Microelectronics Journal 43 (2012) 423–432

Contents lists available at SciVerse ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

3D IC floorplanning: Automating optimization settings and exploring new thermal-aware management techniques

Felipe Frantz, Lioua Labrak*, Ian O'Connor

Institut des Nanotechnologies de Lyon INL-UMR5270, Université de Lyon, CNRS, Ecole Centrale de Lyon, Ecully, F-69134, France

ARTICLE INFO

Article history:
Received 2 August 2011
Received in revised form
3 March 2012
Accepted 6 March 2012
Available online 30 March 2012

Keywords:
3D–IC
Floorplanning
Multi-objective optimization
Thermal
TSV

ABSTRACT

The introduction of 3D chip architectures is an increasingly attractive integration solution due to the potential performance improvement, power consumption reduction and heterogeneous integration. Nevertheless, thermal distribution, evacuation and limitation constitute some of the key issues that can hinder widespread adoption of 3D integration technology. Efficient 3D floorplan algorithms have to be developed to address such complexity. In this paper we first discuss the implementation of such an algorithm and identify parameters that play a role in the solution quality. We then propose the use of a genetic algorithm to discover sets of parameters that guarantee good floorplan quality. Then, we present an improved thermal-aware floorplanner based on a new formulation of the cost function that minimizes not only peak temperature, but also thermal gradients. The temperature minimization goal is reinforced using a smart heuristic that guides 3D moves in the direction of placing power hungry blocks next to the heat sink. Experimental results show the ability of the method to reduce the temperature peak and gradient significantly, while maintaining area, wirelength and computation time.
© 2012 Elsevier Ltd. All rights reserved.

1. Introduction and related work

Three-dimensional integration, where multiple device layers are vertically stacked and interconnected, is perceived as a solution to scale the performance of electronic devices beyond Moore's law. It provides means to drastically decrease interconnect length, which directly results in increased speed [1,2], and to combine various technologies (digital, analog, memory, etc.) [3] and physical domains [4] in a single product, thereby greatly extending the capabilities of systems-on-chip (SoC). Other interesting characteristics such as low power consumption and high performance are expected from 3D integration, and make this solution a good candidate for a wide range of applications (medical, automotive, communication, wireless, etc.) [5].

However, these benefits come at a price: testing becomes difficult [6], yield can decrease rapidly with the number of layers in the stack [7], while the design space grows exponentially with the number of layers. Another major concern is the heat evacuation problem [8]. Indeed, when stacking two 100 W/cm^2 microprocessors, the net power density becomes 200 W/cm^2, which is beyond the heat removal capacity of currently available air-cooled heat sinks. Therefore, inter-layer micro channel cooling [9] is proposed as an alternative for these high performances applications.

In a 3D–IC design flow, we identify three main approaches to reduce on-chip temperature: (i) thermal-aware floorplanning, (ii) thermal via insertion and (iii) package and heat sink design. While both (ii) and (iii) have been proved to help heat dissipation [10–13], they require additional silicon area and external components (fan, pump, etc.), respectively. Thus, to be cost effective, the use of these latter solutions should be limited by tackling the heat problem during floorplanning.

Thermal-aware floorplanners have already been presented by multiple authors, all of them minimizing a weighted sum of area, wirelength and peak temperature. Zhou et al. [14] use an analytical method, where temperature is a force repelling blocks from clustering. Cong et al. [15] use the simulated annealing heuristic (SA) to solve the minimization problem and introduce a fast method to compute temperature. In [16], the authors also use SA, but suggest a two-phase algorithm where temperature is only minimized in the second stage.

In this work, we propose to address several issues to facilitate the implementation and improve the performance of thermal aware floorplanners. First, we consider the process of tuning the floorplan algorithm. Indeed, even if some floorplanners use analytical methods, an overwhelming number rely on the simulated annealing (SA) heuristic to determine the best floorplan. This heuristic (as with all optimization algorithms) needs to be tuned to the specific problem to allow a fast and efficient

* Corresponding author.
E-mail addresses: felipe.frantz@ec-lyon.fr (F. Frantz), lioua.labrak@ec-lyon.fr, lioualabrak@hotmail.com (L. Labrak), ian.O'connor@ec-lyon.fr (I. O'Connor).

0026-2692/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.mejo.2012.03.005

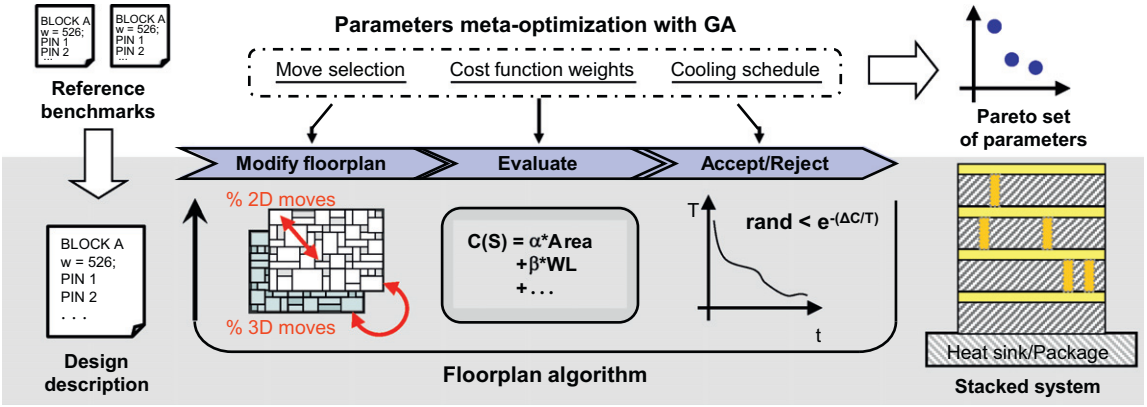

Fig. 1. Workflow of the floorplan algorithm and the meta-optimization.

convergence to a global minimum. This tedious task is generally not discussed, and most attention is devoted to the other crucial aspect, the cost function. We propose techniques to improve the floorplan quality on both aspects in the following ways:

- We use a multi-objective optimization based on Genetic Algorithms (GA) to find the tuning parameters for a 3D–IC floorplanner. This meta-optimization is performed offline (Fig. 1) on a limited set of benchmarks and generates the default parameters to be used on all other problems.
- We propose a new cost function formulation that allows Through Silicon Via (TSV) dimensions to be taken into account. Indeed most other approaches consider the TSV count in the floorplan without including its impact on the total area. Here, we include the area and height of the 3D-via directly into the footprint and wirelength computation, providing means to better evaluate the tradeoffs of using such interconnection.

Second, we propose to increase the capabilities of thermal-aware floorplanners in two ways:

- By using a relevant formulation of the optimization problem to reduce peak temperature as well as thermal gradients while maintaining a good tradeoff in area and wirelength.
- By adding a smart heuristic that will help the floorplan search algorithm to converge to thermally efficient solutions.

