Multi-objective Microarchitectural Floorplanning For 3D Microprocessors

ABSTRACT

In this paper, we present the first multi-objective microarchitectural floorplanning algorithm for processors implemented in deep-submicron 3D ICs. Our floorplanner takes a microarchitectural netlist and determines the dimension as well as the placement of the functional modules into multiple device layers while simultaneously achieving high performance and thermal reliability. The traditional design objectives such as footprint and wirelength are also considered. Our 3D floorplanning algorithm considers the following 3D-specific issues: vertical overlap optimization and bonding-aware layer partitioning. We provide comprehensive experimental results on performance, thermal, footprint, and wirelength tradeoff.

1. INTRODUCTION

Future processors implemented in deep-submicron technologies will spend more time communicating data operands or exchanging control information than actually performing useful computation. Meanwhile, the impact of power and thermal densities on the devices and interconnects continues to increase, thereby raising the cost for cooling solutions, eroding performance gains, and threatening overall circuit reliability. The 3D integrated circuit is an emergent technology that vertically stacks multiple die with a die-to-die interconnect as illustrated in Figure 1. The die-to-die via pitch is very small and provides the possibility of arranging functional unit blocks across multiple die at a very fine level of granularity. This results in a decrease in the overall wire length, which translates into shorter wire delay and less power. Thus, 3D ICs can address the wire delay problem effectively by replacing the long and slow global interconnects with short and fast vertical routes. Advances in 3D integration and packaging are undoubtedly gaining momentum and have become of critical interest to the semiconductor community.

Microarchitectural floorplanning has recently drawn significant interest from both the computer architecture and EDA communities[1, 2, 3, 4, 5]. The main motivation is to tackle the ever-worsening wire-delay problem of high-performance processors with a collaborative effort between microarchitecture and physical CAD. The location of individual microarchitectural modules plays a significant role on many important metrics. First, the floorplan has a huge impact on the performance of a given microarchitecture (measured by IPC) as the global interconnects between modules are likely to be pipelined in order to meet high target clock frequencies. This may increase or decrease the access latency on all intermodule interconnects. Second, the thermal and leakage profile is highly correlated to the floorplan. This is because the temperature of microarchitectural modules is not solely dependent on the heat generation rate of each individual module but also the heat coupling between neighboring modules. Moreover, the leakage power of each transistor is exponentially proportional to the temperature. Third, floorplanning affects the dynamic power consumption of the buses and clock distribution network. The total number of flip-flops (FFs) inserted on global interconnects changes the dynamic power consumed by the clock distribution network. However, the performance and thermal objectives are conflicting with each other since shorter distance among the active blocks improves the performance while exacerbating the thermal issue. The contributions of this work are as follows:

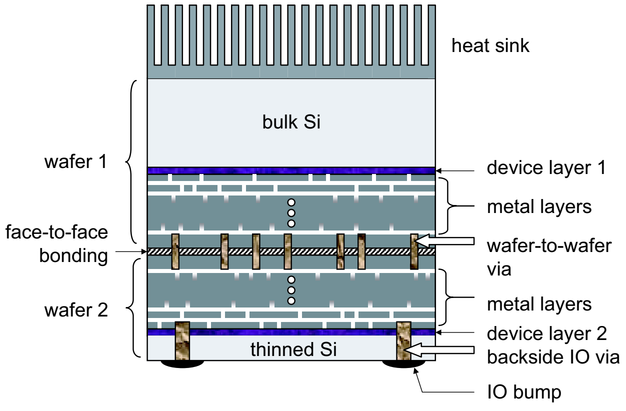

Figure 1: 2-die 3D IC with face-to-face bonding.

- This work proposes the first multi-objective floorplanner for 3D deep-submicron processors at the *microarchitectural* level. Our 3D floorplanners simultaneously consider performance, thermal reliability, footprint, and interconnect length objectives, providing various tradeoff points for different design requirements.

- This work provides in-depth discussions along with effective solutions for the following important 3D-specific problems: *vertical overlap optimization*, and *bonding-style aware layer partitioning*. We show how the vertical overlap among modules in 3D floorplanning affects performance, thermal, and footprint objectives. In addition, we discuss how to perform layer partitioning under different inter-die via requirements existing in face-to-face and face-to-back bonding in 3D stacked ICs.

Recent studies on 2D microarchitectural floorplanning [1, 2, 3, 4, 5] have focused on performance optimization but not thermal concerns. Several microarchitecture research works on thermals [6, 7] provide runtime management of the functional modules but do not perform floorplanning. The existing thermal-aware 3D floorplanners [8, 9] are targeting gate-level circuit blocks and thus are not suitable for evaluating different microarchitecture designs.

