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

Hushrov 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 subthreshold 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, first we 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 20% 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 subthreshold 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. Additionally, circuits primarily operate either in the active 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 active 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 (VTACMOS). 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) replicated units of a design as a means to directly reduce temperature and thereby the leakage. Leakage is an adversarial parameter that is adversely affecting the performance. TM redistributes the active computation of certain high active 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, ..., t_n, computation is transferred 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. As its leakage is considerably reduced, such a transferability 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 leakage 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. 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.

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 capacitance, and the network of resistors and capacitors is used to model heat conduction in the system. The formulation of this system is given by

\[ C \frac{dT}{dt} + G T = P \]

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],

\[ P_L(T) = A T^2 e^{-\frac{\alpha V_{dd}}{\beta}} \]

where \( \alpha \) and \( \beta \) 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 \( S_1 - S_4 \). Each block 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 x 50 mesh. The dynamic power is equally split between

This work was supported in part by the DARPA award FA-8650-04-C-7127
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 \textit{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 \textit{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 Table I, the reduction in leakage of \(B\) due to 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 \(C\) is used) due to reduced primary and secondary TM effects when the replica block \(C\) is placed in TM and its neighboring blocks.

Finally, when using two TM blocks, the floorplan area, its overall leakage and performance criteria are optimized or met. The replica set of \(A\) and \(B\) represents the maximum ratio of the total area of the blocks in the floorplan, \(\Phi_{SA}\) and \(\Phi_{SB}\) respectively.


table I

<table>
<thead>
<tr>
<th>Configuration</th>
<th>TM Blocks(s)</th>
<th>Normalized Leakage</th>
<th>% Leakage Savings in Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>None</td>
<td>1.000</td>
<td>0.00</td>
</tr>
<tr>
<td>(b)</td>
<td>C</td>
<td>0.797</td>
<td>20.3</td>
</tr>
<tr>
<td>(c)</td>
<td>C, D</td>
<td>0.813</td>
<td>18.7</td>
</tr>
<tr>
<td>(d)</td>
<td>C</td>
<td>0.813</td>
<td>18.7</td>
</tr>
<tr>
<td>(e)</td>
<td>D</td>
<td>0.868</td>
<td>13.2</td>
</tr>
<tr>
<td>(f)</td>
<td>C, D</td>
<td>0.655</td>
<td>34.5</td>
</tr>
</tbody>
</table>

B. TM Leakage Sensitivity

As noted above, to compute the set \(\Phi_{R}\), we need a base floorplan, \(F_{P}\) and its associated thermal map. The steady state thermal map is computed using Eq. 1, by setting \(dT/dt = 0\) giving \(T = R - P\), where \(R = C^{\text{TM}}\) is the resistance thermal transfer matrix [16], obtained readily from the floorplan thermal model of HotSpot. Entry \(R_{ij}\) 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 \(B_j \in \Phi_{P}; j \neq i\). The leakage sensitivity of the floorplan, \(S_{P}\), to block \(B_i\) can therefore be written as

\[
S_{P} = \frac{\partial P_{L}}{\partial T_{B_i}} + \sum_{j=1}^{n} \frac{\partial P_{L}(T_j)}{\partial T_{B_i}} S_{T_j}
\]

where \(S_{T_j}\) computes the change in leakage of the other modules in the design with respect to a change in the dynamic power of block \(B_j\) and \(S_{P}\) computes the change in leakage of all the blocks in the floorplan.

III. FLOORPLANNING FOR TASK MIGRATION

This section 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, \(AR_{\text{lim}}\) and \(AR_{\text{max}}\), average dynamic power consumption, \(P_{D}\), and leakage power profile, \(P_{L}(T)\). \(T\) being the temperature of the block. We compute a floorplan with \(\Phi_{P}\) and a set of blocks, \(\Phi_{R}\) which are a subset of \(\Phi_{P}\) and take part in TM. The allowable area budget, \(\rho\), represents the maximum ratio of the total area of the blocks in \(\Phi_{R}\) to the blocks in \(\Phi_{P}\), 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, the leakage of the migrating block \(B_i\) and its replica \(B_{R_{i}}\) are assumed to have a dynamic power dissipation of \(P_{D}/2\).

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, \(\Phi_{P}\), and the replica set of blocks, \(\Phi_{R}\). However, choosing the TM block set, \(\Phi_{R}\), depends not only on the power density and leakage profile of the primary block set \(\Phi_{P}\), 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 \(\Phi_{R}\) 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 \(\Phi_{R}\) and compute the TM enabled floorplan.

