Effective Thermal Via and Decoupling Capacitor Insertion For 3D System-On-Package

Eric Wong, Jacob Minz, and Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332, USA
{ewong, jrminz, limsk}@ece.gatech.edu

***Abstract*— The increased component density of a 3D System-On-Package (SOP) exacerbates the thermal hotspot problem. A popular choice to mitigate the thermal issues is thermal vias (t-vias) that are used to establish thermal paths from the core of an SOP package to the heat sinks. Another major problem with SOP integration is the power supply noise coupling among various mixed signal components constituting the system. In this case, decoupling capacitors (decaps) are inserted to provide the switching currents locally. The goal of our automatic 3D SOP component placement algorithm is to determine the x/y/z location of each component while minimizing the footprint area under thermal and power supply noise constraints. In general, t-vias and decaps are typically inserted in the white space in the placement, whereas the proximity of the t-vias and decaps to the target components determines their effectiveness. Hence, our component placer considers t-via and decap insertion during the early design stage, where the component location can be flexibly changed. Related experiments demonstrate the effectiveness of our approach.**

I. Introduction

The increased component density of a 3D System-On-Package (SOP) structure exacerbates the thermal hotspot problem. That is, more devices packed into a smaller footprint result in a higher maximum temperature. A popular choice to mitigate the thermal issues is thermal vias (t-vias) that are used to establish thermal paths from the core of an SOP package to the heat sinks. Another major problem is the power supply noise coupling among various mixed signal components constituting the system. Due to the low noise floor required for analog components, considerable power supply noise primarily generated by the high-speed digital components occurs through the common inductive impedance of the power/ground return current path. A popular choice to mitigate the power supply noise issues is decoupling capacitors (decaps) that are used to provide the switching currents locally.

The goal of our automatic 3D SOP component placement algorithm is to determine the x/y/z location of each component while minimizing the footprint area under thermal and power supply noise constraints. We note that the placement of components has a significant impact on the amount of thermal vias and decaps required. This is because the temperature of each component depends heavily on thermal coupling with neighboring components. In addition, the simultaneous switching noise (SSN) level of each component is affect by the noise coupling with neighboring components as well as the distance to power supply pins. Lastly, the effectiveness of t-vias and decaps depends on the location of white space (WS), which is determined by the component placement. This is because t-vias and decaps are typically inserted in the WS, whereas the proximity of the t-vias and decaps to the target components determines their effectiveness. Hence, it is important to consider t-via and decap insertion during the early design stage, where the component location can be flexibly changed.

The remainder of the paper is organized as follows. Section II presents an overview of our algorithm. Section III and IV respectively presents our thermal via and decoupling capacitor insertion algorithms. Experimental results are presented in Section V, and we conclude in Section VI.

II. Overview of the Algorithm

The effectiveness of decaps is maximized when the decaps are located adjacent to the blocks that need them. This means some existing whitespace (WS) is not accessible if surrounded by the blocks that do not use it. In addition, any additional WS inserted is required to be adjacent to the target blocks. On the other hand, the effectiveness of t-vias is maximized when the t-vias make straight connections between the top and the bottom heat sinks. Since the components in each layer become obstacles, t-vias can be inserted only at the location where WS vertically overlaps in all layers. Moreover, the WS must be added in all layers with some overlap in case additional WS is desired. We employ a two-stage approach that consists of stochastic optimization via Simulated Annealing followed by an iterative t-via/decap insertion. The purpose of the SA-based optimization is to obtain a 3D component placement solution that requires the minimum amount of t-vias and decaps during the next stage. The purpose of the detailed t-via/decap insertion is to detect, insert, and allocate WS in an iterative manner until the thermal and SSN constraints are met.

