, "2!25"); //close chan 25 from Slot 2
        kei2001_sing_meas (1, 2, 0, &meas1);  //take measurement
                SetCtrlVal (panel, PNL_INDICATOR_T1, meas1);
                tp1[actual_sample_no]=meas1;  //put temperature 1 in its array      }
        //Make measuret for thermocouple 2 from channel 3
        GetCtrlVal (panel, PNL_T2, &status2);
        if (status2==1) {
                kei7001_opn_cls_ch_lst (1, 2, "");          //open all channels
                kei7001_opn_cls_ch_lst (1, 0, "2!16"); //close chan 16 from Slot 2
                kei2001_sing_meas (1, 2, 0, &meas2);  //take measurement
                SetCtrlVal (panel, PNL_INDICATOR_T2, meas2);
                tp2[actual_sample_no]=meas2;  //put temperature 2 in its array      }
        //Make measuret for thermocouple 3 from channel 4
        GetCtrlVal (panel, PNL_T3, &status3);
        if (status3==1) {
                kei7001_opn_cls_ch_lst (1, 2, "");          //open all channels
                kei7001_opn_cls_ch_lst (1, 0, "2!26"); //close chan 26 from Slot 2
                kei2001_sing_meas (1, 2, 0, &meas3);  //take measurement
                SetCtrlVal (panel, PNL_INDICATOR_T3, meas3);
                tp3[actual_sample_no]=meas3;  //put temperature 3 in its array     }
        //Make measuret for thermocouple 4 from channel 5
        GetCtrlVal (panel, PNL_T4, &status4);
        if (status4==1) {
                kei7001_opn_cls_ch_lst (1, 2, "");          //open all channels
                kei7001_opn_cls_ch_lst (1, 0, "2!15"); //close chan 15 from Slot 2
        kei2001_sing_meas (1, 2, 0, &meas4);  //take measurement
                SetCtrlVal (panel, PNL_INDICATOR_T4, meas4);
                tp4[actual_sample_no]=meas4;  //put temperature 4 in its array     }
//Make measuret for thermocouple 5 from channel 6
        GetCtrlVal (panel, PNL_T5, &status5);
```

```
                if (status5==1) {
                        kei7001_opn_cls_ch_lst (1, 2, "");              //open all channels
                        kei7001_opn_cls_ch_lst (1, 0, "2!35"); //close chan 35 from Slot 2
                        kei2001_sing_meas (1, 2, 0, &meas5);  //take measurement
                        SetCtrlVal (panel, PNL_INDICATOR_T5, meas5);
                        tp5[actual_sample_no]=meas5;  //put temperature 5 in its array    }
                //Make measuret for thermocouple 5 from channel 6
                GetCtrlVal (panel, PNL_Tair, &statusair);
                if (statusair==1) {
                        kei7001_opn_cls_ch_lst (1, 2, "");              //open all channels
                        kei7001_opn_cls_ch_lst (1, 0, "2!6"); //close channel 6 from Slot 2
                        kei2001_sing_meas (1, 2, 0, &measair);  //take measurement
                        SetCtrlVal (panel, PNL_INDICATOR_Tair, measair);
                        tpair[actual_sample_no]=measair;                        }
                //Record the starting time according to the system clock
                GetSystemTime(&h_i,&m_i,&s_i);
                time_i = h_i*3600 + m_i*60 + s_i;

                //Record the time at which the actual first iteration sample was taken
                GetSystemTime(&h_a,&m_a,&s_a);
                time_a = h_a*3600 + m_a*60 + s_a;
                //Find out how much time has transcurred since the initial starting time
                time_t = time_a - time_i;
                time_meas[actual_sample_no] = time_t;
                SetCtrlVal (panel, PNL_MEASTIME, time_t+0.0);
                //SetCtrlVal (panel, PNL_MEASTIME, meastime);
                SetCtrlVal (panel, PNL_SAMPLE, actual_sample_no+1.0);
                actual_sample_no = actual_sample_no + 1;
                //Let the timer do the rest of the measurements
                SetCtrlAttribute (panel, PNL_TIMER1, ATTR_ENABLED,TRUE);
                break; //end case RUN
case PNL_RESET :
        SetCtrlVal (panel, PNL_BUSY, 1);
        SetCtrlVal (panel, PNL_VOLT, 0.00);
        SetCtrlVal (panel, PNL_AMP, 0.00);
        SetCtrlVal (panel, PNL_NO_OF_SAMPLES, 0.00);
        SetCtrlVal (panel, PNL_LEDT1, 0);
        SetCtrlVal (panel, PNL_T1, 0);
        SetCtrlVal (panel, PNL_INDICATOR_T1, 0.00);
        SetCtrlVal (panel, PNL_LEDT2, 0);
        SetCtrlVal (panel, PNL_T2, 0);
        SetCtrlVal (panel, PNL_INDICATOR_T2, 0.00);
        SetCtrlVal (panel, PNL_LEDT3, 0);
        SetCtrlVal (panel, PNL_T3, 0);
        SetCtrlVal (panel, PNL_INDICATOR_T3, 0.00);
        SetCtrlVal (panel, PNL_LEDT4, 0);
        SetCtrlVal (panel, PNL_T4, 0);
        SetCtrlVal (panel, PNL_INDICATOR_T4, 0.00);
        SetCtrlVal (panel, PNL_LEDT5, 0);
        SetCtrlVal (panel, PNL_T5, 0);
        SetCtrlVal (panel, PNL_INDICATOR_T5, 0.00);
        SetCtrlVal (panel, PNL_LEDTair, 0);
        SetCtrlVal (panel, PNL_Tair, 0);
        SetCtrlVal (panel, PNL_INDICATOR_Tair, 0.00);
        SetCtrlVal (panel, PNL_LEDELECT, 0);
        SetCtrlVal (panel, PNL_MEASELECT, 0);
        SetCtrlVal (panel, PNL_VOLTMEAS, 0.00);
        SetCtrlVal (panel, PNL_CURRMEAS, 0.00);
        SetCtrlVal (panel, PNL_MEASTIME, 0.00);
        Clear1D (tc1, pts);
        Clear1D (tc2, pts);
        Clear1D (tc3, pts);
        Clear1D (tc4, pts);
        Clear1D (tc5, pts);
        Clear1D (tcair, pts);
        Clear1D (time_array, pts);
```

```
                        Clear1D (volt, pts);
                        Clear1D (curr, pts);
                        Clear1D (pwr, pts);
                        DeleteGraphPlot (panel, PNL_CURVE, -1, 1);
                        DeleteGraphPlot (panel, PNL_CURVELECTRIC, -1, 1);
                        SetCtrlVal (panel, PNL_BUSY, 0);
                        break; //end case RESET
                case PNL_EXIT :
                        hp6xxxa_volt_curr (hp6030a, 0.000, 0.000, 1);
                        return;
                        break;      } //end switch (id)             } //end while (TRUE) } // end main