Algorithm 1 computes the set of TM candidate replica blocks, \(\Phi_{R}\), given base floorplan, \(F_{P}\) with the primary blocks, \(\Phi_{P}\). We first compute the steady state thermal profile of the floorplan \(F_{P}\) followed the leakage reduction measure due to TM of each block, given by \(C_i\) in Eq. 6. The blocks 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 \(\rho_{max}\), from the area budget \(\rho\). Finally, we greedily select the candidate TM blocks, \(\Phi_{R}\).
finding a perfect matching such that every node of the graph is incident
deadspace threshold. In our experiments is computed for this purpose. We denote the fitness of placing block
to correctly allocate the TM replica blocks to the candidate deadspace blocks.
A large increase in the floorplan area and to avoid large perturbations in the
large amount, we choose those candidate deadspaces with area greater than
we first compute the deadspaces in Φ
we can now describe our floorplanning algorithm as shown in Fig. 3.
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, ΦR, into their respective matched deadspaces (given
by matching M), to compute a patched floorplan FR. Step 1 first computes the
horizontal and vertical constraint graphs of the base floorplan FR 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 ΦR, thereby patching FR. 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 FR.
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 ρ = 10% and
Fig. 2(f) with ρ = 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 aspect ratio and
width. At each step, a candidate base floorplan FR is evaluated for
TM by first computing the TM candidates, ΦR, using Algorithm 1. These
are then used to select the candidate deadspaces DSF ⊂ FR in which a
TM block can be inserted (Algorithm 2). We perform a min-cost bipartite

Algorithm 1

```
1: } Algorithm 1
2: DSS = DSSelection (FR, ΦR, ϵ) // FR = base floorplan; ΦR = set
3: of candidate replica blocks partaking in TM; ϵ = deadspace threshold; DSS = subset
4: of deadspaces in floorplan FR that are candidates for accommodating replicas ΦR
5: return DSS
```

Algorithm 2

```
DSR = NULL
1: DS = set of deadspace of non-slicing floorplan FR by plane sweep
2: DSR = deadspaces of non-slicing floorplan FR using plane sweep
3: Amin = ϵ - minimum area of block in ΦR if compute deadspace threshold
4: for all DS, DS ∈ DSR
5: if Area(DS) > Amin, then
6: Insert DS into DSR
7: end if
8: end for
9: return DSR
```

Algorithm 3

```
M = DSSelection (FR, ΦR, DSS) // FR = base floorplan; ΦR = set
1: of candidate replica blocks partaking in TM; DSS = selected deadspaces in floorplan FR;
M = matching between blocks in ΦR and DSS
2: Compute horizontal (HCG) and vertical constraint graphs (VCG), of base floorplan FR
3: Orient FR, to conform with the dimensions of deadspace DSR
4: for all BR ∈ M // finds relative ordering
5: Find deadspace DSR, which matches BR, using M // see Algorithm 3
6: Orient DSR, to conform with the dimensions of deadspace DSR
7: Compute patched floorplan FR from HCG and VCG
8: end if
9: return FR
```

Algorithm 4

```
FR = DSSubstitution (FR, ΦR, M) // FR = base floorplan; ΦR = set
1: of candidate replica blocks partaking in TM; M = matching between blocks in ΦR and
2: floorplan deadspaces; FR = patched floorplan with TM blocks
3: Compute horizontal (HCG) and vertical constraint graphs (VCG), of base floorplan FR
4: in Step 2 we first compute the deadspaces in FR by a plane sweep algorithm using a
5: balanced interval tree [2]. Next, in Steps 4-8 with the candidate TM blocks in ΦR, in order to avoid increasing the area and perturbing FR by
6: a large amount, we choose those candidate deadspaces with area greater than a
7: certain fraction of the block with minimum area. This is called the deadspace threshold. In our experiments ϵ was chosen as 0.75.
8: for all BR ∈ M // finds relative ordering
9: Find deadspace DSR, which matches BR, using M // see Algorithm 3
10: Orient DSR, to conform with the dimensions of deadspace DSR
11: Compute patched floorplan FR from HCG and VCG
12: end if
13: return FR
```

