THREE-DIMENSIONAL MULTI-PHYSICS THERMAL-ANALYSIS SYSTEM FOR ELECTRONIC SYSTEMS

Fukui M.*, Watanabe S., Omura T., Kitagawa Y., Tsukiyama S., and Shirakawa I.
Department of Electronic and Computer Engineering,
Ritsumeikan University,
Kusatsu, Shiga, 525-8577 Japan
e-mail : mfukui@se.ritsumei.ac.jp

ABSTRACT

In this paper a 3D multi-physical thermal analysis procedure is devised dedicatedly for LSI packages by means of a realistic approximative approach, which is constructed by applying a macro-model to each phenomenon of heat generation, heat transfer, and heat dissipation in a solid material. As a test example, a set of LSI package and surrounding air is formulated with the use of a 3D thermal grid network of size 256x256x10, to which the application of the proposed procedure yields the accuracy and rapidity practically guaranteed by comparison with the numerical analysis and survey experiment.

INTRODUCTION

With the rapid advances in integration of circuits as well as in energy density of joule heat generation, the thermal analysis of LSI packages has been getting more and more important, for which a multi-physical methodology may acquire effective solutions. Considering that the precise 3D multi-physical thermal analysis of LSI packages necessitates an integrative approach to the analysis of those complex phenomena in solid materials which are caused by heat generation, heat transfer, and heat dissipation, a huge sequence of highly precise numerical computations might have to be carried out through the use of a sort of supercomputer. However, such a rigorous handling would be too unrealistic for the practical thermal analysis of LSI packages in terms of computing cost and labour.

The thermal analysis of electronic systems was attempted on the basis of a 2D thermal grid network model by applying the backward Euler method run in parallel on GPGPU (General-Purpose computing on Geographic Processing Unit) using CSG (Compressed Sparse expression for Grid structure) for reducing huge memory space [1,2]. To expand this methodology to treat more precisely the vertical heat transfer, the present paper devises a 3D multi-physical thermal analysis procedure by means of a realistic approximation scheme, which is constructed by applying a macro-model to each phenomenon of heat generation, heat transfer, and heat dissipation in LSI packages.

Specifically, the proposed approximation procedure is performed as outlined in what follows:
(1) Heat transfer of solid materials, heat flow of generated heat, and heat resistance and calorific capacity are modelled in a thermal 3D grid network, where the heat dissipation is performed by air flow surrounding a solid, and the heat resistance of air is calculated on the basis of the amount of wind cooling the solid.
(2) The thermal analysis is implemented on the basis of the current and resistance distribution in a thermal 3D grid network, where in order to solve a set of simultaneous linear equations by repeated application of Jacobi method, the backward Euler method is executed in parallel on GPGPU using CSCG (Compressed Sparse for Cubic Grid structure), which is an extension of data structure of CSG.

As a test example, a set of LSI package and surrounding air is modelled by a 3D grid network of size 256x256x10. The computation time spent on the thermal analysis of this model for 10 seconds is 1124 seconds by GPU (NVIDIA GeForce GTX 580 --- 512 Core, 1.54 GHz, double precision 198GFLOPS), which is 22.8 times faster than that obtained by CPU (Core i7 by 64 bit OS). As for the accuracy, a distinctive difference between the Elmer method and the finite element method can be observed only in the temperature by 0.1 degree or less, and no more in other items. Furthermore, computation by proposed method is more efficient than Elmer.

PHYSICAL MODELING BY THERMAL NETWORK

Similarly to the analysis target expressed by a thermal network in [3], the proposed thermal network procedure treats the heat transfer by the analogy of electric current in electric circuits. The model is expressed by the following parameters, (1) node temperature *T*, (2) heat flow *Q*t between nodes, (3) thermal resistance *θ*, and (4) heat capacitance *C*t. The Ohm's law and theories of transient response for electric circuits can be applied similarly to thermal networks. Table I shows the correspondence of the parameters between electric circuits and thermal circuits.

Table 1 Correspondence of parameters

Electric circuit	Thermal circuit
Temperature: T [K]	Voltage: E [V]
Heat flow rate: Qt [J/S]	Current: I [A]
Thermal resistance: θ [K/W]	Resistance: R [Ω]
Heat capacitance: Ct [J/K]	Capacitance: C [F]
Heat quantity: H [J]	Electric charge: q [C]

As an easy target of the thermal analysis, an LSI package as shown in Figure 1 is a good example, where the LSI chip is mounted on the printed board, the solder layer connects the LSI substrate and printed board, and active devices (transistors) are located at the surface of the LSI substrate, and the protection material covers the surface of the LSI. These active devices generate the joule heat in their operation time, and air surrounds the peripherals of the system.

Electric devices illustrated in Figure 1 are modelled by a 3D thermal network. The whole region is divided into connected 3D structures as shown in Figure 2, where Q_0 and C denote the heat generation and heat capacity at each point corresponding to a node, and a thermal resistance is allocated between adjacent nodes.