During the annealing process, we generate a candidate 3D component placement solution and evaluate it in terms of area, thermal, and decap cost. We use a 3D mesh to apply a finite difference approximation for thermal analysis. Each node models a small volume of the SOP packaging structure, and each edge denotes the connectivity between two adjacent regions. Our matrix equation $T = R \cdot P$ computes the temperature of each mesh node, where T, R, and P respectively denote the temperature vector, thermal resistance matrix, and power generation vector. Finally, the thermal cost of a 3D placement is the maximum temperature among all components. We use another 3D mesh to model the 3D power/ground network. The edges in the mesh have inductive and resistive impedances. The dominant path for a component is the path from the nearest current source to the component causing the greatest drop in voltage. Then, the SSN for a

1-4244-0152-6/06/$20.00 ©2006 IEEE

given component is the summation of IR drop and Ldi/dt change (drop or increase) along its dominant path p. It is possible that there exist several edges in p that are shared with the dominant paths for other components. In this case, the total sum of IR drop and Ldi/dt change on these shared edges caused by the related components is used for the SSN computation of individual components. The decap budget for each component is computed according to its current demand and SSN. Finally, the decap cost of a 3D placement is the total decap budget among all components.

During our t-via/decap insertion step, we first detect the existing WS in the 3D component solution. We formulate the WS-to-component allocation problem using a network-flow model, where we attempt to allocate WS for both t-via and decap. The key constraint is that the WS for decaps needs to be adjacent to the related components, where as the WS for t-vias need to have vertical overlap in all layers. In case the existing WS is not enough to suppress the SSN or temperature under a given threshold, we insert additional WS while considering the adjacency and vertical overlap requirements. Lastly, our thermal/SSN analysis is performed to verify the effect of the t-via/decap insertion and detect the next target components. We extend the existing 2D placement encoding scheme named Sequence Pair to 3D in such a way that it is easier to find a way to slide the components in x/y directions so that the vertical overlap among a certain set of WS is always maintained.

III. Thermal Via Insertion

A. Thermal Model

A standard 3D thermal resistance mesh is used for thermal analysis. Each node models a small volume of the 3D die stack (substrate, heat sink, dielectric, metal, or transistor), and each edge denotes the thermal conductivity between two adjacent regions. This is equivalent to using a discrete approximation of the steady state thermal equation $-k\nabla^2 T = P$, where k is thermal conductivity, T is temperature, and P is power. This results in the matrix equation $G \cdot t = p$, where G is a thermal conductivity matrix, p is a power vector, and t is a temperature vector. One way to solve this matrix equation would be to invert the matrix $G^{-1} = R$, which takes $O(n^3)$ time. Then t can be calculated through matrix multiplication $t = R \cdot p$, which takes $O(n^2)$ time.

During thermal driven floorplanning, moving blocks around does not significantly change the thermal conductivities. The power profile changes are mainly responsible for the changes in temperature. This allows the G matrix to be inverted to R once in the beginning and reused for subsequent temperature calculations. Only the power vector needs to be changed, so temperature calculations only require one matrix multiplication. This allows the temperature of each floorplan to be evaluated in $O(n^2)$ time rather than $O(n^3)$ time. This method of reusing R is slightly inaccurate due to the fact that the area of the floorplan will change, which causes slight changes in thermal conductance between thermal grid cells. When inserting thermal vias, however, thermal conductivities change. This means that R cannot be reused, so directly solving the matrix equation would take $O(n^3)$. This is much too slow for use in integrated thermal via floorplanning. To solve this problem we propose another method for calculating temperature.

B. Random Walk-based Thermal Analysis

Random walks correspond to a classical problem in statistics, and their use in solving linear equations dates back to as early as the 1950s [1] [2] [3]. Recently, Qian et al. [4] [5] applied the random walk concept to power grid analysis. In a random walk game, a walker starts at a node in a graph with a certain amount of money. The walker then randomly visits a neighboring node. The probability of each neighbor being visited is based on the weight of its edge to the current node. At each node, the walker either receives a reward or pays a toll. The walk ends when the walker reaches a home node and the walker will have made or lost some money based on the tolls paid and rewards collected.

