Wire Congestion And Thermal Aware Global Placement For 3D VLSI Circuits

Karthik Balakrishnan, Vidit Nanda, Siddharth Easwar, and Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology
{gte245v, gte272u, gte581t, limsk}@ece.gatech.edu

***Abstract*— The recent popularity of 3D IC technology stems from its enhanced performance capabilities and reduced wiring length. However, wire congestion and thermal issues are exacerbated due to the compact nature of these layered technologies. In this paper, we develop techniques to reduce global the temperature gradient and local and global congestions of 3D circuit designs without compromising total intra-layer wirelength or inter-layer via count. Our approach consists of two phases. First, we use a multilevel min-cut based approach with a modified gain function in order to minimize the local wire congestion and power dissipation. Then, we perform simulated annealing with a full-length thermal analysis to reduce the circuit's global congestion and thermal gradient. Experimental results show that when compared to the standard mincut approach, our thermal gradient and local congestion are reduced by 25% each, global congestion is reduced by over 7%. Moreover, we only see a 10% increase in the wiring length and the number of vias required.**

I. Introduction

With the recent advent of three-dimensional Integrated Circuit technologies, there has been a positive impact on the performance and wiring length of these ICs. Typically, the layered placement of transistors in multiple planes (i.e. 2.5D placement) allows for a more compact chip with inherently better performance than one fabricated with traditional 2D placement techniques. However, the stacked nature of these circuits induces and aggravates problems of non-uniform thermal dissipation as well as local and global wire congestion. Simultaneously, it is necessary to minimize the wiring in single layers as well as the interconnect among different layers so as to maintain routability.

Previous work in the area of 2.5D placement [1], [2], [3], [4], [5] has focused on minimizing the intra-layer wirelength and the number of interlayer connections, or vias. The results of [6] indicate an improvement in overall wirelength when implementing the 2.5D layered placement framework instead of equivalent traditional 2D placement. Other work has employed stochastic methods to determine the wirelength distributions, trends in power consumption, and performance capabilities of 2.5D ICs [7]. The conclusions derived from stochastic analysis confirm that 2.5D chips will provide better performance with larger compaction, but do not predict the amount of routing congestion that could be present.

In this paper, we provide a technique to reduce both local and global congestion in a 2.5D chip in order to increase the routability of the chip. We also improve the temperature profile of the circuit using state-of-the-art thermal ADI methodologies. Our approach involves a two-stage refinement procedure: Initially, we use a multilevel min-cut based method to minimize the congestion and power dissipation within confined areas of the chip. This is followed by a simulated annealing-based technique that serves to minimize the amount of congestion created from global wires as well as improve the temperature distribution of the circuit. We show that our congestion and thermal gradient minimization does not have any significant negative impact on the inter-layer wirelength or the number of intra-layer vias.

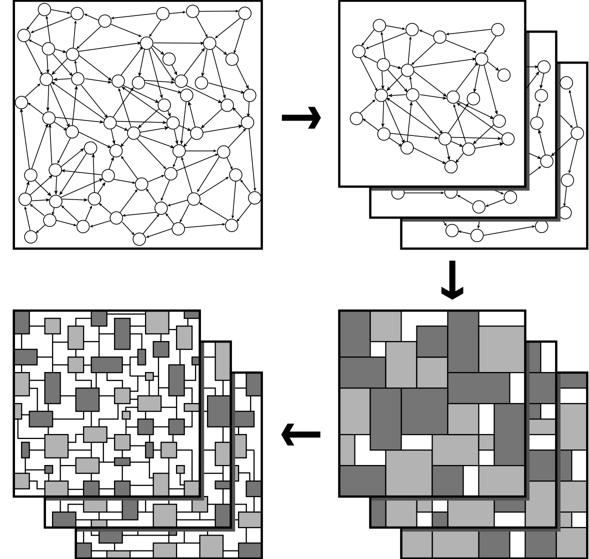


Fig. 1. 3D physical design automation

We provide data to establish that these objectives are pairwise independent in the sense that minimizing the thermal gradient, for example, does not necessarily have a positive impact on total wirelength or wire congestion. Our contributions include a flexible multi-objective optimization technique for 3D VLSI circuits, the incorporation of accurate full-length thermal analysis at the global placement phase in the design process, and a thorough analysis of the correlations among local congestion, global congestion, and thermal quality.

