Thermal-Aware Floorplanning for Task Migration Enabled Active Sub-threshold Leakage Reduction

Hushrav D Mogal and Kia Bazargan
Department of Electrical and Computer Engineering, University of Minnesota,
Minneapolis, MN 55455, {mhush, kia}@umn.edu

***Abstract*— This paper presents a new approach to active sub-threshold leakage reduction using task migration. The main idea is to replicate a *hot* module in a design so as to actively migrate its computation at regular intervals, reducing the on-chip temperature and thereby the sub-threshold leakage. We observe that choosing which blocks to migrate and their placement in a floorplan is a chicken-and-egg problem. To solve this, we propose a two step floorplanning methodology, wherein, given a base floorplan, we first choose the modules to replicate and then effectively utilize the deadspaces in it by exploiting the lateral conduction of heat in the floorplan to place a module's replica. With an optimized floorplan, using task migration we obtain an average savings of 29% in the active sub-threshold leakage at the expense of about 6% additional area.**

I. Introduction

In the sub-100nm era, sub-threshold leakage power becomes a larger fraction of the total power due to lower transistor threshold voltage and increased device density brought about by device scaling [15]. The increased power density creates thermal hotspots in the chip, which degrade performance and reduce the lifetime of the chip. More importantly, the sub-threshold leakage (hereby also referred to as leakage) is exacerbated due to its exponential dependence on temperature, which could result in the phenomenon of thermal runaway due to the positive feedback between power and temperature. Moreover, due to the finite lateral heat conduction of the silicon substrate, the spatial location of devices plays an important role in dictating the temperature and thereby the overall leakage of the design.

Digital circuits primarily operate either in the standby mode, wherein circuit blocks do not perform useful computation (for example memory caches in the idle state) or active mode, wherein circuits perform their intended computation (for example ALU units performing arithmetic operations). Designers leverage the standby mode of operation to reduce subthreshold leakage power [11], using schemes like input vector control, multi-threshold CMOS (MTCMOS), dynamic voltage scaling (DVS) and variable threshold-voltage CMOS (VTCMOS). These schemes are generally difficult to implement in the active mode due to various performance considerations. However, the large reduction in leakage power using schemes like MTCMOS warrant an investigation as to how these can be leveraged easily in the active mode.

Another technique orthogonal to the above mentioned transistor level schemes for sub-threshold leakage reduction is to exploit the lateral heat conduction of the silicon substrate to reduce the on-chip temperature profile. The authors in [12], [13] and [4] develop floorplanning tools to reduce the maximum on-chip temperature. Due to their passive nature, such techniques cannot adapt to the operating environment conditions.

This paper attempts to use task migration (TM) between replicated units of a design as a means to directly reduce temperature and therefore the leakage power without adversely affecting the performance. TM redistributes the active computation of certain high activity blocks in different parts of the chip at regular intervals. Since blocks are replicated, TM may imply an increase in the overall leakage. This is not the case however, if such a paradigm leverages the high reduction obtainable by means of a standby leakage reduction technique, applied to the inactive blocks in the design.

Fig. 1 illustrates the task migration paradigm in concept. Block set A consists of a subset of high activity blocks called the primary blocks replicated into a set B called the shadow or replica blocks in the design. At periodic intervals, denoted $t_1, t_2, \ldots, t_8$, computation is transfered from set A to set B and vice-versa. The length of the switching or migration interval is denoted τ. When block set A is active, block set B is put into standby mode, wherein its leakage is considerably reduced. Such a transfer of activity reduces the overall chip temperature and thereby the leakage. It is important to note that such a scheme can be applied over and above all other schemes that are currently in use to reduce leakage power.

The idea of task migration TM was first proposed in [5], and applied to a set of micro-architecture blocks. DVS was used to obtain better performance for the same maximum chip temperature constraint. TM can also be thought of as a Dynamic Thermal Management (DTM) technique, in which a chip is not allowed to reach a critical temperature threshold. Reactive DTM schemes like [3] and [14] generally require the use of thermal sensors and actuators to detect a thermal emergency and activate the appropriate thermal management scheme. On the other hand, TM can be used as a preventive DTM scheme, in which activity is migrated at regular intervals to prevent temperature build-up. Although easier to implement than reactive schemes, preventive schemes may incur an unnecessary performance overhead, swapping activity even when no thermal emergency is present. However, our main application is not a DTM scheme, but a method to reduce the active leakage of a chip, which is worsening with scaling trends.

This work was supported in part by the DARPA award FA-8650-04-C-7127

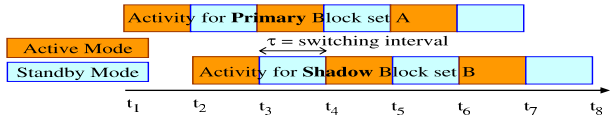


Fig. 1. Task migration illustrating the activity time-line of the primary, and shadow (or replica) set of blocks in a design.

