Microarchitectural Floorplanning Under Performance and Thermal Tradeoff

Michael Healy, Mario Vittes, Mongkol Ekpanyapong, Chinnakrishnan Ballapuram,
Sung Kyu Lim, Hsien-Hsin S. Lee, and Gabriel H. Loh[†]
School of Electrical and Computer Engineering
[†]College of Computing
Georgia Institute of Technology

mbhealy@ece.gatech.edu

***Abstract*— In this paper, we present the first multi-objective microarchitectural floorplanning algorithm for designing high-performance, high-reliability processors in the early design phase. Our floorplanner takes a microarchitectural netlist and determines the placement of the functional modules while simultaneously optimizing for performance and thermal reliability. The traditional design objectives such as area and wirelength are also considered. Our multi-objective hybrid floorplanning approach combining Linear Programming and Simulated Annealing is shown to be fast and effective in obtaining high quality solutions. We evaluate the trade-off of performance, temperature, area, and wirelength and provide comprehensive experimental results.**

I. Introduction

Future processors implemented in nano-scale technologies will spend more time in communicating data operands or exchanging control information than actually performing useful computation. Meanwhile, the impact of power and thermal densities on these nano-scale devices and interconnects continue to increase, thereby raising the cost for cooling solutions, eroding performance gains, and threatening overall circuit reliability. Microarchitectural floorplanning has drawn significant interest from both the computer architecture and EDA communities recently [1], [2], [3], [4], [5]. The main motivation is to tackle the ever-worsening wire delay problem of high-performance processors [6], [7] 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 inter-module interconnects. Second, floorplanning significantly impacts the thermal and leakage profile. This is because the temperature of microarchitectural modules is not only 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 dependent on the temperature. Third, floorplanning impacts the dynamic power consumption of the buses and clock distribution network. The total number of flip-flops (FFs) inserted on global interconnects impacts the dynamic power consumed by the clock distribution network. However, the performance and temperature objectives conflict with each other since shorter distance among the active and hot modules improves the performance while exacerbating the thermal issue.

Recent studies have focused on traditional 2D microarchitectural floorplanning for performance optimization but not temperature concerns [1], [2], [3], [4], [5]. Several microarchitecture research works on temperature [8], [9], [10] and leakage power [11], [12], [13] provide runtime management of the functional modules but do not perform floorplanning. Most existing floorplanning works on a temperature objective [14], [15], [16] target circuit designs, not microarchitectural designs. Thus, the contribution of this paper is as follows:

- This work is the first to propose a multi-objective floorplanner for high-performance, high-reliability processors at the microarchitectural level. Our floorplanner simultaneously considers high performance, thermal reliability, area, and interconnect length and provides various trade-off points.
- Our microarchitectural thermal modeling considers the thermal and leakage inter-dependence for effective thermal runaway avoidance. Our microarchitectural power analysis, which is needed by our thermal analyzer, models the dynamic and leakage power consumed by both functional modules, global interconnect, and the clock distribution network for more accurate computation.

Our floorplanning optimizer consists of two steps: initial solution construction via Linear Programming and stochastic refinement via Simulated Annealing. This hybrid approach proves to be very effective in obtaining high quality solutions in short runtime.

II. Simulation Infrastructure

A. Microarchitectural Model

In order to model performance more faithfully for deep submicron processors, we isolate and model each wire as a separate *resource* which consumes power and has a delay in proportion to its length. Note that architectural simulators that ignore inter-module communication latencies will no longer be useful for evaluating high frequency processors designed with deep submicron technologies due to floorplan constraints and thermal concerns. 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 distance information provided by the floorplanner to determine the latency-related parameters such as pipeline depth and communication/forwarding latencies.

The microarchitectural configuration used in our study is summarized as follows: the machine width is 8. We use a 512-entry gshare branch predictor, a 512-entry reorder buffer, 8KB instruction and data L1 caches, a 128KB unified L2 cache, a 2MB unified L3 cache, 128-entry instruction and data TLBs, 8 ALUs, 4 FPUs, and a 128-entry load store queue. Note that our algorithm is general enough to take in many different configurations. For the sake of expediency, one configuration was chosen for experimentation.

