Thermal Modelling of CPUs with Heat Sink:

Comparing FEM Software and Object-Oriented Modelling Tools

Master Thesis

POLITECNICO
MILANO 1863

Co-Authors:

Can AYVAZ (Matriculation Num: 893775)

Necati Mert ÇELEN (Matriculation Num: 892219)

Supervisors:

Prof. Dr. Alberto LEVA

Prof. Dr. Federico TERRANEO

Scuola di Ingegneria Industriale e dell'Informazione

Master of Science in Automation and Control Engineering

Politecnico di Milano

Academic Year 2018-2019

Abstract

This thesis is concerned with thermal modelling of a microprocessor and its heat sink structure. Thermal modelling of the whole structure is made with two different approaches, which are Finite Element Method (FEM) and Finite Volume Method (FVM). The first method is implemented with the usage of ANSYS which is commercial software, while the second method is implemented with the usage of the open source Modelica coding environment. These models are created with the usage of relevant softwares and heat equations, to have similar results with experimental temperature data for microprocessor cores. Comparison of the temperature data obtained from the simulations are compared to the experimental data. However, the temperature distribution for the heat sink is experimentally unknown, thus, the temperature distribution data obtained through the simulations are compared to each other.

From the point of view of control, the usage of the Modelica coding environment is substantial due to the fact that it uses the FVM approach. This method requires less computational power and time, because it utilizes less state variables without losing its precision. The obtained simulation data can be considered as a plant model for a temperature controller design, thus, it is important to have a viable model with less state variables. Additionally, the OpenModelica environment is free for public usage and easily adaptable for changing structure dimensions and material properties.

Sommario

Questa tesi riguarda la modellizzazione termica di un microprocessore e la sua struttura di dissipatore di calore. La modellizzazione termica dell'intera struttura è realizzata con due approcci diversi, che sono il metodo degli elementi finiti (FEM) e il metodo dei volumi finiti (FVM). Il primo metodo è implementato con l'uso di ANSYS che è un software commerciale, mentre il secondo metodo è implementato con l'uso dell'ambiente di codifica Modelica open source. Questi modelli sono creati con l'utilizzo di software e equazioni di calore pertinenti, per ottenere risultati simili con dati sperimentali sulla temperatura per i nuclei di microprocessore. Il confronto dei dati di temperatura ottenuti dalle simulazioni viene confrontato con i dati sperimentali. Tuttavia, la distribuzione della temperatura per il dissipatore di calore è sperimentalmente sconosciuta, quindi i dati di distribuzione della temperatura ottenuti attraverso le simulazioni vengono confrontati tra loro.

Dal punto di vista del controllo, l'utilizzo dell'ambiente di codifica Modelica è sostanziale in quanto utilizza l'approccio FVM. Questo metodo richiede meno tempo e potenza computazionale, poiché utilizza meno variabili di stato senza perdere la sua precisione. I dati di simulazione ottenuti possono essere considerati come un modello di impianto per la progettazione di un termoregolatore, pertanto è importante disporre di un modello praticabile con meno variabili di stato. Inoltre, l'ambiente OpenModelica è gratuito per uso pubblico e facilmente adattabile per modificare le dimensioni della struttura e le proprietà del materiale.

Acknowledgements

We would like to thank Professor Leva and Professor Terraneo, for sharing their guidance, experience, suggestions and the help they provided to us at every step of this thesis.

We are also grateful to our families and friends for their support throughout this thesis.

August 2019 Can AYVAZ, Necati Mert ÇELEN

Contents

List of Figures

List of Tables

Nomenclature

α	Thermal diffusivity	$[\frac{m^2}{s}]$
$\bar{\bar{\tau}}_{eff}$	Viscous stress tensor	$[\frac{N}{m^2}]$
ϵ_{ext}	Emissivity	
ρ	Density of material	$[\frac{kg}{m^3}]$
σ	Stefan-Boltzmann constant	$[\frac{W}{m^2K^4}]$
$\vec{J}$	Diffusion flux	$[\frac{kg}{m^2s}]$
$\vec{v}$	Velocity	$[\frac{m}{s}]$
c_p	Specific heat for constant pressure	$[\frac{J}{kgK}]$
E	Internal energy per unit mass	$[\frac{J}{kg}]$
h_{ext}	Convective heat transfer constant	$[\frac{W}{m^2K}]$
h_t	Sensible enthalphy of unit mass	$[\frac{J}{kg}]$
k_{eff}	Effective thermal conductivity	$[\frac{W}{mK}]$
k_l	Thermal conductivity without turbulence	$[\frac{W}{mK}]$
k_t	Turbulent thermal conductivity	$[\frac{W}{mK}]$
L_c	Characteristic length	$[m]$
mL	Dimensionless fin index	
p	Pressure	$[\frac{N}{m^2}]$
q	Heat flux	$[\frac{J}{m^2}]$
S_h	Volumetric heat generation term	$[\frac{W}{m^3}]$
T	Temperature	$[K]$
t	Time	$[s]$
Y	Mass fraction	

Introduction

Modelling of a system is one of the most important stages in Control System Design, because, modelling provides information about system dynamics and it can simplify relations between bodies or entities in the model. Additionally, by using the dynamic equations found via modelling, simulation of a system can be built in software environment without the need of different experimental setups. Thus, modelling and simulation can reduce the amount of money and time spent to build real-life experiments to find an optimum solution.

This work focuses on thermal modelling of a microprocessor and its heat sink. Thermal modelling of these electronic components is important, since, materials used in those, have optimum working temperature ranges due to their different material properties. These systems are characterized by an internal heat generation or an external heat input added to the system; hence, it is necessary to have a cooling system assuring this optimum temperature range.

Microprocessors are the command centers of electronic devices having logic gates, transistors etc. to realize different tasks. When realizing these tasks, microprocessors can have different energy needs and it results different values of heat loss occurring during the process, which increases the microprocessor temperature. To keep this temperature within safe limits, it is necessary to assure the heat dissipation via increasing the dissipation surface. For this purpose, heat sinks are attached to the microprocessors. The heat sinks consists of one base plate and the surfaces perpendicular it, which are named as fins. In general, they are made of metals with high thermal conductivity to increase the heat dissipation. The microprocessors are used in data centers, in huge numbers to provide a high calculation capability.

In the data centers, temperature of the center is kept low enough to provide a safe working condition for the microprocessors, while they are working on a full load, hence, cost of keeping the data center at low temperatures is immense due to its size and amount of the heat generated by the microprocessors. Additionally, the microprocessors at the data center are not fully loaded most of the cases, thus, there is an excess energy con-

sumption to keep the temperature at this low value without the actual need. Thus, by modelling and simulating different conditions on the heat sink and the microprocessor, it is possible to obtain the temperature distribution data of the heat sink over time. By using this data, it is possible to understand overall load on the microprocessor, hence, the temperature of the data center could be adjusted by Heating, Ventilating and Air Conditioning (HVAC) system to have an optimum temperature value for different load cases. Instead of assuring constant low temperature for safety, with the aid of this system, the temperature data of the specific points of data center is predicted and when a threshold value is passed, by using HVAC system, the temperature of that specific point could be controlled accordingly. The heat transfer simulations to realize this aim, are made through the commercial Computational Fluid Dynamics softwares like ANSYS, COMSOL etc. Although these softwares give accurate results for simulated systems and have dedicated user-interfaces, they are time consuming, they need high computational power and the license prices of these softwares can get high numbers. The usage of the open source coding environments comes into play at this point. One of these environments is Modelica; it has no built-in functions for structures with specific geometries. However, by introducing the heat transfer equations for the finite volume elements, it is possible to model the temperature distribution in the structure. This method offers a solution with less time and computational power requirements.

In this work, the thermal modelling of a microprocessor with its heat sink is realized through these two different approaches and the results are compared with each other together with the experimental data. The second section addresses to the technical data and the structure of the microprocessor and the heat sink complex. Experimental setup is described, and the test procedures are explained. In the third section, two methods used in modelling and simulations, are explained with their approach to solve the stated problem. The fourth section clarifies the heat transfer theory used in the ANSYS software, the setup parameters used in the simulations and the obtained numerical results. In the fifth section, the modelling of the same system is done through the Modelica environment. The heat transfer formulations, introduction of conduction and convection heat transfer terms in those formulations and creation of the structure with finite volumes are described and the simulation results obtained are given. The sixth section is dedicated to the comparison of the results of ANSYS and Modelica with the experimental data obtained by temperature measurement. A final word is said on the future work related to this work in the conclusion part.

Comparison of Two Modelling Approaches

Two approaches used in the simulations are basically the Finite Element Method (FEM) and the Finite Volume Method (FVM). ANSYS uses the FEM to evaluate the temperature distribution over the bodies via using predefined heat transfer equations. These equations with their theoretical background are explained in Chapter 5. To create a simulation model in ANSYS, it is necessary to introduce a Computer Aided Design (CAD) drawing of the system either by using ANSYS SpaceClaim or other CAD softwares such as SolidWorks, CATIA etc. However, to realize the simulation in the Modelica environment, introduction of the shape under investigation is made through declaring it in the code with mathematical representations.

In the ANSYS calculation procedure, bodies are divided into tetrahedral mesh elements to solve heat equation over time. The vertexes of these tetrahedral elements are the points where the equations are solved. Each point has neighboring points where a shape function is defined between these points. These shape functions are generally selected as weight functions to include the individual heat term contribution of each neighboring point to the temperature of the specified point. These piece-wise defined weight functions are forming together an approximation to the original heat transfer equation when summed up [6].

To avoid divergence of the solution, the time step and the mesh size must be chosen accordingly. Tetrahedral cell dimensions do not need to be equal to evaluate piece-wise integrals. This method reduces to Finite Difference Method (FDM) if the meshes are selected equispaced. More information about FDM can be found in Chapter 5 in the transient formulations part.

On the other hand, the Modelica simulation method is decided through an examination over transient and steady-state formulation for the heat transfer. The FVM is chosen due to its easier and conservative approach on complex geometries to apply the heat equations in discrete time and space. In the Modelica model, the bodies are divided to rectangular prisms and the energy balance equation for these control volumes are introduced. For a specific element, the net heat input coming from the boundary surfaces of the control volume and the heat generation occurring in that element, have a temperature contribu-

tion on that element. The net heat input is calculated by taking integrals of heat fluxes on the boundary surfaces. This integration provides the temperature value of the cell center, which represents the temperature value of the whole volume assuming a uniform distribution throughout the cell. Since the FVM assumes an equal (in magnitude) heat transfer through two neighboring elements, the FVM has more robust characteristics for changing time step and mesh sizes. It is possible to converge to a reasonable solution with a smaller number of elements. Similar to FEM, when the meshes are chosen equispaced, the FVM reduces to the FDM formulations [6].

Experimental Setup and Test Procedure

In the real-life applications, microprocessors realize different tasks, however, the load distribution at their cores and the power consumed by the microprocessor are not adjusted. Thus, for a certain core, the power consumption of that core for a specific task is not known. To simulate different load cases, the thermal test chips, which have only resistors inside their cores, are used instead of the real microprocessors. By adjusting the voltage and current values passing through these resistors, it is possible to realize different load scenarios on each cores of the microprocessor creating different amounts of heat generation.

The setup consists of two main parts, which are the microprocessor and the heat sink. The microprocessor material is silicon and it consists of 16 cores which are located as 4x4 matrix. These cores are called as active silicon and remaining part of the microprocessor is called as passive silicon. The dimensions of the microprocessor is 10.23x10.23x0.625mm. Each core consists of two resistors and one temperature sensor which is located between them. Core thickness determined as 16 % of chip thickness. Technical drawing of microprocessor and all used material properties are given in Appendix A [3].

The heat sink material is copper, it is made of 60x60x2.8mm base plate and it has twenty-three rectangular fin arrays. Seventeen of these fins have dimensions of 60x19.2x0.5mm and other six fins have different length due to screw holes. This fact is considered in the ANSYS simulation, but it is not modelled in the Modelica model. The technical drawing is given in Appendix B [9].

There is rubber pad located between the microprocessor and the heat sink to prevent air cavitation at the interface surfaces of these two. Dimensions of the rubber pad is 10.23x10.23x0.15mm and its technical drawing and material properties are given in Appendix A. Heat transfer from microprocessor to heat sink occurs through this rubber pad. Bottom surface of the microprocessor is assumed as an adiabatic surface and the side surfaces of the microprocessor and the rubber pad are assumed to have negligible convection heat transfer to the medium.

The experiment consists of three stages. The first one is the idle working status of the microprocessor. At this stage, all cores are fed by idle voltage (1.65 V) and current (0.113 A), which represent the idle capacity power value (0.186 W). During this stage, the temperature of the microprocessor reaches to a steady-state value. The time waited to reach that steady-state temperature value for all experiments is determined as 1800 seconds. At the second stage, the voltage and the current values of the specified cores are risen to its full capacity power value (3.0516 W). Sudden temperature rise in the microprocessor occurs at this short period. At the third stage, the temperature of the microprocessor tries to reach another steady-state value. However, except the first load case where only one core of the microprocessor is fully loaded at 1800 seconds, the temperature of the cores cannot reach to a new steady-state value. Thus, the first load case has 3600 seconds of experiment time, while the other cases have less than that value, since the temperature of the full capacity working cores should not pass the limit of 100 °C to prevent the microprocessor damage. When the temperature of the cores reaches limit value, the voltage and the current value of the cores are decreased to zero as a protection mechanism. During the experimental procedure, the temperature data of all cores are recorded for different cases. The experimental procedure is repeated for four different cases including one, four, nine and sixteen cores working in full capacity after 1800 seconds. For each experimental setup, ambient temperature is measured and recorded.

Thermal Simulation of Chip and Heat Sink in ANSYS

This chapter is dedicated to heat transfer simulations realized in ANSYS for the microprocessor and the heat sink complex. Firstly, the heat transfer theory and the formulations used by ANSYS Fluent module are clarified. Secondly, modelling and simulation steps followed in the ANSYS software are explained. In final the section, the simulation results obtained through ANSYS are given and these results are compared with the experimental data for the microprocessor core temperatures. Detailed analysis on the fin temperatures is made in Chapter 6 while comparing the results with the Modelica simulation.

4.1 Heat Transfer Theory in ANSYS

The general heat transfer equation in ANSYS can be written as in Equation 4.1 [2] where k_{eff} is the effective conductivity of the system, which is equal to $k_l + k_t$ and k_t is the turbulent thermal conductivity. $\vec{J_j}$ is the diffusion flux of species j. The terms on the right-hand side represent the energy transfer due to conduction, the species diffusion, the viscous dissipation, and the heat generation in the element due to various sources respectively. The heat generation term S_h includes chemical reaction, radiation, and all other volumetric heat sources defined by user.

$$\frac{\partial}{\partial t}(\rho E) + \nabla(\vec{v}(\rho E + p)) = \nabla(k_{eff}\nabla T - \sum_j h_{t,j}\vec{J_j} + (\bar{\bar{\tau}}_{eff}\vec{v})) + S_h \tag{4.1}$$

On the left-hand side, the first term represents the energy change in time while the second term represents the energy change in space coordinates. Where ρ stands for the density, E represents the internal energy inside the unit mass and can be written as Equation [2]:

$$E = h_t - \frac{p}{\rho} - \frac{v^2}{2} \tag{4.2}$$

In Equation 4.2, h_t stands for the sensible enthalpy, which is the material enthalpy minus its formation enthalpy. For ideal gases, it can be written as Equation 4.3 [2], however, for the incompressible flows, equation is slightly changed, and it is stated in Equation 4.4 [2]:

$$h_t = \sum_j Y_j h_{t,j} \tag{4.3}$$

$$h_t = \sum_j Y_j h_{t,j} + \frac{\rho}{p} \tag{4.4}$$

In Equation 4.3 and 4.4, Y_j and h_j represents the mass fraction and the sensible enthalpy of the j^{th} material. $h_{t,j}$ can be found by Equation 4.5 [2]:

$$h_{t,j} = \int_{T_{ref}}^{T} c_{p,j} dT \tag{4.5}$$

For solid regions, Equation 4.1 becomes as it is given in Equation 4.6 [2]. Since, the energy transfer only occurs in solid region, the internal energy of the system reduces from E to h_t. Additionally, on the right-hand side of the general heat transfer equation, the conduction heat transfer coefficient simply becomes k_t; the species diffusion and the viscous dissipation terms are omitted due to solid region condition.

$$\frac{\partial}{\partial t}(\rho h_t) + \nabla(\vec{v}\rho h_t) = \nabla(k\nabla T) + S_h \tag{4.6}$$

However, Equation 4.6 can be simplified further, since, the second term on the left-hand side stands for the energy change due to rotational or translational motion of the solid body with $\vec{v}$ representing the velocity field of the solid body. Thus, this term becomes zero, when there is no rotational or translational motion of the solid body. Also, change in the density and the specific heat capacity of the solid is negligible due to small temperature variation of the solid body. The formula given in 4.5 becomes $h = \int_{T_{ref}}^{T} c_p dT$ when calculating the sensible enthalpy of the single material solid. Thus, the final form of the heat transfer equation is given in Equation 4.7 [2]:

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T + \frac{\alpha}{k} S_h \tag{4.7}$$

Where $\alpha = \frac{k}{\rho c_p}$, which is the thermal diffusivity of the solid material. S_h corresponds to volumetric heat source generated by the microprocessor cores in the simulation model. In real-life experiment and simulation model the microprocessor cores have three modes, which are off, idle (when it works at 0 % load), and full power (when it works at 100 %

load). Thus, each mode has different power value, which is given as Watt.

The convective and the radiative heat transfer boundary condition is introduced by using Equation 4.8. T_{ext} represents the temperature of the medium, T_w stands for the temperature of the heat sink faces or walls. h_{ext} is the convective heat transfer constant, which is defined by user, ϵ_{ext} is the emissivity of the external wall surface of the heat sink, and σ is Stefan-Boltzmann constant for radiation heat transfer which is equal to $5.670 * 10^{-8} \frac{W}{m^2 K^4}$. On the left-hand side of the Equation 4.8 is the heat flux $q"_s$ through the surface of the heat sink.

$$q"_s = h_{ext}(T_{ext} - T_w) + \epsilon_{ext}\sigma(T_{ext}^4 - T_w^4) \tag{4.8}$$

4.2 ANSYS Modelling of the Experimental Setup

ANSYS simulations of the microprocessor and the heat sink are modelled according to the real-life experimental setup. In the real-life experiment, only measured data are temperature values of 16 cores inside the microprocessor. By matching the results of the measured data and the CFD simulations for the core temperatures, the temperature profile inside the heat sink is found numerically. ANSYS has different modules, which allow 3D design, application of the Finite Element Method, solving the energy and the flow equations, and finally it can display the temperature variation during the whole transient process. First, the CAD design of the microprocessor, the heat sink, and the rubber pad is drawn according to the technical drawings provided by the manufacturers. The microprocessor is divided into two subgroups, which are active and passive silicon.

Mesh module is used to divide parts into infinitesimally small elements. The heat sink is divided into elements which have 0.8mm size. The rubber pad, the active and the passive part of the silicon are divided into elements which have 0.2mm size to have finer result in temperature distribution. The mesh sizes of the body parts are chosen to create a balance between the simulation time and the result accuracy by considering simulation convergence. When the sizes of the heat sink elements are greater than 0.8mm, the solution of the simulation model diverges. Additionally, the rubber pad, the active and the passive silicon element sizes are not chosen smaller than 0.2mm because, when the number of elements are increased, due to the error accumulation, the simulation model diverges, and the simulation time increases significantly.

In the mesh module, the faces of the heat sink are introduced as a Named Selection to be able to declare the heat sink faces as a convection boundary condition. The active parts of the silicon are also introduced as a Named Selection to realize the energy input to the microprocessor. Contact surfaces between all body parts are checked to avoid unnecessary relations.

Material properties of the parts are introduced. Material of the heat sink is chosen as copper, material of the active and the passive part of the microprocessor is chosen as silicon but the thermal conductivity of the silicon is decreased to $110\frac{W}{mK}$, which is 25 % lower than its original value to obtain the experimental temperature rise values of the cores, indeed the active part of the silicon cannot be considered as a pure silicon. Material properties of the rubber pad are found and introduced to the simulation model. By using user defined function specification of the Fluent module, the heat generation term is added to the cores of the microprocessor. Additionally, in the real-life experiments some cores are selected to work in full power for a specific time; user defined function is constructed to realize the heat generation terms used in the real-life experiment for different cases. This user defined function for the power input is given in Appendix C. By using already defined Named Selection for the heat sink faces, the convection boundary condition is introduced to the simulation model.

As a solution method, the second order implicit formulation is used. Only the energy equations are used in the numerical calculation of the simulation model. As an initialization method, hybrid initialization is chosen to check any probable divergences in the residuals of the energy equations. The simulation time step throughout the simulation is chosen as $1.0s$ except the sudden voltage change period. Choosing the time step less than $1.0s$, does not improve the transient temperature rise during the low voltage period while it is not the case for the high voltage one. By setting the time step $0.1s$ during the sudden voltage rise, the solver is able to fully develop the input given by the user defined function so that the temperature values become more accurate.

The result module is used to introduce probes at the sensor locations and the fins to export the temperature values throughout the simulation time; from the exported data, the temperature variation versus time is plotted. Plots of the simulation result and the real-life experiment are compared and they are given in Section 4.3.

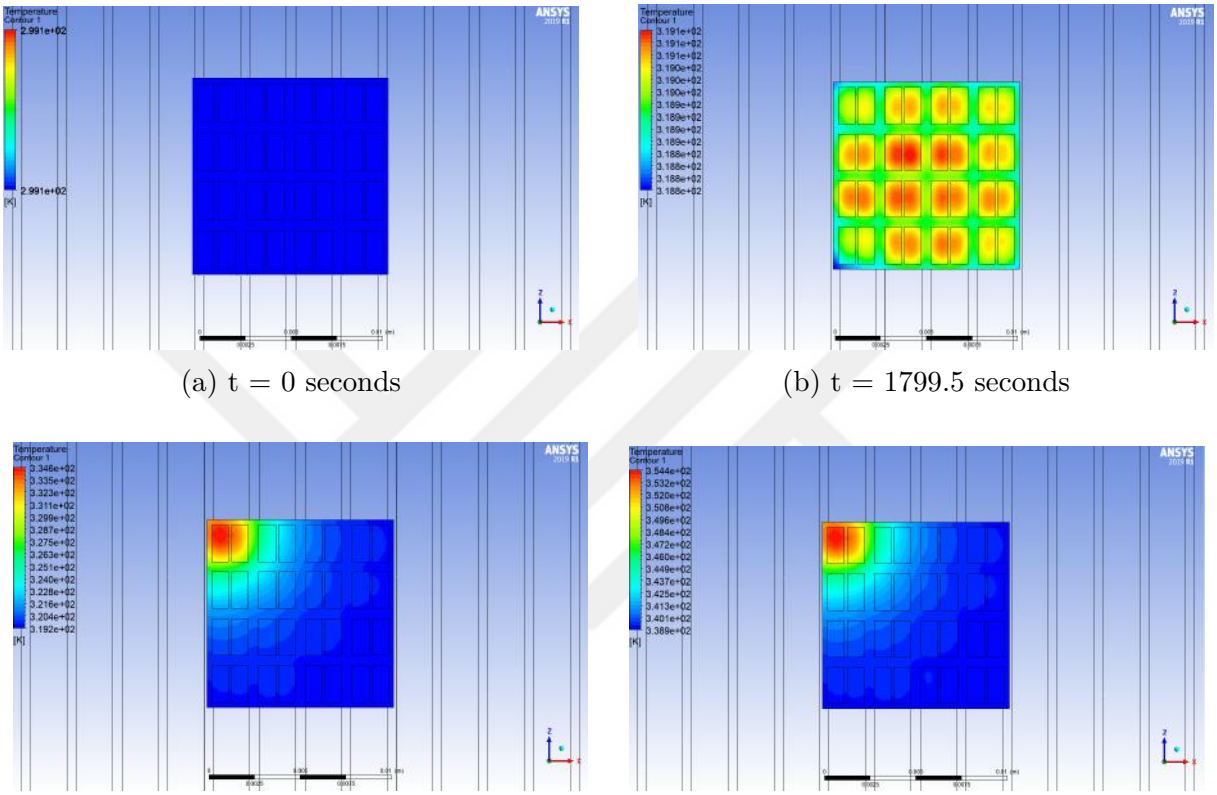
The convective heat transfer coefficient is found in the ANSYS simulations iteratively, because the convection heat transfer coefficient changes its value with changing flow characteristic. Due to the computational power and the time limitations, air medium which envelops the heat sink and the microprocessor complex is not represented. Thus, the flow equations, which are used to calculate the flow characteristics in ANSYS, are omitted. Hence, instead of changing the convective heat transfer coefficient, an average value is

used for the overall surface of the heat sink. This coefficient, which is the only unknown parameter in the simulation model, is adjusted to find the minimum error between the experiment and the simulation results. Final value of the convective heat transfer coefficient is decided as $2.6\frac{W}{m^2K}$

4.3 Simulation Results in ANSYS

As it is mentioned in Chapter 3, the ANSYS simulations are realized for four different microprocessor load cases.