The remainder of this paper is organized as follows. Section 2 is dedicated to the implementation of the floorplan algorithm, discussing the cost function, the floorplan representation and the critical parameters of the SA heuristic. Section III details our meta-optimization approach. Section IV provides a background on the temperature minimization problem identifying key elements and pitfalls. Our proposed methodology for thermal-aware floorplanning is presented in Section V. The experimental results are analyzed and discussed in Section VI. Finally, Section VII concludes our work.

2. 3D floorplan algorithm

In this section we discuss the building blocks of a floorplanning algorithm based on SA and identify, for each of them, design parameters that can affect the convergence of the optimization and the quality of the floorplan. First we recall the general formulation of a 3D floorplan problem. Then we discuss the writing of the cost function, the choice for a floorplan representation, the main design choices for the SA heuristic and finally how to perturb the solution. This description will allow us to define the tuning parameters that must be optimized to build an efficient floorplanner using our meta-optimization approach.

2.1. Problem formulation

Let $B=\{b_1, b_2, \ldots, b_m\}$ be a set of rectangular blocks with height h_i and width w_i, and let $T=\{t_1, t_2, \ldots, t_p\}$ be the set of terminals. Each block has a set of pins P_i that connects it to the pins of other blocks and the terminals, forming nets. Let $L=\{l_i | 1 \le i \le n\}$ be the set of n layers. Let (x_j,y_j,l_j) denote the coordinates of terminal t_j and (x_i,y_i,l_i) denote the coordinates of block b_i. The 3D floorplanning problem is to find a solution S for the assignment of blocks coordinates (x_i,y_i,l_i) so that no two blocks overlap and a cost function $C(S)$ is minimized.

2.2. Cost function

A desired 3D partitioning is one that will reduce the footprint (area of the largest tier), the wirelength (or wireload), the economic cost and that will also respect the temperature budget. Economic cost is out of the scope of this paper, while the thermal aspect will be discussed in detail in Section IV.

The cost function of an optimization problem must reflect the needs of a designer. The most common way to take into account all the performance tradeoffs is the weighted sum method. In our work we evaluate the cost of a solution S with the formula:

$$C(S) = \alpha * Footprint + \beta * SumArea + \gamma * WL \quad (1)$$

where α, β and γ are weight parameters; *Footprint* is the product of the largest width and the largest height; *SumArea* is the sum of areas of all layers and *WL* is the wirelength.

To the best of our knowledge, no previous work has employed the *SumArea* objective, or any equivalent form. The rationale behind the *SumArea* goal is that even if there is no evolution in *Footprint*, solutions with more compact arrangements per layer are still preferred. This rationale holds for die-to-die (D2D) and die-to-wafer (D2W) integration, where dies do not have the constraint of equal footprint as in wafer-to-wafer (W2W).

The normalization of each goal is as follows:

$$Footprint_{Norm} = Footprint_{New}/(Area_{Blocks}/n) \quad (2)$$

$$SumArea_{Norm} = SumArea_{New}/Area_{Blocks} \quad (3)$$

$$WL_{Norm} = WL_{New}/WL_{Old} \quad (4)$$

SumArea and *Footprint* goals are normalized by absolute values. *WL* has to be normalized by a relative value since we do

not know the minimal *WL* that can be achieved and at what area expense.

Some authors have also proposed the minimization of the number of TSVs by adding a factor ξ**nbvias* to the cost function [14,15]. These approaches only minimize the via count without considering the TSV dimensions. It is however clear that the relative size of the TSVs is the factor that will limit their number: for a given design, one can insert a larger number of high density TSVs than medium density TSVs. Therefore, in contrast with the previous approaches, we propose to account for the effects of these interconnects directly in the three objectives of (1). The height of the TSVs is added to the total wirelength, and blocks are surrounded by a guard ring to account for the TSV area overhead.

When formulating a cost function using a weighted sum, the most important and difficult task is to determine the appropriate weights that will guide the optimization process in order to find a solution reflecting the expected tradeoff. The relative importance of the objectives is not generally known until the system's best capabilities are determined and tradeoffs between the objectives are fully understood. As the number of objectives increases, tradeoffs are likely to become complex and less easily quantified. The designer must therefore rely on intuition and ability to express preferences throughout the optimization cycle. The meta-optimization approach removes the uncertainty introduced by this human intervention, by considering the weights of the cost function as optimization variables.

2.3. 3D–SP representation

The sequence pair (SP) is a general floorplan representation proposed by [17]. It consists of two permutation lists of n elements, where n is the number of blocks. Topological information is encoded by the order in which the blocks appear in the two lists, for example;

$(\langle \ldots, b_i, \ldots, b_j, \ldots \rangle, \langle \ldots, b_i, \ldots, b_j, \ldots \rangle) \rightarrow b_i$ is left of b_j
$(\langle \ldots, b_i, \ldots, b_j, \ldots \rangle, \langle \ldots, b_j, \ldots, b_i, \ldots \rangle) \rightarrow b_i$ is above b_j

To represent 3D floorplans, one SP is used for each layer. The blocks in a layer's SP are placed on that layer using the SP packing method to compute their (x), (y) coordinates.

In [18] the authors compare SP to Transitive Closure Graph (TCG) and B^*-Tree, which are other popular representations. SP and TCG capture the same set of floorplans and have a redundant design space in $(n!)^2$. Their original packing method takes $O(n^2)$, but SP is easier to implement. B^*-Tree can be evaluated in amortized $O(1)$, but captures only compacted floorplans (i.e.,: a floorplan where no block can move without overlap or change in the outline): a property that can exclude some (or all) of the interconnect-optimal packings. Also, the redundancy in the SP/TCG design space, often seen as a limitation, can in fact make local search more successful by increasing the number of paths to the global optima.

2.4. Simulated annealing framework

Two factors influence the convergence of SA: (i) how it moves from one solution to another and (ii) the rate at which the possibility of an uphill move decreases. Here we discuss the important points related to the second factor and identify the free parameters.

Cooling schedule: The cooling schedule is the manner in which the annealing temperature T is lowered during annealing. The authors in [19] discuss different cooling schedules and show that Aart's schedule is very regular concerning the number of iterations it requires before convergence. We have thus selected this schedule to be implemented in our algorithm. The temperature update function is written as:

$$T_{k+1} = \frac{T_k}{(1+T_k \log(1+\lambda)/3\sigma_k)} \tag{5}$$

where σ_k is the standard deviation of the cost values observed during the kth loop of the algorithm, and λ is a tuning parameter. The number of iterations at each temperature floor (where the temperature is held constant during a given algorithm loop) is directly related to the cooling schedule, and must be explicitly defined, as we discuss next.