B. Dynamic 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. Note that these detailed traffic activity and dynamic power profiles are only collected once at the very beginning of the entire design flow. The thermal analyzer then uses these power statistics to provide the thermal profile.

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. Since the new floorplan may lead to different interconnect lengths between modules, 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 large 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 grained level. Toward this, we use the accurate clock power model from [17]. This model considers clock distribution network power for memory structure precharge arrays, distribution wiring and drivers, pipeline flip-flops, and the phase locked loop.

C. Leakage Power Modeling

The leakage power is modeled in a separate process within our design flow. The model, which is based on [18], considers different bias conditions, though it only estimates subthreshold leakage power. For array-like structures, such as caches and TLBs, the number of bits (or SRAM cells) stored is multiplied by the amount of leakage current per bit and by the supply voltage to calculate the total leakage power for the structure. To calibrate our model, we also calculate the subthreshold leakage currents using the method in eCACTI [19]. Our model closely matches the leakage power estimated from eCACTI.

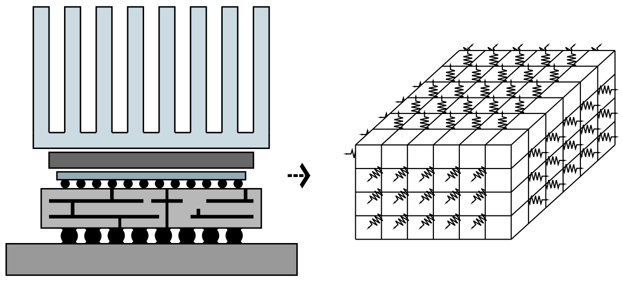

Fig. 1. 3D grid of a chip for thermal modeling

For logic structures, we assume that half the CMOS transistors are leaking at any given time. The number of transistors in these structures is estimated using the area values from GENESYS [20].

Due to this temperature dependence on the subthreshold leakage current, we first use our model to 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 due to their interdependence. We follow the criteria [21] for detecting the scenarios of thermal runaway: (i) the maximum module temperature T_{max} is increasing, (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 $(T_{max} - T_a)/R_t$, where T_a and R_t are ambient temperature and thermal resistance.

D. Thermal Modeling

The chip is divided into a 3D grid as shown in Figure 1 to apply a finite difference approximation to the thermal equation. We rewrite 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 among the modules according to their area. 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 is 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. Each thermal node models a rectangular prism of silicon that may dissipate power if it covers some portion of a block. The total power of each block is distributed according to and among the X-Y area of the nodes that block covers.

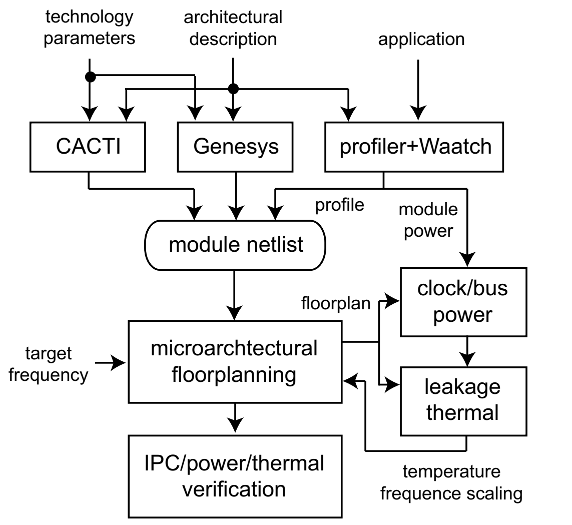

Fig. 2. Overview of our microarchitectural fborplanning framework.