HEAT TRANSFER MODELING OF AIR

Although air as well as a solid can be expressed using heat resistance, they are slightly complicated, since they are fluid. As for the heat conduction of air, macro models in the case of a natural convection and a forced convection are proposed [3-7]. The heat resistance of air is given by formula (1): $A[m^2]$ is the surface area size of the object, and h[W/m^2 ·K] a coefficient of heat transfer, given by formula (2), where *Nu* is Nusslet number, λ_a [W/(m · K)] thermal conductivity of air, and d [m] characteristic length.

$$\theta_a = \frac{1}{Ah} \tag{1}$$

$$h = \frac{Nu\lambda_a}{d} \tag{2}$$

Nusslet number *Nu* is difficult to predict in many cases. Thus it is experientially predicted from many prediction types. In the case of a natural convection for a flat plate, *N*u is given by the following formula (3): *Gr* (Grashof number) and *Pr* (Prandtl number) are given by formulas (4) and (5), where g [m/s^2] denotes acceleration due to gravity, and β [1/K] is a volume expansion coefficient of fluid. In the case of air, β is given approximated by 1/T at temperature T [K], $\nu[m^2/s]$ is kinematic viscosity, $\alpha[m^2/s]$ thermal diffusivity, and T_s *and* T_a surface and ambient temperatures, respectively.

$$Nu = \left(\frac{Pr}{2.4+4.9(Pr)^{1/2}+5Pr}\right)^{1/4}(GrPr)^{1/4} \tag{3}$$

$$Gr = \frac{g\beta(T_s - T_a)d^3}{\nu^2} \tag{4}$$

$$Pr = \frac{\nu}{a} \tag{5}$$

In the case of a forced convection, *Nu* is given by formula (6), where *Re* is Reynolds number, *C* and *n* are experientially fixed as $C = 0.193$ and $n = 0.618$, respectively, and *Re* is given by formula (7), where *U*[m/s] is wind velocity. Formula (8) is derived with the use of formulas (5-7). By increasing wind velocity *U*, the value of θ_a becomes smaller. Consequently, the object temperature can be reduced. Here, K is a compensation parameter to the experimental condition, which is set to 0.30.

$$Nu = C(Re^n \cdot P_r^{\frac{1}{3}}) \tag{6}$$

$$Re = \frac{U \cdot d}{\nu} \tag{7}$$

$$\theta_a = \frac{K \cdot d}{A \cdot C\left((\frac{U \cdot d}{\nu})^n \cdot P_r^{\frac{1}{3}}\right)\lambda_a} \tag{8}$$

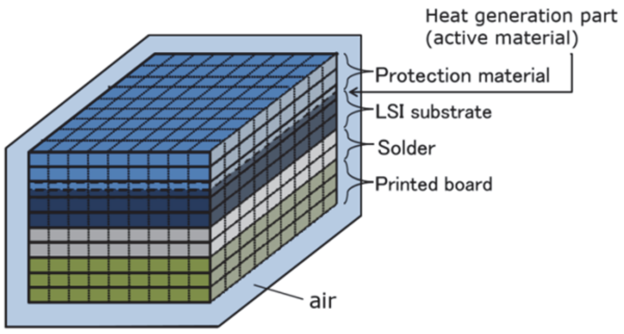

Figure 1 An example of the thermal analysis object

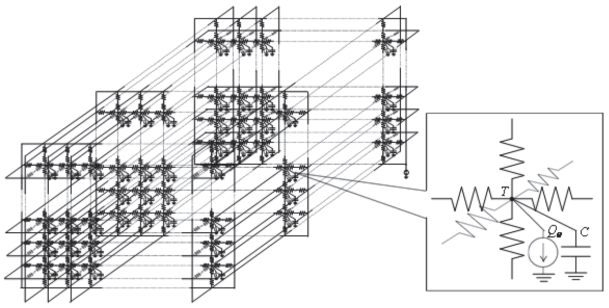

Figure 2 3D thermal network mode

HEAT ANALYSIS ALGORITHM

Each node temperature of the thermal network can be analyzed by solving the differential equation, where the backward Euler algorithm [8] is used. It has a high convergence, and is generally used for circuit simulators. The general formula of the algorithm is represented by equations (9) and (10).

$$\frac{dh(t+\Delta t)}{dt} = \lim_{\Delta t \to 0} \frac{h(t+\Delta t) - h(\Delta t)}{\Delta t} \approx \frac{h(t+\Delta t) - h(t)}{\Delta t}, \quad \Delta t > 0 \tag{9}$$

$$h(t+\Delta t) = h(t) + \Delta t\dot{h}(t+\Delta t), \quad \Delta t > 0 \tag{10}$$

Temperature $T^{(i,j,k)}$of node (i,j,k) is calculated by formula (11). In this formula, $H^{(i,j,k)}$ and $C^{(i,j,k)}$ are heat quantity and heat capacity, respectively, of the node.

$$T^{(i,j,k)} = \frac{H^{(i,j,k)}}{C^{(i,j,k)}} \tag{11}$$

The heat quantity $H^{(i,j,k)}(t+\Delta t)$ is given by $H^{(i,j,k)}(t)$ and the heat flow rate $Q\_all^{(i,j,k)}\ (t+\Delta t)$, as shown in formula (12).