Moves per temperature floor (*innerIter*): Using a large number of moves at the same temperature floor does not help convergence, while using a low number can present a noisy σ_k and affect the cooling schedule. In our implementation, *innerIter* is linked to the number of blocks (*nBlocks*) by the parameter k:

$$innerIter = k * nBlocks. \tag{6}$$

Initial temperature: a suitable initial temperature T_0 is one that results in a high probability χ_0 of accepting solutions that increase $C(S)$, allowing a large exploration of the neighborhood of S_0 in the early iterations. Some authors suggest this probability to be around 0.8 [19]. It is clear that T_0 will depend on the scale of $C(S)$ and, hence, be tied to the magnitude of the weights in the cost function and to the circuit under floorplan. To eliminate this dependency, it is possible to estimate T_0 in a first approach. We do this by accepting all solutions that increase C and calculating the average cost increase ΔC^+_{Avg}. Then T_0 is given by:

$$T_0 = -\Delta C^+_{Avg} / \ln(\chi_0) \tag{7}$$

In our implementation, ΔC^+_{Avg} is measured during the first *innerIter* moves.

In the described implementation, only two parameters control the annealing schedule: the λ parameter in the Aart's schedule and the k multiplier that defines the number of moves performed at each temperature floor.

2.5. Solution perturbation

The core of a floorplanning algorithm is the manner in which it modifies the solution and improves it over time. The simplest and most robust way to perturb the solution is to randomly permute the blocks position. However this can be time consuming. Therefore efficient algorithms make use of more sophisticated moves to guide the search.

In [20], the authors present a series of heuristic moves that can be applied to 2D SP to achieve good results for area and wirelength minimization with limited runtime. Among the important contributions are (i) *WL* minimization moves and (ii) the notion of slack in a floorplan:

A *WL* minimization move (Fig. 2(a)) corresponds to moving a block b_i close to the "ideal" location that would minimize the

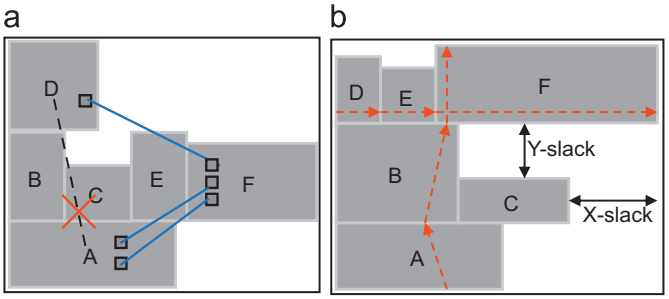

Fig. 2. The notions of *WL* minimization move and slack demonstrated on a 2D floorplan example. (a) The cross indicates the ideal location of block F. (b) The vertical and horizontal critical paths and the slack measure for block C.

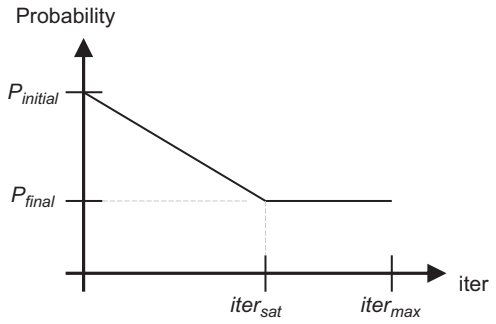

Fig. 3. Probability of trying a 3D move action as a function of the number of iterations.

wirelength of its incident nets. This "ideal" location (x_a,y_a) is simply the average of the position of all modules connected to b_i weighted by the net degree (number of connected pins). Once a block b_j, sitting close to (x_a,y_a) is identified, b_i is moved next to b_j in the SP.

The notion of slack in a floorplan is analog to that in Static Timing Analysis (STA). It is the distance that a block can be moved in a given direction without changing the floorplan outline. As in STA, there is also the notion of a critical path, which constrains the floorplan dimensions (Fig. 2(b)). A slack move then consists of removing a block in the critical path and inserting it in a position with large slack.

Our implementation is based on Parquet-4.5 [20]. We have extended its original SP representation to 3D and defined new move actions using the notions above. We can perturb the solution in the following ways:

- 3D move action set:
  1 Swap blocks from two layers
  2 Move block to another layer
  3 Slack move between two layers
  4 *WL* minimization move between two layers.
- 2D move action set:
  5 Random permutation of SP
  6 Change orientation
  7 Same as (3), but on a single layer
  8 Same as (4), but on a single layer
  9 Same as (7), but with orientation change to better fill the slack.

The difficulty here is how to define with what probability each move action should be applied. Moreover, it is known that 3D moves cause the greatest perturbations in the floorplan and, when using the SA heuristic, they are increasingly unlikely to be accepted as the algorithm progresses [15]. Therefore we reduce the probability of calling the set of 3D move actions with a Piecewise Linear (PWL) function of the maximum number of moves $Iter_{\max}$ (a stop criteria of SA), as depicted in Fig. 3.

Finding a good distribution of the move actions in the 2D and 3D sets (five and four variables, respectively) and adjusting the shape of the PWL function (three variables) plays a major role in the efficiency and robustness of the optimization, and is consequently part of our meta-optimization problem.

3. Meta-optimization based floorplanner tuning

The previous section identified the tuning parameters of each constituent block of our SA floorplanner algorithm. The task of assigning values to these parameters has been formulated as a second optimization problem, to which we have applied a genetic algorithm (GA). This second problem is solved prior to the release of the algorithm to the end user. It searches a set of parameters that are able to minimize the footprint and wirelength of a reduced, but representative, set of floorplan problems (training problems). This method gives a means to automate the floorplanner algorithm tuning, thus avoiding a tedious and knowledge based process. In this section we recall the features of GA and then detail how the optimization problem was set up.

Table 1
Meta-optimization variables.

Algorithm structure	No of variables	Optimization phase
SA schedule	2	ϕ_1
C(F) weights	3	ϕ_1
2D moves selection	5	ϕ_1
3D moves selection	4	ϕ_2
Porbability 2D/3D	3	ϕ_2

3.1. Multi-objective GA and Pareto optimality

GA is a stochastic global search method that mimics natural evolution. It acts over a population of potential solutions, applying intensification (crossover) and diversification (mutation) operators to explore the problem space. The fittest individuals are selected and are used to generate a new population, in the hope of improving the solution quality.

Since GA manages a population, it accepts the cost function to be a vector of objectives, instead of a weighted sum. The GA will then return a set of Pareto-optimal solutions (Pareto front), instead of a single solution. A solution S is Pareto-optimal if there exists no other S' that is superior to S in terms of all objectives. As opposed to the weighted sum formulation, the vectorized form does not require weighting or normalizing of the objectives. Therefore, no prior knowledge about the problem is required.

3.2. Problem setup

The 3D floorplan problem is an extension of the 2D problem. A floorplanner must already demonstrate good performance in 2D cases to ensure good results with 3D cases. Therefore, we have divided the meta-optimization into two phases. The first phase of the optimization only takes into account 2D related parameters (Table 1) and executes the training problems for the planar case. The optimization continues in the second phase running the training problems for a 3D case and acting on the remaining variables. In this approach, the first step guarantees a good usage of the available intra-layer moves (2D), while the second one adapts the frequency and the manner in which blocks are exchanged between layers. The number of variables in each phase is 10 and 7, respectively.