III. Microarchitectural Floorplanning

A. Integrated Design Flow

Our design flow incorporates the power, leakage, performance, and temperature analysis discussed earlier into our floorplanner. An overview of this design flow is illustrated in Figure 2. First, we estimate the area and delay of the microarchitectural modules using analytical tools such as CACTI [22] and GENESYS [20]. Then we perform a cycle-level simulation using SimpleScalar [23] to collect and extract the amount of traffic between modules for each given benchmark. Wattch [24] was integrated into our framework to provide an estimation of the dynamic power during this simulation. We also integrate the clock power estimation from [17] and the leakage estimation from [18] as described above into our thermal analyzer. We then feed the module-level netlist, statistical interconnection traffic, and a target frequency to our thermal/profile-guided floorplanner.

Our floorplanner consists of two steps: initial solution construction via Linear Programming (LP) and stochastic refinement via Simulated Annealing (SA). We recursively bipartition the floorplan area until each module is contained in its own partition. Each bipartitioning solution is optimized by an LP-based approach, where performance and thermal objectives are simultaneously considered under the leakage power constraint. We then call our power/thermal/leakage analyzer upon each bipartitioning to update the thermal and leakage profile. The interdependence between leakage power and temperature creates the possibility of thermal runaway [13], in which the temperature and leakage are caught in a positive feedback loop and both blow up to infinity. If the floorplanner decides that thermal runaway is unavoidable given the current clock frequency then it scales the frequency down until it is successful in avoiding runaway. 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 thermal/leakage analyzer is again used to guide our optimization. When the final solution is obtained, we use SimpleScalar [23] and our power/thermal/leakage analyzer to evaluate the final solution for IPC, power, and temperature metrics.

B. LP-based Floorplanning

The basic idea behind our algorithm is to perform recursive bipartitioning until each partition contains a single module. After we choose a partition to be divided, we perform thermal/leakage analysis to get module temperature. We then use LP-based floorplanning to simultaneously optimize the performance and thermal distribution under the clock period, leakage, center of gravity constraints (to remove overlap among the modules), and boundary constraints (to keep the blocks within the chip boundary). An *iteration* in the LP phase of our algorithm combines a single bipartitioning and a subsequent LP-based floorplanning of all modules. Thus, we perform $k-1$ iterations if there are k modules in the netlist. Note that each iteration can be repeated multiple times to obtain different cutlines. This is because there exist multiple solutions that satisfy the boundary and center of gravity constraints during each bipartitioning. Thus, we perform each bipartitioning several times and pick the best solution in terms of performance and thermal profile. The following variables are used for our LP-based floorplanning formulation:

- N: set of all modules in the netlist.
- x_i, y_i: location of module i.
- w_i, h_i: half width and half height of module i
- a_i, g_i: area and delay of module i
- $w_m(i)$, $w_x(i)$: minimum/maximum width of module i
- $\lambda_{i,j}$: normalized profile weight on wire (i,j)
- $z_{i,j}$: number of flip-flops on wire (i,j) after insertion
- $X_{i,j} = |x_i - x_j|$ and $Y_{i,j} = |y_i - y_j|$
- $T_{i,j}$: normalized product of the temperature of i and j
- A: aspect ratio of the chip
- X_{max}, Y_{max}: maximum among all x_i/y_i values
- C: target cycle period
- d_r: repeated wire delay

Our LP floorplanner determines the values for the following decision variables: x_i, y_i, w_i, h_i, and z_{ij}. The following are the variables used for bipartitioning:

- $B(u)$: set of all modules at iteration u
- $M_j(u)$: set of all modules in partition j at iteration u
- $S_{j,k}(u)$: set of modules assigned to subpartition k ($k \in \{1,2\}$ for bipartitioning) in partition j at iteration u
- $(\bar{x}_{jk}, \bar{y}_{jk})$: center of subpartition k in partition j
- r_j,v_j,t_j,b_j: the right, left, top, and bottom boundaries of partition j

Our LP-based slicing floorplanning algorithm is formulated as follows:

Minimize:

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

Subject to:

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

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

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

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

$$w_m(i) \leq w_i \leq w_x(i),\ i \in N \tag{6}$$

$$x_i, y_i \geq 0,\ i \in N \tag{7}$$

$$X_{max} \geq x_i \text{ and } A \cdot X_{max} \geq y_i,\ i \in N \tag{8}$$

Boundary Constraints:

$$x_i + w_i \leq r_j,\ i \in M_j(u), j \in B(u) \tag{9}$$

$$x_i - w_i \geq v_j,\ i \in M_j(u), j \in B(u) \tag{10}$$

$$y_i + m_i w_i + k_i \leq t_j,\ i \in M_j(u), j \in B(u) \tag{11}$$

$$y_i - m_i w_i - k_i \geq b_j,\ i \in M_j(u), j \in B(u) \tag{12}$$

Center of Gravity Constrains: for $k \in \{1,2\}, j \in B(u)$

$$\sum_{i \in S_{jk}(u)} a_i x_i = \sum_{i \in S_{jk}(u)} a_i \times \bar{x}_{jk} \tag{13}$$

$$\sum_{i \in S_{jk}(u)} a_i y_i = \sum_{i \in S_{jk}(u)} a_i \times \bar{y}_{jk} \tag{14}$$

Our objective function shown in Equation (1) contains three terms: profile-weighted wirelength (= $\lambda_{ij} \cdot z_{ij}$), thermal-weighted wirelength (= $(1 - T_{ij})(X_{ij} + Y_{ij})$), and area (= X_{max}), where λ_{ij} is the profiled activity factor of the wire between modules i and j.[1] The minimization of the first term improves IPC while the minimization of the second term increases the distance between a pair of hot modules, thereby reducing thermal coupling. Since minimizing $X_{max} \cdot Y_{max}$ (= floorplan area) is non-linear, we only minimize X_{max} since the constraint (8) enforces $A \cdot X_{max}$ to be greater than all y values. Note that α, β, and γ are user defined parameters to weight the performance, thermal, and area objectives. In case $\alpha = 0$, our floorplanner optimizes performance+area only. In case $\beta = 0$, our floorplanner optimizes thermal+area objective only. Lastly, the conventional area/wirelength-driven floorplanner is using the following new objective function:

$$\gamma \cdot X_{max} + \delta \cdot \sum_{(i,j) \in E} (X_{ij} + Y_{ij}) \tag{15}$$

We provide an extensive comparison on these four different floorplanning objectives (simultaneous performance+thermal+area, performance+area, thermal+area, and simultaneous area+wirelength) in Section IV.

Constraint (2) is obtained from the definition of latency. If there is no FF on a wire (i,j), the delay of this wire is calculated as $d(i,j) = d_r(X_{ij} + Y_{ij})$. 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 block boundary constraints (9)–(12) require that all modules in the block be enclosed by these block boundaries. The center of gravity constraints (13)–(14) require that the area-weighted mean (= center of gravity) among all modules in each sub-block corresponds to the center of the sub-block.

[1]Since we add performance and thermal-related weights to the pure wirelength, we do not explicitly consider non-weighted pure wirelength objective. However, we report the wirelength metric in all of our experiments to show the impact of this multi-objective on wirelength.

C. *Stochastic Refinement*

The standard LP relaxation of the floorplanning problem introduces several non-optimalities. The recursive bipartitioning process also yields only slicing floorplans. In order to address these issues we implemented a simulated annealing based refinement engine for our floorplanner. This allows us to search around the local space and find a local minimum without being constrained by linearity. We derive a sequence pair from the LP floorplanning result and perform low temperature annealing with them. We use the *gridding* scheme described in [25] 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. Next we compute the initial annealing temperature by setting the probability of accepting bad moves to a low value. This reduces the runtime required for the annealing process significantly and focuses on results that are near the LP based result, which is assumed to be fairly close to optimal. We use the following cost function during our annealing:

$$cost = \alpha \cdot perf\_wire + \beta \cdot max\_temp + \gamma \cdot area$$

where $perf\_wire$ 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.

