Thermal-ADI: a Linear-Time Chip-Level Dynamic Thermal Simulation Algorithm Based on Alternating-Direction-Implicit (ADI) Method

Ting-Yuan Wang Charlie Chung-Ping Chen
Department of Electrical and Computer Engineering
University of Wisconsin at Madison
Madison, WI 53706
wangt@cae.wisc.edu chen@engr.wisc.edu

ABSTRACT

Due to the dramatic increase of clock frequency and integration density, power density and on-chip temperature in high-end VLSI circuits rise significantly. To ensure the timing correctness and the reliability of high-end VLSI design, efficient and accurate chip-level transient thermal simulations are of crucial importance.

In this paper, we develop and present an efficient transient thermal simulation algorithm based on the alternating-direction-implicit method. Our algorithm, Thermal-ADI, not only has *a linear runtime and memory requirement*, but also is *unconditionally stable* which ensures that time-step is not limited by any stability requirement. Extensive experimental results show that our algorithm is not only orders of magnitude faster than the traditional thermal simulation algorithms but also highly accurate, and efficient in memory usage.

1. INTRODUCTION

Due to the relentless push for high speed, high performance, and high component density, the power density and the on-chip temperature in the high-end VLSI circuits rise significantly. The 1999 International Technology Roadmap for Semiconductors (ITRS) shows that the maximum power, number of metal layers, and the wire current density will significantly increase for the future high-performance Microprocessor Unit (MPU). This trend shows the importance of thermal issues on VLSI design.

High temperature not only causes timing failures for both transistors and interconnects but also degrades chip reliability. For example, electromigration (EM) effect for the interconnects is exponentially proportional to the temperature, not to mention electrostatic discharge(ESD) or other effects. For the next generation process, the low dielectric constant (low-k) materials will exaggerate the thermal effects because of their low thermal conductivity. To effectively analyze the thermal distribution and locate the hot spots, chip-level thermal analyses are of crucial importance. Furthermore, for the finite thermal conductivity of the complicated packaging problem, the uniform heat distribution does not guarantee uniform temperature profile. Thus, it is valuable to know the temperature profile and hot spots, not only for the steady state but also the transient state.

Several approaches have been proposed to perform thermal analysis [3]-[9]. However, due to the complexity of solving the large scale matrix, the existing direct matrix-solving algorithms suffer superlinear runtime and memory consumption for large scale problems. In this paper, we propose an algorithm by using the alternating direction implicit (ADI) method to simulate the temperature profile. Our method, Thermal-ADI, is not only unconditionally stable but also has a linear runtime and a linear memory requirement. The experimental results show orders of runtime improvement over the traditional algorithms.

The remainder of the paper is organized as follows. The thermal simulation physics is discussed in Section 2. Section 3 presents an overview of the numerical formulation of the heat transfer. Section 4 deals with the thermal simulation by ADI method. The implementation and experimental results are presented in Section 5, followed by the conclusion in Section 6.

2. THERMAL SIMULATION PHYSICS

As shown in Figure (1), the temperature distribution in a chip is governed by the following partial differential equation of heat conduction [1] from the energy conservation law [12]:

$$\rho c_p \frac{\partial T(\vec{r},t)}{\partial t} = \bigtriangledown \cdot [\kappa(\vec{r},T) \bigtriangledown T(\vec{r},t)] + g(\vec{r},t) \qquad (1)$$

subject to the following thermal boundary conditions

$$\kappa(\vec{r},T)\frac{\partial T(\vec{r},t)}{\partial n_i} + h_i T(\vec{r},t) = f_i(\vec{r}_{s_i},t), \qquad (2)$$

where T is the temperature, ρ is the density of the material, c_p is the specific heat, κ is the thermal conductivity, g is the heat energy generation rate, h_i is the heat transfer coefficient on the boundary surface of the chip, $f_i(\vec{r}_s,t)$ is any function on the boundary surface, and $\partial/\partial n_i$ is the differentiation along the outward direction normal to the boundary surface s_i.

ISPD'01, April 1-4, 2001, Sonoma, California, USA.