The proposed meta-optimization approach provides a good way to automate the simulated annealing algorithm setting, involved in almost all the floorplanners. Moreover, splitting the meta-optimization process into multiple steps allows the number of optimization variables to be limited (to keep the problem tractable) and enables the incremental addition of more features: 2D and 3D floorplanning and, as will be discussed in the next sections, thermal-aware features.

4. Temperature minimization background

In this section, we recall the main concepts involved in a thermal aware floorplanner and highlight some practical issues.

4.1. Understanding the tradeoffs

Commonly, thermal-aware floorplanners extend the area–wirelength minimization problem presented earlier by adding the maximal temperature (T_{max}) in the weighted cost function[1]:

$$C(S) = \alpha * Area + \beta * WL + \chi * T_{max} \quad (8)$$

In a first approach, it is important to understand the relations between these objectives. It is known that the solution with optimal wirelength usually does not correspond to the solution with optimal area. But, in a general way, wirelength benefits from area reduction. On the other hand, temperature and area are completely opposite objectives: compact floorplans present high temperature while sparser arrangements are cooler. Therefore, a motivation of this work is to propose solutions to the difficult task of finding a balance between these two concurrent objectives.

4.2. The choice of a thermal model

During the floorplanning process, millions of iterations are needed until the search converges to thermally efficient solutions. With such a high number of solutions to evaluate, it is not feasible to run a detailed finite element simulation to evaluate the thermal profile of each one. Several authors have thus proposed the use of simplified thermal models that are suitable for use in a floorplanning algorithm. We have identified two models that represent different degrees of accuracy. Both rely on the thermal–electrical analogy and use cubes to mesh the chip volume. The temperature values are obtained solving the linear system $T = P * Rth$, where Rth is the thermal resistivity matrix and P is the power vector.

Among the identified models, HotSpot [21] is the most refined. It solves the linear system with an iterative multi-grid method, starting with a coarse mesh and then successively refining the solution. HotSpot also models the heat flow from the chip to the heat sink and the board.

A faster alternative was used in [15]. Instead of solving the complete linear system, the lateral heat flow is neglected and tile stacks are analyzed individually (Fig. 4). Because there is no interaction between tile stacks, this approach is less accurate and can produce a noisy thermal profile. Nevertheless, in [16] it is shown that the correlation between this model and HotSpot is 0.82, making it a reasonable choice for floorplanning. In our implementation, we use both approaches, neglecting lateral heat flow during the optimization and using Hotspot with a fine mesh to evaluate the final solution.

4.3. The impact of the thermal profile on the search algorithm

Floorplanning algorithms are usually initialized randomly. Random initializations generally produce disorganized (sparse) floorplans. This largely favors the T_{max} objective and can impede the search algorithm to move to solutions of smaller area and wirelength.

Moreover, the thermal conductivity of the bonding interface material (epoxy, 0.05 W/mK) is much lower than that of silicon (150 W/mK) and copper (285 W/mK). This large difference creates a barrier to heat flow, leading to significant temperature increases at each bonding interface. Consequently, it disturbs the search and impedes the efficient use of the upper layers. As an example, in [14], [15], where the formulation (8) has been used, the temperature reduction comes at the expense of area increase, of the order of 16%. An alternative to this problem is proposed in [16], using a two-phase algorithm. As described in the next section, this approach has been adapted in our implementation with different characteristics to improve its efficiency.

[1] For the sake of clarity, the terms *footprint* and *sumArea* are noted as a single term *Area*.

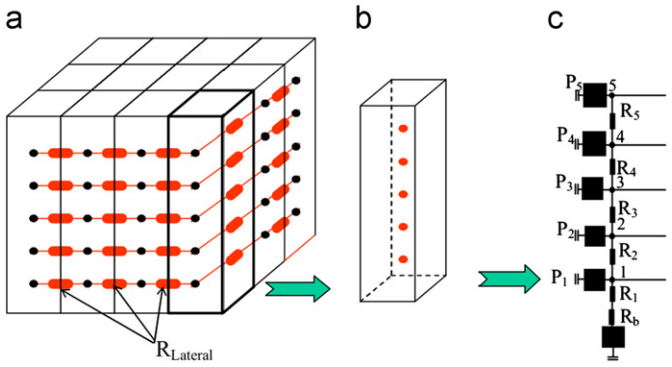

Fig. 4. Simplified thermal model [15]. (a) Tiles Stack Array, (b) Single Tile Stack and (c) Tile Stack Analysis.

5. Proposed methodology for thermal floorplaning

In Section IV we have shown that using temperature measurements on sparse floorplans can disturb the search. Thus, explicit temperature minimization should be left to a stage where the area is sufficiently compact. In this section, we present a two-phase algorithm, where temperature is minimized implicitly during the first phase and then explicitly during the second. We also define the criteria based on area to switch from phase one to two and a heuristic that will help the search algorithm converge to cooler solutions.

Our two-phase algorithm differs from that in [16] in the following aspects: first, we switch from phase one to phase two without restarting the annealing schedule. Second, we are able to reduce temperature in the first phase. Third, our cost function for phase two keeps all the objectives of phase one, in contrast with [16] in which the wirelength objective is not considered when optimizing temperature.

5.1. Phase 1 – a thermally efficient power density distribution

From a 1D approximation of the vertical heat flow on the chip (Fig. 5(a)) it can be noticed that a power distribution with a pyramidal shape (Fig. 5(b)) will implicitly reduce peak temperature.

Therefore, during the first phase, we seek to arrange the blocks in n layers so that the power density is maximized and the more power-hungry blocks are placed closer to the heat sink. For this purpose the cost function is written as:

$$C(S) = \alpha * Area + \beta * WL + \delta * (1/P_{Dens}) \quad (9)$$

where P_{Dens} is a weighted sum of the power density of each layer:

$$P_{Dens} = \sum_{i=1}^{n} \frac{P_i}{Area} \cdot q_i, \quad (10)$$

with

$$q_i = R_1 / \sum_{j=1}^{i} R_j \quad (11)$$

and the weighting factors q_i decrease for higher layers (away from the heat sink). In such a formulation, the term P_{Dens} is maximized both when the area is reduced and when the power-hungry blocks are moved to the lower layers. Hence, it provides a means

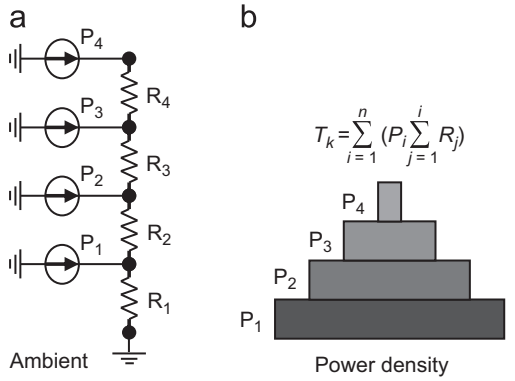

Fig. 5. (a) Vertical heat flow model (b) profile of a suitable power distribution that minimizes temperature.