The temperature of a thermal grid cell is calculated by placing a walker with no money at the cell. First, the walker will receive a reward of

$$r(i) = \frac{p_i}{\sum_j^{d(i)} g_{ij}} \tag{1}$$

where p_i is the power of the current cell i, $d(i)$ is the edge degree of cell i, and g_{ij} is the thermal conductance between cell i and its neighbor j. The walker will then visit one of its six neighboring cells. The probability of each neighbor j being chosen from cell i is

$$p(i,j) = \frac{g_{ij}}{\sum_j^{d(i)} g_{ij}}. \tag{2}$$

At each step, the walker will receive a reward and visit another neighbor. The walk ends when the walker hits a boundary cell at this point the walker will receive the final reward r = ambient temperature. The total amount of money collected by the walker is an approximation of the temperature of the cell that the walker started from. According to the Central Limit Theorem, if many walks are performed and the results are averaged, then the error is a zero mean Gaussian variable with a variance inversely proportional to the number of walks k. This gives a tradeoff between runtime and accuracy. The runtime of the random walk is $O(kmn)$, where k is the number of walks per cell, m is the average length of a walk, and n is the number of cells. Typically, k and m are much smaller than n, so a random walk will run much faster than solving the matrix equation $G \cdot t = p$ with a runtime of $O(n^3)$.

Several techniques can be used to speed up the random walk-based thermal analysis. It is possible for a random walk to wander around inside the thermal grid and not reach a boundary cell for an extremely long time. To combat this problem, a limit on the path length of a random walk m_{max} is imposed. If m_{max} is set too low, then many random walks will be cut short. Losing too many long walks will tend to cause the calculated temperatures to be low. When m_{max} is set high enough, few random walks will be affected and

the underestimation becomes negligible. The next speed up technique is to create new home cells. When the temperature of a cell is calculated, it becomes an additional home cell with a reward equal to its temperature for subsequent random walks starting elsewhere. The new homes cut down on the average length of walks significantly. The temperatures of individual cells can be calculated without having to solve the entire thermal grid, which is done by performing random walks starting from the cells of interest and not performing random walks starting elsewhere. This is especially useful for thermal via insertion since this allows the local impact of thermal vias on a target hot-spot to be calculated without recalculating the entire temperature profile.

C. Thermal Via Insertion Algorithm

An iterative method is used for thermal via insertion. First the thermal grid cell with the highest temperature is found. Then the target thermal conductivity of the cell is calculated according to the formula

$$k_{new} = k_{old} \cdot \frac{t_{curr}}{t_{target}} \tag{3}$$

where k_{old} is the current thermal conductivity of the cell, t_{curr} is the current temperature of the cell, and t_{target} is the target temperature. The via density of the x-y location of the cell is calculated with the formula

$$v = \min\left(v_{max},\ c \cdot \frac{k_{new} - k_{old}}{k_{via} - k_{old}}\right) \tag{4}$$

where v_{max} is the maximum thermal via density, c is a user defined constant, and k_{via} is the thermal conductivity of a thermal via. Next, the thermal conductivities are updated according to the thermal via density. Random walk is used to calculate the temperature of the cells that the new vias pass through as well as the temperature of adjacent cells. Then another grid cell with the highest temperature is found and the process repeats. This process is iterated until the maximum temperature is less than the target temperature or when the maximum number of iterations has been reached.

After thermal via insertion, blocks that occupy areas with thermal vias need to be expanded to make room for the vias. The average via density of a block is the amount that it will expand by. Next, a sequence pair floorplan compaction calculation is used to update the location of the expanded blocks. With updated block sizes and locations, a final temperature calculation can be performed. If the via insertion is integrated into the floorplanning, then the random walk thermal analyzer is used for the temperature calculation. If the via insertion is done as a postprocess, then the temperature is calculated with the matrix thermal analyzer. The result of the thermal via inserter is a 2-dimensional thermal via density map. Thermal vias can then be placed according to the thermal via density map, where they will be fixed obstacles during the placement phase of physical design.

D. Integrated Floorplanning with Thermal Vias

The floorplanner is based on simulated annealing. An array of sequence pairs was used to represent to solution space, with one sequence pair per layer. Each move is made by modifying the sequence pair. Then, the area of the floorplan and the location of the blocks is calculated from the sequence pair using an algorithm based on longest common subsequence [6]. The wirelength of a net is estimated by drawing a bounding box around the blocks connected by the net and taking the half perimeter of the bounding box. The temperature before thermal via insertion is calculated using the fast matrix thermal analyzer. Then thermal vias are inserted. The random walk based thermal analyzer is used to calculate the temperature after thermal via insertion. Then a weighted average of the area, wirelength and temperature *after* thermal via insertion is used as the cost function for the simulated annealer for integrated thermal via floorplanning.

