Thermal-aware Placement of Standard Cells and Gate Arrays: Studies and Observations

Prasun Ghosal  
Bengal Engg. & Sc. University, India  
p_ghosal@it.becs.ac.in

Tuhina Samanta  
Bengal Engg. & Sc. University, India  
t_samanta@it.becs.ac.in

Hafizur Rahaman  
Bengal Engg. & Sc. University, India  
rahaman_h@it.becs.ac.in

Parthasarathi Dasgupta  
Indian Institute of Management Calcutta, India  
partha@iimcal.ac.in

Abstract

In high-performance VLSI circuits, the on-chip power densities are playing dominant role due to increased scaling of technology, increasing number of components, frequency and bandwidth. The consumed power is usually converted into dissipated heat, affecting the performance and reliability of a chip. In this paper, we consider the placement of standard cells and gate arrays (modules) under thermal considerations. Our contributions include: (i) an algorithm for optimal placement of the gates or cells to minimize the possible occurrence of hot spots, (ii) results of sensitivity analysis of thermal characteristic of a layout with respect to the power densities of the modules in the layout, and identifying three classes of modules, and (iii) an algorithm for optimal placement of modules, with minimum possible occurrence of hot spots, and reasonable estimated interconnect lengths. Experimental results on randomly generated and standard benchmark instances are quite encouraging.

1 Introduction

Increased technology scaling, and increased number of components in high-performance integrated circuits have resulted in greater emphasis on thermal considerations. The increasing temperature is likely to affect multiple design parameters, such as transistor delay, interconnect delay, electro-migration effects, leakage power, and of course, the reliability of a chip. With increasing chip sizes, due to increased number of components, the wire length is also likely to increase. As such, both timing control and energy consumption dominate the design process.

Moreover, for increased power consumption, under design, fabrication and packaging constraints, the heat dissipation by the modules of the chip are usually uneven, producing hot spots. As such, physical design of a VLSI chip requires an optimal placement of the modules such that heat dissipation by these modules are evenly distributed.

1.1 Placement Problems for Standard cells and Gate Array

Placement phase of VLSI design is well-researched [11], having objectives such as minimization of area and wirelength estimates, satisfying delay budgets, and so on [10, 2]. Recent challenges in the performance-driven paradigm include routability of interconnects, congestion minimization, noise and crosstalk minimization, and thermal considerations [7, 4]. High-quality placements are essential for various VLSI design models and sizes. Standard cell placement typically involves a number of rows of cells where each cell represents a simple circuit viz. flip-flop, logic gate etc and is stored in the cell library [10]. In Gate Array placement, however, each cell is an array of transistors and can implement gate or latch by interconnecting transistors. The major difference between placement of Standard cells and Gate Array is that in the former case, cells may be of different width. As such, interchange of cells without consideration of their sizes is likely to result in overlapping of cells. The latter, however, can be taken care of through efficient overlap elimination techniques [1].

In this paper, we consider the placement of Standard cells or Gate arrays with three major objectives: (i) finding an optimal placement with minimum possible occurrence of hot spots, (ii) performing sensitivity analysis of overall thermal characteristic of a layout with respect to the power densities of the modules in the layout, and thereby identifying three classes of modules, (iii) finding optimal placement
with minimum possible occurrence of hot spots, and reasonable estimated interconnect lengths. Hereafter, we use the term module to refer to a gate or a standard cell.

The rest of the paper is organized as follows. Section 2 provides the preliminaries and literature survey related to the work along with the motivation for the work. Section 3 introduces the problem. Sections 4, 5 and 6 respectively discuss the algorithm for finding the optimal thermal placement, the framework for the Sensitivity Analysis, and the algorithm for the composite objective function. Section 7 illustrates the empirical observations for some random instances and benchmarks, and Section 8 concludes the paper.