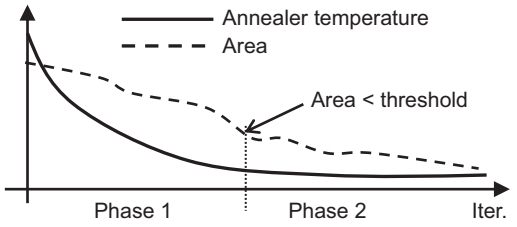

Fig. 6. The proposed two-phase algorithm and the switching criteria.

to combine the opposite objectives of area and temperature reduction in the same direction.

5.2. Phase 2 – minimize thermal gradients

The second phase involves thermal simulations and starts once the area value reaches a threshold λ that ensures that all layers are occupied. This threshold must be at least $\lambda \leq A_{2D}/(n-1)$, where A_{2D} is the area of the design in a 2D configuration and n is the number of available layers (Fig. 6). With this condition, we avoid the problems mentioned in Section IV.C.

During the second phase, the cost function (9) is augmented with a temperature term:

$$C(S) = \alpha * Area + \beta * WL + \delta * (1/P_{\mathrm{Dens}}) + \chi * \Sigma T_{\max} \quad (12)$$

Instead of minimizing the maximal temperature, our approach minimizes the sum of maximal temperatures on each layer. This formulation is motivated by the observation that the x–y coordinates of the hottest spot in each layer does not always coincide, as exemplified in Fig. 7. When these situations occur, our formulation is able to reward the improvements in the thermal profile of each internal layer (i.e., gradient reduction). Furthermore we note that the temperature in the top layers is naturally higher than that in the lower layers and, consequently, more reward is given to improvements at the top-most layer.

5.3. The power density heuristic

Based on the observation presented in Section V.A, we propose a heuristic consisting of biasing all the inter-layer moves in the direction of attaining a pyramidal power distribution. Therefore, a block b_i in layer l_k is only allowed to move to a lower (resp. upper) layer l_p if it meets the condition of having a power density greater (resp. lesser) than the average power density of the blocks in l_k. When performing a 3D move, we randomly select a block in l_k until this condition is satisfied.

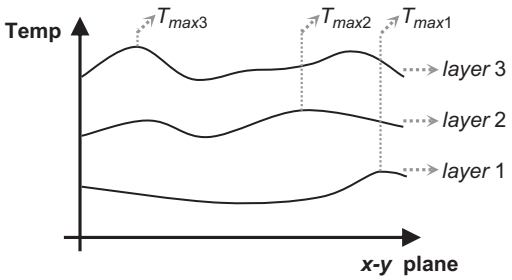

Fig. 7. Thermal profile of each layer along a cross section of the x–y plane.

6. Experimental results

The results are presented in two sets. The first set demonstrates the relevance of the meta-optimization approach. We show how, using a GA optimization, we can set the floorplanner parameters with limited human intervention and obtain results equivalent to or better than the state of the art. The optimized floorplanner is compared to previous works in the context of the area–wirelength minimization problem, for which common benchmarks exist. Additionally we demonstrate the advantages of taking into account TSV dimensions. We also complete the problem formulation with an *a priori* insertion of TSVs to improve 3D via awareness during floorplanning.

In the second set of results, the emphasis is on the thermal aspect. We extend our floorplanner to implement the propositions in Section V. We present a complete analysis comparing the performance of the two-phase algorithm, the power density heuristic and the combination of both.

The floorplanner is implemented in C++ and compiled with g++ 4.5 on Linux. The meta-optimization, as well as all validation tests, were performed on an Intel Xeon machine (4 CPU, 8 GB RAM, 2.4 GHz). MCNC and GSRC benchmarks [24] are used for comparison.

6.1. Meta optimization experimental results

6.1.1. Meta-optimization settings

Two benchmarks with different characteristics have been used as reference problems for the meta-optimization: *n*100 (GSRC) and *ami*49 (MCNC). In the first, the blocks have very similar sizes, while in the second, they cover a wide range of dimensions. The parameters of the floorplanner were explored using the multi-objective GA implemented in Matlab [22]. The GA was arbitrarily configured for 32 individuals and 60 generations and the population was initialized randomly. The fitness of the each individual is composed by the four objectives to be minimized, i.e., area and wirelength of *n*100 and *ami*49. This fitness is measured as the average of eight floorplan runs so as to mitigate the influence of the starting point. The evaluation step is parallelized with a shell script, speeding up the resolution by four (number of CPUs).

Fig. 8 shows the obtained four-dimensional Pareto front. Two of the four axes are inverted to render the plot more intuitive. From this plot we have selected a solution that can simultaneously provide a good tradeoff for both benchmark problems (outlined by the dashed rectangle). This solution becomes the default parameterization for our floorplanner (Fig. 9).

6.1.2. Validation with benchmarks

The meta-optimization approach is validated comparing the solution quality of our floorplanner to the state of the art. All results for this work are averaged over 100 runs. Table 2

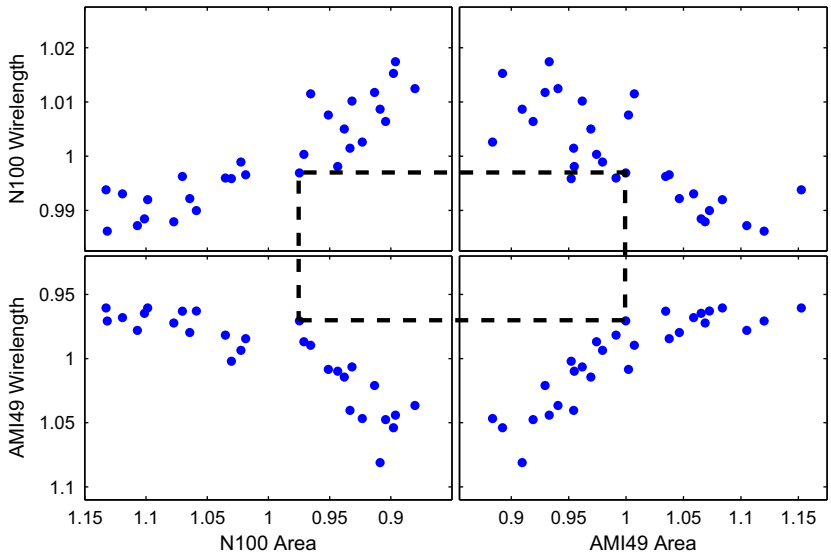

Fig. 8. Four-Dimensional Pareto front obtained with GA. Values are normalized by the average of each objective. The dashed rectangle indicates the selected tradeoff.

