

Thermal-Aware Floorplanning Using Genetic Algorithms

W-L. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T. Theocharides, and M. J. Irwin
The Pennsylvania State University, University Park, PA 16802, USA
{whung,yuanxie,vijay,addoquay,theochar,mji}@cse.psu.edu

Abstract

In this work, we present a genetic algorithm based thermal-aware floorplanning framework that aims at reducing hot spots and distributing temperature evenly across a chip while optimizing the traditional design metric, chip area. The floorplanning problem is formulated as a genetic algorithm problem, and a tool called HotSpot is used to calculate floorplanning temperature based on the power dissipation, the physical dimension, and the location of modules. Area and/or temperature optimizations guide the genetic algorithm to generate the final fittest solution. The experimental results using MCNC benchmarks and a face detection chip show that our combined area and thermal optimization technique decreases the peak temperature sufficiently while providing floorplans that are as compact as the traditional area-oriented techniques.

1. Introduction

Floorplanning is an important step in VLSI physical design. It can roughly estimate the layout of a given set of functional blocks and this problem is known to be NP-complete [1]. The primary objective for floorplanning is to minimize the total area required to accommodate all of the functional blocks on a chip. In this paper, a thermal-aware floorplanning algorithm is proposed, which can reduce the hotspot temperature and achieve a thermal balanced design.

Process scaling and aggressive performance improvements have resulted in a dramatic power consumption increase. Power density directly translates into heat; as a result, the temperature in modern high-performance VLSI circuits increases dramatically due to smaller feature size, higher packing density, and rising power consumption. The hotspot in a modern chip might have a temperature of more than 100°C, while the intra-chip temperature differentials can be larger than 10~20°C [17]. *Temperature can have a dramatic impact on circuit performance, power, and reliability: MOS current drive capability decreases approximately 4% for every 10°C temperature increase, and interconnect (Elmore) delay*

increases approximately 5% for every 10°C increase [3], while the leakage current increases exponentially with the temperature increase. Many chip failure mechanisms (e.g., electromigration and stress migration) are significantly accelerated [3], which leads to an overall decrease in reliability. Therefore, it is very important to reduce or eliminate hotspots and have a thermal balanced design.

Power-aware design alone is not able to address the temperature challenge, because the thermal distribution profile depends on not only the power density but also the physical size and location of each functional block [4]. Therefore, even though it is related to the power-aware design area, thermal-aware design itself is a distinct and important research area. Most previous thermal-aware chip design techniques have mainly focused on the microarchitecture-level [4] or standard cell placement [5][6]. In this paper, we present a thermal aware floorplanning framework based on genetic algorithm. To the best of our knowledge, this is the first floorplanning algorithm that takes temperature into account.

The paper is organized as follows. Section 2 reviews previous work related to this research. Section 3 and 4 explains how to estimate temperature and gives a brief introduction on genetic algorithm. Section 5 presents our thermal-aware floorplanning framework based on genetic algorithm power optimization framework based on genetic algorithm. Section 6 presents and discusses experimental results. Section 7 concludes the paper.

2. Related work

Area-optimized floorplanning techniques have been explored for a long time. Both slicing floorplanning [1][7] and non-slicing floorplanning [1][8] methods all performed well at area minimization. Genetic algorithms can also be used for floorplanning and result in good compact floorplan [16]. Nevertheless, area is not the only constraint considered for floorplanning in future complex chip designs. For example, Yong *et al.* [9] presented a unified method to handle multiple constraints (such as pre-place constraint, range constraint, and alignment constraint). However, there exists no floorplanning approach, which considers the thermal impact incurred by extremely high area compaction. In contrast, thermal-aware placement

This work was supported in part by grants from PDG, NSF CAREER Awards 0093085 and GSRC.

methods for standard cell ASIC design have been investigated. For example, Chu and Wong used matrix synthesis problem (MSP) to model the thermal placement problem and three algorithms were proposed to solve it [5]. Tsai and Kang [2] also proposed a method to derive temperature, based on power estimation for standard cell placement. Chen *et al.* proposed a partition-driven thermal placement model [6] for standard cells, making use of multigrid-like approach to simplify the thermal problem at each level of finer granularity, facilitating the inclusion of temperature constraints on the placement. Hung *et al.* [10] proposed a thermal-aware IP placement algorithm for uniform-size Network-on-Chip IP cores. However, none of the above methods can be applied directly to the floorplanning problem because the objects in floorplanning have non-uniform dimension.