2. ARCHITECTURAL SIMULATION

2.1 Performance Modeling

The microarchitectural configuration used in our study is summarized as follows: the machine width is 8. We use a 1024-entry gshare branch predictor, a 512-entry register update unit, 16KB instruction and data L1 caches, a 256KB unified L2 cache, 128-entry instruction and data TLBs, 8 ALUs, 4 FPUs, and a 64-entry load store queue. Our architectural simulator faithfully models all inter-module communication latencies, which is a critical factor in high-frequency processors. Essentially, the inter-module latency is a function of the distance and the number of flip-flops between modules and must be taken into account in both performance evaluation and floorplanning. For this reason, we use the distances generated by the floorplanner to determine the latency-related parameters such as pipeline depth and communication/forwarding latencies for performance simulation. The IPC computation for 3D is similar to the 2D case except that the access latency on each interconnect is calculated based on 3D floorplanning that involves delay in the z-dimension.

2.2 Dynamic and Clock Power Modeling

While collecting the inter-module traffic, we also generate the power consumption profile for each microarchitectural module cumulatively for every hundred thousand cycles. The rationale for such sampling is that the temperature is very unlikely to elevate abruptly within a processor's operation period of a few hundred thousand cycles. Wattch [10] has been integrated into our framework to provide an estimation of the dynamic power during this simulation. We assume that the intra-module dynamic power consumption remains the same for different floorplans as the module activity factors primarily depend on the program behavior rather than the relative positions. However, the new floorplan may lead to different interconnect lengths between modules. Thus, our tool recomputes all of the inter-module interconnect power based on the new lengths and adds it to the dynamic per-module power collected earlier.

The number of flip-flops inserted on the wires for an extremely high clock frequency can create a larger load on the clock distribution network. This combined with the increasing percentage of the power budget that the clock distribution network consumes necessitates modeling the clock power at a finer granularity. Toward this, we use the accurate clock power model from [11]. This model considers clock distribution network power for memory structure precharge arrays, distribution wiring and drivers, pipeline flip-flops, and the phase locked loop.

2.3 Leakage Power Modeling

Our leakage power computation is based on [12]. To calibrate our model, we also calculate the subthreshold leakage currents using the method in eCACTI [13]. Our model closely matches the leakage power estimated from eCACTI. For logic structures, we assume that half of the CMOS transistors are leaking at any given time. The number of transistors in these structures is estimated using the area values from GENESYS [14]. Due to the temperature dependence on the subthreshold leakage current, we first estimate the leakage power based on an initial temperature. The results of this estimate are then fed to our thermal analyzer so that it will estimate the temperature and the leakage power more accurately. This is done within the thermal analyzer by modeling their interdependence. We follow the criteria proposed in [15] for detecting the scenarios of thermal runaway: (i) the maximum module temperature T_{max} is increasing, and (ii) the increment of power is larger than the increment of package's heat removal ability. The package's heat removal ability is defined as $(T_{max} - T_a)/R$, where T_a and R are ambient temperature and thermal resistance of the package.

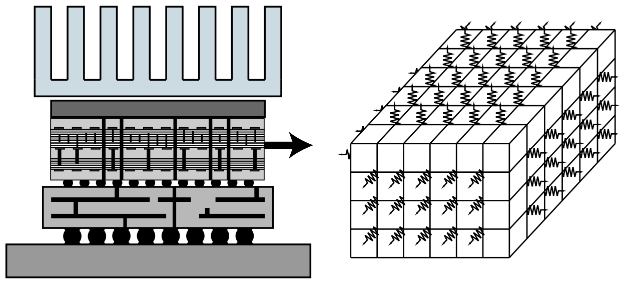
Figure 2: 3D grid of a chip for thermal modeling

2.4 Temperature Modeling

The chip is divided into a 3D grid as shown in Figure 2 to apply a finite difference approximation. We write the thermal equation into the following matrix form: $R \cdot P = T$, where R is the thermal resistance matrix ($R_{i,j}$ is the thermal resistance between node i and node j), P is the power profile vector (P_i is the power dissipation of node i), and T is the temperature profile vector (T_i is the temperature of node i). Thus, the temperature of all the active nodes can now be calculated from the power profile using a single matrix-vector multiplication. The clock power is distributed evenly across the modules according to their areas. The bus power for each net is added to the total power of the source block. Then, the leakage power and temperature of each module are calculated iteratively using our model until they either converge or thermal runaway is detected.

In order to facilitate fast but reasonably accurate temperature calculation, we use a non-uniform 3D thermal resistor mesh, where grid lines are defined at the center of each microarchitectural module. These grid lines are defined for the X and Y directions and extend through the Z direction to form planes. The intersection of grid lines in the X and Y directions define the thermal nodes of the resistor mesh. The total power of each block is distributed according to and among the X-Y area of the nodes that block covers.