For the first load case, only one core is full loaded; its temperature distribution is obtained in ANSYS for specific times and it is given in Figure 4.1.



(a) t = 0 seconds

(b) t = 1799.5 seconds

(c) t = 1803 seconds

(d) t = 3600 seconds

Figure 4.1: Temperature distribution of microprocessor for first load case

In Figure 4.2, the transient temperature change obtained from the experimental setup and ANSYS are plotted and given for one core case. In the first stage, the cores of the microprocessor reach to a steady-state temperature value, when all cores supplied with the idle voltage. As it can be seen from the Figure 4.2, the ANSYS simulation results closely track the experimental data. At 1800 seconds, the sudden voltage change is applied to only one core of the microprocessor, thus, power of this core goes from the idle capacity power value to the full capacity power value; due to that change, the sudden temperature jump occurs. Thus, the error between the ANSYS simulation and the experimental result reaches its maximum value at the sudden temperature change region or at the end of the experiment due to the error accumulation.

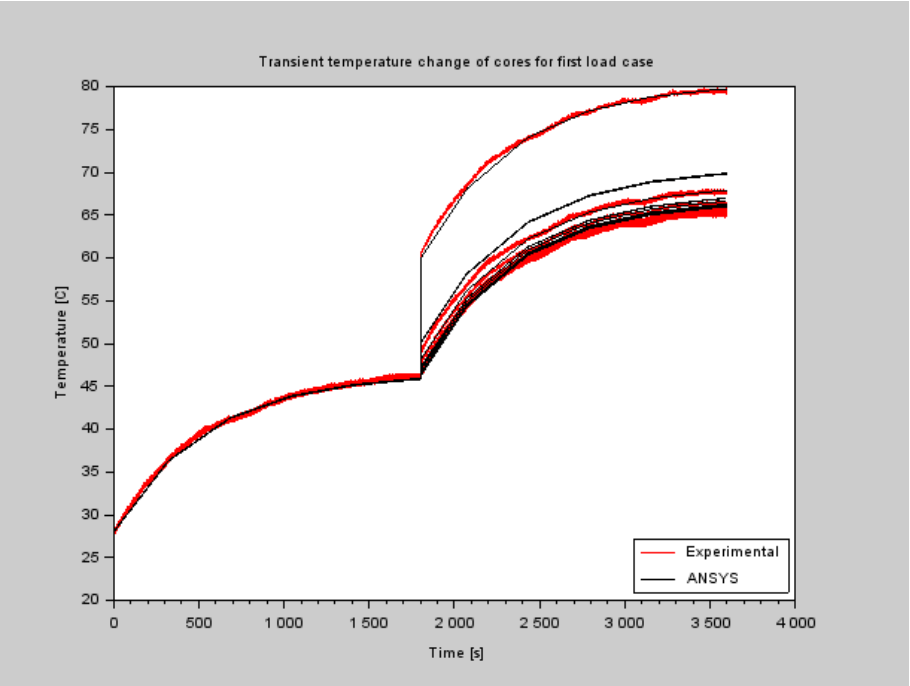


Figure 4.2: Transient temperature change for first load case

In Table 4.1, the temperature values of the experimental setup and the ANSYS simulations are given for the important time steps. These time steps are the ones used in Figure 4.1; they represent the initial and final time of the experiment and also the time steps right before and after the sudden power change. These steps are chosen for the error calculation because they are prone to highest amount of error. The error is calculated by using the internal energy values of the cores for both experimental and simulations results. The error is calculated by using Equation 4.9 where T_{sim} and T_{exp} are the simulation end the experimental temperatures in K respectively.

$$\frac{mc_pT_{sim} - mc_pT_{exp}}{mc_pT_{exp}} * 100 = \frac{T_{sim} - T_{exp}}{T_{exp}} * 100 \tag{4.9}$$

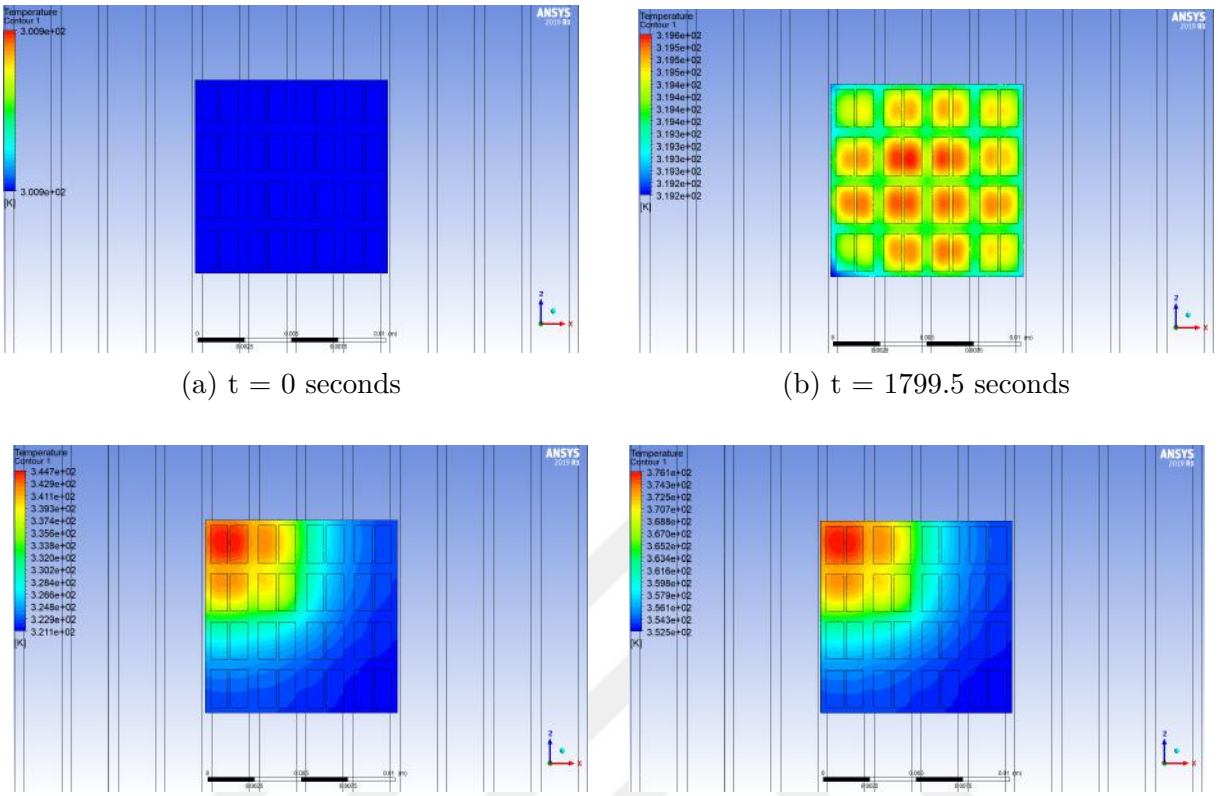
The calculated error percentages for these time steps are given in Table 4.1. The maximum error between the experimental results and the ANSYS results occur on the second core at $t = 3600s$ as 0.604 %. By using the given time step errors of the microprocessor, the overall error in ANSYS simulation is calculated by using Root Mean Square (RMS) method and which is equal to 0.026 %.

Table 4.1: Temperature and error data for first load case in ANSYS

Cores	Experimental Temperatures [K] at t = 0s	ANSYS Temperatures [K] at t = 0s	Percent Error %	Experimental Temperatures [K] at t = 1799.5s	ANSYS Temperatures [K] at t = 1799.5s	Percent Error %
1	299.15	299.15	0.000	319.05	318.98	-0.022
2	299.25	299.15	-0.033	319.25	319.02	-0.072
3	299.25	299.15	-0.033	319.25	319.03	-0.069
4	299.15	299.15	0.000	319.25	319.01	-0.075
5	299.15	299.15	0.000	319.25	319.03	-0.069
6	299.15	299.15	0.000	319.35	319.07	-0.088
7	299.15	299.15	0.000	319.35	319.06	-0.091
8	299.25	299.15	-0.033	319.25	319.02	-0.072
9	299.15	299.15	0.000	319.15	319.05	-0.031
10	299.25	299.15	-0.033	319.35	319.06	-0.091
11	299.15	299.15	0.000	319.25	319.05	-0.063
12	299.25	299.15	-0.033	319.35	319.04	-0.097
13	299.25	299.15	-0.033	319.25	318.97	-0.088
14	299.15	299.15	0.000	319.25	319.03	-0.069
15	299.15	299.15	0.000	319.25	319.04	-0.066
16	299.25	299.15	-0.033	319.15	319.00	-0.047

Cores	Experimental Temperatures [K] at t = 1803s	ANSYS Temperatures [K] at t = 1803s	Percent Error %	Experimental Temperatures [K] at t = 3600s	ANSYS Temperatures [K] at t = 3600s	Percent Error %
1	333.65	333.16	-0.147	352.65	352.88	0.065
2	322.15	323.19	0.323	340.85	342.91	0.604
3	319.85	320.37	0.163	338.65	340.09	0.425
4	319.45	319.68	0.072	338.15	339.39	0.367
5	322.25	323.3	0.326	340.65	343.02	0.696
6	320.65	321.3	0.203	339.45	341.02	0.463
7	319.85	320.05	0.063	338.65	339.77	0.331
8	319.55	319.61	0.019	338.35	339.32	0.287
9	319.85	320.39	0.169	338.55	340.11	0.461
10	320.05	320.04	-0.003	338.65	339.76	0.328
11	319.65	319.7	0.016	338.35	339.41	0.313
12	319.55	319.51	-0.013	338.35	339.22	0.257
13	319.65	319.67	0.006	338.35	339.39	0.307
14	319.55	319.64	0.028	338.35	339.36	0.299
15	319.55	319.53	-0.006	338.25	339.24	0.293
16	319.45	319.4	-0.016	338.15	339.1	0.281

For the second load case, four cores are fully loaded; its temperature distribution is obtained in ANSYS for specific times and it is given in Figure 4.3.



(a) t = 0 seconds

(b) t = 1799.5 seconds

(c) t = 1803 seconds

(d) t = 2073 seconds

Figure 4.3: Temperature distribution of microprocessor for second load case

In Figure 4.4, the transient temperature change obtained from the experimental setup and ANSYS are plotted and given for four core case. Similar procedure is also followed in the remaining load cases.

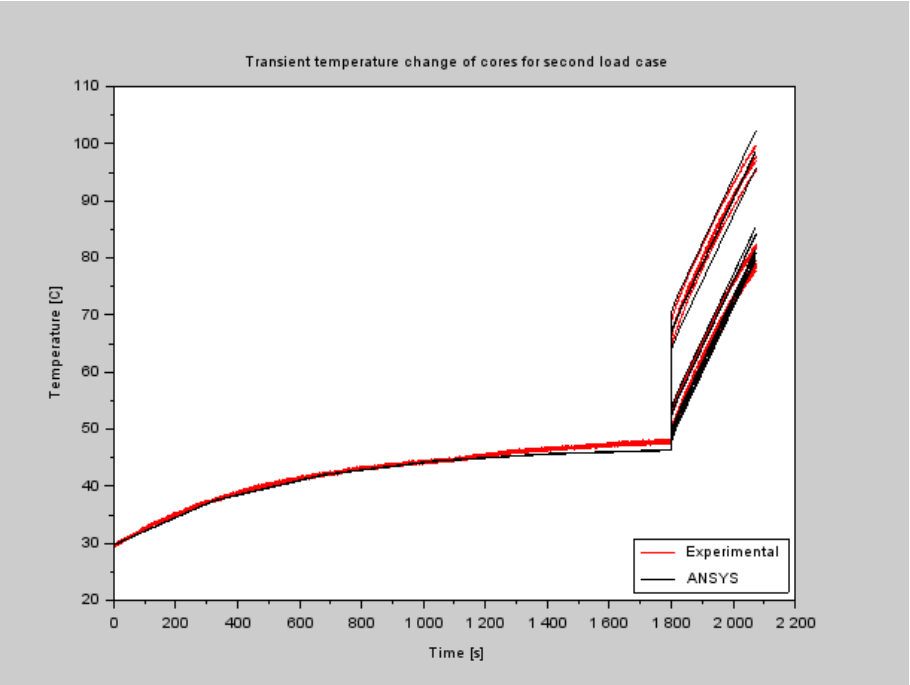


Figure 4.4: Transient temperature change for second load case

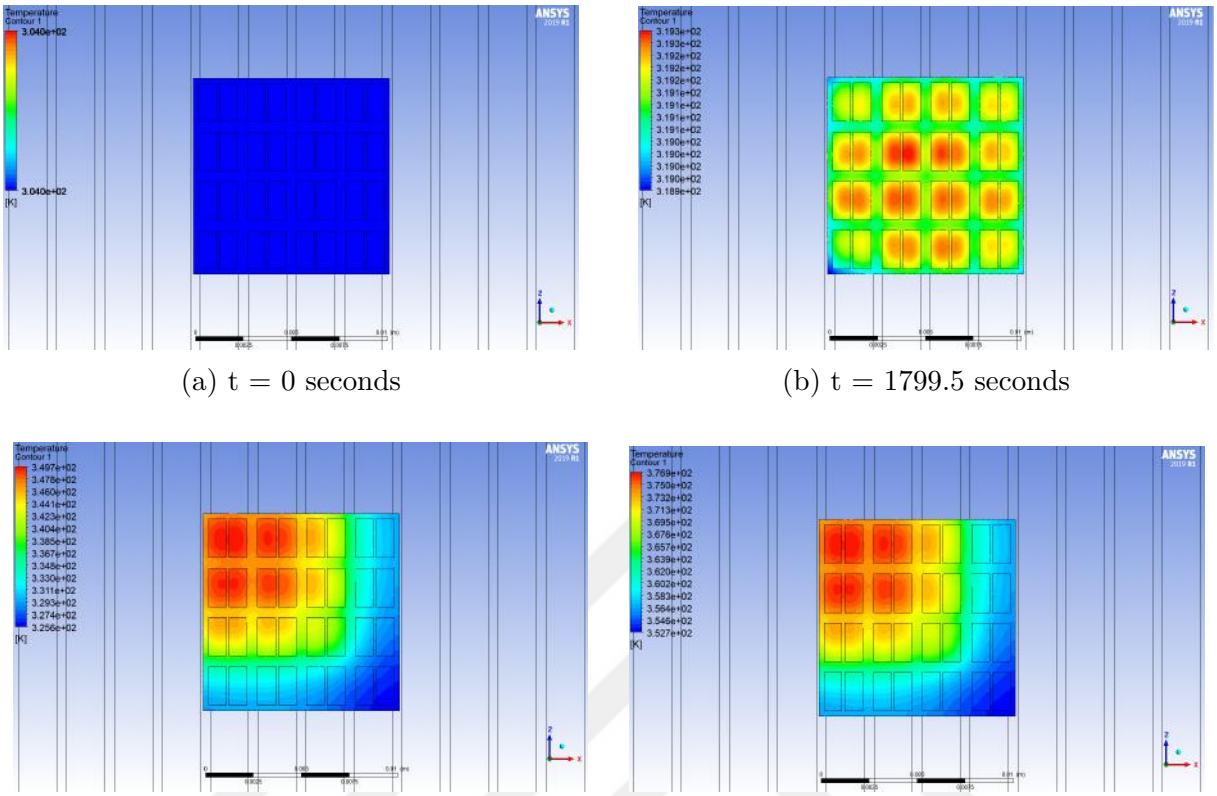
In Table 4.2, temperature values of the ANSYS simulation and the experimental setup are given together with the maximum error percentage between them. The maximum error between the experimental results and the ANSYS results occur on the third core at $t = 2073s$ as 0.997 %. The overall RMS error is calculated as 0.053 %.

Table 4.2: Temperature and error data for second load case in ANSYS

Cores	Experimental Temperatures [K] at t = 0s	ANSYS Temperatures [K] at t = 0s	Percent Error %	Experimental Temperatures [K] at t = 1799.5s	ANSYS Temperatures [K] at t = 1799.5s	Percent Error %
1	300.75	300.85	0.033	320.75	319.43	-0.412
2	300.85	300.85	0.000	320.95	319.47	-0.461
3	300.75	300.85	0.033	321.05	319.47	-0.492
4	300.75	300.85	0.033	320.75	319.45	-0.405
5	300.85	300.85	0.000	321.05	319.48	-0.489
6	300.75	300.85	0.033	321.05	319.52	-0.477
7	300.75	300.85	0.033	321.05	319.5	-0.483
8	300.85	300.85	0.000	320.95	319.47	-0.461
9	300.85	300.85	0.000	320.95	319.49	-0.455
10	300.85	300.85	0.000	321.05	319.5	-0.483
11	300.95	300.85	-0.033	320.95	319.5	-0.452
12	300.85	300.85	0.000	321.05	319.48	-0.489
13	300.85	300.85	0.000	321.05	319.41	-0.511
14	300.85	300.85	0.000	321.15	319.48	-0.520
15	300.85	300.85	0.000	321.15	319.49	-0.517
16	300.95	300.85	-0.033	321.15	319.45	-0.529

Cores	Experimental Temperatures [K] at t = 1803s	ANSYS Temperatures [K] at t = 1803s	Percent Error %	Experimental Temperatures [K] at t = 2073s	ANSYS Temperatures [K] at t = 2073s	Percent Error %
1	342.55	344.09	0.450	372.75	375.51	0.740
2	339.95	340.33	0.112	370.25	371.74	0.402
3	326.15	327.3	0.353	355.15	358.69	0.997
4	322.85	323.04	0.059	351.65	354.42	0.788
5	340.35	340.59	0.071	370.75	372.01	0.340
6	338.25	337.51	-0.219	368.45	368.92	0.128
7	325.65	326.09	0.135	354.85	357.48	0.741
8	322.85	322.66	-0.059	351.65	354.03	0.677
9	326.25	327.22	0.297	355.35	358.63	0.923
10	325.85	325.85	0.000	354.95	357.25	0.648
11	323.75	323.41	-0.105	352.55	354.8	0.638
12	322.65	322	-0.201	351.35	353.37	0.575
13	323.15	323.1	-0.015	351.85	354.52	0.759
14	322.95	322.75	-0.062	351.85	354.15	0.654
15	322.65	322.06	-0.183	351.35	353.45	0.598
16	322.45	321.45	-0.310	350.95	352.82	0.533

For the third load case, nine cores are fully loaded; its temperature distribution is obtained in ANSYS for specific times and it is given in Figure 4.5.



(a) t = 0 seconds

(b) t = 1799.5 seconds

(c) t = 1803 seconds

(d) t = 1892 seconds

Figure 4.5: Temperature distribution of microprocessor for third load case

In Figure 4.6, the transient temperature change obtained from the experimental setup and ANSYS are plotted and given for nine core case.

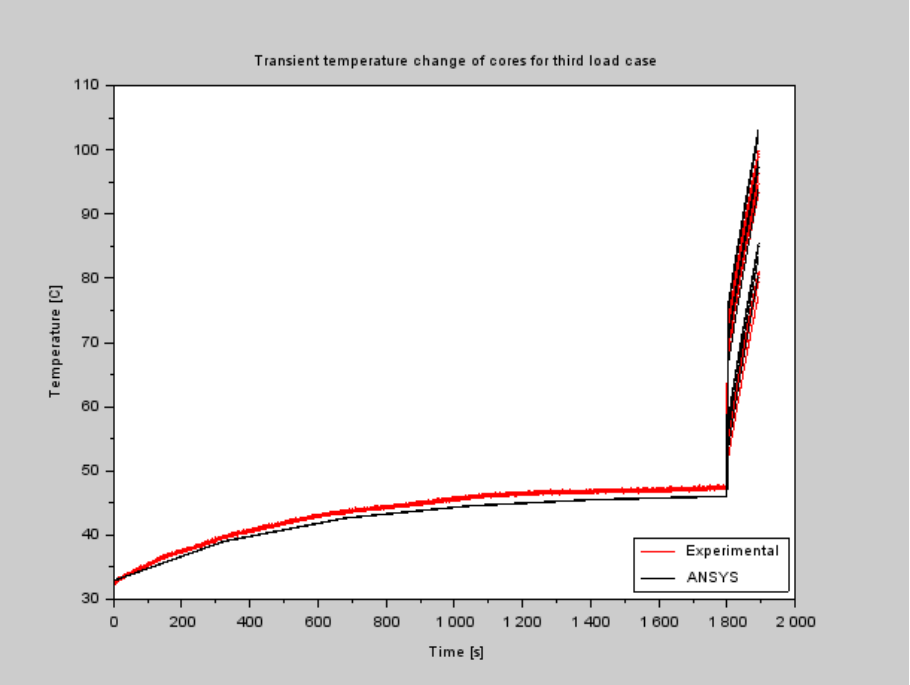


Figure 4.6: Transient temperature change for third load case

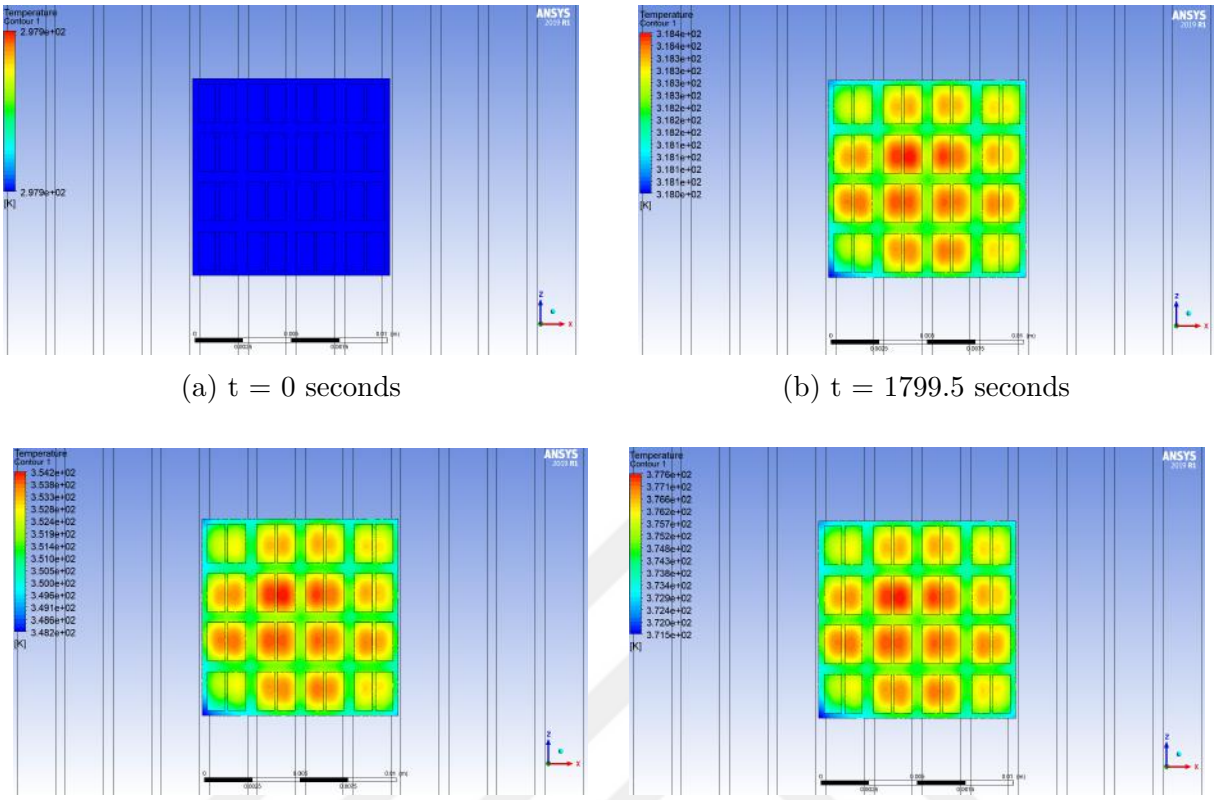
In Table 4.3, the temperature values of the ANSYS simulation and the experimental setup are given together with the maximum error percentage between them. The maximum error between the experimental results and the ANSYS results occur on the thirteenth core at $t = 1892s$ as 1.483 %. The overall RMS error is calculated as 0.079 %.