3. Temperature estimation

The temperature of the chip depends on the power consumption of each functional block and the relative positions of the functional blocks. Skadron *et al.* [4] proposed a thermal modeling tool called HotSpot, which is easy to use and computationally efficient for modeling thermal effects at the block level. Hotspot provides a simple compact model, where the heat dissipation within each functional block and the heat flow among blocks are accounted for. The basic idea is that, we define the transfer thermal resistance R_{ij} of functional block_{*i*} with respect to block_{*j*} as the temperature rise at block_{*i*} due to one unit of power dissipated at block_{*j*}:

$$R_{ij} = \Delta T_{ij} / \Delta P_j$$

such that we can get a transfer thermal resistance matrix as R^t . For any power distribution on the floorplan, we can calculate each block's temperature by using the following equation:

$$\begin{bmatrix} T_1 \\ T_1 \\ \vdots \\ T_m \end{bmatrix} = \begin{bmatrix} R'_{11} & R'_{12} & \dots & R'_{1m} \\ R'_{21} & R'_{22} & \dots & R'_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ R'_{m1} & R'_{m2} & \dots & R'_{mm} \end{bmatrix} \begin{bmatrix} P_1 \\ P_1 \\ \vdots \\ P_m \end{bmatrix}$$

where P_i is the power consumption of the functional block_{*i*} and T_i is the temperature of the corresponding functional block_{*i*}. The transfer thermal resistance matrix can be obtained from Hotspot, given the floorplan for a set of blocks.

4. Genetic algorithms

Genetic algorithms (GA) [11] are a class of search and optimization methods that mimic the evolutionary principles in natural selection. Figure 1 shows a genetic

algorithm optimization flow.

The solution is usually encoded into a binary string called *chromosome*. Instead of working with a single solution, the search begins with a random set of chromosomes called *initial population*. Each chromosome is assigned a *fitness score* that is directly related to the *objective function* of the optimization problem.

The population of chromosomes is modified to a new *generation* by applying three *operators* similar to natural selection operators – *reproduction*, *crossover* and *mutation*. Reproduction selects good chromosomes based on the fitness function and duplicates them. Crossover picks two chromosomes randomly and some portions of the chromosomes are exchanged with a probability P_c . Finally, mutation operator changes a 1 to a 0 and vice versa with a small mutation probability P_m . A genetic algorithm successively applies these three operators in each generation until a termination criterion is met. It can very effectively search a large solution space while ignoring regions of the space that are not useful. This algorithmic methodology leads to very time-efficient searches. In general, a genetic algorithm has the following steps:

1. Generation of initial population.
2. Fitness function evaluation.
3. Selection of chromosome.
4. Reproduction, Crossover, Mutation operations.

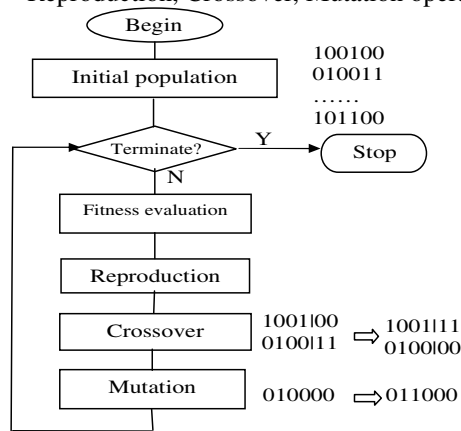


Figure 1. Genetic algorithm flow

5. Thermal-aware floorplanning framework

Our thermal-aware floorplan optimization flow is based on the genetic algorithm and is shown in Figure 2. The representation of a floorplan design is encoded into integer and bit mixed strings called chromosome. The optimization flow begins with a randomly generated *initial population*, which consists of many randomly generated chromosomes (floorplans). The orientation of each block and the polish expression of slicing tree are obtained from individual rotate and slicing strings. The area estimation function and HotSpot are invoked to calculate dead space and