2 Preliminaries and Motivation of the work

2.1 Hot spot generation in placement

Heat dissipation is important in VLSI module placement. Total heat dissipation in an integrated circuit comprises static and dynamic components, along with increasing leakage power dissipation [6]. The dynamic power dissipation $P_{\text{dyn}}$ resulting from voltage transitions as transistors turn on and off, is given by

$$P_{\text{dyn}} = \alpha \times C \times V^2 \times f_{\text{clk}}$$  \hspace{1cm} (1)

where $\alpha$ is the % of transistors switching at a given time, $C$ is the total capacitance, $V$ is the supply voltage, and $f_{\text{clk}}$ is the the operating frequency. This dissipation increases with increasing number of circuits in a chip (increasing total capacitance), and increasing operating speed. Scaling of feature sizes from micron to nanometer scales do cause reduction in individual transistor capacitances and hence reduces the power dissipation. However, with increasing number of transistors, and on-chip local frequencies, the heat dissipation increases rapidly. For instance, heat dissipation for high-end CPUs is forecast to increase to 200 W, with increase in number of transistors per chip (553 M to 1106 M) and doubling of on-chip local clock frequency.

2.2 Non-uniform power dissipation and hot spots

Localized areas of high heat flux are becoming prominent with increasing scaling of process technology, along with increase in total power dissipation. These localized areas create hot spots, which are usually significantly above the average die temperature. Hot spots limit the performance of the circuit, reliability and yield. In high-performance integrated circuits, the close packing of transistors yields local high-temperature regions. Moreover, disparities in local heat generations are aggravated by the placement of core processors that dissipate considerable power in close proximity to cache memory, the latter dissipating little power.

2.3 Related works

Electrothermal issues arising due to power dissipation both at micro- and macro-scale is key to the development of current microprocessors [8]. The associated problem of thermal placement has been studied in the past for printed circuit boards and multichip modules [3]. However, since thermal placement of gates within a single chip was not of major concern in the past, existing placement algorithms focus on minimizing area and delay [7, 2], and generally do not consider heat dissipation. The problem of ensuring uniformity of sources of heat in a placement is considered in [4, 9]. The work in [5] proposes a force-directed method for distributing cells based on a heat map. However, the work reported in [14] shows that a uniform heat distribution does not necessarily lead to a uniform temperature distribution. [13] provides a solution for improving circuit performance considering the thermal parameters.

2.4 Motivation of the work

To the best of our knowledge, very little have been reported so far on ensuring uniformity of power dissipation across standard cell or gate array layouts. It is thus quite reasonable to look for an efficient method for uniform thermal arrangement for such a layout. Considering the importance of both wirelength and thermal considerations, it seems imperative to consider both the traditional HPWL estimate, and the thermal uniformity. In this paper, we propose to solve this composite problem as well. An interesting related issue would be to study the effect of the power dissipations of the individual modules on the overall thermal condition of the entire layout. We propose a scheme of classification based on this sensitivity analysis. As an outcome of this analysis, we broadly identify three classes of modules. We believe that such classification would serve as a guiding factor to define the routing paths in the subsequent phases. We adopt the matrix synthesis problem (MSP) [4] as the basic framework of our work.

3 The problem

The Matrix Synthesis Problem models the thermal placement problem and [4] suggests three ways to solve it. A method to calculate temperature based on power estimation for standard cell placement has been suggested in [14]. A methodology to distribute the temperature of gates evenly on a chip while reducing the power consumption has been introduced in [9].
Finding Critical Threshold for a Placement

The basis of our problem formulation is a matrix $M$ of nonnegative real numbers representing the power dissipations of the modules. A $t \times t$ sub-matrix of $M$ corresponds to a region of the chip, and $S_t(M)$ denotes the set of all such disjoint submatrices of $M$. The sub-matrix having the largest sum represents hottest region on the chip. $MSP$ attempts to find a placement of the modules such that the temperature of the hottest region is as small as possible.

The parameter $t$ models the rate of heat transfer. In general, the power dissipation of a cell or gate is affected by several factors like circuit structure, functionality of the gate, wire loading, input data and so on. In fact, estimation of power dissipation itself is a well-known hard problem [12]. Consider Figure 1 from [4] illustrating a $MSP$.

![Figure 1. An example MSP with $m = 4$, $n = 3$ and $t = 2$. (a) is a bad solution (maximum sum = 27), (b) is an optimal solution (maximum sum = 13)](image)

We define Critical Threshold as follows.