/*================================FUNCTIONS==================================*/
void ClearBuff (b)
char b[20];{
        int i;
        /*--------------------------------------------------------------------------------*/
        /*      Clear buffer in memory                                                   */
        /*-------------------------------------------------------------------------------- */
        for (i=0; (i<20); i++)
                b[i]=' ';                       } //end ClearBuff function
// When the RUN button is pressed, the timer is activated and the measurements are taken
int CVICALLBACK timer_activado (int panel, int control, int event,void *callbackData, int eventData1,
                                int eventData2)                         {
        switch (event)                  {
                case EVENT_TIMER_TICK:
                        if (actual_sample_no < no_of_samples){
                        //Make electrical measurements from power supply
                                GetCtrlVal (panel, PNL_MEASELECT, &elec_meas);
                                if (elec_meas==1) {
                                        hp6xxxa_read_output (hp6030a, 1, &measvolt, &meascurr);
                                        SetCtrlVal (panel, PNL_VOLTMEAS, measvolt);
                                        SetCtrlVal (panel, PNL_CURRMEAS, meascurr);
                                        voltms[actual_sample_no]=measvolt;
                                        currms[actual_sample_no]=meascurr;
                                        watts[actual_sample_no]=measvolt*meascurr;                      }
                                //Make calibration for thermal measurement
                                kei7001_opn_cls_ch_lst (1, 2, "");    //open all channels
                                kei7001_opn_cls_ch_lst (1, 0, "2!1"); //close channel 1 from Slot 2
                                Delay (0.01);
                                ibwrt (keit2001, ":TEMP:RJUN1:RSEL REAL", 21);
                                ibwrt (keit2001, ":TEMP:RJUN1:REAL:TCO 0.0002", 27);
                                ibwrt (keit2001, ":TEMP:RJUN1:REAL:OFFS 0.05463", 29);
                                ibwrt (keit2001, ":TEMP:RJUN1:ACQ", 15);
                                Delay (0.10);
                                //Make measuret for thermocouple 1 from channel 2
                                GetCtrlVal (panel, PNL_T1, &status1);
                                if (status1==1) {
                                        kei7001_opn_cls_ch_lst (1, 2, "");              //open all channels
                                        kei7001_opn_cls_ch_lst (1, 0, "2!25"); //close chan 25 from Slot 2
                                        kei2001_sing_meas (1, 2, 0, &meas1);  //take measurement
                                        SetCtrlVal (panel, PNL_INDICATOR_T1, meas1);
                                        tp1[actual_sample_no]=meas1;  //                        } //end if (status)
                                //Make measuret for thermocouple 2 from channel 3
                                GetCtrlVal (panel, PNL_T2, &status2);
                                if (status2==1) {
                                        kei7001_opn_cls_ch_lst (1, 2, "");              //open all channels
                                        kei7001_opn_cls_ch_lst (1, 0, "2!16"); //close chan 16 from Slot 2
                                        kei2001_sing_meas (1, 2, 0, &meas2);  //take measurement
                                        SetCtrlVal (panel, PNL_INDICATOR_T2, meas2);
                                        tp2[actual_sample_no]=meas2;            } //end if (status)
                                //Make measuret for thermocouple 3 from channel 4
                                GetCtrlVal (panel, PNL_T3, &status3);
                                if (status3==1) {
                                        kei7001_opn_cls_ch_lst (1, 2, "");              //open all channels
                                        kei7001_opn_cls_ch_lst (1, 0, "2!26"); //close chanl 26 from Slot 2
                                kei2001_sing_meas (1, 2, 0, &meas3);  //take measurement
                                SetCtrlVal (panel, PNL_INDICATOR_T3, meas3);
                                tp3[actual_sample_no]=meas3;                    } //end if (status)
                        //Make measuret for thermocouple 4 from channel 5
                        GetCtrlVal (panel, PNL_T4, &status4);
                        if (status4==1) {
                                kei7001_opn_cls_ch_lst (1, 2, "");              //open all channels
                                kei7001_opn_cls_ch_lst (1, 0, "2!15"); //close chan 15 from Slot 2
                                kei2001_sing_meas (1, 2, 0, &meas4);  //take measurement
                                SetCtrlVal (panel, PNL_INDICATOR_T4, meas4);
                                tp4[actual_sample_no]=meas4;                    } //end if (status)
                        //Make measuret for thermocouple 5 from channel 6
                        GetCtrlVal (panel, PNL_T5, &status5);
                        if (status5==1) {
                                kei7001_opn_cls_ch_lst (1, 2, "");              //open all channels
                                kei7001_opn_cls_ch_lst (1, 0, "2!35"); //close chan 35 from Slot 2
                                kei2001_sing_meas (1, 2, 0, &meas5);  //take measurement
                                SetCtrlVal (panel, PNL_INDICATOR_T5, meas5);
                                tp5[actual_sample_no]=meas5;                    } //end if (status)
                        //Make measuret for air thermocouple from channel 6
                        GetCtrlVal (panel, PNL_Tair, &statusair);
                        if (statusair==1) {
                                kei7001_opn_cls_ch_lst (1, 2, "");              //open all channels
                                kei7001_opn_cls_ch_lst (1, 0, "2!6"); //close channel 6 from Slot 2
                                kei2001_sing_meas (1, 2, 0, &measair);  //take measurement
                                SetCtrlVal (panel, PNL_INDICATOR_Tair, measair);
                                tpair[actual_sample_no]=measair;                        } //end if (status)
                                //Record the time at which the actual iteration samples were taken
                                GetSystemTime(&h_a,&m_a,&s_a);
                                time_a = h_a*3600 + m_a*60 + s_a;
                                time_t = time_a - time_i;
                                time_meas[actual_sample_no] = time_t;
                                //time_meas[points] = meastime;
                                SetCtrlVal (panel, PNL_MEASTIME, time_t+0.0);
                                //SetCtrlVal (panel, PNL_MEASTIME, meastime);
                                SetCtrlVal (panel, PNL_SAMPLE, actual_sample_no+1.0);
                                actual_sample_no = actual_sample_no + 1;   }
                //GetCtrlVal (panel, PNL_TIMER1, &current_time);
                        //} //end while (points)
                else{
                        pts=no_of_samples;
                        for (i=0; (i<pts); i++) {
                                tc1[i]=tp1[i];
                                tc2[i]=tp2[i];
                                tc3[i]=tp3[i];
                                tc4[i]=tp4[i];
                                tc5[i]=tp5[i];
                                tcair[i]=tpair[i];
                                time_array[i]=time_meas[i];
                                volt[i]=voltms[i];
                                curr[i]=currms[i];
                                pwr[i]=watts[i];                }
                        SetCtrlVal (panel, PNL_BUSY, 0);
                        hp6xxxa_volt_curr (hp6030a, 0.000, 0.000, 1);
                        SetCtrlAttribute (panel, PNL_TIMER1, ATTR_ENABLED,FALSE);      }
                break;                  }
        return 0;               }
```

APPENDIX D

HEAT SINK MODEL DESCRIPTION

A simple heat sink model can be based on thermal component networks [2, 3], where the lumped concept is used. In this case, an electrical-thermal analogy is used to elaborate the IPEM heat sink model. Heat transfer inside the heat sink is modeled from several thermal resistances and capacitances, as is shown in Figure D.1.

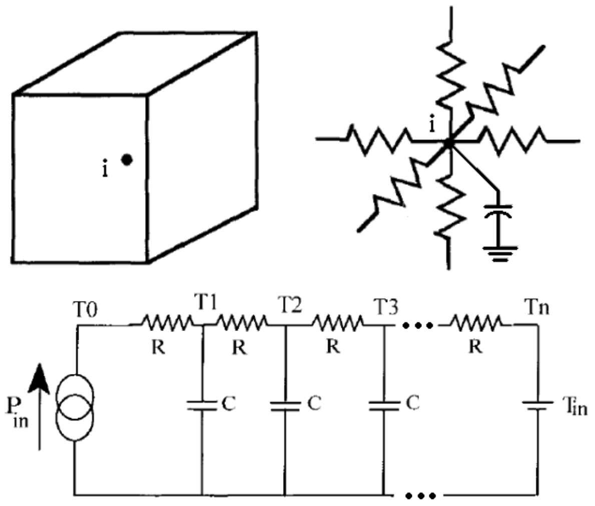


Figure D.1. Finite difference model as the thermal modeling approach.

Here, the heat diffusion equation is used to define the thermal behavior, which is discretized using finite difference methods (FDM). In general, the model considers a quasi-one-dimensional heat transfer process using the rectangular coordinates:

$$k\frac{\partial^2 T}{\partial s^2} = \rho C_p \frac{\partial T}{\partial t}$$

with boundary conditions,

$$-k\frac{\partial T}{\partial s} = h_1(T - T_\infty) \qquad (s = 0,\ \ t > 0)$$

$$k\frac{\partial T}{\partial s} = h_2(T_\infty - T) \qquad (s = L,\ \ t > 0)$$

According to IPEM-LTCM model is assumed a natural convection using the heat transfer coefficient given by Hefner *et al.* in [1-4]:

$$h_i = 4.84x10^{-4}\frac{A_{fin}}{w_{fin}^{0.35}}$$

where A_{fin} is the total heat sink fin area, and w_{fin} is the heat sink fin height (or orientation parameter). Finally, the initial condition is given by,

$$T(s,0) = F(s) \qquad (0 < s < L)$$

For various symmetry conditions, the heat equation for the rectangular coordinate system with x- and y-axis symmetry is

$$A\frac{\partial}{\partial z}\left[k\left(\frac{\partial T}{\partial z}\right)\right] = A\rho C_p\frac{\partial T}{\partial t}$$

This equation can be discretized into a finite number of first-order ordinary time-dependent differential equations of the form

$$\frac{T_{i+1} - T_i}{R_{i,i+1}} - \frac{T_i - T_{i-1}}{R_{i-1,i}} = \frac{dH_i}{dt}$$

where

$$H_i = C_i T_i$$

$$C_i = \rho_i C_{pi} V_i$$

$$R_i = \frac{z_{i+1} - z_i}{A_i k_i} \qquad \text{(conduction case)}$$

$$R_i = \frac{1}{h_i (T_i - T_\infty)^{0.35}} \qquad \text{(convection case)}$$

In the discretization process of the heat diffusion equation, it is assumed that the temperature gradient and thermal conductivity do not vary substantially between adjacent grid points. The model accuracy is determined by the number and location of the nodes within the heat sink.

In general terms, the heat sink is divided in many nodes, each node represent a volume (in this case, all node are cubes, because the real heat sink can be represented by many of this form), in others words, a lumped, as is observed in the Figure D.2. Each lumped inside the heat sink is defined by the discretized equation (using FDM). In the case of the surface nodes, these nodes include both conduction and convection resistance, according to previous description of such resistances.

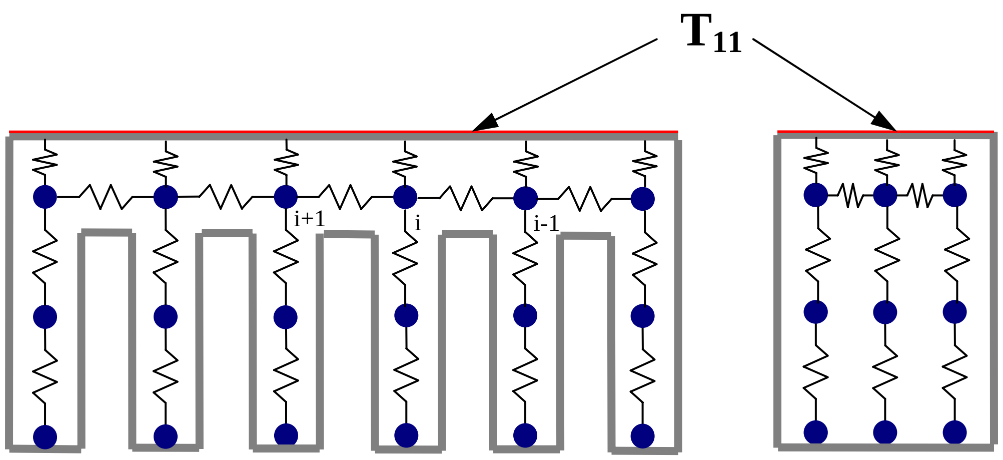
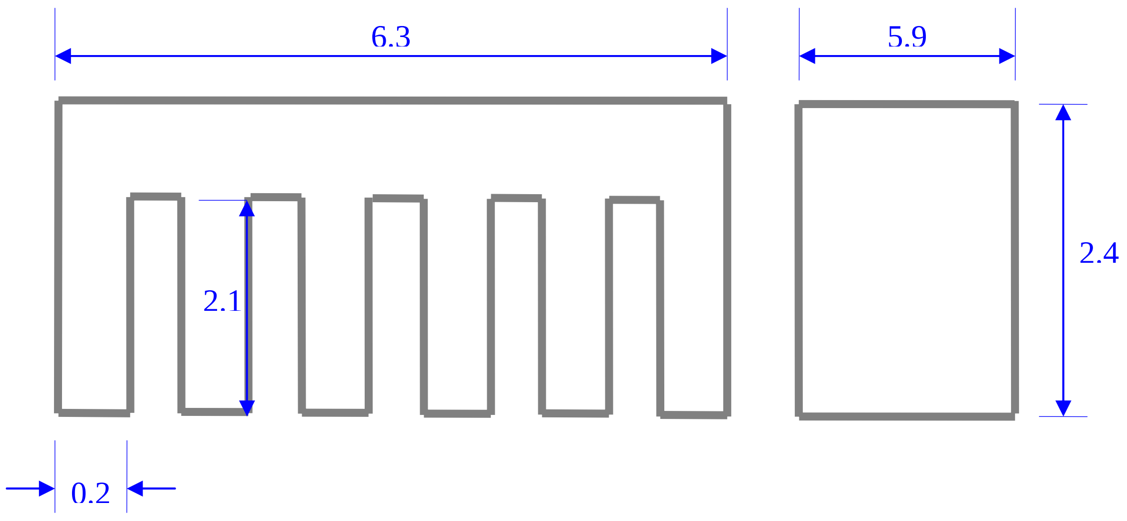


Figure D.2. Heat sink’s grid used in the analysis.



All dimensions are in centimeters.
The heat sink has eight fins.

Figure D.3. Heat sink’s dimensions.

APPENDIX E

FULL C++/CVI CODE OF THE THERMAL TRANSIENT PROGRAM DEVELOPED FOR FAST THERMAL RESPONSE EXPERIMENT