temperature. The fitness is assigned to each population based on the evaluation of these two variables. The optimization flow is an iterative procedure. The chromosomes with better fitness will survive at each generation and the three different operations (reproduction, crossover and mutation) are invoked to derive a new set of chromosomes – or new floorplans. The iteration continues until the termination criterion is met.

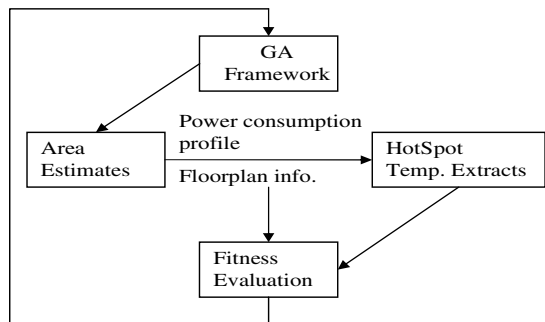
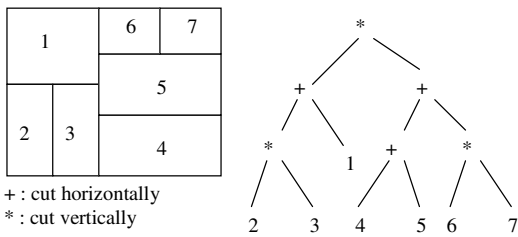


Figure 2. Thermal-aware floorplanning flow

5.1 Chromosome encoding

The representation of a floorplan can be divided into two categories. One is slicing and the other is non-slicing representations. Although slicing floorplan can be sub-optimal, empirical evidence shows that it is still quite efficient in tightly packing modules as evident in the work of Young and Wong [12]. Because of the simplicity and the effectiveness of slicing method [7], we use a slicing tree representation to incorporate with the genetic algorithm. A slicing floorplan is a rectangular area that is sliced recursively by a horizontal or vertical slicing line into a set of rectangular regions to accommodate a set of functional modules. A slicing floorplan can also be represented by Polish expression [7]. Figure 3 shows an example of slicing tree, the polish expression, and its corresponding floorplan. The slicing tree is a binary tree and is bottom-up constructed.



Polish Expression : 2 3 * 1 + 4 5 + 6 7 * + *

Figure 3. Slicing tree representation and its corresponding floorplan

One example of the chromosome encoding is shown in Figure 4. The slicing array contains four unique integers which represent four functional blocks and two slicing operators, “*” and “+”. The “*” operator is used for

vertical cutting while “+” operator represents cutting horizontally. There is another bit array called *rotate* in our encoding which is used to decide whether a block should rotate; that is, switch its width and height. Each integer maps to a fixed block to which it associates with the width and height measurements. The purpose of bit array is to add more flexibility in finding good solutions (minimal area) as the evolution progress. For example, the floorplan as shown on the right in Figure 4 accounts for chromosome A, while chromosome B represents the floorplan on the left. With this representation, the exact location and orientation of each block on a chip can be extracted easily. After obtaining a valid chromosome, the temperature calculation is invoked by using HotSpot with the floorplan information and power consumptions associated with each block. A fitness function is then used to evaluate the suitability of the chromosome, based on different optimization objectives.

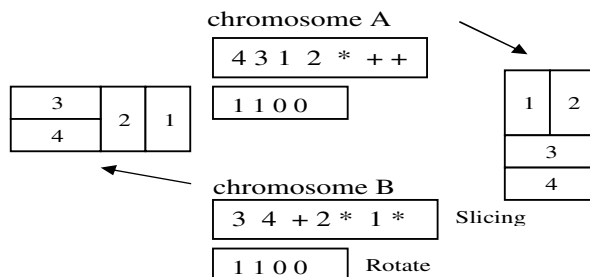


Figure 4. Chromosome encoding for floorplanning

5.2 Fitness function

The fitness function, which decides the surviving chance for a specific chromosome, is related to the mapping goals. Depending on the optimization goal, the fitness function of the genetic algorithm is different. Below we show two different optimization goals.