$$H^{(i,j,k)}(t+\Delta t) = H^{(i,j,k)}(t) + Q\_all^{(i,j,k)}(t+\Delta t) \times \Delta t \tag{12}$$

$Q\_all^{(i,j,k)}\ (t+\Delta t)$ is given by sum of heat flow rate from the six neighbour nodes and $-Q_0^{(i,j,k)}$ (see Figure 2). Here, ${Q_{all}}^{(i,j,k)}(t+\Delta t)$ is equal to $\dot{H}^{(i,j,k)}(t+\Delta t)$, and hence it is shown as an actual example of formula (10).

$$Q_{all}^{(i,j,k)}(t+\Delta t) = \sum_{\substack{(x,y,z)=(i-1,j,k),(i+1,j,k),(i,j-1,k),\\(i,j+1,k),(i,j,k+1),(i,j,k-1)}} \frac{T^{(\mathrm{i,j,k})}(t+\Delta t)-T^{(\mathrm{x,y,z})}(t+\Delta t)}{R_{(i,j,k)-(x,y,z)}} - Q_0 \qquad (13)$$

Equations (11-13) derive the following formula.

$$\frac{\Delta \mathrm{t}}{\mathrm{Ct}}\begin{bmatrix}-\frac{1}{\mathrm{R}} & -\frac{1}{\mathrm{R}} & -\frac{1}{\mathrm{R}} & -\frac{1}{\mathrm{R}} & -\frac{1}{\mathrm{R}} & -\frac{1}{\mathrm{R}} & \frac{6\Delta \mathrm{t}+\mathrm{CtR}}{\Delta \mathrm{tR}}\end{bmatrix}\begin{bmatrix}T^{(i-1,j,k)}(t+\Delta t)\\ T^{(i+1,j,k)}(t+\Delta t)\\ T^{(i,j-1,k)}(t+\Delta t)\\ T^{(i,j+1,k)}(t+\Delta t)\\ T^{(i,j,k+1)}(t+\Delta t)\\ T^{(i,j,k-1)}(t+\Delta t)\\ T^{(i,j,k)}(t+\Delta t)\end{bmatrix} = T^{(\mathrm{i,j,k})}(t)-\frac{\Delta \mathrm{t}}{\mathrm{Ct}}Q_0 \qquad (14)$$

Equation (14) is given for each node, and equations of all nodes form linear simultaneous linear equations. Also, by the regularity of the structure the matrix is diagonal advantage.

Next, the application method to a three-dimensional thermal network model is described. If circuit size applies to the three-dimensional thermal network of 3x3x3, it becomes simultaneous linear equations of an equation (15). Heat resistance θ_a is connected between a surrounding node and peripheral node of ambient temperatureT_a, and the node of a circuit end touches air. The heat resistance theta used here uses the thermal conductivity of each domain.

$$\begin{bmatrix}\mathrm{S} & \mathrm{U} & 0\\ \mathrm{U} & \mathrm{T} & \mathrm{U}\\ 0 & \mathrm{U} & \mathrm{S}\end{bmatrix}[\mathrm{V}]=[\mathrm{W}] \qquad (15)$$

S =

$$\begin{bmatrix}
\frac{3\Delta t}{C\theta}+\frac{3\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0 & 0 & 0\\
-\frac{\Delta t}{C\theta} & \frac{4\Delta t}{C\theta}+\frac{2\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0 & 0\\
0 & -\frac{\Delta t}{C\theta} & \frac{3\Delta t}{C\theta}+\frac{3\Delta t}{C\theta_a}+1 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0\\
-\frac{\Delta t}{C\theta} & 0 & 0 & \frac{4\Delta t}{C\theta}+\frac{2\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & 0 & 0\\
0 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & \frac{5\Delta t}{C\theta}+\frac{1\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & 0\\
0 & 0 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & \frac{4\Delta t}{C\theta}+\frac{2\Delta t}{C\theta_a}+1 & 0 & 0 & -\frac{\Delta t}{C\theta}\\
0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & \frac{3\Delta t}{C\theta}+\frac{3\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta} & 0\\
0 & 0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & \frac{4\Delta t}{C\theta}+\frac{2\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta}\\
0 & 0 & 0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & \frac{3\Delta t}{C\theta}+\frac{3\Delta t}{C\theta_a}+1
\end{bmatrix}$$