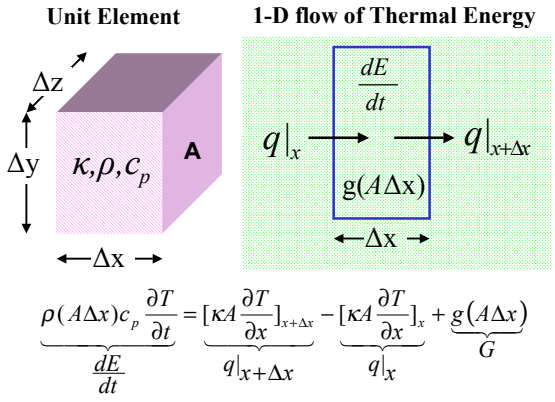

Figure 1: Energy Conservation and Heat Conduction Equation. The one-dimensional system is shown on the right-hand side, and the heat equation derived by energy conservation is shown on the bottom. $q|_x = -[\kappa A \frac{\partial T}{\partial x}]_x$ **is the conduction heat into the system.** $q|_{x+\Delta x} = -[\kappa A \frac{\partial T}{\partial x}]_{x+\Delta x}$ **is the conduction heat out of the system.** $\frac{dE}{dt}$ **is the change rate of energy stored inside the system.** G **is the rate of energy generation inside the system.**

In general, thermal conductivity $\kappa(\vec{r}, T)$ is dependent on the position and the temperature. The heat generation rate $g(\vec{r}, t)$ comes from each gate and power/ground/clock interconnect. The energy generation rate in the interconnects can be expressed as $J^2_{RMS} \cdot R_\rho$ [7], where J_{RMS} is the RMS current density, and R_ρ is the temperature-dependent resistivity of the interconnects.

The time constant of the heat conduction is much larger than the clock cycles. Thus we are able to use the heat generation rate defined above to simulate the transient temperature profile. The full chip temperature profile is supposed to become stable when the thermal steady state is reached after enough time. There are three different situations for the boundary conditions.

- *Specified Temperature* The temperature is prescribed along the boundary surface, i.e.,
$$T(\vec{r}, t) = f_i(\vec{r}_s, t).$$
This is the boundary condition of the first kind.
- *Heat Flux* The specified heat flux along the boundary surface can be expressed as:
$$\kappa(\vec{r}, T)\frac{\partial T(\vec{r}, t)}{\partial n_i} = q_{s_i},$$
where q_{s_i} is the heat flux on the boundary surface s_i. This is the second kind of boundary condition. For the adiabatic boundary condition, we have $q_{s_i} = 0$.
- *Convection Boundary Condition* The heat transfers from the considered boundary surface s_i to the ambient by convection can be expressed as:
$$\kappa(\vec{r}, T)\frac{\partial T(\vec{r}, t)}{\partial n_i} + h_i T(\vec{r}, t) = h_i T_\infty,$$
where T_∞ is the ambient temperature, and h_i is the equivalent heat transfer coefficient. This is the boundary condition of the third kind.

3. A FINITE-DIFFERENCE FORMULATION OF THE HEAT CONDUCTION

The two-dimensional heat conduction equation can be rewritten from Equation (1) as:

$$\frac{\partial T(x,y,t)}{\partial t} = \alpha\frac{\partial^2 T(x,y,t)}{\partial x^2} + \alpha\frac{\partial^2 T(x,y,t)}{\partial y^2} + \frac{1}{\rho c_p}g(x,y,t), \quad (3)$$

where $\alpha = \frac{\kappa}{\rho c_p}$. This equation is a second-order parabolic partial differential equation. The first step to establish a finite-difference method of the partial differential equation is to discretize the continuous space domain into a mesh with a finite number of grid points. As illustrated in Figure (2), the temperature $T(x, y, t)$ at each point in the chip will be

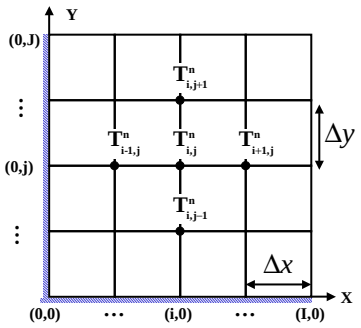

Figure 2: Finite Difference Mesh on the x-y plane

replaced by $T(i\Delta x, j\Delta y, n\Delta t)$ which will be denoted as $T^n_{i,j}$ for the rest of the paper. The first-order partial derivative of T with respect to x can be converted to the forward finite-difference representation as:

$$\frac{\partial T}{\partial x}\Big|^n_{i,j} = \frac{T^n_{i+1,j} - T^n_{i,j}}{\Delta x} + O(\Delta x) \approx \frac{T^n_{i+1,j} - T^n_{i,j}}{\Delta x},$$