Area minimization design

This optimization strategy tries to achieve a minimum total chip area to accommodate a set of modules. The fitness of a chromosome is defined in equation (1). The *area(B)* is the total area of blocks and *area(C)* is the total area of the floorplan with respect to the chromosome. The difference of *area(C)* and *area(B)* is the so-called *dead space*. The fitness calculates the proportion of the dead space of a floorplan and normalizes it to the total modules’ area. Higher fitness implies better chance for the chromosome to survive to the next generation. In area-based approach, there is no information about the temperature profile; thus uneven temperature distribution and several hotspots might appear due to packing high power density blocks tightly together.

$$Fitness = 1 / \frac{area(C) - area(B)}{area(B)} \quad (1)$$

Temperature balanced design

The goal of thermal-aware floorplanning is to distribute temperature across a chip evenly and minimize the hotspot temperature, under a specified area constraint. The easiest way is to separate the heat sources as far as possible in order to allow the heat drives away. However, that may result in large dead space and/or total chip area. In order to consider area and temperature constraints together, we first choose those floorplans that satisfy the given dead space ratio by equation (1). For those unsatisfied chromosomes, a penalty is given to make them less unfavorable. The fitness for thermal effect is defined as:

$$Fitness = \frac{1}{(T_{max} - T_{avg}) / T_{max}} - penalty \quad (2)$$

where T_{avg} is the average temperature for all modules and T_{max} represents the maximal temperature produced by this floorplan. The penalty is a large value if the dead space ratio exceeds a user-defined threshold, such that those valid chromosomes with satisfactory dead space ratio and with lower average and maximum temperatures have better fitness to survive.

5.3 Control parameters

While generating the initial population, we have to set an appropriate population size, and the crossover probability P_c , as well as the mutation probability P_m . If the population size is too small, the genetic diversity within the population may not increase for many generations. On the other hand, a large population size increases the computation time for each generation but it may take fewer generations to find the best solution. Schaffer *et al.* [13] had conducted extensive simulation on a wide range of functions and concluded that a small population of size 20 to 30, a crossover probability in the range of 0.75 to 0.95, and a mutation probability in the range of 0.005 to 0.01 perform very well. In our implementation, we set the population size to be 100, crossover probability P_c to be 0.8 and the mutation probability P_m to be 0.02. The termination of the iterative evolution can be user-defined. We set a maximum generation to be 100,000 and specify that if the fitness improvement is less than 0.001% during the last 1000 generations, the evolution stops without going through all generations.

6. Experimental results

To evaluate our genetic algorithm based approach, we implemented the algorithm in C, and the experiment was conducted on a dual Intel Xeon processors (3.2 GHz, 2GB RAM) machine running Linux. We performed experiments on a set of MCNC benchmarks and one real application for face detection. The runtime of 100,000 generations is about 900 seconds for the largest benchmark.

6.1 Experiment on MCNC benchmarks

We first use area optimization to demonstrate that our approach is comparable to non-slicing approaches. Table 1 lists the names of the circuits, the number of modules and the required area for different approaches. From the table, we can see that slicing approach still performed comparable to non-slicing approaches. The required area for ami49 circuit is a little larger than that for other methods while that of the rest of the benchmark circuits are analogous.

To conduct experiment on temperature-balanced optimization, the power values from 0.05 mW to 3 W were randomly assigned to the blocks in different benchmark due to the lack of information on the internals of each block. In addition, a real application with accurate power estimation will be shown in the next sub-section.