In area and wirelength driven floorplanning, the cost function is a weighted average of the area and wirelength. In thermal driven floorplanning the cost function is a weighted average of the area, wirelength, and temperature *without* vias. When thermal vias are inserted for the final floorplan, the matrix thermal analyzer is used to calculate the temperature before and after thermal via insertion to ensure accurate final results.

IV. Decoupling Capacitor Insertion

A. 3D Power Supply Noise Modeling

We use a 3D grid to model the power/ground (P/G) network for 3D SOP. Each P/G layer in the multi-layer structure is represented as a mesh. The edges in the mesh have inductive and resistive impedances. The mesh contains power-supply points and connection points. The connection points consume currents. The current is drawn from all the sources by the consumers, and the amount of current drawn along a path is inversely proportional to the impedance of the path in the power supply mesh. The *dominant current source* for a block is defined as the voltage source supplying significantly more power to the block than any other neighboring sources. The *dominant path* for a block is the path from the dominant supply to the block causing the most drop in voltage. It has been shown experimentally in [7] that the shortest path between the dominant current source (nearest Vdd pins) and the block offers highly accurate SSN estimation within reasonable runtime. Let P_k be a dominant current path for block k. Then $T^k = \{P_j : P_j \cap P_k \neq \emptyset\}$ denotes the set of all other dominating paths overlapping with P_k (T^k includes P_k itself). Let P_{jk} be the overlapping segments between path P_j and P_k. Let $R_{P_{jk}}$ and $L_{P_{jk}}$ denote the resistance and inductance of P_{jk}. After the current paths and their values have been determined for all blocks, the SSN for B_k is given by

$$V^k_{noise} = \sum_{P_j \in T^k} (i_j \cdot R_{P_{jk}} + L_{P_{jk}} \frac{di_j}{dt})$$

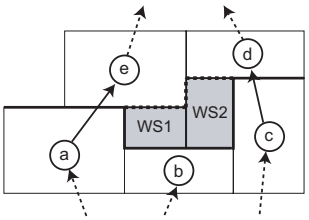

Fig. 1. Whitespace detection. Blocks a, b, c are in the lower level. Blocks d, e are in the next level. The bold line is the lower boundary, while the dotted line is the upper boundary. $ws1, ws2$ are the detected white spaces.

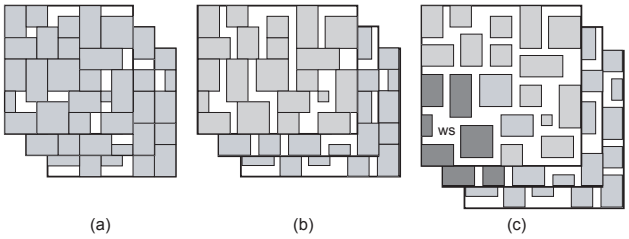

Fig. 2. Illustration of 3D decap allocation. (a) 3D placement, (b) X-expansion, (c) XY-expansion, where the darker blocks denote the neighboring blocks of the decap (= white space) inserted. Note that blocks from other layers can utilize the white space for decap insertion.

where i_j is the current in the path P_j, which is the sum of all currents through this path to various consumers. The weight of i_j and its rate of change are the resistive and inductive components of the path.

In the worse case, a module would draw all of its switching current from its decap. Let $Q^k = \int_0^{t_s} I^k(t) \cdot dt$ denote the maximum charge drawn from the power supply by block B_k, where $I^k(t)$ is the current demand and t_s is the switching time. The decap budget can then be calculated as $C^k = Q^k/V_{tol},\ 1 \leq k \leq M$, where M denotes the total number of blocks. This base decap budget is for the case where there is no resistance between a block and its decap.