2.5 Integrated Design Flow

Our design flow incorporates the dynamic power, leakage power, performance, and thermal analysis discussed earlier into our floorplanner. An overview of this design flow is illustrated in Figure 3. First, we estimate the area and delay of the microarchitectural modules using CACTI [16] and GENESYS [14]. Then we perform a cycle-accurate simulation using SimpleScalar [17] to collect and extract the amount of traffic between modules for each benchmark. We also collect the dynamic power consumption of the modules. We then feed the module-level netlist, statistical interconnection traffic, dynamic module power, and a target frequency to our multi-objective 3D floorplanner.

We first partition the modules into multiple layers (consideration of 3D bonding styles is discussed in Section 3). Our subsequent floorplanning step then determines the location and dimension of the modules in all layers simultaneously. During our 3D floorplanning step, we first recursively bipartition the 3D floorplan area un-

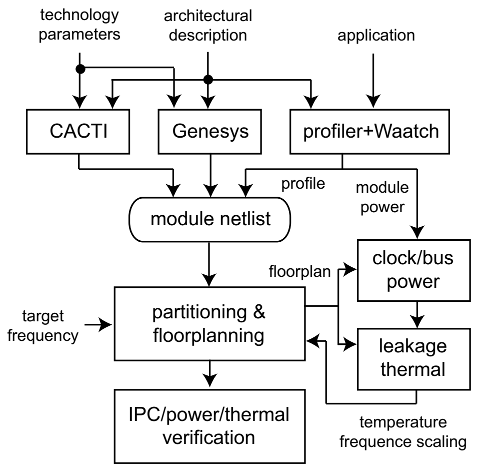

Figure 3: Overview of our 3D microarchitectural floorplanning framework.

til each module is confined in its own partition. Each bipartitioning solution is optimized by an LP-based approach, where performance and temperature objectives are simultaneously considered under the leakage power constraint. We then update the clock and bus power based on the new floorplan. Once the recursive bipartitioning is finished, we further optimize the current solution during our SA-based refinement. We perform low-temperature annealing to fine-tune the LP-based solution, where the temperature/leakage analyzer is again used to guide our optimization. The last step is to evaluate the final solution for IPC, power, and thermal metrics using our simulation infrastructure.

3. 3D FLOORPLANNING ALGORITHM

3.1 Vertical Overlap Optimization

A unique challenge that exists in 3D floorplanning is the issue of *vertical module overlap*. The primary benefit that a 3D IC provides is the ability to place tightly connected modules *on top of* each other instead of *adjacent to* each other as in the 2D case. This significantly reduces the length and thus the delay/power of the related interconnects. Since the parasitics associated with the inter-die vias are similar to those of short interconnects, the additional freedom in the z-dimension promises higher quality floorplans in terms of footprint, performance, and power consumption. In addition, the shorter interconnects naturally mitigate the interconnect congestion problems. More specifically, the vertical overlap affects the quality of 3D microarchitectural floorplanning in the following ways:

- **Performance**: the performance of a 3D microarchitectural floorplan tends to improve when the vertical overlap is maximized among the blocks with higher access frequencies. This is mainly caused by the shorter interconnect and thus the lower access latency among the frequently communicating modules.

- **Thermal**: the thermal profile of a 3D microarchitectural floorplan tends to deteriorate due to compressed space. More hotspots are created when the vertical overlap is maximized among the hot modules. This harmful thermal coupling causes the leakage power to increase, raising the likelihood of thermal runaway.

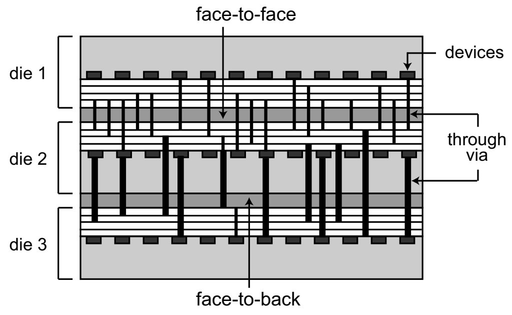

Figure 4: Through vias in 3D ICs with face-to-face and face-to-back bonding.

- **Power**: the dynamic module power and clock power are rarely affected by the vertical overlap. However, the overall bus power consumption tends to decrease with more vertical overlap among the modules with higher switching activities. This is because the dynamic power saving is greater when highly active modules drive shorter interconnects. Note that this is in conflict with the thermal objective since highly active modules tend to become hotter.