Table 4.3: Temperature and error data for third load case in ANSYS

Cores	Experimental Temperatures [K] at t = 0s	ANSYS Temperatures [K] at t = 0s	Percent Error %	Experimental Temperatures [K] at t = 1799.5s	ANSYS Temperatures [K] at t = 1799.5s	Percent Error %
1	303.95	304.05	0.033	320.15	319.16	-0.309
2	304.05	304.05	0.000	320.45	319.2	-0.390
3	303.95	304.05	0.033	320.45	319.2	-0.390
4	303.95	304.05	0.033	320.35	319.18	-0.365
5	303.95	304.05	0.033	320.35	319.21	-0.356
6	303.95	304.05	0.033	320.45	319.25	-0.374
7	304.05	304.05	0.000	320.45	319.23	-0.381
8	304.05	304.05	0.000	320.45	319.2	-0.390
9	304.05	304.05	0.000	320.35	319.22	-0.353
10	304.05	304.05	0.000	320.55	319.23	-0.412
11	303.95	304.05	0.033	320.45	319.22	-0.384
12	304.05	304.05	0.000	320.45	319.21	-0.387
13	304.05	304.05	0.000	320.45	319.14	-0.409
14	304.05	304.05	0.000	320.45	319.21	-0.387
15	304.05	304.05	0.000	320.45	319.22	-0.384
16	304.05	304.05	0.000	320.55	319.18	-0.427

Cores	Experimental Temperatures [K] at t = 1803s	ANSYS Temperatures [K] at t = 1803s	Percent Error %	Experimental Temperatures [K] at t = 1892s	ANSYS Temperatures [K] at t = 1892s	Percent Error %
1	345.65	349.14	1.010	372.05	376.36	1.158
2	345.75	348.47	0.787	372.55	375.68	0.840
3	342.35	344.11	0.514	369.45	371.31	0.503
4	328.25	331.31	0.932	354.05	358.48	1.251
5	345.15	348.69	1.026	371.55	375.9	1.171
6	346.15	348.02	0.540	372.95	375.23	0.611
7	342.45	343.54	0.318	369.55	370.73	0.319
8	328.35	330.86	0.764	354.25	358.03	1.067
9	341.45	344.25	0.820	367.95	371.46	0.954
10	342.45	343.22	0.225	369.15	370.42	0.344
11	339.65	339.86	0.062	366.45	367.05	0.164
12	327.15	329.24	0.639	352.95	356.41	0.980
13	327.95	331.38	1.046	353.35	358.59	1.483
14	328.05	330.94	0.881	353.85	358.15	1.215
15	326.95	329.36	0.737	352.75	356.55	1.077
16	324.85	326.54	0.520	350.35	353.7	0.956

Lastly for the fourth load case, all of the cores are fully loaded; its temperature distribution is obtained in ANSYS for specific times and it is given in Figure 4.7.



(a) t = 0 seconds

(b) t = 1799.5 seconds

(c) t = 1803 seconds

(d) t = 1844 seconds

Figure 4.7: Temperature distribution of microprocessor for fourth load case

In Figure 4.8, the transient temperature change obtained from the experimental setup and ANSYS are plotted and given for sixteen core case.

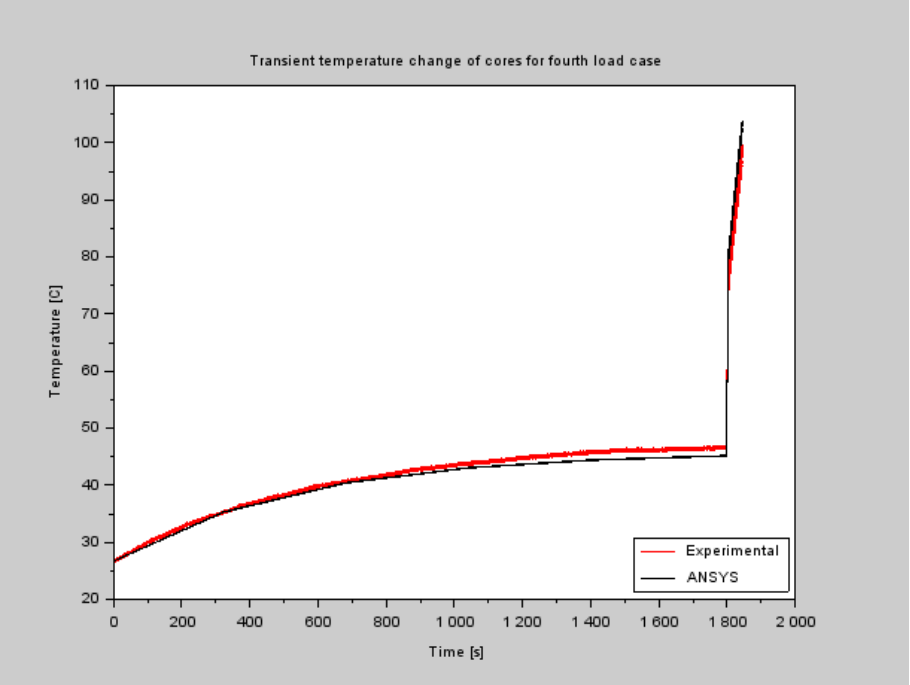


Figure 4.8: Transient temperature change for fourth load case

In Table 4.3, the temperature values of the ANSYS simulation and the experimental setup are given together with the maximum error percentage between them. The maximum error between the experimental results and the ANSYS results occur on the ninth core at $t = 1803s$ as 1.895 %. The overall RMS error is calculated as 0.13 %.

Table 4.4: Temperature and error data for fourth load case in ANSYS

Cores	Experimental Temperatures [K] at t = 0s	ANSYS Temperatures [K] at t = 0s	Percent Error %	Experimental Temperatures [K] at t = 1799.5s	ANSYS Temperatures [K] at t = 1799.5s	Percent Error %
1	297.75	297.85	0.034	319.55	318.26	-0.404
2	297.75	297.85	0.034	319.75	318.3	-0.453
3	297.75	297.85	0.034	319.75	318.3	-0.453
4	297.75	297.85	0.034	319.65	318.28	-0.429
5	297.85	297.85	0.000	319.85	318.31	-0.481
6	297.75	297.85	0.034	319.85	318.35	-0.469
7	297.75	297.85	0.034	319.85	318.33	-0.475
8	297.95	297.85	-0.034	319.85	318.3	-0.485
9	297.85	297.85	0.000	319.75	318.33	-0.444
10	297.75	297.85	0.034	319.95	318.34	-0.503
11	297.75	297.85	0.034	319.85	318.33	-0.475
12	297.85	297.85	0.000	319.75	318.32	-0.447
13	297.95	297.85	-0.034	319.65	318.24	-0.441
14	297.75	297.85	0.034	319.75	318.31	-0.450
15	297.85	297.85	0.000	319.75	318.32	-0.447
16	297.95	297.85	-0.034	319.85	318.28	-0.491

Cores	Experimental Temperatures [K] at t = 1803s	ANSYS Temperatures [K] at t = 1803s	Percent Error %	Experimental Temperatures [K] at t = 1844s	ANSYS Temperatures [K] at t = 1844s	Percent Error %
1	346.65	352.12	1.578	369.85	375.44	1.511
2	347.85	352.79	1.420	371.55	376.12	1.230
3	347.85	352.83	1.432	371.75	376.17	1.189
4	347.15	352.49	1.538	370.95	375.81	1.310
5	347.25	352.97	1.647	370.35	376.29	1.604
6	349.05	353.64	1.315	372.75	376.99	1.137
7	348.85	353.36	1.293	372.85	376.71	1.035
8	348.15	352.77	1.327	372.05	376.1	1.089
9	346.65	353.22	1.895	369.75	376.55	1.839
10	348.65	353.37	1.354	372.25	376.71	1.198
11	348.65	353.24	1.317	372.35	376.58	1.136
12	347.65	353.03	1.548	371.55	376.36	1.295
13	346.35	351.79	1.571	369.25	375.12	1.590
14	347.35	352.98	1.621	370.75	376.32	1.502
15	347.45	353.14	1.638	370.95	376.49	1.493
16	346.85	352.39	1.597	370.75	375.71	1.338

As it mentioned in section 4.2, the convective heat transfer coefficient h_{ext} is found iteratively by using the ANSYS simulation results, because, h_{ext} is the only unknown in the system. However, also the other constants used in the system are not exactly known, which means they can have a varying value between $\pm$ 50 % of their original value. This variation occurs due to the material impurities, processes applied on the material, and lack of precise knowledge about the used materials in the experimental setup. Additionally, the thickness of the rubber pad, and the active part of the microprocessor are not precisely known, but their dimensions can be estimated proportionally. The convective heat transfer coefficient significantly affects the steady-state temperature value and its effect given in Figure 4.9. As it can be seen from Figure 4.9, increasing h_{ext} decreases the steady-state and the final temperatures vice versa.

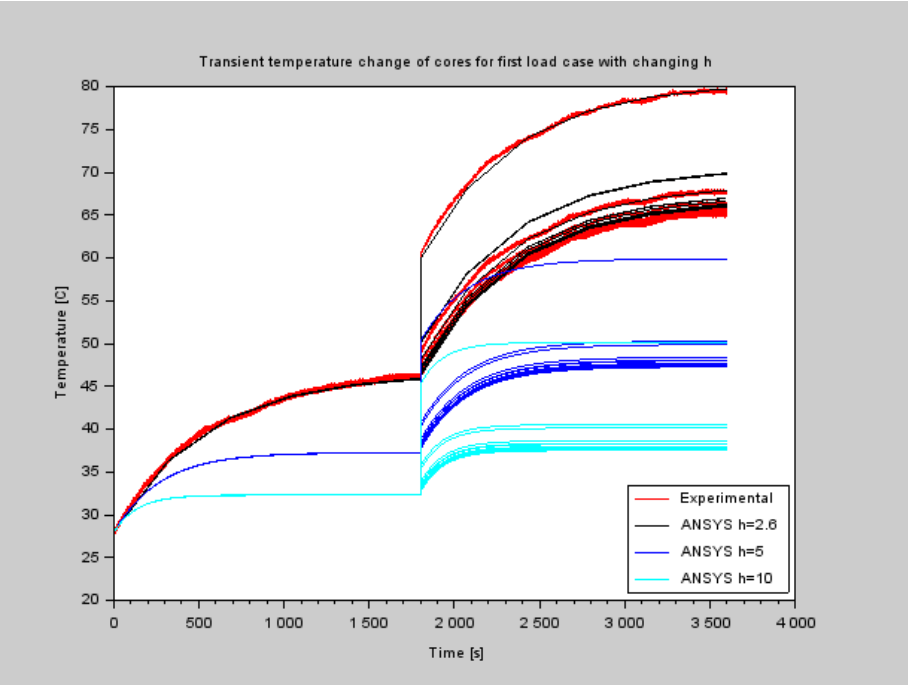


Figure 4.9: Transient temperature change for varying convective heat transfer coefficient

The thermal conductivity of the rubber pad significantly affects the sudden temperature rise and its effect given in Figure 4.10. As it can be seen from Figure 4.10, increasing k_{rb} decreases the magnitude of the temperature rise.

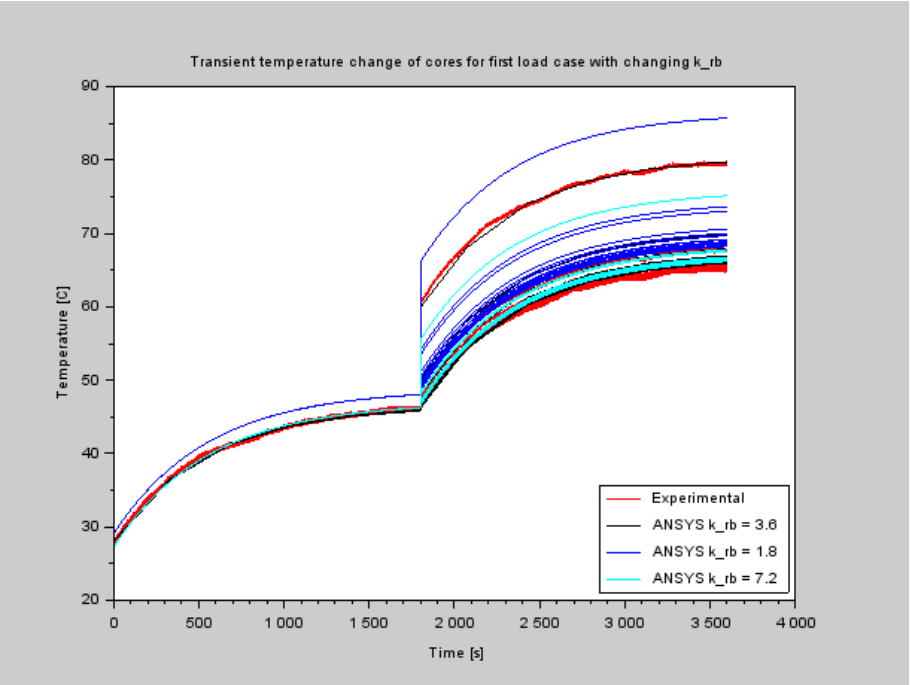


Figure 4.10: Transient temperature change for varying conduction heat transfer coefficient of rubber pad

The thermal conductivity of the silicon, which is used in the microprocessor, affects the heat exchange between the cores of the microprocessor, hence, it increases the temperature differences between the cores as its value decreased and its effect is given in Figure 4.11.

Thus, the convective heat transfer coefficient is found by changing other parameters in determined range to minimize the error between the ANSYS simulations and the experimental data. The reason of applying four different load cases, is to minimize this error globally for h_{ext} and the other constant parameters. By applying this approach, the system behavior under different load cases, which are not stated here, could be found more precisely.

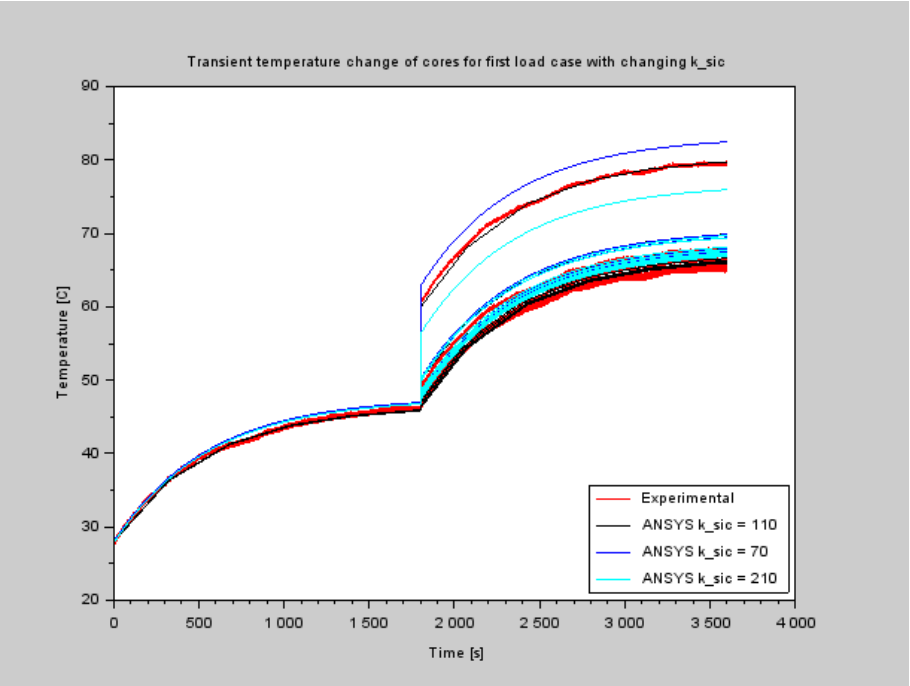


Figure 4.11: Transient temperature change for varying conduction heat transfer coefficient of microprocessor

After this procedure, all parameters of the system are found and the ANSYS simulations are realized. The main reason behind that approach is to find the temperature distribution of the heat sink fins' over the time which is not measured through the experiments. Considering the matching between the core temperatures of the experiment and the simulations, it can be said that, the approximation realized in model is correct; the fin temperatures found by the ANSYS simulation, which are under investigation, can be used as real data set with a minimum presence of error, since the experimental ones are not available. The temperature measurements on the fins are made by locating probes at the specified coordinates of the system, which can be seen in Figure 4.12. The choice of the selected fins is made according to their distance from the microprocessor heating region.

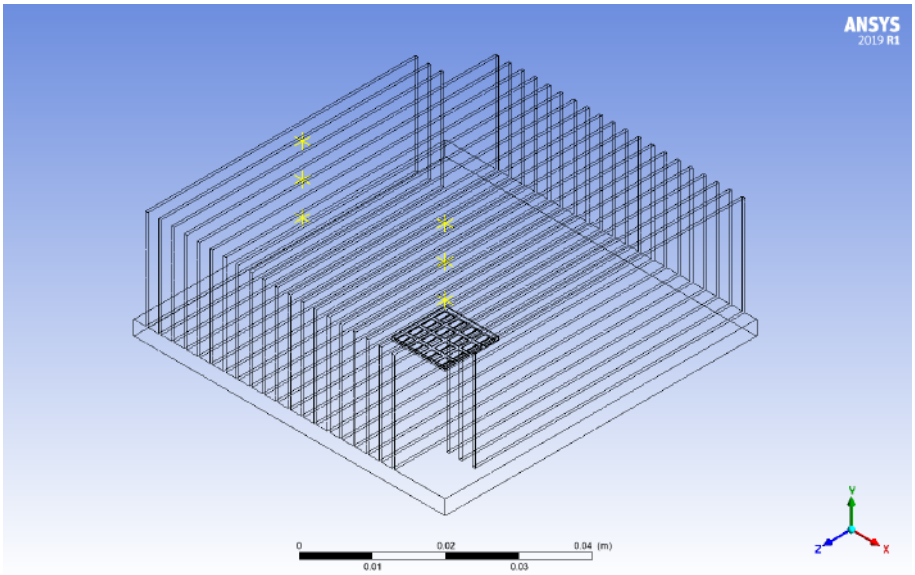


Figure 4.12: Selected points on fin structure

The temperature variation over time for the 1^{st} and the 12^{th} fins in the first load case are given in Figure 4.13 as a representative. For the same case, it is possible to see whole temperature distribution on the heat sink, at the end of experiment, in Figure 4.14. These fin temperature results obtained through the ANSYS simulations are used in the system design and simulation made in the Modelica as reference values for comparison.

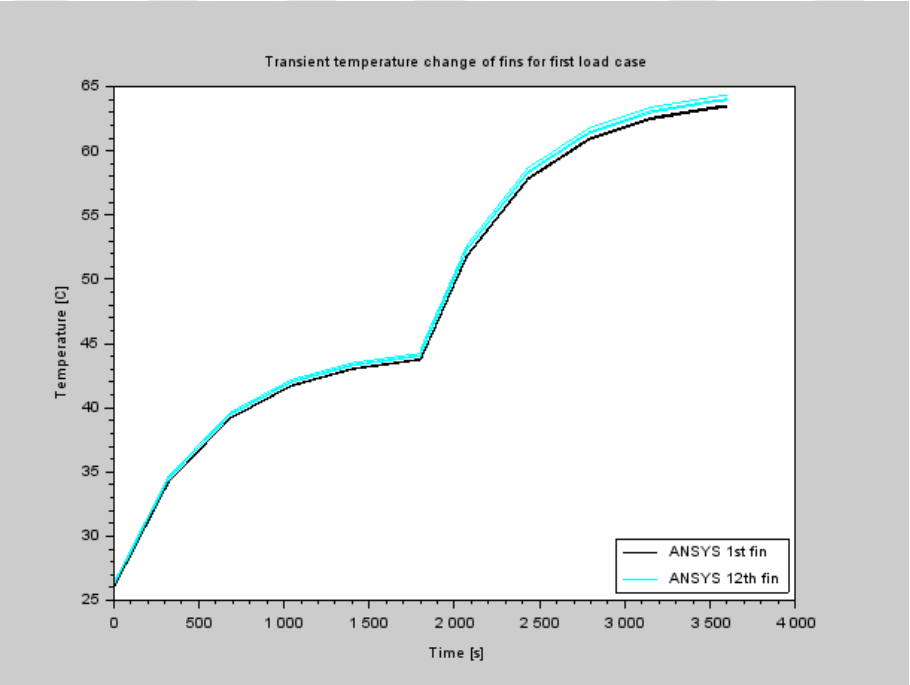


Figure 4.13: Transient temperature change on selected fin points

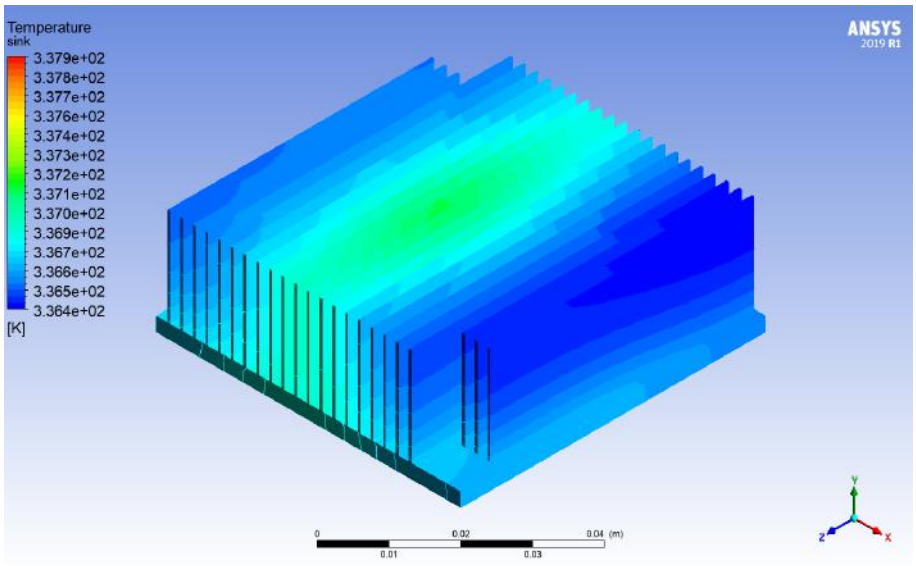


Figure 4.14: Final temperature distribution on fin structure for first load case

Heat Transfer Simulation in Modelica

The simulation in ANSYS already provides a verification data for the heat sink temperature distribution which is not known experimentally. Thus, the complete distribution throughout the microprocessor and the heat sink complex is now available for further usage.

It is important to note that ANSYS Fluent is a very well established tool for the heat transfer and the flow simulations. However, ANSYS contains other tools for mechanical, electrical, combustion modelling etc. which are not used for the experimental setup under examination. Even Fluent tool has some properties left untouched. In addition to that, the amount of mesh elements created during the meshing operation to avoid model divergence and to have a finer result, is 898.597 which significantly increases the number of variables and thus, the computational time. Lastly, it is not a common practice for companies to use ANSYS due to economic reasons. The power of the Modelica simulation lies in these facts. In this part, Modelica code to realize the simulation, the theory behind it and the results will be discussed. Implemented code is given in Appendix C.

5.1 Heat Transfer Theory

To model the heat transfer in the Modelica code it is necessary to revise the heat transfer theory to find the possible ways to introduce the heat transfer equations to the coding. The transient and the steady-state conduction and convection formulations are examined and found approaches are explained in this section. Consequently, some approaches had no significant contribution to model while the others partly or fully used in Modelica code.

5.1.1 Transient Formulations

In the transient heat transfer analysis, there are three main approaches to solve the heat equation in simplified forms by using some assumptions. The lumped capacitance and

the semi-infinite solid approaches are analytical ones used to determine the temperature distribution without the need of evaluating numerical analysis techniques for simple cases. The finite difference and the finite volume methods are numerical approaches to solve the temperature distribution problem.

5.1.1.1 Lumped Capacitance Analysis

This method is used to determine the transient temperature of a body when suddenly exposed to a uniform external temperature. With this method, the temperature along the body is considered equal and is changing only with time. To be able to apply this method, it is necessary to have a Biot number (Bi) smaller than 0.1 ($Bi << 1$). The calculated Biot number for the heat sink fins is equal to $1.64 * 10^{-6}$ from Equation 5.1 [4].

$$Bi = \frac{h_{ext} L_c}{k} \tag{5.1}$$

where h_{ext} is the convective heat transfer coefficient, k is the conductive heat transfer coefficient and L_c is the characteristic length of the rectangular fin which can be expressed as the perimeter over the cross sectional area in yz plane.

