Partition-Driven Standard Cell Thermal Placement

Guoqiang Chen
Synopsys Inc.
700 E Middlefield Road.
Mountain View, CA 94043
jerryc@synopsys.com

Sachin Sapatnekar
ECE Dept. University of Minnesota
200 Union ST SE
Minneapolis, MN 55455
sachin@ece.umn.edu

ABSTRACT

The thermal problem has been emerged as one of the key issues for next-generation IC design. In this paper, we propose a scheme to achieve better thermal distribution for partition-driven standard cell placement. The proposed heuristic uses a multigrid-like method that simplifies the thermal equation at each level of partitioning and makes it possible to incorporate temperature considerations directly as placement constraints, thus leading to better thermal distribution. Our experimental results verify the effectiveness of our scheme. We also describe an algorithm to derive a compact thermal model with a complexity of $O(mn + m^2)$, where m is the number of the mesh nodes on the substrate surface and n is the number of all internal mesh nodes.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and routing*; J.6 [**Computer-aided Engineering**]: Computer-aided design

General Terms

Algorithms, Design, Experimentation

Keywords

VLSI, standard cell, partition, placement, thermal model, temperature

1. INTRODUCTION

Since the feature sizes of VLSI chip continue to shrink and the clock frequency progressively rises from one technology node to the next, it is projected that the thermal problem will become a major bottleneck for next-generation circuit designs. High temperature can have a dramatic impact on the reliability of the chip and a large temperature gradient across the device will sometimes cause the malfunction of the device [1, 2]. Therefore it is important to consider thermal issues during the physical design process, and this paper considers the steady states of the thermal conduction. Transient effects are not considered here since the damping time of the thermal conduction for modern VLSI chip is typically of the order of milliseconds, which is several orders of magnitude larger than the operating frequency of modern circuits.

According to heat transfer theory, if we ignore the temperature dependency of thermal conductivity, the steady state thermal profile satisfies the following equation [2]:

$$k(x, y, z) \bigtriangledown^2 T(x, y, z) + g(x, y, z, t) = 0, \qquad (1)$$

where $k(x, y, z)$ is the thermal conductivity, $T(x, y, z)$ is the temperature and $g(x, y, z, t)$ is the heat source. Figure 1 shows a typical heat conduction environment for a wafer. Our interest lies in measuring, and eventually controlling, the thermal profile across the wafer surface. With the discretization shown in Figure 1 and using the compact thermal model [2], we have the following equation:

$$T = T_{fixed} + \begin{bmatrix} R_{11} & R_{12} & \dots & R_{1n} \\ R_{21} & R_{22} & \dots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ R_{m1} & R_{m2} & \dots & R_{mn} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix} \qquad (2)$$

Here T is a $m \times 1$ vector and represents m points where we monitor the temperature, $P_i (i = 1..n)$ are n heat sources, and R is the transfer thermal resistance matrix.

At the placement stage, as far as the thermal constraints are concerned, our goals are to minimize the maximal on-chip temperature gradient and obtain an even temperature distribution. The degree of freedom that is available to us for this purpose during placement is our control over the 2-dimension linear ordering of the heat sources, namely, the standard cells.

In order to minimize the maximal on-chip temperature gradient, we formulate the following problem:

> Find a permutation π of $P_i : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$
> such that $max(|T_i - T_{i,neighbors}|)$ is minimum.

This problem is an NP-hard problem [3] [4]

In our discussion so far, we have assumed that that P_i will be constant through the permutation. In today's deep sub-micron designs, however, as the cells are placed in different locations, the power consumption of the cell will indeed change. The power consumption of a cell is: $P_{cell} = \alpha f V_{dd}^2 (C_{output} + C_{interconnect})$ and the interconnect capac-

ISPD'03, April 6–9, 2003, Monterey, California, USA.

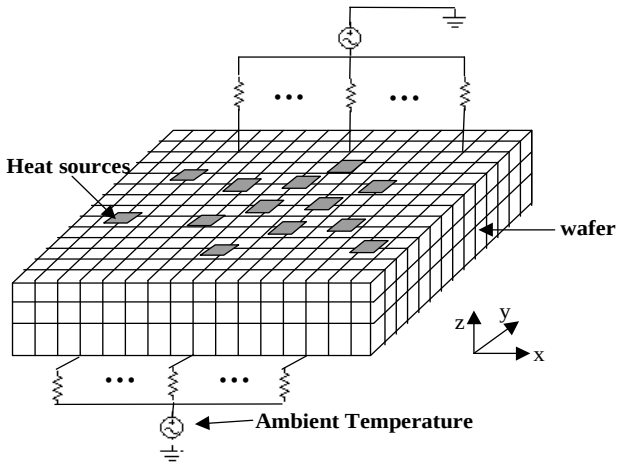

Figure 1: Heat conduction for the wafer and the 3-D mesh model

itance usually dominates the total capacitance for $0.18\mu m$ designs and below.