In summary, our 3D floorplanning tries to maximize the vertical overlap among the frequently communicating and highly switching modules while minimizing the vertical overlap among the hot modules.[1] Since these objectives are competing with each other, trading one objective for the others is inevitable.

3.2 Bonding-aware Layer Partitioning

A 3D IC requires special kinds of vias for inter-die connection called *through-vias*. There are three kinds of through-vias depending on the style of bonding mechanism used to bond two die together: *face-to-face* (F2F), *face-to-back* (F2B), and *back-to-back* (B2B) through-vias, as illustrated in Figure 4. The “face” refers to the metal layer side of a die, whereas the substrate side is called “back”. F2F through-vias ($\approx 0.5\mu \times 0.5\mu$) have a smaller pitch than F2B and B2B through-vias ($\approx 5\mu \times 5\mu$) [18]. In addition, too many F2B/B2B through-vias fabricated on a single thinned wafer may adversely affect reliability [19] since these vias actually penetrate the substrate. Thus, it is desirable to *reduce* the number of inter-die connections in F2B/B2B bonding. In the case of F2F bonding, however, it is desirable to *increase* the number of inter-die connections since the via density is much higher (almost the same as intra-die via density) and thus enables a significantly higher bandwidth for inter-layer communication. Note that F2B/B2B bonding is inevitable if the number of die exceeds two. Moreover, in the case that all three bonding styles are used in a single 3D IC, the 3D floorplanning has to be done carefully to manage the different bonding styles.

In our two-step approach for 3D floorplanning, we first partition the modules into layers (= die) and then floorplan these layer. The goal during our layer partitioning is to exploit the bonding style and vertical overlap opportunities, whereas our floorplanning optimizes the vertical overlap for performance, footprint area, and thermal objectives. During our layer partitioning, we assign a layer to each module such that the connection at the F2F boundary is maximized while the F2B/B2B connection is minimized. Next, we split pairs

[1]Note that it is possible to impose vertical overlap constraints among related groups of modules. The investigation of this direction is out of the scope of this paper.

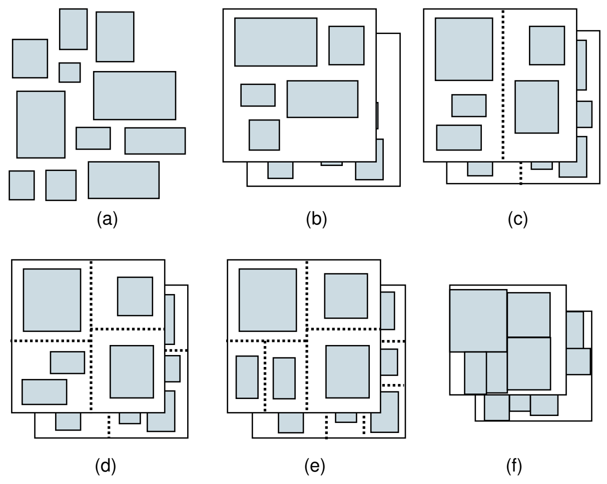

Figure 5: Illustration of our 3D microarchitectural floorplanning. (b) layer partitioning, (c-e) LP-based 3D slicing floorplan, (f) non-slicing floorplan refinement.

of modules connected via high profile-weighted edge into two layers with F2F bonding so that we can vertically overlap them during the subsequent floorplanning step for achieving better performance. In addition, we split highly active modules in the same way, i.e., two layers with F2F bonding, such that the shorter interconnect connected to these modules help reduce the dynamic power. Since the temperature of the modules requires floorplanning information, our layer partitioning is not temperate-aware. Finally, we separate the modules with large area such as the RUU into different layers to help minimize the footprint area and reduce the amount of white space. In our greedy construction algorithm, we sort the modules according to their size, power density, and switching activity. We then assign the best possible layer for each module based on the performance, power, and footprint objectives mentioned earlier.

3.3 LP-based 3D Floorplanning

The goal of our 3D floorplanning is to determine the dimension and location among the modules to minimize our multi-objective function. Our LP-based approach is based on slicing floorplanning to handle multiple layers simultaneously. The basic idea is to perform recursive bipartitioning until each partition contains a single module. Moreover, we insert each slicing cutline to cut *all* layers simultaneously as illustrated in Figure 5. After we choose a partition to be divided, we perform thermal/leakage analysis to obtain module temperature. We then use LP to simultaneously optimize the performance and thermal distribution under the target frequency, leakage, and the center of gravity constraints (to remove overlap among the modules within the same die).