```
#define  TRUE    1
#define  FALSE  0
#define  ON        1
#define  OFF      0
#define  VV        2
#define  XY1      1
#define  XY2      2
#define  XY3      3
#define  XY4      4
/* Used in selecting curve to save                              */
#define  ALL_SC   0   /* Save all scope channel              */
#define  DISP_SC  1   /* Default: scope channel displayed               */
#define  TRAN      2   /* Save transient                      */
/* Used in selecting waveform input source (load function)               */
#define  SCP          0   /* Default: TDS644A                   */
#define  SCPFILE  1   /* Oscilloscope File                    */
#define  TRAFILE 2   /* Transient  File                      */
#define  SAM         0
#define  ENV         1
#define  AVE         2
/*Used in selecting curve to display                                     */
#define  SCPCURV  0   /* Default: Scope loaded                  */
#define  TRANCURV 1   /* Transient file loaded                  */
/*Used in selecting mode transient                                       */
#define  TRUN     0   /* Default: Truncated transient        */
#define  EXTR     1   /* Extrapolated transient              */
/*=======================FUNCTION DECLARATION========================*/
void Makechan (double[], double, int);
void Makexy (double[],double[],int,double[],double[]);
void CalcWaveforms (int,int, int, int, int, int, int, int, int);
void Makefile (int,int,int,int,int,int,int,double[],double[],double[],double[],double, double[], double[]);
void ClearBuff(char[]);
/*=======================VARIABLE DECLARATION========================*/
static double wavein1[2000],wavein2[2000],wavein3[2000],wavein4[2000];
static double wave1[2000],wave2[2000],wave3[2000],wave4[2000];
static double wave1s[2000],wave2s[2000],wave3s[2000],wave4s[2000],wavex[2000],wavey[2000];
static double wavetime[2000],timescp[2000],voltscp[2000],tempscp[2000];
static double voltcal [200],tempcal [200],volt_igbt[200],temp_igbt[200];
static double voltscp_max,voltscp_min, voltcal_max,voltcal_min,c_strt,x_incr, x_strt,ch1_scale;
static char filin[512];
static int tk644,cnt,tek644,count,err,pts,hp6035a;
/*===========================MAIN PROGRAM===========================*/
main() {
double difference,sfr1,sfr2,sfr3,sfr4,cvxmult,cvymult,cvamp,cvvolt,slope,intercept,mse,output[2000];
double voltios, amper, set_temp;
char bf[21],aqmode[21],str[320],colon[2],ch[2],str2[21];
char buf[21];
int c,calpoints,indice,ij,i,j,k,larg,flhand,filout,saval,x,y,loader,display,option,id,panel,handle;
int chan1,chan2,chan3,chan4,chan5,cvpts,samp,acmode,envp,fhand;
pts = 2000;
panel = LoadPanel (0, "transient.uir", SCOPE);
if (panel < 0){
  FmtOut("Unable to load the required panel from the resource file.\n");
  return;                    }
DisplayPanel(panel);
//Initialize Power Supply instrument
hp6xxxa_init ("GPIB::8::INSTR", VI_OFF, VI_OFF, 4, &hp6035a);
//Initialize Oscilloscope
tk644 = ibdev (0, 9, NO_SAD, T3s, 1, 0);
```

```
tek644a_init (9, 0, 0, &tek644);
tek644a_wvfm_acquisition (tek644, 1, &cnt);
ibwrt (tk644, "HOR:RECO 2000",13);
/* Determine the current acqusition mode.        */
ibwrt (tk644, "ACQ:MOD?",8);
ClearBuff(aqmode);
ibrd (tk644,aqmode,10);
if (aqmode[0]=='S')acmode=SAM;
if (aqmode[0]=='E')acmode=ENV;
if (aqmode[0]=='A')acmode=AVE;
switch (acmode) {
                case SAM :
                  SetCtrlIndex (panel, SCOPE_ACQMODE, SAM);
                  break;
                case ENV :
                  SetCtrlIndex (panel, SCOPE_ACQMODE, ENV);
/* Read the number of envelopes being taken        */
      ibwrt (tk644, "ACQ:NUME?",9);
      ibrd (tk644,bf,15);
      Fmt (&envp, "%i<%s",bf);
      SetCtrlVal (panel, SCOPE_SAMPS, envp);
      break;
    case AVE :
      SetCtrlIndex (panel, SCOPE_ACQMODE, AVE);
/* Read the number of samples being taken        */
      ibwrt (tk644, "ACQ:NUMAV?",10);
      ibrd (tk644,bf,15);
      Fmt (&samp, "%i<%s",bf);
      SetCtrlVal (panel, SCOPE_SAMPS, samp);
      break;  }
while (TRUE){
  GetUserEvent (TRUE, &handle, &id);
  switch (id) {
                case SCOPE_ST :
                  SetCtrlVal (panel, SCOPE_WORK, 1);
      GetCtrlVal(panel, SCOPE_VOLT_SUPPLY, &voltios);
      GetCtrlVal(panel, SCOPE_AMP_SUPPLY, &amper);
      //Set Voltage and Current from panel
      hp6xxxa_volt_curr (hp6035a, voltios, amper, 1);
                  //Set Oscilloscope Mode from panel
                  err = GetCtrlVal (panel, SCOPE_ACQMODE, &acmode);
                  err = GetCtrlVal (panel, SCOPE_SAMPS,&ij);
                  switch (acmode) {
                          case SAM :
                            tek644a_acquisition_setup (tek644, 1, ij, 1);
                            tek644a_wvfm_acquisition (tek644, 1, &err);
                            break;
                          case ENV :
                            tek644a_acquisition_setup (tek644, 2, ij, 1);
                            tek644a_wvfm_acquisition (tek644, 1, &err);
                            break;
                          case AVE :
                            tek644a_acquisition_setup (tek644, 3, ij, 1);
                            tek644a_wvfm_acquisition (tek644, 1, &err);
                            break;                    }
                  SetCtrlVal(panel,SCOPE_WORK,0);
                  break;
                // Load oscilloscope data displayed, an oscilloscope data file  or a transient data file
                case SCOPE_LOAD :
                  SetCtrlVal(panel,SCOPE_WORK,1);
                  GetCtrlVal(panel,SCOPE_LD,&loader);
                  switch (loader) {
                          //Load oscilloscope data displayed
                          case SCP :
                            /* Initialize the tds644 A/C/D Digitizing Oscilloscope */
                            /* Determine the current acquisition mode.        */
```