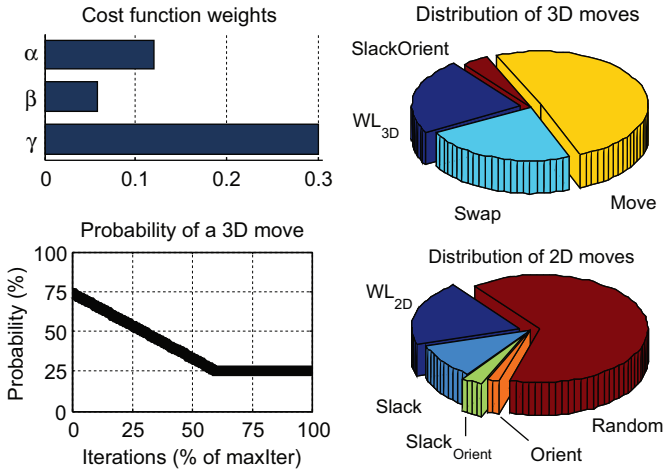

Fig. 9. Visual representation of an optimized parameter set.

Table 2
Performance comparison for 2D test cases.

Circuit	No. Iter.	**Parquet-4.5 (SP)**		**This work**	
	($\times 10^3$)	Area (WS) (mm^2)	Wire (μm)	Area (WS) (mm^2)	Wire (μm)
*ami*33	66	1.32 (14.1%)	59,983	1.31 (13.4%)	53,802
*ami*49	98	40.6 (14.5%)	800,884	40.89 (15.4%)	689,881
*n*100	200	0.197 (9.75%)	234,346	0.196 (9.14%)	233,666
*n*200	400	0.195 (11.0%)	441,127	0.193 (9.46%)	442,237
*n*300	600	0.304 (11.3%)	623,822	0.300 (9.94%)	621,030

compares its 2D performance to Parquet for an equal number of iterations.[2] Our implementation achieves similar wirelength and a slight reduction in white space (WS). The proposed meta-optimization proved to be successful in parameterizing the floorplanner. The performance in a 3D scenario is compared to [14] and [15] for a four layer stack. To enable comparison with these works, the objective of minimizing the number of vias is introduced in our cost function and its weighting factor is manually set to obtain similar via count. Table 3 shows that our implementation significantly reduces the white space compared to both CBA and 3D–STAF. The wirelength is, in average, 5% longer than 3D–STAF, but still 8% smaller than CBA. We underline however that, for a four layer stack, any solution with white space larger than 30% presents little practical interest, since the same area could be achieved with only three layers. Therefore, the ability of our floorplanner to find the most compact solutions, regardless of the number of blocks in the benchmark, demonstrates the power of this approach.

6.1.3. *Accounting for TSV dimensions*

In this section we show the improvement that can be achieved if TSV dimensions are handled directly in the cost function. Here we consider the two following approaches:

A posteriori insertion: The floorplan is obtained minimizing the TSV count (as in Table 3). The solution is then modified to reserve area for the TSVs and their height is added to the total wirelength. Fig. 10(a) and (b) illustrate this approach.

A priori insertion: The cost function includes the TSV contribution to area and wirelength during the entire optimization process. Fig. 10(c) shows that this approach can make better usage of the available area across the layers.

To compare these approaches, a stack with all four layers in face-to-back (F2B) orientation and terminals on top is considered. Dimensions for TSVs are taken from ITRS projections [23] and two cases are considered: for smaller benchmarks (*n*100, *n*200, *n*300) TSVs have pitch of $4 \times 4\ \mu m^2$ and height of 10 μm, dimensions compatible with W2W bonding. For *ami*33 and *ami*49, larger TSVs compatible with D2D and D2W are used: pitch $8 \times 8\ \mu m^2$ and height 20 μm.

Results are presented in Table 4.[3] For each metric the first column is for *a posteriori* via insertion and the second for the proposed approach. The wirelength metric includes the TSV

[2] Parquet is run using Sequence Pair and default weights. Terminals are placed at the boundary of the chip as defined in the benchmarks available at http://vlsicad.eecs.umich.edu/BK. Results for Parquet are also averaged over 100 runs.

[3] The first column of each metric corresponds to adding the TSV overhead to the results from table 3, while the second corresponds to the proposed approach.

Table 3
Footprint, wirelength and 3D-vias minimization for four layer stack.

Circuit	**This work**				**3D–STAF** [14]				**CBA** [15]			
	Area (WS) (mm^2)	Wire (mm)	Vias (count)	Time (s)	Area (WS) (mm^2)	Wire (mm)	Vias (count)	Time (s)	Area (WS) (mm^2)	Wire (mm)	Vias (count)	Time (s)
*ami*33	0.364 (25.9%)	26.1	116	2	0.379 (31.1%)	22	122	52	0.353 (22.1%)	22.5	93	23
*ami*49	11.48 (29.6%)	427.2	194	5	13.49 (52.2%)	437.5	227	57	14.90 (68.2%)	446.8	179	86
*n*100	0.050 (11.5%)	92.7	884	20	0.059 (31.5%)	91.3	828	68	0.053 (17.9%)	100.5	955	313
*n*200	0.048 (10.1%)	173.9	1810	84	0.059 (34.3%)	168.6	1729	397	0.058 (31.4%)	210.3	2093	1994
*n*300	0.075 (10.3%)	257.2	1914	160	0.097 (42.0%)	237.9	1554	392	0.089 (30.3%)	315	2326	3480

To allow comparison, terminals are placed at the center of the top-most layer. Results for this work are averaged over 100 independent runs. The data for CBA and 3D-STAF is presented as in the original publications for the area–wirelength minimization problem. The runtime data is not directly comparable.