T =

$$\begin{bmatrix}
\frac{4\Delta t}{C\theta}+\frac{2\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0 & 0 & 0\\
-\frac{\Delta t}{C\theta} & \frac{5\Delta t}{C\theta}+\frac{1\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0 & 0\\
0 & -\frac{\Delta t}{C\theta} & \frac{4\Delta t}{C\theta}+\frac{2\Delta t}{C\theta_a}+1 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0\\
-\frac{\Delta t}{C\theta} & 0 & 0 & \frac{5\Delta t}{C\theta}+\frac{1\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & 0 & 0\\
0 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & \frac{6\Delta t}{C\theta}+1 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & 0\\
0 & 0 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & \frac{5\Delta t}{C\theta}+\frac{1\Delta t}{C\theta_a}+1 & 0 & 0 & -\frac{\Delta t}{C\theta}\\
0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & \frac{4\Delta t}{C\theta}+\frac{2\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta} & 0\\
0 & 0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & \frac{5\Delta t}{C\theta}+\frac{1\Delta t}{C\theta_a}+1 & -\frac{\Delta t}{C\theta}\\
0 & 0 & 0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & -\frac{\Delta t}{C\theta} & \frac{4\Delta t}{C\theta}+\frac{2\Delta t}{C\theta_a}+1
\end{bmatrix}$$

U =

$$\begin{bmatrix}
-\frac{\Delta t}{C\theta} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\Delta t}{C\theta} & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\Delta t}{C\theta}
\end{bmatrix}$$

$$\mathrm{V}=\begin{bmatrix}
T^{(0,\ 0,\ 0)}(t+\Delta t)\\ T^{(0,\ 1,\ 0)}(t+\Delta t)\\ T^{(0,\ 2,\ 0)}(t+\Delta t)\\ T^{(1,\ 0,\ 0)}(t+\Delta t)\\ T^{(1,\ 1,\ 0)}(t+\Delta t)\\ T^{(1,\ 2,\ 0)}(t+\Delta t)\\ T^{(2,\ 0,\ 0)}(t+\Delta t)\\ T^{(2,\ 1,\ 0)}(t+\Delta t)\\ T^{(2,\ 2,\ 0)}(t+\Delta t)\\ T^{(0,\ 0,\ 1)}(t+\Delta t)\\ T^{(0,\ 1,\ 1)}(t+\Delta t)\\ T^{(0,\ 2,\ 1)}(t+\Delta t)\\ T^{(1,\ 0,\ 1)}(t+\Delta t)\\ T^{(1,\ 1,\ 1)}(t+\Delta t)\\ T^{(1,\ 2,\ 1)}(t+\Delta t)\\ T^{(2,\ 0,\ 1)}(t+\Delta t)\\ T^{(2,\ 1,\ 1)}(t+\Delta t)\\ T^{(2,\ 2,\ 1)}(t+\Delta t)\\ T^{(0,\ 0,\ 2)}(t+\Delta t)\\ T^{(0,\ 1,\ 2)}(t+\Delta t)\\ T^{(0,\ 2,\ 2)}(t+\Delta t)\\ T^{(1,\ 0,\ 2)}(t+\Delta t)\\ T^{(1,\ 1,\ 2)}(t+\Delta t)\\ T^{(1,\ 2,\ 2)}(t+\Delta t)\\ T^{(2,\ 0,\ 2)}(t+\Delta t)\\ T^{(2,\ 1,\ 2)}(t+\Delta t)\\ T^{(2,\ 2,\ 2)}(t+\Delta t)
\end{bmatrix}
\qquad
\mathrm{W}=\begin{bmatrix}
T^{(0,\ 0,\ 0)}(t)-\frac{\Delta t}{C}(Q_s-3\frac{T_a}{\theta_a})\\
T^{(0,\ 1,\ 0)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(0,\ 2,\ 0)}(t)-\frac{\Delta t}{C}(Q_s-3\frac{T_a}{\theta_a})\\
T^{(1,\ 0,\ 0)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(1,\ 1,\ 0)}(t)-\frac{\Delta t}{C}(Q_s-\frac{T_a}{\theta_a})\\
T^{(1,\ 2,\ 0)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(2,\ 0,\ 0)}(t)-\frac{\Delta t}{C}(Q_s-3\frac{T_a}{\theta_a})\\
T^{(2,\ 1,\ 0)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(2,\ 2,\ 0)}(t)-\frac{\Delta t}{C}(Q_s-3\frac{T_a}{\theta_a})\\
T^{(0,\ 0,\ 1)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(0,\ 1,\ 1)}(t)-\frac{\Delta t}{C}(Q_s-\frac{T_a}{\theta_a})\\
T^{(0,\ 2,\ 1)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(1,\ 0,\ 1)}(t)-\frac{\Delta t}{C}(Q_s-\frac{T_a}{\theta_a})\\
T^{(1,\ 1,\ 1)}(t)-\frac{\Delta t}{C}Q_s\\
T^{(1,\ 2,\ 1)}(t)-\frac{\Delta t}{C}(Q_s-\frac{T_a}{\theta_a})\\
T^{(2,\ 0,\ 1)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(2,\ 1,\ 1)}(t)-\frac{\Delta t}{C}(Q_s-\frac{T_a}{\theta_a})\\
T^{(2,\ 2,\ 1)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(0,\ 0,\ 2)}(t)-\frac{\Delta t}{C}(Q_s-3\frac{T_a}{\theta_a})\\
T^{(0,\ 1,\ 2)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(0,\ 2,\ 2)}(t)-\frac{\Delta t}{C}(Q_s-3\frac{T_a}{\theta_a})\\
T^{(1,\ 0,\ 2)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(1,\ 1,\ 2)}(t)-\frac{\Delta t}{C}(Q_s-\frac{T_a}{\theta_a})\\
T^{(1,\ 2,\ 2)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(2,\ 0,\ 2)}(t)-\frac{\Delta t}{C}(Q_s-3\frac{T_a}{\theta_a})\\
T^{(2,\ 1,\ 2)}(t)-\frac{\Delta t}{C}(Q_s-2\frac{T_a}{\theta_a})\\
T^{(2,\ 2,\ 2)}(t)-\frac{\Delta t}{C}(Q_s-3\frac{T_a}{\theta_a})
\end{bmatrix}$$