**Definition 1** For a given set of elements of a matrix $M$, representing power dissipations of an array of modules and a given value of $t$, the Critical threshold of $M$ is the minimum value of the maximum aggregate power dissipations of the sub-matrices of order $t \times t$, among alternative possible arrangements of the elements of $M$.

Let $\xi(M)$ denotes the Critical Threshold for an input matrix $M$. The problem (similar to the $MSP$ [4]) can then be stated as follows:

For a given layout of modules, let the power dissipations of the modules be represented by a $m \times n$ matrix $M$ of real numbers. Let $S_t$ be the set of $t \times t$ sub-matrices of $M$. Let $\sigma(M)$ be the sum of all entries in $M$. Let $\mu_t(M) = \max_{S\subseteq S_t(M)} \sigma(S)$. The problem is to arrange the elements of $M$ such that the value of $\mu_t(M)$ is minimum.

Thus, by definition, $\xi(M) = \min(\mu_t(M))$. Finding an arrangement in a matrix for the critical threshold is equivalent to finding a placement of the modules such that the temperature of the hottest region is minimum among alternative possible placements.

4 Finding Critical Threshold for a Placement

In the matrix layout for our problem, the sub-matrix with the largest sum corresponds to the hottest region on the chip. For poor heat transfer, the effect of a gate is mostly on neighboring gates, and $t = 2$ may be a good model to use. On the other hand, if the heat transfer is good, we may want to consider larger regions and hence a larger $t$.

4.1 Proposed Algorithm

The proposed algorithm starts with an initial arbitrary placement $\Omega_0$ of modules, and their power dissipations represented by real numbers arranged in the matrix $M$. W.l.o.g., for simplicity, we assume that $m = n = tq$ for some integer $q$, i.e. order of the matrix is $tq \times tq$, and $t^2 \times q^2$ cells or gates are to be placed. Hence, $S_t(M)$ is a set of $q^2$ non-overlapping sub-matrices that covers the whole matrix $M$.

The notations used in the description of the algorithm are given in Table 1.

| $\text{outer}_{\text{iter}}$ | Maximum number of iterations in outer loop |
| $\text{inner}_{\text{iter}}$ | Maximum number of iterations in inner loop |
| $\max_{\text{ascent}}$ | Maximum allowed increment in the value of the objective function in the inner loop |
| $t$ | A small integer |
| $M$ | $m \times n$ matrix for all the modules |
| $S_t$ | $t \times t$ sub-matrix of $M$ |
| $\Omega_t(M)$ | Placement of all modules in outer $t^{th}$ iteration |
| $\sigma_{\text{max}}(S_t)$ | Maximum Aggregate value of $S_t$ at any point in inner loop |
| $\sigma_{\text{min}}(S_t)$ | Minimum Aggregate value of $S_t$ at any point in inner loop |
| $\mu_t(M)$ | Minimum value of $\mu_t(M)$ in a particular inner loop |
| $\mu_t^\text{best}(M)$ | Global minimum value of $\mu_t(M)$ at any point of execution |
| $\sigma_{\text{max}}^\text{best}(M)$ | Module having maximum value at an outer $t^{th}$ iteration and an inner $t^{th}$ iteration corresponding to $\sigma_{\text{max}}(S_t)$ |
| $\sigma_{\text{min}}^\text{best}(M)$ | Module having minimum value at an outer $t^{th}$ iteration and an inner $t^{th}$ iteration corresponding to $\sigma_{\text{min}}(S_t)$ |
| $\mu_t^\text{best}(M)$ | Minimum value of the composite objective function in inner loop |
| $\mu_t^\text{best}(M)$ | Global minimum value of composite objective function |
| $\mu_t^\text{best}(M)$ | Value of composite objective function at any point |

Table 1. Table of notations

The parameters $\text{outer}_{\text{iter}}$, $\text{inner}_{\text{iter}}$ and $\max_{\text{ascent}}$ are configurable. At every inner iteration, a new random matrix is generated, and the algorithm gets into the inner loop again. When the algorithm comes out of the inner loop, a new random matrix is generated, and the algorithm gets into the inner loop again. When the algorithm gets out of the inner loop, the current value of $\mu_t^\text{best}(M)$ is compared with the value of $\mu_t^\text{best}(M)$. If $\mu_t^\text{best}(M)$ is found to be less than $\mu_t^\text{best}(M)$, the value of $\mu_t^\text{best}(M)$ is set to the new value of $\mu_t^\text{best}(M)$. The
algorithm terminates when the outer loop count exceeds 2.