We differ from [5] in many important aspects. First, the authors use a very simplistic thermal model, whereas we use a more detailed model to determine the temperature profile of the chip, with the given boundary conditions. Secondly, no consideration is given to the placement of the replica blocks in the floorplan. Given the dependence of the on-chip temperature profile to the spatial distribution of the blocks in the floorplan, we directly try to optimize the leakage consumption by a judicious placement of replica blocks using floorplanning. Moreover, we do not rely on the assumption that reducing the hotspot temperature would as a consequence reduce the overall leakage consumption. This is because large blocks like the cache may be very leaky despite having low temperature. Finally, adding replica blocks can potentially affect the overall performance of the floorplan, which we take into account. To the best of our knowledge, this is the first time such a problem has been explored in the floorplanning context.

II. Preliminaries

This section briefly describes the thermal and leakage modeling followed by an example motivating the benefits and issues in task migration (TM).

A. Thermal Model

We use the HotSpot [6] thermal modeling tool to provide a detailed thermal profile of the chip. HotSpot uses the duality between the heat diffusion in a system and current flow in an electrical RC network. It provides the user with two thermal models. A *block* thermal model, wherein each block is represented in terms of its equivalent thermal resistance and capacitace, and the network of resistors and capacitors is used to model heat conduction in the system. The formulation of this system is given by,

$$C\ \frac{dT}{dt} + G\ T \quad = \quad P \tag{1}$$

where G and C are the thermal conductance and capacitance matrices respectively, T is the temperature and P the power vector giving the temperature and power at each node (or block) in the thermal RC network respectively. The *grid* model in HotSpot, models the silicon substrate as a rectangular mesh for a finer granularity of modeling temperature. The number of nodes in Eq. 1 depends on the grid size. For a fine grid, this model is slower to evaluate than the *block* model for obvious reasons.

B. Leakage Model

The sub-threshold leakage power, $P_L(T)$, of a block at temperature T, with area proportional to A and supply voltage V_{dd} is given by [7],

$$P_L(T) \quad = \quad A\ T^2\ e^{-\left(\frac{\alpha\ V_{dd}+\beta}{T}\right)} \tag{2}$$

where α and β are empirical constants depending on whether the block is a functional unit such as an arithmetic unit or a memory unit such as a cache.

C. Motivating Example

Fig. 2 illustrates the use of task migration (TM) for the reduction of active leakage via floorplanning. The base floorplan consists of 7 blocks labeled $A - G$ and the main deadspace regions labeled $S_1 - S_3$.

Each floorplan is evaluated for total leakage by performing a simulation of Eq. 1 with $P = P_D + P_L(T)$, where P_D is the dynamic power and $P_L(T)$ is the leakage power of a block at temperature T given by Eq. 2, iterating until convergence is reached. The hotspot *grid* model is used with the substrate divided into a 50×50 mesh. The dynamic power is equally split between

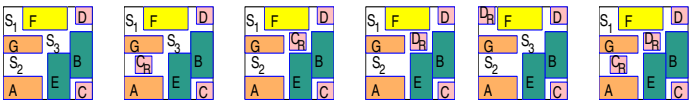


Fig. 2. An example motivating the need for TM via floorplanning. Fig. 2(a) shows design blocks A-G, and S_1-S_3 represent the main deadspace regions in the floorplan. Figs. 2(b) to 2(f) show different configurations of the primary and replica blocks (denoted with a '$_R$' suffix) for TM. Shaded regions indicate, pink for high power density combinational blocks (C, D), orange for low power density combinational blocks (A, G), yellow for high power density memory blocks (F) and green for low power density memory blocks (B, E).

TABLE I
LEAKAGE POWER EVALUATION FOR THE TM CONFIGURATIONS IN FIG. 2.

Config-uration	TM Block(s)	Normalized Leakage	% Leakage Savings in Blocks		
			Overall	TM	Non-TM
a	None	1.000	0.0	0.00	0.00
b	C	0.797	20.3	37.98	8.89
c	C	0.813	18.7	36.97	6.93
d	D	0.813	18.7	36.87	7.12
e	D	0.868	13.2	25.24	5.57
f	C, D	0.655	34.5	40.62	12.50

the primary and replica blocks assuming an equal migration interval. We ignore the difference in their dynamic power due to the different interconnect capacitances driven by them. We also ignore the switching penalty that arises due to switching between the primary set of blocks with its replica. This is reasonable because the average thermal time constant is orders of magnitude greater than the frequency of operation of the digital circuit. The different configurations in Figs. 2(b)-2(f) are used to illustrate various aspects of the TM technique, the results of which are tabulated in Table I.