Our thermal analysis is the runtime bottleneck during our refinement since we need to perform the analysis for potentially many candidate solutions. Assuming that the thermal conductivity of functional modules are similar (they are mostly silicon), swapping the location of modules would not 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. Swapping two blocks usually has a small effect on the power profile, so ΔP should be sparse. This reduces the number of multiplications used by the second method at the expense of doing extra additions and subtractions. Lastly, the 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.

IV. Experimental Results

Our experiments were performed on 10 programs from the SPEC2000 benchmark suite. We chose 4 from the floating point and 6 from the integer benchmark suites. For IPC evaluation, we ran each benchmark on the average case

TABLE I

MULTI-OBJECTIVE FLOORPLANNING RESULTS WITH PERFORMANCE (P), MAXIMUM BLOCK TEMPERATURE (T), AREA (A), WIRELENGTH (W), AND RUNTIME REPORTED. THE LP+SA-BASED FLOORPLANNER IS USED.

	A+W		A+P		A+T		A+P+T	
bench	IPC	temp	IPC	temp	IPC	temp	IPC	temp
gzip	1.22	98.57	1.30	99.6	1.02	84.4	1.23	93
swim	1.12	68.08	1.18	67.1	0.99	63.9	1.12	65.7
vpr	1.03	80.85	1.11	80.9	0.84	72.6	1.04	77.2
art	1.01	107.53	1.08	109.8	0.84	90.5	1.01	101.1
mcf	2.00	70.14	2.28	69.4	1.74	65.4	2.11	67.6
equake	1.84	65.61	1.93	64.4	1.52	62.2	1.84	63.4
lucas	3.64	131.12	3.72	135.9	3.33	106.8	3.64	122.7
gap	1.36	74.68	1.45	74.8	1.14	68.7	1.38	72.1
bzip2	1.24	105.67	1.32	106.4	1.07	88.7	1.25	98.8
twolf	0.92	106.19	1.01	109.2	0.73	90.3	0.93	100.6
RATIO	1.00	1.00	1.06	1.01	0.86	0.87	1.01	0.95
AREA	153.38		194.96		201.22		206.84	
WIRE	402.5		617.6		781.2		502.87	
TIME	647		1245		1886		1348	

floorplan using a modified SimpleScalar 3.0 [23] by fast-forwarding 100 million instructions and simulating the next 100 million instructions. The reported temperature is simulated after all floorplanning steps and is adjusted relative to a $45°C$ ambient temperature. Wirelength is reported in mm and Area is reported in mm^2. The runtime of our 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 I presents a comparison of the IPC, temperature, area, wirelength, and runtime of 4 different objective functions. All data in this table is taken from the combined LP+SA approach. The area and wirelength only objective is used as the basis for the ratios in all the tables. One can see that the average temperature is increased slightly for the performance only objective and the maximum increases by $4°C$ over the baseline. The IPC of the performance only objective is the best among the experiments with an average IPC improvement over the baseline of 6%. The thermal only objective decreases the average temperature by about 14% over the performance only objective while maintaining reasonable IPC values and decreasing the maximum temperature by $29°C$ over the performance only objective. The hybrid performance and thermal objective decreases the temperature by 6% over the performance only objective while mainting high IPC value of 1% over the baseline and decreasing the maximum temperature by $14°C$.

A tradeoff between performance and temperature is shown in Figure 3. Temperature and IPC are reported as averages over the 10 benchmarks. The performance and area weights are held constant while the thermal weight is varied. As expected the graph shows that as the thermal weight is given more consideration by the floorplanner the performance drops. Ideally there would be some separation between the curves to indicate that high reduction in temperature could occur with