```
            //Stop the acquisition for data transfert.
            tek644a_wvfm_acquisition (tek644, 0,&count );
            ibwrt (tk644, "ACQ:MOD?",8);
            ClearBuff(aqmode);
            ibrd (tk644,aqmode,10);
            if (aqmode[0]=='S')acmode=SAM;
            if (aqmode[0]=='E')acmode=ENV;
            if (aqmode[0]=='A')acmode=AVE;
            switch (acmode) {
              case SAM :
                SetCtrlIndex (panel, SCOPE_ACQMODE, SAM);
                break;
              case ENV :
                SetCtrlIndex (panel, SCOPE_ACQMODE, ENV);
                // Read the number of envelopes being taken
                ibwrt (tk644, "ACQ:NUME?",9);
                ibrd (tk644,bf,15);
                Fmt (&envp, "%i<%s",bf);
                SetCtrlVal (panel, SCOPE_SAMPS, envp);
                break;
              case AVE :
                SetCtrlIndex (panel, SCOPE_ACQMODE, AVE);
                // Read the number of samples being taken
                ibwrt (tk644, "ACQ:NUMAV?",10);
                ibrd (tk644,bf,15);
                Fmt (&samp, "%i<%s",bf);
                SetCtrlVal (panel, SCOPE_SAMPS, samp);
                break;              }
            // Check if each channel is on, and get waveform from each channel
            ibwrt (tk644, "SEL:CH1?", 8);
            ibrd (tk644,bf,10);
            Fmt (&k, "%i<%s",bf);
            if (k != 0)
                tek644a_read_wvfm_array (tek644, 1, 0, 1, 2000, wavein1, &pts, &x_strt, &x_incr);
            ibwrt (tk644, "SEL:CH2?", 8);
            ibrd (tk644,bf,10);
            err = Fmt (&k, "%i<%s",bf);
            if (k != 0)
                tek644a_read_wvfm_array (tek644, 2, 0, 1, 2000, wavein2, &pts,&x_strt, &x_incr);
            ibwrt (tk644, "SEL:CH3?", 8);
            ibrd (tk644,bf,10);
            Fmt (&k, "%i<%s",bf);
            if (k != 0)
                tek644a_read_wvfm_array (tek644, 3, 0, 1, 2000, wavein3, &pts, &x_strt, &x_incr);
            ibwrt (tk644, "SEL:CH4?", 8);
            ibrd (tk644,bf,10);
            Fmt (&k, "%i<%s",bf);
            if (k != 0)
                tek644a_read_wvfm_array (tek644, 4, 0, 1, 2000, wavein4, &pts, &x_strt, &x_incr);
            tek644a_wvfm_acquisition (tek644, 1,&count );//Restart Scope after data transfert.
            for (i=0; (i<pts); i++) {
                 wavetime[i]=i*x_incr;
                 wave1[i]=wavein1[i];
                 wave2[i]=wavein2[i];
                 wave3[i]=wavein3[i];
                 wave4[i]=wavein4[i];                     }
            break;
    //Load an oscilloscope data file, add Popup file selection from oscilloscope files
    case SCPFILE:
            err = FileSelectPopup ("Last_data", "*.dat", "", "Load Window",
                                         VAL_LOAD_BUTTON, 0, 0, 1, 1, filin);
            if (err!=0){
                flhand = OpenFile (filin, 1, 2, 1);
                ClearBuff(buf);
                ScanFile (flhand, "%s>%i", &c);
                ScanFile (flhand, "%s>%i", &pts);
```

```
                        Clear1D(wavein1,pts);
                        Clear1D(wavein2,pts);
                        Clear1D(wavein3,pts);
                        Clear1D(wavein4,pts);
                        for (i=0; (i<pts);i++) {
                             for (j=0; (j<c);j++) {
                                  switch (j) {
                                    case 0 :
                                          ScanFile (flhand, "%s>%f", &wavetime[i]);
                                          break;
                                        case 1 :
                                          ScanFile (flhand, "%s>%f", &wavein1[i]);
                                      break;
                                    case 2 :
                                          ScanFile (flhand, "%s>%f", &wavein2[i]);
                                          break;
                                        case 3 :
                                          ScanFile (flhand, "%s>%f", &wavein3[i]);
                                      break;
                                    case 4 :
                                          ScanFile (flhand, "%s>%f", &wavein4[i]);
                                          break;                    }                    }                    }
                        for (i=0; (i<pts); i++) {
                             wave1[i]=wavein1[i];
                             wave2[i]=wavein2[i];
                             wave3[i]=wavein3[i];
                             wave4[i]=wavein4[i];          }
                        err = CloseFile (flhand);      }
                    break;
                  //Load a transient data file, add Popup file selection from transient files
   case TRAFILE :
     err=FileSelectPopup ("newfile", "*.dat", "", "Load Window", VAL_LOAD_BUTTON,
                                      0, 0, 1, 1, filin);
     if (err!=0){
         flhand = OpenFile(filin, 1, 2, 1);
         ClearBuff(buf);
     ScanFile(flhand, "%s>%i", &pts);
     Clear1D(timescp, pts);
     Clear1D(tempscp, pts);
         for (i=0; (i<pts);i++) {
                  ScanFile(flhand, "%s>%f", &timescp[i]);
           ScanFile(flhand, "%s>%f", &tempscp[i]);              }
 err = CloseFile (flhand);           }
 break;       } /* end switch (loader) */
  SetCtrlVal(panel,SCOPE_WORK,0);
  break;
case SCOPE_DISP :
  SetCtrlVal(panel,SCOPE_WORK,1);
  GetCtrlVal(panel,SCOPE_DP,&display);
  switch (display)  {
    //Display curve from oscilloscope
    case SCPCURV :
      //Query the panel for each channel selection and calculate the waveform for each channel
      GetCtrlVal(panel,SCOPE_CH1,&chan1);
      GetCtrlVal(panel,SCOPE_CH2,&chan2);
      GetCtrlVal(panel,SCOPE_CH3,&chan3);
      GetCtrlVal(panel,SCOPE_CH4,&chan4);
      GetCtrlVal(panel,SCOPE_CH5,&chan5);
      GetCtrlVal(panel,SCOPE_CH5X,&x);
      GetCtrlVal(panel,SCOPE_CH5Y,&y);
      CalcWaveforms(panel,chan1,chan2,chan3,chan4,chan5,x,y,pts);
      //Clear the graph of its current plots. Plot the waveform selected for each channel
      //Light the LED for each active channel
      DeleteGraphPlot(panel,SCOPE_GRAPH, -1, 1);
      if (chan5 != OFF) {
                        if (y==1)
```