where the truncation error is $O(\Delta x)$. Similarly, the central finite-difference representation of the second-order partial derivative of T with respect to x can be expressed as:

$$\begin{aligned}\frac{\partial^2 T}{\partial^2 x}\Big|^n_{i,j} &= \frac{T^n_{i+1,j} - 2T^n_{i,j} + T^n_{i-1,j}}{(\Delta x)^2} + O(\Delta x)^2 \\ &\approx \frac{T^n_{i+1,j} - 2T^n_{i,j} + T^n_{i-1,j}}{(\Delta x)^2} = \frac{\delta^2_x T^n}{(\Delta x)^2}\end{aligned}$$

where the truncation error is $O((\Delta x)^2)$, and

$$\delta^2_x T = T_{i-1,j} - 2T_{i,j} + T_{i+1,j}.$$

The next step is to consider the time marching problem for the finite difference equations. Since Equation (1) comes from the energy conservation, it can be explained physically as the increasing rate of the stored energy in a control unit volume equals to the net rate of energy transferring into the volume. Hence the forward-difference with time on the left-hand side of Equation (1) is the energy stored from time step n to $n+1$ in the control unit volume. Three time marching methods are considered.

Simple Explicit Method

Applying the explicit update on the right-hand side of Equation (3) at time step n, we get:

$$\frac{T^{n+1} - T^n}{\Delta t} = \alpha[\frac{\delta^2_x T^n}{(\Delta x)^2} + \frac{\delta^2_y T^n}{(\Delta y)^2}] + \frac{1}{\rho c_p}g. \quad (4)$$

Note that there is only one unknown in each equation with respect to point (i, j). Therefore it is not necessary to solve any equations. The truncation error is $O[(\Delta t), (\Delta x)^2, (\Delta y)^2]$. The stability constraint γ can be described as follows:

$$\gamma = \alpha \Delta t (\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}) \leq \frac{1}{2}. \quad (5)$$

It restricts the size of the time step, Δt, for the given space increment, Δx and Δy.

Simple Implicit Method

Applying the simple implicit update on the right hand side of Equation (3) at time step n, we get:

$$\frac{T^{n+1} - T^n}{\Delta t} = \alpha [\ \frac{\delta_x^2 T^{n+1}}{(\Delta x)^2} + \frac{\delta_y^2 T^{n+1}}{(\Delta y)^2}\ ] + \frac{1}{\rho c_p} g. \quad (6)$$

There are five unknowns in each equation at point (i, j) for the time step $n+1$. The truncation error is also $O[(\Delta t), (\Delta x)^2, (\Delta y)^2]$. However, the simple implicit method is unconditionally stable.

Crank-Nicolson Method

Crank and Nicolson developed another method by taking the average of simple explicit and implicit on the right hand side of Equation (3), they obtained [2]:

$$\frac{T^{n+1} - T^n}{\Delta t} = \alpha [\ \frac{\delta_x^2 T^{n+1} + \delta_x^2 T^n}{2(\Delta x)^2} + \frac{\delta_y^2 T^{n+1} + \delta_y^2 T^n}{2(\Delta y)^2}\ ] + \frac{g}{\rho c_p}. \quad (7)$$

There are five unknowns in each equation at point (i, j) for the time step $n + 1$. However, the Crank-Nicolson method has the best accuracy with the truncation error $O[(\Delta t)^2, (\Delta x)^2, (\Delta y)^2]$. It is also unconditionally stable.

The difference equation of the heat conduction Equation by Crank-Nicolson method can be expressed as follows:

$$\begin{aligned} &-r_x T^{n+1}_{i-1,j} - r_y T^{n+1}_{i,j-1} + 2(1 + r_x + r_y) T^{n+1}_{i,j} - \\ &r_x T^{n+1}_{i+1,j} - r_y T^{n+1}_{i,j+1} = r_x T^n_{i-1,j} + r_y T^n_{i,j-1} + \\ &2(1 - r_x - r_y) T^n_{i,j} + r_x T^n_{i+1,j} + r_y T^n_{i,j+1} + \frac{2\Delta t}{\rho c_p} g_{i,j} \\ &i = 1, 2, ..., I - 1 \\ &j = 1, 2, ..., J - 1, \end{aligned} \quad (8)$$

where $r_x = \frac{\alpha \Delta t}{(\Delta x)^2}$ and $r_y = \frac{\alpha \Delta t}{(\Delta y)^2}$.