In the above formula, the total power is actually dissipated by both the switching transistor and the interconnecting wires. Except for long global wires, the driver resistance is typically much larger than the metal resistance, and therefore most of the power is dissipated in the cells, and we will ignore the part consumed by the metal wires. Even through self-heating of wires plays a very important role in the electro-migration lifetime of the metal wires [1], during the placement stage, we do not have the exact information about the location of the wires and we will ignore the self-heating of the wires at this stage. However, it is potentially possible to take this into account during a later stage of placement using the congestion information, though this is beyond the scope of our work.

Incorporating thermal considerations cannot be our only goal in placement. Other critical objectives are to minimize the total wire length and meet the timing constraints. Traditional non-thermal placement methods that consider these criteria can be divided into several classes:

Randomized methods Simulated annealing [5] is the most well-known example of this class of methods. Even through it is true that this method can reach arbitrary close to the global minimum if the cooling schedule is slow enough, it suffers from long run times for large circuits.

Analytical approaches This class includes methods such as force-driven placement and quadratic programming, and GORDIAN [6] is one example of this class. This method is used in many commercial placement tools. Combined with an iterative linear solver, this approach is fast and generates good results [7].

Partition-based methods The Kernighan-Lin method and its Fiduccia-Mathheyses implementation are well known min-cut methods that are utilized in partition-driven placement. Traditionally, partition methods are known to have a few problems: for example, that the wirelength is optimized indirectly through the optimization of the min-cut, and that the locally greedy approach may sometimes lose the global view. However, recent progress in multilevel hyper-graph partition and partition-based placement has produced very impressive results [8] [9] [10]. An advantage of this method is that it can handle complicated constraints.

Although partition-based placement is known to handle a number of complicated constraints, it is far from straightforward to see how it can handle the thermal constraints. Some of the obvious techniques, on closer analysis, do not turn out to be viable. For example, one way could be to use the equation (2) directly during placement. However the computation cost is high, and in the initial partition steps, we do not know the exact location of the cells. Another approach could be to first calculate the desired power profile from the desired temperature distribution using equation (2) and try to match the power distribution during partitioning. However, using this approach, we would lose the information about thermal coupling between nearby cells during the placement.

The standard cell thermal placement problem has been studied by several authors in the past. In [3], the thermal placement problem is modeled as a matrix synthesis problem. This approach assumes a specific thermal conductivity matrix and does not consider the effect of the interconnect capacitance on the cell power. In [2], a compact substrate thermal model was developed, and two algorithms for standard cell and macro cell style design were presented. For the standard cell style designs, the targeted power distribution is first computed from the desired temperature profile and this is then used to impose constraints during placement. However, a close approximation to the targeted power distribution does not necessarily lead to a close approximation of the targeted temperature profile as it ignores the thermal interactions between different cells. It is also difficult to come up with the initial power budget without a detailed understanding of the thermal environment. In both of the cases, the simulated annealing algorithm is used to perform the placement. However, due to its long run times for large designs, it is desirable to consider other alternative placement algorithms, and it is not clear how these approaches can be extended to other placement techniques.

In this paper, we present a scheme for performing thermally constrained partition-driven placement for standard cell designs. This scheme can also be used in the partitioning stage of the analytical methods like GORDIAN. We simplify the compact thermal model for the partition-driven placement and as a result, we can directly use the temperature in the inner placement loop as a constraint allowing for a better thermal distribution. We also describe an algorithm to derive the compact thermal model on the order of $O(mn + m^2)$, where m is the number of the mesh nodes on the substrate surface and n is the number of all other mesh nodes.