The base floorplan is shown in Fig. 2(a) where no TM is performed. The high activity blocks C and D are surrounded by memory blocks B, E and F and impact their temperature (and therefore sub-threshold leakage) adversely. There are two main effects into play when considering TM. First, the power density of the block participating in TM (for example block C in Fig. 2(b)) reduces thereby reducing its temperature and leakage power. We call this the *primary TM effect*. Configuration b results in more than 20% overall leakage savings in large part due to the reduction in leakage of TM block C. The reduction of leakage in block C due to the primary TM effect (shown in column 5 in Table I) is almost 38%. Second, due to the lateral conduction of heat in the substrate, a block can affect the temperature and hence leakage of its neighboring modules. We call this the *secondary TM effect*. Configurations c and d shown in Figs. 2(c) and 2(d) respectively, result in less leakage reduction compared to configuration b. This is mainly because the replica module adversely affects the leakage of blocks B, E and F, resulting in a weaker secondary TM effect compared to configuration b. From column 6 of Table I, the reduction of leakage due to the secondary TM effect decreases from about 9% to 7% between configurations b and c. Configuration e results in the least leakage savings (when one TM block is used) due to reduced primary and secondary TM effects when the replica block is placed in the chip corner. Finally, when using two TM blocks, C and D, we get a larger leakage savings due to an increase of both the primary (with 41% savings) and secondary (with 13% savings) TM effects.

III. FLOORPLANNING FOR TASK MIGRATION

This section first describes the problem statement in the context of task migration (TM). This is followed by our approach to the thermal-aware floorplanning algorithm which reduces sub-threshold leakage via TM.

A. Problem Statement

The input is a primary set of n blocks, $\Phi_P = \{B_1, \ldots, B_n\}$, each with its associated area, A_i, aspect ratio bounds, AR_i^{min} and AR_i^{max}, average dynamic power consumption, P_{Di}, and leakage power profile, $P_{Li}(T)$, T being the temperature of the block. We compute a floorplan with Φ_P and a new set of blocks, $\Phi_R = \{B_{R1}, \ldots, B_{Rm}\}$, $m \leq n$, such that certain performance criteria are optimized or met. The replica set of m blocks, Φ_R, are a subset of Φ_P and take part in TM. The allowable area budget, ρ, represents the maximum ratio of the total area of the blocks in Φ_R to the blocks in Φ_P, and is given as an input to constrain the total area overhead due to TM. We aim to optimize the floorplan area, its overall leakage and the half-perimeter wirelength. In computing the leakage of a floorplan with TM, a migrating block B_i, and its replica B_{Ri}, are each assumed to have a dynamic power dissipation of $P_{Di}/2$.

We note an interesting chicken-and-egg dilemma for the above problem statement. To compute a TM-enabled floorplan we need both the primary set of blocks, Φ_P, and the replica set of blocks, Φ_R. However, choosing the TM block set, Φ_R, depends not only on the power density and leakage profile of the primary block set Φ_P, but also on their spatial distribution in the floorplan. This is because of the finite lateral conduction of heat in the silicon substrate wherein the temperature of a block (and therefore its leakage) is affected by that of its neighboring modules. Thus, a base floorplan and its associated temperature map are essential to compute the set Φ_R, thereby leading to a chicken and egg situation. The following sections describe in detail our method of floorplanning to choose a set of replica blocks Φ_R and compute the TM enabled floorplan.

Algorithm 1 Φ_R = TMCandidates ($\Phi_P, \mathcal{F}_\mathcal{P}, \rho$) // Φ_P = set of primary blocks in floorplan $\mathcal{F}_\mathcal{P}$; ρ = allowable area budget; Φ_R = set of candidate replica blocks partaking in TM

1: $\Phi_R = NULL$
2: Compute the steady state temperature profile of $\mathcal{F}_\mathcal{P}$
3: **for all** $B_i \in \Phi_P$ **do**
4: Compute C_i // see Eq. 6
5: **end for**
6: Sort Φ_P in non-increasing order of C_i
7: $A_o = \rho \cdot$ total area of all blocks in Φ_P // allowable total replica block area
8: $i = 0$
9: **while** $A_o > 0$ and $i < |\Phi_P|$ **do** // iterate through all blocks
10: **if** Area(B_i) $< A_o$ **then**
11: Insert B_i into Φ_R
12: $A_o = A_o -$ Area(B_i); $i = i + 1$
13: **end if**
14: **end while**
15: **return** Φ_R

B. TM Leakage Sensitivity

As noted above, to compute the set Φ_R, we need a base floorplan, $\mathcal{F}_\mathcal{P}$ and its associated thermal map. The steady state thermal map is computed using Eq. 1, by setting $dT/dt = 0$ giving $T = R \cdot P$, where $R = G^{-1}$ is the resistance thermal transfer matrix [16], obtained readily from the block thermal model of HotSpot. Entry $R(i, j)$ denotes the temperature increase at node i due to a unit power increase at node j.