B. Whitespace Detection and Insertion

The white space present in a floorplan can be used to fabricate decap. If the existing white space is insufficient or unreachable by modules needing decap, then white space insertion through floorplan expansion may be necessary. Hence detection of all existing white spaces in a floorplan is highly desirable. This is done by using the longest path tree calculation based on the vertical constraint graph. All nodes at the i^{th} level in the tree are at an edge distance of i from the source node. Each level is ordered by the horizonal constraint graph. The white spaces at level i are detected by comparing the *upper* boundary of blocks at level i and the *lower* boundary of the blocks at level $i+1$. If the boundaries are not incident on each other, then there is whitespace. In Figure. 1, blocks a, b, c are in the same level and blocks d, e are in the next level. The algorithm compares the upper boundary of a, b, c, to the lower boundary of d, e. The mismatched boundaries allows the algorithm to find white spaces $ws1, ws2$. This algorithm is capable of detecting all white spaces, and runs in $O(n)$ time, given the ordered longest path tree, where n is the total number of blocks. Typically, longest path tree calculations from constraint graphs are used to convert sequence pairs into floorplans.

If sufficient decap cannot be allocated from the existing white space to suppress the SSN, then more white space is added by expanding the floorplan in the X and Y direction as illustrated in Figure 2. A naive approach is to look at the additional decap needed for each layer and expand as necessary, splitting the X and Y expansion evenly. However, this does not take advantage of the 3D structure. Our *Footprint-aware area expansion* algorithm finds the X and Y slack of each layer relative to the footprint and expands in the direction with more slack. If a particular layer is the bottle-neck layer, i.e. it has maximum width and height, then some of the expansion is shifted to adjacent layers. Allowing blocks to use decaps in other layers is made possible by effective distance [8].

Note that there may be iteration between decap allocation and whitespace insertion before sufficient decap is allocated to all blocks. The XY-expansion of each layer is controlled by α and β parameters, where α and β are the percent expansions in the X and Y directions. Simple expansion would set α and β equal to each other. In *footprint-aware* expansion, the X and Y slack of each layer are defined as $S_x = Footprint_{width} - Layer_{width}$. Then the equation $\beta/\alpha = S_y/S_x$ is used to make the white space insertion favor the direction with more slack. After each iteration, the α and β are increased until the decap demands are met.

C. Flow-based Decap Allocation

In this work, the decap allocation problem is modeled by *generalized network flow* as illustrated in Figure 3. Generalized network flow problems generalize traditional network flow problems by adding a *gain factor* $\gamma(e) > 0$ for each arc e. For each unit of flow that enters the arc, $\gamma(e)$ units must exit. For traditional network flows, the gain factor is one. Capacity constraints and node conservation constraints are satisfied by the generalized networks, as in traditional network flows. This model accurately captures the decap allocation problem with effective distance [8]. Generalized network flow is a well studied problem, but elegant exact and approximate algorithms have only been proposed recently [9].

The nodes on the left represent the blocks. The capacities of the incoming edges are the decap demands of the blocks. The costs of these edges are zero and the gains are unity. The nodes on the right represent the whitespace modules. The capacities of the outgoing edges are the areas of the whitespace modules. The gains are unity, and the costs are set to one. If a circuit module is close enough to draw decap from a whitespace module, they are connected with an edge of infinite capacity, zero cost, and gain factor γ_{eff} to represent the effectiveness of the whitespace, based on distance. The gain factor of the edge between a block and a white space is the amount of area needed to satisfy unit decap. A min-cost maximum flow

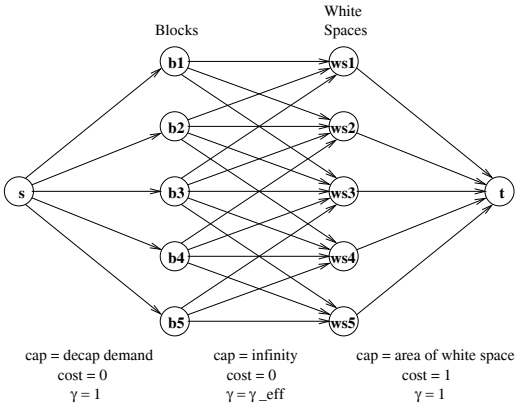

Fig. 3. Network flow model for decap assignment

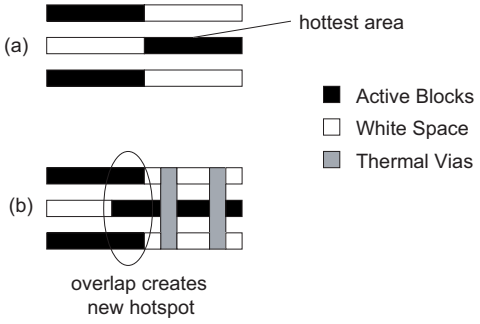

Fig. 4. A three layer floorplan before thermal via insertion (a) and after thermal via insertion (b). In this case, adding thermal vias expanded the block in the middle layer and created a new hotspot.