Here, we must consider the fact that θ_a changes by the change of the temperature of solid surface. Thus, we need to add a step to update θ_a . Then, the Jacobi Method is employed to solve simultaneous linear equations by repeating the calculation with the inverse matrix approximately. Jacobi Method repeats the calculation that follows equation (16) until solution has sufficient accuracy.

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i}^{N} a_{ij}x_j^{(k-1)}\right) \qquad (16)$$

It is suited for parallelization, because each calculation has no dependence of the order. It converges since the matrix is diagonal advantage. The backward Euler Method consumes memory space for the matrix in equation (15). To improve the efficiency of the calculation in equation (16), we must use an efficient data structure for the sparse matrix.

COMPRESSED SPARSE FOR CUBIC GRID STRUCTURE

In GPU program, the reduction of memory space is very important along with parallelism allocation. To solve this problem, CSCG (Compressed Sparse for Cubic Grid structure) is employed, where CSCG is a modification of CSG (Compressed Sparse expression for Grid structure), which is suitable for 2D power grid structure [1, 2]. Figure 3 shows an example of CSG. Power grid circuit matrix A is divided into diagonal matrix D and sparse matrix R. The diagonal matrix D stores only diagonal elements as column vector d. Diagonal elements is located at the position of d, so it can be directly accessed. CSG is using effectively the regularity by which the node of a 2D network is restricted to connection with four-direction nodes. In addition, in a 3D thermal network, there is connection with a total of six adjacent nodes of the upper lower layer vertically and horizontally. Therefore, in addition to CSG, CSCG has the value and index including an upper-and-lower layer to a total of six adjacent nodes. Let i, j, and k denote the numbers of each row, column, and layer, respectively, and let N be the horizontal number of grids. Since CSCG as shown in Figure 4 refers to the index and value of each ingredient directly, it processes efficiently. As a prior work, DIA (diagonal format) [9], one of improvement of CSR, was used in the sparse matrix-vector multiplication. In order to perform reference of $x_q{}^{(m-1)}$ and b_p from the peak (i, j, k), a total of 18 multiplications, 15 additions, three subtractions, 14 references to a shared memory, and 50 steps is required for DIA. However, CSCG(s) are 20 references to a shared memory, and are about 40% of steps compared with DIA. Since a Jacobi method needs the repeated calculation of a formula (9), the increase in efficiency of data access by this technique is useful.

CALCULATION BY GPGPU

First, the simulator generates the matrix of the thermal network. It is compressed to CSCG. Then, initial distribution of the temperature is set, it is set as initial solution of Jacobi method. The CPU performs these initializations. Then, the GPU performs Jacobi calculation and updating of θ_a, thermal resistance of air. At each time step, Jacobi calculation is repeated until output approaches set error enough to have sufficient accuracy. After that, θ_a is recalculated for each peripheral node by the node temperature and air temperature. When simulation time reached to the pre-set time, simulation result is submitted to user. The data which is needed by GPU is written in the global memory by CPU at first. Later, the CSCG data is rewritten in the shared memory because the temperature calculation is executed repeatedly by GPU to improve the performance. That is because the accessing time to the shared memory is about 1000 times faster than to the global memory. By CSCG, memory usage is reduced significantly, and it becomes possible to use the shared memory of the small amount.

$$
\begin{array}{c}
d \\
\begin{bmatrix} a_{11} \\ a_{22} \\ a_{33} \\ a_{44} \\ a_{55} \\ a_{66} \\ a_{77} \\ a_{88} \\ a_{99} \end{bmatrix}
\end{array}
\qquad
\begin{array}{c}
R\_value \\
\begin{array}{ccccc}
 & Top & Left & Right & Down \\
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{matrix} &
\begin{bmatrix} * \\ * \\ * \\ a_{41} \\ a_{52} \\ a_{63} \\ a_{74} \\ a_{85} \\ a_{96} \end{bmatrix} &
\begin{bmatrix} * \\ a_{21} \\ a_{32} \\ * \\ a_{54} \\ a_{65} \\ * \\ a_{87} \\ a_{98} \end{bmatrix} &
\begin{bmatrix} a_{12} \\ a_{23} \\ * \\ a_{45} \\ a_{56} \\ * \\ a_{78} \\ a_{89} \\ * \end{bmatrix} &
\begin{bmatrix} a_{14} \\ a_{25} \\ a_{36} \\ a_{47} \\ a_{58} \\ a_{69} \\ * \\ * \\ * \end{bmatrix}
\end{array}
\end{array}
\qquad
\begin{array}{c}
R\_position \\
\begin{array}{ccccc}
 & Top & Left & Right & Down \\
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{matrix} &
\begin{bmatrix} * \\ * \\ * \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix} &
\begin{bmatrix} * \\ 1 \\ 2 \\ * \\ 4 \\ 5 \\ * \\ 7 \\ 8 \end{bmatrix} &
\begin{bmatrix} 2 \\ 3 \\ * \\ 5 \\ 6 \\ * \\ 8 \\ 9 \\ * \end{bmatrix} &
\begin{bmatrix} 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ * \\ * \\ * \end{bmatrix}
\end{array}
\end{array}
$$