2. COMPACT THERMAL MODEL

Let us now consider the scenario as shown in Figure 2. The chip is divided into m regions and each region is small enough that we are not interested in the actual detailed thermal distribution inside these regions. We will refer to these

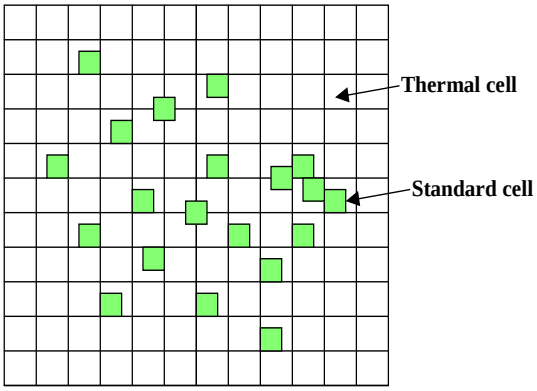

Figure 2: Compact Thermal Model

small regions as "thermal cells." If T_i is the temperature in thermal cell i and P_i is the total power dissipation in thermal cell i, we have the following equation:

$$\begin{bmatrix} G_{11} & G_{12} & \dots & G_{1m} \\ G_{21} & G_{22} & \dots & G_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{m1} & G_{m2} & \dots & G_{mm} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_m \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_m \end{bmatrix} \quad (3)$$

Considering the electrical/thermal duality, we can consider T_i as the voltage at the node i and P_i as the current source at node i.

The above conductance matrix G can be derived from the finite-difference of equation (1) as follows. This finite-difference form can be written as [2]:

$$\begin{bmatrix} G_P & G_C^T \\ G_C & G_I \end{bmatrix} \begin{bmatrix} T \\ T_I \end{bmatrix} = \begin{bmatrix} P \\ 0 \end{bmatrix} \quad (4)$$

where $T = [T_1, T_2, \cdots, T_m]^T$ and $P = [P_1, P_2, \cdots, P_m]^T$, as in equation (3), and T_I is a vector that represents the temperature at the n internal nodes.

Eliminating all the internal nodes from equation (4) and comparing it with equation (3), we have:

$$G = G_P - G_C^T G_I^{-1} G_C \quad (5)$$

The computational cost for directly calculating the right hand side of equation (5) as in [2] is $O(n^3)$ if $n \gg m$, which is not trivial. As an example, for a mesh with a 40×40 grid (i.e., $m = 1600$), in the $x - y$ direction and 6 grids in the z direction, we have $n = 8000$.

We propose a procedure that generates the matrix G by the column, observing that the i^{th} column of G is given by Ge_i, where e_i is a $m \times 1$ vector that is zero in all positions except the i^{th} position, which is 1. From equation (5), we have

$$Ge_i = G_P e_i - G_C^T G_I^{-1} G_C e_i \quad (6)$$

We calculate the right hand side of this in a two-step manner:

1. Let us consider the second term of equation (6). To find $q = G_I^{-1} G_C e_i$, we must solve $G_I q = G_C e_i$; note that $G_C e_i$ is simply the i^{th} column of G_C. Since G_I is a sparse positive definite matrix, this set of equations can be solved using the Conjugate Gradient Method with incomplete Chebyshev preconditioning [11]. Practically this type of iterative method converges within a constant number of iterations, and therefore, the complexity for this step is $O(n)$.

2. Next, we compute the entire right hand side by calculating $G_P e_i + G_C^T q$. The complexity for this step is $O(m)$, since G_C^T is a $m \times n$ sparse matrix and $G_P e_i$ is the i^{th} column of G_P.

We repeat the same process for all $e_i, 1 \leq i \leq m$ to find all of the columns of G, so that the overall complexity is $O(mn + m^2)$. If $n \gg m$, as is typical, the complexity is $O(mn)$.

3. SIMPLIFIED THERMAL MODEL FOR PARTITIONING

Now let us look at how the compact thermal model can be simplified for partitioning-driven placement. During thermal placement, our goal is to place the cells so that T is evenly distributed across the chip. Equation (3) cannot be directly used, since at each partition level, the only location information that is available is identity of the partitioning blocks that the cell belongs to. If we assume that all the cells belonging to a block are located at its center, the corresponding analysis will correspond to an incorrect thermal analysis of the partition. For example, it is easy to see that this will result in an exaggerated thermal gradient within the partition, since the temperature at the center of the partition will be much higher than that at its periphery.