Algorithm for Finding Critical Threshold of a placement

Input: Integer \( m = \text{size}(q) \); order of matrix to be synthesized
Integer \( t \); order of the sub-matrix
A list of \( n^2 = t^2 \times q^2 \) non-negative real numbers

Output: Critical threshold value, \( \zeta(M) \) for \( M \) for a specific \( t \)

1. \( \Omega_t = \text{gen_random}(); \) (* generate initial random placement *)
2. find \( \sigma_{\text{max}}(S_t); \) find \( \sigma_{\text{min}}(S_t); \)
3. if \( \sigma_{\text{max}}(S_t) = \sigma_{\text{min}}(S_t) \) then return
4. \( \mu_{\text{best}}(M) = \sigma_{\text{max}}(S_t); \) \( i = 0 \)
5. repeat (* outer loop begins *)
   6. \( \mu_{\text{loc}}(M) = \sigma_{\text{max}}(S_t); \) \( j = 0 \)
   7. repeat (* inner loop begins *)
5. Swap(\( \mu_{\text{max}}^i \cdot \mu_{\text{min}}^j \))
9. find \( \sigma_{\text{max}}(S_t); \) find \( \sigma_{\text{min}}(S_t); \)
10. if \( \sigma_{\text{max}}(S_t) > (\mu_{\text{best}}(M) + \max_{\text{ascent}}) \) then break
11. if \( \sigma_{\text{max}}(S_t) < \mu_{\text{best}}(M) \) then \( \mu_{\text{loc}}(M) = \sigma_{\text{max}}(S_t) \)
12. \( j = j + 1 \)
13. until \( j \geq \text{inner\_iter} \)
14. if \( \mu_{\text{loc}}(M) < \mu_{\text{best}}(M) \) then \( \mu_{\text{best}}(M) = \mu_{\text{best}}(M) \)
15. \( \Omega_t = \text{gen_random}(); \) (* generate new random placement *)
16. until \( (i > \text{max\_outer\_iter}) \)
17. \( \zeta(M) = \mu_{\text{best}}(M) \)
18. return \( \zeta(M) \) (* critical threshold *)

Figure 2. Finding Critical Threshold

The proposed algorithm has two levels of iteration. In the inner loop, attempt is made to reach a local optimum. The outer loop starts with a different starting solution, and through the inner loop attempts to find a different local optimum. The final solution is the best of all such local optima located at different positions of the search space. To ensure that the random placements generated in the outer loop do not repeat, the positions of the last two modules interchanged in the inner loop are preserved. A new random placement is obtained by swapping any one of the last two modules swapped with any other module of the placement at random. This reduces the probability of generating duplicate placement.

The following observation on the swapping done within the inner loop is obvious.

Observation 1 The swap operation in the proposed algorithm can never increase the value of the objective function for a solution from its immediate predecessor.

5 Sensitivity Analysis: A framework

In these experiments, we incrementally change the power dissipation values for the individual modules, and study their effects on the critical threshold of the placement. We consider static situation, where power dissipations of the respective modules are gradually varied in turn, and the variations of the corresponding critical thresholds are noted. Three types of temporal variations are considered: linear, quadratic, and sub-linear. For each module, a set of values of power dissipations are considered, and the corresponding critical thresholds of the layout are obtained. Let \( \kappa_i \) denotes the number of times the critical threshold increases for an increase in power dissipation value of module \( i \), and \( N \) denotes the total number of observations for every module. Then, a module is classified as strongly sensitive if \( \frac{\kappa_i}{N} \geq 0.5 \), and weakly sensitive if \( \frac{\kappa_i}{N} \leq 0.1 \). All other modules are classified as moderately sensitive.