```
                                        err = SetCtrlVal(panel, SCOPE_Y, " Channel    1 ");
                                if (y==2)
                                        err = SetCtrlVal(panel, SCOPE_Y, " Channel    2 ");
                                if (y==3)
                                        err = SetCtrlVal(panel, SCOPE_Y, " Channel    3 ");
                                if (y==4)
                                      err = SetCtrlVal(panel, SCOPE_Y, " Channel    4 ");
                                 if (x==1)
                                      err = SetCtrlVal(panel, SCOPE_X, "Channel  1   ");
                                 if (x==2)
                                      err = SetCtrlVal(panel, SCOPE_X, "Channel  2   ");
                                if (x==3)
                                      err = SetCtrlVal(panel, SCOPE_X, "Channel  3   ");
                                if (x==4)
                                      err = SetCtrlVal(panel, SCOPE_X, "Channel  4   ");
                                PlotXY(panel,SCOPE_GRAPH,wavex,wavey,pts,VAL_DOUBLE,VAL_DOUBLE,
                               VAL_THIN_LINE,VAL_EMPTY_SQUARE, VAL_SOLID,1, DOSColorToRGB   (13));      }
                      if (chan1 != OFF)
                      PlotXY(panel,SCOPE_GRAPH,wavetime,wave1,pts,VAL_DOUBLE,VAL_DOUBLE,VAL_THIN_LINE,VAL_E
                      MPTY_SQUARE, VAL_SOLID,1,DOSColorToRGB (15));
                      if (chan2 != OFF)
                      PlotXY(panel,SCOPE_GRAPH,wavetime,wave2,pts,VAL_DOUBLE,VAL_DOUBLE,VAL_THIN_LINE,VAL_E
                      MPTY_SQUARE, VAL_SOLID,1,DOSColorToRGB (2));
                      if (chan3 != OFF)
                      PlotXY(panel,SCOPE_GRAPH,wavetime,wave3,pts,VAL_DOUBLE,VAL_DOUBLE,VAL_THIN_LINE,VAL_E
                      MPTY_SQUARE, VAL_SOLID,1,DOSColorToRGB (14));
                      if (chan4 != OFF)
                      PlotXY(panel,SCOPE_GRAPH,wavetime,wave4,pts,VAL_DOUBLE,VAL_DOUBLE,VAL_THIN_LINE,VAL_E
                      MPTY_SQUARE, VAL_SOLID,1,DOSColorToRGB (9));
                     if (chan5 == OFF) {
                        SetCtrlVal(panel, SCOPE_Y, "Voltage V  ");
                        SetCtrlVal(panel, SCOPE_X, " Time (s)    ");             }
                     SetCtrlVal(panel,SCOPE_STATE1,chan1);
                     SetCtrlVal(panel,SCOPE_STATE2,chan2);
                     SetCtrlVal(panel,SCOPE_STATE3,chan3);
                     SetCtrlVal(panel,SCOPE_STATE4,chan4);
                     SetCtrlVal(panel,SCOPE_STATE5,chan5);
                     SetCtrlVal(panel,SCOPE_WORK,0);
                     break;
                    //Display curve from transient file
                    case TRANCURV :
                     DeleteGraphPlot (panel, SCOPE_CURVE, -1, 1);
                     PlotXY (panel, SCOPE_CURVE, timescp, tempscp, pts, VAL_DOUBLE,VAL_DOUBLE,
                                VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,VAL_WHITE);
                     break;    }//end switch (display)
                   SetCtrlVal(panel,SCOPE_WORK,0);
                   break;
    case SCOPE_SAVE :
                   SetCtrlVal(panel,SCOPE_WORK,1);
                   GetCtrlVal(panel,SCOPE_CURV,&saval);
                   GetCtrlVal(panel,SCOPE_SF1,&sfr1);
                   GetCtrlVal(panel,SCOPE_SF2,&sfr2);
                   GetCtrlVal(panel,SCOPE_SF3,&sfr3);
                   GetCtrlVal(panel,SCOPE_SF4,&sfr4);
                   for (i=0; (i<pts); i++) {
                              wave1[i] = wavein1[i] * sfr1;
                              wave2[i] = wavein2[i] * sfr2;
                              wave3[i] = wavein3[i] * sfr3;
                              wave4[i] = wavein4[i] * sfr4;
                              }
                   //Save data displayed from one channel, data all channels or data displayed from transient curve
Makefile(pts,chan1,chan2,chan3,chan4,saval,panel,wave1,wave2,wave3,wave4,x_incr,timescp,tempscp);
                   SetCtrlVal(panel,SCOPE_WORK,0);
                   break;
                  //Acquire data from oscilloscope file
   case SCOPE_OSCIL :
```

```
 SetCtrlVal(panel,SCOPE_WORK,1);
 err=FileSelectPopup("Oscilloscope", "*.dat", "", "Load Window", VAL_LOAD_BUTTON,0, 0, 1, 1,
                                   filin);
 if (err!=0){
    flhand = OpenFile(filin, 1, 2, 1);
    ClearBuff(buf);
    ScanFile(flhand, "%s>%i", &c);
    ScanFile(flhand, "%s>%i", &pts);
    Clear1D(wavein1,pts);
    Clear1D(wavein2,pts);
    Clear1D(wavein3,pts);
    Clear1D(wavein4,pts);
    for (i=0; (i<pts);i++){
                   for (j=0; (j<c);j++) {
                       switch (j) {
                                   case 0 :
                                           ScanFile(flhand, "%s>%f", &wavetime[i]);
                                           timescp[i] = wavetime[i];
                                           break;
                                      case 1 :
                                           ScanFile(flhand, "%s>%f", &wavein1[i]);
                                           voltscp[i] = wavein1[i];
                                           break;
                                    case 2 :
                                            ScanFile(flhand, "%s>%f", &wavein2[i]);
                                            voltscp[i] = wavein2[i];
                                            break;
                                    case 3 :
                                            ScanFile(flhand, "%s>%f", &wavein3[i]);
                                            voltscp[i] = wavein3[i];
                                            break;
                                    case 4 :
                                            ScanFile (flhand, "%s>%f", &wavein4[i]);
                                            voltscp[i] = wavein4[i];
                                            break;                    }           }          }
     err = CloseFile(flhand);         }
 SetCtrlVal(panel,SCOPE_WORK,0);
 break;
//Acquire data from calibration file
case SCOPE_IGBTCALIB :
 SetCtrlVal(panel,SCOPE_WORK,1);
 err=FileSelectPopup("calibra", "*.tsp", "", "Load Window", VAL_LOAD_BUTTON,0, 0, 1, 1, filin);
 if (err!=0){
    flhand = OpenFile(filin, 1, 2, 1);
    ClearBuff(buf);
    ScanFile(flhand, "%s>%i", &calpoints);
    for (i=0; (i<calpoints);i++) {
         ScanFile(flhand, "%s>%f", &temp_igbt[i]);
         tempcal[i] = temp_igbt[i];
         ScanFile (flhand, "%s>%f", &volt_igbt[i]);
         voltcal[i] = volt_igbt[i];             }
         err = CloseFile (flhand);         }
 SetCtrlVal(panel,SCOPE_WORK,0);
 break;
// Compare and change the data from oscilloscope and calibration to transient. There are two ways to
   obtain transient: Truncate and Extrapolation
case SCOPE_CHANGE :
 SetCtrlVal(panel,SCOPE_WORK,1);
 //Compare and obtain the maximun and minimun values from oscilloscope data with calibration data
 LinFit (tempcal, voltcal, calpoints, output, &slope, &intercept, &mse); //With this command the slope
                                                                                  and intercept is obtained
 GetCtrlVal (panel, SCOPE_OPTION, &option);
 //Define transient way
 switch (option) {
         // The truncated transient way
            case TRUN :
                for (i=0; (i<pts);i++) {
                        indice = 0;
                        difference = 10000000.0;
                        for (j=0; (j<calpoints);j++) {
                           if (fabs (voltscp[i] - voltcal[j])<difference) {
                                 indice = j;
                                 difference = fabs(voltscp[i] - voltcal[j]);      }                    }
                        //Obtain the correct temperature for each oscilloscope voltage
                        if ((voltscp[i]>voltcal[0]) && (voltscp[i]<voltcal[calpoints-1]))
                          tempscp[i] = ((((voltscp[i] - voltcal[indice-1])/(voltcal[indice+1] - voltcal[indice-1]))*
                                    ((tempcal[indice+1]) - (tempcal[indice-1]))) + tempcal[indice-1]);
                        else
                          tempscp[i] = tempcal[indice];              }
                DeleteGraphPlot(panel, SCOPE_CURVE, -1, 1);
                PlotXY (panel, SCOPE_CURVE, timescp, tempscp, pts, VAL_DOUBLE, VAL_DOUBLE,
                        VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_DK_RED);
                break;
             // The Extrapolated transient way
             case EXTR :
              for (i=0; (i<pts);i++) {
              //Inside calibration range
              if ((voltscp[i]>voltcal[0]) && (voltscp[i]<voltcal[calpoints-1])) {
                indice = 0;
                difference = 10000000.0;
                for (j=1; (j<calpoints);j++) {
                   if (fabs (voltscp[i] - voltcal[j])<difference) {
                     indice = j;
                     difference = fabs(voltscp[i] - voltcal[j]);        }                        }
                  //Obtain the correct temperature for each oscilloscope voltage
                  if ((voltscp[i]>voltcal[0]) && (voltscp[i]<voltcal[calpoints-1])) {
                      tempscp[i] = ((((voltscp[i] - voltcal[indice-1])/(voltcal[indice+1] - voltcal[indice-1]))*
                               ((tempcal[indice+1]) - (tempcal[indice-1]))) + tempcal[indice-1]);      }      }
                //Outside calibration range (extrapolation)
                else
                    tempscp[i] = ((voltscp[i] - intercept) / (slope));          }
             DeleteGraphPlot(panel, SCOPE_CURVE, -1, 1);
             PlotXY (panel, SCOPE_CURVE, timescp, tempscp, pts, VAL_DOUBLE, VAL_DOUBLE,
                     VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_DK_RED);
             break;                    }/* end switch (option) */
           SetCtrlVal(panel,SCOPE_WORK,0);
           break;
   case SCOPE_QUIT :
     return;
     break;    }/* end switch (id) */                }/* end switch while */             }/* end main */
/*==========================CALCULATE WAVEFORMS=======================*/
void CalcWaveforms (pan,chan1, chan2, chan3, chan4, chan5,cx,cy,pts)
int pan,pts,chan1,chan2,chan3,chan4,chan5,cx,cy;                          {
  int i;
  double s1,s2,s3,s4;
  /*--------------------------------------------------------------------*/
  /*  Determine if channel 1 is to dislplay or has been              */
  /*  turned off.                                                     */
  /*--------------------------------------------------------------------*/
  GetCtrlVal(pan,SCOPE_SF1,&s1);
  GetCtrlVal(pan,SCOPE_SF2,&s2);
  GetCtrlVal(pan,SCOPE_SF3,&s3);
  GetCtrlVal(pan,SCOPE_SF4,&s4);
  for (i=0; (i<pts); i++)    {
    wave1s[i]=wavein1[i]*s1;
    wave2s[i]=wavein2[i]*s2;
    wave3s[i]=wavein3[i]*s3;
    wave4s[i]=wavein4[i]*s4;    }
  switch (chan1) {
    case ON :
       for (i=0; (i<pts); i++)         {
         wave1[i]=wave1s[i];        }
```