The rest of this paper is organized as follows. Section 2 provides preliminaries of our approach. Section 3 discusses our min-cut based approach to reduce local wire congestion.

Section 4 explains our simulated annealing approach aimed for global wire congestion. Section 5 provides experimental results. Section 6 concludes our paper and describes the ongoing research in this field.

II. Problem Formulation

Given a sequential gate-level netlist $NL(C, N)$, let C denote cells consisting of gates and flip-flops, and N denote nets that connect the cells. The purpose of the 3D Global Placement Problem (3D-GP) is to assign cells in NL to a given $m \times n \times l$ slots in a 3D while satisfying the prescribed area constraints. Given a 3D-GP solution P, our primary objective is to minimize *local/global wire congestion* and *maximum thermal gradient*. As secondary objectives, we minimize *wirelength* and *via* induced by P. The formal definition of the problem is as follows.

Definition 1: **3D Global Placement (3D-GP) Problem** Given a netlist $NL(C, N)$, a set of $m \times n \times l$ $(= K)$ slots $S = \{S_1(x_1, y_1, z_1), S_2(x_2, y_2, z_2), \cdots, S_K(x_K, y_K, z_K)\}$ with $S_i \subset C$, and area constraints A= (L_i, U_i) for $1 \leq i \leq K$ has a solution $P : C \rightarrow S$, where each cell in C is assigned to a unique slot. P is optimal if it satisfies the following conditions; (1) $L_i \leq |S_i| \leq U_i$ for $1 \leq i \leq K$, (2) $S_1 \cup S_2 \cup \cdots \cup S_K = C$, (3) $S_i \cap S_j = \emptyset$ for all $i \neq j$, (4) wire congestion and thermal gradient are minimized.

A. Thermal and Congestion Objectives

Let $T(B)$ denote the temperature of a block B. Then, the *maximum temperature gradient* of the entire 3D-GP solution is defined as follows:

$$TG = \max\{T(B_i) - T(B_j)\},\ 1 \leq i, j \leq K$$

Given a block B from the 3D-GP problem, we define the local wiring congestion cost $LC(B)$ as:

$$LC(B) = \sum_{n \in N, n \cup B \neq \emptyset} |n| - 1$$

This value corresponds to the minimum number of wires required to construct the tree representation of a net of size $|n|$. Then, the local congestion of the entire solution is given by:

$$LC = \max\{LC(B_i) - LC(B_j)\},\ 1 \leq i, j \leq K$$

For any two adjacent blocks B_i, B_j from the 3D-GP solution, H_{ij} denotes a set of all hyperedges having cells in both B_i and B_j. Then, the global congestion at the boundary $[B_i, B_j]$, denoted $GC(B_i, B_j$, is simply $|H_{ij}|$. Then, the global congestion of the entire solution is given by:

$$GC = \max\{GC(B_i, B_j) - GC(B_k, B_l)\}$$

where B_i and B_j are adjacent and B_k and B_k in the given 3D grid.

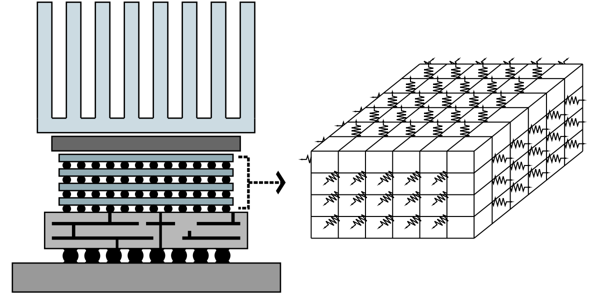


Fig. 2. 3D thermal modeling

B. Wirelength and Via Objective

We model the netlist NL using a hypergraph $H = (V, E_H)$, where the vertex set V represents cells, and the hyperedge set E_H represents nets in NL. Each hyperedge is a non-empty subset of V. The x-span of hyperedge h, denoted h_x, is the maximum distance between any x-coordinates of cells in h. The y-span of hyperedge h, denoted h_y, is the based on the y-coordinates. Then the wirelength of a hyperedge h, denoted w_h is simply $h_x + h_y$. The wirelength of the entire 3D-GP solution is:

$$WL = \sum_{h \in NL} w_h$$

The *via cost* of a hyperedge h is the z-span of h, denoted v_h, is the maximum distance between any z-coordinates of cells in h. Then, the via cost of the entire 3D-GP solution is:

$$VIA = \sum_{h \in NL} v_h$$

III. 3D Thermal Analysis

A high-speed thermal simulator was implemented based on the three dimensional Alternate Direction Implicit Method (3D-ADI) [8]. The simulator has a linear runtime, and memory requirement. The motivation for a thermal driven partitioning was to maximize the efficiency of heat transfer to the environment, and thereby reduce the maximum temperature gradient. A brief explanation of the 3D-ADI method is provided in this section.

The temperature profile of a design is governed by the following partial differential equation for heat conduction:

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

where ρ is the density of the material, C_p is the specific heat of the material, κ is the thermal conductivity of the material, T is the time dependent temperature, and g is the heat generation rate. The solution for this equation is set the by the following boundary condition equation:

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

where h_i is the heat transfer coefficient, and f_i is an arbitrary function on the boundary surface.

In order to solve Equation 1 using the finite difference method, it has to be discretized in the time as well as space domain. Assuming that the material is homogenous, and the thermal conductivity κ independent of temperature, the divergence of the product of thermal conductivity and temperature gradient can be rewritten as:

$$\frac{\partial T(x,y,z,t)}{\partial t} = \alpha\Big[\frac{\partial^2 T(x,y,z,t)}{\partial x^2} + \frac{\partial^2 T(x,y,z,t)}{\partial y^2} + \frac{\partial^2 T(x,y,z,t)}{\partial z^2}\Big] + \frac{1}{\rho c} g(x,y,z,t)$$

where $\alpha = \kappa/\rho C_p$. The second order partial differential terms in the above equation can be approximated according to central finite difference method. For example, the second order partial derivative with respect to x can be approximated as

$$\frac{\partial^2 T}{\partial x^2}\Big|_{i,j,k}^{n} \approx \frac{T_{i+1,j,k}^n - 2T_{i,j,k}^n + T_{i-1,j,k}^n}{(\Delta x)^2} = \frac{\delta_x^2 T^n}{(\Delta x)^2}$$

Following this step, the average of an explicit and implicit update, also known as the Crank-Nicholson method, is applied. After applying these updates to the previous equation, we have

$$\frac{T^{n+1} - T^n}{\Delta t} = \alpha\Big[\frac{\delta_x^2 T^{n+1} + \delta_x^2 T^n}{2(\Delta x)^2} + \frac{\delta_y^2 T^{n+1} + \delta_y^2 T^n}{2(\Delta y)^2} + \frac{\delta_z^2 T^{n+1} + \delta_z^2 T^n}{2(\Delta z)^2}\Big] + \frac{1}{\rho C_p}$$

In order to improve the run time, an Alternate Direction Implicit (ADI) scheme developed by Douglas and Gunn is used. Hence, the equation for x-direction is

$$T^{n+1/3} - T^n = \frac{r_x \delta_x^2}{2}(T^{n+1/3} + T^n) + r_y \delta_y^2 T^n + r_z \delta_z^2 T^n + \frac{\Delta t}{\rho C_p} g$$

the equation for y-direction is

$$T^{n+2/3} - T^n = \frac{r_x \delta_x^2}{2}(T^{n+1/3} + T^n) + \frac{r_y \delta_y^2}{2}(T^{n+2/3} + T^n) + r_z \delta_z^2 T^n + \frac{\Delta t}{\rho C_p} g$$

the equation for z-direction is

$$T^{n+1} - T^n = \frac{r_x \delta_x^2}{2}(T^{n+1/3} + T^n) + \frac{r_y \delta_y^2}{2}(T^{n+2/3} + T^n) + \frac{r_z \delta_z^2}{2}(T^{n+1} + T^n) + \frac{\Delta t}{\rho C_p} g$$

The above equation is then solved iteratively for several iterations until a steady state temperature has been achieved.