Table 1. Comparisons for area requirements among O-tree, iterative and simulated annealing B*-tree, cluster refinement. (Data taken from [2] for non-slicing approaches)

MCNC Benchmarks			Non-Slicing				Slicing
Circuit	# of cells	Cell area (mm ²)	O-tree	B*-tree (Iter.)	B*-tree (SA)	Cluster refinement	Our GA
apte	9	46.56	47.46	46.92	46.92	48.4	47.52
xerox	10	19.35	20.18	20.06	19.83	20.3	20.26
hp	11	8.83	9.49	9.17	8.95	9.58	9.44
ami33	33	1.15	1.25	1.27	1.27	1.21	1.27
ami49	49	35.44	38.6	37.43	36.8	37.7	39.16

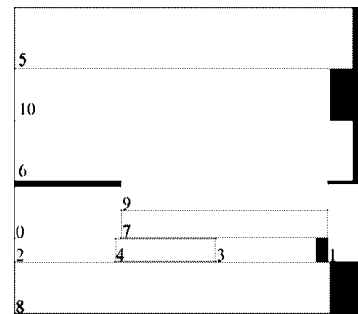


Figure 5. The floorplan of circuit hp with area-optimized only

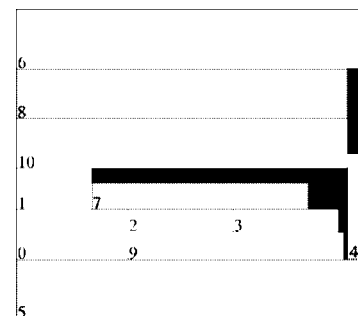


Figure 6. The floorplan of hp with thermal-aware optimization

The area-optimized floorplan of example *hp* is shown in Figure 5. The black areas depict the dead space while the numbered blocks represent different modules. The shaded blocks are the top four modules that had the largest power consumption. To obtain a thermal balanced design, we use the minimum area achieved in the area-optimized approach as a constraint and apply the thermal balanced optimization using fitness function (2). Figure 6 shows the floorplan of the same example circuit *hp*. The hotspot temperature (maximum temperature) drops from 123 °C to 120 °C, and the average temperature drops from 111 °C to 110 °C, while the resulting area is the same. Comparing Figures 5 and 6, we can see that the thermal-aware floorplanning spread out the modules with high power consumption, such that it can acquire lower hotspot temperature and thermal-balanced floorplan.

By relaxing the area constraint, the thermal-aware optimization approach can achieve temperature reduction further for two reasons: First, the heat in hotter regions can flow through more dead areas that consume no power; second, it has more flexibility for allocating high-power consuming modules into less hotter regions. The following results confirm our rationale. The maximal and average temperatures of *ami49* under different dead space ratios are shown in Figure 7. Temperature decreases with gradually increasing dead space ratio.

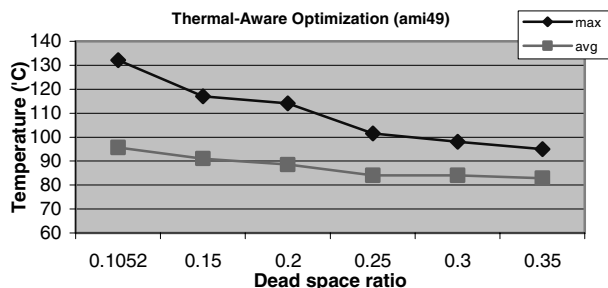


Figure 7. The comparison of temperature and dead area ratio for *ami49* under thermal-aware optimization

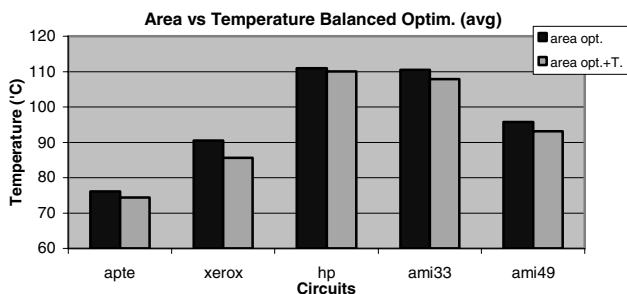


Figure 8. The comparisons of average temperatures for area optimization and temperature-aware optimization