There are some difficulties for solving the above difference equations. For a two-dimensional mesh with size $(I + 1) \times (J + 1)$, the number of variables needed for this system is $(I + 1)(J + 1)$, which requires a matrix A with size $(I + 1)(J + 1) \times (I + 1)(J + 1)$ to store the coefficients. To solve the equations $Ax = b$ by LU decomposition or Cholesky decomposition, the runtime and memory requirement are superlinear with sparse matrix techniques.

4. ALTERNATING DIRECTION IMPLICIT METHODS

Since the Crank-Nicolson method is computationally intensive, Peaceman and Rachford [10], and Douglas and Gunn [11] developed the alternating direction implicit (ADI) methods. The ADI method is a process to reduce the two-dimensional or three-dimensional problems to a succession of two or three one-dimensional problems.

4.1 Peaceman-Rachford Algorithm

Equation (7) can be rearranged as follows:

$$\begin{aligned} &(1 - \frac{r_x}{2}\delta_x^2)(1 - \frac{r_y}{2}\delta_y^2)\ T^{n+1} = \\ &(1 + \frac{r_x}{2}\delta_x^2)(1 + \frac{r_y}{2}\delta_y^2)\ T^n + \frac{\Delta t}{\rho c_p} g. \end{aligned} \quad (9)$$

Peaceman and Rachford propose to solve the above equation by two steps instead of one step.

- **Step I**

$$(1 - \frac{r_x}{2}\delta_x^2) T^{n+\frac{1}{2}} = (1 + \frac{r_y}{2}\delta_y^2) T^n + \frac{\Delta t}{2\rho c_p} g \quad (10)$$

- **Step II**

$$(1 - \frac{r_y}{2}\delta_y^2) T^{n+1} = (1 + \frac{r_x}{2}\delta_x^2) T^{n+\frac{1}{2}} + \frac{\Delta t}{2\rho c_p} g \quad (11)$$

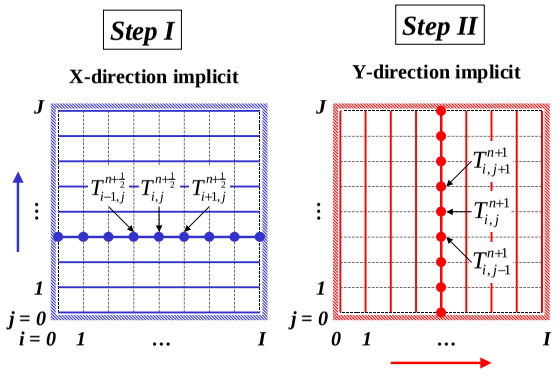

Figure 3: For *Step I*, the x-direction is implicit and the y-direction is explicit. There are $I - 1$ equations corresponding to $I - 1$ points for each iteration j. Each point (i, j) at iteration j is related to unknowns, point $(i - 1, j)$ and pont $(i + 1, j)$, which introduces a tridiagonal matrix for each iteration. *Step II* has a similar process to *Step I*.

This algorithm separates the time step from n to $n + 1$ into two sub time steps: from n to $n + \frac{1}{2}$ and from $n + \frac{1}{2}$ to $n + 1$. As shown in Figure (3), in *Step I*, it applies the implicit update in the x-direction and the explicit update in the y-direction. For every j, there are $I - 1$ equations for the corresponding (i, j) point. Since each point (i, j) is related to two points $(i - 1, j)$ and $(i + 1, j)$, the coefficient matrix is in tridiagonal form which can be easily solved by Thomas Algorithm [2] with $O(n)$ time. In *Step II*, it applies the implicit update in the y-direction and the explicit update in the x-direction similar as *Step I*.

The detail difference equations in *Step I* for each row from Equation (10) can be derived as follows:

$$\begin{aligned} &-r_x T^{n+\frac{1}{2}}_{i-1,j} + (2 + 2r_x) T^{n+\frac{1}{2}}_{i,j} - r_x T^{n+\frac{1}{2}}_{i+1,j} = \\ &r_y T^n_{i,j-1} + (2 - 2r_y) T^n_{i,j} + r_y T^n_{i,j+1} + \frac{\Delta t}{\rho c_p} g_{i,j} \\ &i = 1, 2, ..., I - 1 \\ &j = 1, 2, ..., J - 1. \end{aligned} \quad (12)$$