```
      Makechan(wave1,0,pts);
      break;
  case OFF :
      Clear1D(wave1,pts);
      break;}
/*-----------------------------------------------------------------------*/
/*  Determine if channel 2 is to dislplay or has been            */
/*  turned off.                                                    */
/*-----------------------------------------------------------------------*/
switch (chan2) {
  case ON :
      for (i=0; (i<pts); i++)         {
             wave2[i]=wave2s[i];    }
       Makechan(wave2,0,pts);
       break;
  case OFF :
        Clear1D(wave2,pts);
        break;                  }
/*-----------------------------------------------------------------------*/
/*  Determine if channel 3 is to dislplay or has bee             */
/*  turned off.                                                    */
/*-----------------------------------------------------------------------*/
switch (chan3) {
  case ON :
      for (i=0; (i<pts); i++)         {
         wave3[i]=wave3s[i];        }
      Makechan(wave3,0,pts);
      break;
  case OFF :
      Clear1D(wave3,pts);
       break;                  }
/*-----------------------------------------------------------------------*/
/*  Determine if channel 4 is to dislplay or has been            */
/*  turned off.                                                    */
/*-----------------------------------------------------------------------*/
switch (chan4) {
  case ON :
      for (i=0; (i<pts); i++)         {
         wave4[i]=wave4s[i];         }
      Makechan(wave4,0,pts);
      break;
  case OFF :
      Clear1D(wave4,pts);
      break;                  }
/*-----------------------------------------------------------------------*/
/*  Determine if channel 5 is to dislplay                        */
/*  or if it has been turned off.                                */
/*-----------------------------------------------------------------------*/
switch (chan5) {
  case ON :
    switch (cx) {
      case XY1 :
        Makexy(wave1s,wavey,pts,wavex,wavey);
      break;
      case XY2 :
        Makexy(wave2s,wavey,pts,wavex,wavey);
      break;
      case XY3 :
        Makexy(wave3s,wavey,pts,wavex,wavey);
      break;
      case XY4 :
        Makexy(wave4s,wavey,pts,wavex,wavey);
      break;        }
    switch (cy) {
      case XY1 :
        Makexy(wavex,wave1s,pts,wavex,wavey);
        break;
        case XY2 :
          Makexy(wavex,wave2s,pts,wavex,wavey);
        break;
        case XY3 :
          Makexy(wavex,wave3s,pts,wavex,wavey);
        break;
        case XY4 :
          Makexy(wavex,wave4s,pts,wavex,wavey);
        break;        } /* end switch (cy) */
      break;
    case OFF :
      Clear1D(wavey,pts);
      Clear1D(wavex,pts);
      break;      }                          }
/*===================MAKE scaled WAVE====================*/
void Makechan (waveform, offset,pt)
double waveform[];
double offset;
int pt;                         {
  int i;
  double scale_val;
  scale_val = 1.0;
  for (i=0; (i < pt); i++)
                  waveform[i] = offset + waveform[i]*scale_val;             }
  void Makexy (waveform1,waveform2,ps,waveform3,waveform4)
  double waveform1[],waveform2[],waveform3[],waveform4[];
  int ps;                       {
  int i;
  for (i=0; (i<ps); i++)    {
     waveform3[i]=waveform1[i];
     waveform4[i]=waveform2[i];     }                            }
  //Making the function to save the several options
  void Makefile (pt,c1,c2,c3,c4,saver,pan,wavef1,wavef2,wavef3,wavef4,incr,timescp,tempscp)
  int pt,c1,c2,c3,c4,saver,pan;
  double wavef1[2000],wavef2[2000],wavef3[2000],wavef4[2000];
  double incr,timescp[2000],tempscp[2000];                   {
  int i,fhand,c;
  char filin[512];
  char buff[20];
  double rl;
  err = FileSelectPopup ("savedata", "*.dat", "", "Save Window", VAL_SAVE_BUTTON, 0, 0, 1, 1, filin);
  if (err!=0){
    fhand = OpenFile (filin, 2, 0, 1);
    switch (saver)  {
          case ALL_SC :
                  ClearBuff(buff);
                  c=5;
                  err = Fmt (buff, "%s<%i",c);
                  err = WriteLine (fhand,buff, 6);
                  ClearBuff(buff);
                  err = Fmt (buff, "%s<%i",pt);
                  err = WriteLine (fhand,buff, 6);
                  for (i=0; (i<pt); i++)        {
                         ClearBuff(buff);
                         rl = i*incr;
                         err = Fmt (buff, "%s<%f",rl);
                         err = WriteFile (fhand,buff,15);
                         ClearBuff(buff);
                         err = Fmt (buff, "%s<%f",wavef1[i]);
                         err = WriteFile (fhand,buff,15);
                         ClearBuff(buff);
                         err = Fmt (buff, "%s<%f",wavef2[i]);
                         err = WriteFile (fhand,buff,15);
                         ClearBuff(buff);
                         err = Fmt (buff, "%s<%f",wavef3[i]);
```

```
                        err = WriteFile (fhand,buff,15);
                        ClearBuff(buff);
                        err = Fmt (buff, "%s<%f",wavef4[i]);
                        err = WriteLine (fhand,buff,15);       }
                break;
        case DISP_SC :
                c=1;
                if (c1 != OFF)
                   c=c+1;
             if (c2 != OFF)
                    c=c+1;
             if (c3 != OFF)
                c=c+1;
             if (c4 != OFF)
                c=c+1;
             if (pt == 0)
                c=0;
             ClearBuff(buff);
             err = Fmt (buff, "%s<%i",c);
             err = WriteLine (fhand,buff, 6);
             ClearBuff(buff);
             err = Fmt (buff, "%s<%i",pt);
             err = WriteLine (fhand,buff, 6);
             for (i=0; (i<pt); i++)       {
                        ClearBuff(buff);
                        rl=i*incr;
                        err = Fmt (buff, "%s<%f",rl);
                   err = WriteFile (fhand,buff,15);
                   if (c1 != OFF) {
                                ClearBuff(buff);
                                err = Fmt (buff, "%s<%f",wavef1[i]);
                                err = WriteFile (fhand,buff,15);          }
                        if (c2 != OFF) {
                                ClearBuff(buff);
                                err = Fmt (buff, "%s<%f",wavef2[i]);
                                err = WriteFile (fhand,buff,15);          }
                        if (c3 != OFF) {
                                ClearBuff(buff);
                                err = Fmt (buff, "%s<%f",wavef3[i]);
                                err = WriteFile (fhand,buff,15);          }
                        if (c4 != OFF) {
                                ClearBuff(buff);
                                err = Fmt (buff, "%s<%f",wavef4[i]);
                                err = WriteFile (fhand,buff,15);          }
                       WriteLine (fhand," ",1);         }
                break;
     case TRAN :
       ClearBuff(buff);
       err = Fmt(buff, "%s<%i",pt);
       err = WriteLine(fhand,buff,6);
       for (i=0; (i<pt); i++) {
           ClearBuff(buff);
           err = Fmt(buff, "%s<%f",timescp[i]);
           err = WriteFile(fhand,buff,15);
           ClearBuff(buff);
           err = Fmt(buff, "%s<%f",tempscp[i]);
           err = WriteLine(fhand,buff,15);           }
       break;   } /* end switch (saver) */
  CloseFile (fhand);     }                }
void ClearBuff(b)
char b[20];{
int i;
for (i=0; (i<20); i++)
  b[i]=' ';                     }
```

APPENDIX F

FULL MAST CODE OF THE THERMAL TEMPLATE DEVELOPED IN SABER^{TM}