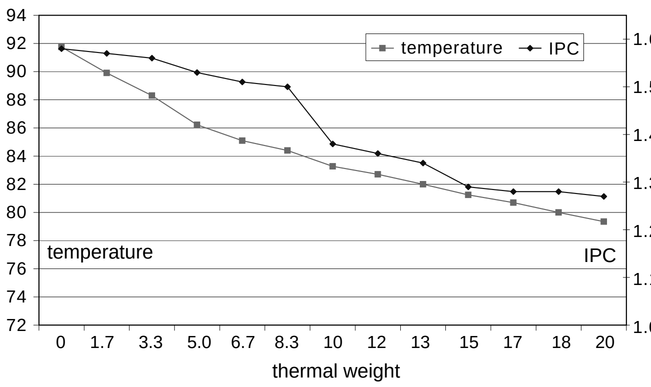

Fig. 3. Tradeoff between performance and temperature. Performance and area weights are held constant while thermal weight varies.

degradation in IPC value. The sweet spot of the curve appears to be around the middle point, as expected. The IPC drops sharply after this and so would be undesirable for the reduction in temperature achieved. One can observe that there is a 14% reduction in IPC and a 20% reduction in average maximum temperature between the performance-only objective (0) and the highest weight hybrid objective (20).

Experimental results were also gathered across the three floorplanning algorithms; linear programming only, simulated annealing, and the combined approach of linear programming followed by simulated annealing refinement. Table II presents a comparison of the IPC, temperature, area, wirelength, and runtime of these three floorplanning algorithms. One can observe from the table that the LP floorplanner does very poorly on the area of the floorplan and is not as good as the combined approach for temperature. 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 LP-only approach creates large area the wirelength values are still comparable. This is because while wirelength was an objective during the recursive bipartitioning phase of the LP the area is 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 the combined approach produces better solution quality.

A breakdown of the total power for each benchmark and the averages are provided in Table III. This demonstrates that our power profile simulators were in the range of other published works. This table also shows that our thermal objective was able to reduce the percentage of total power on average caused by leakage. Figure 4 shows a snapshot of our floorplanning solution.

V. CONCLUSIONS

In this paper, we presented the first multi-objective *microarchitecture-level* floorplanning algorithm for designing high-performance, high-reliability microprocessors. We simultaneously considered both performance and temperature objectives such that our automated floorplanner can provide a balanced or goal-directed processor organization that achieves both design objectives. Moreover, we integrated

TABLE II

COMPARISON AMONG PURE-SA, PURE-LP, AND LP+SA APPROACHES. THE OBJECTIVE USED IS A LINEAR COMBINATION OF PERFORMANCE, THERMAL, AND AREA ALL WITH EQUAL WEIGHT.

	pure SA		pure LP		LP+SA	
bench	IPC	temp	IPC	temp	IPC	temp
gzip	1.20	100.9	1.15	96.9	1.23	93
swim	1.11	67.3	3.62	66.3	1.12	65.7
vpr	0.99	81.5	0.56	79.1	1.04	77.2
art	0.96	111.5	0.22	105.4	1.01	101.1
mcf	1.81	69.5	1.07	68.5	2.11	67.6
equake	1.54	64.5	0.56	63.9	1.84	63.4
lucas	3.44	137.6	4.22	129.4	3.64	122.7
gap	1.31	74.7	0.55	73.4	1.38	72.1
bzip2	1.22	108.0	2.20	103.5	1.25	98.8
twolf	0.87	109.8	0.32	105.0	0.93	100.6
RATIO	0.94	1.02	0.95	0.98	1.01	0.95
AREA	197.84		664.94		206.84	
WIRE	508.07		535.80		502.87	
TIME	1453		1068		1452	

TABLE III

POWER BREAKDOWN FOR VARIOUS FLOORPLANNING APPROACHES. WE REPORT THE MODULE AND BUS (M+B), CLOCK (CLK), AND LEAKAGE (LEAK) POWER AS A PERCENTAGE OF TOTAL POWER, WHICH IS IN THE RANGE OF 40 TO 50 WATTS.