For simplicity, let us consider the thermal model for a top-down bipartitioning process; this process can be easily extended to a top-down k-way partitioning process. At any one particular partition stage, a single block is being partitioned into two sub-blocks so that the cuts between the boundary of sub-blocks are minimized. For now, we are not especially concerned about the actual temperature profile in the sub-blocks since cells inside the blocks will be further partitioned later, and we can consider it at that time. However, we are concerned about the temperature discrepancy between the blocks. If we accumulate some high-power cells into one of the blocks, then at a later stage we will not have a chance to move these cells out of the block, due to the divide-and-conquer nature of top-down partitioning. It is reasonable to assume that the temperature inside each of the sub-blocks are uniform, since if such an objective were to be enforced at every step of the partitioning, then a uniform temperature distribution would indeed result. Under this assumption, we will obtain a simplified thermal model and we do not have to make any assumptions about the precise cell locations inside the sub-block.

We can also look at the problem from the point of view of multigrid methods [13]. It has been known that Poisson equation (1) can be solved using multigrid methods effectively. The spatial variation of temperature can be thought of as having "high frequency" and "low frequency" components. A very uniform temperature distribution over space can be thought of as having predominantly "low frequency" components and very small "high frequency" components, and a very widely varying distribution can be considered to

show the opposite property. The multigrid method solves equation (1) on an $n \times n$ grid by using the following ideas: it builds a series of gradually refined meshes: $m_1 \times m_1, m_2 \times m_2, ..., m_k \times m_k$, where $m_k = n$, and each mesh is a coarsened mesh for all the meshes after it. It then solves for the lower "frequency" terms of the spatial distribution of T on the coarse meshes and interpolates the result onto the refined meshes. This method is based on the observation that lower frequency components of T can be effectively solved on a coarse mesh.

In our approach, if we limit our partition lines to the thermal meshes during the top-down hierarchical partitioning, we can think about the partition process as a series of operations on a set of gradually refined meshes. At any particular level, we are only concerned about the spatial distribution of the temperature in a specific "frequency range," and as the mesh is refined further during the top-down partitioning method, higher frequency terms, corresponding to local variations, are incorporated.

Now consider the first step in top down partitioning, where the chip is partitioned into two blocks, the left block and the right block. For simplicity, let us assume that number of thermal cells in each region is the same, although this assumption can easily be discarded. We will say that thermal cells $i = 1$ .. $m/2$ are in the left block and cells $i = m/2 + 1$ .. m belong to the right block. Now let us assume, as stated above, that the block on the left has an even temperature of T_l and the temperature of the block to the right is T_r. We can now simplify equation (3) to:

$$\begin{bmatrix} \sum_{i=1}^{m/2}\sum_{j=1}^{m/2} G_{ij} & \sum_{i=1}^{m/2}\sum_{j=\frac{m}{2}+1}^{m} G_{ij} \\ \sum_{i=\frac{m}{2}+1}^{m}\sum_{j=1}^{m/2} G_{ij} & \sum_{i=\frac{m}{2}+1}^{m}\sum_{j=\frac{m}{2}+1}^{m} G_{ij} \end{bmatrix} \begin{bmatrix} T_l \\ T_r \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m/2} P_i \\ \sum_{i=\frac{m}{2}+1}^{m} P_i \end{bmatrix} \quad (7)$$

This reasoning can be extended to a general case where we partition the chip into k regions, each with possibly a different number of cells. Each region i,$(i = 1, \cdots, k)$ has the same temperature $T_i^{'}$ and contains a thermal cell set S_i: $s_i^1, s_i^2, \cdots, s_i^{n_i}$, n_i is the number of thermal cells in the region, and S_i, $(i = 1..k)$ is a k-way partition of the index set $\{1, 2, \cdots, m\}$. Equation (3) can then be simplified into the following form:

$$\begin{bmatrix} G_{11}^{'} & G_{12}^{'} & \dots & G_{1k}^{'} \\ G_{21}^{'} & G_{22}^{'} & \dots & G_{2k}^{'} \\ \vdots & \vdots & \ddots & \vdots \\ G_{k1}{}' & G_{k2}^{'} & \dots & G_{kk}^{'} \end{bmatrix} \begin{bmatrix} T_1^{'} \\ T_2^{'} \\ \vdots \\ T_k^{'} \end{bmatrix} = \begin{bmatrix} P_1^{'} \\ P_2^{'} \\ \vdots \\ P_k^{'} \end{bmatrix} \quad (8)$$

where $G_{ij}^{'} = \sum_{i' \in S_i} \sum_{j' \in S_j} G_{i'j'}$ and $P_i^{'} = \sum_{i' \in S_i} P_{i'}$.