Figure 3 CSG: Compressed sparse expression for grid structure

Direction		[Value, Index]
Horizontal	upper	$\left[a_{iN+j+k(N\times N)},\ (i-1)N+j+k(N\times N)\right]$
	lower	$\left[a_{iN+j+k(N\times N)},\ (i+1)N+j+k(N\times N)\right]$
	left	$\left[a_{iN+j+k(N\times N)},\ iN+j-1+k(N\times N)\right]$
	right	$\left[a_{iN+j+k(N\times N)},\ iN+j+1+k(N\times N)\right]$
Vertical	upper layer	$\left[a_{iN+j+k(N\times N)},\ iN+j+(k+1)(N\times N)\right]$
	lower layer	$\left[a_{iN+j+k(N\times N)},\ iN+j+(k-1)(N\times N)\right]$

Figure 4 Stored data of CSCG

EVALUATION OF ACCURACY

In order to check the accuracy of a simulation, comparison with actual measurement is performed. A 40-ohm Nichrome wire, polycarbonate plates of 16 cm x 16 cm x 2-mm-thick are used for the actual measurement as shown in Figure 5. Figure 6 shows the heat photograph after 10 minutes of electric current flow. Figure 7 shows simulation result for the same condition. Figure 8 shows the simulation and measured result of the same conditions at different points A, B, shown in Figure 5. Table 3 is parameters of the materials. Next, accuracy comparison with Elmer [10, 11], which is finite element method, is performed. The heat transfer coefficient $0.027\,[\mathrm{W/(m^2 \cdot K)}]$, the whole calorific capacity 157.5 [K/W], and a time step are set to 0.12928 [s]. Physical size, generation of heat, and circumference temperature are the same as Table 3. Resolutions of both simulations are set same size, 256x256x9. The simulation results of 15 minutes after are Figures 9 and 10. Both pictures show very similar heat distributions. Comparison of temperature is Figure 11 and almost same accurate result is obtained. The computation time by proposed method, Elmer is 594[s], 10083[s], respectively. Thus, 17 times speed up is acquired.

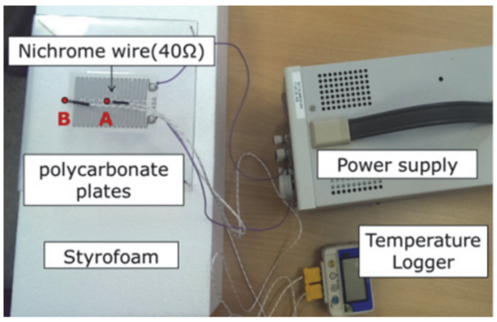

Figure 5 Actual measurement

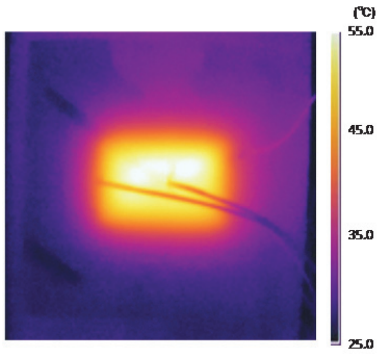

Figure 6 Heat photograph of actual measurement after 10 minutes heating

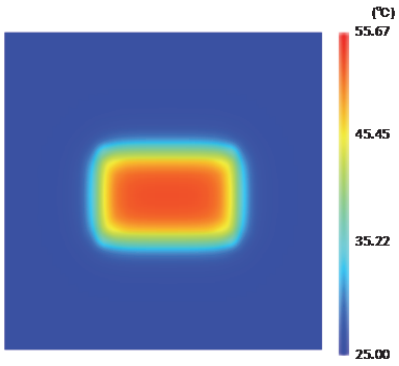

Figure 7 Simulation result after 10 minutes heating by proposed method

Table 3 Parameters of materials

Parameter	Value
Physical size of entire sheet (L×W×H) [cm]	16.0×16.0×0.2
Thermal conductivity of polycarbonate [W/(m・K)]	0.235
Heat Capacity [J/K]	161.28
Ambient Temperature[K]	298.0