However, since the whole structure has a complex geometry and it is subjected to different temperatures from different boundaries and also there is no constant temperature boundary condition due to changing temperature of the microprocessor cores, it is not possible to apply this assumption to obtain an analytical solution.

The only contribution of this approach to the designed Modelica simulation is that it shows that, it is not necessary to divide the fins in y-direction to more than one element. It is because the conduction in x- direction is more important than the convection term in Biot number, so that the temperature distribution can be considered uniform along y-direction of the fin. The same argument could also be used for the z- direction, however, there are significant changes in the base temperature due to the heat contribution of the microprocessor, thus, it is unavoidable to divide the fins in meshes along that direction.

5.1.1.2 Semi-Infinite Solid Analysis

This method can be used for the temperature calculations of the objects which can be considered as semi-infinite. The object is exposed to a constant temperature or heat flux from its finite end, and, the heat flow occurs through its infinite part. Determining a solid as semi-infinite is related to its Fourier number (Fo) which is the dimensionless time with the formula given in Equation 5.2 [7].

$$Fo = \frac{\alpha t}{L^2} \tag{5.2}$$

When $Fo < 0.2$, solid can be considered as semi-infinite and the analytical solution for constant surface temperature is given in Equation 5.3 and 5.4 and can be applied to obtain the temperature distribution of the solid [7].

$$Bi = \frac{T(x,t) - T_s}{T_i - T_s} = erf(\frac{x}{2\sqrt{\alpha t}}) \tag{5.3}$$

$$q"_s(t) = \frac{k(T_s - T_i)}{\sqrt{\pi \alpha t}} \tag{5.4}$$

With increasing Fourier number, in fact with increasing time, the farthest parts of the solid starts to change its temperature, thus, the solution becomes ambiguous. The fins of the heat sink have a small dimension in x- direction. By using the formula given in Equation 5.2, time of the experiment should be between t=0 and t=0.001 to satisfy $Fo < 0.2$ condition. For $Fo > 0.2$, there exist analytical and numerical solutions for different geometries. However, these solutions assume that body is subjected to a constant temperature or a heat flux suddenly from its boundaries, this condition also presents problem when $Fo < 0.2$. Thus, semi-infinite solid approach is not applicable to the current modelling.

5.1.1.3 Finite Difference Method

This method is used to solve the transient heat transfer problems when the analytical solutions are not applicable due to complex geometries, and boundary conditions. Evaluating the partial derivatives in Equation 4.7 in discrete time and space, the temperature distribution inside the material could be found. There are two approaches to solve discretized time and space equations; the first one is explicit, which uses the forward difference method, and the second one is implicit, which uses the backward difference method.

In the explicit method, the temperature value of the position in the future time step is calculated by using present temperature values. Thus, each equation is solved separately for different locations. The formulation of the explicit method is given in Equation 5.5 for a generic three-dimensional element which is located inside the material volume, hence, formulation has no convective heat transfer terms. The dimensionless time constant takes the form shown in Equation 5.6 [4].

$$\frac{1}{\alpha}\frac{T^{p+1}_{m,n} - T^{p}_{m,n}}{\Delta t} = \frac{T^{p}_{m+1,n} + T^{p}_{m-1,n} - 2T^{p}_{m,n}}{\Delta x^2} + \frac{T^{p}_{m,n+1} + T^{p}_{m,n-1} - 2T^{p}_{m,n}}{\Delta y^2} \tag{5.5}$$

$$Fo = \frac{\alpha \Delta t}{\Delta x^2} \tag{5.6}$$

In the implicit method, the temperature value of the specific position in the future time step depends on other positions temperature values in the future time step. In this approach, only known value is the present time step temperature of each position element. Thus, to solve the equation for the future time step temperature values, a set of equations must be solved simultaneously by using Gauss-Seidel iteration or matrix inversion method on equations. The formulation of the implicit method is given in Equation 5.7 for a generic interior element [4].

$$\frac{1}{\alpha}\frac{T_{m,n}^{p+1} - T_{m,n}^{p}}{\Delta t} = \frac{T_{m+1,n}^{p+1} + T_{m-1,n}^{p+1} - 2T_{m,n}^{p+1}}{\Delta x^2} + \frac{T_{m,n+1}^{p+1} + T_{m,n-1}^{p+1} - 2T_{m,n}^{p+1}}{\Delta y^2} \tag{5.7}$$

As it is stated in Chapter 2, FVM, which is derived from FDM, is used to solve the heat transfer equations in discrete time and space. However, this method offers more generic solution for the complex three-dimensional geometries with different mesh sizes and materials. Thus, it is chosen as a solution model for the Modelica simulation.

5.1.2 Steady-State Formulations

The fin design is made calculating the steady-state heat transfer to the medium. The temperature of the base is assumed to have a constant value and by knowing the temperature of the medium, the overall temperature distribution on the fins' x-direction could be found. The energy input of the fins is dissipated along the x-direction from the surface to the medium via the convective heat transfer. There are two Biot numbers used in the formulation: $Biot_{fin}$ determines the conduction in the transverse direction, which is normal to x-direction of the fins and $Biot_{ax}$ which determines the conduction in x-direction. Since, $Biot_{fin}$ is much smaller than 1, the temperature variations along y- and z- direction can be considered negligible for the constant base temperature. However, the base temperature of the heat sink is not constant along z- axis. As it is stated in the Section 5.1.1.1, it is unavoidable to divide the heat sink base element in z- direction. Note that, it is unnecessary to divide the fin elements more than the heat sink base elements in z-direction due to small $Biot_{fin}$. It is because, the only reason of the temperature difference in the z- direction of the fins, are due to the lateral conduction caused by the different input temperatures of the base elements but not due to the contribution of the convective terms.

Steady-State formulation has two different boundary conditions at the tip of the fin. The first one assumes the insulated fin tip boundary condition, which implies that the

temperature of the fin tip is equal to the ambient temperature and its formulation is given in Equation 5.8 [5]. The second one assumes the uninsulated fin tip boundary condition which implies that the temperature of the fin tip is different than the ambient temperature, thus, there is a convective heat transfer term at the tip of the fin, which has a different convective heat transfer coefficient than the side surfaces of the fin. Also, its formulation is given in Equation 5.9 [5]. However, the convective heat transfer coefficient h_L in Equation 5.9, $Biot_{ax}$ cannot be determined analytically. Thus, this term is replaced with the overall convective heat transfer coefficient h_{ext}.

$$\begin{aligned} (T - T_\infty)_{x=0} &= T_0 + T_\infty \\ \frac{\mathrm{d}(T - T_\infty)}{\mathrm{d}x}\bigg|_{x=L} &= 0 \end{aligned} \tag{5.8}$$

$$\begin{aligned} (T - T_\infty)_{x=0} &= T_0 + T_\infty \\ -kA\frac{\mathrm{d}(T - T_\infty)}{\mathrm{d}x}\bigg|_{x=L} &= h_L(T - T_\infty)_{x=L} \end{aligned} \tag{5.9}$$

The Theta (Θ) formulations for the insulated and the uninsulated tip is given in Equation 5.11 [8] and Equation 5.12 [8] respectively. As it can be seen, the theta values of these two approaches does not different from each other, since, the convective heat transfer term from the tip is negligible because of the small convective surface. The Θ distribution obtained from Equation 5.11 and Equation 5.12 give a result when the steady-state temperature is achieved for the heat sink base, which is T_0; hence, it is not possible to assume constant spatial distribution of Θ on the fins over time. However, this formulation can be used to compare the steady-state fin tip temperatures obtained in simulation with the theoretical value. In Equation 5.10 [8], m is a group appearing in the heat transfer differential equation. A_c stands for the cross-sectional area of the fin and P is the perimeter of that area.

$$m = \sqrt{\frac{h_{ext}P}{kA_c}} \tag{5.10}$$

$$\Theta = \frac{T - T_\infty}{T_0 - T_\infty} = \frac{\cosh mL(1 - \xi)}{\cosh mL} \tag{5.11}$$

$$\Theta = \frac{T - T_\infty}{T_0 - T_\infty} = \frac{\cosh mL(1 - \xi) + (Bi_{ax}/mL)\sinh mL(1 - \xi)}{\cosh mL + (Bi_{ax}/mL)\sinh mL} \tag{5.12}$$

5.2 Mesh Regions in Simulation

The whole microprocessor and the heat sink complex are divided to various mesh regions to implement the heat transfer equations. These regions have different sized mesh elements and materials.

The heat sink is divided to a base plate and twenty-three fins. Also, the base plate is divided into twenty-three elements in y- direction. The main reason of choosing this value is to have one fin finite element over one base finite element. The number of fins is a prime number, thus, dividing the heat sink base to a different number than twenty-three, results in changing number of fin finite elements on top of each base element which complicates the analysis. In addition to that, both fin and base elements are divided into three in z-direction, to have a more precise solution by assuring a constant base temperature under each fin element, which has a contact with a base element. The temperature data for the base elements are kept in $T_b[j, n, m]$, where j, n and m are corresponding indices in y-, z- and x- directions. The fin temperatures are kept as $T[i, n, m]$, where i, n and m are corresponding indices in x-, z- and y- directions. The meshed model for the one of the twenty-three fin and heat sink base are given in Figure 5.1 and Figure 5.2 respectively.

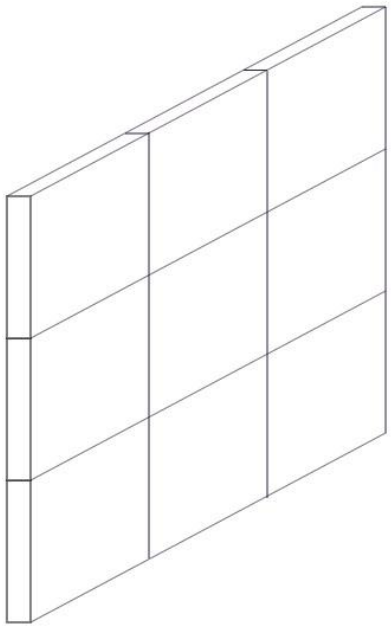


Figure 5.1: Meshing realized on the fin region

The rubber pad is considered as two-dimensional plate due its small dimension in x-direction, and it is divided into sixteen elements in yz plane to better represent the heat coming from the microprocessor. The temperature data of these elements are registered in $T_r[w, c]$, where w and c represent the position in y- and z- directions. The meshed model of the rubber pad is given in Figure 5.3.

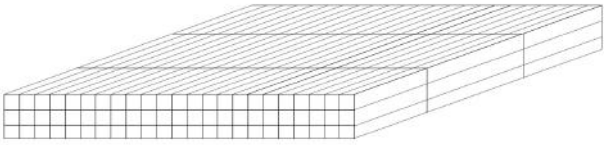


Figure 5.2: Meshing realized on the heat sink base region

Figure 5.3: Meshing realized on the rubber pad region

The most complicated part is the meshing of the microprocessor due to its complex geometry compared to other regions. The silicon microprocessor is divided to the passive and the active parts to obtain more realistic results in the simulation. The passive part which has 0.525mm thickness, is dived to five elements in x- direction and sixteen elements in yz plane as rectangular boxes. The temperature data is stored in $T_{cp}[q, u, q1]$ array, with q, u and $q1$ representing y-, z- and x- direction respectively. Additionally, the meshed model of the passive part of the microprocessor is given in Figure 5.4.

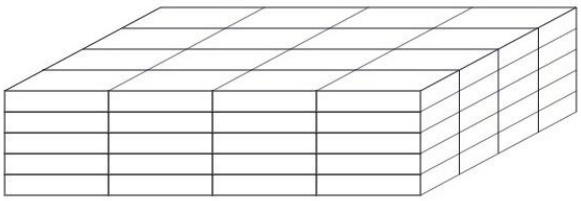


Figure 5.4: Meshing realized on the passive part of the microprocessor

The active part which has 0.1mm thickness, is considered to have only thirty-two resistors and the other regions staying between these resistors are omitted to decrease the number of the state variables. The input voltage and the current are supplied to the active parts of the silicon through these resistors which are represented as rectangular elements; their temperature data are kept in $T_{ca}[q, u, n]$ array, with q, u representing the core position y- and z- direction and n representing each resistor element inside the core. Since the temperature sensors are located between two resistors of every core, the temperature value of them are calculated by using the conduction equation. The Modelica code is given in Figure 5.5.

```
// Temperature of the Sensor Measurements Calculations
for q in 1:yc loop

  for u in 1:zc loop

    Ts[q,u] = (Tca[q,u,1]+Tca[q,u,2])/2;

  end for;

end for;
```

Figure 5.5: Sensor temperatures introduced in Modelica code

By assuming a zero-capacitance value for each volume containing sensor, it is possible to just calculate the temperatures without creating new mesh regions which increases the number of the state variables; these temperature data are stored in $T_s[q, u]$ array, where q and u are position indices in y- and z- direction. $T_s[q, u]$ values becomes arithmetic mean of two neighboring resistors by using this approach. The mesh structure for one resistor couple of the active part is given in Figure 5.6.

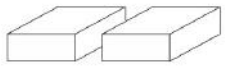


Figure 5.6: Meshing realized on the active part of the microprocessor

5.3 Energy Input of the Simulation Model

The energy input of the system appears on the right-hand side of the differential equations belonging to the resistor element temperature $T_{ca}[q, u, n]$, as an additional heat generation term. Two heat generation terms are introduced to the Modelica code as $W[q, u, n]$ and $W_2[q, u, n]$, having the same position indices with $T_{ca}[q, u, n]$ array, to realize the changing power input to the microprocessor. Each term of $W[q, u, n]$ and $W_2[q, u, n]$ which are represented by 4x4x2 arrays, are calculated from the voltage and the current data of the experiments made and added as Watts in the equations. The total number of array elements is thirty-two which is equal to the number of resistors. The added terms in equation for first load case for the first resistor can be seen in Figure 5.7.

```
for q in 1:yc loop

  for u in  1:zc loop

    for n in 1:2 loop

      if time <= 1800.0 then

        if q==1 and u==1 and n==1 then
          rhosic*VCA*Csic*der(Tca[q,u,n]) = - (1/(dcx/
(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/
(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];
                              :
      // For 1 Core
      elseif time > 1800.0 and q==1 and u==1 and n==1 then
       rhosic*VCA*Csic*der(Tca[q,u,n]) = - (1/(dcx/
(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+d*/sy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/
(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];
```

Figure 5.7: Power input introduced in Modelica code

The $W[q,u,n]$ terms in the resistor heat equations are equal to 0.093 Watt while, the $W_2[q,u,n]$ terms are equal to 1.5258 Watt for fully activated cores. After 1800 seconds some cores takes the value of $W_2[q,u,n]$ while, other cores have still $W[q,u,n]$ as input power.

5.4 Conduction Modelling

The heat transfer equation in its simplest form in Equation 4.7, is a non-homogeneous partial differential equation. The Modelica solver can evaluate the time derivatives on the left-hand side of this equation, however, it needs to have an information about the space derivatives of the temperature appearing on the right-hand side. As the whole body is divided into the meshes, these derivatives must be introduced as discretized formulas [1].

5.4.1 Conduction in a Mesh Region

The conduction formulation for a generic cubic element is realized through the FVM. Thinking the whole element is at the same temperature, the heat transfer at the boundaries of that element are formulated. As the temperature gradient is zero all over the element, Equation 4.7 can be integrated in the volume to get Equation 5.13. The constant volumetric heat source term S_h becomes the heat input Q_{hg} by multiplying with the volume V. The terms A_{yz}, A_{xz}, A_{xy} in Equation 5.13 are the cross-sectional areas in x-,y- and z- directions respectively.

$$V\frac{\delta T}{\delta t} = \alpha(A_{yz}\frac{\delta T}{\delta x} + A_{xz}\frac{\delta T}{\delta y} + A_{xy}\frac{\delta T}{\delta z}) + \frac{\alpha}{k}Q_{hg} \tag{5.13}$$

The discretization of space derivatives is made through the subtraction of the neighboring elements, which is actually the first order term of the Taylor expansion of that derivative. By also, multiplying both sides of Equation 5.13 with ρc_p, the final form of the conduction equations used in the Modelica code, are obtained. The introduced terms in the code for the conduction in the passive part of the microprocessor for a generic interior element can be seen in Figure 5.8.

```
            else
               rhosic*VC*Csic*der(Tcp[q,u,ql]) =
(- ksic*CCSA*(Tcp[q,u,ql]-Tcp[q-1,u,ql])/dcz -
ksic*CCSA*(Tcp[q,u,ql]-Tcp[q+1,u,ql])/dcz -
ksic*CCSA*(Tcp[q,u,ql]-Tcp[q,u-1,ql])/dcz -
ksic*CCSA*(Tcp[q,u,ql]-Tcp[q,u+1,ql])/dcz -
ksic*CSA*(Tcp[q,u,ql]-Tcp[q,u,ql+1])/dcx -
ksic*CSA*(Tcp[q,u,ql]-Tcp[q,u,ql-1])/dcx);
```

Figure 5.8: Conduction terms introduced in Modelica code

5.4.2 Conduction with Neighboring Region

As the energy transfer within time must be equal for a boundary between the two elements belonging to different regions, there must be an equivalent conduction heat transfer coefficient on these boundaries. Disregarding that fact, results in unbalanced energy transfer through the boundary due to changing mesh size and conduction coefficient. Since the temperature of the cells are in fact their middle point temperatures, the equivalent conduction coefficient can be found by using Equation 5.14; the corresponding conduction schematic is given in Figure 5.9.

$$Q = k_1 A\frac{(T_1 - T_w)}{L_1} = k_2 A\frac{(T_w - T_2)}{L_2} \tag{5.14}$$

By collecting k and L terms at the left-hand side, subtracting two equations and leaving Q term alone, Equation 5.15 is obtained.

$$Q = \frac{1}{(\frac{L_1}{k_1} + \frac{L_2}{k_2})}A(T_1 - T_2) \tag{5.15}$$

The implemented formulation in the code can be seen in Figure 5.10.

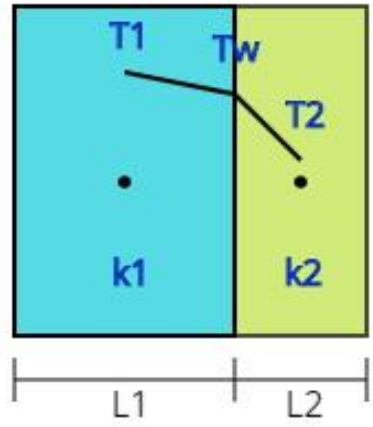


Figure 5.9: Conduction schematic between neighboring regions

```
// Rubber pad heat transfer equations
for w in 1:yr loop

  for c in  1:zr loop

    if w==1 and c==1 then
      rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w+1,c])/dcz -
krb*CCSAR*(Tr[w,c]-Tr[w,c+1])/dcz - (1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-
Tcp[w,c,xc]) + Qo[w,c];
```

Figure 5.10: Conduction terms for neighboring regions introduced in Modelica code

5.5 Convection and Radiation Modelling

The temperature rise occurs, due to Watt terms added to the resistor equations as an input term. The convective and the radiative terms can be considered as an output of the system. Due to the presence of these terms, the system reaches to a steady-state equilibrium temperature value for specific Watt term at a certain time. These boundary conditions are introduced to the faces of the heat sink, except the bottom surface which has contact with rubber pad. These phenomena occur due to the temperature differences between the fluid medium and the heat sink surfaces (which are solid). For a constant ambient temperature, these terms become more significant due to the increasing temperature of the heat sink surfaces. The convective heat transfer coefficient value found in the ANSYS simulation model is also used in the Modelica simulation by decreasing its value 7.7 %. The emissivity value of copper is found as 0.03 and used in the Modelica model to introduce the radiative heat loss, which occurs at the heat sink surfaces. However, the radiative heat loss is significantly smaller than the convective heat loss due to the low emissivity value of copper and the small temperature difference between the fin surfaces and the medium. For instance, the contribution of the radiative heat loss takes a maximum value of 8.4 % of the total heat dissipation to the ambient for first load case at the 3600 seconds. Thus, the radiative heat loss term is neglected in Section 5.7 while

realizing the Modelica simulations. The introduced formulation of the convective and the radiative heat transfer in the fin elements, is modelled in the code as shown in Figure 5.11.

```
                else
                  rho*V*C*der(T[i,n,m]) = (- h*SAM*(T[i,n,m]-Tamb)
/*- sigma*e*SAM*((T[i,n,m])^4-(Tamb)^4)*/) - k*CSAX*(T[i,n,m]-
T[i+1,n,m])/dx - k*CSAX*(T[i,n,m]-T[i-1,n,m])/dx - k*CSAZ*
(T[i,n,m]-T[i,n-1,m])/dz - k*CSAZ*(T[i,n,m]-T[i,n+1,m])/dz;
```

Figure 5.11: Convection and terms introduced in Modelica code

5.6 Mapping between Rubber Pad and Heat Sink Base

One of the most complex things in the Modelica simulation, is designing the mapping between non-overlapping mesh surfaces of the rubber pad and the heat sink. This mapping process is created to realize the conductive heat transfer terms between the rubber pad and the heat sink base element. Since, the heat sink base is divided to twenty-three elements and the rubber pad is divided to four elements in y-direction, the problem depends only on one parameter which is the element size of the heat sink base in z-direction. For the simplicity, the element size of the heat sink base in z-direction is chosen to have a larger value than element size of the rubber pad in z-direction. This decision is made by comparing the ANSYS simulations and the Modelica model results. Having smaller elements in z-directions does not improve the precision of the solution obtained.

As it is mentioned above, the heat sink base and the rubber pad divided into the constant number of elements in y-direction so that, the relation between these two depends their constant start and end positions. Due to the division made on parts, the elements of the heat sink base between the eleventh and the thirteenth, overlap with two rubber pad elements in y-direction. However, the tenth and the fourteenth element of the heat sink base only overlap with a single rubber pad element. This overlapping areas are shown in Figure 5.12. Finally, all other elements of heat sink base in y-direction does not have any contact region with the rubber pad elements.

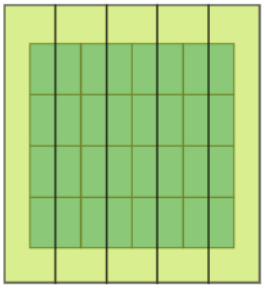
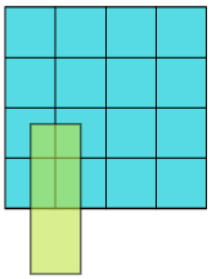
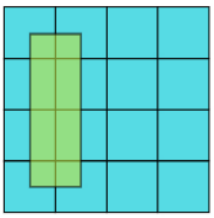


Figure 5.12: Convection and terms introduced in Modelica code

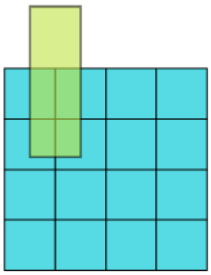
In z-direction, four different cases are possible to model the heat transfer from the heat sink base to the rubber pad. These cases of different overlapping regions occurring due to the various starting and ending positions of the elements in z-direction, are shown in Figure 5.13.



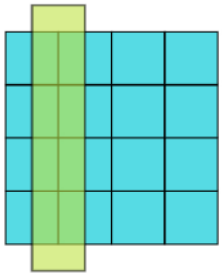
(a) Type 1: start position outside, end position inside



(b) Type 2: start position inside, end position inside



(c) Type 3: start position inside, end position outside



(d) Type 4: start position outside, end position outside

Figure 5.13: Heat sink base element according to its position on rubber pad element

The heat transfer term evolving due to these four cases, are introduced in the equations as $Q[j,n]$ terms where j and n are the position indices corresponding y- and z- directions respectively for the heat sink base elements. In Modelica, to create a more accurate model in z-direction another heat transfer matrix is introduced and it is named as $Q_1[j,i,n]$,where i is the position indice of each core element in z- direction and j and n indices are the same as $Q[j,n]$. Each term of $Q[j,n]$ matrix is calculated as the sum of $Q_1[j,i,n]$ elements with the corresponding j and n indices. This matrix represents the heat terms coming from each rubber pad element in z-direction to the heat sink base element. If there is no contact between the heat sink base and the rubber pad elements, the corresponding term of $Q_1[j,i,n]$ takes the value of 0 W. The representative scheme of the matrix relation between $Q[j,n]$ and $Q_1[j,i,n]$ is given in Figure 5.14. Finally, the introduction of $Q[j,n]$ terms in the heat sink base equations is stated in Figure 5.15.