Figure 8 and Figure 9 show the average and maximal temperatures of all five MCNC benchmarks with/without thermal consideration. It shows that the area optimization

results in higher average and maximal temperature for all benchmarks. The temperature can be effectively reduced through thermal-aware optimization combined with area constraint, such that lower temperature is achieved with the same compact floorplan. The overall average temperature is reduced around 1~5 °C, as shown in Figure 8. The hotspot temperature (peak temperature) is reduced by as much as 24 °C in example *apte* in Figure 9. However, some benchmarks could not achieve large temperature reduction. For example, the hot spot temperature reduction for *ami33* is only 1 °C. The reason behind this is that the power density is relatively high for a limited small area and thus there is not much flexibility for algorithm to discover a good solution.

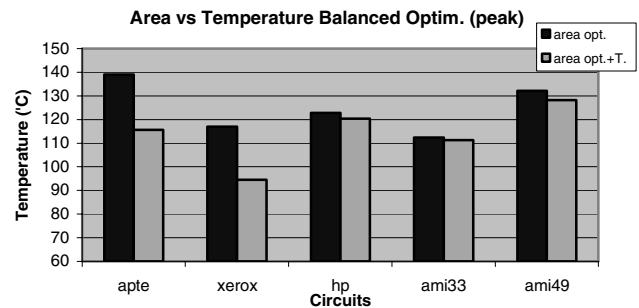


Figure 9. The comparison of maximal temperatures (hot spot) for area optimization and temperature-aware optimization

6.2 Experiment on a face detection application chip

To demonstrate our algorithm, we mapped a neural network used in detecting the angle of rotation on face images. The network is part of an on-chip face detection application [14] and consists of three layers of neurons, input layer neurons, hidden layer neurons and output layer neurons. There are 15 input layer neurons, 15 hidden layer neurons, and 36 output layer neurons. The three types of neurons differ from each other, resulting in different area and power consumption over a single detection.

To obtain the power consumption and dimensions of each neuron type, we first designed and synthesized the neural network on an on-chip network using Verilog HDL and Synopsys Design Compiler. We then simulated the entire network operation using a Network-On-Chip simulator [15] which we used to obtain the network traffic including addressing and control data. We used this network traffic as input to Synopsys Power Compiler, thus simulating the actual switching activity of the neurons, and obtained the average power consumption for a single detection. The overall simulation procedure is shown in Figure 10. Table 2 shows the resulting power consumption and dimension for different neuron blocks.

Table 2. Power consumption and neuron dimension profile of the face detection application chip

	Type A	Type B	Type C
Power (mw)	214.53	52.11	14.17
Dimens. (um ²)	1200x450	500x260	200x400
Number	15	15	36

In Figure 11, the result from applying thermal-aware floorplanning to the face detection application is shown. The area, average and maximal temperatures are normalized. As can be seen from the figure, the maximal temperature reduces sharper than average temperature before 0.2 dead space ratio. After that point, the maximal temperature reduction is small. This experimental result shows that, with minor area penalties, both the hot spot and average temperatures can be effectively reduced.

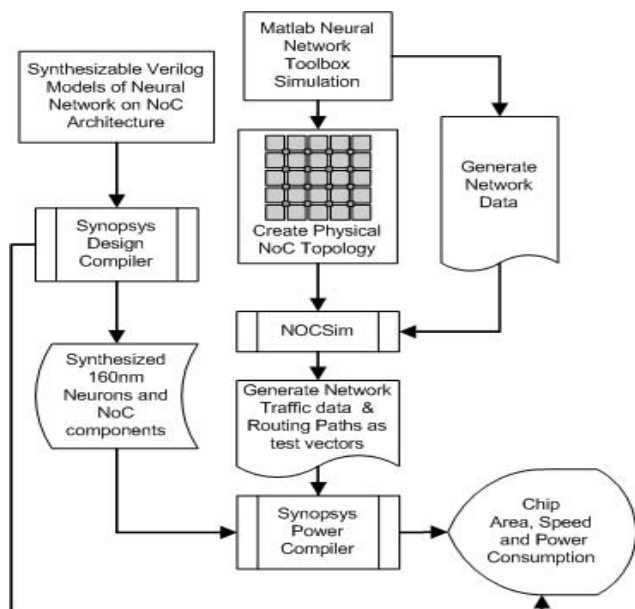


Figure 10. The overall design methodology for face detection application's power estimation