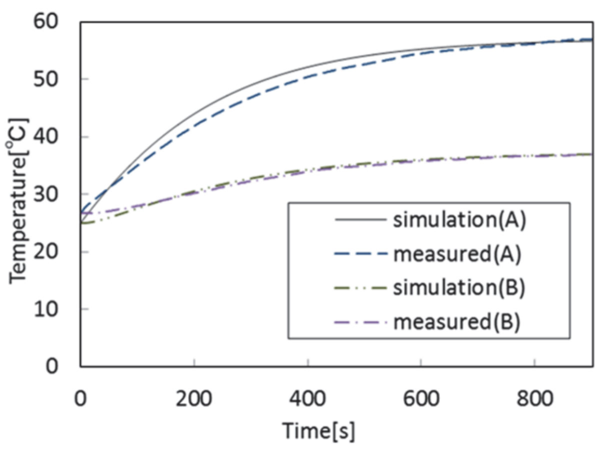

Figure 8 Temperature comparisons of simulation and measurement at points A and B in Figure 5

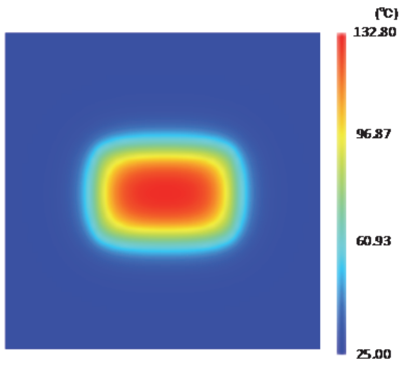

Figure 9 Simulation result after 15 minutes heating by proposed method

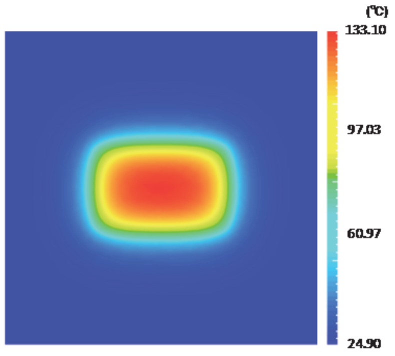

Figure 10 Simulation result after 15 minutes heating by Elmer [10]

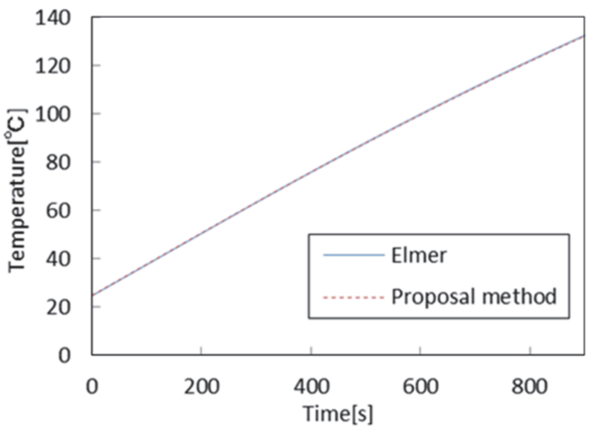

Figure 11 Temperature comparisons of Elmer [10] and proposed method

EXPERIMENTAL RESULTS

Figure 12 shows the simulation result of 10 seconds heating. The thermal circuit size is set to 64x64x10. The layer number is shown at the bottom of each picture. Table 4 shows the computation time for different resolutions from 16x16x10 to 256x256x10. The graph of the speed improvement ratio by using GPU is shown in Figure 13. By execution by 32 bit OS, when the size is 256x256x10, 22.8 times of speeding up has been acquired.

CONCLUSION

In this paper a multi-physical thermal analysis procedure has been devised, which is implemented on GPU (NVIDIA GeForce GTX 580 --- 512 Core, 1.54 GHz, double precision 198GFLOPS) by using a heat transfer coefficient of air.

As a test example, a set of LSI package and surrounding air is modelled by a 3D grid network of size 256x256x10, for which the computation time spent by GPU on the thermal analysis for 10 seconds is 1124 seconds, which is 22.8 times faster than that attained by CPU (Core i7 by 64 bit OS). Moreover, as for the simulation accuracy, a distinctive difference between the Elmer method and the finite element method has been observed only in the temperature by 0.1 degree or less. Furthermore, Computation by proposed method is about 17 times more efficient than Elmer.

Thus, although the proposed 3D thermal analysis procedure is constructed of an approximative approach, not only its efficiency but also its accuracy is so high. The macro-model devised for thermal analysis can be practically used. Future work needs verification for many cases of convection model to the air.

REFERENCES

[1] Lin L., Fukui M., and Tsukiyama S., "A GPGPU implementation of parallel backward Euler algorithm for power grid circuit simulation," *Proceedings of 11th IEEE International NEWCAS Conference*, 2013, pp. 1–4,.

[2] Omura T., Lin L., Meng L., and Fukui M., "A two-dimensional thermal analysis system for LSI packages by GPGPU," *Proceedings of International Technical Conference on Circuits/Systems, Computers and Communications*, 2014, pp. 380–383.

[3] Christophe F., Dinh D.V., Guy F., Mathieu M., and Charles D., "Thermal modeling of a cylindrical LiFePO4/graphite lithium-ion battery," *Journal of Power Sources*, Vol. 195, 2010, pp. 2961–2968.