Similarly, the difference equations in *Step II* for each column can be expressed as:

$$-r_y T_{i,j-1}^{n+1} + (2+2r_y)T_{i,j}^{n+1} - r_y T_{i,j+1}^{n+1} = r_x T_{i-1,j}^{n+\frac{1}{2}} + (2-2r_x)T_{i,j}^{n+\frac{1}{2}} + r_x T_{i+1,j}^{n+\frac{1}{2}} + \frac{\Delta t}{\rho c_p} g_{i,j}$$
$$i = 1, 2, ..., I-1$$
$$j = 1, 2, ..., J-1. \quad (13)$$

Now, we can reduce the two-dimensional problems into a succession of two one-dimensional problems by the Peaceman-Rachford ADI algorithm. For every one-dimensional problem, the tridiagonal matrix can be solved by the Thomas Algorithm in $O(n)$ time. Given an $I \times J$ mesh, the runtime and memory requirement for each time step is $O(I \times J)$ and hence is linear with respect to the problem size.

4.2 Douglas-Gunn Algorithm

Douglas and Gunn developed another general algorithm for the ADI schemes that are unconditionally stable and retain second order accuracy. As shown in [11], equation (7) can be rewritten as:

$$\frac{T^{n+1} - T^n}{\Delta t} = \frac{\alpha \delta_x^2}{2(\Delta x)^2}(T^{n+1} + T^n) + \frac{\alpha \delta_y^2}{2(\Delta y)^2}(T^{n+1} + T^n) + \frac{1}{\rho c_p} g. \quad (14)$$

Following the ADI scheme [11], Equation (14) can be solved by two sub time-steps:

- **Step I**

$$T^{n+\frac{1}{2}} - T^n = \frac{r_x \delta_x^2}{2}(T^{n+\frac{1}{2}} + T^n) + r_y \delta_y^2 T^n + \frac{\Delta t}{\rho c_p} g \quad (15)$$

- **Step II**

$$T^{n+1} - T^n = \frac{r_x \delta_x^2}{2}(T^{n+\frac{1}{2}} + T^n) + \frac{r_y \delta_y^2}{2}(T^{n+1} + T^n) + \frac{\Delta t}{\rho c_p} g. \quad (16)$$

Since the Douglas-Gunn Algorithm uses a similar ADI scheme to the Peaceman-Rachford algorithm, we will ignore the detailed discussion about the time marching scheme. The detail difference equations in *Step I* for each row can be derived as follows:

$$-r_x T_{i-1,j}^{n+\frac{1}{2}} + 2(1+r_x)T_{i,j}^{n+\frac{1}{2}} - r_x T_{i+1,j}^{n+\frac{1}{2}} = r_x T_{i-1,j}^{n} + 2r_y T_{i,j-1}^{n} + 2(1-r_x-2r_y)T_{i,j}^{n} + r_x T_{i+1,j}^{n} + 2r_y T_{i,j+1}^{n} + \frac{2\Delta t}{\rho c_p} g_{i,j}$$
$$i = 1, 2, ..., I-1$$
$$j = 1, 2, ..., J-1. \quad (17)$$

The difference equations in *Step II* for each column are as follows:

$$-r_y T_{i,j-1}^{n+1} + 2(1+r_y)T_{i,j}^{n+1} - r_y T_{i,j+1}^{n+1} = (r_x T_{i-1,j}^{n+\frac{1}{2}} - 2r_x T_{i,j}^{n+\frac{1}{2}} + r_x T_{i+1,j}^{n+\frac{1}{2}}) + r_x T_{i-1,j}^{n} + r_y T_{i,j-1}^{n} + 2(1-r_x-r_y)T_{i,j}^{n} + r_x T_{i+1,j}^{n} + r_y T_{i,j+1}^{n} + \frac{2\Delta t}{\rho c_p} g_{i,j}$$
$$i = 1, 2, ..., I-1$$
$$j = 1, 2, ..., J-1. \quad (18)$$

Similarly, the above tridiagonal equation sets can be solved by the Thomas Algorithm in linear time and hence the Douglas-Gunn Algorithm has a linear runtime and memory requirement which is the same as the Peaceman-Rachford algorithm.