The 3D-ADI algorithm is shown in Figure 3. Each time step is split into three equal smaller time steps, with only one direction implicit in each step. Since each of the equations in ADI method can be represented as a tridiagonal system of equations, and hence be solved using the Thomas algorithm, run time is linearly proportional to the number of temperature nodes. Once a steady state temperature profile has been achieved, various physical parameters such as maximum temperture, temperture gradient, and average temperture are calculated.

```
============================================
Algorithm: 3D THERMAL ADI
Input: BN,dx, dy, dz, dt, L, M, N, Tamb
output: Ti,j,k
--------------------------------------------
read physical parameters from tech-file;
read x, y, z dicretization size L, M, N
read x, y, z, and time discretization size:
    dx, dy, dz, dt
read block info BN and build material matrix
build temperature matrix with all temp
nodes assigned to ambient temperature Tamb
for all T(i,j,k)
    compute g, rho, and Cp in the x,y,z dir;
for all T(i,j,k)
    compute rx, ry, and rz in the x,y,z dir;
do
    x-ADI for all i,j,k
    y-ADI for all i,j,k
    z-ADI for all i,j,k
while (T.(n+1)(i,j,k) - T.n(i,j,k) != 0)
return Ti,j,k;
============================================
```

Fig. 3. 3D-ADI thermal simulation algorithm

IV. Local Wire Congestion Minimization

The purpose of this algorithm is to balance the amount of local congestion while maintaining wirelength and via results comparable to those of pure mincut-based techniques. The approach involves modifying the gain function of a multi-level cutsize based partitioner to reduce local congestion.

Our cut sequence is an extension of the two cut sequence techniques used in [6]. Their first method performs via-minimizing interlayer cuts (z cuts) before performing intra-layer cuts (x, y) to minimize the 2D wirelength. Their second cut sequence does the opposite, making all (x, y) cuts first before performing z-cuts to achieve minimal wirelength. For the purposes of maintaining a balanced combination of via count and wirelength during our algorithm, we devise a new cut sequence, $(z, x, y, z, x, y, \cdots)$. We experimentally determined that the best results in terms of balanced wirelength and via count were produced by this new cut sequence.

Instead of focusing only on wirelength and via minimization while making cell moves, LC-CUT reduces the overall local congestion as necessary. The necessity of congestion-driven moves is determined by a variable called thresh, which controls the nature of gain computations. Figure 3 shows a pseudocode representation of this algorithm, which is explained in detail below.

A. Initialization Phase

The first stage of the LC-CUT algorithm involves the initialization of a bucket structure, which stores the weighted gains of all cells in C. First, all cells in the netlist are

```
==============================================
Algorithm: LC-BICUT
Input: netlist, area constraint, threshold T
output: bipartitioning result
----------------------------------------------
g_c: cutsize gain
g_l: local wire congestion gain

[Bi,Bj] = initial_partition();
compute g_c and g_l for all cells;
compute LC(Bi) and LC(Bj);
old = LC(Bi) - LC(Bj);
while (there exists a free cell)
    c = maximum gain cell;
    update g_c for neighbors of c;
    if (cong_mode)
        update g_l for neighbors of c;
    new = LC'(Bi) - LC'(Bj);
    if (new < T)
        cong_mode = TRUE;
    if (|old| > T & |new| > T & old*new < 0)
        recalculate g_l;
    else if (|old| <= T < |new|)
        recalculate g_l;
    else if (|new| <= T < |old|)
        g_l = 0 for all cells;
    old = new;
return Bi and Bj;
==============================================
```