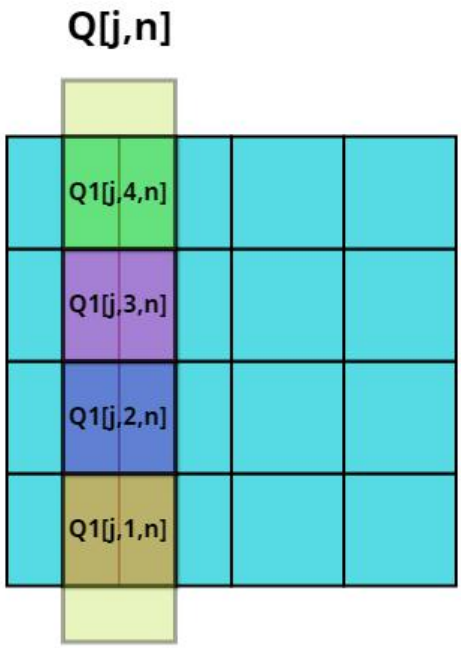


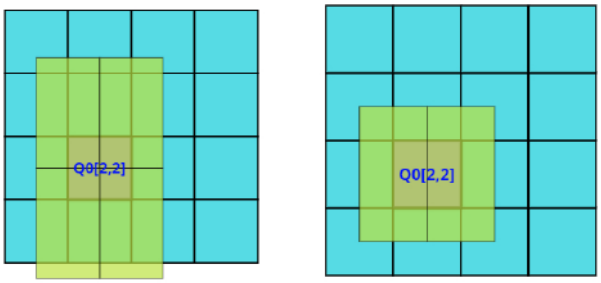
Figure 5.14: Schematic of relation between Q and Q_1 terms in Modelica code

```
            elseif m==1 then

              rho*VB*C*der(Tb[j,n,m]) = - k*CSABX*(Tb[j,n,m]-
Tb[j,n,m+1])/dbx - k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-
Tb[j,n+1,m])/dbz - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz +
Q[j,n];
```

Figure 5.15: Q terms introduced in Modelica code

On the other hand, $Q_0[j,i]$, where j and i are the position indices of each core element in y- and z- direction, represents the heat transfer term from the rubber pad element to the heat sink base element. There are two cases to realize this heat transfer term due to the start and the end positions of the heat sink base element in z-direction; these cases are shown in Figure 5.16. The introduction of $Q_0[j,i]$ terms in the rubber pad equations is stated in Figure 5.17.



(a) Type 1: Rubber pad element is connected to four base elements

(b) Type 2: Rubber pad element is connected to two base elements

Figure 5.16: Rubber pad element according to its position on heat sink base element

```
        else
           rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w-1,c])/
dcz - krb*CCSAR*(Tr[w,c]-Tr[w+1,c])/dcz - krb*CCSAR*(Tr[w,c]-Tr[w,c-1])/
dcz - krb*CCSAR*(Tr[w,c]-Tr[w,c+1])/dcz - (1/(dcx/(2*ksic)+(drx/
(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];
```

Figure 5.17: Q_0 terms introduced in Modelica code

It can be seen that the sum of $Q_0[j,i]$ terms is equal to sum of $Q[j,n]$ terms in magnitude for any time step during the simulation. In Table 5.1, the Modelica results for $Q[j,n]$ and $Q_0[j,i]$ at $t = 3600s$ are given to clarify this equality. In both tables, the sum of Watt terms are equal to 5.8414 Watt in magnitude. This result is obtained due to the conservation of the total energy transferred through the interface between the rubber pad and the heat sink base.

Table 5.1: Q and Q_0 values for first load case in Modelica

$Q[j,n]$	n=2	$Q_0[j,i]$	i=1	i=2	i=3	i=4
j=10	1.2203	j=1	-1.4207	-0.5506	-0.2985	-0.2304
j=11	1.9536	j=2	-0.5866	-0.3594	-0.2489	-0.2108
j=12	1.2093	j=3	-0.3231	-0.2627	-0.2213	-0.2038
j=13	0.9660	j=4	-0.2577	-0.2372	-0.2192	-0.2104
j=14	0.4922					

5.7 Simulation Results in Modelica

The results obtained by the realized simulations through the Modelica code are discussed in this section. The microprocessor core temperature results are compared to the ones obtained from the experiment and the ANSYS simulation, for each experimental case. The simulation time step for the Modelica simulation is chosen as 0.1 seconds to have the same temperature data with the experimental setup and to supply the sudden voltage value in a more precise manner. The procedure realized in Section 4.3 to represent the simulation results, is also followed for the Modelica results in this section.

In the first case, having only one core fully energized at 1800 seconds, the transient temperature change obtained from the experimental measurements, and the ANSYS and the Modelica simulations are plotted in Figure 5.18.

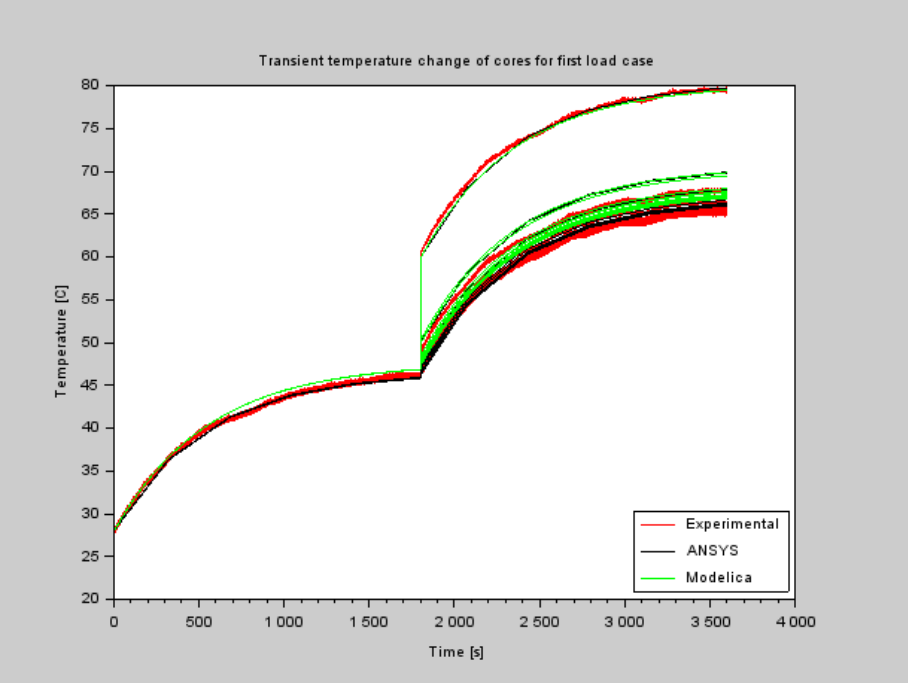


Figure 5.18: Transient temperature change of microprocessor cores for first load case

In Table 5.2, the temperature values of the important time steps mentioned in Section 4.3, obtained through the Modelica and the experimental results are given. The maximum error between the experimental results and the Modelica results occurs on the fifth core at $t = 3600s$ as 0.688 %. By using given time step errors of the microprocessor, the overall error in the Modelica simulation is calculated by using Root Mean Square (RMS) method and which is equal to 0.042 %.

Table 5.2: Temperature and error data for first load case in Modelica

Cores	Experimental Temperatures [K] at t = 0s	Modelica Temperatures [K] at t = 0s	Percent Error %	Experimental Temperatures [K] at t = 1799.5s	Modelica Temperatures [K] at t = 1799.5s	Percent Error %
1	299.15	299.15	0.000	319.05	319.986	0.293
2	299.15	299.15	0.000	319.25	319.986	0.231
3	299.15	299.15	0.000	319.25	319.986	0.231
4	299.05	299.15	0.033	319.25	319.986	0.231
5	299.15	299.15	0.000	319.25	320.014	0.239
6	299.05	299.15	0.033	319.35	320.014	0.208
7	299.05	299.15	0.033	319.35	320.014	0.208
8	299.15	299.15	0.000	319.25	320.014	0.239
9	299.35	299.15	-0.067	319.15	320.015	0.271
10	299.15	299.15	0.000	319.35	320.015	0.208
11	299.05	299.15	0.033	319.25	320.015	0.240
12	299.25	299.15	-0.033	319.35	320.015	0.208
13	299.25	299.15	-0.033	319.25	319.987	0.231
14	299.15	299.15	0.000	319.25	319.987	0.231
15	299.15	299.15	0.000	319.25	319.987	0.231
16	299.25	299.15	-0.033	319.15	319.987	0.262

Cores	Experimental Temperatures [K] at t = 1803s	Modelica Temperatures [K] at t = 1803s	Percent Error %	Experimental Temperatures [K] at t = 3600s	Modelica Temperatures [K] at t = 3600s	Percent Error %
1	333.65	333.039	-0.183	352.65	352.601	-0.014
2	322.15	323.037	0.275	340.85	342.6	0.513
3	319.85	321.129	0.400	338.65	340.691	0.603
4	319.45	320.736	0.403	338.15	340.298	0.635
5	322.25	323.434	0.367	340.65	342.993	0.688
6	320.65	321.556	0.283	339.45	341.115	0.490
7	319.85	320.863	0.317	338.65	340.422	0.523
8	319.55	320.674	0.352	338.35	340.233	0.557
9	319.85	321.218	0.428	338.55	340.772	0.656
10	320.05	320.832	0.244	338.65	340.385	0.512
11	319.65	320.625	0.305	338.35	340.179	0.541
12	319.55	320.554	0.314	338.35	340.108	0.520
13	319.65	320.635	0.308	338.35	340.184	0.542
14	319.55	320.538	0.309	338.35	340.086	0.513
15	319.55	320.469	0.288	338.25	340.018	0.523
16	319.45	320.441	0.310	338.15	339.99	0.544

In the second case, having four cores fully energized at 1800 seconds, the transient temperature change obtained from the experimental measurements, and the ANSYS and the Modelica simulations are plotted in Figure 5.19.

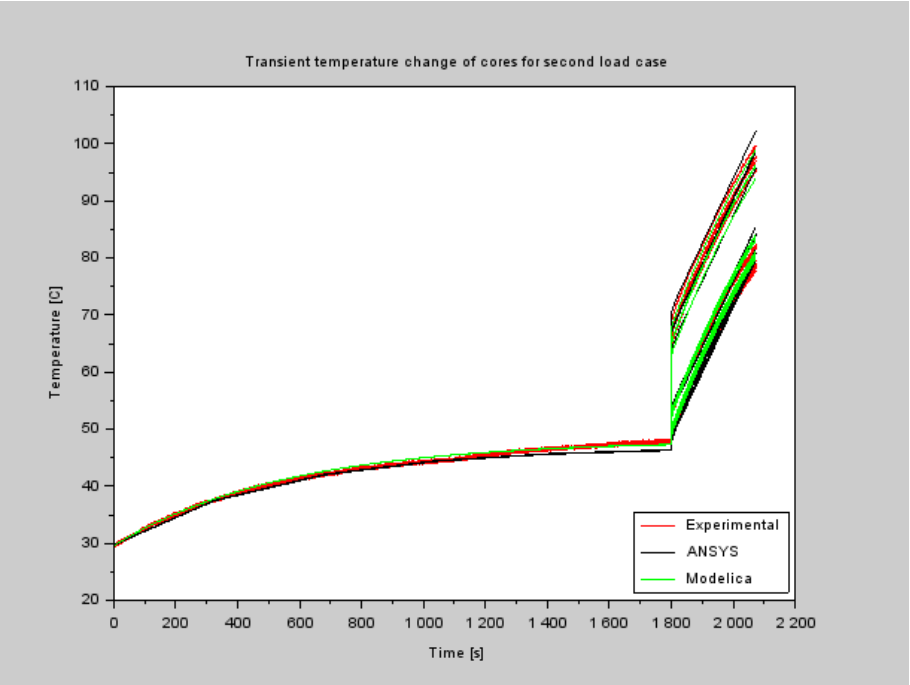


Figure 5.19: Transient temperature change of microprocessor cores for second load case

In Table 5.3, obtained from the Modelica results and the experimental results are given. The maximum error between the experimental results and the Modelica results occurs on the fourth core at $t = 2073s$ as 0.791 %. By using given time step errors of the microprocessor, the overall error in the Modelica simulation is calculated by using Root Mean Square (RMS) method and which is equal to 0.038 %.

Table 5.3: Temperature and error data for second load case in Modelica

Cores	Experimental Temperatures [K] at t = 0s	Modelica Temperatures [K] at t = 0s	Percent Error %	Experimental Temperatures [K] at t = 1799.5s	Modelica Temperatures [K] at t = 1799.5s	Percent Error %
1	300.75	300.85	0.033	320.75	320.436	-0.098
2	300.85	300.85	0.000	320.95	320.436	-0.160
3	300.75	300.85	0.033	321.05	320.436	-0.191
4	300.75	300.85	0.033	320.75	320.436	-0.098
5	300.85	300.85	0.000	321.05	320.465	-0.182
6	300.75	300.85	0.033	321.05	320.465	-0.182
7	300.75	300.85	0.033	321.05	320.465	-0.182
8	300.85	300.85	0.000	320.95	320.465	-0.151
9	300.85	300.85	0.000	320.95	320.465	-0.151
10	300.85	300.85	0.000	321.05	320.465	-0.182
11	300.95	300.85	-0.033	320.95	320.465	-0.151
12	300.85	300.85	0.000	321.05	320.465	-0.182
13	300.85	300.85	0.000	321.05	320.438	-0.191
14	300.85	300.85	0.000	321.15	320.438	-0.222
15	300.85	300.85	0.000	321.15	320.438	-0.222
16	300.95	300.85	-0.033	321.15	320.438	-0.222

Cores	Experimental Temperatures [K] at t = 1803s	Modelica Temperatures [K] at t = 1803s	Percent Error %	Experimental Temperatures [K] at t = 2073s	Modelica Temperatures [K] at t = 2073s	Percent Error %
1	342.55	341.487	-0.310	372.75	372.098	-0.175
2	339.95	338.886	-0.313	370.25	369.496	-0.204
3	326.15	326.424	0.084	355.15	357.034	0.530
4	322.85	323.822	0.301	351.65	354.433	0.791
5	340.35	338.687	-0.489	370.75	369.29	-0.394
6	338.25	336.572	-0.496	368.45	367.175	-0.346
7	325.65	325.72	0.021	354.85	356.324	0.415
8	322.85	323.605	0.234	351.65	354.209	0.728
9	326.25	326.705	0.139	355.35	357.295	0.547
10	325.85	325.943	0.029	354.95	356.533	0.446
11	323.75	323.75	0.000	352.55	354.34	0.508
12	322.65	322.988	0.105	351.35	353.578	0.634
13	323.15	323.641	0.152	351.85	354.215	0.672
14	322.95	323.365	0.129	351.85	353.94	0.594
15	322.65	322.783	0.041	351.35	353.357	0.571
16	322.45	322.508	0.018	350.95	353.082	0.607

In the third case, having nine cores fully energized at 1800 seconds, the transient temperature change obtained from the experimental measurements, and the ANSYS and the Modelica simulations are plotted in Figure 5.20.

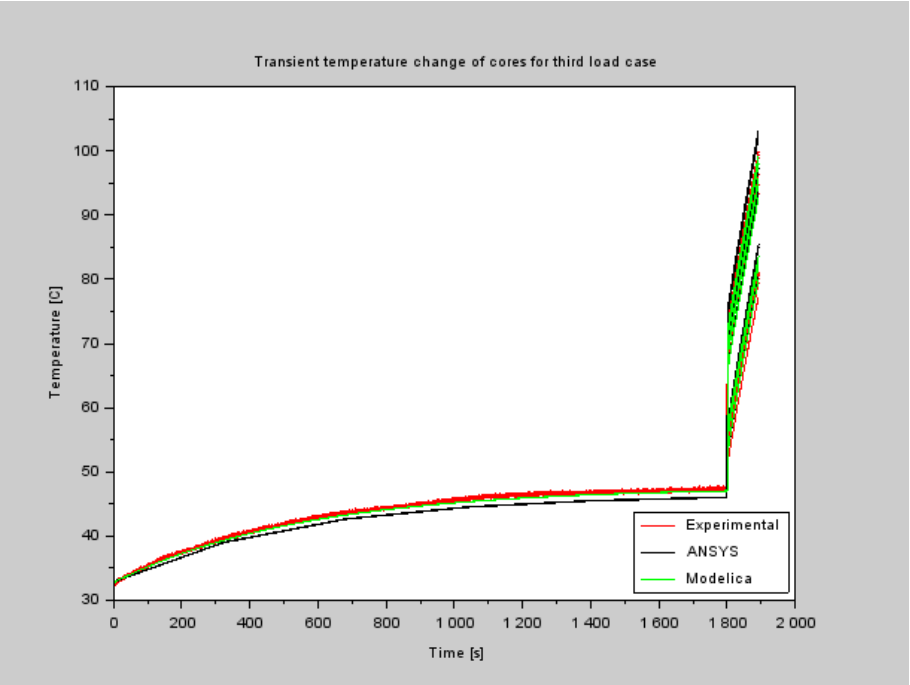


Figure 5.20: Transient temperature change of microprocessor cores for third load case

In Table 5.4, obtained from the Modelica results and the experimental results are given. The maximum error between the experimental results and the Modelica results occurs on the thirteenth core at $t = 1892s$ as 0.962 %. By using given time step errors of the microprocessor, the overall error in the Modelica simulation is calculated by using Root Mean Square (RMS) method and which is equal to 0.043 %.

Table 5.4: Temperature and error data for third load case in Modelica

Cores	Experimental Temperatures [K] at t = 0s	Modelica Temperatures [K] at t = 0s	Percent Error %	Experimental Temperatures [K] at t = 1799.5s	Modelica Temperatures [K] at t = 1799.5s	Percent Error %
1	303.95	304.05	0.033	320.15	320.175	0.008
2	304.05	304.05	0.000	320.45	320.175	-0.086
3	303.95	304.05	0.033	320.45	320.175	-0.086
4	303.95	304.05	0.033	320.35	320.175	-0.055
5	303.95	304.05	0.000	320.35	320.204	-0.046
6	303.95	304.05	0.033	320.45	320.204	-0.077
7	304.05	304.05	0.033	320.45	320.204	-0.077
8	304.05	304.05	0.000	320.45	320.204	-0.077
9	304.05	304.05	0.000	320.35	320.204	-0.046
10	304.05	304.05	0.000	320.55	320.204	-0.108
11	303.95	304.05	-0.033	320.45	320.204	-0.077
12	304.05	304.05	0.000	320.45	320.204	-0.077
13	304.05	304.05	0.000	320.45	320.177	-0.085
14	304.05	304.05	0.000	320.45	320.177	-0.085
15	304.05	304.05	0.000	320.45	320.177	-0.085
16	304.05	304.05	-0.033	320.55	320.177	-0.116

Cores	Experimental Temperatures [K] at t = 1803s	Modelica Temperatures [K] at t = 1803s	Percent Error %	Experimental Temperatures [K] at t = 1892s	Modelica Temperatures [K] at t = 1892s	Percent Error %
1	345.65	345.822	0.050	372.05	372.349	0.080
2	345.75	345.17	-0.168	372.55	371.697	-0.229
3	342.35	342.361	0.003	369.45	368.888	-0.152
4	328.25	330.095	0.562	354.05	356.622	0.726
5	345.15	345.212	0.018	371.55	371.739	0.051
6	346.15	344.602	-0.447	372.95	371.128	-0.489
7	342.45	341.932	-0.151	369.55	368.458	-0.295
8	328.35	329.954	0.489	354.25	356.48	0.629
9	341.45	342.033	0.171	367.95	368.547	0.162
10	342.45	341.541	-0.265	369.15	368.055	-0.297
11	339.65	339.357	-0.086	366.45	365.871	-0.158
12	327.15	328.872	0.526	352.95	355.385	0.690
13	327.95	330.256	0.703	353.35	356.75	0.962
14	328.05	329.969	0.585	353.85	356.463	0.738
15	326.95	329	0.627	352.75	355.494	0.778
16	324.85	326.638	0.550	350.35	353.132	0.794

In the fourth case, having sixteen cores fully energized at 1800 seconds, the transient temperature change obtained from the experimental measurements, and the ANSYS and the Modelica simulations are plotted in Figure 5.21.

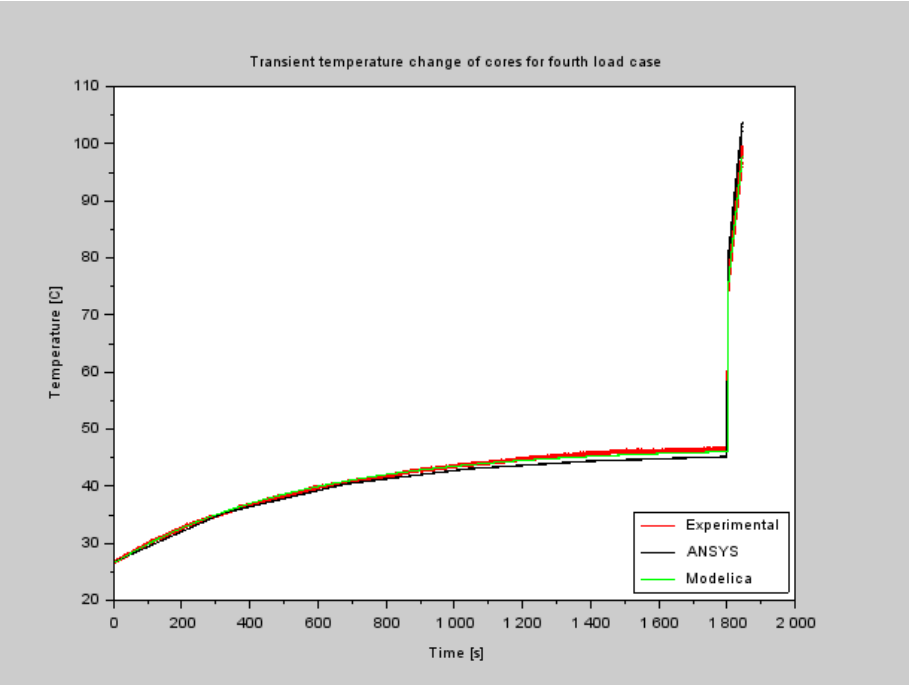


Figure 5.21: Transient temperature change of microprocessor cores for fourth load case

In Table 5.5, obtained from the Modelica results and the experimental results are given. The maximum error between the experimental results and the Modelica results occurs on the ninth core at $t = 1844s$ as -0.510 %. By using given time step errors of the microprocessor, the overall error in the Modelica simulation is calculated by using Root Mean Square (RMS) method and which is equal to 0.026 %.

Table 5.5: Temperature and error data for fourth load case in Modelica

Cores	Experimental Temperatures [K] at t = 0s	Modelica Temperatures [K] at t = 0s	Percent Error %	Experimental Temperatures [K] at t = 1799.5s	Modelica Temperatures [K] at t = 1799.5s	Percent Error %
1	297.75	297.85	0.034	319.55	319.187	-0.114
2	297.75	297.85	0.034	319.75	319.187	-0.176
3	297.75	297.85	0.034	319.75	319.187	-0.176
4	297.75	297.85	0.034	319.65	319.187	-0.145
5	297.85	297.85	0.000	319.85	319.206	-0.201
6	297.75	297.85	0.034	319.85	319.206	-0.201
7	297.75	297.85	0.034	319.85	319.206	-0.201
8	297.95	297.85	-0.034	319.85	319.206	-0.201
9	297.85	297.85	0.000	319.75	319.207	-0.170
10	297.75	297.85	0.034	319.95	319.207	-0.232
11	297.75	297.85	0.034	319.85	319.207	-0.201
12	297.85	297.85	0.000	319.75	319.207	-0.170
13	297.95	297.85	-0.034	319.65	319.188	-0.145
14	297.75	297.85	0.034	319.75	319.188	-0.176
15	297.85	297.85	0.000	319.75	319.188	-0.176
16	297.95	297.85	-0.034	319.85	319.188	-0.207

Cores	Experimental Temperatures [K] at t = 1803s	Modelica Temperatures [K] at t = 1803s	Percent Error %	Experimental Temperatures [K] at t = 1844s	Modelica Temperatures [K] at t = 1844s	Percent Error %
1	346.65	347.71	0.306	369.85	370.627	0.210
2	347.85	347.71	-0.040	371.55	370.627	-0.248
3	347.85	347.71	-0.040	371.75	370.627	-0.302
4	347.15	347.71	0.161	370.95	370.627	-0.087
5	347.25	348.015	0.220	370.35	370.949	0.162
6	349.05	348.015	-0.297	372.75	370.949	-0.483
7	348.85	348.015	-0.239	372.85	370.949	-0.510
8	348.15	348.015	-0.039	372.05	370.949	-0.296
9	346.65	348.016	0.394	369.75	370.957	0.326
10	348.65	348.016	-0.182	372.25	370.957	-0.347
11	348.65	348.016	-0.182	372.35	370.957	-0.374
12	347.65	348.016	0.105	371.55	370.957	-0.160
13	346.35	347.712	0.393	369.25	370.647	0.378
14	347.35	347.712	0.104	370.75	370.647	-0.028
15	347.45	347.712	0.075	370.95	370.647	-0.082
16	346.85	347.712	0.249	370.75	370.647	-0.028

Comparison of ANSYS and Modelica Simulation Results

In this chapter, the result comparison of the ANSYS and the Modelica simulations is made for the fin temperatures. This comparison is done for the specific fins due to huge amount of data obtained from the simulations. The 1^{st}, the 7^{th}, and the 12^{th} fins are selected for the temperature data comparison. The reason behind this approach is that the thermal conduction coefficient of the heat sink is high enough to assure an almost constant temperature distribution over the body, hence, the temperature differences among the fins are not very significant. However, this characteristic changes with the increasing number of fully loaded cores. Thus, the farthest and the closest fins from the microprocessor, which are the 1^{st}, the 12^{th}, are selected. Additionally, another fin between these two, which is the 7^{th}, is selected to show the gradual decrease towards the farthest points from the center. The 17^{th} and the 23^{rd} fins are not included in the analysis due to the symmetrical geometry of the heat sink.