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The flowchart of the transient thermal simulation by the ADI algorithm is shown in Figure (4). The main features of implementing the algorithm is listed below:

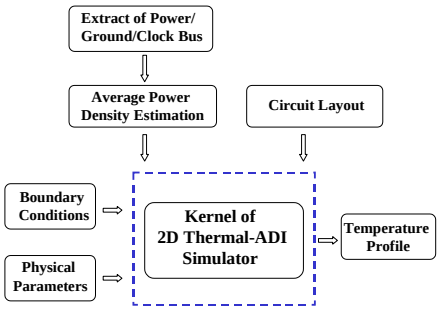

Figure 4: The flow chart for the FDTD ADI thermal simulation

1. The core part of Thermal-ADI is a fast 2D transient thermal simulator. The first step for the thermal simulation is to partition the chip into a mesh, and choose the size of Δt, Δx and Δy. The second step is to extract the layout of transistors and metal wires from circuits. There are three situations of layout extraction as shown in Figure (5). The simulator reads the given circuit descriptions, the coordinates of the gates, metal wires, and power/ground/clock interconnects for thermal simulation, and then calculates the information at each node on the mesh according to the physical parameters at that point. Finally, it iteratively calculates the temperature at each node by the ADI Algorithm and outputs the temperature profile.

2. We use band structure to implement the Crank-Nicolson method in order to reduce the runtime and the memory usage. For the chip with mesh points $(I+1) \times (J+1)$, we need a matrix A with size $(I+1)(J+1) \times (I+2)$ rather than $(I+1)(J+1) \times (I+1)(J+1)$.

We implemented the Thermal-ADI algorithm with C++ language, and executed it on an Alpha workstation with Dual SLOTB 667 MHz Alpha 21264 processors.

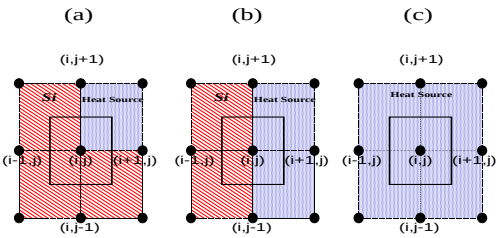

Figure 5: Three different situations of the transistors and metal wires layout extraction.

# Nodes	CN (seconds)	PR (seconds)	DG (seconds)
2500	141	2	2
4900	502	4	4
10000	1665	7	6
40000	25303	25	28
90000	133433	64	69
160000	576750	121	140
250000	2520000	223	252
360000	-	297	336
490000	-	405	465
640000	-	561	672
810000	-	711	845
1000000	-	942	1182
4000000	-	3772	5047
9000000	-	9113	11729
25000000	-	28235	37122
49000000	-	60375	85625
64000000	-	83500	113830
81000000	-	106525	143414
100000000	-	134708	-

Table 1: Runtime of Crank-Nicolson method, Peaceman-Rachford Algorithm, and Douglas-Gunn Algorithm

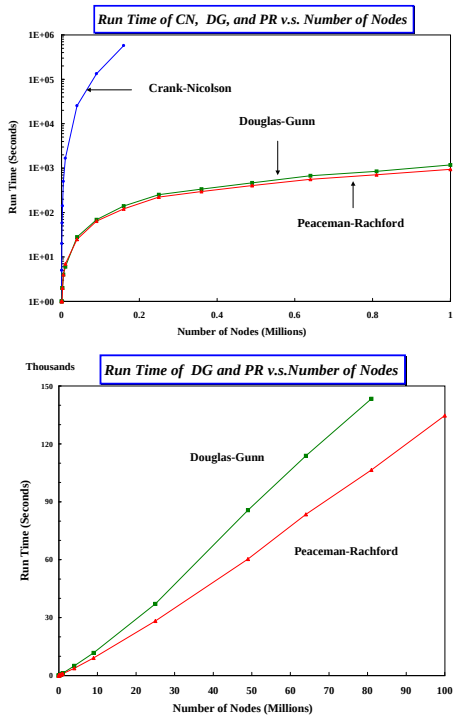

Figure 6: Runtime comparisons of the Crank-Nicolson, Douglas-Gunn, and Peaceman-Rachford Algorithms

The runtime comparison of the Crank-Nicolson method, Peaceman-Rachford algorithm, and Douglas-Gunn algorithm is shown in Figure (6) and Table 1.

We executed all these methods with 500 time steps where each time step is 0.75 ns. As can be seen in Figure (6), the runtime of the Douglas-Gunn and Peaceman-Rachford algorithm is linearly proportional to the number of nodes as shown with size up to 10^8. However, the runtime of Crank-Nicolson method increases dramatically.