A major difference between the 2D and 3D slicing floorplan is the interaction with different layers, which is the key element for an effective 3D floorplan. More specifically, the vertical overlap discussed in Section 3.1 has a high impact on performance and thermal objectives. In addition, area optimization has to be footprint-aware: the area increase from the smallest layer can be easily tolerated since it is less likely to increase the overall footprint area. Our LP formulation reflects this new optimization goal, unique to 3D floorplanning. Since the layer partitioning has already addressed the bonding-style-related issues, we do not allow the modules to move to other layers during the floorplanning.

The following variables are used for our LP-based 3D slicing floorplanning formulation: N and E are the set of all modules and buses in the netlist. x_i and y_i are the location of module i. l_i is the layer of module i. w_i and h_i are the half width and half height of module i. a_i and g_i are the area and delay of module i. $w_m(i)$ and $w_x(i)$ are the minimum/maximum width of module i. $\lambda_{i,j}$ is the normalized profile weight on wire (*i,j*). $z_{i,j}$ is the number of flip-flops on wire (*i,j*) after insertion. $X_{i,j} = |x_i - x_j|$, $Y_{i,j} = |y_i - y_j|$, and $L_{ij} = |l_i - l_j|$. $T_{i,j}$ is the normalized product of the temperature of modules i and j. A is the aspect ratio constraint of the 3D footprint. X_x and Y_x are the maximum among x_i and y_i, respectively. C is the target cycle time. d_r is the unit length delay of repeated interconnects. Finally, d_z is the delay of inter-layer vias. Our LP floorplanner determines the values for the following decision variables: x_i, y_i, w_i, h_i, and z_{ij}. Our LP-based slicing floorplanning is to minimize:

$$\sum_{(i,j)\in E} (\alpha \cdot \lambda_{ij} \cdot z_{ij} + \beta \cdot (1 - T_{ij})(X_{ij} + Y_{ij}) + \gamma \cdot X_x) \quad (1)$$

Subject to:

$$z_{ij} \geq \frac{g_i + d_r(X_{ij} + Y_{ij}) + d_z L_{ij}}{C},\ (i,j) \in E \quad (2)$$

$$X_{ij} \geq x_i - x_j \text{ and } X_{ij} \geq x_j - x_i,\ (i,j) \in E \quad (3)$$

$$Y_{ij} \geq y_i - y_j \text{ and } Y_{ij} \geq y_j - y_i,\ (i,j) \in E \quad (4)$$

$$z_{ij} \geq 0,\ (i,j) \in E \quad (5)$$

$$w_m(i) \leq w_i \leq w_x(i),\ i \in N \quad (6)$$

$$x_i,\ y_i \geq 0,\ i \in N \quad (7)$$

$$X_x \geq x_i \text{ and } A \cdot X_x \geq y_i,\ i \in N \quad (8)$$

$$\sum_{i\in S(j)} a_i x_i = \sum_{i\in S(j)} a_i \times \bar{x}_j,\ i \in N,\ j \in B \quad (9)$$

$$\sum_{i\in S(j)} a_i y_i = \sum_{i\in S(j)} a_i \times \bar{y}_j,\ i \in N,\ j \in B \quad (10)$$

where B is the set of all partitions in the current slicing floorplan, $S(j)$ is the set of modules in partition j, and $(\bar{x}_j, \bar{y}_j)$ is the center location of partition j.

Our objective Equation (1) contains three terms: profile-weighted wirelength (= $\lambda_{ij} \cdot z_{ij}$) for performance optimization, thermal-weighted wirelength (= $(1-T_{ij})(X_{ij}+Y_{ij})$), and footprint (= X_x). The performance-related term minimizes the distance between the frequently communicating modules if these are in the same layer. Otherwise, the vertical overlap will be maximized as long as the reference point of module location is consistent, e.g., lower left corner of each module. In addition, the thermal-related term separates two hot modules in the same layer *and* minimizes the vertical overlap between two hot modules in different layers while maintaining linearity. Finally, the area-related term still captures the minimization of 3D footprint area as long as the X_x and Y_x are computed based on the modules in all layers. Since minimizing $X_x \cdot Y_x$ (= footprint area) is non-linear, we only minimize X_x since the constraint (8) enforces $A \cdot X_x$ to be greater than all y values. Note that α, β, and γ are user-defined parameters for weighing the performance (P), thermal (T), and footprint (F) objectives. In the case that $\beta = 0$, our floorplanner optimizes P+F only. In the case that $\alpha = 0$, our floorplanner optimizes T+F objective only. Lastly, the conventional footprint/wirelength-driven (F+W) floorplanner uses the following new objective function:

$$\gamma \cdot X_x + \delta \cdot \sum_{(i,j)\in E} (X_{ij} + Y_{ij}) \quad (11)$$

We provide an extensive comparison on these four different floorplanning objectives (P+T+F, P+F, T+F, and F+W) in Section 4. Note that the footprint objective is used in all of these variations. The footprint objective has a positive impact on performance and wirelength objectives and a negative impact on thermal objective.