We have the following theorem for the new matrix $G^{'}$:

THEOREM 1. *If G is a positive definite matrix, the derived matrix $G^{'}$, as defined in equation (8), is also positive definite.*

PROOF. Let us construct a $k \times m$ matrix C, defined as follows:

$$C_{ij} = \begin{cases} 1 & \text{if } j \in S_i \\ 0 & \text{if } j \notin S_i \end{cases}$$

Since $S_i (i = 1..k)$ is a k-way partition of the index set $\{1, 2, \cdots, m\}$, the row vectors of C are linearly independent and $rank(C) = k$. It is also easy to verify that $G^{'} = CGC^T$ for any value of k; for example, for $k = 2$, it is easy to see that we obtain the matrix $G^{'}$ defined in Equation (7).

Now let us assume that $G^{'}$ is not positive definite and that there exists a nontrivial $1 \times k$ vector x such that $xG^{'}x^T \leq 0$. This leads to $(xC)G(xC)^T \leq 0$. Since $rank(C) = k$, xC is not a zero vector, and since G is given to be positive definite, this contradicts our assumption that $G^{'}$ is not a positive definite matrix. □

4. PARTITION-DRIVEN THERMAL PLACEMENT

Since the purpose of this study is to show how to handle the thermal constraints in the partitioner, we will limit our discussion to a top-down two-way partitioner using the original Fiduccia-Mathheyses algorithm [14]. The scheme described here can be extended naturally to incorporate a state-of-the-art multi-level partitioner.

First we define a concept called the "effective thermal influence region" of a block. Let us assume that we have already partitioned the chip into k blocks. The effective thermal influence region for one of the block is then defined as follows: if we have a unit heat source on the block, it corresponds to the area outside of which the temperature induced by the unit heat source is less than a certain percentage of the maximum temperature induced by the unit heat source. This notion is introduced to reduce the computational cost for incremental temperature updates for a large number of blocks, and it can be easily computed once we know the thermal resistance matrix.

Next, we introduce the parameter δT_S, which is used to guide our algorithm:

$$\delta T_S = \max\left(|T_i - T_{i,neighbors}|\right), i \in S, \quad (9)$$

where $T_{i,neighbors}$ are the temperature for cells adjacent to cell i and S is a set of blocks.

If S is a set of all the blocks on the chip at any given partition level, it is a measure of the maximum temperature gradient over the chip. We will use δT_{chip} for evaluating the quality of our results, where *chip* corresponds to the set of all blocks on the chip.

Figure 4 shows the top level algorithm for the flow. Our process starts with the computation of the conductivity matrix as described in section 2, after which we start the partition steps. Since we are performing top-down bipartitioning, we will start from the top level block and partition the every block into two blocks. This is done recursively until the number of standard cells contained in each block is less than a certain threshold.

Before we start partitioning at any partition level, we first decide the partition lines for all the blocks. This is necessary since we do not want to regenerate the thermal matrix for each individual block partition. The partition lines are aligned with the thermal meshes. Then we perform the following steps:

```
Algorithm Partition-Driven Thermal Aware Placement
  compute_conductivity_matrix();
  do {
    find_bipartition_locations_for_all_blocks();
    compute_simplified_thermal_model();
    for_each_blocks {
       generate_initial_solutions();
       initial_thermal_setup();
       while passes improve the result {
           create_gain_queue();
           while get_highest_gain_move() {
               if (!violate_thermal_constraints() and
                   !violate_all_other_constraints()) {
                  do_move();
                  update_gains();
               }
           }
           restore_best_result_in_pass();
       }
    }
  } until cells in each block are less than threshold
```

Figure 3: Top level flow for partition-driven thermal placement.

1. We first create the simplified thermal conductivity matrix $G^{'}$ as shown in equation (8). We then invert $G^{'}$ to obtain the thermal resistance matrix $R^{'}$, since this is required for the incremental update that is to be performed later. Here in the worst case, we will invert an $m \times m$ matrix, where m is the number of mesh node on the wafer surface.

2. Using the thermal influence region concept we introduced above, we convert $R^{'}$ to a sparse matrix $R^{'}_{sp}$ as follows: for each element $R^{'}_{ij}$, we compare it against the diagonal element $R^{'}_{ii}$. If $R^{'}_{ij}/R^{'}_{ii} < \epsilon$, where ϵ is a small number, we assign $R^{'}_{ij}$ to 0. The purpose of this is to reduce the computational expense of evaluating moves with a small loss in accuracy that can be bounded.