in this generalized network, allocates the maximum possible decap and uses the minimum white-space area.

If the flow in the source edges are saturated, then the decap demands of all the circuit modules can be met. Assigning cost to the sink edges minimizes the use of the whitespace. If the flow in some source edges are less than capacity, then there is not enough whitespace to fulfill the decap demands of the circuit modules. In this case the floorplan must be expanded to add additional whitespace. In the 3D environment, the smaller layers will be expanded first to avoid increasing the footprint area of the entire package. This expansion can also help circuit modules on other layers since the effective distance formulation allows circuit modules to draw decap from other layers.

V. Experimental Results

The algorithms in the paper were implemented in C++. The experiments were run on Pentium IV 2.4 Ghz dual processor systems running linux. Ten GSRC benchmarks [10] were used. The blocks were randomly assigned power densities between 10^6 W/m^2 and 5×10^6 W/m^2. All floorplans have four placement layers.

A. Thermal Via Insertion Results

Table I shows the results of the AWF algorithm (area and wirelength driven floorplanning) with thermal via insertion as a postprocess. The eighth column shows what the temperature of the floorplan would be if the floorplan were expanded but thermal vias were not added. Floorplan expansion was responsible for a temperature drop of approximately 4%, while the increase in thermal conductivity due to thermal vias was responsible for an additional 13% temperature drop. Average thermal via density is the proportion of the area reserved for thermal vias. Note that this is not necessarily equal to the area expansion because the expansion of individual blocks is not uniform. An average thermal via density of under 3% was able to decrease temperatures by almost 17% while expanding the area by less than 4% and increasing wirelength by only 1%.

Table II shows the results of the TDF algorithm (thermal driven floorplanning) followed by thermal via insertion. In half the cases, adding thermal vias actually increased the temperature. The temperature without thermal via column suggests the reason for this. The TDF tends to separate high power blocks. The area expansion of blocks due to thermal vias can cause the blocks to shift enough to bring some high power density blocks closer together, which can increase temperature. Figure 4 shows an example of this effect. The increased thermal conductivity from the thermal vias can sometimes make up for this effect, but often it cannot. TDF without thermal vias is more effective at reducing temperatures than AWF followed by thermal via insertion. However, TDF has higher area due to looser module packing.

Table III shows the results of IVF algorithm (integrated thermal via floorplanning). IVF solved the problem that TDF had with thermal vias by being aware of thermal vias throughout floorplanning. This allowed it to produce the lowest temperatures out of the three methods. We conclude the following based on the three tables so far: adding thermal vias to AWF reduced the temperature by 17% at a cost of 4% area expansion and 1% wirelength increase. TDF without thermal vias reduced temperature by 32% at a cost of 20% area increase and 5% wirelength increase. Finally, IVF reduced temperature by 38% at a cost of 47% area increase and 22% wirelength increase. The thermal via density of IVF averages 2.5%, so most of the area increase came from loose module packing.