This latency constraint (2) considers the delay of inter-layer via delay as well as interconnect delay during the computation of FFs needed to satisfy the clock period constraint C. If there is no FF on a wire (i,j), the delay of this wire is calculated as $d(i,j) = d_r(X_{ij} + Y_{ij}) + d_z L_{ij}$. Then, $g_i + d(i,j)$ represents the latency of module i accessing module j, where $d(i,j)$ denotes the delay between i and j. Since C denotes the clock period constraint, $(g_i + d(i,j))/C$ denotes the minimum number of FFs required on (i,j) in order to satisfy C. Absolute values on x and y distance are given in (3)–(4). Constraint (5) requires that the number of FFs on each edge is non-negative. The center of gravity constraints (9)–(10) require that the area-weighted mean (= center of gravity) among all modules in each sub-block corresponds to the center of the sub-block.

Table 1: Multi-objective floorplanning results with performance (P), maximum block temperature (T), footprint (F), wirelength (W), and runtime reported. The LP+SA-based floorplanner is used.

	F+W		F+P		F+T		F+P+T	
bench	IPC	Tx	IPC	Tx	IPC	Tx	IPC	Tx
gzip	2.74	104.7	3.98	125.9	2.75	98.9	2.94	107.9
swim	0.71	92.9	0.85	106.9	0.72	84.1	0.88	90.7
vpr	1.30	111.5	1.40	137.0	1.25	107.1	1.23	117.9
art	0.52	95.6	0.59	111.4	0.52	87.9	0.64	94.8
mcf	0.10	92.0	0.11	105.4	0.10	83.1	0.11	89.2
equake	0.54	91.7	0.58	105.0	0.55	82.6	0.70	88.8
lucas	0.87	116.9	0.92	145.3	0.88	113.0	1.14	126.8
gap	1.59	97.0	1.59	114.2	1.62	89.6	1.70	97.3
bzip2	1.94	106.8	2.05	129.0	1.98	101.5	2.27	110.7
twolf	0.81	114.6	1.03	142.2	0.84	111.0	0.98	122.6
RATIO	1.00	1.00	1.18	1.19	1.01	0.94	1.14	1.02
FOOTPRINT	15.84		23.69		17.97		26.45	
WIRE	217.20		323.43		252.08		247.02	
TIME	180		438		16913		20016	

3.4 3D Stochastic Refinement

The next step refines the 3D slicing floorplanning result using a non-slicing floorplanning approach. We derive a sequence-pair per layer from the 3D slicing floorplanning result and perform low temperature simulated annealing (SA) with them. We use the *gridding* scheme described in [20] to derive the corresponding sequence-pair representation from the slicing floorplan. Specifically, we draw the positive and negative loci for each module and order these loci to obtain the sequence pair. Since our SA-based refinement is restricted to intra-layer perturbation in order to maintain the bonding-aware layer separation, a 3D-packing encoding scheme such as 3D TCG [21] is not necessary. We use the following cost function during annealing:

$$cost = \alpha \cdot prof + \beta \cdot max\_temp + \gamma \cdot footprint$$

where $prof$ is the profile-weighted wirelength, and max_temp is the maximum module temperature. We use the same weighting constants α and β used in Equation (1) between the performance and thermal objectives. It is important, however, to note that our temperature is *not* the weighted distance between two hot blocks but the *actual* temperature we obtain from our thermal analyzer. Thus, our thermal analysis is the runtime bottleneck during our refinement since we need to perform the analysis for potentially many candidate solutions during the annealing process.

Assuming that the thermal conductivity of functional modules are similar (they are mostly silicon), swapping the location of modules does not significantly change the thermal resistance matrix R. This means that matrix R only needs to be computed once in the beginning. To calculate the temperature profile of a new floorplan, the power profile P needs to be updated and then multiplied by R. Alternatively, a change in power profile ΔP can be defined. Multiplying R and ΔP will give change in temperature profile ΔT. Adding ΔT to the old temperature profile will give the new temperature profile. On the other hand, leakage and clock power update are done faster since it basically involves evaluating a set of equations based on the new module locations and temperature values.