6 Critical Threshold and Wirelength

Traditional placements primarily based on the objective of total wirelength. Assuming Manhattan interconnects, length of a net is typically measured in terms of half the perimeter of the minimal bounding-box (HPWL) encircling the pins. The composite objective function is the geometric mean of the total wirelength (HPWL), and \( \sigma_{\text{max}}(S_t) \), for a given \( t \). This tends to assign equal weights to the two constituent objectives. The proposed algorithm is same as that of Figure 2, except that \( \sigma_{\text{max}}(S_t) \), \( \mu_{\text{loc}}(M) \) and \( \mu_{\text{best}}(M) \) are respectively replaced by \( \text{obj}(M) \), \( \text{obj}_{\text{loc}}(M) \), and \( \text{obj}_{\text{best}}(M) \). The rest of the algorithm is the same as that in Figure 2. Observation 1 does not generally hold in this case.

7 Empirical Observations

The experiments are performed on a PC running on Intel chip with 1 GB memory and 3.6 GHz clock speed.

It is important to note that, in all our implementations on Critical Threshold, we consider only non-overlapping submatrices. As such, all possible groups of modules are not considered. As a result, we may not always guarantee optimality of the final solution. Experiments are performed with a set of MCNC benchmark circuits, and some random instances. Since thermal data of the MCNC circuits are not available, the power dissipations for the individual gates are generated with a uniform random number generator. The same is true for the random instances as well. The value of \( t \) is 2.

For the random instances, we generate set of matrices \( M \) of different orders between 10 \( \times \) 10 and 200 \( \times \) 200. For each \( M \), we vary \( t \) between 2 and half the order of \( M \). We note the initial and final values of \( \mu_i(M) \), and compute the percentage improvement. Since we do not use Simulated Annealing, the results cannot be better than those reported in [4]. However, the algorithm proposed by us is expected to be faster than that proposed in [4]. For \( t = 2 \), our results
are at par with those of [4] when the order of matrix $M$ is small. Our results degrade for larger submatrices. Tables 2 and 3 summarize the results for the experiments on Critical Threshold for random instances and benchmarks. In Table 3, $m$ stands for a $m \times m$ matrix. We summarize below our empirical observations (Figures 3 and 4).

- The critical threshold, as expected, increases with the value of $t$ (i.e., with the conductivity of the substrate).
- The relative improvement in $\mu_t(M)$ is high for very small values of $t$, and gradually decreases for larger values of $t$. This is because for larger $t$, there is very little scope of swapping in our algorithm.
- The improvement in $\mu_t(M)$ is high for very small $m$, and very gradually decreases for larger values of $t$. This is expected since as the size of $M$ increases, the initial value of $\mu_t(M)$ becomes larger, thereby rendering its relative improvement insignificant.

<table>
<thead>
<tr>
<th>Order of main matrix</th>
<th>Order of sub-matrix</th>
<th>Initial Threshold</th>
<th>Critical Threshold</th>
<th>% improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>248.1</td>
<td>198.7</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>313.5</td>
<td>210</td>
<td>33</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>1278.7</td>
<td>1336.8</td>
<td>2.2</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>4942.95</td>
<td>4910.9</td>
<td>0.65</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
<td>31791.7</td>
<td>31197</td>
<td>1.9</td>
</tr>
<tr>
<td>150</td>
<td>2</td>
<td>387.1</td>
<td>321.5</td>
<td>16.1</td>
</tr>
<tr>
<td>300</td>
<td>2</td>
<td>387.1</td>
<td>331</td>
<td>14.5</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
<td>498925.46</td>
<td>498625.62</td>
<td>0.06</td>
</tr>
</tbody>
</table>

Table 2. Summary of results for experiments on Critical Threshold for random instances

<table>
<thead>
<tr>
<th>Circuits</th>
<th>Name</th>
<th>Size</th>
<th>$m$</th>
<th>Initial Threshold</th>
<th>Critical Threshold</th>
<th>% improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>s5378</td>
<td>2978</td>
<td>56</td>
<td>362.25</td>
<td>264.3</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>s4234</td>
<td>5844</td>
<td>78</td>
<td>371.8</td>
<td>294.8</td>
<td>21.1</td>
<td></td>
</tr>
<tr>
<td>s33201</td>
<td>3727</td>
<td>94</td>
<td>318.7</td>
<td>227.1</td>
<td>13.3</td>
<td></td>
</tr>
<tr>
<td>s38580</td>
<td>20871</td>
<td>146</td>
<td>384.6</td>
<td>316.4</td>
<td>17.7</td>
<td></td>
</tr>
<tr>
<td>s31413</td>
<td>24061</td>
<td>156</td>
<td>383.1</td>
<td>325.3</td>
<td>15.1</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Summary of results for experiments on Critical Threshold for MCNC benchmarks