B. Decoupling Capacitor Insertion Results

Table IV compares area/wirelength driven floorplanning and decap driven floorplanning. For the large (200 block) circuits, decap driven floorplans have better area than the area/wirelength driven floorplans. However, this improvement comes at the expense of wirelength. Having a 3D structure has many benefits over 2D. The following observations can be made from Table V. The wirelength *decreases* by 28% when going from a single to double layered floorplan, and *decreases* by 50% for a floorplan with four layers. The decap amount *decreases* by 24% and 60% for double and quadruple layered floorplans, respectively. Original area *decreases* by 48% and 72% when increasing layers to two and four. The reduction in expanded area after decap allocation is slightly greater. This suggests that 3D structures offer greater flexibility in decap

TABLE I

AREA AND WIRELENGTH DRIVEN FLOORPLANNING WITH THERMAL VIA INSERTION AS A POSTPROCESS.

	before via insertion			after area expansion					
benchmarks	area	wirelength	temp	area	wirelength	temp w/ vias	temp w/o vias	average via density	time
n50	58491	91521	136.6	59309	91986	126.5	132.4	0.011	121
n50b	66490	87838	145.1	72564	90886	115.7	145.0	0.065	110
n50c	63666	92418	129.2	64251	92900	122.1	125.6	0.008	111
n100	57664	135970	123.6	61431	138729	92.3	113.6	0.046	193
n100b	49950	120431	112.9	51095	121297	98.0	112.2	0.013	408
n100c	53040	132142	128.8	54135	133800	95.4	117.6	0.027	251
n200	50190	215549	135.6	52472	218601	105.2	131.9	0.036	1407
n200b	55385	226447	125.9	57579	228792	103.7	123.6	0.031	887
n200c	52877	250970	123.6	53601	251855	110.8	120.8	0.011	643
n300	81340	313680	186.9	83801	316041	146.9	174.1	0.026	4395
RATIO	1.000	1.000	1.000	1.035	1.012	0.831	0.963	-	-

TABLE II

THERMAL DRIVEN FLOORPLANNING WITH THERMAL VIA INSERTION AS A POSTPROCESS.

	before via insertion			after area expansion					
benchmarks	area	wirelength	temp	area	wirelength	temp w/ vias	temp w/o vias	average via density	time
n50	62517	91363	120.7	66393	93772	98.5	118.3	0.057	852
n50b	68694	85173	118.1	71571	86781	130.8	144.2	0.042	1307
n50c	64532	91808	110.6	66912	93797	105.3	113.8	0.051	1106
n100	68480	142521	82.0	69541	143942	90.2	93.4	0.028	1708
n100b	61490	127801	84.8	63066	129821	81.8	96.2	0.052	2175
n100c	63745	138324	85.7	65655	139790	91.3	105.6	0.032	1425
n200	62220	270123	97.9	63406	271742	91.0	97.9	0.019	3389
n200b	70596	250672	69.1	72039	253473	105.1	109.7	0.051	5509
n200c	66150	250582	74.4	67874	252339	69.7	78.2	0.032	6202
n300	117600	334304	51.4	118397	335528	67.5	67.6	0.046	17659
RATIO	1.000	1.000	1.000	1.028	1.013	1.071	1.169	-	-

TABLE III

INTEGRATED THERMAL VIA FLOORPLANNING

benchmarks	area	wirelength	temp	average via density	time
n50	86093	102425	94.1	0.035	9175
n50b	82925	94088	108.6	0.017	13107
n50c	80303	100013	86.8	0.050	8979
n100	83311	155972	87.7	0.007	16110
n100b	81893	148806	71.7	0.022	23514
n100c	81596	152045	76.4	0.020	24524
n200	74414	310017	75.8	0.029	29417
n200b	82599	284590	98.7	0.022	25140
n200c	77465	304035	68.7	0.022	32053
n300	136907	468086	56.4	0.027	47337

allocation. Decap decreases because the compact 3D structure allows for shorter paths from blocks to power pins. For the 2D floorplans, there is a much larger area expansion for decap allocation since footprint awareness is unavailable.

VI. CONCLUSIONS

We presented a component placer for 3D System-On-Package that considers thermal via and decoupling capacitor insertion during the early design stage, where the component location can be flexibly changed. First, a fast approximation algorithm for thermal analysis was presented. This thermal analyzer was incorporated into an efficient thermal via insertion algorithm. The thermal via inserter successfully lowered temperatures with minimal thermal via densities. Integrating thermal via insertion into the floorplanner resulted in lower temperatures than inserting vias as a postprocess. Our placer also aims at reducing the amount of decoupling capacitance (decap) needed to suppress the simultaneous switching noise without compromising traditional design metrics such as area and wirelength. We performed footprint-aware decap insertion to allow functional blocks to access decaps in other layers.