In the Modelica simulation, the fins of the heat sink are divided into nine control volumes. Due to the FVM approach, the temperature data of these volumes are represented by the temperature value of their center point. Thus, to make an exact comparison, the points having the same coordinates are marked on the ANSYS simulation geometry to get the temperature data. The mentioned points can be seen in Figure 6.1.

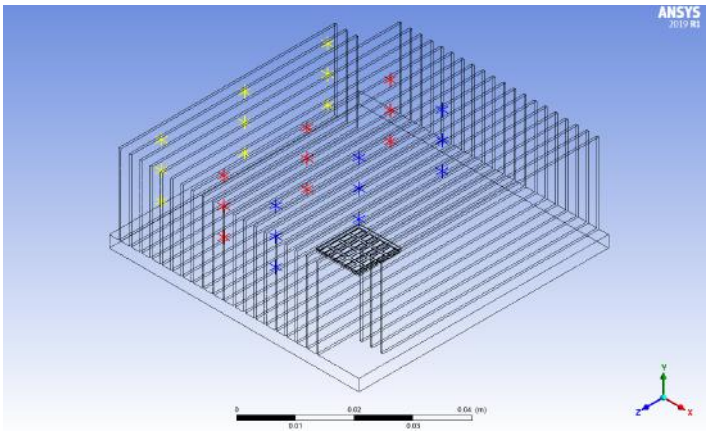
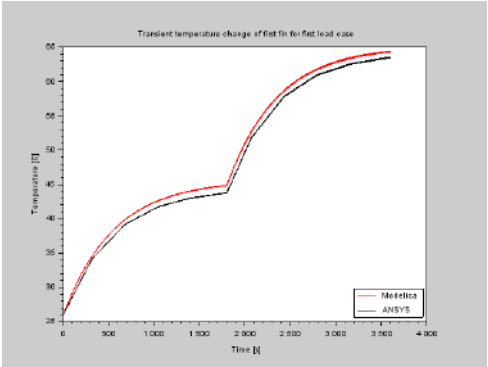


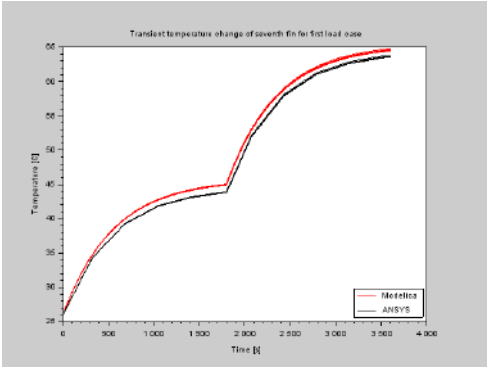
Figure 6.1: Temperature data points on the fins

The transient temperature change for these specified control volumes are found for different load cases. In Figure 6.2-6.5, these four results of the fin temperatures obtained in the ANSYS and the Modelica simulations are given; each figure contains three sub-figures to represent the 1^{st}, the 7^{th} and the 12^{th} fins of the heat sink for the same load case. In Figure 6.2, the transient temperature change of the 1^{st}, the 7^{th} and the 12^{th} fins are shown for the first load case. As it can be seen from Figure 6.2, the temperature values of the specified fins reach to a steady-state value, which is around 65 ^{0}C, at the end of the simulation time.

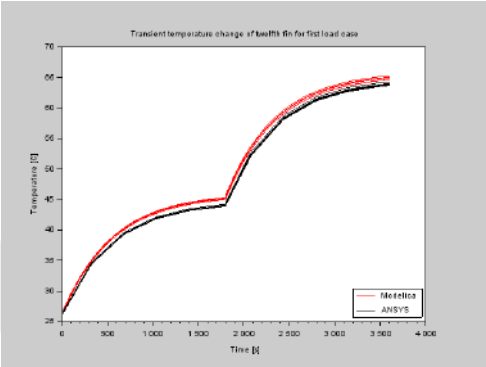
As it can be seen from Figure 6.2, the temperature of the nine control values on each fin have similar values due to the high thermal conductivity of the heat sink material. Only the 12^{th} fin has slightly higher temperature values at the control volumes due to its close distance to the microprocessor. Additionally, as it is given in Figure 6.2c the temperature values of the control volumes on the 12^{th} fin is more scattered than the other cases due to the same reason. Finally, the temperature difference between the ANSYS and the Modelica simulations which can be seen in Figure 6.2, is caused by the coarser mesh elements used in the Modelica simulation. For instance, when the heat sink fins are divided into higher number of mesh elements in the x- direction, the temperature variation in z- direction increases due to the decrease in the heat transfer areas between the volumes in that direction. This fact causes an increase at the middle elements of the fins on z- direction and it causes a slight decrease on the extremity elements in same direction. Additionally, when the microprocessor's passive part is divided more elements in x- direction, there is no significant effect on the temperature distribution of the fins. Finally, when the heat sink base and the fins are divided to a higher number of elements in z- direction, due to the decrease of the cross sectional area in x- direction, the temperature values of all fin elements increase. These variations due to the mesh sizing, create a deviation around 0.57 %, however, when the number of the state variables are increased in the Modelica model, the calculation time of the model increases. Thus, these



(a)
Transient temperature change of first fin



(b)
Transient temperature change of seventh fin

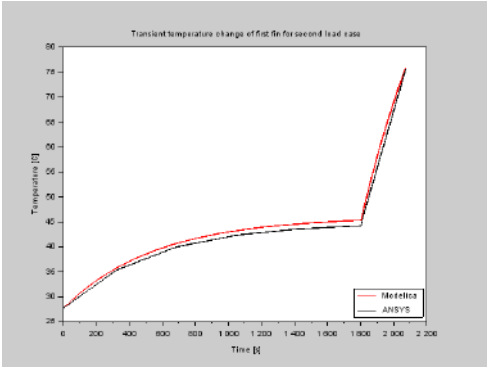


(c)
Transient temperature change of twelfth fin

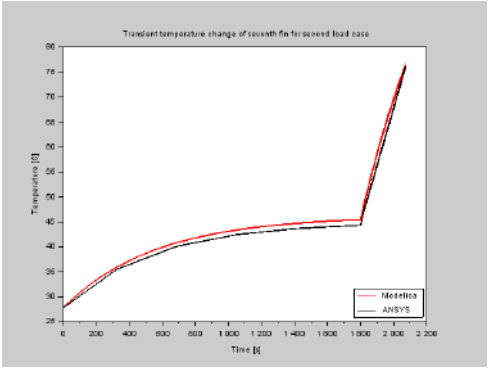
Figure 6.2: Transient temperature change of specified fins for first load case

deviations are neglected to avoid long computational times.

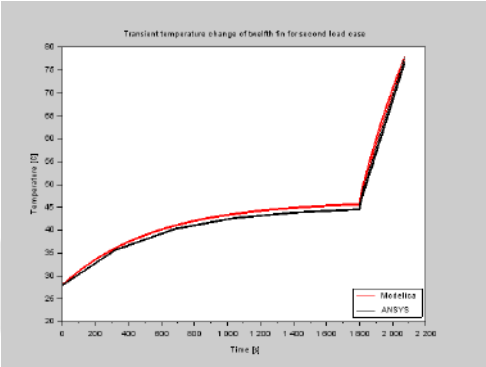
In Figure 6.3, the transient temperature change of the 1^{st}, the 7^{th} and the 12^{th} fins are shown for the second load case. As it can be seen from Figure 6.3, the temperature values of the specified fins does not reach to a steady-state value at the end of the simulation time and the temperature of the fin control volumes reach a value between 75-80 ^{0}C.



(a)
Transient temperature change of first fin



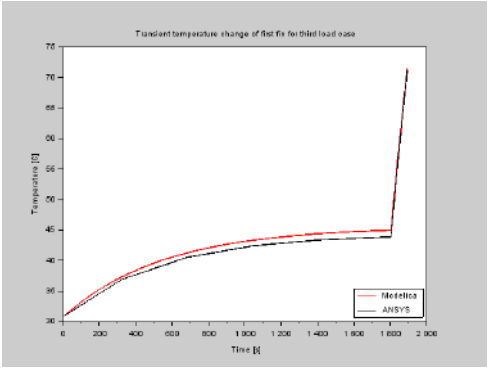
(b)
Transient temperature change of seventh fin



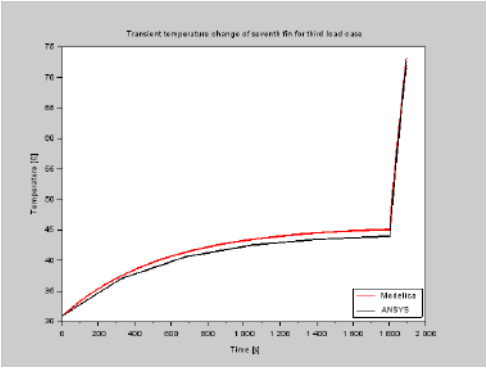
(c)
Transient temperature change of twelfth fin

Figure 6.3: Transient temperature change of specified fins for second load case

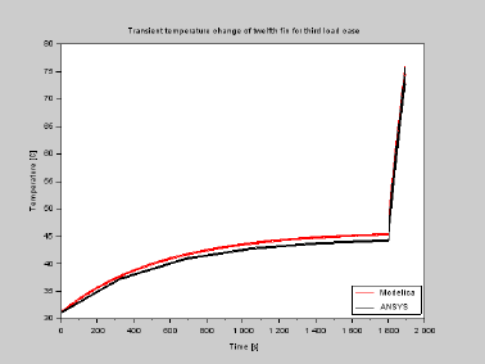
In Figure 6.4, the transient temperature change of the 1^{st}, the 7^{th} and the 12^{th} fins are shown for the third load case. As it can be seen from Figure 6.4, the temperature values of the specified fins does not reach to a steady-state value at the end of the simulation time and the temperature of the fin control volumes reach a value between 70-80 ^{0}C. The temperature of the specified fins have smaller value than the second load case, because, due to the increased number of fully loaded cores, the fully loaded working time is decreased which also decreases the total heat transfer time between the microprocessor and the heat sink.



(a)
Transient temperature change of first fin



(b)
Transient temperature change of seventh fin

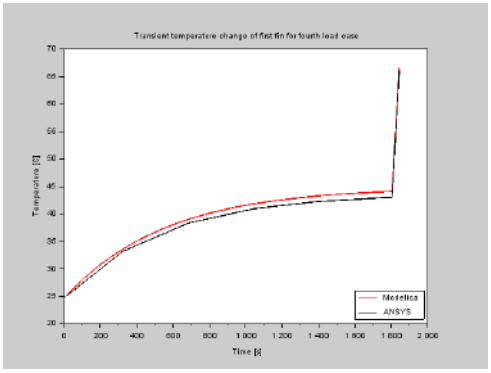


(c)
Transient temperature change of twelfth fin

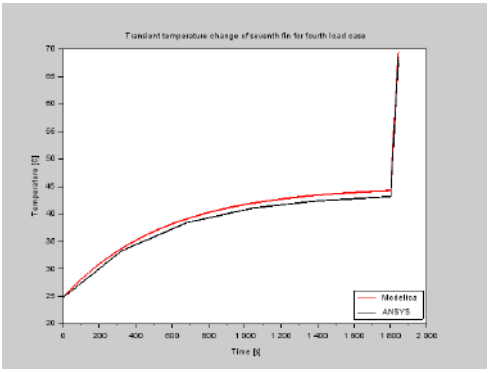
Figure 6.4: Transient temperature change of specified fins for third load case

In Figure 6.5, the transient temperature change of the 1^{st}, the 7^{th} and the 12^{th} fins are shown for the fourth load case. As it can be seen from Figure 6.5, the temperature values of the specified fins does not reach to a steady-state value at the end of the simulation time and the temperature of the fin control volumes reach a value between 65-75 ^{0}C. The temperature of the specified fins have even smaller value than the third load case, due to the same reasoning with the third load case.

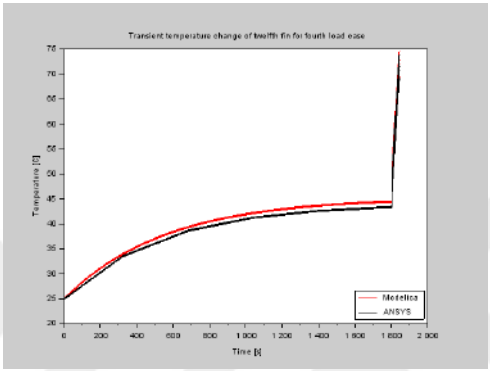
The steady-state formulations given in Section 5.1.2 are used to check the validity of the simulation models for the fin temperatures. These formulations can be only applied to the first load case, since, it is the only case where the steady-state temperature value is achieved. The base and the fin tip temperature values of these three fins are obtained in the ANSYS and the Modelica simulations. To find the Θ values, Equation 5.11 and Equation 5.12 are used; the Θ values found from these equations are equal to 0.995 and 0.9949 respectively. Since, the cross-sectional area at the fin tip is small compared to the other convection surfaces of the fin, the insulated and the uninsulated tip approaches give similar results. Due to this reason, the theoretical fin tip temperature values are calculated by using the fin base temperature values and the insulated fin tip approach for



(a)
Transient temperature change of first fin



(b)
Transient temperature change of seventh fin



(c)
Transient temperature change of twelfth fin

Figure 6.5: Transient temperature change of specified fins for fourth load case

the ANSYS and the Modelica simulations. In Table 6.1, these theoretical values and the simulation results for the ANSYS and the Modelica are compared and the error between them are calculated with the same method used in Section 5.7.

Table 6.1: Comparison of fin tip temperatures

Fin Elements	Modelica Temperature Values for Fin Base [K]	Theoretical Temperature Values for Fin Tip from Modelica [K]	Modelica Temperature Values for Fin Tip [K]	Percent Error %
1_1	337.575	335.8975	337.4	0.45
1_2	337.674	335.996	337.431	0.43
1_3	337.542	335.8646	337.378	0.45
7_1	337.716	336.0378	337.592	0.46
7_2	338.057	336.3771	337.713	0.40
7_3	337.698	336.0199	337.58	0.46
12_1	337.822	336.1432	337.809	0.50
12_2	338.687	337.0039	338.12	0.33
12_3	337.815	336.1363	337.804	0.50

Fin Elements	ANSYS Temperature Values for Fin Base [K]	Theoretical Temperature Values for Fin Tip from ANSYS [K]	ANSYS Temperature Values for Fin Tip [K]	Percent Error %
1_1	336.71	335.0368	336.53	0.50
1_2	336.75	335.0766	336.54	0.50
1_3	336.72	335.0467	336.54	0.50
7_1	336.86	335.186	336.72	0.50
7_2	337.09	335.4149	336.76	0.50
7_3	336.82	335.1462	336.68	0.50
12_1	336.96	335.2855	336.92	0.50
12_2	337.76	336.0816	337.06	0.50
12_3	336.88	335.2059	336.86	0.50

Conclusion and Future Work

In this thesis, thermal modelling of the microprocessor and the heat sink complex is realized for further usage in control. The modelling procedure is done by using two different modelling approaches. The first approach was modelling through ANSYS which is a commercial software and the second one was modelling through Modelica which is a open source coding environment. The working procedures of both approaches are explained and the obtained results for these simulations are stated. The comparison of these results with the experimental and the theoretical data is realized. The error between the experimental and the simulation results are less than 2 % for all time steps by using the contained energy inside the unit volume, which makes the results within the acceptable margin of 10 %, when designing the controller algorithm for cooling system.

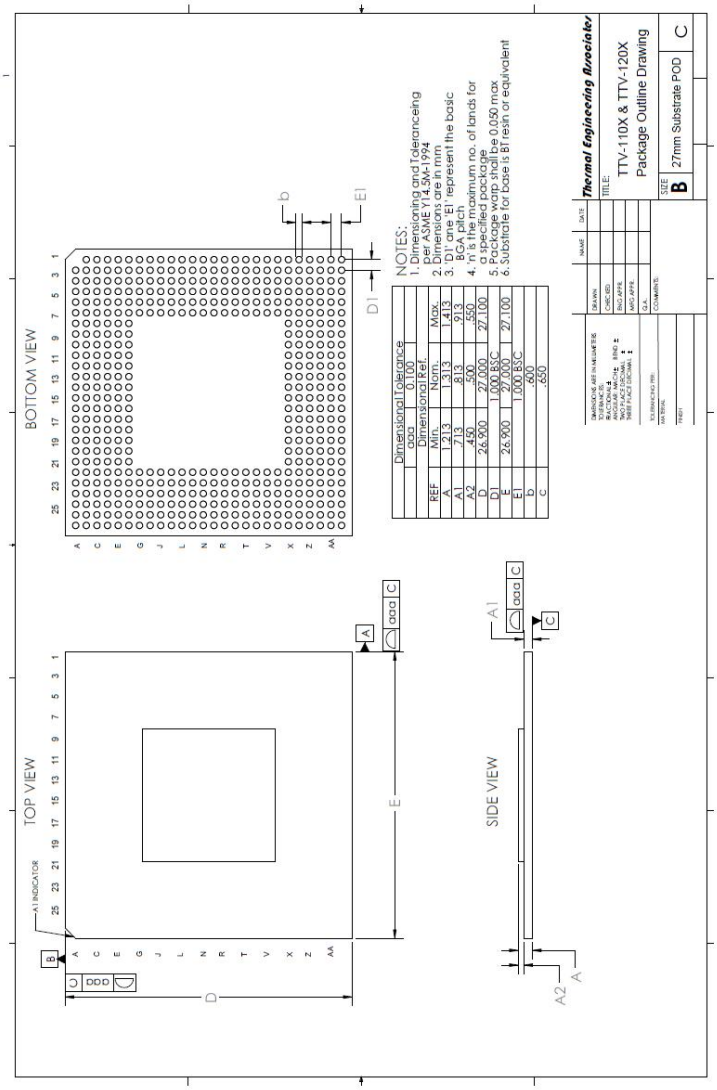
By using the Modelica model, the transient temperature behavior of the microprocessor and the heat sink is modelled with less number of state variables which makes the computational time significantly less than the ANSYS simulations. Since data centers have a huge number of microprocessors and heat sinks, it is possible to model each microprocessor and heat sink couple with appropriate modification in the Modelica code. Hence, the transient temperature behavior of the whole data center can be simulated accordingly. Additionally, to control the microprocessor temperatures inside the data center by knowing the transient temperature distributions obtained from these simulations, it is also important to have a knowledge on HVAC system, since it is necessary to take the control action in advance for the systems having slower dynamics.

In the data centers, the system dynamics from the fastest to the slowest could be ordered as the microprocessor, the heat sink and the HVAC system. Due to its fast dynamics, the sudden temperature variations occur on the microprocessor, when it is subjected to sudden voltage changes. However, these variations cannot be seen in the heat sink as fast as in the microprocessor, due to its larger dimensions and the slow heat transfer from the rubber pad to the heat sink. Also, it is necessary to control the air temperature inside the data center to be able to cool down the heat sink, which generates an interaction surface between the microprocessor and the ambient. Thus, to adjust the microprocessor temperature, it is necessary to control the ambient temperature. By knowing the HVAC system and the heat sink dynamics, the appropriate temperature threshold for the heat

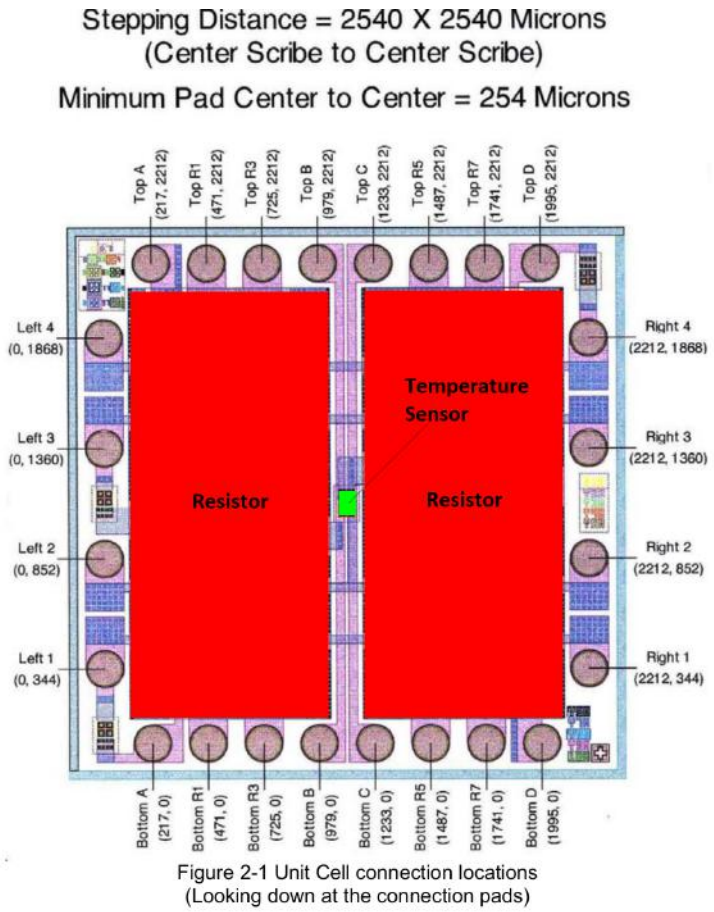
sink could be determined to provide the optimum working temperature and to prevent damage on the microprocessor.