Fig. 4. algorithm

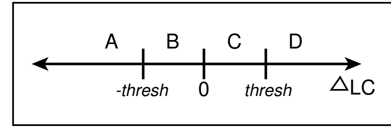
inserted into either B_i or B_j such that the area constraints are satisfied. Each cell will have the opportunity to move from its current block to a neighboring block in the later stages of the algorithm. Next we perform the computation of $g_c(c)$, the cutsize gain. Additionally, the local congestion gain (defined below) is also computed. For a given cell c, moved from Bi to Bj, the change in congestion for block B_i, denoted δ_i, is computed as follows:

$$\delta_i = LC(B_i) - LC(B_i - \{c\})$$

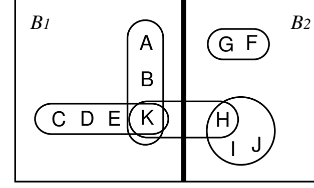
δ_j is computed using $LC(B_j)$. Finally, the local congestion gain of moving c from its current block B_i to a neighboring block B_j is given by:

$$g_l = |LC(B_i) - LC(B_j)| - |LC(B_i) - LC(B_j) + \delta_i + \delta_j|$$

The local congestion values for the two blocks Bi and Bj are calculated. We then perform the initialization of two important variables that will determine the frequency of congestion-driven moves. The first is a threshold value T, for which the maximum cell degree must be a lower bound. We use this lower bound value plus 4% of $LC(B_i) + LC(B_j)$. This is to ensure the accuracy of congestion gain computations. The second is cong_mode, a boolean which, if true, will initiate local congestion-driven cell moves. Otherwise, the moves will be made purely on the basis of g_c , the cutsize gain. Then, the difference between the local congestions of the two blocks is stored.



(a)



(b)

Fig. 5. (a) Optimization mode for LC-CUT, (b) example for local wire congestion computation.

B. Cell Movement Phase

The second stage of the LC-CUT algorithm continuously moves cells of maximum gain until there is no more positive gain left. The cell of maximum gain, c, is extracted from the bucket. Then, we make sure that moving c will not result in an area constraint violation. We update B_i and $B - j$, making the cell move, and updates g_c for all neighbors of c. Then, we check to see if the balance gain computations are necessary, and if so, updates all g_l for all neighbors of c, according to the above equation. These updates for local congestion gain can be done incrementally since the values of LW for each block are stored after every move and the calculations for δ_i and δ_j are trivial.

The overall gain is calculated as a weighted cost function of g_c and g_l. If this gain value is less than zero, then the loop is exited and B_i and B_j are returned. Otherwise, the values of LC are updated, and new becomes $LC(B_i) - LC(B_j)$. As shown in Figure 2, the magnitude of the difference between $LC(B_i)$ and $LC(B_j)$ determines whether or not congestion-based moves are made.

During the next portion of the algorithm, the values of $g_l(c)$ are updated if necessary. Figure 2 shows the four regions of possible values for local wire congestion balance Δ: A, B, C and D. When Δ is in B or C, the congestions in B_i and B_j are relatively equal. However, when Δ is in A or D, one block is significantly more locally congested than the other. For this reason, local congestion is considered when Δ is in A or D, and it is ignored when Δ is in B or C. This serves to improve runtime since congestion gain calculations are unnecessary when Δ is in $[-T, T]$.

Certain moves that cause Δ to shift from one particular region to another will necessitate a bucket re-initialization. The moves $A \to D$ and $D \to A$ result in the re-computation of g (c) for every cell c and a bucket reset. This is necessary since the sign of Δ has changed, and according to the equation above, $g_l(c)$ will change. The algorithm checks for a cell move from $\{B, C\}$ to $\{A, D\}$. For this case, local congestion must

again be considered, so g_l is computed and the bucket is reset. We check for a transition of Δ from $\{A, D\}$ to $\{B, C\}$. In this situation, the local congestion gains for all cells are set to zero and the bucket is reset, and therefore contains only cutsize gain information.