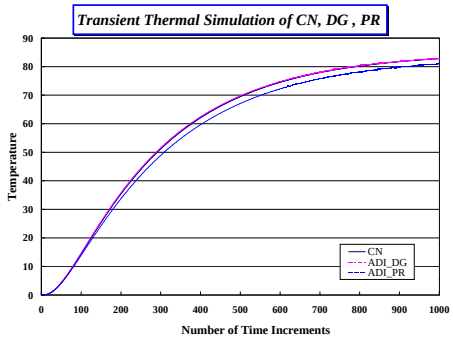

Figure 7: The simulation results of the Crank-Nicolson method, the Douglas-Gunn, and the Peaceman-Rachford Algorithms at a point in the chip are shown above.

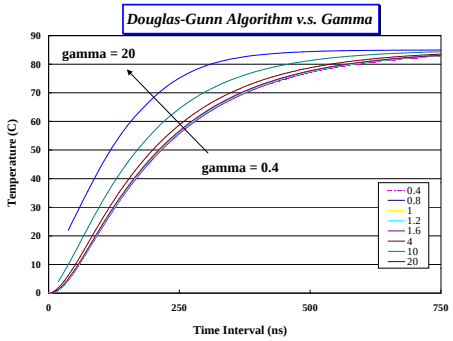

Figure 8: Douglas-Gunn algorithm is unconditionally stable. Since the γ values exceed the criteria value 0.5, the results are still stable rather than oscillating.

The temperature of transient thermal simulation at a random chosen point is shown in Figure (7). This is done by a 100×100 mesh with time interval $\Delta t = 0.75ns$ and the time steps 1000. The error of the Douglas-Gunn algorithm compared to the Crank-Nicolson method is less than 0.1%. However, the error of the Peaceman-Rachford algorithm compared to the Crank-Nicolson method is 2.25%. The transient temperature curve of the Douglas-Gunn Algorithm is very close to the curve of the Crank-Nicolson method.

The stability constraint γ in Equation (5) was varied from 0.4 to 20 as shown in Figure (8), where $\frac{1}{2}$ is the stability limit. As illusrated in Figure (8), the Douglas-Gunn algorithm is unconditionally stable. Figure (8) also shows that the larger Δt of the Douglas-Gunn algorithm turns out to have the bigger deviation away from the curve of the Crank-Nicolson method.

The error of the Douglas-Gunn Algorithm is shown in Figure (9) with different γ at time step 1000. Here we only change the value of Δt in γ while keeping the other factors fixed. The error increases linearly with respect to the γ value.

As illustrated in Figure (10) and Table 2, the memory usages of the Douglas-Gunn and Peaceman-Rachford Algorithms are linearly proportional to the number of nodes up to 10^8. However, the memory usage of Crank-Nicolson Method increases dramatically.

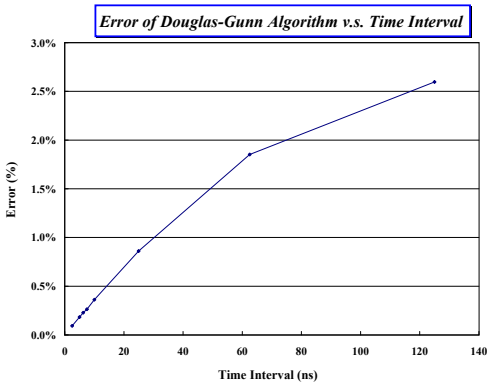

Figure 9: The errors of the Douglas-Gunn Algorithm compared to the Crank-Nicolson Method at time step 1000 as shown in Figure 8 are proportional to Δt.

Table 2: Memory Usages of the Crank-Nicolson method, the Peaceman-Rachford, and Douglas-Gunn Algorithms

# Nodes	CN (Mb)	PR (Mb)	DG (Mb)
4900	7.048	1.104	1.144
10000	17	1.192	1.272
40000	128	1.68	1.992
90000	424	2.48	3.184
160000	998	3.592	4.848
250000	1900	5.016	6.976
360000	-	6.752	9.68
490000	-	8.8	12
640000	-	10	15
810000	-	13	19
1000000	-	16	24
4000000	-	62	93
9000000	-	138	207
25000000	-	383	574
49000000	-	749	1100
64000000	-	978	1460
81000000	-	1200	1800
100000000	-	1500	-

6. CONCLUSIONS

An efficient Thermal-ADI algorithm for transient thermal simulation has been developed. The unconditional stability and linear runtime and memory requirements have been demonstrated. The numerical simulation also shows that the Thermal-ADI Algorithm not only speeds up the runtime orders of magnitude over the Crank-Nicolson method but also reduces the memory usages. The error of the Douglas-Gunn Algorithm can be reduced to 0.1 % with a suitable choice of Δt and time step.