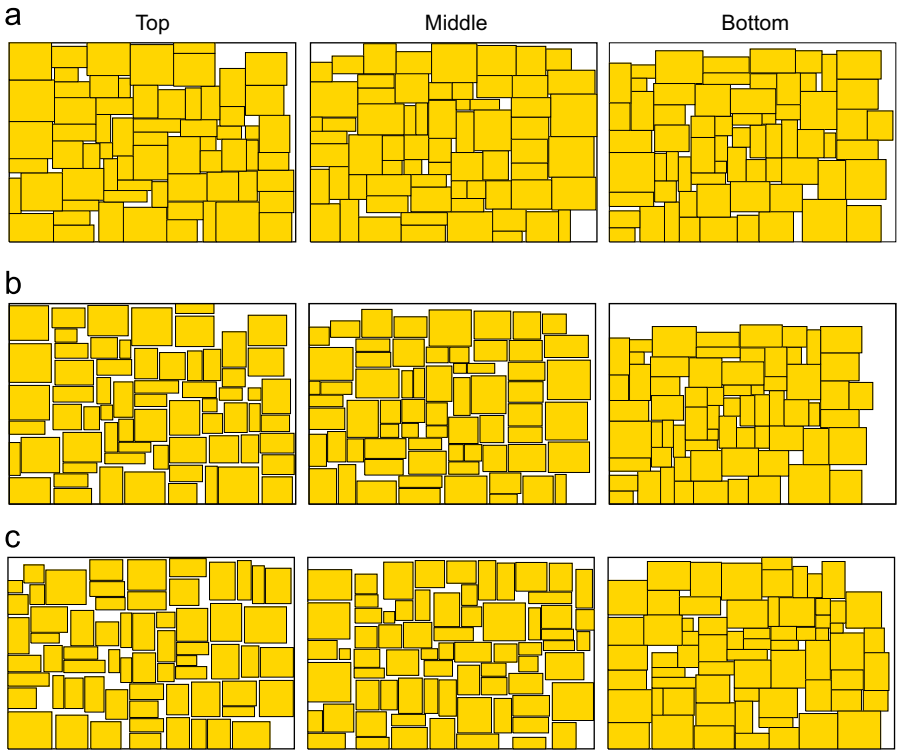

Fig. 10. (a) *Footprint* = 65,475 μm^2, *WL* = 195,782 μm, *vias* = 1218. (b) *Footprint* = 76,012 μm^2, *WL* = 219,907 μm, *vias* = 1218. (c) *Footprint* = 73,647 μm^2, *WL* = 215,570 μm, *vias* = 1244. Three-layer floorplan results for *n*300 with various strategies. (a) Minimizing via count and then (b) reserving area for vias or (c) accounting for via dimensions during the whole optimization process. Here solution (c) uses more vias but has about 3% less area and wirelength than (b).

Table 4
Results considering TSV dimensions.

Circuit	Area		Wire + TSV		Vias	
*ami*33	0.367	0.37	28.6	28.3	116	137
*ami*49	11.49	11.55	431.6	328.9	194	448
*n*100	0.056	0.054	105.5	104.5	884	966
*n*200	0.06	0.057	207.5	208.6	1810	1856
*n*300	0.088	0.086	292.8	289.5	1914	2100

The first column of each metric corresponds to adding the TSV overhead to the results from Table 3, while the second corresponds to the proposed approach.

height. As can be seen, the *a priori* approach, in most cases, reduces both area and wirelength, while TSVs can be added (or removed) as long as they contribute to the improvement of the floorplan. It is thus more relevant to take into account for the impact of TSV dimensions on area and wirelenght during the whole floorplanning process rather than to explicitly minimize the total via count.

6.2. Thermal-aware floorplanning

The thermal aware floorplanner uses the cost functions described in Eqs. (9) and (12), The additional weights δ and χ relative to the temperature effects were determined applying an incremental meta-optimization based on the result from Section VI.A.1.

Reported results are an average taken from over 20 independent runs. Dimensions and material properties for the stack are taken from [24]. The MCNC and GSRC benchmarks are annotated with power values available at [25]. In these benchmarks, the power density of the blocks varies from 10 to 10^3 (W/cm^2). During floorplanning we evaluate temperature neglecting the lateral heat flow and using a grid size that is close to

the dimensions of the smallest block. The best reported solution is evaluated with a finer resolution (smaller grid size) using HotSpot with its default heat sink model. The obtained temperature values are tightly related to the stack properties (materials and dimensions) and to the power values assigned to the blocks. Since related works [14–16] do not detail these aspects, direct comparison is not possible.

Table 5 presents the temperature for results obtained considering only area and wirelength minimization. They serve as a reference to quantify the impact of our proposed methodology. In comparison to Table 3, here we have let the algorithm run for a longer time in order to achieve even more compact floorplans and exacerbate the temperature effect. From this table, one can notice that the maximal gradients for *ami*33 and, more significantly, *ami*49, are very high. This can be explained by the fact that, for these small benchmarks, it is difficult to achieve a solution over four layers with low whitespace. In addition, the packing methods used in most of the floorplanners systematically compact the design towards one corner (lower left), accumulating the whitespace on the opposite corner (upper right) of each layer.

To underline the impact of each of our propositions, we perform three tests (Table 6): (i) the heuristic applied to area–wirelength minimization (weights δ and γ set to zero), (ii) the proposed two-phase algorithm with the proposed cost functions (weights δ and γ determined by meta-optimization) and (iii) the two-phase algorithm combined with the heuristic. Case (i) shows that the simple use of the heuristic can considerably reduce temperature, even though temperature is not among the explicit objectives of the cost function. In case (ii), the explicit goal of temperature reduction improves heat reduction but forces the solution to occupy a larger area. Finally, in case (iii), the combination of both techniques yields better results, further reducing the temperature and gradients with smaller area than that in case (ii). Additionally, the use of the heuristic in case (iii) delayed the transition to the second phase, reducing the number of thermal simulations and leading to shorter runtime. Experimental results have shown that the use of the aforementioned heuristic is very efficient, capable of reducing temperature by -25% and gradients by -47% with little area ($+2\%$) and wirelength ($+5\%$) increase. Its combination with the two-phase algorithm was able to further reduce temperature (-37%) and gradients (-56%), but compromising area with an average 11% increase.

7. Conclusion

The performance of optimization algorithms always depends on its tuning parameters and on the cost function formulation. In this work, we focus on the 3D–IC floorplanning optimization problem. We discuss the algorithm implementation and identify 17 parameters that play a role in its performance. The tedious task of discovering sets of parameters that can drive the floorplanner to good results is automated using a Genetic Algorithm in a meta-optimization loop. The approach is applied to our implementation of a 3D floorplanner, but can be easily adapted to others. We also propose a problem formulation that takes into account the overhead of 3D-vias during the entire floorplan process in an improved way.

Furthermore, two methods to reduce peak temperature and thermal gradients in 3D ICs are discussed. On one hand, we propose to use a smart heuristic that favours high power density close to the heat sink. On the other hand, we use a two-phase simulated annealing process which uses two different cost functions. In the first phase we combine the concurrent objectives of area and temperature minimization in a single term. During the second phase, the cost function formulation is augmented with a temperature term that gives means to distribute blocks to minimize not only the maximal temperature, but also the gradients inside the layers. The results obtained show that combining the smart heuristic with the cost function reduces the average peak temperature and gradient by -37% and -56% respectively, while limiting area increase to 11%, and that of wirelength to 14%. Our approach shows that, with a smarter problem formulation, one can efficiently explore the margins left for temperature reduction during floorplanning. Algorithms that perform thermal via insertion can take advantage of this formulation to further improve the quality of the solution. Moreover, the approach has been formulated in such a way as to render fairly straightforward the inclusion of additional objectives in the floorplanning problem, such as fabrication cost.

Table 5
Reference values from area–wirelength minimization.