```
#*****************************************************************************
# THERMAL MODEL OF GENERATION II IPEM                                        *
# Description: This template develop the reduced thermal model, based on     *
#              LTCM, of Generation II IPEM. Here, this IPEM is divided in    *
#              several lumped (each one with a mathematical expression that  *
#              describe its thermal behavior) which conform a set nonlinear  *
#              equation system. The implemented model is an array of eleven  *
#              equations with eleven variables.                              *
# Filename : "ltcm_ipem.sin"                                                 *
# NOTE : The material of Gate Driver was assumed as Silicon (Si), all        *
#        properties correspond to that material. Also, the power dissipated  *
#        by the gate driver is simulated as a constant.                       *
# ===================== Template and Header Declaration =====================
template ltcm_ipem l2

# Declaration of connections
thermal_c l2
external number temp      # Ambient temperature function from netlist
# ============================ Template's Body =============================
{                         # Start of template body
<units.sin
# CONSTANT VALUES

# For CONVECTION

# Convection Lengths (m)
number   l1h = 2.108e-2,  # Horizontal convection length of Lumped 1
          l1v = 2.54e-4,   # Vertical convection length of Lumped 1
          l2h = 7.19e-3,   # Horizontal convection length of Lumped 2
          l3h = 7.19e-3,   # Horizontal convection length of Lumped 3
          l4h = 4.902e-3,  # Horizontal convection length of Lumped 4
          l4v = 2.54e-4,   # Vertical convection length of Lumped 4
          l5h = 4.06e-3,   # Horizontal convection length of Lumped 5
          l5v = 1.143e-3,  # Vertical convection length of Lumped 5
          l6h = 4.902e-3,  # Horizontal convection length of Lumped 6
          l6v = 2.54e-4,   # Vertical convection length of Lumped 6
          l7h = 4.06e-3,   # Horizontal convection length of Lumped 7
          l7v = 1.143e-3,  # Vertical convection length of Lumped 7
          l8v = 1.143e-3,  # Vertical convection length of Lumped 8
          l9h = 3.0e-2,    # Horizontal convection length of Lumped 9
          l9v = 8.89e-4,   # Vertical convection length of Lumped 9
         l10v = 6.35e-4,   # Vertical convection length of Lumped 10
         l11v = 2.54e-4,   # Vertical convection length of Lumped 11
         l12h = 5.07e-2,   # Horizontal convection length of Lumped 12
         l12v = 4.0e-3     # Vertical convection length of Lumped 12

# Convection Areas (m**2)
number   A1h = 1.79e-4,   # Horizontal convective area of Lumped 1
          A1v = 1.5e-5,    # Vertical convective area of Lumped 1
          A2h = 6.36e-5,   # Horizontal convective area of Lumped 2
          A3h = 6.36e-5,   # Horizontal convective area of Lumped 3
          A4h = 4.86e-5,   # Horizontal convective area of Lumped 4
          A4v = 9.72e-6,   # Vertical convective area of Lumped 4
          A5h = 1.49e-5,   # Horizontal convective area of Lumped 5
          A5v = 6.66e-6,   # Vertical convective area of Lumped 5
          A6h = 4.86e-5,   # Horizontal convective area of Lumped 6
          A6v = 9.72e-6,   # Vertical convective area of Lumped 6
          A7h = 1.49e-5,   # Horizontal convective area of Lumped 7
          A7v = 6.66e-6,   # Vertical convective area of Lumped 7
          A8v = 3.25e-5,   # Vertical convective area of Lumped 8
          A9h = 5.16e-4,   # Horizontal convective area of Lumped 9
          A9v = 9.04e-5,   # Vertical convective area of Lumped 9
         A10v = 7.23e-5,   # Vertical convective area of Lumped 10
         A11v = 2.89e-5,   # Vertical convective area of Lumped 11
         A12h = 9.67e-4,   # Horizontal convective area of Lumped 12
         A12v = 6.86e-3    # Vertical convective area of Lumped 12


# For RADIATION

# Emisivities
number  emi1 = 0.86,      # Emisivity of Lumped 1
        emi2 = 0.86,      # Lumped 2
        emi3 = 0.86,      # Lumped 3
        emi4 = 0.052,     # Lumped 4
        emi5 = 0.052,     # Lumped 5
        emi6 = 0.052,     # Lumped 6
        emi7 = 0.052,     # Lumped 7
        emi8 = 0.052,     # Lumped 8
        emi9 = 0.335,     # Lumped 9
       emi10 = 0.335,     # Lumped 10
       emi11 = 0.052,     # Lumped 11
       emi12 = 0.052      # Lumped 12

# Geometry Factors
number  F1inf = 0.99,      # Geometry factor for radiation between Lumped 1
                           # and environment
        F2inf = 0.94,      # Between Lumped 2 and environment
        F3inf = 0.96,      # Between Lumped 3 and environment
        F4inf = 1.0,       # Between Lumped 4 and environment
        F5inf = 1.0,       # Between Lumped 5 and environment
        F6inf = 1.0,       # Between Lumped 6 and environment
        F7inf = 1.0,       # Between Lumped 7 and environment
        F8inf = 1.0,       # Between Lumped 8 and environment
        F9inf = 1.0,       # Between Lumped 9 and environment
       F10inf = 1.0,       # Between Lumped 10 and environment
       F11inf = 1.0,       # Between Lumped 11 and environment
       F12inf = 1.0,       # Between Lumped 12 and environment
          F12 = 4.9e-3,    # Between Lumped 1 and Lumped 2
          F13 = 5.2e-3,    # Between Lumped 1 and Lumped 3
          F21 = 5.8e-2,    # Between Lumped 2 and Lumped 1
          F31 = 4.6e-2     # Between Lumped 3 and Lumped 1

# Radiation Areas (m**2)
number  A1inf = 1.943e-4,  # Radiation area between Lumped 1 and ambient
        A2inf = 3.638e-5,  # Between Lumped 2 and ambient
        A3inf = 3.638e-5,  # Between Lumped 3 and ambient
        A4inf = 5.836e-5,  # Between Lumped 4 and ambient
        A5inf = 3.393e-5,  # Between Lumped 2 and ambient
        A6inf = 5.836e-5,  # Between Lumped 6 and ambient
        A7inf = 3.393e-5,  # Between Lumped 7 and ambient
        A8inf = 4.491e-5,  # Between Lumped 8 and ambient
        A9inf = 6.063e-4,  # Between Lumped 9 and ambient
       A10inf = 7.229e-5,  # Between Lumped 10 and ambient
       A11inf = 2.892e-5,  # Between Lumped 11 and ambient
       A12inf = 7.823e-3,  # Between Lumped 12 and ambient
          A12 = 5.354e-6,  # Between Lumped 1 and Lumped 2
          A13 = 5.354e-6,  # Between Lumped 1 and Lumped 3
          A21 = 3.638e-5,  # Between Lumped 2 and Lumped 1
          A31 = 3.638e-5   # Between Lumped 3 and Lumped 1

# Boltzmann's Constant
number  sigma = 5.67e-8    # (W/m**2.K**4)


# For THERMAL CONTACT RESISTANCE (K/W)
```

```
number    R19 = 0.15614,    # Thermal contact resistance between Lumped 1 and
                            # Lumped 9
           R28 = 0.1585,     # Between Lumped 2 and Lumped 8
           R38 = 0.1585,     # Between Lumped 3 and Lumped 8
           R29 = 0.1546,     # Between Lumped 2 and Lumped 9
           R39 = 0.1546,     # Between Lumped 3 and Lumped 9
           R24 = 0.1539,     # Between Lumped 2 and Lumped 4
           R36 = 0.1539,     # Between Lumped 3 and Lumped 6
           R98 = 0.1571,     # Between Lumped 9 and Lumped 8
           R59 = 0.1577,     # Between Lumped 5 and Lumped 9
           R79 = 0.1577,     # Between Lumped 7 and Lumped 9
          R810 = 0.1524,     # Between Lumped 8 and Lumped 10
         R1011 = 0.1551,     # Between Lumped 10 and Lumped 11
           R45 = 0.1559,     # Between Lumped 4 and Lumped 5
           R67 = 0.1559,     # Between Lumped 6 and Lumped 7
           R58 = 0.1596,     # Between Lumped 5 and Lumped 8
           R78 = 0.1596,     # Between Lumped 7 and Lumped 8
         R1112 = 0.1538,     # Between Lumped 11 and Lumped 12
     R12isoth = 0.1597      # Between Lumped 12 and the isothermal plate


# For TRANSIENT EFFECTS

# Volumes (m**3)
number     v1 = 1.14e-7,    # Lumped 1 - gate driver
            v2 = 5.7e-8,     # Lumped 2 - left Si-chip
            v3 = 5.7e-8,     # Lumped 3 - right Si-chip
            v4 = 1.01e-7,    # Lumped 4 - Cu-metallization layer portion
            v5 = 1.48e-8,    # Lumped 5 - Cu-metallization layer portion
            v6 = 1.01e-7,    # Lumped 6 - Cu-metallization layer portion
            v7 = 1.48e-8,    # Lumped 7 - Cu-metallization layer portion
            v8 = 2.05e-7,    # Lumped 8 - copper trace layer
            v9 = 5.72e-7,    # Lumped 9 - Al2O3-DBC ceramic layer
           v10 = 5.13e-7,    # Lumped 10 - Al2O3-DBC ceramic base
           v11 = 2.05e-7,    # Lumped 11 - copper base layer
           v12 = 7.098e-6    # Lumped 12 - copper heat spreader