When we start to partition one block, first we randomly generate multiple solutions. For each solution, we compute δT_S, where S is a set of blocks which are adjacent to the blocks being partitioned. Supposing the maximum and minimum values are δT_{max} and δT_{min}, we then set the thermal budget for this partition to be $(1-\alpha)\delta T_{max} + \alpha \delta T_{min}$, where $0 \leq \alpha \leq 1$. The choice of α is a trade-off between partition quality and thermal constraints. The solution with the lowest δT is chosen as the initial solution for partitioning.

When the partitioner decides to move a cell, the thermal constraint will be one of the constraints that will decide whether the move is legal or not. This constraint determines whether move should be accepted or not using the following steps:

1. Compute the delta power vector.

   $$\Delta P = [0, \cdots, 0, \delta P_{i_1}, 0, \cdots, 0, \delta P_{i_2}, 0, \cdots, 0, \delta P_{i_l}, 0, \cdots, 0]$$

   where l is the number of blocks whose power dissipation is affected, and δP_{i_k} is the power dissipation change of block i_k induced by the movement. This set of blocks includes not only the blocks that the cell is moving from and moving to, but also blocks that contain cells that drive the input of the cell to be moved, since the wire load for these cells will also change.

2. Compute the delta temperature vector $\Delta T = R^{'}_{sp} \Delta P$ and correspondingly update δT_S using Equation (9). Since this involves a sparse matrix-vector multiplication, the computation cost is practically constant.

3. If the δT_S is within the current budget, the move is accepted. Otherwise, it is rejected.

One of the challenging tasks for the partitioner lies in finding a fast and accurate online wire length estimator. In this work, we use the "fast expected minimum heuristic" proposed by [15]. The wire capacitances are added as part of the load capacitances of the driver when we compute the cell power consumption.

Table 1: Parameters used in the experiment

Parameter	Value
Thermal conductivity for the top (W/m^2)	10
Thermal conductivity for the bottom (W/m^2)	8800
Thermal conductivity for the side (W/m^2)	7
Thermal conductivity for the silicon (W/m^2)	150
Unit wire capacitance (pF/m)	242
Input pin capacitance (pF)	0.1
Clock frequency (MHz)	800
Wafer thickness (μm)	500
Row height (μm)	7

5. EXPERIMENTAL RESULTS

Our experiments are implemented on top of the framework provided by [16], [17], with the Meschach Library [18] being used as our sparse matrix solver. The code is implemented in C++ on a Sun SparcV9 450 MHz machine running SunOS 5.8.

We have applied our algorithm to 8 standard cell test cases in the MCNC benchmark suite [19]. These test cases are row-based and have a fixed area. Our experiments on the test cases were based on zero row spacing, which is the style for modern designs. The experimental parameters are adopted from [2], with the exception of the thermal conductivity near the top of the wafer. We choose a low thermal conductivity number considering the passivation layer on top of the chip [20]. These parameters are listed in Table 1. The thermal mesh that we use in the (x, y) direction is aligned to the row boundaries; in case different rows have different heights, a nonuniform mesh may be used. The pitch of the mesh is one row for small testcases and two rows for the large testcases, so that the mesh size in the (x, y) direction varies from 23×23 to 59×59 and the mesh size is the z direction is 6. The ambient temperature is assumed to be 0 °C. The switching activity factor for the driver is randomly generated in a range from 0 to 1. In our experiment, we generate 10 initial solutions to compute the thermal budget for a partition and choose α to be 0.5.

Table 2 shows the experimental results for 8 test cases. The results from the runs with and without thermal constraints are shown. Here T_{max} is the maximum temperature

Table 2: Experiment result for MCNC benchmarks

Benchmark		Without thermal constraints				With thermal constraints			
Name	cells	Wire length	T_{max}	T_{ave}	δT_{chip}	Wire length	T_{max}	T_{ave}	δT_{chip}
struct	1952	0.0476	15.3	13.2	0.67	0.0501(105%)	15.6	13.1	0.50 (75%)
primary1	833	0.0480	11.7	9.74	0.56	0.496(103%)	11.9	9.75	0.54 (96%)
primary2	3014	0.184	40.3	34.6	1.95	0.192(104%)	39.9	34.4	1.52 (78%)
biomed	6514	0.258	85.8	44.4	13.4	0.260(101%)	64.0	45.2	5.64 (42%)
industry2	12637	1.04	55.6	44.1	4.60	1.04(100%)	57.5	44.1	3.92 (85%)
industry3	15433	0.986	57.3	44.0	2.50	1.03(104%)	56.6	43.7	1.80 (72%)
avqsmall	21918	0.637	138.5	37.6	40.9	0.667(105%)	111.5	37.8	34 (83%)
avqlarge	25178	0.701	158.2	36.4	45.4	0.746(106%)	102.0	35.7	27.6 (61%)