7. REFERENCES

[1] M. Necati Özisik, "Boundary Value Problems of Heat Conduction," *Dover Publications Inc.* 1968.

[2] M. Necati Özisik, "Finite Difference Methods in Heat Transfer," *CRC Press* 1994.

[3] Yi-Kan Cheng, Prasun Raha, Chin-Chi Teng, Elyse Rosenbaum, and Sung-Mo Kang, "ILLIADS-T: An Electrothermal Timing Simulator for Temperature-Sensitive Reliability Diagnosis of CMOS VLSI Chips" *IEEE Trans. Computer-Aided Design* pp668-681, Vol.17, No.8, Aug. 1998.

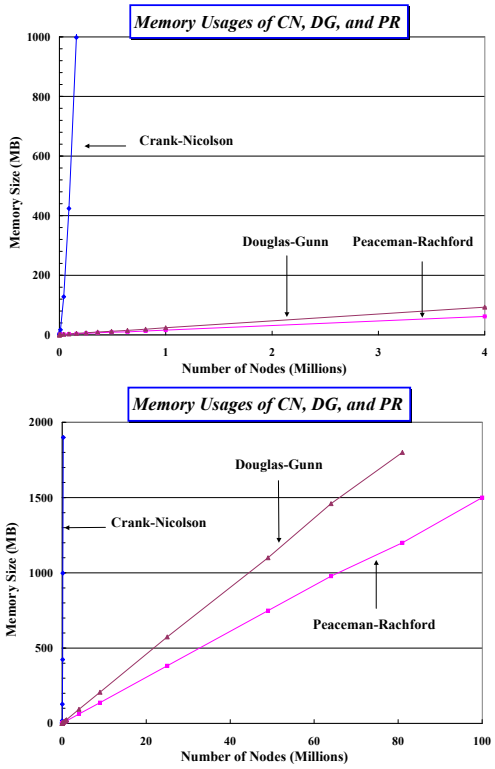

Figure 10: The Comparison of Memory Usages of the Crank-Nicolson Method, the Douglas-Gunn and Peaceman-Rachford Algorithms.

[4] Li-Pen Yuan, Chih-Chi Teng, and Sung-Mo Kang, "Statistical Estimation of Average Power Dissipation" *DAC97 34th Design Automation Conference* 1997

[5] Chih-Chi Teng, Yi-Kan Cheng, Elyse Rosenbaum, and Sung-Mo Kang, "iTEM: A Temperature-Dependent Electromigration Reliability Diagnosis Tool" *IEEE Trans. Computer-Aided Design of Integrated Circuit and Systems* pp882-893, Vol.16, No.8, Aug. 1997

[6] Ching-Han Tasi, and Sung-Mo Kang, "Cell-Level Placement for Improving Substrate Thermal Distribution," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems* pp253-266, Vol.19, No.2, Feb. 2000

[7] Danqing Chen, Erhong Li, Elyse Rosenbaum, and Sung-Mo Kang, "Interconect Thermal Modeling for Accurate Simulation of Circuit Timing and Reliability," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems* pp197-205, Vol.19, No.2, Feb. 2000

[8] Kaustav Banerjee, Amit Mehrotra, Alberto Sangiovanni-Vincentelli, and Chenming Hu "On Thermal Effects in Deep Sub-Micron VLSI Interconnects," *DAC99* pp885-891, 1999.

[9] Zhiping Yu, Dan Yergeau, and Robert W. Dutton, "Full Chip Thermal Simulation," *ISQED2000 Quality Electronic Design*

[10] Peaceman, D.W., and H. H. Rachford, Jr., "The numerical solution of parabolic and elliptic differential equations," *J. Soc. Indust. Appl. Math.,* pp28-41, 3, 1955

[11] Jim Douglas, Jr., and James E. Gunn, "A General Formulation of Alternating Direction Methods– Part I. Parabolic and Hyperbolic Problems," *Numerische Mathematik* pp428-453, 6, 1964

[12] Glen E. Myers, "Analytical Methods in Conduction Heat Transfer," *Journal of Digital Systems* Genium Publishing Corp. Chap 8, 2nd Edition, 1998.