C. TM-Aware Floorplan Patching
After selecting replica block set ΦR as described above, we judiciously place
them taking use of the available deadspaces in the base floorplan FR. The
primary objective in using the existing deadspaces in FR is to avoid a
large increase in the floorplanning 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 deadspaces was based on FR. 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 ΦR with the selected deadspaces, and
finally inserting the replica blocks in their allotted deadspaces.
1) Deadspace Selection: Algorithm 2 describes the procedure of select-
candidate deadspaces in the non-slicing base floorplan FR. In Step 2 we
first compute the deadspaces in FR by a plane sweep algorithm using a
balanced interval tree [2]. Next, in Steps 4-8 with the candidate TM
blocks in ΦR, in order to avoid increasing the area and perturbing FR by
a large amount, we choose those candidate deadspaces with area greater than a
certain fraction 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 c implying that it is important
correctly allocate the TM replica blocks to the candidate deadspaces blocks.
To prudently assign modules in ΦR to deadspaces in DSR, we need a
measure of the fitness of allocating TM block BR ∈ ΦR to deadspace
deadspace DSR. A fitness measure similar to Eq. 6 described in Section III-B
is computed for this purpose. We denote the fitness of placing block BR at
deadspace DSR as CIR, given by Eq. 7 below,
CIR = SIR · PIR
(7)
where SIR is the leakage sensitivity and PIR the dynamic power of the replica
block BR placed in deadspace DSR. As in Eq. 6, this is given by
SIR = DP(1) − DP(0) (8)
where DP(1) is the floorplanning leakage. As in Eqs. 4, 5 we compute SIR as
SIR = SP + SR,
SP = Jij(Ti) · R(DFS) · R(BR) (9)
where Jij(Ti) is the Jacobian of the leakage of replica block BR, computed
at Ti, the temperature of deadspace DFS and R(BR, DFS) is the increase
in temperature of primary block BR due to a unit increase in the dynamic
power of DFS. As before, SIR and SR captures the primary and secondary
TM effect of placing replica block BR at deadspace DFS 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(uvi, vj)
between node uvi ∈ U and vj ∈ V. For every candidate replica block in
BR ∈ ΦR, we create a node uvi in U and for every candidate deadspace
DFS ∈ DSR we create a node vj in V. Edges are created between every
oui ∈ U to vj ∈ V pair and the weight of edge e(uvi, vj), wj, is given by the
fitness measure CIR of allocating a replica block BR to deadspace
BR, 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 CIR of the assignment is minimized.
The complexity of this algorithm is O(n · (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.
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
to turn 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.
### TABLE II

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>N_T</th>
<th>N_C</th>
<th>N_M</th>
<th>Avg. ( P_{D,\alpha} ) (W/m²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d095 )</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>3.14e²</td>
</tr>
<tr>
<td>( f905 )</td>
<td>12</td>
<td>8</td>
<td>6</td>
<td>2.6e²</td>
</tr>
<tr>
<td>( p91024 )</td>
<td>14</td>
<td>6</td>
<td>8</td>
<td>2.21e²</td>
</tr>
<tr>
<td>( p94932 )</td>
<td>19</td>
<td>4</td>
<td>15</td>
<td>2.23e²</td>
</tr>
</tbody>
</table>

### TABLE III

<table>
<thead>
<tr>
<th>Bench-mark</th>
<th>Area Increase</th>
<th>( N_T )</th>
<th>Leakage Savings</th>
<th>Max. Temperature (°C)</th>
<th>( HPWL ) ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d095 )</td>
<td>9.40%</td>
<td>4</td>
<td>18.11%</td>
<td>142.4</td>
<td>116.6</td>
</tr>
<tr>
<td>( f905 )</td>
<td>7.06%</td>
<td>3</td>
<td>24.13%</td>
<td>141.8</td>
<td>115.9</td>
</tr>
<tr>
<td>( y1023 )</td>
<td>6.68%</td>
<td>3</td>
<td>39.47%</td>
<td>139.2</td>
<td>98.9</td>
</tr>
<tr>
<td>( p94932 )</td>
<td>5.31%</td>
<td>3</td>
<td>34.98%</td>
<td>152.2</td>
<td>91.9</td>
</tr>
</tbody>
</table>

### Fig. 4

Chip temperature contours for floorplans without and with TM. The replica module is labeled \( SOC_8 \).

### 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 tree. 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 used by giving the 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 \((5.0e^{-5},1.0e^{-6})\) for combinational blocks and \((1.0e^{-5},4.0e^{-6})\) 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 \(50 \times 50\), 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 \(\rho\) 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 in consonance with \(\rho\) 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

### REFERENCES