across the chip, T_{ave} is the average temperature and δT_{chip} is the maximum on-chip temperature gradient as defined in (9), where S is the whole chip. The unit for temperatures is °C. The wire length, in meters, is measured using the half parameter of the net bounding box. From the results, we can see that the wire-length is typically 0% to 6% longer with the thermal constraints. The run time overhead varies from 250% to 350% and most of the run time overhead is related to the matrix computations after the partition lines are determined, and to the generation of multiple initial solutions. From the experiment results, we can see that our scheme effectively reduces the maximum temperature gradient across the chip.

6. CONCLUSION

In this paper, we have described an algorithm to derive a compact thermal model with a computational complexity of $O(mn+m^2)$ and have presented a simplified thermal model to be used in partition-driven placement. We have also proposed a scheme to perform top-down partition-driven placement using this simplified thermal model. Since we are able to use the temperature directly as a constraint, this enables us to better control of the thermal profile. Our experiments show that our scheme is successful in effectively smoothing out the temperature profile.

7. REFERENCES

[1] K. Banerjee, A. Mehrotra, A. Sangiovanni-Vincentelli, and C. Hu, "On thermal effects in deep sub-micron VLSI interconnects," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 885-891, 1999.

[2] C. Tsai and S. Kang, "Cell-level placement for improving subtrate thermal distribution," *IEEE Transactions on Computer-Aided Design*, Vol. 19, No. 2, pp. 253-266, Feb 2000.

[3] C. Chu and D. F. Wong, "A matrix synthesis approach to thermal placement," *IEEE Transactions on Computer-Aided Design*, Vol. 17, No. 11, pp. 1166-1174, Nov 1998.

[4] G. Ausiello *et al.*, *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability*, Berlin, Germany: Springer, 1999.

[5] W. Sun and C. Sechen, "Efficient and effective placement for very large circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 14, No. 3, pp. 349-359, March 1995.

[6] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich, "GORDIAN: VLSI Placement by quadratic programming and slicing optimization," *IEEE Transactions on Computer-Aided Design*, Vol. 10, No. 3, pp. 356-365, March 1991.

[7] C. J. Alpert, T. Chan, A. B. Kahng, I. Markov, and P. Mulet, "Faster minimization of linear wirelength for global placement," *IEEE Transactions on Computer-Aided Design*, Vol. 17, No. 1, pp.3-13, Jan 1998.

[8] C. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 530-533, 1997.

[9] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: application in VLSI design," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 526-529, 1997.

[10] K. Zhong and S. Dutt, "Effective partition-driven placement with simultaneous level processing and global net views," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 254-259, 2000.

[11] G. Golub and C. F. Van Loan, *Matrix Computation*, Baltimore, MD: John Hopkins University Press, 1996.

[12] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge, UK: Cambridge University Press, 1985

[13] U. Trottenberg, C. Oosterlee and A Schuller, *Multigrid*, New York: Academic Press, 2001.

[14] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 175-181, 1982.

[15] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov, and A. Zelikovsky, "On Wirelength Estimations for Row-Based Placement," *IEEE Transactions on Computer-Aided Design*, Vol. 18, No. 9, pp. 1265-1278, Sep 1999.

[16] http://vlsicad.cs.ucla.edu/software/PDtools/

[17] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Design and Implementation of Move-Based Heuristics for VLSI Hypergraph Partitioning," *ACM Journal of Experimental Algorithms*, Vol. 5, 2000.

[18] http://www.netlib.org/c/meschach/

[19] P. Madden, "Reporting of Standard Cell Placement Results," *Proceedings of the International Symposium on Physical Design*, pp. 30-35, 2001.

[20] Y. Cheng and S. Kang, "A Temperature-Aware Simulation Environment for Reliable ULSI Chip Design," *IEEE Transactions on Computer-Aided Design*, Vol. 19, No. 10, pp. 1211-1220, Oct 2000.