Appendices

Appendix A



Technical drawing of the microprocessor



Technical drawing of unit core cell

Thermal properties of used materials

Materials	Silicon	Copper	Rubber
Specific Heat $[\frac{J}{kgK}]$	875	381	711
Thermal Conductivity Coefficient $[\frac{W}{mK}]$	140	387.6	3.6
Density $[\frac{kg}{m^3}]$	2923	8978	1120

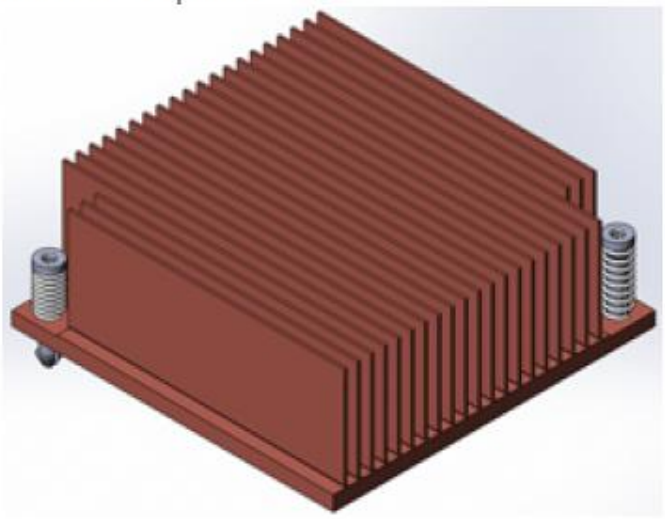
Appendix B

» FPGA / BGA Heat Sinks

342946

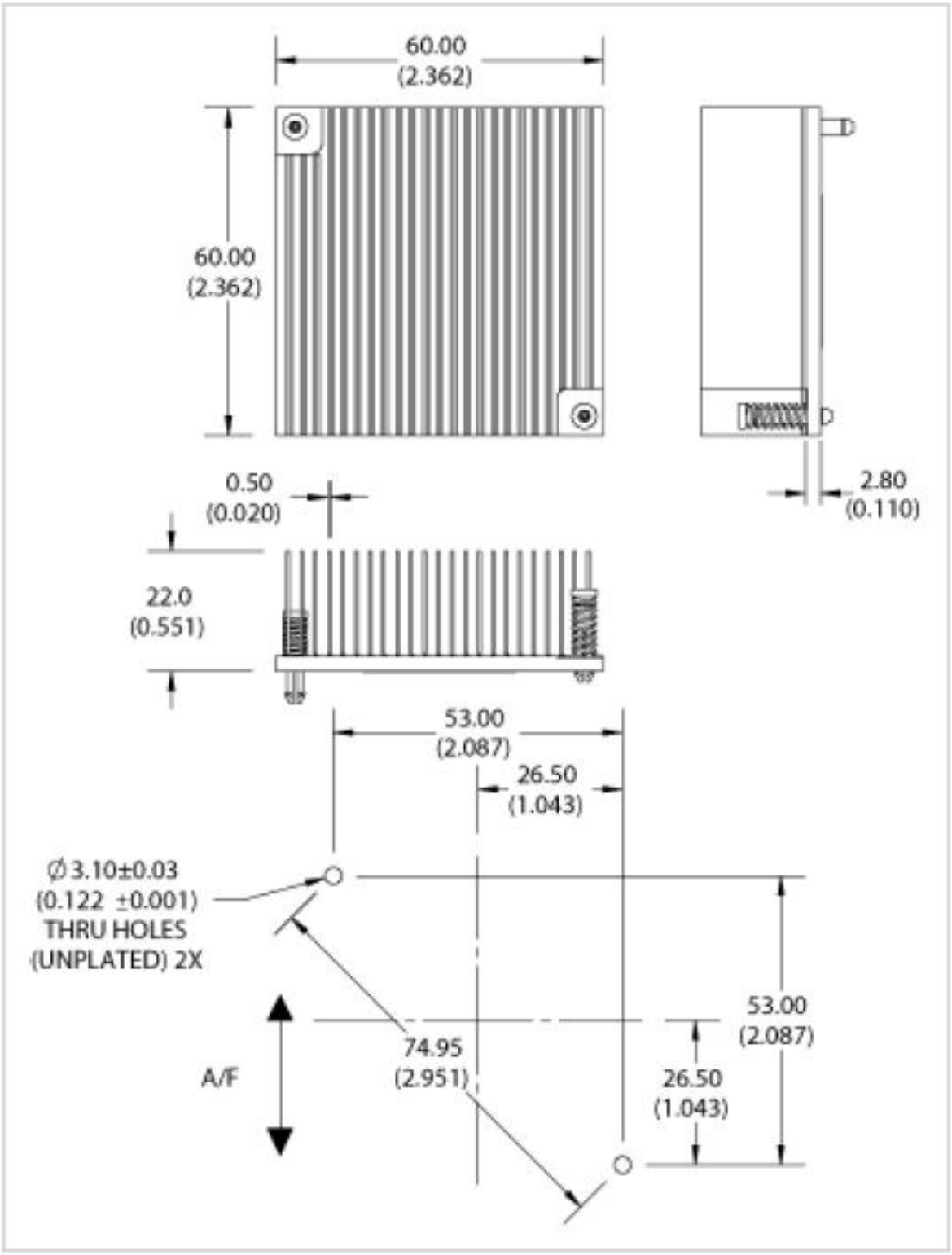
SQUARE HEATSINK WITH PUSH PIN ATTACHMENT

RoHS: compliant



Width: 60.00
Length: 60.00
Height: 22.00
Base Thickness: 2.80
Fin Thickness: 0.50
# of Fins: 23
Finish: AavSHIELD 3C

3-D representative scheme of heat sink



Technical drawing of heat sink

Appendix C

```
#include "udf.h"
#include "prop.h"

real sub_heat_release(cell_t cell, Thread *thread, real dS[], int eqn)
{
    real x = RP_Get_Real("flow-time");
    real t;
    dS[eqn] = 0.0;
    if (x >= 0.0 && x < 1800.0) {
        t = 93648648.642*5;
    }
    else if (x >= 1800.0 && x <= 1844.0) {
        t = 1649535135.14*5;
    }
    return (t);
}
```