[4] Seyama Y., Shimozono T., Nisiyama K., Nakamura H., and Sonoda T., "Development of large-scale lithium ion batteries "LIM series" for industrial applications," *GS NEWS TECH. REP*, Vol.62, 2003, pp.73-78.

[5] Onda K., Ohshima T., Nakayama M., Fukuda K., and Araki T., " Thermal behavior of small lithium-ion battery during rapid charge and discharge cycles," *Journal of Power Sources*, Vol. 158, 2006, pp.535–542.

[6] Chris M., Ben L., Derrick B., and Naoki O., "Advanced electro-thermal modeling of Lithium-ion battery system for hybrid electric vehicle applications," *Proceedings of IEEE Vehicle Power and Propulsion Conference*, 2007, pp. 107-111.

[7] Mousavi M., Hoque S., Rahnamayan S., Dincer I., and Naterer G. F., "Optimal design of an air-cooling system for a Li-ion battery pack in electric vehicles with a genetic algorithm," *Proceedings of Evolutionary Computation Cong.*, 2011, pp. 1848- 1855.

[8] Pillage L., Rohrer R. A., and Visweswariah C., "Electronic circuit & system simulation methods," McGraw-Hill, 1998.

[9] Bell N. and Garland M., "Efficient sparse matrix-vector multiplication on CUDA," *NVIDIA Technical Report*, 2008.

[10] *Elmer—Open Source Finite Element Software for Multiphysical Problems*. [Online]. Available: https://www.csc.fi/web/elmer, accessed 18 Feb. 2016.

[11] Keranen J., Pippuri J., Malinen M., Ruokolainen J., Raback P., Lyly M., Tammi K., "Efficient parallel 3-D computation of electrical machines with Elmer," *IEEE Transaction on Magnetics*, Vol.51, 2015.

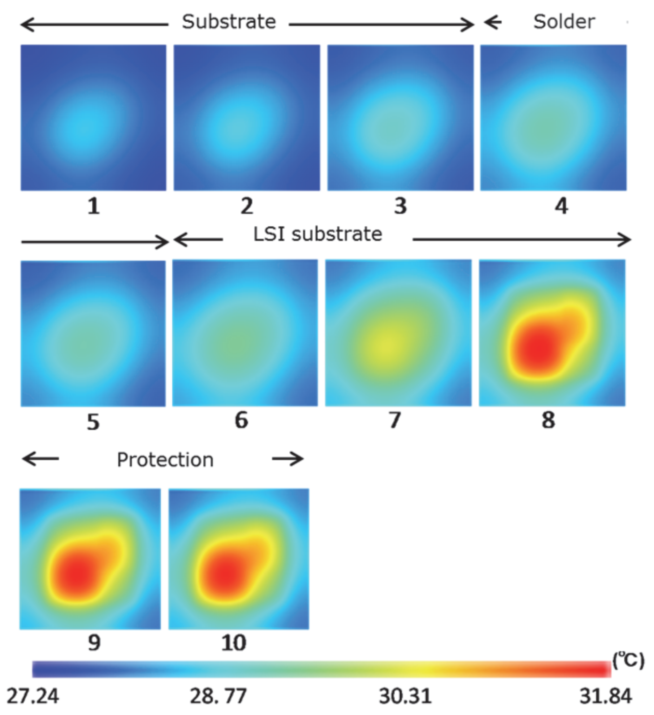

Figure 12 Simulation result for the example of Figure 1

Table 4 Computation time of simulation

Circuit size	CPU (32bit)	GPU (32bit)	CPU (64bit)	GPU (64bit)
16×16×10	7.0[s]	7.1[s]	4.1[s]	7.7[s]
32×32×10	23.7[s]	9.2[s]	12.3[s]	8.7[s]
64×64×10	100.8[s]	14.3[s]	47.8[s]	18.4[s]
128×128×10	1656.9[s]	113.2[s]	749.9[s]	114.2[s]
256×256×10	26739.5[s]	1175.2[s]	15514.7[s]	1124.1[s]

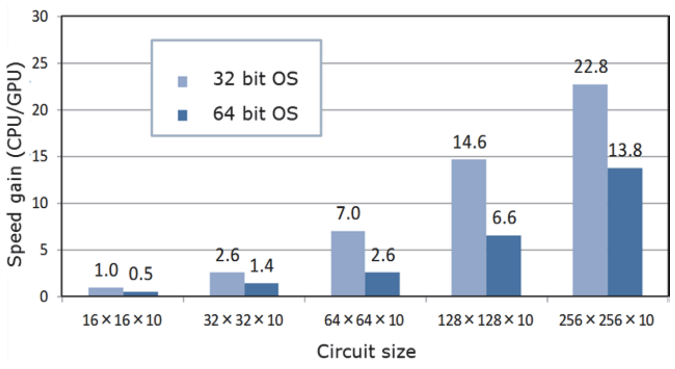

Figure 13 Simulation speed gain by GPU against CPU