Consider candidate block $B_i \in \Phi_P$ for TM. In choosing B_i, we need to consider its impact vis-a-vis the floorplan leakage. As described in Section II-C, we need to account for both primary and secondary TM effects of the migrating block B_i. Primary TM effects relate to the reduction of leakage of the block B_i, whereas secondary TM effects relate to the reduction of leakage of the rest of the blocks $B_j, \forall B_j \in \Phi_P, j \neq i$. The leakage sensitivity of the floorplan, $\mathcal{F}_\mathcal{P}$ to block B_i can therefore be written as

$$\begin{array}{rcccc} \frac{\Delta P_L}{\Delta P_{Di}} & = & \frac{\partial P_{Li}(T_i)}{\partial P_{Di}} & + & \sum_{j=1}^{n, j\neq i} \frac{\partial P_{Lj}(T_j)}{\partial P_{Di}} \\ \Rightarrow S_i & = & S_i^P & + & S_i^R \end{array} \quad (3)$$

where S_i gives us the change in the total leakage of the floorplan due to a change in the dynamic power of block B_i. The first term, S_i^P, gives us the change in leakage of B_i, with respect to a change in its dynamic power and relates to the primary TM effect. S_i^R computes the change in leakage of the other modules in the design with respect to a change in the dynamic power of block B_i and relates to the secondary TM effect of B_i. Given the steady state thermal profile, and the leakage profile of all the blocks in the floorplan, we can write the individual terms of Eq. 3 as

$$S_i^P = \frac{\partial P_{Li}(T_i)}{\partial T_i} \cdot \frac{\partial T_i}{\partial P_{Di}} = J_i(T_i) \cdot R(B_i, B_i) \quad (4)$$

where $J_i(T_i)$ is the jacobian of the leakage power with respect to temperature computed at T_i and $R(B_i, B_i)$ denotes the change in temperature of B_i due to a change in its dynamic power. Similarly, S_i^R can be written as,

$$S_i^R = \sum_{j=1}^{n, j\neq i} \frac{\partial P_{Lj}(T_j)}{\partial T_j} \cdot \frac{\partial T_j}{\partial P_{Di}} = \sum_{j=1}^{n, j\neq i} J_j(T_j) \cdot R(B_j, B_i) \quad (5)$$

where $J_j(T_j)$ denotes the jacobian of the leakage power of block B_j computed at temperature T_j and $R(B_j, B_i)$ denotes the change in the temperature of B_j due to a change in the dynamic power of block B_i.

To compute the overall impact of block B_i in TM mode, we need to weight its sensitivity with its total dynamic power. We compute, C_i, a measure of the reduction in leakage power due to a participating block B_i as,

$$C_i = S_i \cdot P_{Di} \quad (6)$$

Algorithm 1 computes the set of TM candidate replica blocks, Φ_R, given base floorplan, $\mathcal{F}_\mathcal{P}$ with the primary blocks, Φ_P. We first compute the steady state thermal profile of the floorplan $\mathcal{F}_\mathcal{P}$ followed the leakage reduction measure due to TM of each block, given by C_i in Eq. 6. The blocks in $B_i \in \Phi_P$ are then sorted in decreasing order of their C_i values so that the highest priority is given to the block with the largest C_i value. Step 7 computes the allowable total area of the replica modules, denoted A_o, from the area budget ρ. Finally, we greedily select the candidate TM blocks, Φ_R,

Algorithm 2 DS_R = DSSelection ($\mathcal{F}_\mathcal{P}, \Phi_R, \epsilon$) // $\mathcal{F}_\mathcal{P}$ = base floorplan; Φ_R = set of candidate replica blocks partaking in TM; ϵ = deadspace threshold; DS_R = subset of deadspaces in floorplan $\mathcal{F}_\mathcal{P}$ that are candidates for accommodating repplicas Φ_R

1: $DS_R = NULL$
2: DS_P = deadspaces of non-slicing floorplan $\mathcal{F}_\mathcal{P}$ using plane sweep
3: $A_{thresh} = \epsilon \cdot$ minimum area of block in Φ_R // compute deadspace threshold
4: **for all** $DS_i \in DS_P$ **do**
5: **if** Area(DS_i) $> A_{thresh}$ **then**
6: Insert DS_i into DS_R
7: **end if**
8: **end for**
9: **return** DS_R

in Steps 9-14, until we exhaust the available area overhead. This set of blocks is then inserted into $\mathcal{F}_\mathcal{P}$ to compute a new floorplan $\mathcal{F}_\mathcal{R}$, a procedure we call *floorplan patching*, described next.

Although the problem of selecting replica modules can be solved using dynamic programming (similar to the knapsack problem), we settled on a simple heuristic. After all, the gain C_i is only an approximation of the power savings due to the fact that the modules in the floorplan change location as a result of resizing after the replicas have been inserted.