Figure 4 depicts an example of a stage in LC-CUT where the highlighted cell K is the one to be moved. Currently, $LC(B_i) = 5$ and $LC(B_j) = 3$. Therefore, $\Delta = 2$. Then, $g_c(K) = 1 - 1 - 1 = -1$, $\delta_i = -2$ and $\delta_j = 1$. So, $g_l(K) = |5 - 3| - |5 - 3 - 2 + 1| = 1$. After moving cell K, $LC(B_i) = 3$ and $LC(B_j) = 4$. Then, $\Delta = -1$. Additionally, the cell gains for A, B, C, D, E and H must be updated before the next cell move is made.

V. Global Wire Congestion and Thermal Optimization

A. Overview of the Approach

We use a simulated annealing-based block movement technique for minimization of global wirelength, via count, global wire congestion and maximum thermal gradient of the placement solution obtained above. Since local pair-wise congestion, 2D wirelength and via count have already been minimized, it is sufficient to swap entire blocks (rather than individual gates, or clusters of gates) and check for improvements over the previous solutions. We use the min-cut results as the initial solution and compute initial temperature T0. The temperature is then decremented in a standard one-parameter exponential format ($T_{i+1} = a \cdot T_i, a < 1$). We let $N(T)$ be the number of random block swaps made at every temperature T. The cost C of a global placement solution is defined as follows:

$$C = a_1 \cdot VIA + a_2 \cdot WL + a_3 \cdot GC + a_4 \cdot TG$$

where VIA, WL, GC, and TG respectively denote the via, wirelength, global wire congestion, and maximum thermal gradient cost as discussed in Section II. As is standard with all annealing algorithms, improvements are guaranteed only at a significant runtime expense. In order to make the procedure as efficient as possible, it becomes necessary to perform highly optimized incremental evaluation, which is described in detail below.

B. Incremental Global Wire Congestion Computation

Recall the global congestion metric: Given a boundary $[B_i, B_j]$ between neighboring blocks B_i and B_j, we defined the boundary congestion to be the number of hyperedges crossing that boundary, denoted by $|H_{ij}|$. That is: $GC_{ij} = |H_{ij}|$. Consider such a hyperedge h in H_{ij}. Let its bounding box lie between $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$ as shown in Figure 5 below.

Note that when two blocks are swapped, boundaries need to be updated only for nets in Hij. This property allows us to conveniently ignore all nets that are not incident upon the two blocks central to the current move, thereby achieving remarkable improvements in overall runtime. We now define

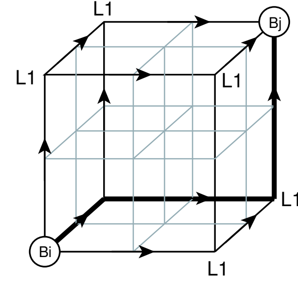


Fig. 6. Incremental maze routing

acyclic paths between the source and destination blocks of h (say B_{hs} and B_{hd}) as finite sequences of boundaries:

$$P(h) = \{b1, b2, \cdots, bn\}, b1 \in B_{hs}, bn \in B_{hd}, bi \neq bj$$

The boundary congestion of a path is simply defined to be the sum of individual congestions of its component boundaries. Once Bi and Bj have been randomly selected for the Ith move, the boundary congestion contribution of all nets h in the corresponding Hij is computed. Then, we perform incremental maze routing on h within the bounding box. During this process, we use Dijkstra's algorithm to compute the path of least boundary congestion for every such h. The difference in global congestions of the solutions before and after the move is measured, and $\Delta C(I)$ is updated.

Given B_i and B_j, the blocks to be swapped for the i-th move at temperature T, we only update h_x and h_y, the wirelength and via contributions of every hyperedge h in H_{ij}. Thus, VIA and WL can be incrementally updated at relatively low runtime costs.

C. Thermal Analysis

Though the 3D-ADI thermal simulator has a linear runtime and memory requirement, and is unconditionally stable, it is still too expensive from a runtime point of view to call a full thermal simulation at every move in simulated annealing. A less accurate yet faster alternative approach that was adopted to solve this problem was to call a full thermal simulation every 'K' moves, where 'K' is dependent on the circuit being analyzed. A brief synopsis is provided in the following paragraph.