4. EXPERIMENTAL RESULTS

Experiments were performed on 10 programs from the SPEC 2000 benchmark suite. For IPC evaluation, each benchmark was run on the average-case floorplan using a modified SimpleScalar 3.0 [17] by fast-forwarding 4 billion instructions and simulating the next 4 billion instructions. The reported temperature is simulated after all floorplanning steps and is adjusted relative to a $45°C$ ambient temperature. The 3D floorplan is based on a 4-layer stacked IC. Face-to-face bonding between layer 0 (bottom-most) and 1 and layer 2 and 3 are assumed. Back-to-back bonding is used between layer 1 and 2. The heat sink is attached to layer 3. Wirelength is reported in mm. The footprint area of the 4-layer floorplan (= maximum width $\times$ maximum height) is reported in mm^2. The runtime of the framework was collected on Pentium Xeon 2.4 GHz dual-processor systems. The runtime of profiling 4 billion instructions after fast-forwarding 4 billion instructions was about 4 hours per benchmark as was the power collection simulation for the same sets of instructions. The floorplanning steps took approximately 25 minutes and the simulations for the reported values of temperature and IPC took approximately 2 minutes and 1 hour per benchmark, respectively.

Table 1 presents a comparison of the performance (P), temperature (T), footprint (F), wirelength (W), and runtime of 4 different objective functions. All data in this table are taken from the combined LP+SA approach. The baseline algorithm is F+W. First it is noted that the footprint result of F+W is the best among all objective functions. F+P increases the IPC by 18% over F+W but it also increases the temperature by 19%. As expected F+T significantly decreases the temperature result and achieves the best temperature results among all four 3D algorithms. The 4X increase in grid size for the temperature simulations in the 3D case causes the runtime of those objectives incorporating temperature calculations to increase dramatically. The hybrid F+P+T retains a temperature close to that of F+W while increasing the IPC by 14%. In summary, F+P+T (i) obtains balanced results that are between those of F+T and F+P and (ii) outperforms F+W in terms of performance with comparable temperature. In case the temperature should be more emphasized, the thermal weight can be increased, which will likely lead to performance degradation.

Table 2 presents a comparison among the pure SA, pure LP, and hybrid LP+SA. One can observe that the pure LP floorplanner does very poorly on the footprint of the floorplan, which allows it to achieve a better average maximum block temperature than the combined approach. The IPC values also maintain a trend similar to the temperature. The wirelength values are within an acceptable range for all approaches, though it is interesting to note that while the pure LP approach creates a large footprint the wirelength values are still comparable. This is because while wirelength was an objective during the recursive bipartitioning phase of the LP, the footprint is

Table 2: Comparison among pure-SA, pure-LP, and LP+SA approaches. The objective used is a linear combination of performance, thermal, and footprint all with equal weight.

	pure SA		pure LP		LP+SA	
bench	IPC	temp	IPC	temp	IPC	temp
gzip	2.74	109.5	2.31	97.5	2.94	107.9
swim	0.71	91.8	0.70	86.7	0.88	90.7
vpr	1.07	119.8	1.24	103.4	1.23	117.9
art	0.52	95.7	0.51	89.0	0.64	94.8
mcf	0.10	90.4	0.10	85.9	0.11	89.2
equake	0.54	90.0	0.53	85.7	0.70	88.8
lucas	0.87	128.7	0.85	108.1	1.14	126.8
gap	1.59	98.9	1.49	90.9	1.69	97.3
bzip2	1.94	112.2	1.81	99.4	2.27	110.7
twolf	0.81	124.8	0.77	106.2	0.98	122.6
RATIO	0.98	1.11	0.93	0.99	1.14	1.09
FOOTPRINT	21.59		70.64		24.44	
WIRE	230.47		207.57		241.77	
TIME	25157		18207		20113	

not because the formulation has no way to constrain overlap. The runtime of all approaches was roughly equivalent, showing that in a similar amount of time LP+SA produces a higher-quality solution.

Figure 6 shows snapshots of a floorplanning solution. LP+SA with footprint, performance, and temperature objectives is used. The whitespace of the floorplan is somewhat less than optimal due to higher weights for performance and temperature. The design flow provides users with the ability to modify the objective weights to suit their needs. This figure demonstrates that there is indeed thermal coupling between adjacent modules and that the thermal portion of the objective has attempted to separate the hottest modules while the performance portion of the objective has caused some of the hottest modules to remain grouped.

5. CONCLUSIONS

We presented the first multi-objective microarchitectural floorplanning algorithm for high-performance, high-reliability microprocessors targeting 3D ICs. We considered performance and thermal objectives such that our floorplanner can provide a balanced or goal-directed processor organization that achieves user-specified design objectives. We studied the impact of the vertical overlap optimization and bonding-aware layer partitioning issues that are unique in 3D ICs. Our ongoing work tries to address the area optimization under a multi-objective, multi-layer environment more effectively. In addition, we are working on utilizing the whitespace in 3D floorplan for decoupling capacitor, thermal via, and buffer insertion. Lastly, we are investigating faster thermal analysis based on the random walk scheme.