C. TM-Aware Floorplan Patching

After selecting replica block set Φ_R as described above, we judiciously place them making use of the available deadspaces in the base floorplan, $\mathcal{F}_\mathcal{P}$. The primary objective in using the existing deadspaces in $\mathcal{F}_\mathcal{P}$ is to avoid a large increase in the floorplan area and to avoid large perturbations in the floorplan. This is in keeping with the chicken-and-egg dilemma described in Section III-A, since our selection of Φ_R was based on $\mathcal{F}_\mathcal{P}$. Moreover, as was seen in the example in Section II-C, it is important to place the replica blocks in the appropriate deadspace to optimize the leakage reduction. Therefore, the task of deadspace allocation is divided into three parts, selection of the deadspaces, matching the blocks in Φ_R with the selected deadspaces, and finally inserting the replica blocks in their alloted deadspaces.

1) Deadspace Selection: Algorithm 2 describes the procedure of selecting candidate deadspaces in the non-slicing base floorplan $\mathcal{F}_\mathcal{P}$. In Step 2 we first compute the deadspaces in $\mathcal{F}_\mathcal{P}$ by a plane sweep algorithm using a balanced interval tree [2]. Next, in Steps 4-8 with the candidate TM blocks in Φ_R, in order to avoid increasing the area and perturbing $\mathcal{F}_\mathcal{P}$ by a large amount, we choose those candidate deadspaces with area greater than a certain fraction, ϵ, of the block with minimum area. This is called the deadspace threshold. In our experiments ϵ was chosen as 0.75.

2) Deadspace Allocation Via Bipartite Matching: As seen from the example in Section II-C, for a single TM block, configuration b results in much better leakage savings than configuration e implying that it is important to correctly allocate the TM replica blocks to the candidate deadspace blocks. To prudently assign modules in Φ_R to deadspaces in DS_R, we need a measure of the fitness of allocating TM block $B_{Ri} \in \Phi_R$ to deadspace $DS_{Rj} \in DS_R$. A fitness measure similar to Eq. 6 described in Section III-B is computed for this purpose. We denote the fitness of placing block B_{Ri} at deadspace DS_{Rj} as C_i^j, given by Eq. 7 below,

$$C_i^j = S_i^j \cdot P_{Di}^j \tag{7}$$

where S_i^j is the leakage sensitivity and P_{Di}^j the dynamic power of the replica block B_{Ri} placed in deadspace D_{Rj}. As in Eq. 6, this is given by

$$S_i^j = \frac{\Delta P_L(T)}{\Delta P_{Di}^j} \tag{8}$$

where $P_L(T)$ is the floorplan leakage. As in Eqs. 4, 5 we compute S_i^j as

$$\begin{aligned} S_i^j &= S_i^{jP} + S_i^{jR} \\ &= J_i(T_j) \cdot R(DS_{Rj}, DS_{Rj}) + \sum_{k=1}^{n} J_k(T_k) \cdot R(B_k, DS_{Rj}) \end{aligned} \tag{9}$$

where $J_j(T_j)$ is the jacobian of the leakage of replica block B_{Ri} computed at T_j, the temperature of deadspace D_{Rj} and $R(B_k, DS_{Rj})$ is the increase in temperature of primary block B_k due to a unit increase in the dynamic power at DS_{Rj}. As before, S_i^{jP} and S_i^{jR} captures the primary and secondary TM effect of placing replica block B_{Ri} at deadspace DS_{Rj} respectively.

We next compute a bipartite graph $G(U, V, E)$ where U and V denote the two sides of the partition and E denotes the set of edges, an edge $e(u_i, v_j)$ between node $u_i \in U$ and $v_j \in V$. For every candidate replica block in $B_{Ri} \in \Phi_R$, we create a node u_i in U and for every candidate deadspace in $DS_{Rj} \in DS_R$ we create a node v_j in V. Edges are created between every $u_i \in U$ to $v_j \in V$ pair and the weight of edge $e(u_i, v_j)$, w_{ij}, is given by the fitness measure C_i^j of allocating a replica block B_{Ri} to deadspace D_{Rj} computed from Eq. 7. The problem of bipartite assignment is that of finding a perfect matching such that every node of the graph is incident on a matched edge. Because the problem of heat conduction in a system follows the superposition principle, the total cost of the assignment is a good indication of the leakage increase of the floorplan with the replica blocks brought into active mode. We use a *minimum weighted bipartite assignment* algorithm [9] to pair each candidate replica block with a single dead space such that the total leakage measure C_i^j of the assignment is minimized. The complexity of this algorithm is $O(n \cdot (m + n \log n))$ where n is the number of nodes and m is the number of edges in the bipartite graph. The overall procedure is shown in Algorithm 3