REFERENCES

[1] G. Forsythe and R. Leibler, "Matrix inversion by a monte carlo method," *Mathematical Tables and Other Aids to Computation*, 1950.

[2] W. Wasow, "A note on the inversion of matrices by random walks," *Mathematical Tables and Other Aids to Computation*, 1952.

[3] P. Doyle and J. Snell, "Random walks and electric networks," *Mathematical Association of America*, 1984.

[4] Q. H. Qian, S. Nassif, and S. Sapatnekar, "Random walks in a supply network," in *Proc. ACM Design Automation Conf.*, 2003.

[5] Q. H. Qian and S. Sapatnekar, "Hierarchical random-walk algorithms for power grid analysis," in *Proc. Asia and South Pacific Design Automation Conf.*, 2004.

TABLE IV

AREA/WIRELENGTH-DRIVEN VS DECAP-DRIVEN 3D FLOORPLANNING. EFFECTIVE DECAP DISTANCE AND FOOTPRINT-AWARE DECAP INSERTION SCHEMES ARE USED FOR BOTH.

	area/wirelength-driven FA ED				decap-driven FA ED			
ckt	wire length	decap	area before	area after	wire length	decap	area before	area after
n50	21541	17	22185	22185	25784	2	25258	25258
n50b	21065	11	23944	23944	22266	3	23828	23828
n50c	18505	10	18340	18340	21449	1	18720	18720
n100	54264	80	35148	35148	65470	52	36860	36860
n100b	40848	86	33990	34302	57145	53	33998	33998
n100c	53240	83	33286	33286	66365	50	36966	36966
n200	155370	235	67599	67751	173997	219	52948	54211
n200b	166159	244	68021	68636	182016	231	53352	54632
n200c	152917	242	63612	63917	171901	233	52675	52974
ratio	1.000	1.000	1.000	1.000	1.178	0.574	0.968	0.970
time	677				1894			

TABLE V

IMPACT OF THE NUMBER OF PLACEMENT LAYERS. EFFECTIVE DISTANCE AND FOOTPRINT AWARENESS ARE USED.

	single layer				2 layers				4 layers			
ckt	wire length	decap	area before	area after	wire length	decap	area before	area after	wire length	decap	area before	area after
n50	50764	66	83790	84959	36806	38	50052	50104	25784	2	25258	25258
n50b	45422	58	80032	81097	35662	27	45346	45346	22266	4	23828	23828
n50c	43081	61	66515	67089	35287	20	39780	39780	21449	1	18720	18720
n100	126017	137	128904	131217	92714	122	59787	60877	65470	52	36860	36860
n100b	106440	142	101985	106269	68810	116	60977	61080	57145	53	33998	33998
n100c	117112	145	130800	133164	88819	126	60977	62120	66365	50	36966	36966
n200	407024	279	198628	205829	249744	264	98820	102958	173997	219	52948	54211
n200b	354319	284	249066	256141	269754	271	102718	105962	182016	231	53352	54632
n200c	398234	288	190437	197270	245559	275	98245	101436	171901	233	52675	52974
ratio	1.000	1.000	1.000	1.000	0.718	0.757	0.524	0.519	0.499	0.401	0.282	0.277
time	1424				3222				1895			

[6] X. Tang, R. Tian, and D. F. Wong, "Fast evaluation of sequence pair in block placement by longest common subsequence computation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2001.

[7] S. Zhao, C. Koh, and K. Roy, "Decoupling capacitance allocation and its application to power supply noise aware floorplanning," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 81–92, 2002.

[8] E. Wong, J. Minz, and S. K. Lim, "Power Noise-aware 3D Floorplanning for System-On-Package," in *Proc. IEEE Electrical Performance of Electronic Packaging*, 2005.

[9] Y. Tsai, A. Ankadi, N. Vijaykrishnan, M. Irwin, and T. Theocharides, "ChipPower: An Architecture-Level Leakage Simulator," in *Proc. IEEE Int. SOC Conf.*, 2004.

[10] GSRC, http://www.cse.ucsc.edu/research/surf/GSRC/GSRCbench.html.