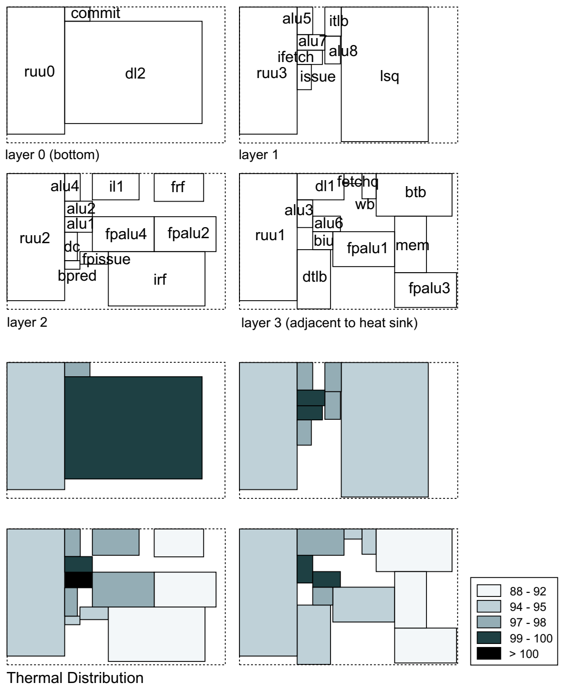

Figure 6: 3D floorplan with thermal distribution

6. REFERENCES

[1] C. Long, L. Simonson, W. Liao, and L. He. Floorplanning optimization with trajectory piecewise-linear model for pipelined interconnects. In *Proc. ACM Design Automation Conf.*, 2004.

[2] J. Cong, A. Jagannathan, G. Reinman, and M. Romesis. Microarchitecture evaluation with physical planning. In *Proc. ACM Design Automation Conf.*, 2003.

[3] M. Casu and L. Macchiarulo. Floorplanning for throughput. In *Proc. Int. Symp. on Physical Design*, 2004.

[4] M. Ekpanyapong, J. Minz, T. Watewai, H.-H. S. Lee, and S. K. Lim. Profile-guided microarchitectural floorplanning for deep submicron processor design. In *Proc. ACM Design Automation Conf.*, 2004.

[5] V. Nookala, Y. Chen, D. Lilja, and S. Sapatnekar. Microarchitecture-Aware Floorplanning Using a Statistical Design of Experiments Approach. In *Proc. ACM Design Automation Conf.*, 2005.

[6] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proc. IEEE Int. Symp. on Computer Architecture*, pages 2–13, 2003.

[7] David Brooks and Margaret Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture*, page 171, Monterrey, Mexico, 2001. IEEE Computer Society.

[8] J. Cong, J. Wei, and Y. Zhang. A Thermal-Driven Floorplanning Algorithm for 3D ICs. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2004.

[9] J. Minz, E. Wong, and S. K. Lim. Thermal and Power Integrity-aware Floorplanning for 3D Circuits. In *Proc. IEEE Int. SOC Conf.*, 2005.

[10] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. IEEE Int. Symp. on Computer Architecture*, 2000.

[11] David Duarte, Vijaykrishnan, and Mary Jane Erwin. A clock power model to evaluate the impact of architectural and technology optimizations. *IEEE Transactions on VLSI Systems, Volume 10, Issue 6*, pages 844–855, December 2002.

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

[13] eCACTI. http://www.ics.uci.edu/ maheshmn/eCACTI/main.htm.

[14] J. C. Eble, V. K. De, D. S. Wills, and J. D. Meindl. A Generic System Simulator (GENESYS) for ASIC Technology and Architecture Beyond 2001. In *Int'l ASIC Conference*, 1996.

[15] L. He, W. Liao, and M. Stan. System Level Leakage Reduction Considering Leakage and Thermal Interdependency. In *Proc. ACM Design Automation Conf.*, 2004.

[16] P. Shivakumar and N. P. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model. Technical Report 2001.2, HP Western Research Labs, 2001.

[17] T. M. Austin. Simplescalar tool suite. http:/www.simplescalar.com.

[18] S. B. Horn. Vertically Integrated Sensor Arrays VISA. In *Defense and Security Symposium*, 2004.

[19] M. Umemoto, K. Tanida, Y. Nemoto, M. Hoshino, K. Kojima, Y. Shirai, and Kenji Takahashi. High-Performance Vertical Interconnection for high-density 3D Chip Stacking Package. In *IEEE Electronic Components and Technology Conf.*, 2004.

[20] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. Rectangle packing based module placement. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 472–479, 1995.

[21] P. Yuh, C. Yan, and Y. Chang;. Temporal floorplanning using the T-tree formulation. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2004.