Algorithm 3 M = DSMatching ($\mathcal{F}_\mathcal{P}, \Phi_R, DS_R$) // $\mathcal{F}_\mathcal{P}$ = base floorplan; Φ_R = set of candidate replica blocks partaking in TM; DS_R = selected deadspaces in floorplan $\mathcal{F}_\mathcal{P}$; M = matching between blocks in Φ_R and DS_R

1: **for all** $B_{Ri} \in \Phi_R, DS_j \in DS_R$ **do**
2: Compute C_i^j // see Eq. 7
3: **end for**
4: Create bipartite graph $\mathcal{G}_\mathcal{B}(U, V, E)$ where node $u_i \in U$ represents $B_{Ri} \in \Phi_R$ and node $v_i \in V$ represents $DS_{Rj} \in DS_R$. Edge $E(u_i, v_j)$ has weight $w_{ij} = C_i^j$
5: M = Minimum-weighted Bipartite Matching of $\mathcal{G}_\mathcal{B}$
6: **return** M

Algorithm 4 $\mathcal{F}_\mathcal{R}$ = DSSubstitution ($\mathcal{F}_\mathcal{P}, \Phi_R, M$) // $\mathcal{F}_\mathcal{P}$ = base floorplan; Φ_R = set of candidate replica blocks partaking in TM; M = matching between blocks in Φ_R and floorplan deadspaces; $\mathcal{F}_\mathcal{R}$ = patched floorplan with TM blocks

1: Compute horizontal (HCG) and vertical constraint graphs (VCG), of base floorplan $\mathcal{F}_\mathcal{P}$ // finds relative ordering
2: **for all** $B_{Ri} \in \Phi_R$ **do**
3: Find deadspace D_{Rj} which matches B_{Ri} using M // see Algorithm 3
4: Orient B_{Ri} to conform with the dimensions of deadspace D_{Rj}
5: **end for**
6: **if** there exists a non-conforming block B_{Ri} inserted in D_{Rj} **then**
7: Compute patched floorplan $\mathcal{F}_\mathcal{R}$ from HCG and VCG
8: **end if**
9: **return** $\mathcal{F}_\mathcal{R}$

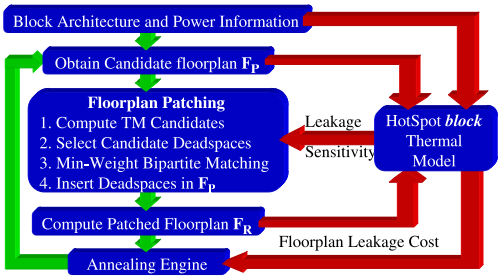


Fig. 3. Overall flow of our floorplanning approach.

Note the use of a min-cost objective in deciding the assignment of TM replica blocks to deadspaces as opposed to their selection in Algorithm 1. This is because in the former a replica block is optimized when turning on from idle to active mode whereas in the latter a primary block is optimized when turning off from active to idle mode. This results in a bias in the optimization of the primary block in switching from its active to idle mode and is owing the lack of prior knowledge of the blocks participating in TM.

3) Deadspace Substitution: Algorithm 4 shows our method of inserting the candidate TM blocks, Φ_R into their respective matched deadspaces (given by matching M), to compute a patched floorplan $\mathcal{F}_\mathcal{R}$. Step 1 first computes the horizontal and vertical constraint graphs of the base floorplan $\mathcal{F}_\mathcal{P}$ so that the relative ordering of the blocks is obtained. Next, Steps 2-5 replace the width and height of the matched deadspaces with those of the candidate TM blocks in Φ_R, thereby patching $\mathcal{F}_\mathcal{P}$. When inserting the block, we orient it such that its dimensions conform to that of the deadspace. However, this step could result in an invalid floorplan with overlaps. In such a case (Steps 6-8), we recompute the block coordinates from the horizontal and vertical constraint graphs and return the newly patched floorplan $\mathcal{F}_\mathcal{R}$.

Application of the floorplan patching procedure to the toy example in Section II-C gives us the floorplan in Fig. 2(b) with an area budget $\rho = 10\%$ and Fig. 2(f) with $\rho = 15\%$, illustrating the efficacy of our algorithm.

D. Bringing it Together: Floorplanning

We can now describe our floorplanning algorithm as shown in Fig. 3. We use the Parquet floorplanning tool [1], which is a fixed die non-slicing floorplanner based on the simulated annealing combinatorial optimization algorithm and can handle multiple constraints like area, aspect ratio and wirelength. At every step, a candidate base floorplan $\mathcal{F}_\mathcal{P}$ is evaluated for TM by first computing the TM candidates, Φ_R, using Algorithm 1. These are then used to select the candidate deadspaces $DS_R \subset \mathcal{F}_\mathcal{P}$ in which a TM block can be inserted (Algorithm 2). We perform a min-cost bipartite

TABLE II

THE SOC BENCHMARKS. N_C, N_M AND N_T DENOTE THE COUNT OF COMBINATIONAL, MEMORY AND TOTAL BLOCKS IN THE DESIGN. P_{den} DENOTES THE AVERAGE POWER DENSITY OF A MODULE IN THE DESIGN.