Implemented heat generating code in ANSYS

```
model HeatSinkModelv1_4

  // Types are introduced here
  type Temperature = Real(unit="K",min=0);
  type ConvecCoeff = Real(unit="W/(m^2*K)", min=0);
  type ConducCoeff = Real(unit="W/(m*K)", min=0);
  type Mass = Real(unit="kg", min=0);
  type SpecificHeat = Real(unit="J/(K*kg)", min=0);
  type Density = Real(unit="kg/m^3", min=0);
  type Area = Real(unit="m^2", min=0);
  type Volume = Real(unit="m^3", min=0);
  type Height = Real(unit="m", min=0);
  type Width = Real(unit="m", min=0);
  type Length = Real(unit="m", min=0);
  type DimensionMesh = Real(unit="m", min=0);
  type Emissivity = Real(unitless,min=0);
  type StefanBoltzman = Real(unit="W/(m^2*K^4)",min=0);

  // Radiation Constants
  parameter StefanBoltzman sigma = 5.67*10^(-8);
  parameter Emissivity e = 0.03;

  // Introduce dimensions for base
  parameter Height heightBase = 2.8*10^(-3);
  parameter Width widthBase = 60*10^(-3);
  parameter Length lengthBase = 60*10^(-3);

  // Introduce dimensions for fin
  parameter Height height = 19.2*10^(-3);
  parameter Width width = 0.5*10^(-3);
  parameter Length length = 60*10^(-3);

  // Parameters of Copper
  parameter Density rho = 8978;
  parameter ConvecCoeff h = 2.4;         // 2.4
  parameter ConducCoeff k = 387.6;
  parameter SpecificHeat C = 381;

  // Parameter of Silicon
  parameter Density rhosic = 2329;
  parameter ConducCoeff ksic = 210;      // 100
  parameter SpecificHeat Csic = 875;

  // Parameters of Rubber Pad
  parameter Density rhorb = 1120;
  parameter ConducCoeff krb = 3.6;       // 3.6
  parameter SpecificHeat Crb = 711;

  // Ambient Temperature for One Chip
  parameter Temperature Tamb = 297.85 "Ambient temperature";
  // Ambient Temperature for different setups 298.55 298.95 298.55 297.85

  // Fin Finite Elements
  parameter Integer x = 3;
  parameter Integer z = 3;
  parameter Integer a = 23;

  // Fin Base Finite Elements
  parameter Integer yb = 23;
  parameter Integer zb = 3;
  parameter Integer xb = 3;

  // Chip Finite Elements
  parameter Integer yc = 4;
  parameter Integer zc = 4;
  parameter Integer xc = 5;


  // Rubber Pad Finite Elements
  parameter Integer yr = 4;
  parameter Integer zr = 4;
```

```
// Finite Area and Volume of Fin
parameter DimensionMesh dx = height / x;
parameter DimensionMesh dz = length / z;
parameter Area SA = (2 * dz + width) * dx;
parameter Area SAM = 2 * dz * dx;
parameter Area CSAX = width * dz;
parameter Area CSAZ = width * dx;
parameter Volume V = dx * width * dz;
// Energy terms
parameter Real W[yc,zc,2] = fill(0.186/2,yc,zc,2);          // Initial watt value of chip
parameter Real W2[yc,zc,2] = fill(3.0516/2,yc,zc,2);


// Finite Area and Volume of Heat Sink Base
parameter DimensionMesh dby = widthBase / yb;               // y element of the Fin Base
parameter DimensionMesh dbz = lengthBase / zb;              // z element of the Fin Base
parameter DimensionMesh dbx = heightBase / xb;              // x element of the Fin Base
parameter Area SABF = dby*dbz + dbx*(dby+dbz) - dbz*width; // Corners of the Fin Base with Fin
parameter Area SAB = dby*dbz + dbx*(dby+dbz);           // Corners of the Fin Base without Fin
parameter Area SAMBF = dby*dbz + dbx*dbz - dbz*width;   // Edges of the Fin Base with Fin
parameter Area SAMB = dby*dbz + dbx*dbz;                // Edges of the Fin Base without Fin
parameter Area SAMB8F = dby*dbz + dbx*dby - dbz*width;  // Edges of the Fin Base with Fin
parameter Area SAMB8 = dby*dbz + dbx*dby;    // Edges of the Fin Base without Fin
parameter Area SAMMBF = dby*dbz - dbz*width; // Middle of the Fin Base with Fin
parameter Area SAMMB = dby*dbz;  // Middle of the Fin Base which is not on the chip
parameter Area CSABY = dbx*dbz;  // Cross Sectional Area of the Fin Base in y direction
parameter Area CSABZ = dbx*dby;  // Cross Sectional Area of the Fin Base in z direction
parameter Area CSABX = dbz*dby;" // Cross Sectional Area of the Fin Base in x direction
parameter Volume VB = dbx*dbz*dby;            // Volume of the Fin Base Element

// Position values in y direction
parameter Real sypos[yb] = {(i-1)*dby  for i in 1:yb};
parameter Real eypos[yb] = {i*dby  for i in 1:yb};

// Position values in z direction
parameter Real spos[zb] = {(i-1)*(dbz)  for i in 1:zb};
parameter Real epos[zb] = {i*(dbz)  for i in 1:zb};


// Finite Area and Volume of Chip
parameter Length lengthCorex = 0.525*10^(-3);
parameter DimensionMesh dcx = lengthCorex/xc;
parameter DimensionMesh dcxa = 0.1*10^(-3);
parameter DimensionMesh dcya = 0.925*10^(-3);
parameter DimensionMesh dcza = 2*10^(-3);
parameter DimensionMesh dsy = 0.15*10^(-3);
parameter DimensionMesh drth = 0.28*10^(-3);
parameter DimensionMesh dcz = 10.23*10^(-3)/4;
parameter DimensionMesh drx = 0.15*10^(-3);             // x element of the rubber pad

// Areas of the Chip elements
parameter Area CSA = dcz*dcz;        // Surface Area of the Each Chip
parameter Area CSAP = 2.541*10^(-6); // Cross Sectional Area of the Passive Part of the Active
Element of Chip
parameter Area CSAAP1 = dcza*dcxa;
parameter Area CSAAP2 = dcya*dcxa;
parameter Area CSARP = 1.85*10^(-6);  // Heat transfer area from resistor to passive element
parameter Area CSASP = 0.3*10^(-6);   // Heat transfer area from sensor to passive element
parameter Area CCSA = dcz*dcx;        // Cross Sectional Area of the Passive Element of Chip
parameter Area CCSAR = dcz*drx;       // Cross Sectional Area of the Each Rubber Pad Element

// Volumes of the Chip Elements
parameter Volume VC = dcz*dcz*dcx;                  // Volume of the Passive Element of Chip
parameter Volume VCA = dcya*dcza*dcxa;              // Volume of the Active Element of Chip
parameter Volume VR = dcz*dcz*drx;                  // Volume of the Rubber Element

// Describe T matrix of fin volume and fin base volume
Temperature T[x,z,a];       // Temperature of fins
Temperature Tb[yb,zb,xb];   // Temperature of fin base
Temperature Tcp[yc,zc,xc];  // Temperature of passive part of chip
Temperature Tca[yc,zc,2];   // Temperature of active part of chip
Temperature Ts[yc,zc];      // Temperature of Sensor
Temperature Tr[yr,zr];      // Temperature of Rubber Pad
```

```
  // Watt matrices introduced to chip volume
  Real Q[yb,zb];              // Watt term will be added to fin base through rubber pad
  Real Q1[yb,yc,zb];
  Real Qo[yr,zr];             // Watt term from fin base throughout rubber pad

initial equation
    // Initial temperature and watt of rubber pad of fin volume and fin base volume
    // Microprocessor temperatures for different setups 299.15 300.85 304.05 297.85
    T = fill(297.85,x,z,a);
    Tb = fill(297.85,yb,zb,xb);
    Tcp = fill(297.85,yc,zc,xc);
    Tca = fill(297.85,yc,zc,2);
    Tr = fill(297.85,yr,zr);

  equation

    // Heat Generation Terms at Passive Part of the Chip is calculated. Checked hopefully true
    for q in 1:yc loop

      for u in  1:zc loop

        for q1 in 1:xc loop

          if q==1 and u==1 and q1==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcz -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcz  - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,1]) -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,2]);

          elseif q==1 and u==1 and q1==xc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcz -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcz - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx -
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tcp[q,u,q1]-Tr[q,u]);

          elseif q==yc and u==1 and q1==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcz -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcz  - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,1]) -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,2]);

          elseif q==yc and u==1 and q1==xc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcz -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcz - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx -
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tcp[q,u,q1]-Tr[q,u]);

          elseif q==1 and u==zc and q1==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcz -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcz  - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,1]) -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,2]);

          elseif q==1 and u==zc and q1==xc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcz -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcz - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx -
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tcp[q,u,q1]-Tr[q,u]);

          elseif q==yc and u==zc and q1==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcz -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcz  - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,1]) -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,2]);

          elseif q==yc and u==zc and q1==xc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcz -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcz - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx -
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tcp[q,u,q1]-Tr[q,u]);

          elseif q==1 and u==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcz -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcz - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx;
```

```
          elseif q==1 and u==nc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx;

          elseif q==nc and u==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx;

          elseif q==nc and u==nc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx;

          elseif q==1 and q1==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,1]) -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,2]);

          elseif q==nc and q1==1 then
             rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,1]) -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,2]);

          elseif q==1 and q1==nc then
             rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx - (1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tcp[q,u,q1]-
Tr[q,u]);

          elseif q==nc and q1==nc then
             rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx - (1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tcp[q,u,q1]-
Tr[q,u]);

          elseif u==1 and q1==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx  -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,1]) -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,2]);

          elseif u==nc and q1==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx  -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,1]) -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,2]);

          elseif u==1 and q1==nc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx - (1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tcp[q,u,q1]-
Tr[q,u]);

          elseif u==nc and q1==nc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx - (1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tcp[q,u,q1]-
Tr[q,u]);

          elseif q==1 then
             rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx;
```

```
          elseif q==yc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx;

          elseif u==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx  -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx;

          elseif u==zc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx  -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx;

          elseif q1==1 then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx  - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx  - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,1]) -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tcp[q,u,q1]-Tca[q,u,2]);

          elseif q1==zc then
            rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx  - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx  - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx-
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tcp[q,u,q1]-Tr[q,u]);

          else
             rhosic*VC*Csic*der(Tcp[q,u,q1]) = - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q-1,u,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q+1,u,q1])/dcx - ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u-1,q1])/dcx -
ksic*CCSA*(Tcp[q,u,q1]-Tcp[q,u+1,q1])/dcx - ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1+1])/dcx -
ksic*CSA*(Tcp[q,u,q1]-Tcp[q,u,q1-1])/dcx;

        end if;

      end for;

      end for;

    end for;


    // Heat Generation Terms at Resistor Element of the Chip is calculated.
    for q in 1:yc loop

      for u in 1:zc loop

        for n in 1:2 loop

          if time <= 1800.0 then

            if q==1 and u==1 and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W[q,u,n];

            elseif q==1 and u==zc and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W[q,u,n];

            elseif q==yc and u==1 and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W[q,u,n];

            elseif q==yc and u==zc and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W[q,u,n];
```

```
            elseif q==1 and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

            elseif q==yc and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

            elseif u==1 and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

            elseif u==1 and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

            elseif u==zc and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

            elseif u==zc and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

            elseif n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

            elseif n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

            end if;

          // For 16 Cores
          else
            if q==1 and u==1 and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W2[q,u,n];

            elseif q==1 and u==zc and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W2[q,u,n];

            elseif q==yc and u==1 and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W2[q,u,n];
```

```
            elseif q==yc and u==zc and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W2[q,u,n];

            elseif q==1 and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

            elseif q==yc and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

            elseif u==1 and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

            elseif u==1 and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

            elseif u==zc and n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

            elseif u==zc and n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

            elseif n==1 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

            elseif n==2 then
              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];



//            // For 4 Core
//            elseif time > 1800.0 and q==1 and u==1 and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W2[q,u,n];

//            elseif time > 1800.0 and q==1 and u==1 and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];
```

```
//          elseif time > 1800.0 and q==2 and u==1 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==2 and u==1 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==1 and u==2 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==1 and u==2 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==2 and u==2 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==2 and u==2 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          else

//            if q==1 and u==1 and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W[q,u,n];

//            elseif q==1 and u==xc and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W[q,u,n];

//            elseif q==yc and u==1 and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W[q,u,n];

//            elseif q==yc and u==xc and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W[q,u,n];

//            elseif q==1 and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcxa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];
```

```
//            elseif q==yc and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];


//            elseif u==1 and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

//            elseif u==1 and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

//            elseif u==zc and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

//            elseif u==zc and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

//            elseif n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

//            elseif n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];


//          // For 9 Core
//          elseif time > 1800.0 and q==1 and u==1 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==1 and u==1 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==2 and u==1 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==2 and u==1 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];
```

```
//          elseif time > 1800.0 and q==3 and u==1 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==3 and u==1 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==1 and u==2 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==1 and u==2 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==2 and u==2 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==2 and u==2 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==3 and u==2 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==3 and u==2 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==1 and u==3 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==1 and u==3 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==2 and u==3 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];
```

```
//          elseif time > 1800.0 and q==2 and u==3 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==3 and u==3 and n==1 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          elseif time > 1800.0 and q==3 and u==3 and n==2 then
//            rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W2[q,u,n];

//          else

//            if q==1 and u==1 and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W[q,u,n];

//            elseif q==1 and u==xc and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W[q,u,n];

//            elseif q==yc and u==1 and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W[q,u,n];

//            elseif q==yc and u==xc and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W[q,u,n];

//            elseif q==1 and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

//            elseif q==yc and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

//            elseif u==1 and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

//            elseif u==1 and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];
```

```
//             elseif u==sc and n==1 then
//               rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

//             elseif u==sc and n==2 then
//               rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

//             elseif n==1 then
//               rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

//             elseif n==2 then
//               rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];



          // For 1 Core
//           elseif time > 1800.0 and q==1 and u==1 and n==1 then
//             rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+d*/sy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) -
(ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

//           elseif time > 1800.0 and q==1 and u==1 and n==2 then
//             rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W2[q,u,n];

//           else

//             if q==1 and u==1 and n==1 then
//               rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W[q,u,n];

//             elseif q==1 and u==sc and n==1 then
//               rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W[q,u,n];

//             elseif q==yc and u==1 and n==2 then
//               rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) + W[q,u,n];

//             elseif q==yc and u==sc and n==2 then
//               rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u-1,n]) + W[q,u,n];

//             elseif q==1 and n==1 then
//               rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcsa+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];
```

```
//            elseif q==yc and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-
Tca[q,u+1,n]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

//            elseif u==1 and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

//            elseif u==1 and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

//            elseif u==zc and n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

//            elseif u==zc and n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) + W[q,u,n];

//            elseif n==1 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n+1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q-
1,u,n+1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

//            elseif n==2 then
//              rhosic*VCA*Csic*der(Tca[q,u,n]) = -
(1/(dcx/(2*ksic)+(dcxa/(2*ksic))))*CSARP*(Tca[q,u,n]-Tcp[q,u,1]) -
(ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-Tca[q,u,n-1]) - (ksic/(dcya+dsy))*CSAAP1*(Tca[q,u,n]-
Tca[q+1,u,n-1]) - (ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u-1,n]) -
(ksic/(dcza+(2*drth)))*CSAAP2*(Tca[q,u,n]-Tca[q,u+1,n]) + W[q,u,n];

            end if;

          end if;

        end for;

      end for;

    end for;


    // Temperature of the Sensor Measurements Calculations
    for q in 1:yc loop

      for u in 1:zc loop

        Ts[q,u] = (Tca[q,u,1]+Tca[q,u,2])/2;

      end for;

    end for;
```

```
    // Rubber pad heat transfer equations
    for w in 1:yr loop

      for c in  1:zr loop

        if w==1 and c==1 then
          rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w+1,c])/dcz - krb*CCSAR*(Tr[w,c]-
Tr[w,c+1])/dcz - (1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];

        elseif w==yc and c==1 then
          rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w-1,c])/dcz - krb*CCSAR*(Tr[w,c]-
Tr[w,c+1])/dcz - (1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];

        elseif w==1 and c==zc then
          rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w+1,c])/dcz - krb*CCSAR*(Tr[w,c]-
Tr[w,c-1])/dcz - (1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];

        elseif w==yc and c==zc then
          rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w-1,c])/dcz - krb*CCSAR*(Tr[w,c]-
Tr[w,c-1])/dcz - (1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];

        elseif w==1 then
          rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w+1,c])/dcz - krb*CCSAR*(Tr[w,c]-
Tr[w,c+1])/dcz - krb*CCSAR*(Tr[w,c]-Tr[w,c-1])/dcz -
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];

        elseif w==yc then
          rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w-1,c])/dcz - krb*CCSAR*(Tr[w,c]-
Tr[w,c+1])/dcz - krb*CCSAR*(Tr[w,c]-Tr[w,c-1])/dcz -
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];

        elseif c==1 then
          rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w-1,c])/dcz - krb*CCSAR*(Tr[w,c]-
Tr[w+1,c])/dcz - krb*CCSAR*(Tr[w,c]-  Tr[w,c+1])/dcz  -
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];

        elseif c==zc then
          rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w-1,c])/dcz - krb*CCSAR*(Tr[w,c]-
Tr[w+1,c])/dcz - krb*CCSAR*(Tr[w,c]-Tr[w,c-1])/dcz -
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];

        else
           rhorb*VR*Crb*der(Tr[w,c]) = - krb*CCSAR*(Tr[w,c]-Tr[w-1,c])/dcz - krb*CCSAR*(Tr[w,c]-
Tr[w+1,c])/dcz - krb*CCSAR*(Tr[w,c]-Tr[w,c-1])/dcz - krb*CCSAR*(Tr[w,c]-Tr[w,c+1])/dcz -
(1/(dcx/(2*ksic)+(drx/(2*krb))))*CSA*(Tr[w,c]-Tcp[w,c,xc]) + Qo[w,c];

        end if;

      end for;

    end for;
// Energy term will generated here in chips are added fin base elements
  for j in 1:yb loop
    if eypos[j] < 0.024885 then
      for n1 in 1:zb loop
        for n3 in 1:zc loop
          Q1[j, n3, n1] = 0.0;
        end for;
        Q[j, n1] = (Q1[j, 1, n1] + Q1[j, 2, n1] + Q1[j, 3, n1] + Q1[j, 4, n1]) / dbx;
      end for;
    elseif sypos[j] > 0.024885 + dcz * yc then
      for n2 in 1:zb loop
        for n4 in 1:zc loop
          Q1[j, n4, n2] = 0.0;
        end for;
        Q[j, n2] = (Q1[j, 1, n2] + Q1[j, 2, n2] + Q1[j, 3, n2] + Q1[j, 4, n2]) / dbx;
      end for;

// First if condition in y position. Chip and base elements are on top of each other.
    elseif eypos[j] < 0.024885 + dcz * yc and sypos[j] > 0.024885 then
      for n in 1:zb loop
        for i in 1:zc loop
          if dbz > dcz then
            if spos[n] > 0.024885 and epos[n] < 0.024885 + dcz * zc then
              if 0.024885 + dcz * (i - 1) < spos[n] and spos[n] < 0.024885 + dcz * i then
```

```
                Q1[j, i, n] = (-1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcx * i -
spos[n]) * (0.024885 + dcx * (j - 10) - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i])) - 1 / (drx /
(2 * krb) + dbx / (2 * k)) * (0.024885 + dcx * i - spos[n]) * (eypos[j] - (0.024885 + dcx * (j -
10))) * (Tb[j, n, 1] - Tr[j - 9, i]);
              elseif epos[n] > 0.024885 + dcx * i and spos[n] < 0.024885 + dcx * i then
                Q1[j, i, n] = (-1 / (drx / (2 * krb) + dbx / (2 * k)) * dcx * (0.024885 + dcx *
(j - 10) - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i])) - 1 / (drx / (2 * krb) + dbx / (2 * k)) *
dcx * (eypos[j] - (0.024885 + dcx * (j - 10))) * (Tb[j, n, 1] - Tr[j - 9, i]);
              elseif spos[n] > 0.024885 + dcx * i then
                Q1[j, i, n] = 0.0;
              end if;
              if epos[n] > 0.024885 + dcx * (i - 1) and epos[n] <= 0.024885 + dcx * i then
                Q1[j, i, n] = (-1 / (drx / (2 * krb) + dbx / (2 * k)) * (epos[n] - (0.024885 +
dcx * (i - 1))) * (0.024885 + dcx * (j - 10) - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i - 1])) -
1 / (drx / (2 * krb) + dbx / (2 * k)) * (epos[n] - (0.024885 + dcx * (i - 1))) * (eypos[j] -
(0.024885 + dcx * (j - 10))) * (Tb[j, n, 1] - Tr[j - 9, i - 1]);
                for a in i + 1:4 loop
                  Q1[j, a, n] = 0.0;
                end for;
              end if;
            elseif spos[n] < 0.024885 and epos[n] < 0.024885 + dcx * zc and epos[n] > 0.024885
then
              if epos[n] > 0.024885 + dcx * (i - 1) then
                Q1[j, i, n] = (-1 / (drx / (2 * krb) + dbx / (2 * k)) * dcx * (0.024885 + dcx *
(j - 10) - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i])) - 1 / (drx / (2 * krb) + dbx / (2 * k)) *
dcx * (eypos[j] - (0.024885 + dcx * (j - 10))) * (Tb[j, n, 1] - Tr[j - 9, i]);
              elseif epos[n] > 0.024885 + dcx * (i - 2) and epos[n] <= 0.024885 + dcx * (i - 1)
then
                Q1[j, i, n] = 1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcx * (i - 1)
- epos[n]) * (0.024885 + dcx * (j - 10) - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i - 1]) + 1 /
(drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcx * (i - 1) - epos[n]) * (eypos[j] - (0.024885
+ dcx * (j - 10))) * (Tb[j, n, 1] - Tr[j - 9, i - 1]);
                for a1 in i + 1:4 loop
                  Q1[j, a1, n] = 0.0;
                end for;
              end if;
            elseif epos[n] > 0.024885 + dcx * zc and spos[n] > 0.024885 and 0.024885 + dcx * zc
> spos[n] then
              if 0.024885 + dcx * (i - 1) < spos[n] and spos[n] <= 0.024885 + dcx * i then
                Q1[j, i, n] = (-1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcx * i -
spos[n]) * (0.024885 + dcx * (j - 10) - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i])) - 1 / (drx /
(2 * krb) + dbx / (2 * k)) * (0.024885 + dcx * i - spos[n]) * (eypos[j] - (0.024885 + dcx * (j -
10))) * (Tb[j, n, 1] - Tr[j - 9, i]);
              elseif spos[n] >= 0.024885 + dcx * i then
                Q1[j, i, n] = 0.0;
              else
                Q1[j, i, n] = (-1 / (drx / (2 * krb) + dbx / (2 * k)) * dcx * (0.024885 + dcx *
(j - 10) - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i])) - 1 / (drx / (2 * krb) + dbx / (2 * k)) *
dcx * (eypos[j] - (0.024885 + dcx * (j - 10))) * (Tb[j, n, 1] - Tr[j - 9, i]);
              end if;
            elseif epos[n] > 0.024885 + dcx * zc and spos[n] < 0.024885 then
              Q1[j, i, n] = (-1 / (drx / (2 * krb) + dbx / (2 * k)) * dcx * (0.024885 + dcx * (j
- 10) - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i])) - 1 / (drx / (2 * krb) + dbx / (2 * k)) * dcx
* (eypos[j] - (0.024885 + dcx * (j - 10))) * (Tb[j, n, 1] - Tr[j - 9, i]);
            elseif epos[n] < 0.024885 then
              Q1[j, i, n] = 0.0;
            elseif spos[n] > 0.024885 + dcx * yc then
              Q1[j, i, n] = 0.0;
            end if;
            if i == 4 then
              Q[j, n] = Q1[j, 1, n] + Q1[j, 2, n] + Q1[j, 3, n] + Q1[j, 4, n];
            end if;
          end if;
        end for;
      end for;

// Second if condition in y position to evaluate 10^th heat sink base element.

    elseif eypos[j] > 0.024885 and sypos[j] < 0.024885 then
      for n in 1:zb loop
        for i in 1:zc loop
          if dbx > dcx then
            if spos[n] > 0.024885 and epos[n] < 0.024885 + dcx * zc then
              if 0.024885 + dcx * (i - 1) < spos[n] and spos[n] < 0.024885 + dcx * i then
```

```
                Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcz * i -
spos[n]) * (eypos[j] - 0.024885) * (Tb[j, n, 1] - Tr[j - 9, i]);
              elseif epos[n] > 0.024885 + dcz * i and spos[n] < 0.024885 + dcz * i then
                Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz * (eypos[j] -
0.024885) * (Tb[j, n, 1] - Tr[j - 9, i]);
              elseif spos[n] > 0.024885 + dcz * i then
                Q1[j, i, n] = 0.0;
              end if;
              if epos[n] > 0.024885 + dcz * (i - 1) and epos[n] <= 0.024885 + dcz * i then
                Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * (epos[n] - (0.024885 +
dcz * (i - 1))) * (eypos[j] - 0.024885) * (Tb[j, n, 1] - Tr[j - 9, i + 1]);
                for a in i + 1:4 loop
                  Q1[j, a, n] = 0.0;
                end for;
              end if;
            elseif spos[n] < 0.024885 and epos[n] < 0.024885 + dcz * zc and epos[n] > 0.024885
then
              if epos[n] > 0.024885 + dcz * (i - 1) then
                Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz * (eypos[j] -
0.024885) * (Tb[j, n, 1] - Tr[j - 9, i]);
              elseif epos[n] > 0.024885 + dcz * (i - 2) and epos[n] <= 0.024885 + dcz * (i - 1)
then
                Q1[j, i, n] = 1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcz * (i - 1)
- epos[n]) * (eypos[j] - 0.024885) * (Tb[j, n, 1] - Tr[j - 9, i - 1]);
                for a1 in i + 1:4 loop
                  Q1[j, a1, n] = 0.0;
                end for;
              end if;
            elseif epos[n] > 0.024885 + dcz * zc and spos[n] > 0.024885 and 0.024885 + dcz * zc
> spos[n] then
              if 0.024885 + dcz * (i - 1) < spos[n] and spos[n] <= 0.024885 + dcz * i then
                Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcz * i -
spos[n]) * (eypos[j] - 0.024885) * (Tb[j, n, 1] - Tr[j - 9, i]);
              elseif spos[n] >= 0.024885 + dcz * i then
                Q1[j, i, n] = 0.0;
              else
                Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz * (eypos[j] -
0.024885) * (Tb[j, n, 1] - Tr[j - 9, i]);
              end if;
            elseif epos[n] > 0.024885 + dcz * zc and spos[n] < 0.024885 then
              Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz * (eypos[j] - 0.024885)
* (Tb[j, n, 1] - Tr[j - 9, i]);
            elseif epos[n] < 0.024885 then
              Q1[j, i, n] = 0.0;
            elseif spos[n] > 0.024885 + dcz * yc then
              Q1[j, i, n] = 0.0;
            end if;
            if i == 4 then
              Q[j, n] = Q1[j, 1, n] + Q1[j, 2, n] + Q1[j, 3, n] + Q1[j, 4, n];
            end if;
          end if;
        end for;
      end for;

// Second if condition in y position to evaluate 14th heat sink base element.
    elseif eypos[j] > 0.024885 + dcz * yc and sypos[j] < 0.024885 + dcz * yc then
      for n in 1:zb loop
        for i in 1:zc loop
          if dbz > dcz then
            if spos[n] > 0.024885 and epos[n] < 0.024885 + dcz * zc then
              if 0.024885 + dcz * (i - 1) < spos[n] and spos[n] < 0.024885 + dcz * i then
                Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcz * i -
spos[n]) * (0.024885 + dcz * zc - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i]);
              elseif epos[n] > 0.024885 + dcz * i and spos[n] < 0.024885 + dcz * i then
                Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz * (0.024885 + dcz *
zc - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i]);
              elseif spos[n] > 0.024885 + dcz * i then
                Q1[j, i, n] = 0.0;
              end if;
              if epos[n] > 0.024885 + dcz * (i - 1) and epos[n] <= 0.024885 + dcz * i then
                Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * (epos[n] - (0.024885 +
dcz * (i - 1))) * (0.024885 + dcz * zc - sypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i + 1]);
                for a in i + 1:4 loop
                  Q1[j, a, n] = 0.0;
                end for;
```

```
            end if;
          elseif spos[n] < 0.024885 and epos[n] < 0.024885 + dcz * zc and epos[n] > 0.024885
then
            if epos[n] > 0.024885 + dcz * (i - 1) then
              Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz * (0.024885 + dcz *
zc - zypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i]);
            elseif epos[n] > 0.024885 + dcz * (i - 2) and epos[n] <= 0.024885 + dcz * (i - 1)
then
              Q1[j, i, n] = 1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcz * (i - 1)
- epos[n]) * (0.024885 + dcz * zc - zypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i - 1]);
              for a1 in i + 1:4 loop
                Q1[j, a1, n] = 0.0;
              end for;
            end if;
          elseif epos[n] > 0.024885 + dcz * zc and spos[n] > 0.024885 and 0.024885 + dcz * zc
> spos[n] then
            if 0.024885 + dcz * (i - 1) < spos[n] and spos[n] <= 0.024885 + dcz * i then
              Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcz * i -
spos[n]) * (0.024885 + dcz * zc - zypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i]);
            elseif spos[n] >= 0.024885 + dcz * i then
              Q1[j, i, n] = 0.0;
            else
              Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz * (0.024885 + dcz *
zc - zypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i]);
            end if;
          elseif epos[n] > 0.024885 + dcz * zc and spos[n] < 0.024885 then
            Q1[j, i, n] = -1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz * (0.024885 + dcz * zc
- zypos[j]) * (Tb[j, n, 1] - Tr[j - 10, i]);
          elseif epos[n] < 0.024885 then
            Q1[j, i, n] = 0.0;
          elseif spos[n] > 0.024885 + dcz * yc then
            Q1[j, i, n] = 0.0;
          end if;
          if i == 4 then
            Q[j, n] = Q1[j, 1, n] + Q1[j, 2, n] + Q1[j, 3, n] + Q1[j, 4, n];
          end if;
        end if;
      end for;
    end for;
  end if;
end for;

// Heat transfer from  pad to heat sink base
for j in 1:yr loop
  for n in 1:zb loop
    for i in 1:zr loop
      if epos[n] < 0.024885 + dcz * i and epos[n] > 0.024885 + dcz * (i - 1) then
        Qo[j, i] = (-1 / (drx / (2 * krb) + dbx / (2 * k)) * (epos[n] - (0.024885 + dcz * (i -
1))) * (eypos[j + 9] - (0.024885 + dcz * (j - 1))) * (Tr[j, i] - Tb[j + 9, n, 1])) - 1 / (drx /
(2 * krb) + dbx / (2 * k)) * (epos[n] - (0.024885 + dcz * (i - 1))) * (0.024885 + dcz * j -
zypos[j + 10]) * (Tr[j, i] - Tb[j + 10, n, 1]) - 1 / (drx / (2 * krb) + dbx / (2 * k)) *
(0.024885 + dcz * i - spos[n + 1]) * (eypos[j + 9] - (0.024885 + dcz * (j - 1))) * (Tr[j, i] -
Tb[j + 9, n + 1, 1]) - 1 / (drx / (2 * krb) + dbx / (2 * k)) * (0.024885 + dcz * i - spos[n +
1]) * (0.024885 + dcz * j - zypos[j + 10]) * (Tr[j, i] - Tb[j + 10, n + 1, 1]);
      end if;
      if spos[n] < 0.024885 + dcz * (i - 1) and epos[n] > 0.024885 + dcz * i then
        Qo[j, i] = (-1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz * (eypos[j + 9] - (0.024885 +
dcz * (j - 1))) * (Tr[j, i] - Tb[j + 9, n, 1])) - 1 / (drx / (2 * krb) + dbx / (2 * k)) * dcz *
(0.024885 + dcz * j - zypos[j + 10]) * (Tr[j, i] - Tb[j + 10, n, 1]);
      end if;
    end for;
  end for;
end for;
```

```
      // Heat dissipation at finite fin base volume
      for j in 1:yb loop

        for n in 1:zb loop

          for m in 1:xb loop
            // Heat dissipation at fin base volume
            if j==1 and n==1 and m==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAB*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAB*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz +
Q[j,n];//

            elseif j==1 and n==1 and m==xb then
              rho*VB*C*der(Tb[j,n,m]) = - h*SABF*(Tb[j,n,m]-Tamb) /*-
sigma*e*SABF*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz -
(1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(Tb[j,n,m]-T[1,n,j]);

            elseif j==yb and n==1 and m==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAB*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAB*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz +
Q[j,n];// + Q[j,n]

            elseif j==yb and n==1 and m==xb then
              rho*VB*C*der(Tb[j,n,m]) = - h*SABF*(Tb[j,n,m]-Tamb) /*-
sigma*e*SABF*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz -
(1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(Tb[j,n,m]-T[1,n,j]);

            elseif j==1 and n==zb and m==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAB*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAB*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz +
Q[j,n];// + Q[j,n]

            elseif j==1 and n==zb and m==xb then
              rho*VB*C*der(Tb[j,n,m]) = - h*SABF*(Tb[j,n,m]-Tamb) /*-
sigma*e*SABF*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz -
(1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(Tb[j,n,m]-T[1,n,j]);

            elseif j==yb and n==zb and m==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAB*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAB*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz +
Q[j,n];// + Q[j,n]

            elseif j==yb and n==zb and m==xb then
              rho*VB*C*der(Tb[j,n,m]) = - h*SABF*(Tb[j,n,m]-Tamb) /*-
sigma*e*SABF*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz -
(1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(Tb[j,n,m]-T[1,n,j]);

            elseif j==1 and m==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAMB*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAMB*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz -
k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz + Q[j,n];// + Q[j,n]

            elseif j==1 and m==xb then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAMBF*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAMBF*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz -
k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz - (1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(Tb[j,n,m]-
T[1,n,j]);
```

```
            elseif j==yb and m==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAMB*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAMB*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz -
k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz + Q[j,n];// + Q[j,n]

            elseif j==yb and m==xb then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAMBF*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAMBF*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz -
k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz - (1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(Tb[j,n,m]-
T[1,n,j]);

            elseif n==1 and m==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAMB8*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAMB8*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby -
k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz + Q[j,n];// + Q[j,n]

            elseif n==1 and m==xb then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAMB8F*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAMB8F*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby -
k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz - (1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(Tb[j,n,m]-
T[1,n,j]);

            elseif n==zb and m==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAMB8*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAMB8*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby -
k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz + Q[j,n];// + Q[j,n]

            elseif n==zb and m==xb then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAMB8F*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAMB8F*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby -
k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz - (1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(Tb[j,n,m]-
T[1,n,j]);

            elseif n==1 and j==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*(CSABY+CSABZ)*(Tb[j,n,m]-Tamb) /*-
sigma*e*(CSABY+CSABZ)*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-
Tb[j,n,m+1])/dbx - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx - k*CSABY*(Tb[j,n,m]-
Tb[j+1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz;

            elseif n==1 and j==yb then
              rho*VB*C*der(Tb[j,n,m]) = - h*(CSABY+CSABZ)*(Tb[j,n,m]-Tamb) /*-
sigma*e*(CSABY+CSABZ)*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-
Tb[j,n,m+1])/dbx - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx - k*CSABY*(Tb[j,n,m]-Tb[j-
1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz;

            elseif n==zb and j==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*(CSABY+CSABZ)*(Tb[j,n,m]-Tamb) /*-
sigma*e*(CSABY+CSABZ)*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-
Tb[j,n,m+1])/dbx - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx - k*CSABY*(Tb[j,n,m]-
Tb[j+1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz;

            elseif n==zb and j==zb then
              rho*VB*C*der(Tb[j,n,m]) = - h*(CSABY+CSABZ)*(Tb[j,n,m]-Tamb) /*-
sigma*e*(CSABY+CSABZ)*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-
Tb[j,n,m+1])/dbx - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx - k*CSABY*(Tb[j,n,m]-Tb[j-
1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz;

            elseif j==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*CSABY*(Tb[j,n,m]-Tamb) /*-
sigma*e*CSABY*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx - k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby -
k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz;
```

```
            elseif j==yb then
              rho*VB*C*der(Tb[j,n,m]) = - h*CSABY*(Tb[j,n,m]-Tamb) /*-
sigma*e*CSABY*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx - k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby -
k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz;

            elseif n==1 then
              rho*VB*C*der(Tb[j,n,m]) = - h*CSABZ*(Tb[j,n,m]-Tamb) /*-
sigma*e*CSABZ*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx - k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz;

            elseif n==zb then
              rho*VB*C*der(Tb[j,n,m]) = - h*CSABZ*(Tb[j,n,m]-Tamb) /*-
sigma*e*CSABZ*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx - k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz;

            elseif m==1 then

              rho*VB*C*der(Tb[j,n,m]) = - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby -
k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz + Q[j,n];

            elseif m==xb then
              rho*VB*C*der(Tb[j,n,m]) = - h*SAMMBF*(Tb[j,n,m]-Tamb) /*-
sigma*e*SAMMBF*((Tb[j,n,m])^4-(Tamb)^4)*/ - k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx -
k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby - k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby -
k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz -
(1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(Tb[j,n,m]-T[1,n,j]);

            else
              rho*VB*C*der(Tb[j,n,m]) = - k*CSABY*(Tb[j,n,m]-Tb[j+1,n,m])/dby -
k*CSABY*(Tb[j,n,m]-Tb[j-1,n,m])/dby - k*CSABZ*(Tb[j,n,m]-Tb[j,n-1,m])/dbz  -
k*CSABZ*(Tb[j,n,m]-Tb[j,n+1,m])/dbz - k*CSABX*(Tb[j,n,m]-Tb[j,n,m+1])/dbx -
k*CSABX*(Tb[j,n,m]-Tb[j,n,m-1])/dbx;

            end if;

          end for;

        end for;

      end for;


    // Heat dissipation at finite fin volume

      for m in 1:a loop

        for i in 1:x loop

          for n in 1:z loop
              // Heat dissipation at finite fin volume
              if i==1 and n==1 then
                rho*V*C*der(T[i,n,m]) = - h*SA*(T[i,n,m]-Tamb) /*-
sigma*e*SA*((T[i,n,m])^4-(Tamb)^4)*/ - k*CSAX*(T[i,n,m]-T[i+1,n,m])/dx -
k*CSAZ*(T[i,n,m]-T[i,n+1,m])/dz - (1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(T[i,n,m]-
Tb[m,n,xb]);

              elseif i==x and n==1 then
                rho*V*C*der(T[i,n,m]) = - h*SA*(T[i,n,m]-Tamb) /*-
sigma*e*SA*((T[i,n,m])^4-(Tamb)^4)*/ - h*CSAX*(T[i,n,m]-Tamb) /*-
sigma*e*CSAX*((T[i,n,m])^4-(Tamb)^4)*/ - k*CSAX*(T[i,n,m]-T[i-1,n,m])/dx -
k*CSAZ*(T[i,n,m]-T[i,n+1,m])/dz;
```

```
              elseif i==1 and n==z then
                rho*V*C*der(T[i,n,m]) = - h*SA*(T[i,n,m]-Tamb) /*-
sigma*e*SA*((T[i,n,m])^4-(Tamb)^4)*/ - k*CSAX*(T[i,n,m]-T[i+1,n,m])/dx -
k*CSAZ*(T[i,n,m]-T[i,n-1,m])/dz - (1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(T[i,n,m]-
Tb[m,n,xb]);

              elseif i==x and n==z then
                rho*V*C*der(T[i,n,m]) = - h*SA*(T[i,n,m]-Tamb) /*-
sigma*e*SA*((T[i,n,m])^4-(Tamb)^4)*/ - h*CSAX*(T[i,n,m]-Tamb) /*-
sigma*e*CSAX*((T[i,n,m])^4-(Tamb)^4)*/ - k*CSAX*(T[i,n,m]-T[i-1,n,m])/dx -
k*CSAZ*(T[i,n,m]-T[i,n-1,m])/dz;

              elseif i==1 then
                rho*V*C*der(T[i,n,m]) = - h*SAM*(T[i,n,m]-Tamb) /*-
sigma*e*SAM*((T[i,n,m])^4-(Tamb)^4)*/ - k*CSAX*(T[i,n,m]-T[i+1,n,m])/dx -
k*CSAZ*(T[i,n,m]-T[i,n+1,m])/dz - k*CSAZ*(T[i,n,m]-T[i,n-1,m])/dz -
(1/(dx/(2*k)+(dbx/(2*k))))*CSAX*(T[i,n,m]-Tb[m,n,xb]);

              elseif i==x then
                rho*V*C*der(T[i,n,m]) = - h*SAM*(T[i,n,m]-Tamb) /*-
sigma*e*SAM*((T[i,n,m])^4-(Tamb)^4)*/ - h*CSAX*(T[i,n,m]-Tamb) /*-
sigma*e*CSAX*((T[i,n,m])^4-(Tamb)^4)*/ - k*CSAX*(T[i,n,m]-T[i-1,n,m])/dx -
k*CSAZ*(T[i,n,m]-T[i,n+1,m])/dz - k*CSAZ*(T[i,n,m]-T[i,n-1,m])/dz;

              elseif n==1 then
                rho*V*C*der(T[i,n,m]) = - h*SA*(T[i,n,m]-Tamb) /*-
sigma*e*SA*((T[i,n,m])^4-(Tamb)^4)*/ - k*CSAX*(T[i,n,m]-T[i+1,n,m])/dx -
k*CSAX*(T[i,n,m]-T[i-1,n,m])/dx - k*CSAZ*(T[i,n,m]-T[i,n+1,m])/dz;

              elseif n==z then
                rho*V*C*der(T[i,n,m]) = - h*SA*(T[i,n,m]-Tamb) /*-
sigma*e*SA*((T[i,n,m])^4-(Tamb)^4)*/ - k*CSAX*(T[i,n,m]-T[i+1,n,m])/dx -
k*CSAX*(T[i,n,m]-T[i-1,n,m])/dx - k*CSAZ*(T[i,n,m]-T[i,n-1,m])/dz;

              else
                rho*V*C*der(T[i,n,m]) = - h*SAM*(T[i,n,m]-Tamb) /*-
sigma*e*SAM*((T[i,n,m])^4-(Tamb)^4))*/ - k*CSAX*(T[i,n,m]-T[i+1,n,m])/dx -
k*CSAX*(T[i,n,m]-T[i-1,n,m])/dx - k*CSAZ*(T[i,n,m]-T[i,n-1,m])/dz -
k*CSAZ*(T[i,n,m]-T[i,n+1,m])/dz;

              end if;

            end for;

          end for;

        end for;

end HeatSinkModelv1_4;
```

Implemented model in Modelica

Bibliography

[1] Open Source Modelica Consortium. Openmodelica user's guide. Available at `https://openmodelica.org/doc/OpenModelicaUsersGuide/OpenModelicaUsersGuide\-latest.pdf` (2019/07/09).

[2] ANSYS Inc. Ansys fluent 12.0 theory guide. Available at `http://www.afs.enea.it/project/neptunius/docs/fluent/html/th/main_pre.htm` (2019/05/12).

[3] Thermal Engineering Associates Inc. TTC-1002 thermal test chip applications manual. Available at `http://thermengr.net/TTC/TTC-1002_%20Manual.pdf` (2019/03/05).

[4] F.P. Incropera, D.P. Dewitt, T.L. Bergman, and A.S. Lavine. *Foundations of Heat Transfer*. John Wiley & Sons, Singapore, 6 edition, 2013.

[5] J.H. Lienhard IV and J.H. Lienhard V. *A Heat Transfer Textbook*. Phlogiston Press, Cambridge, Massachusetts, 4 edition, 2018.

[6] S.J. Sherwin J. Peiro. Finite difference, finite element and finite volume methods for partial differential equations. In S. Yip, editor, *Handbook of Materials Modeling, Volume I: Methods and Models*, pages 1–32. Springer, Netherlands, 2005.

[7] D.A. Kaminski and M.K. Jensen. *Introduction to Thermal and Fluid Engineering*. John Wiley & Sons, New York, 1 edition, 2005.

[8] A.D. Kraus, A. Aziz, and J. Welty. *Extended Surface Heat Transfer*. John Wiley & Sons, Singapore, 1 edition, 2001.

[9] Aavid Thermalloy. HS483-ND heat sink. Available at `https://www.digikey.com/product-detail/en/aavid-thermal-division-of-boyd-corporation/342946/HS483-ND/6150561` (2019/03/06).