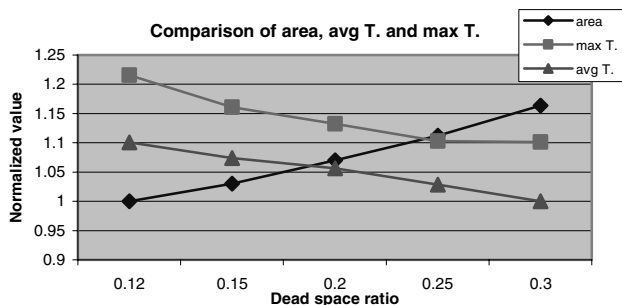


Figure 11. The comparison of area, average and maximal temperatures under thermal-aware optimization

7. Conclusion and future work

In this paper, we present a thermal aware floorplanning algorithm based on Genetic Algorithms. Our experimental results show that our combined area and thermal optimization technique decreases the chip temperatures sufficiently while providing floorplans that are as compact as the traditional area-oriented techniques.

Currently, our genetic algorithm based framework considers blocks' dimensions only; we will extend our work in the future by including interconnections among modules and their related power consumption modeling.

References

- [1] N. A. Sherwani, "Algorithms for VLSI Physical Design Automation", Kluwer Academic Publishers, Norwell, MA, 1999
- [2] C. Tsai and S. Kang, "Stand Cell Placement for Even On-Chip Thermal Distribution", ISPD 1999.
- [3] J.Srinivasan, S.V.Adve, P.Bose, and J.Rivers, "The Impact of Technology Scaling on Lifetime Reliability", Proc. of International Conference on Dependable Systems and Networks, June 2004.
- [4] K. Skadron, T. Abdelzaher, and M. Stan, "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management", HPCA 2002.
- [5] C.N. Chu and D.F. Wong, "Matrix Synthesis Approach to Thermal Placement", Proc. Int. Sym. On Physical Design, pp.163-168, 1997.
- [6] G. Chen and S. Sapatnekar, "Partition-Driven Standard Cell Thermal Placement", ISPD 2003.
- [7] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design", DAC 1986.
- [8] P.-N. Guo, C.-K. Cheng and T. Yoshimura, "An O-tree Representation of Non-Slicing Floorplan and Its Application", DAC 1999.
- [9] E. Yong, C. Chu and M. Ho, "Placement Constraints in Floorplan Design", IEEE Transactions on VLSI, Vol.12, No.7, pp.735-745, 2004
- [10] W. Hung *et al.*, "Thermal-Aware IP Virtualization and Placement for Networks-on-Chip Architecture", Proceedings of IEEE 22nd International Conference on Computer Design, 2004.
- [11] Goldberg, D.E. Genetic Algorithms in Search, Optimization, and Machine Learning, New York: Addison-Wesley, 1989.
- [12] F. Y. Young and D. F. Wong, "How Good Are Slicing Floorplans", Integration VLSI J., pp. 61-73, 1997.
- [13] J.Schaffer *et al.*, "A study of control parameters affecting online performance of genetic algorithms for function optimization," Proc. of international Conference on Genetic Algorithms, pp.51-60, 1989.
- [14] T. Theocharides *et.al.*, "Embedded Hardware Face Detection", in the Proceedings of International Conference on VLSI Design, Mumbai, India, Jan 2004.
- [15] D. Whelihan, "The CMU NOCSim Simulator", <http://www.ece.cmu.edu/~djw2/NOCSim/>
- [16] M. Rebaudengo and M. Reorda, "GALLO: A Genetic Algorithm for Floorplan Area Optimization", IEEE Transaction on CAD, Vol.15, No.8, August 1998.
- [17] <http://www.isonics.com>