Benchmark Name	Number of Modules			Avg. P_{den} (W/m^2)
	N_T	N_C	N_M	
$d695$	10	6	4	$3.14e^5$
$h953$	8	3	5	$2.8e^5$
$g1023$	14	6	8	$2.21e^5$
$p34932$	19	4	15	$2.23e^5$

assignment of TM blocks to candidate deadspaces with Algorithm 3 and use Algorithm 4 to obtain a TM-aware floorplan $\mathcal{F}_{\mathcal{R}}$. This floorplan is used to compute the cost term during annealing as follows,

$$C = \alpha \cdot A + \beta \cdot HPWL_w + \gamma \cdot L_T + (1 - \alpha - \beta) \cdot AR \quad (10)$$

where A is the area, $HPWL_w$ is the weighted half-perimeter wirelength, L_T is the temperature dependent leakage cost and AR is the aspect ratio of the TM-aware floorplan $\mathcal{F}_{\mathcal{R}}$. Factors α, β and γ weight the individual cost terms. Parquet in fixed-outline floorplanning mode lets the user to specify a bound on the total deadspace. We replicate nets which contain TM blocks, adding their contribution to $HPWL$ when performing TM-aware floorplanning.

To avoid time consuming iterations for the convergence of the leakage power of a floorplan, we use a leakage cost formulation similar to [10]:

$$L_T = \frac{\Delta P_L(T)}{\Delta t} = J(T_s) \cdot C^{-1} \cdot P_L(T_s)\Big|_{T_s = G^{-1} \cdot P_D} \quad (11)$$

where G and C are the thermal conductance and capacitance matrices respectively, T_s is the steady state temperature distribution vector of the blocks when dissipating their dynamic power P_D, $P_L(T_s)$ (see Eq. 2) is the vector of block leakage powers computed at temperature T_s and $J(T_s)$ is the jacobian of block leakage powers with respect to temperature evaluated at T_s. The basic idea is to capture the transient effect by computing the temperature dependent increase in the leakage power over time Δt given the average dynamic power dissipation P_D.

IV. EXPERIMENTAL SETUP AND RESULTS

This section describes the benchmarks evaluated in our work, along with the experimental setup and simulation results. All experiments were performed on a Linux Intel Pentium 3.2 GHz processor with 2GB RAM, in the $100nm$ technology node.

A. SOC Benchmarks

To evaluate our task migration (TM) aware floorplanning approach we chose the SOC test benchmarks [8]. The benchmarks designs are expressed in a tree-based hierarchy format. For our floorplanning, we chose those modules that are in the bottom most level of the tee. Every internal node of the design tree was considered a net with its terminals as the modules in its bottom most leaf nodes. The weight of the net was assigned a value proportional to the height of the internal node in the tree. Architectural information was extracted by using the given number of module terminals and applying a Rent's rule based formulation with a suitable Rent's coefficient to obtain the number of logic gates in a module. This, in combination with the design technology node ($100nm$ for our work), gives us the block area information. It is important to distinguish between combinational and memory type modules mainly due to their different power densities and leakage sensitivities to temperature. Due to the lack of dynamic power information, the power density was chosen randomly between the range $\{5.0e^4, 1.0e^6\}$ for combinational blocks and $\{1.0e^4, 4.0e^4\}$ for memory blocks. Table II shows various parameters of interest for the benchmarks used in our experiments.

B. Results

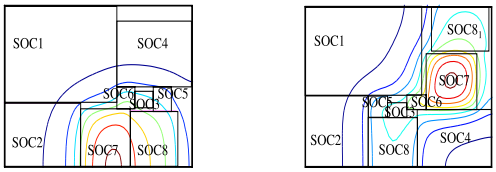
Although our floorplanning framework uses the HotSpot *block* model for thermal modeling, when evaluating an optimized floorplan, we use the *grid* model with a mesh of 50×50, to get better accuracy. For each benchmark, we run the floorplanner 5 times, first without any task migration (TM) of modules, and second in the TM-aware mode with an area budget $\rho = 10\%$, and choose the best floorplan. Recall that ρ is the factor used in determining the number of candidate modules chosen for TM using Algorithm 1.

Table III shows the savings in leakage obtained by our TM-aware floorplanner. On average, we obtain leakage savings of about 29% with a 5.7% increase in area. An important benefit of TM is to lower the maximum chip temperature by greater than 20°. Moreover, the area penalty is in consonance with ρ indicating that our floorplanner is effectively able to assign deadspaces to the TM blocks. Also note that there is no correspondence between the number of TM replica modules and the leakage power savings, and relates to the interdependency between the allowable deadspace and the chosen TM replica blocks. Due to a small number of nets in each benchmark evaluated, the $HPWL$ term is primarily a reflection of the chip dimensions as a result of which the degradation is proportional to the square-root of the increase in area (which cannot be avoided in TM-aware floorplanning).