# Densities (kg/m**3)
number     d1 = 2329,
            d2 = 2329,
            d3 = 2329,
            d4 = 8900,
            d5 = 8900,
            d6 = 8900,
            d7 = 8900,
            d8 = 8900,
            d9 = 3900,
           d10 = 3900,
           d11 = 8900,
           d12 = 8900

# Heat Capacities (J/kg.C)
number    cp1 = 702,
           cp2 = 702,
           cp3 = 702,
           cp4 = 390,
           cp5 = 390,
           cp6 = 390,
           cp7 = 390,
           cp8 = 390,
           cp9 = 850,
          cp10 = 850,
          cp11 = 390,
          cp12 = 390

# Isothermal plate (Kelvin degree)
number Tisoth = 296.15
```

```
# VARIABLE / VALUES

# To calculate Heat Transfer Coefficients of each lumped
val nu h1h, h1v, h2h, h3h, h4h, h4v, h5h, h5v, h6h, h6v, h7h, h7v, h8v, h9h,
       h9v, h10v, h11v, h12h, h12v

# To calculate Lumped Temperatures in Celcius degree
var tk     T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12

# Dissipated power by the Gate Driver (W)
var p heat2

values {
        # Obtains heat transfer coefficients
        h1h = 1.32*((abs(T1 - (temp+273.15))/l1h)**0.25)
        h1v = 1.42*((abs(T1 - (temp+273.15))/l1v)**0.25)
        h2h = 1.32*((abs(T2 - (temp+273.15))/l2h)**0.25)
        h3h = 1.32*((abs(T3 - (temp+273.15))/l3h)**0.25)
        h4h = 1.32*((abs(T4 - (temp+273.15))/l4h)**0.25)
        h4v = 1.42*((abs(T4 - (temp+273.15))/l4v)**0.25)
        h5h = 1.32*((abs(T5 - (temp+273.15))/l5h)**0.25)
        h5v = 1.42*((abs(T5 - (temp+273.15))/l5v)**0.25)
        h6h = 1.32*((abs(T6 - (temp+273.15))/l6h)**0.25)
        h6v = 1.42*((abs(T6 - (temp+273.15))/l6v)**0.25)
        h7h = 1.32*((abs(T7 - (temp+273.15))/l7h)**0.25)
        h7v = 1.42*((abs(T7 - (temp+273.15))/l7v)**0.25)
        h8v = 1.42*((abs(T8 - (temp+273.15))/l8v)**0.25)
        h9h = 1.32*((abs(T9 - (temp+273.15))/l9h)**0.25)
        h9v = 1.42*((abs(T9 - (temp+273.15))/l9v)**0.25)
        h10v = 1.42*((abs(T10 - (temp+273.15))/l10v)**0.25)
        h11v = 1.42*((abs(T11 - (temp+273.15))/l11v)**0.25)
        h12h = 1.32*((abs(T12 - (temp+273.15))/l12h)**0.25)
        h12v = 1.42*((abs(T12 - (temp+273.15))/l12v)**0.25)

      } # end of value section


control_section {
                initial_condition(T1,temp+275.15)
                initial_condition(T2,temp+275.15)
                initial_condition(T3,temp+275.15)
                initial_condition(T4,temp+275.15)
                initial_condition(T5,temp+275.15)
                initial_condition(T6,temp+275.15)
                initial_condition(T7,temp+275.15)
                initial_condition(T8,temp+275.15)
                initial_condition(T9,temp+275.15)
                initial_condition(T10,temp+275.15)
                initial_condition(T11,temp+275.15)
                initial_condition(T12,temp+275.15)

               } # end of control_section

equations{
T1:             d_by_dt(T1) = (-((T1-(temp+273.15))*((h1h*A1h)+(h1v*A1v)))-
                             (emi1*sigma*F1inf*A1inf*((T1**4)-((temp+273.15)**4)))-
                             (emi1*sigma*F12*A12*((T1**4)-(T2**4)))-
                             (emi1*sigma*F13*A13*((T1**4)-(T3**4)))+
                             (emi2*sigma*F21*A21*((T2**4)-(T1**4)))+
                             (emi3*sigma*F31*A31*((T3**4)-(T1**4)))-
                             ((T1-T9)/R19))/(d1*cp1*v1)

T2:             d_by_dt(T2) = (heat2-(h2h*A2h*(T2-(temp+273.15)))-
                             (emi2*sigma*F2inf*A2inf*((T2**4)-((temp+273.15)**4)))-
                             (emi2*sigma*F21*A21*((T2**4)-(T1**4)))+
                             (emi1*sigma*F12*A12*((T1**4)-(T2**4)))-
```

```
                              ((T2-T4)/R24)-((T2-T8)/R28)-((T2-T9)/R29))/(d2*cp2*v2)

T3:             d_by_dt(T3) = (-(h3h*A3h*(T3-(temp+273.15)))-
                              (emi3*sigma*F3inf*A3inf*((T3**4)-((temp+273.15)**4)))-
                              (emi3*sigma*F31*A31*((T3**4)-(T1**4)))+
                              (emi1*sigma*F13*A13*((T1**4)-(T3**4)))-
                              ((T3-T6)/R36)-((T3-T8)/R38)-((T3-T9)/R39))/(d3*cp3*v3)

T4:             d_by_dt(T4) = (-((T4-(temp+273.15))*((h4h*A4h)+(h4v*A4v)))-
                              (emi4*sigma*F4inf*A4inf*((T4**4)-((temp+273.15)**4)))+
                              ((T2-T4)/R24)-((T4-T5)/R45))/(d4*cp4*v4)

T5:             d_by_dt(T5) = (-((T5-(temp+273.15))*((h5h*A5h)+(h5v*A5v)))-
                              (emi5*sigma*F5inf*A5inf*((T5**4)-((temp+273.15)**4)))+
                              ((T4-T5)/R45)-((T5-T8)/R58)-((T5-T9)/R59))/(d5*cp5*v5)

T6:             d_by_dt(T6) = (-((T6-(temp+273.15))*((h6h*A6h)+(h6v*A6v)))-
                              (emi6*sigma*F6inf*A6inf*((T6**4)-((temp+273.15)**4)))+
                              ((T3-T6)/R36)-((T6-T7)/R67))/(d6*cp6*v6)

T7:             d_by_dt(T7) = (-((T7-(temp+273.15))*((h7h*A7h)+(h7v*A7v)))-
                              (emi7*sigma*F7inf*A7inf*((T7**4)-((temp+273.15)**4)))+
                              ((T6-T7)/R67)-((T7-T8)/R78)-((T7-T9)/R79))/(d7*cp7*v7)

T8:             d_by_dt(T8) = (-(h8v*A8v*(T8-(temp+273.15)))-
                              (emi8*sigma*F8inf*A8inf*((T8**4)-((temp+273.15)**4)))+
                              ((T9-T8)/R98)+((T5-T8)/R58)+((T2-T8)/R28)+
                              ((T7-T8)/R78)+((T3-T8)/R38)-((T8-T10)/R810))/(d8*cp8*v8)

T9:             d_by_dt(T9) = (-((T9-(temp+273.15))*((h9h*A9h)+(h9v*A9v)))-
                              (emi9*sigma*F9inf*A9inf*((T9**4)-((temp+273.15)**4)))+
                              ((T1-T9)/R19)+((T2-T9)/R29)+((T3-T9)/R39)-
                              ((T9-T8)/R98)+((T5-T9)/R59)+((T7-T9)/R79))/(d9*cp9*v9)

T10:           d_by_dt(T10) = (-(h10v*A10v*(T10-(temp+273.15)))-
                              (emi10*sigma*F10inf*A10inf*((T10**4)-((temp+273.15)**4)))+
                              ((T8-T10)/R810)-((T10-T11)/R1011))/(d10*cp10*v10)

T11:           d_by_dt(T11) = (-(h11v*A11v*(T11-(temp+273.15)))-
                              (emi11*sigma*F11inf*A11inf*((T11**4)-((temp+273.15)**4)))+
                              ((T10-T11)/R1011))/(d11*cp11*v11)

T12:           d_by_dt(T12) = (-((T12-(temp+273.15))*((h12h*A12h)+(h12v*A12v)))-
                              (emi12*sigma*F12inf*A12inf*((T12**4)-((temp+273.15)**4)))+
                              ((T11-T12)/R1112)-((T12-Tisoth)/R12isoth))/(d12*cp12*v12)

               p(l2) += heat2
               heat2: tc(l2) = T2 - 273.15

                } # end of equation section

} # end of Template's Body
```