Table 4 summarizes the results for experiments on the composite objective function of Critical Threshold and Total wirelength. [4] performs a similar experiment on thermal uniformity and wirelength in two phases. It attempts to reduce the value of $\mu_t(M)$ in the first phase, and then attempts to reduce the wirelength. Our methodology is unique in that it assigns equal weights to both the objectives, and attempts to minimize the composite objective function. It is interesting to note that our methodology improves both the values of $\mu_t(M)$ and the total $HPWL$ in all the cases observed. For the MCNC benchmark $s5378$, for instance, for $t = 2$ and $m = 56$, the initial threshold is 362.25, and the initial total wirelength is 20673294. The final Critical Threshold and final total wirelength are respectively 320.6 and 20672694. Thus, the gain in Critical Threshold and wirelength are respectively 11.6% and 0.003%. The composite objective function attains optimal value within very few iterations. It is observed to be very high for small matrices, and gradually falls for larger orders of $M$. Again, this is expected since with increasing size of $M$, relative improvement of the composite objective becomes insignificant.

The sensitivity analysis is done on a given $20 \times 20$ matrix of power densities, and the corresponding submatrix has $t = 2$. Three sets of temporal variations of the power densities of the blocks are considered. For each of these sets, 10 different time instances are considered, and the power densities of the individual modules are varied linearly, quadratically, and sub-linearly with time. Figures 5 and 6 show the heat maps for the sub-linear and quadratic variations of power densities of the modules. The black, white, and gray rectangles respectively represent the strongly sensitive, the weakly sensitive, and the moderately sensitive modules. As expected, the numbers of gray and black rectangles increase as the dissipation varies from sub-linear to quadratic order.
Table 4. Summary of results for experiments for composite objective function

<table>
<thead>
<tr>
<th>No. of</th>
<th>Order of</th>
<th>Order of</th>
<th>Initial</th>
<th>Initial total</th>
<th>Final</th>
<th>Final total</th>
<th>% improvement of</th>
<th>No. of</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>main matrix</td>
<td>submatrix</td>
<td>threshold</td>
<td>wirelength</td>
<td>threshold</td>
<td>wirelength</td>
<td>objective function</td>
<td>iterations</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>5</td>
<td>1751.7</td>
<td>19074</td>
<td>1236.8</td>
<td>1888.2</td>
<td>2.15</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>15</td>
<td>1475.7</td>
<td>138935</td>
<td>1299.8</td>
<td>1888.2</td>
<td>6.2</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>5043.3</td>
<td>138935</td>
<td>4911.7</td>
<td>138935</td>
<td>2.1</td>
<td>3</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>15</td>
<td>11718</td>
<td>4460226</td>
<td>11162.3</td>
<td>4455492</td>
<td>9.4</td>
<td>6</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>5</td>
<td>1548.3</td>
<td>18776379</td>
<td>1877582</td>
<td>1877582</td>
<td>1.9</td>
<td>36</td>
</tr>
</tbody>
</table>

Figure 5. Sensitivity analysis for modules on Critical Threshold for sub-linear variation of power dissipation

Figure 6. Sensitivity analysis for modules on Critical Threshold for quadratic variation of power dissipation

8 Conclusions and Future Scopes of work

The paper considers the effect of power dissipation of a number of standard cells, or gates on the overall thermal characteristics of a chip. It defines a new metric for thermal uniformity in a layout, called the Critical Threshold. Experiments are performed to study the thermal sensitivity of the chip with respect to the power dissipations of the individual modules, and thus, three classes of modules are identified.

The proposed algorithms and the observations are very useful for at least two reasons: (i) it helps in guiding the subsequent routing phase, so as to avoid regions of high thermal sensitivity in a chip; and (ii) can be used to select the device layers for the different modules in a 3D chip, based on their varying thermal behaviors.

References