	performance			thermal			perf+thermal		
bench	m+b	clk	leak	m+b	clk	leak	m+b	clk	leak
gzip	60	20	20	62	19	19	61	21	18
swim	62	15	23	64	16	20	61	16	24
vpr	14	40	46	15	34	51	14	35	51
art	28	38	34	32	35	33	29	38	33
mcf	58	17	25	56	17	27	57	18	25
equake	70	11	19	71	11	18	70	12	18
lucas	19	49	32	23	44	33	19	44	37
gap	22	34	44	22	30	48	23	35	42
bzip2	58	24	18	59	21	20	54	22	24
twolf	39	35	26	40	30	30	38	31	31
AVG	49.5	24.6	25.9	50.9	22.2	26.7	48.9	23.5	27.5

leakage modeling into our thermal analyzer and monitored the temperature/leakage interaction to prevent thermal runaway. Our hybrid approach that combines Linear Programming and Simulated Annealing proved to be very effective in obtaining a high quality solution in short runtime.

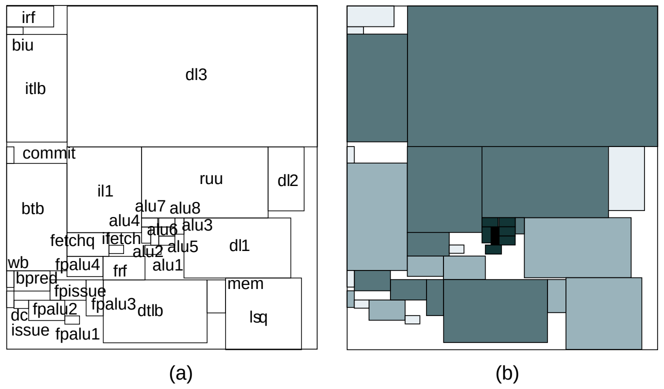

Fig. 4. (a) snapshot of our LP+SA floorplanning with hybrid (area, performance, thermal) objective, (b) temperature profile, where darker color denotes higher block temperature.

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. 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] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger, "Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures," in *Proc. IEEE Int. Conf. on Computer Architecture*, 2000.

[7] R. Ho, K. W. Mai, and M. A. Horowitz, "The Future of Wires," *Proceedings of the IEEE*, 2001.

[8] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proc. IEEE Int. Conf. on Computer Architecture*, 2003, pp. 2–13.

[9] M. Huang, J. Renau, S.-M. Yoo, and J. Torrellas, "A framework for dynamic energy efficiency and temperature management," in *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, Monterey, California, 2000, pp. 202–213.

[10] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture*. Monterrey, Mexico: IEEE Computer Society, 2001, p. 171.

[11] N. Kim, K. Flautner, D. Blaauw, and T. Mudge, "Drowsy instruction caches: leakage power reduction using dynamic voltage scaling and cache sub-bank prediction," in *Proc. Annual Int. Symp. Microarchitecture*, 2002.

[12] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: exploiting generational behavior to reduce cache leakage power," in *Proceedings of the 28th annual international symposium on Computer architecture*, Gteborg, Sweden, 2001, pp. 240–251.

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

[14] C. N. Chu and D. F. Wong, "A matrix synthesis approach to thermal placement," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 1998.

[15] W. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T. Theocharides, and M. Irwin, "Thermal-aware floorplanning using genetic algorithms," in *Proc. Int. Symp. on Quality Electronic Design*, 2005.

[16] 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.

[17] D. Duarte, Vijaykrishnan, and M. J. Erwin, "A clock power model to evaluate the impact of architectural and technology optimizations," *IEEE Transactions on VLSI Systems, Volume 10, Issue 6*, pp. 844–855, Dec. 2002.

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

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

[20] 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.

[21] W. Liao, F. Li, and L. He, "Microarchitecture level power and thermal simulation considering temperature," in *Proc. Int. Symp. on Low Power Electronics and Design*, 2003.

[22] P. Shivakumar and N. P. Jouppi, "CACTI 3.0: An Integrated Cache Timing, Power, and Area Model," HP Western Research Labs, Tech. Rep. 2001.2, 2001.

[23] T. M. Austin, "Simplescalar tool suite," http:/www.simplescalar.com.

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

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