Circuit	Power (mW)	Area (mm^2)	Wire (mm)	Temp (°C)	Grad (°C)	Runtime (s)
*ami*33	703.5	0.3553	24.68	243.32	11.16	5
*ami*49	14,910	10.48	386.20	221.89	71.35	13
*n*100	78.25	0.0496	83.64	197.02	1.04	53
*n*200	78.4	0.0478	158.64	206.08	0.72	279
*n*300	130.5	0.0741	229.11	216.63	1.14	670

Table 6
Impact of the proposed techniques on temperature and gradient reduction. Results are relative to Table 5.

Circuit	**Area–wirelength with heuristic**					**Proposed algorithm**					**Proposed algorithm with heuristic**				
	Area	Wire	Temp	Grad	Runtime	Area	Wire	Temp	Grad	Runtime	Area	Wire	Temp	Grad	Runtime
*ami*33	1.04	1.01	0.72	0.79	1.08	1.09	1.08	0.71	0.67	1.67	1.07	1.06	0.63	0.61	1.41
*ami*49	1.03	1.03	0.73	0.49	1.07	1.20	1.15	0.64	0.47	1.53	1.13	1.14	0.62	0.37	1.36
*n*100	1.01	1.06	0.77	0.49	1.09	1.12	1.13	0.67	0.48	2.03	1.12	1.16	0.65	0.41	1.95
*n*200	1.01	1.09	0.76	0.48	1.09	1.15	1.16	0.65	0.42	1.30	1.13	1.18	0.63	0.37	1.25
*n*300	1.01	1.08	0.75	0.42	1.09	1.13	1.12	0.66	0.50	1.46	1.12	1.15	0.63	0.42	1.37
Average	1.02	1.05	0.75	0.53	1.08	1.14	1.13	0.66	0.51	1.60	1.11	1.14	0.63	0.44	1.47

References

[1] J.W. Joyner, R. Venkatesan, P. Zarkesh-Ha, J.A. Davis, J.D. Meindl, Impact of three-dimensional architectures on interconnects in gigascale integration, IEEE Trans. Very Large Scale Integr. VLSI Syst. 9 (6) (2001) 922–928.

[2] A. Rahman, R. Reif, System-level performance evaluation of three-dimensional integrated circuits,, IEEE Trans. Very Large Scale Integr. VLSI Syst. 8 (6) (2000) 671–678.

[3] W. Davis, E. Oh, A. Sule, P. Franzon, Application exploration for 3-D integrated circuits: TCAM, FIFO, and FFT case studies,, IEEE Trans. Very Large Scale Integr. VLSI Syst. 17 (4) (2009) 496–506.

[4] B. Aull, J. Burns, C. Chen, B. Felton, H. Hanson, C. Keast, J. Knecht, A. Loomis, M. Renzi, A. Soares, V. Suntharalingam, K. Warner, D. Wolfson, D. Yost, and D. Young, Laser radar imager based on 3d integration of Geiger-mode avalanche photodiodes with two soi timing circuit layers, in ISSCC, 2006, pp. 1179–1188.

[5] V. Pavlidis, E. Friedman, Interconnect-based design methodologies for three-dimensional integrated circuits, Proc. IEEE 97 (1) (2009) 123–140.

[6] E.J. Marinissen, Testing TSVv-based three-dimensional stacked ICs, in DATE, 2010, pp. 1689–1694.

[7] E.-K. Kim, J. Sung, Yield challenges in wafer stacking technology, Microelectron. Reliab. 48 (2008) 112–1105.

[8] W. Huang, M. Stan, S. Gurumurthi, R. Ribando, and K. Skadron, Interaction of scaling trends in processor architecture and cooling,in SEMI-THERM, feb. 2010, pp. 198–204.

[9] M. Sridhar, A. Raj, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D.Atienza Alonso, 3d-ICE: Fast compact transient thermal modelling for 3D-ICs with inter-tier liquid cooling, in Proceedings of the 2010 International Conference on Computer-Aided Design (ICCAD 2010), vol. 1, no. 1. New York: ACM and IEEE Press, 2010, pp. 1–8.

[10] H. Yu, Y. Shi, L. He, T. Karnik, in: W. Islped, M.R. Nebel, A. Stan, J. Raghunathan, Henkel, D. Marculescu (Eds.), Thermal Via Allocation for 3D ICs Considering Temporally and Spatially Variant Thermal Power, ACM, 2006, pp. 156–161.

[11] B. Goplen, S. Sapatnekar, Placement of thermal vias in 3D ICs using various thermal objectives, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 25 (4) (2006) 692–709.

[12] Y. Ma, D. Chong, C. Wang, and A. Sun, Development of ball grid array packages with improved thermal performance, in Proc. EPTC'05, Vol. 2, dec. 2005, p. 6 pp.

[13] B. Agostini, M. Fabbri, J.E. Park, L. Wojtan, J.R. Thome, et al., State-of-the-art of high heat flux cooling technologies, Heat Transfer Eng. 28 (4) (2007) 258–281.

[14] P. Zhou, Y. Ma, Z. Li, R.P. Dick, L. Shang, H. Zhou, X. Hong, Q. Zhou, 3D-STAF: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits, in: G.G.E. ICCAD, Gielen (Eds.), IEEE, 2007, pp. 590–597.

[15] J. Cong, J. Wei, and Y. Zhang, A thermal-driven floorplanning algorithm for 3D ICs, in ICCAD. IEEE Computer Society/ACM, 2004, pp. 306–313.

[16] L. Xiao, S. Sinha, J. Xu, and E. Young, Fixed-outline thermal-aware 3D floorplanning, in ASP-DAC, Jan. 2010, pp. 561–567.

[17] H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani, VLSI module placement based on rectangle-packing by the sequence-pair, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 15 (12) (1996) 1518–1524.

[18] H.H. Chan, S.N. Adya, and I.L. Markov, Are floorplan representations important in digital design, in In Proc. ISPD'05. ACM, 2005, pp. 129–136.

[19] H. Cohn, M. Fielding, Simulated annealing: searching for an optimal temperature schedule, SIAM J. Optim. (1999) 779–802.

[20] S. Adya, I. Markov, Fixed-outline floorplanning: enabling hierarchical design, IEEE Trans. Very Large Scale Integr. VLSI Syst. 11 (6) (2003) 1120–1135.

[21] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, Temperature-aware microarchitecture, in Proc. 30th International Symposium on Computer Architecture, June 2003, pp. 2–13, http://lava.cs.virginia.edu/HotSpot/documentation.htm.

[22] MATLAB genetic algorithm toolbox – user's guide.

[23] International Technology Roadmap for Semiconductors 2009.

[24] P. Wilkerson, M. Furmanczyk, and M. Turowski, Compact thermal modeling analysis for 3D integrated circuits, 11th International Conference Mixed Design of Integrated Circuits and Systems, June 2004.

[25] [Online]. Available: http://cadlab.cs.ucla.edu/three_d/3dic.html.