In the beginning, a full thermal simulation is called at the beginning of simulated annealing. Following this, 'K' moves are made without any thermal simulation. These 'K' solutions are evaluated and compared to each other using the cost function which is dependent on global congestion, and wire length. The best solution determined within these K moves is thermally simulated and is compared to previous fully simulated solution on the basis of global congestion, wire length, as a well as thermal profile. The better solution is then set as the best current solution, and simulated annealing continues on this solution. This process is repeated for 'K' moves, after which a full thermal simulation is called on the best solution among the latest 'K' moves. This solution is

again compared with the previous best solution based on local congestion, wirelength, and thermal. The simulated annealing process is repeated until the cost is brought below the desired amount.

VI. Experimental Results

Our algorithms were implemented in C++/STL, compiled with gcc v2.96 with -O3, and run on Pentium III 746 MHz machines. The benchmark set consisted of six circuits from ISCAS89 and five circuits from ITC99 suites. The relevant statistical information of the benchmark circuits is shown in Table 1. We ran our experiments using 8 x 8 x 4 block placement. The pure mincut method was implemented using our own multilevel recursive bipartitioning with a (z,x,y) cut sequence, which is a balanced combination of the two techniques suggested in [6]. The LC-CUT algorithm was run under the same framework as the aforementioned.

Four flavors of LC-CUT were used on the benchmark set, with suitably modified gain functions. The results are presented here in Table 1. The first flavor defined gain purely in terms of reduction in local wirelength and via numbers, the second focused on local congestion, the third restricted gain to power optimization, and the fourth balanced all these objectives. The results reveal that minimizing along only one of these variables is unlikely to improve the remaining parameters in the solution space, and that the best way to ensure balanced improvement in all objectives is to select appropriate nonzero weights for all terms in the gain function.

Furthermore, our second simulated-annealing based phase provides uniform improvement over the LC-CUT results, as seen in Table 2. Again, four cost functions were tried, and uniform improvement was achieved by using experimentally determined balanced weights. Local congestion is not reported here; since no changes are made below the block-level at this stage, local wire congestion remains unchanged from LC-CUT.

VII. Conclusions

Thus, we achieve thermal gradient and local congestion improvement of 25% over the standard mincut approach, global congestion improves by 7%, and the wirelength/via numbers are not raised by more than 10%. The versatility of the approach allows us to target one or more of these design objectives and suitably alter the gain function to provide user-specific particular results in this large solution space. We are currently devising an adaptive approach to automatically determine the globally optimal values of weights in the gain function of LC-CUT.

References

[1] Y. Deng and W. Maly, "Interconnect characteristics of 2.5-d system integration scheme," in *Proc. Int. Symp. on Physical Design*, 2001.

[2] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2003.

[3] T. Tanprasert, "An analytical 3-d placement that reserves routing space," in *Proc. IEEE Int. Symp. on Circuits and Systems*, 2000.

[4] R. Ravichandran, J. Minz, M. Pathak, S. Easwar, , and S. K. Lim, "Physical layout automation for system-on-packages," in *IEEE Electronic Components and Technology Conference*, 2004.

[5] P. H. Shiu, R. Ravichandran, S. Easwar, and S. K. Lim, "Multi-layer floorplanning for reliable system-on-package," in *Proc. IEEE Int. Symp. on Circuits and Systems*, 2004.

[6] S. Das, A. Chandrakasan, and R. Reif, "Design tools for 3-d integrated circuits," in *Proc. Asia and South Pacific Design Automation Conf.*, 2003.

[7] R. Zhang, K. Roy, C.-K. Koh, and D. B. Janes, "Exploring soi device structures and interconnect architectures for 3-dimensional integration," in *Proc. ACM Design Automation Conf.*, 2001.