TABLE III

PERFORMANCE RESULTS FOR TM-AWARE FLOORPLANNING WITH RESPECT TO THE FLOORPLAN AREA, LEAKAGE POWER, MAXIMUM CHIP TEMPERATURE AND HALF-PERIMETER WIRELENGTH. N_{TM} DENOTES THE NUMBER OF REPLICA MODULES FOR THE TM-AWARE FLOORPLAN.

Bench-mark	Area Increase	N_{TM}	Leakage Savings	Max. Temperature °C		$HPWL$ ratio
				No TM	TM	
$d695$	9.40%	3	18.11%	142.4	116.6	1.25
$h953$	7.06%	1	24.13%	141.8	115.9	1.11
$g1023$	0.68%	3	39.47%	139.2	98.9	1.09
$p34932$	5.31%	3	34.98%	152.2	91.9	1.47



(a)No TM (b)TM-aware

Fig. 4. Chip temperature contours for floorplans without and with TM. The replica module is labeled $SOC8_1$.

Fig. 4 shows the contour plots of the chip temperature distribution of the benchmark $h953$ for floorplans obtained without and with TM-aware floorplanning. In spite of block $SOC7$ being the module with the greatest leakage power and the hotspot in the design, it cannot be repliacted due to the limited area budget. Moreover, right neighboring high power density block $SOC8$ restricts the spread of heat from $SOC7$ in the horizontal direction. By migrating activity from $SOC8$, we take advantage of the secondary effect of TM to allow $SOC7$ to better dissipate its heat laterally, albeit at a slightly higher leakage for $SOC8$ due to a reduced primary TM effect. Our floorplanning algorithm effectively trades off the primary and secondary TM effects of a block in both its selection and placement.

REFERENCES

[1] S. Adya and I. Markov, "Fixed-outline floorplanning: enabling hierarchical design," *IEEE TVLSI*, vol. 11, no. 6, pp. 1120–1135, Dec 2003.

[2] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 1989.

[3] M. Gomaa, M. D. Powell, and T. N. Vijaykumar, "Heat-and-run: leveraging smt and cmp to manage power density through the operating system," in *ASPLOS-XI*. New York, NY, USA: ACM Press, 2004, pp. 260–270.

[4] Y. Han, I. Koren, and C. A. Moritz, "Temperature aware floorplanning," in *Second Workshop on Temperature-Aware Computer Systems(TACS-2)*, 2005.

[5] S. Heo, K. Barr, and K. Asanovic;, "Reducing power density through activity migration," in *ISLPED '03*. New York, NY, USA: ACM Press, 2003, pp. 217–222.

[6] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotspot: a compact thermal modeling methodology for early-stage vlsi design," *IEEE TVLSI*, vol. 14, no. 5, pp. 501–513, May 2006.

[7] W. Liao, L. He, Lepak, and K.M., "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE TCAD*, vol. 24, no. 7, pp. 1042–1053, July 2005.

[8] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A set of benchmarks for modular testing of SOCs," in *ITC '02*, Oct. 7–10, 2002, pp. 519–528.

[9] K. Melhorn and S. Naher, *Leda: A platform for combinatorial and geometric computing*. New York, NY: Cambridge University Press, 1999.

[10] H. D. Mogal and K. Bazargan, "Microarchitecture floorplanning for sub-threshold leakage reduction," in *DATE '07*. San Jose, CA, USA: EDA Consortium, 2007, pp. 1238–1243.

[11] S. G. Narendra and A. Chandrakasan, *Leakage in Nanometer CMOS Technologies*. Newyork, NY: Springer US, 2005.

[12] V. Nookala, D. J. Lilja, and S. S. Sapatnekar, "Temperature-aware floorplanning of microarchitecture blocks with ipc-power dependence modeling and transient analysis," in *ISLPED '06*. New York, NY, USA: ACM Press, 2006, pp. 298–303.

[13] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron, "A case for thermal-aware floorplanning at the microarchitectural level," *JILP*, vol. 7, Oct 2005. [Online]. Available: http://www.jilp.org/vol7/

[14] A. Shayesteh, E. Kursun, T. Sherwood, S. Sair, and G. Reinman, "Reducing the latency and area cost of core swapping through shared helper engines," in *ICCD '05*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 17–23.

[15] SIA. International technology roadmap for semiconductors. [Online]. Available: http://www.itrs.net

[16] C.-H. Tsai and S.-M. S. Kang, "Standard cell placement for even on-chip thermal distribution," in *ISPD '99*. New York, NY, USA: ACM Press, 1999, pp. 179–184.