[8] T.-Y. Wang and C. C.-P. Chen, "3-d thermal-adi: A linear-time chip level transient thermal simulator," *IEEE Trans. on Computer-Aided Design*, pp. 1434–1445, 2002.

TABLE I

LOCAL WIRE CONGESTION AND THERMAL OPTIMIZATION RESULTS

ckts	wirelength/via-driven				local-congestion-driven				power-driven				all			
name	wire	via	l-wc	ther	wire	via	l-wc	ther	wire	via	l-wc	ther	wire	via	l-wc	ther
s5378	2315	232	26	12.39	3386	418	14	11.09	2517	281	21	13.5	2374	267	19	13.5
s9234	2232	250	59	30.23	3931	476	16	19.52	2639	276	54	16.98	2726	271	36	26.24
s13207	2332	293	77	11.7	4793	618	21	27.78	2934	347	99	21.48	3139	413	63	18.23
b14_opt	5633	635	40	34.81	8204	1110	17	44.67	7045	724	39	30.95	7024	727	22	34.3
b15_opt	8478	960	40	20.86	12418	1856	18	22.04	9445	1081	41	22.66	9355	1119	39	29.42
b20_opt	8859	1039	70	19.51	15185	2184	27	23.5	10582	1057	60	20.81	11035	1123	49	17.36
b21_opt	9445	1069	86	28.95	14750	1732	27	20.54	10631	1182	62	16.31	11259	1304	56	18.92
b22_opt	11020	1410	93	25.73	19351	2847	32	16.75	12748	1346	87	18.4	12154	1321	73	19.57
s38417	3945	433	144	10.08	9560	1548	55	9.94	4234	445	135	14.2	4578	510	110	12.07
s35932	2922	323	72	45.93	7292	944	30	18.99	3951	393	80	10.47	3864	389	72	11.48
s38584	4645	407	120	16.15	10845	1021	31	20.03	5420	623	126	14.34	5524	636	84	25.75
RATIO	1.00	1.00	1.00	1.00	1.77	2.09	0.35	0.92	1.17	1.10	0.97	0.78	1.18	1.15	0.75	0.88
TIME	216				1467				844				1622			

TABLE II

GLOBAL WIRE CONGESTION AND THERMAL OPTIMIZATION RESULTS

ckts	wirelength/via-driven				global-congestion-driven				thermal-driven				all			
name	wire	via	g-wc	ther	wire	via	g-wc	ther	wire	via	g-wc	ther	wire	via	g-wc	ther
s5378	2095	257	27	13.40	3296	298	23	9.72	2741	299	29	13.50	2235	278	27	11.13
s9234	2068	244	30	33.18	3998	360	33	18.58	2845	301	42	15.79	2106	255	38	17.54
s13207	2357	215	31	12.85	4514	491	24	26.84	3120	355	55	20.15	2587	233	29	24.91
b14_opt	5071	649	57	37.34	8067	953	60	43.73	7641	768	84	32.79	5548	688	66	32.09
b15_opt	7491	942	100	25.39	12633	1734	82	21.10	10245	1254	108	20.81	7698	1054	98	24.17
b20_opt	8883	945	88	22.28	15076	2033	61	22.56	11598	1200	99	19.31	9056	956	72	21.63
b21_opt	9189	1034	95	30.15	13514	1576	60	19.60	12455	1278	124	16.55	9312	1154	71	15.39
b22_opt	10146	1242	102	29.58	21399	2936	92	15.81	14106	1569	146	17.56	12454	1465	103	19.21
s38417	3615	464	49	10.30	9520	1433	24	9.01	4415	455	56	13.14	3945	497	28	18.31
s35932	3059	325	38	49.98	7343	823	24	18.05	4046	416	42	9.53	3264	377	29	12.23
s38584	4628	566	54	19.71	10501	890	52	19.09	5745	667	88	12.30	4956	604	66	15.70
RATIO	1.00	1.00	1.00	1.00	1.87	1.97	0.80	0.79	1.35	1.24	1.30	0.67	1.08	1.10	0.93	0.75
TIME	2034				7936				9951				11114			